bCatalog

PFC – J2EE

Anna Casacuberta Puig

Enginyeria Informàtica

Semestre Febrer 2017

Índex

Descripció del projecte

El projecte final de carrera consistirà en l'anàlisi d'eines de persistència per a poder crear un framework per aïllar la capa de dades i poder-ne abstreure la seva complexitat.

Mitjançant JEE per a aplicacions distribuïdes, hi ha diverses alternatives per implementar-les i es realitzarà un anàlisi comparatiu per a poder escollir quina serà la millor opció. També s'estudiaran diferents tipus de persistència de dades indicant punts en comú, avantatges i inconvenients, rendiment,...

Un cop escollides les opcions s'implementarà el framework per dotar de les funcionalitats bàsiques de la capa de persistència que juntament amb una petita aplicació per a gestionar un catàleg de llibres en demostrarà el seu funcionament.

L'aplicació ha de permetre gestionar una col·lecció de llibres, poder marcar i valorar els llegits, indicar la localització: si es troba guardat en un magatzem, a casa els pares, s'ha deixat a un company,... també s'ha de poder catalogar-los per gènere, assignar-los "*TAG*" i fer-ne la cerca ràpida, crear una llista de desitjos tant a llegir com a comprar,... El gestor de llibres "*bCatalog*" ha de ser multi idioma. Tots aquests requeriments poden modificar-se durant el cicle de vida del PFC i s'aniran documentant i detallant.

El projecte ha d'aplicar patrons de disseny per a resoldre problemes de disseny que es repeteixen i que es presenten en situacions particulars. Així amb la seva aplicació s'aconsegueix una solució amb èxit.

En aquest projecte no es pretén crear una aplicació visualment atractiva ni amb infinitat de funcionalitats, sinó que es busca adquirir els coneixements d'altres tecnologies i aplicar els diversos conceptes / matèries estudiats durant l'Enginyeria Informàtica amb la finalitat de demostrar el potencial i les capacitats de les eines utilitzades.

Un cop comparades les solucions disponibles del mercat, es realitzarà la part d'implementació del projecte. Es crearà el framework de persistència realitzant el previ anàlisi, disseny i implementació. La finalitat no és crear un súper framework sinó un framework útil i lleuger per a poder ser escalable, extensible i reutilitzable en altres projectes futurs.

Per tal que pugui ser escalable i extensible, s'utilitzarà programació orientada a objectes, patrons de disseny, principis SOLID i s'aplicaran bones pràctiques.

Com que no es disposa de molt temps per a la seva realització, el framework de persistència incorporarà les funcionalitats bàsiques amb la qual cosa ha de ser una implementació oberta per a poder incorporar funcionalitats més avançades i poder millor el framework fent-lo més robust.

Totes les possibles millores, com per exemple crear l'aplicació adaptada a dispositius mòbils o tablets, poder registrar els llibres mitjançant un lector QR o per ISBN, associar llibres en format digital i poder-los visualitzar sense necessitat d'estar en línia,... podran ser implementades a futur i no entren dins els objectius d'aquest projecte.

Durant el seu desenvolupament s'utilitzarà Trello per organitzar les tasques i fer el seguiment de la gestió del projecte. Per al desenvolupament s'utilitzarà l'IDE Eclipse i es crearà un repositori de codi mitjançant GitHub.

Objectius generals i específics

L'objectiu principal és adquirir els coneixements d'aplicacions distribuïdes i les diferents tecnologies utilitzades. Així com també, analitzar les diferents opcions que hi ha actualment per a implementar la capa de persistència per a aplicacions JEE combinant amb diferents gestors de bases de dades.

S'ha de realitzar l'anàlisi, disseny i implementació del framework de mapeig entitat – relació i finalment crear una aplicació per gestionar un catàleg de llibres per tal de demostrar l'ús del framework i les seves funcionalitats.

Objectius principals

- Adquirir els coneixements i familiaritzar-me amb la tecnologia J2EE i els seus frameworks.
- Aprofundir en el llenguatge de programació Java.
- Utilitzar patrons de disseny.
- Aplicar bones pràctiques.
- Analitzar gestors de bases de dades i eines per a la persistència de les dades.
- Crear un framework de persistència.
- Crear una aplicació per a gestionar llibres i demostrar les funcionalitats del framework creat.

Eines i tecnologies escollides

Per a poder assolir els objectius marcats pel projecte, s'utilitzaran les següents eines, tecnologies o frameworks,

- **IDE Eclipse:** per al desenvolupament de l'aplicació utilitzant el llenguatge Java.
- **Persistència de dades:** mitjançant Java Data Objects JDO i Java Persistence API JPA.
- **Gestor de codi:** GitHub.
- **Altres:** Trello, MagicDraw, Microsoft Project,...

Planificació amb fites i temporització

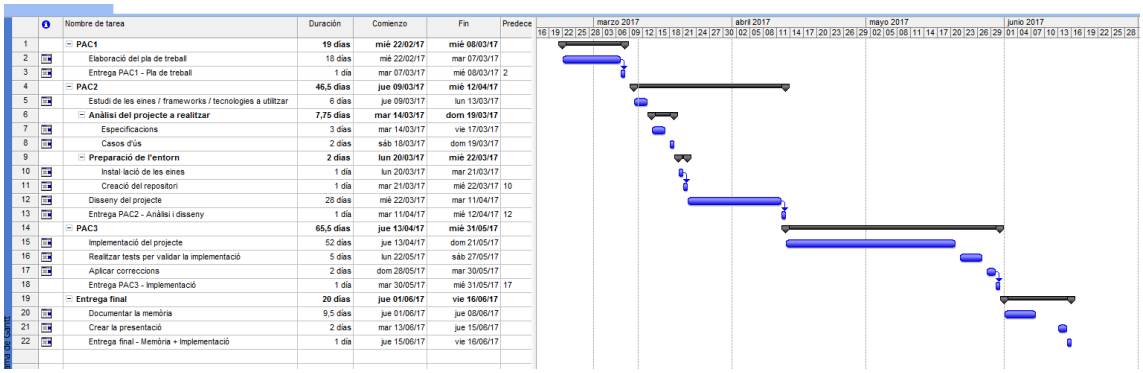
Per a realitzar la planificació del projecte s'utilitzarà el programa Microsoft Project per a gestionar els recursos, tasques a realitzar i la seva temporització, genera el Diagrama de Gantt,...

El seguiment de les tasques durant el desenvolupament del projecte es realitzarà mitjançant un tauler al Trello.

Dates de les entregues

-	PAC1	Pla de treball	08/03/2017
-	PAC2	Anàlisi i disseny	12/04/2017
-	PAC3	Implementació	31/05/2017
-	Entrega final	Memòria + implementació	16/06/2017

Diagrama de Gantt



	Nombre de tarea	Duración	Comienzo	Fin	Predece
1	**PAC1**	**19 días**	**mié 22/02/17**	**mié 08/03/17**	
2	Elaboració del pla de treball	18 días	mié 22/02/17	mar 07/03/17	
3	Entrega PAC1 - Pla de treball	1 día	mar 07/03/17	mié 08/03/17	2
4	**PAC2**	**46,5 días**	**jue 09/03/17**	**mié 12/04/17**	
5	Estudi de les eines / frameworks / tecnologies a utilitzar	6 días	jue 09/03/17	lun 13/03/17	
6	**Anàlisi del projecte a realitzar**	**7,75 días**	**mar 14/03/17**	**dom 19/03/17**	
7	Especificacions	3 días	mar 14/03/17	vie 17/03/17	
8	Casos d'ús	2 días	sáb 18/03/17	dom 19/03/17	
9	**Preparació de l'entorn**	**2 días**	**lun 20/03/17**	**mié 22/03/17**	
10	Instal·lació de les eines	1 día	lun 20/03/17	mar 21/03/17	
11	Creació del repositori	1 día	mar 21/03/17	mié 22/03/17	10
12	Disseny del projecte	28 días	mié 22/03/17	mar 11/04/17	
13	Entrega PAC2 - Anàlisi i disseny	1 día	mar 11/04/17	mié 12/04/17	12
14	**PAC3**	**65,5 días**	**jue 13/04/17**	**mié 31/05/17**	
15	Implementació del projecte	52 días	jue 13/04/17	dom 21/05/17	
16	Realitzar tests per validar la implementació	5 días	lun 22/05/17	sáb 27/05/17	
17	Aplicar correccions	2 días	dom 28/05/17	mar 30/05/17	
18	Entrega PAC3 - Implementació	1 día	mar 30/05/17	mié 31/05/17	17
19	**Entrega final**	**20 días**	**jue 01/06/17**	**vie 16/06/17**	
20	Documentar la memòria	9,5 días	jue 01/06/17	jue 08/06/17	
21	Crear la presentació	2 días	mar 13/06/17	jue 15/06/17	
22	Entrega final - Memòria + Implementació	1 día	jue 15/06/17	vie 16/06/17	

Sistemes gestors de bases de dades

Introducció

Un sistema gestor de bases de dades (*DataBase Management System*) és un sistema que permet la creació, gestió y administració de bases de dades, com també la gestió de les estructures necessàries pel seu emmagatzemament i la cerca d'informació de la manera més eficient possible. A més de proporcionar les eines per a afegir, modificar i analitzar les dades, els usuaris poden accedir a la informació utilitzant eines específiques de consulta i de generació d'informes o mitjançant aplicacions específiques.

Aquests sistemes també proporcionen mètodes per a mantenir la integritat de les dades, per a administrar l'accés dels usuaris a les dades i poder recuperar la informació si el sistema pateix algun mal funcionament i les dades es malmeten. També permeten presentar la informació en diversos formats i la generació d'informes amb gràfics i taules.

Per tal d'accedir a les seves dades, s'utilitzen llenguatges de consulta. Són llenguatges d'alt nivell que simplifiquen les consultes i la presentació de la informació.

Les característiques principals d'un SGBD són,

- Permet controlar l'accés a les dades.
- Assegurar-ne la seva integritat.
- Gestionar-ne el seu accés concurrent.
- Recuperar les dades davant de qualsevol fallada del sistema.
- Realitzar còpies de seguretat.

Els seus objectius principals,

- Realitzar consultes no predefinides i complexes.
- Flexibilitat i independència entre les dades i els processos per a poder realitzar qualsevol canvi tecnològic i variació de la base de dades.
- Evitar els problemes de redundància i el risc d'inconsistència.
- Aplicar regles d'integritat per protegir les dades.
- Permetre la concurrència d'usuaris i assegurar les múltiples escriptures en els registres per no provocar inconsistència de dades.
- Polítiques de seguretat per protegir la confidencialitat, encriptació de dades, les autoritzacions, els drets d'accés,...

Història i evolució

Les base de dades s'han utilitzat des dels primers dies dels ordinadors. A diferència dels sistemes moderns, que es poden aplicar a dades i a necessitats diverses, la majoria dels sistemes originals estaven enfocats a les bases de dades específiques i pensades per guanyar velocitat a costa de perdre flexibilitat. En els seus orígens, només estaven a l'abast de les grans organitzacions que es podien permetre disposar de les complexes màquines necessàries.

Sistemes de navegació 1960

A mesura que els ordinadors van començar a guanyar en velocitat i capacitat, va aparèixer l'interès en obtenir un estàndard i Charles Bachman va fundar Database Task Group. El 1971 van publicar el seu estàndard que va passar a ser conegut com "*l'aproximació CODASYL*".

Aquesta estratègia està basada en la navegació manual per un conjunt de dades vinculades en xarxa. Quan s'arrencava la base de dades, el programa retornava un enllaç al primer registre, el qual contenia punters a altres dades. Per tal de trobar un registre concret, el programador havia d'anar seguint els punters fins arribar al registre buscat.

El fabricant Prime va crear un SGBD mitjançant CODASYL i basat en arbres binaris. En canvi, IBM també tenia el seu propi SGBD amb conceptes similars a CODASYL però utilitzava una jerarquia estricta d'ordenació de les dades.

Sistemes relacionals 1970

Un treballador de IBM, Edgar Codd, estava descontent amb el model de navegació CODASYL per la seva falta de operació en la cerca. El 1970 va escriure alguns articles on perfilava una nova aproximació que va acabar amb el document "*A Relational Model of Data for Large Shared Data Banks*". En aquest article es va descobrir un nou sistema per emmagatzemar i treballar amb grans bases de dades. Enlloc de guardar els registres de forma arbitraria en una llista encadenada, l'idea de Codd era utilitzar una taula de registres de mida fixa.

Aquest model relacional, resol el problema de les llistes encadenades i la poca eficiència en emmagatzemar dades disperses on algunes dades d'un registre poder estar en blanc. S'aconsegueix dividint les dades en una sèrie de taules normalitzades, on les quals els elements optatius han de ser extrets de la taula principal per ocupar espai només si el necessiten. En aquest model, els registres es relacionen mitjançant una clau.

Sistemes SQL a finals de la dècada de 1970

A principis de 1970, IBM va començar a treballar en un prototip basat en els conceptes de Codd anomenat System R. D'aquesta manera van començar els sistemes multi taula, el qual les dades es podien disgregar de manera que tota la informació d'un registre no té per què estar emmagatzemat en una única part gran. Les versions multi usuari van ser aprovades pels usuaris entre 1978 i 1979, quan el llenguatge SQL va ser estandarditzat.

Sistemes orientats a objectes 1980

Durant la dècada de 1980, amb l'augment de la programació orientada a objectes influeix en la manera de manipular la informació de les bases de dades. Es comencen a tractar les dades com a objectes. Per tant, si les dades d'una persona estan a la base de dades, els atributs de la

persona es consideren pertanyents a la persona i no són dades estranyes. Això permet establir relacions entre objectes i atributs més que entre camps individuals.

Un altre gran focus d'atenció va ser l'increment de la velocitat i la fiabilitat en l'accés. Al 1989, dos professors de la Universitat de Wisconsin van publicar un article en la que exposaven els seus mètodes per a millorar les prestacions de les bases de dades. La idea consistia en replicar la informació important i més sol·licitada en una base de dades temporal petita amb enllaços a la base de dades principal. Això feia que es pogués buscar molt més ràpid i la millora de prestacions va introduir la indexació.

Sistemes NoSQL 2000

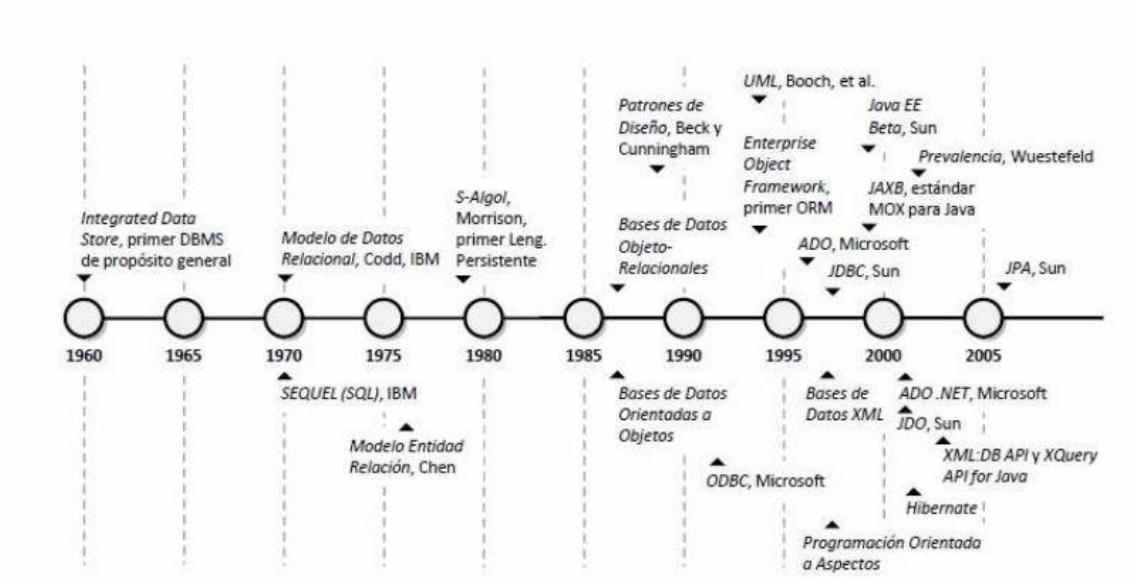
El segle XXI va portar una nova tendència, les bases de dades NoSQL. Aquesta introduirà una línia no relacional significativament diferent de les clàssiques. Generalment no es requereixen esquemes fixes, eviten les operacions *join* emmagatzemant dades desnormalitzades i estan dissenyades per escalar-se horitzontalment. La majoria d'elles es poden qualificar com a magatzems clau – valor o bases de dades orientades a documents.

Recentment hi ha hagut una forta demanda de bases de dades distribuïdes amb tolerància a particions però, d'acord amb el teorema CAP[1], no és possible aconseguir un sistema distribuït que proporcioni simultàniament consistència, disponibilitat i tolerància a les particions.

Sistemes XML 2010

Les bases de dades XML formen part d'un subconjunt de les bases de dades NoSQL. Utilitzen el format d'emmagatzematge XML ja que és obert, llegible per humans i màquines com també àmpliament utilitzat per interoperabilitat.

Hi podem trobar: DB2, BaseX, eXist, MarkLogic Server, MonetDB / XQuery, Sedna,...



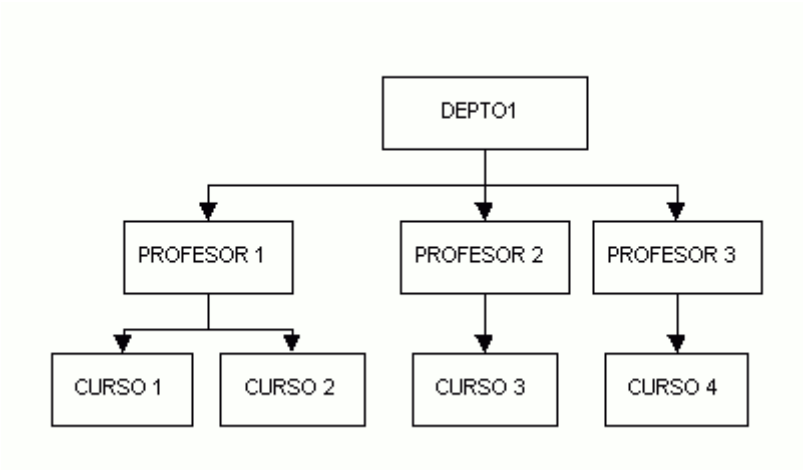
[1] Teorema CAP també conegut com teorema de Brewer. Formula que és impossible garantir les 3 característiques següents en una aplicació distribuïda. Consistència, Disponibilitat i Tolerància a la partició.

Estructura dels esquemes

Totes les bases de dades suportades per un SGBD han de tenir uns esquemes modelats adequadament que coincidint amb la seva evolució històrica també han anat canviant. Els SGBD esperen un model determinat per a poder accedir de manera senzilla a la base de dades.

Estructura jeràrquica

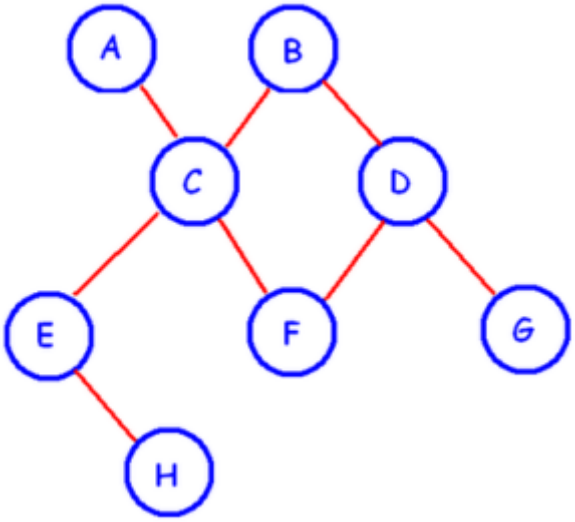
L'estructura jeràrquica va ser utilitzada pels primers SGBD. Les relacions entre registres formen una estructura en forma d'arbre de forma simple però inflexible, les relacions són del tipus 1:N.



Estructura en xarxa

L'estructura en xarxa conté relacions més complexes que les jeràrquiques. Admet relacions de cada un dels registres amb diversos i per diferents camins. Per tant, permet relacions N:N.

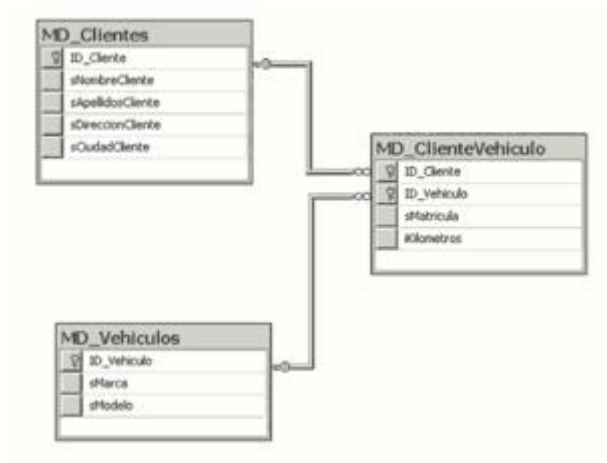
Aquest model permet representar de forma flexible els objectes i les seves relacions. La seva principal qualitat és que l'esquema, vist com un conjunt de nodes connectats per arcs, no té cap restricció. El seu creador va ser Charles Bachman amb l'estàndard CODASYL.



Estructura relacional

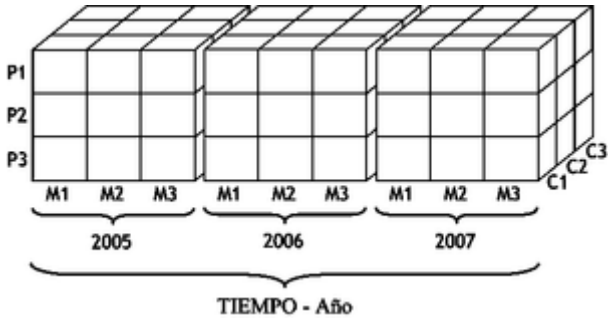
L'estructura relacional és la més estesa avui en dia. S'utilitza en mainframes, ordinadors mitjans i microordinadors. Emmagatzema les dades en files (tuples) i columnes (atributs). Les seves

taules poden estar connectades entre si per claus comuns. El model pot resultar complex per a consultar dades ja que requereix una complexa combinació de les taules.



Estructura multidimensional

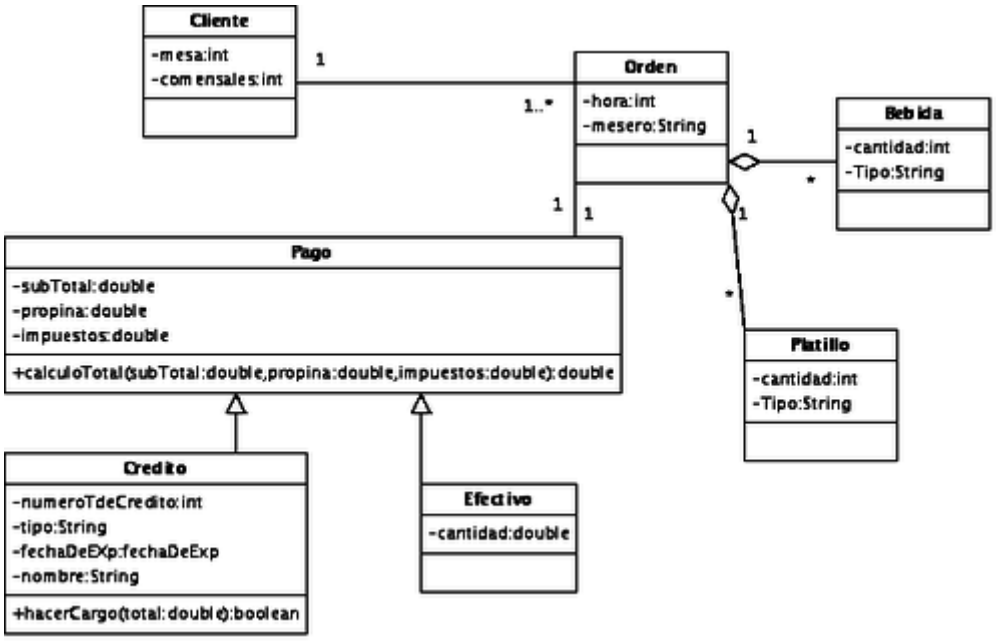
L'estructura multidimensional té semblances al model relacional però tanmateix, enlloc de dues dimensions files – columnes en té N. Aquesta estructura ofereix un aspecte de fulla de càlcul. Les seves característiques són la seva facilitat de manteniment i entendre, ja que els seus registres es guarden de la mateixa manera com es veuen. Les seves altes prestacions han fet que sigui la base de dades més popular per al processat analític de transaccions en línia OLAP[2].



[2] OLAP Online Analytical Processing és una solució que subministra respostes ràpides de consultes a una base de dades.

Estructura orientada a objectes

L'estructura orientada a objectes està dissenyada seguint el paradigma dels llenguatges orientats a objectes. D'aquesta manera suporta tipus de dades gràfics, imatges, veu i text de manera natural. Aquesta estructura té una gran difusió en aplicacions web per a aplicacions multimèdia.



Llenguatges de consulta i components

Els llenguatges de consulta de bases de dades i la generació d'informes permeten consultar bases de dades, analitzar les dades i actualitzar-les segons els privilegis d'usuari. També permet controlar-ne la seguretat per a prevenir accessos no autoritzats.

L'arquitectura d'un SGBD l'especifiquen els seus components, descripció funcional i les seves interfícies. Els principals components d'un SGBD són,

- Interfícies externes. Són els mitjans per a comunicar-se amb el SGBD en ambdós sentits i explotar totes les seves funcions.
- Intèrpret o processador del llenguatge. La major part de les operacions s'efectuen mitjançant un llenguatge de base de dades com ara SQL.
- Optimització de consultes. Realitza la optimització de cada petició i escull el pla d'actuació més eficient per executar-lo.
- Motor de la base de dades. Realitza les operacions requerides sobre la base de dades.
- Mecanisme d'emmagatzematge. Encarregat de traduir les operacions a llenguatge de baix nivell per tal d'accedir a les dades. En algunes arquitectures, aquest mecanisme va integrat amb el motor de la base de dades.
- Motor de transaccions. Per tal d'aconseguir correcció i fiabilitat, la majoria de les operacions internes del SGB es realitzen encapsulades dins de transaccions. Aquest motor segueix l'execució de les transaccions i en gestiona la seva execució d'acord amb les regles que té establertes.

- Gestió i operació de SGBD. Abarca molts altres components que tracten els aspectes de gestió i operativa del SGBD com ara, monitor de prestacions, gestió d'emmagatzemament, mapes d'emmagatzematge.

Persistència de dades

La persistència de dades fa referència a la qualitat de fer sobreviure les dades. És l'acció de preservar la informació d'un objecte de forma permanent i a la vegada, poder-la recuperar per a utilitzar-la de nou. Aquesta informació ha de sobreviure al període d'execució del programa.

Actualment, tota aquesta informació és molt important per a les empreses per tant qualsevol millora o procés que en millori la seguretat, disponibilitat, rendiment en E/S,... d'aquestes dades és de gran interès. Així que, és de gran importància tenir en compte la persistència de les dades a l'hora de desenvolupar un programari nou.

Donat que és un aspecte tant important per a una aplicació en els següents apartats es farà un anàlisi dels principals frameworks de persistència i el mapeig objecte – relacionat (ORM[3]) per tal de poder escollir el framework a utilitzar en el desenvolupament del PFC.

Objectius d'una capa de persistència

Una capa de persistència robusta i orientada a una base de dades relacional ha de complir els següents requisits,

- Diversificació dels mecanismes de persistència.
- Encapsulació del mecanisme de persistència.
- Accions sobre múltiples objectes.
- Transaccions.
- Identificadors dels objectes.
- Cursors.
- Proxies.
- Registres.
- Múltiples arquitectures.
- Utilització de bases de dades de diferents proveïdors.
- Múltiples connexions.
- Controladors natius.
- Consultes SQL.

Definició de framework

Un framework són un conjunt de funcions o codi genèric per a realitzar tasques comuns i freqüents a qualsevol aplicació com ara, registre del log d'aplicació, creació dels objectes, patrons estructurals, gestió de la connexió de la base de dades, neteja de les dades, gestió de la caché,... Per tant, la utilització de frameworks específics permet poder obviar aquestes tasques més repetitives i centrar-se en el desenvolupament de l'aplicació.

"Un framework és un esquema o esquelet per al desenvolupament i implementació d'una aplicació".

[3] ORM Object – relational mapping és una tècnica de programació per a convertir dades de llenguatges de programació orientats a objectes a la seva representació en bases de dades relacionals.

Generalment són implementats amb llenguatges orientats a objectes i en fan més modulars els seus components i n'optimitzen la reutilització del codi. Per al seu desenvolupament, s'apliquen diversos patrons de disseny que n'asseguren la seva escalabilitat.

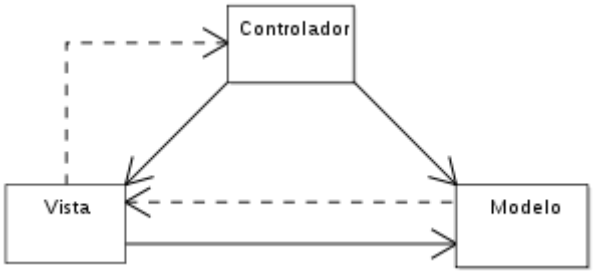
Avantatges d'utilització d'un framework

- El programador no necessita plantejar-se una estructura global de l'aplicació sinó que el framework ja li proporciona un esquelet que ha d'omplir. Accelera el procés de desenvolupament.
- Facilita la col·laboració entre programadors.
- Reutilització de codi ja existent.
- Promoure les bones pràctiques com l'ús de patrons.
- És més fàcil trobar eines, utilitats o llibreries per adaptar-les al framework en qüestió i facilitar el desenvolupament.

Patró MVC

El patró Model Vista Controlador és una proposta de disseny d'aplicacions utilitzat per a implementar sistemes que requereixen de l'ús de les interfícies d'usuari. Davant la necessitat de crear aplicacions robustes amb cicles de vida més llargs i on es potencia la facilitat de manteniment, reutilització del codi i la separació dels conceptes sorgeix el patró MVC.

Bàsicament es separa en tres capes segons la seva responsabilitat. Tenim els models, les vistes i els controladors.



Ús del patró MVC amb un framework

Per aplicar el patró MVC amb un framework cal,

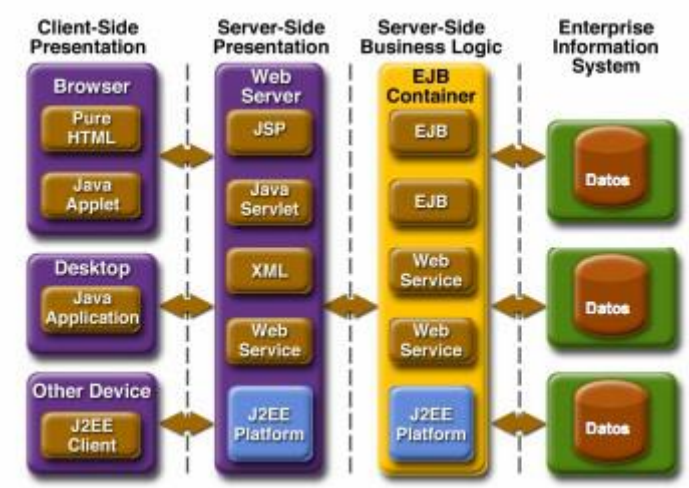
- El framework accedeix al codi de la nostra aplicació.
- El programador ha d'implementar interfícies i/o estendre classes abstractes que seran proporcionades al framework.

Arquitectura J2EE

Com que la realització d'aquest PFC es realitza utilitzant la tecnologia J2EE cal destacar les seves característiques.

- Ús del llenguatge Java.
- Tecnologia potent i madura.
- Cobreix les necessitats existents com ara, les pàgines dinàmiques JSP, lògica de negoci utilitzant Java,...
- Integrable amb altres tecnologies com HTML, Javascript, XML,...
- Ampli suport a la xarxa amb APIs, manuals,...

- De codi obert i amb eines gratuïtes.
- Utilitats ja creades i fàcils d'integrar.
- Facilitat de connexió amb diferents bases de dades mitjançant divers.
- Existència de frameworks de desenvolupament.
- Desenvolupament d'aplicacions multicapa.



Mecanismes de persistència

Existeixen diversos mecanismes de persistència,

- Accés directe a base de dades
  S'utilitza directament la base de dades per implementar la persistència sense cap capa intermèdia entre l'aplicació i el SGBD. Depenent del tipus de base de dades hi ha opcions diferents.
- Mapejadors
  Mecanisme basat en la traducció bidireccional entre les dades encapsulades en els objectes i la font de dades que s'utilitza. S'encarregarà de resoldre els problemes de mapeig[4] entre els diferents paradigmes. Hi ha els mapejadors: Mapping Objecte – Relacional i Mapping Objecte – XML.
- Generadors de codi
  Aquest mecanisme utilitza eines que generen el codi correcte, robust i aplica patrons a partir de les metadates del projecte.
- Orientació a aspectes
  La programació orientada a aspectes AOP[5] és un paradigma que permet definir abstraccions que encapsulen característiques transversals del sistema. Les classes es dissenyen i s'implementen de manera separada als aspectes i es fusionen a posterior, tractant la persistència com un aspecte ortogonal a les funcionalitats, desacoblant el codi de la resta del sistema i així, es modularitza la persistència i es pot reutilitzat el codi generat.
- Llenguatges orientats a objectes

[4] Impedance mismatch.
[5] AOP Aspect – Oriented Programming.

S'utilitzen les funcionalitats pròpies del llenguatge de programació per a resoldre el problema. Mitjançant llibreries estàndards, llenguatges persistents i llenguatges de consulta integrat.

- Esquemes de prevalença
  Es mantenen tots els objectes en memòria i periòdicament s'apliquen mecanismes de serialització per a guardar imatges dels objectes. Aquest mecanisme és senzill i força ràpid però presenta problemes d'escalabilitat.

Mapejador d'objectes relacional ORM

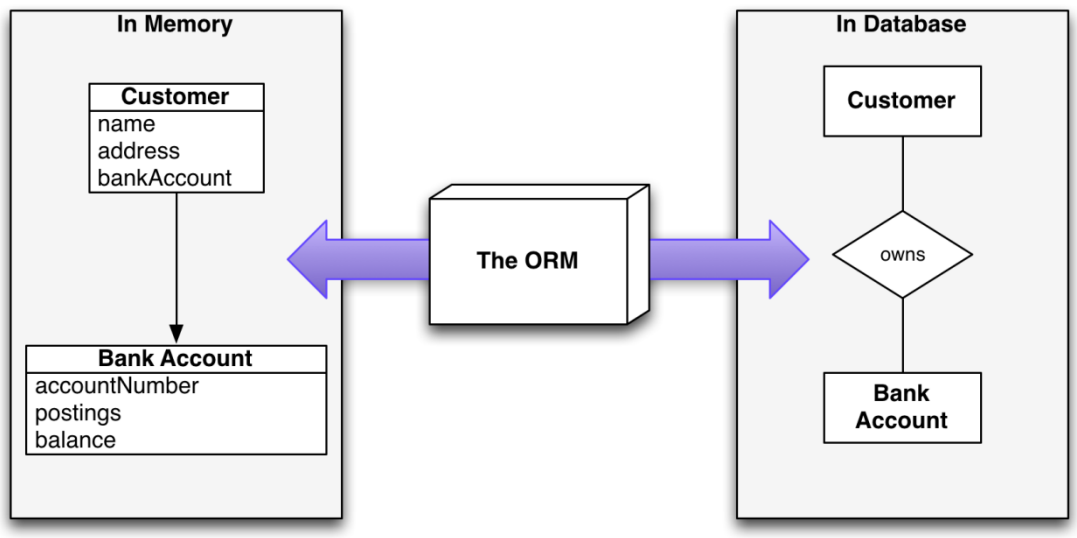
Per a desenvolupar el framework de persistència del PFC utilitzaré un mecanisme de mapeig d'objectes relacional. Aquest mapeig és una tècnica de programació que converteix dades de llenguatges de programació orientats a objectes a la seva representació en bases de dades relacionals.

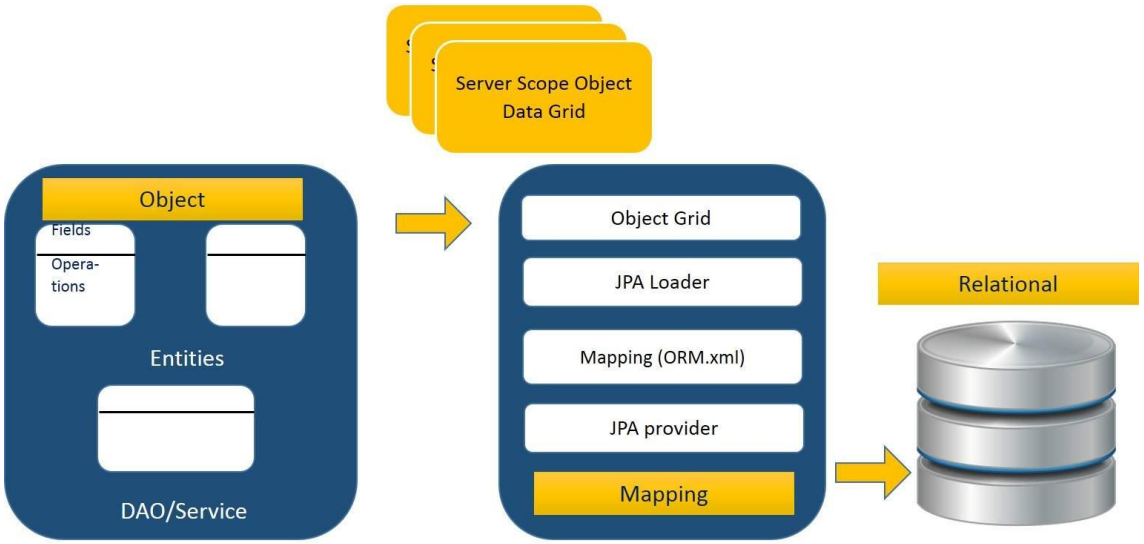
Avantatges

- Facilitat i velocitat d'ús.
- Abstracció de la base de dades utilitzada.
- Seguretat de la capa d'accés a dades contra els atacs.

Inconvenients

- Pot empitjorar el rendiment en entorns amb gran càrrega.
- Aprendre un nou llenguatge de l'ORM.





Object Relational Mapping

Quines són les característiques d'un ORM?

- Patró CRUD

  Tots els objectes que han de poder ser persistits a base de dades s'han de poder Crear – Llegir – Actualitzar – Borrar CRUD[6].

- Càrrega de les relacions

  Per evitar problemes de rendiment, s'ha de controlar quines són les dependències que es carreguen i quines sota demanda (lazy loading). Generalment no interessa carregar tots els objectes relacionats amb un determinat objecte ja que poden portar associats grans volums d'informació i es carreguen en el moment d'accedir-hi (patró Proxy). En funció del tipus de relació interessa que si que es carreguin sempre però això ja queda delegat al desenvolupador del sistema.

- Concurrència

  El framework de persistència ha de permetre l'accés a la mateixa dada a múltiples usuaris i ha d'assegurar la seva integritat a les dades. En aquest sentit, s'apliquen dues tècniques: bloqueig pessimista i bloqueig optimista.

  - Bloqueig pessimista: no es permet l'accés fins que l'usuari que modifica les dades fa el *commit* o *rollback*. Es poden provocar bloquejos i deadlocks[7].
  - Bloqueig optimista: no es bloqueja el registre que s'està modificant i s'assumeix que les dades no canviaran des de la seva lectura.

- Memòria caché

  Per norma general, les aplicacions apliquen la regla del 80 – 20, és a dir, el 80% dels accessos de lectura corresponen al 20% de les dades de l'aplicació. Per tant, hi ha pocs objectes que siguin accedits freqüentment i cal millorar les consultes repetides. Per això sorgeix la necessitat de treballar amb una memòria caché.

  Aquesta memòria és d'accés temporal. Proporciona accés ràpid a les dades d'ús freqüent i totalment transparent per a l'usuari.

  La forma de treballar és simple, es cerquen les dades sol·licitades primerament a la memòria caché i en cas que no estiguin disponibles, es fa la cerca al motor de la base de dades.

[6] CRUD Create – Read – Update – Delete.

[7] Deadlock és un bloqueig en espera permanent al trobar-se dos usuaris actualitzant alhora un mateix registre.

Frameworks de persistència de dades

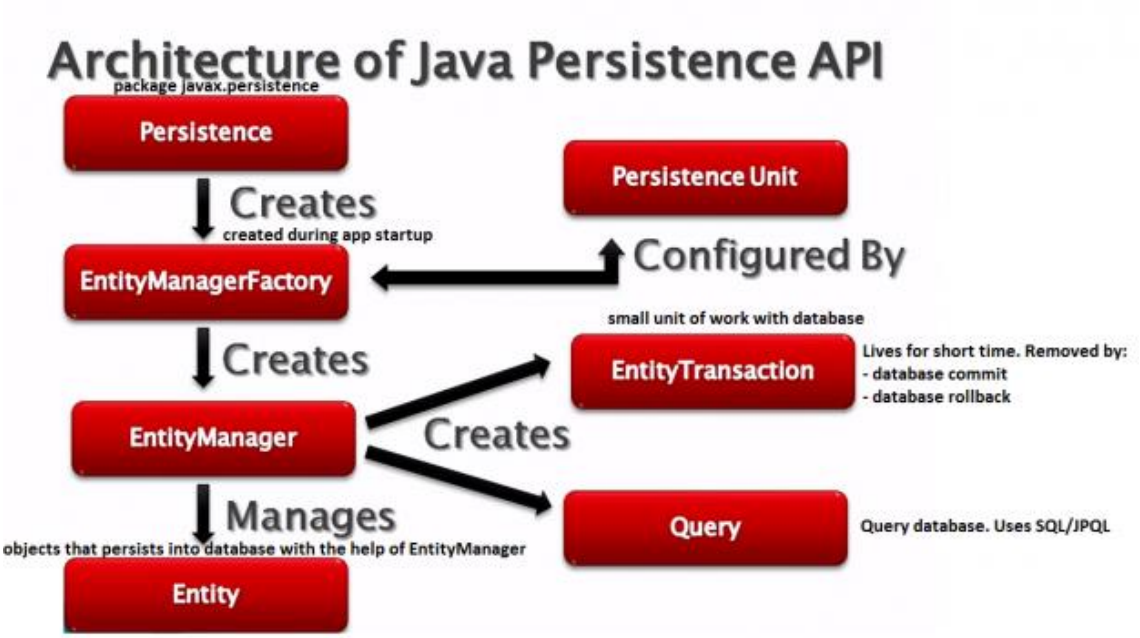
Tal com s'ha definit anteriorment, un framework defineix un conjunt estandarditzat de conceptes i bones pràctiques per afrontar una determinada problemàtica de referència.

En aquest apartat em centraré en descriure els frameworks més destacats per a treballar amb J2EE.

Java Persistence API JPA

Java Persistence API o JPA és una API de persistència desenvolupada per a la plataforma J2EE. Permet treballar amb dades relacionals a les aplicacions utilitzant les tecnologies Java. El seu objectiu és no perdre els avantatges de la orientació a objectes al interactuar amb les bases de dades aplicant el patró de mapeig objecte – relacional.

El mapeig objecte – relacional és la relació entre les entitats Java i les taules de la base de dades. Es realitza mitjançant anotacions a les pròpies classes de l'entitat.



Característiques

- Abstracció del codi del proveïdor de persistència.
- Incorpora llenguatges de consulta orientat a objectes JPQL[8]. D'aquesta manera s'elimina la necessitat d'escriure codi SQL ja que les consultes es realitzaran sobre els objectes i no sobre les taules de la base de dades.
- Permet relacions N:1 ManyToOne, 1:1 OneToOne, 1:N OneToMany i N:N ManyToMany mitjançant les anotacions.
- Incorpora el control de transaccions JTA[9].
- Incorpora una caché de primer nivell per tal d'optimitzar l'accés a la base de dades.

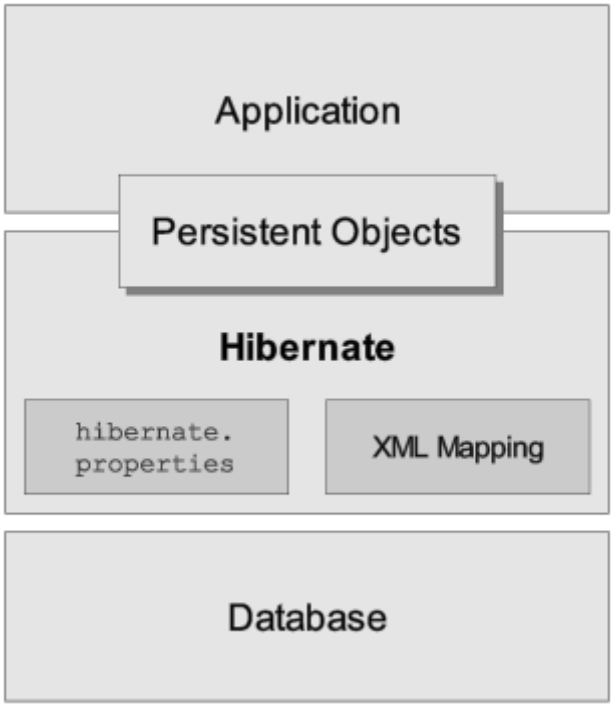
[8] JPQL Java Persistence Query Language.
[9] JTA Java Transaction API.

- Permet incorporar cachés de segon nivell per tal d'incrementar l'escalabilitat i el rendiment del sistema.

Hibernate

Hibernate és un framework ORM concebut per ajudar a la nostra aplicació a persistir les seves dades en una base de dades relacional RDBMS. El seu objectiu principal és el d'abstreure al programador de la capa de persistència de dades i poder adaptar-se a una base de dades ja existent.

Hibernate es base en el mapeig de POJOs. Permet utilitzar els objectes Java definits per l'usuari aplicant reflexió. També disposa d'un conjunt d'utilitats (*Hibernate Tools*) disponibles per Ant i Eclipse per facilitar-ne l'ús.



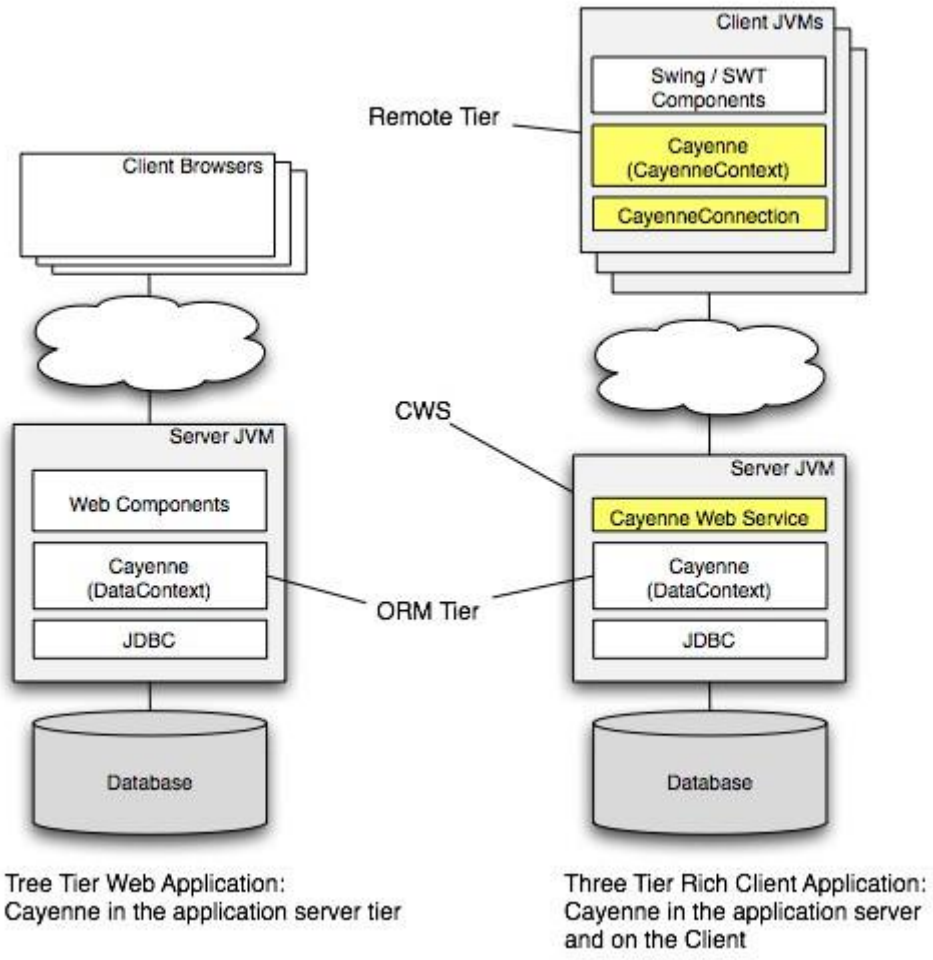
Característiques

- Llicència LGPL.
- Disposa d'un llenguatge de consultes propi HQL.
- Disposa d'una API per a construir les consultes Criteria.
- Es pot configurar mitjançant XML o anotacions.
- Implementa l'estàndard JPA.
- Augmenta la productivitat evitant el codi confús i l'abstracció de la persistència.
- Codi de més bon mantenir per la simplificació del codi.
- Augment del rendiment per l'ús de cachés.
- Independència del proveïdor de base de dades.
- Amplia comunitat.
- Suporta les transaccions.
- Permet herència i polimorfisme.
- Disposa de càrrega diferida (*lazy loading*).

Cayenne

És un framework de persistència de codi lliure sota llicència Apache License. Proveeix un mapping objecte – relacional ORM juntament amb serveis d'accés remot. Cayenne pot abordar una àmplia gamma de necessitats de persistència. Uneix un o més esquemes de bases de dades directament amb objectes Java, en gestiona la confirmació atòmica i la generació d'SQL.

Està dissenyat per ser utilitzat fàcilment sense sacrificar la flexibilitat o el disseny. Incorpora un motor generador de classes basat en Velocity i l'eina CayenneModeler per a realitzar enginyeria inversa.



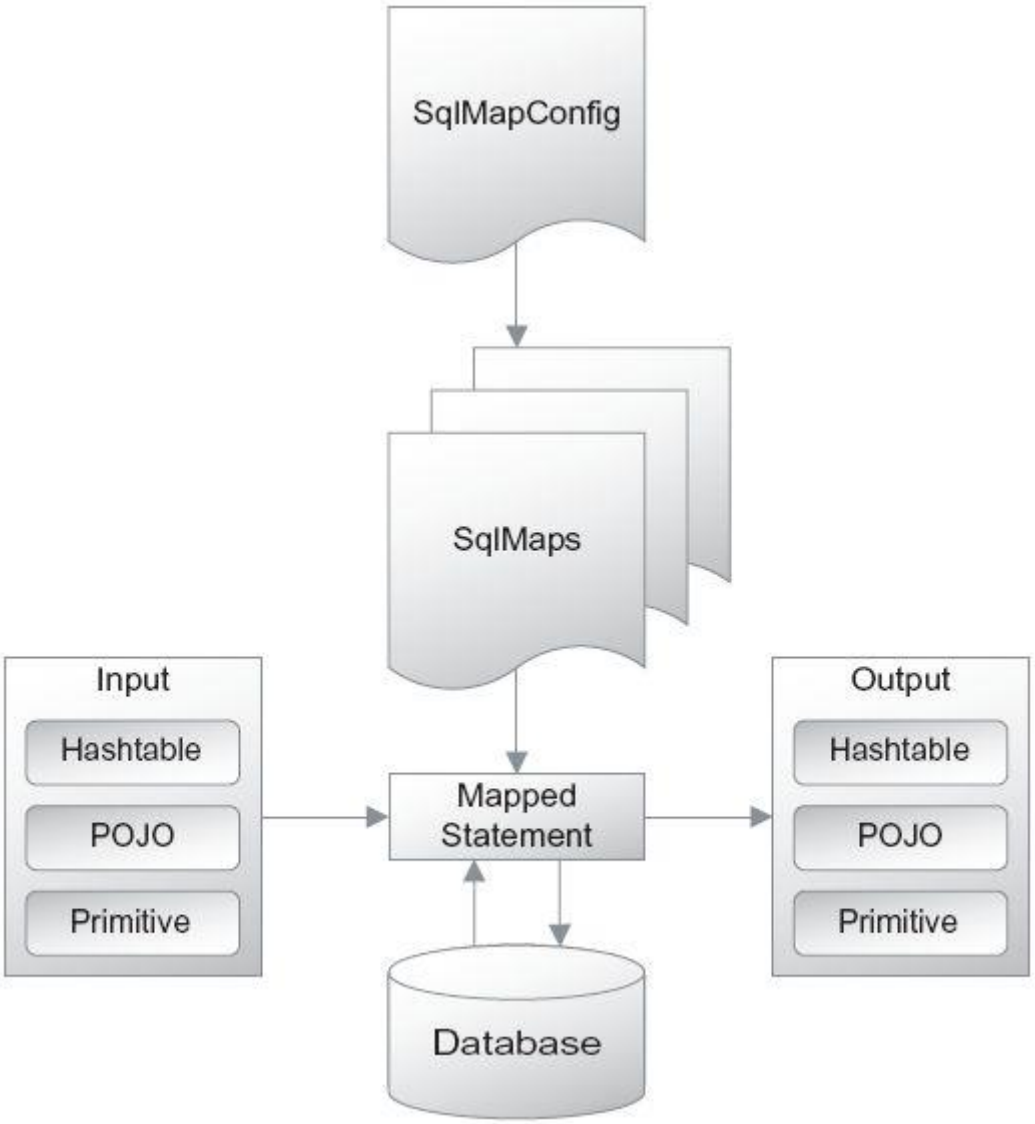
Característiques

- Permet ús de cachés.
- Incorpora una sintaxi completa de consulta d'objectes.
- Permet herència d'objectes.
- Es pot escalar cap a dalt o cap a baix.
- De codi obert.
- Àmplia comunitat.
- Implementa per defecte la càrrega diferida (lazy loading).
- Permet paginació dels resultats.
- Configuració optimista per a garantir la integritat de les dades.

MyBatis

MyBatis és un framework de persistència lliure i desenvolupat sota llicència Apache 2.0. Elimina pràcticament tot el codi JDBC. Permet configuració mitjançant XML o anotacions.

En comparació amb els anteriors, no és un framework ORM. Per tant, no fa mapeig d'objectes Java a taules sinó de mètodes a sentències SQL. D'aquesta manera permet utilitzar totes les funcionalitats de la base de dades com a procediment emmagatzemats, vistes o consultes complexes. És aconsellable per a bases de dades legacy, desnormalitzades o simplement quan es requereix un control total sobre la consulta SQL generada.



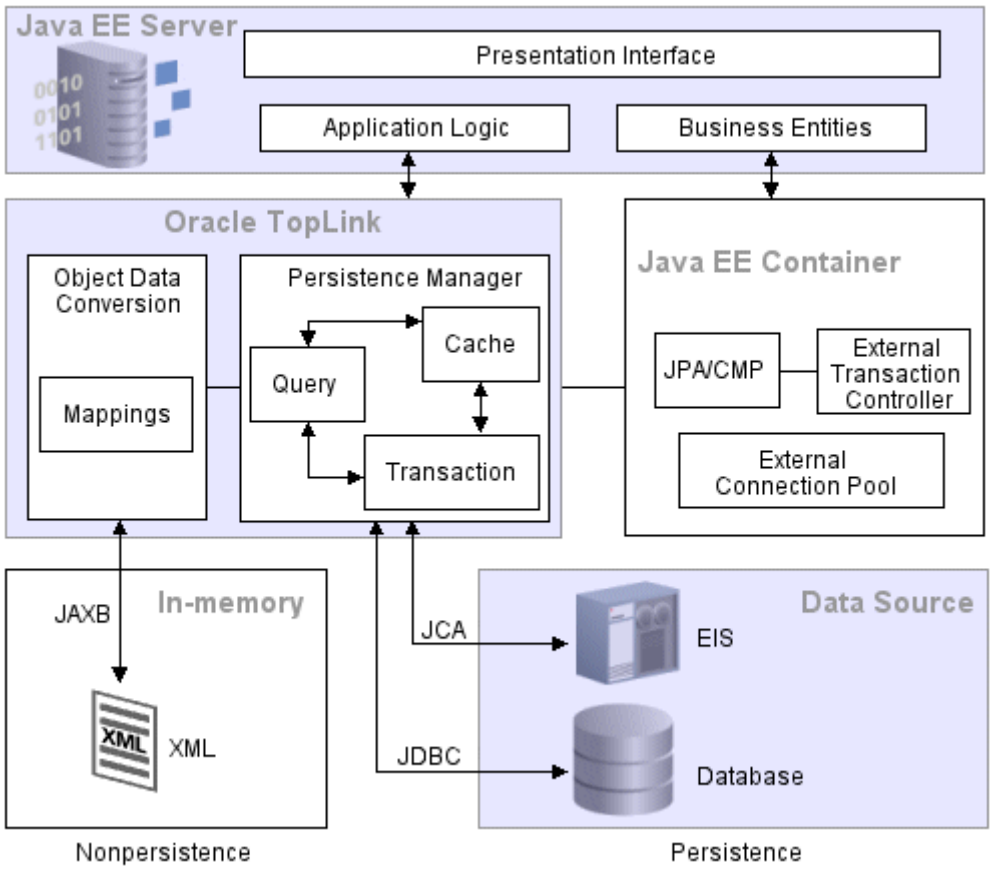
Característiques

- Permet ús de cachés.
- Permet configuració de sentències SQL dinàmiques mitjançant un llenguatge amb sintaxi XML.
- Proporciona un motor de resultats SQL a arbres d'objectes.
- Simplifica la programació envers JDBC raw.
- Integració amb el framework Spring.

- Proveeix generació de codi i artefactes per a realitzar les operacions CRUD.
- Incorpora eines per a Eclipse.
- Incorpora el mòdul de migracions per fer el seguiment dels canvis d'estructura de la base de dades.

TopLink

Oracle TopLink és un avançat framework objecte – relacional que incorpora eines de desenvolupament i millores en el rendiment per a les funcionalitats del sistema. Permet persistir objectes en una base de dades relacional.



Característiques

- Configuració mitjançant anotacions o XML.
- Control de concurrència tant pessimista com optimista.
- Permet utilització de cachés.
- Persistència d'objectes Java a bases de dades relacionals mitjançant Java Databse Connectivity JDBC.
- Flexibilitat arquitectònica.
- Optimitzat per a un rendiment altament escalable i concurrència amb opcions d'optimització extensa de rendiment.
- Eines d'integració amb Ecplise.

Avaluació comparativa

	JPA	Hibernate	Cayenne	MyBatis	TopLink
Base de dades	DB relacionals	DB relacionals	DB relacionals	DB relacionals	DB relacionals
Llenguatge	SQL, JPQL,...	HQL, SQL, Criteria,...	SQL	SQL	SQL, JPQL,...
OOP	Si	Si	Si	Si	Si
Anotacions	Si	Si	Si	No	Si
Complexitat	Baixa	Mitjana / Alta	Mitjana	Baixa	Mitjana
Configuració	1 sol fitxer	1 per configuració i 1 per mapejos	1 sol fitxer	Fitxers XML	1 per configuració i 1 per mapejos
Caché	Si	Si	Si	Si	Si
Transaccions	Si	Si			Si
Concurrència		Pessimista / Optimista	Optimista		Pessimista / Optimista

MyBatis vs ORM

Comparant MyBatis amb els frameworks ORM s'ha de descatar,

- Més lleugera.
- Més control sobre les sentències SQL.
- Millor integració amb bases de dades legacy o desnormalitzades.
- De més fàcil aprenentatge.

Anàlisi i disseny del framework de persistència

El framework a implementar es dirà *acasacuberta_pPersistence* i s'utilitzarà JPA + MySql. Per a implementar aquest framework s'utilitzaran patrons de disseny, s'aplicaran bones pràctiques i principis SOLID. D'aquesta manera, es podran aplicar millores més fàcilment i es garanteix un programari de millor qualitat.

Característiques

- Per a la gestió de la connexió s'aplicarà el patró Singleton, una única instància de connexió.
- S'aplicaran els principis,
    - DRY Don't Repeat Yourself.
    - SRP Single Responsability Principle.
    - ISP Interface Segregation Principle.
    - OCP Open Close Principle.
    - IoC Inversion of Control.
- Per a la gestió dels estats de la connexió s'aplicarà el patró State.
- S'utilitzarà el patró Factory per a la creació d'objectes.
- S'estructura el codi amb els següents paquets,
    - configuration: accions de configuració del sistema i gestió de la base de dades.
    - core: accions comuns de persistència.
    - utils: classes d'utilitats comunes.
- Actors del sistema:
    - Usuari
    - Administrator

Diagrama de classes

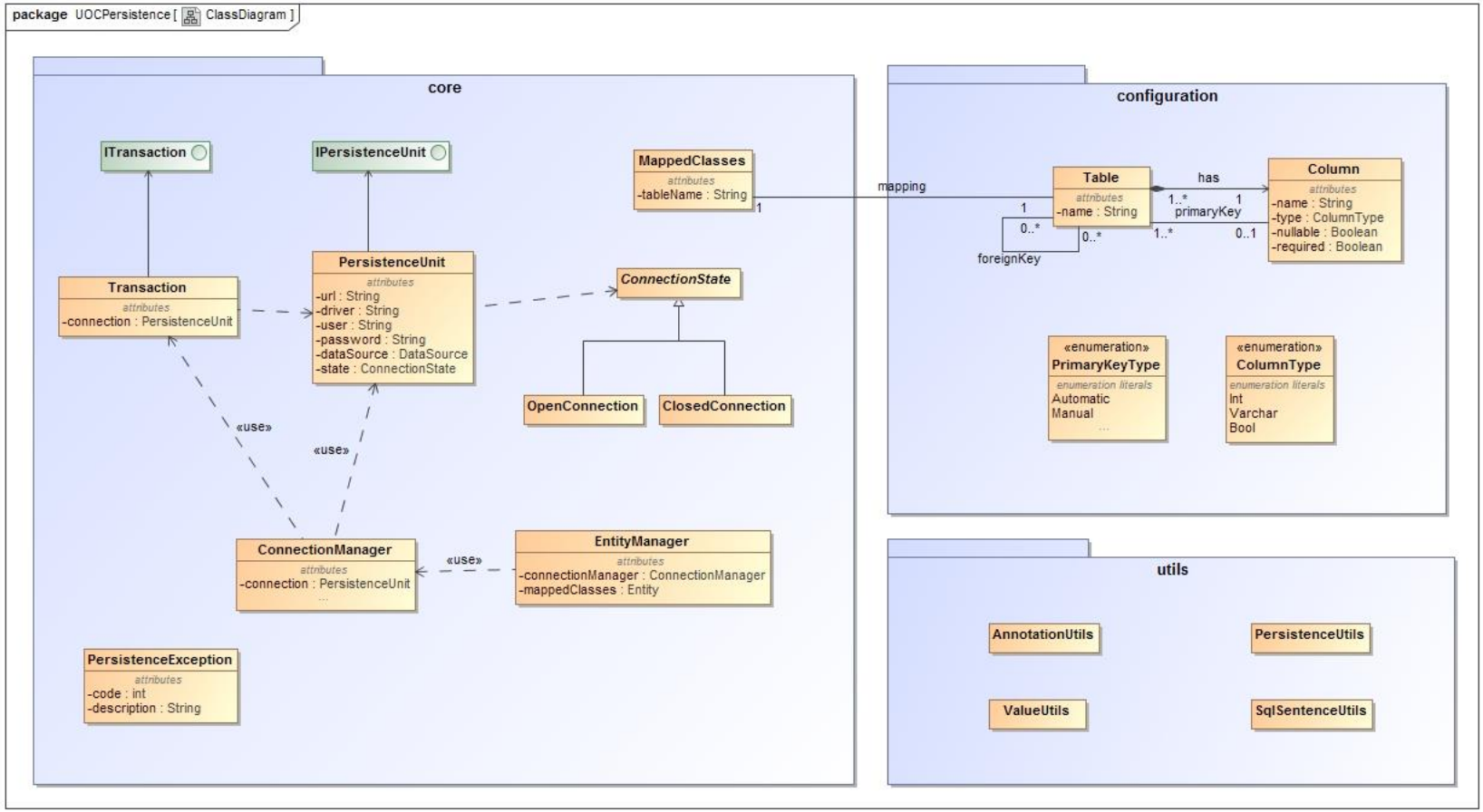
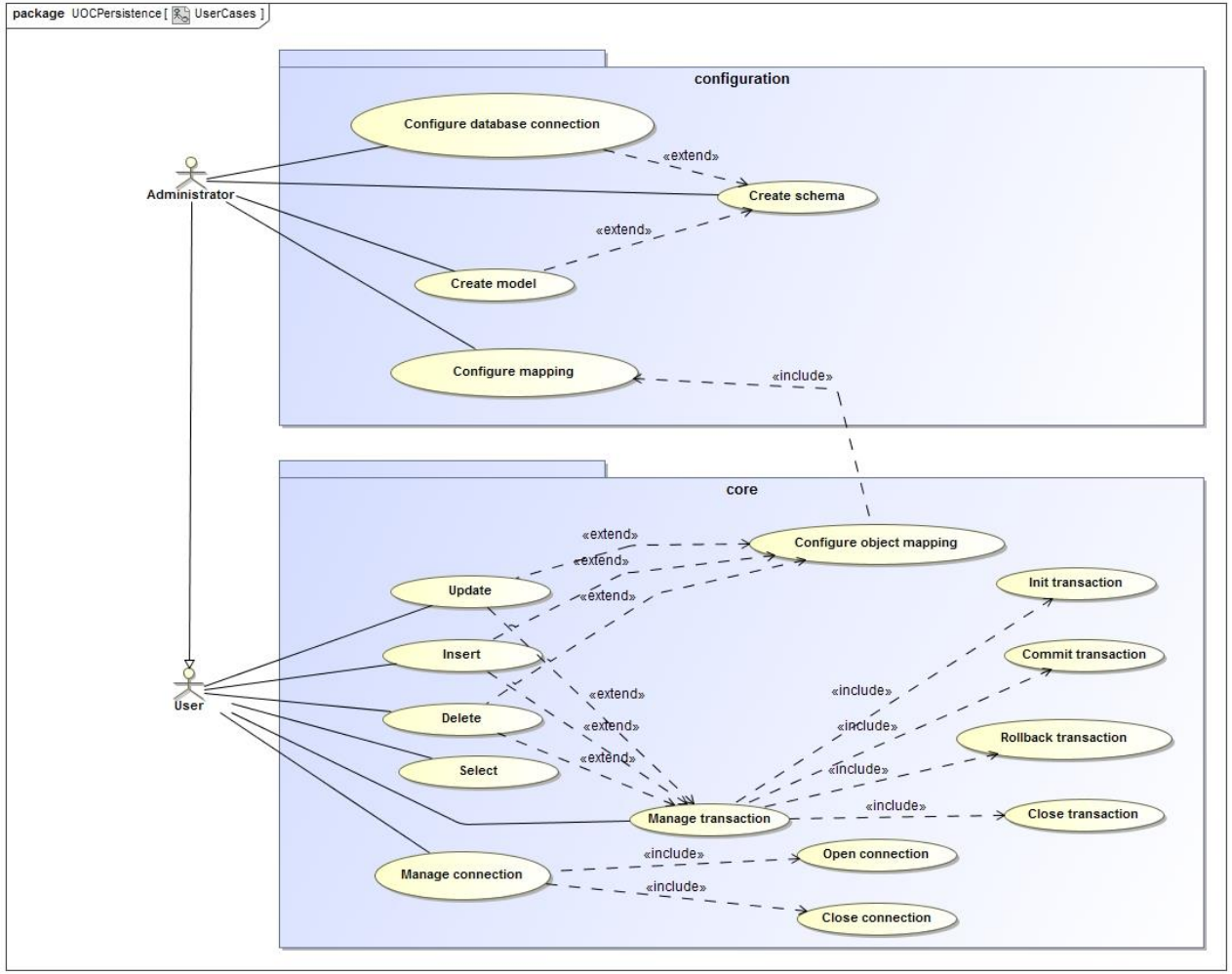


Diagrama casos d'ús



Descripció dels casos d'ús

Configure database connection		
Descripció	Permet configurar les dades de connexió de la base de dades amb les que treballarà el framework de persistència.	
Flux principal	1	L'administrador crea el fitxer de configuració.
	2	L'administrador omple les dades.
	2.1	Introdueix les dades del host.
	2.2	Introdueix l'usuari.
	2.3	Introdueix el password.
	2.4	Introdueix el port.
	2.5	Introdueix el nom de la base de dades.
Flux alternatiu	3	L'administrador en valida les dades introduïdes i guarda els canvis.

Create schema		
Descripció	Permet crear l'usuari amb el que es treballarà per crear el model de la base de dades.	
Flux principal	1	L'administrador introdueix el nom de l'usuari.
	2	L'administrador introdueix el password de l'usuari.

Create model		
Descripció	Es crea el model de la base de dades segons la configuració de la connexió.	
Flux principal	1	L'administrador introdueix les credencials d'accés. Usuari / Password de l'schema.
	2	L'administrador executa la generació del model.
Flux alternatiu	1	Si les credencials no són correctes es genera la corresponent excepció.
	2	Si l'execució de la generació del model genera algun error es registra l'excepció i el model no es crea.

Configure mapping		
Descripció	Permet configurar el mapeig entre els objectes POJO que farà servir l'aplicació i les taules del model.	
Flux principal	1	L'administrador crea un objecte POJO que serà el mapeig d'una de les taules de la base de dades.
	1.1	S'indica quina és la taula a la que es referencia utilitzant l'anotació *@Table*.
	1.2	Es crea la propietat que farà referència a la columna de la taula mitjançant l'anotació *@Column.*
	1.3	Es repeteix el pas 1.2 per totes les columnes de la taula.
	3	Es repeteix el pas 1 per a totes les taules de la base de dades a mapejar.

Configure object mapping		
Descripció	Aplicar la traducció de l'objecte POJO generat a codi SQL.	
Flux principal	1	L'usuari realitza una operació CRUD d'un objecte.
	2	El sistema recupera la configuració concreta de la base de dades.
	3	El sistema aplica els mapejos entre l'objecte POJO i la taula.
	4	El sistema aplica la traducció de l'acció a codi SQL.
	5	El sistema executa la sentència.
	6	El sistema mapeja les dades de la taula a l'objecte POJO.
	7	L'usuari visualitza les dades retornades.
Flux alternatiu	2	Si les dades de configuració no són correctes es genera l'excepció corresponent i es cancel·la el procés.
	3	Si hi ha qualsevol error en el mapeig de l'objecte POJO i la taula es genera l'excepció corresponent.
	4	Si no es pot generar la sentència SQL es genera l'excepció corresponent.
	5	Si es produeix un error en l'execució de la sentència es genera una excepció.
	6	Si el sistema no mapeja correctament les dades de la taula amb l'objecte POJO es genera una excepció.

Update		
Descripció	Modificar els valors d'un registre ja existent a la base de dades.	
Flux principal	1	L'usuari obté les dades d'un objecte POJO.
	2	L'usuari en modifica les dades.
	3	L'usuari n'aplica els canvis.
	4	El sistema aplica el mapeig de l'objecte.
	5	L'usuari visualitza el resultat de l'execució.
Flux alternatiu	4	Si es produeix qualsevol error en el mapeig / execució, es genera l'excepció corresponent.
	5	Si s'ha produït algun error, es mostra a l'usuari i es genera l'excepció corresponent.
Comentaris	Aquest cas d'ús es pot realitzar conjuntament amb altres englobats en una transacció.	

Insert		
Descripció	Insertar dades d'un registre nou a la base de dades.	
Flux principal	1	L'usuari introdueix les dades d'un nou objecte POJO.
	2	El sistema en valida les dades.
	3	L'usuari aplica els canvis.
	4	El sistema aplica el mapeig de l'objecte.
	5	L'usuari visualitza el resultat de l'execució.
Flux alternatiu	2	Si hi ha errors de validació en alguns dels camps de l'objecte es mostra a l'usuari per què ho modifiqui.
	4	Si es produeix qualsevol error en el mapeig / execució, es genera l'excepció corresponent.
	5	Si s'ha produït algun error, es mostra a l'usuari i es genera l'excepció corresponent.

Comentaris	Aquest cas d'ús es pot realitzar conjuntament amb altres englobats en una transacció.

Delete		
Descripció	Eliminar un registre ja existent a la base de dades.	
Flux principal	1	L'usuari obté les dades d'un objecte POJO.
	2	L'usuari selecciona eliminar-lo.
	3	El sistema aplica el mapeig de l'objecte.
	4	L'usuari visualitza el resultat de l'execució.
Flux alternatiu	3	Si es produeix qualsevol error en el mapeig / execució, es genera l'excepció corresponent.
	4	Si s'ha produït algun error, es mostra a l'usuari i es genera l'excepció corresponent.
Comentaris	Aquest cas d'ús es pot realitzar conjuntament amb altres englobats en una transacció.	

Select		
Descripció	Consultar les dades d'un conjunt de registres en funció d'uns filtres de cerca.	
Flux principal	1	L'usuari introdueix els filtres de cerca.
	2	L'usuari n'aplica la cerca.
	3	El sistema aplica el mapeig de l'objecte.
	4	L'usuari visualitza el resultat de l'execució.
Flux alternatiu	1	Si l'usuari no introdueix filtres de cerca no es permet la cerca.
	3	Si es produeix qualsevol error en el mapeig / execució, es genera l'excepció corresponent.
	4	Si s'ha produït algun error, es mostra a l'usuari i es genera l'excepció corresponent.

Manage transaction		
Descripció	Gestionar les transaccions de les accions que en permeten atomicitat.	
Flux principal	1	L'usuari sol·licita una nova transacció.
	2	L'usuari en confirma els canvis.
	3	El sistema en tanca la transacció.
Flux alternatiu	2	Si es produeix algun error, es genera l'excepció corresponent i es reverteixen els canvis.

Init transaction		
Descripció	Inicialitzar una nova transacció.	
Flux principal	1	L'usuari sol·licita una nova transacció.
	2	El sistema l'inicialitza.
	3	L'usuari rep la nova transacció.
Flux alternatiu	1	Si la connexió no es troba oberta, s'obre.
	2	Si es produeix algun error, es genera l'excepció corresponent.

Commit transaction		
Descripció	Confirmar tots els canvis fets dins una transacció.	
Flux principal	1	L'usuari en confirma els canvis.
	2	El sistema els aplica.
Flux alternatiu	2	Si es produeix algun error, es genera l'error corresponent i es reverteixen els canvis.

Rollback transaction		
Descripció	Revertir tots els canvis fets en cas de produir-se algun error.	
Flux principal	1	El sistema reverteix tots els canvis fets.
	2	L'usuari és informat de l'error.

Close transaction		
Descripció	Tancar la transacció creada.	
Flux principal	1	El sistema tanca la transacció creada un cop finalitzat.

Manage connection		
Descripció	Gestionar la connexió amb la base de dades.	
Flux principal	1	L'usuari inicialitza la connexió amb la base de dades.
	2	L'usuari té la connexió inicialitzada.
Flux alternatiu	1	Si la configuració de connexió amb la base de dades és incorrecte, es genera l'excepció corresponent i s'informa a l'usuari.

Open connection		
Descripció	Obrir la connexió amb la base de dades.	
Flux principal	1	L'usuari disposa d'una connexió.
	2	L'usuari obre la connexió.
Flux alternatiu	1	Si l'usuari no disposa d'una connexió, s'inicialitza de nou.
	2	Si la connexió no està oberta, s'obre.

Close connection		
Descripció	Tancar la connexió amb la base de dades.	
Flux principal	1	L'usuari disposa d'una connexió.
	2	L'usuari tanca la connexió.
Flux alternatiu	1	Si l'usuari no disposa d'una connexió, s'inicialitza de nou.
	2	Si la connexió està oberta, es tanca.

Anàlisi i disseny de l'aplicació bCatalog

L'aplicació d'exemple que utilitzarà el framework de persistència creat, s'anomenarà acasacuberta_*bCatalog*. A continuació es detallen les seves característiques.

Característiques

- Aplicació web desplegada en un servidor d'aplicacions.
- Utilitzarà una base de dades MySQL per a la seva persistència.
- Utilitzarà el framework de persistència creat.
- El web disposarà de part pública i part privada.
    - Part pública: mostrarà el formulari de login.
    - Part privada: mostrarà la llista de llibres i les accions que es poden realitzar amb ells.
- Arquitectura client – servidor orientada a objectes distribuïts.
- MVC per la part client.
- Actors del sistema:
    - Usuari
    - Usuari identificat

Diagrama de classes

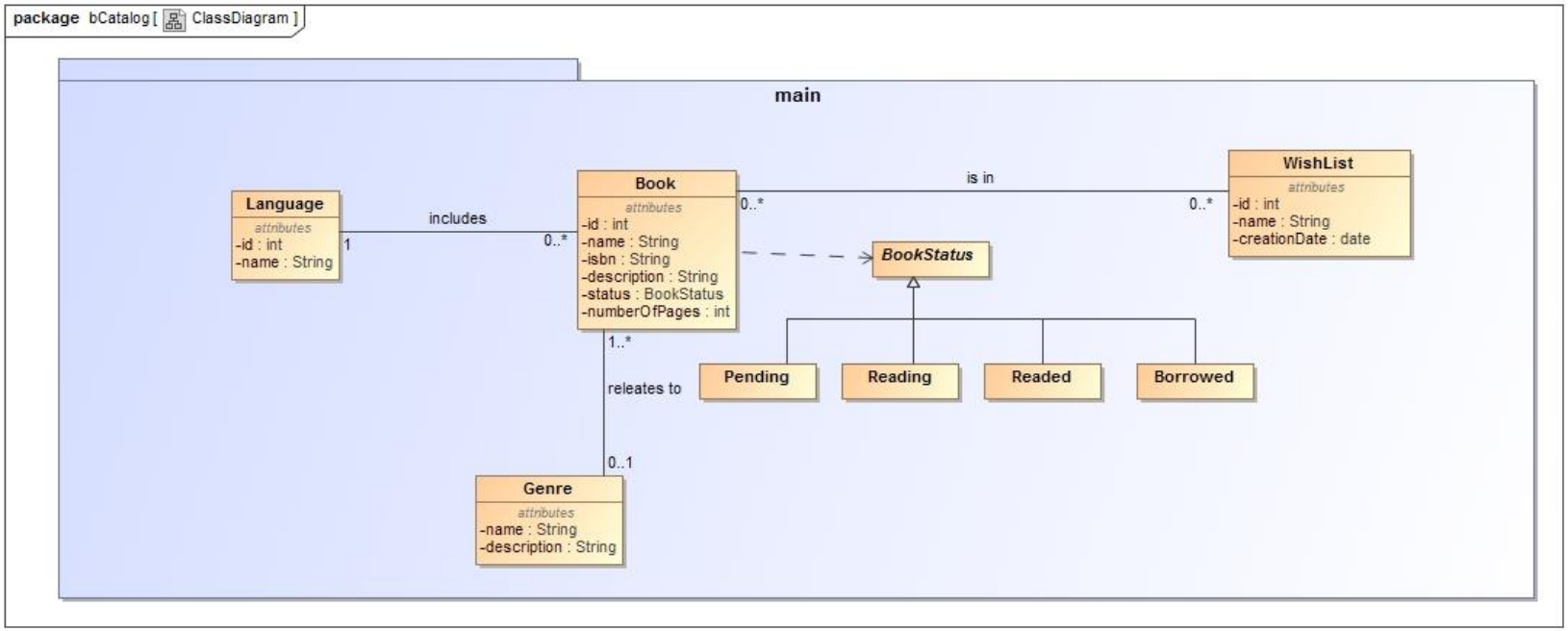
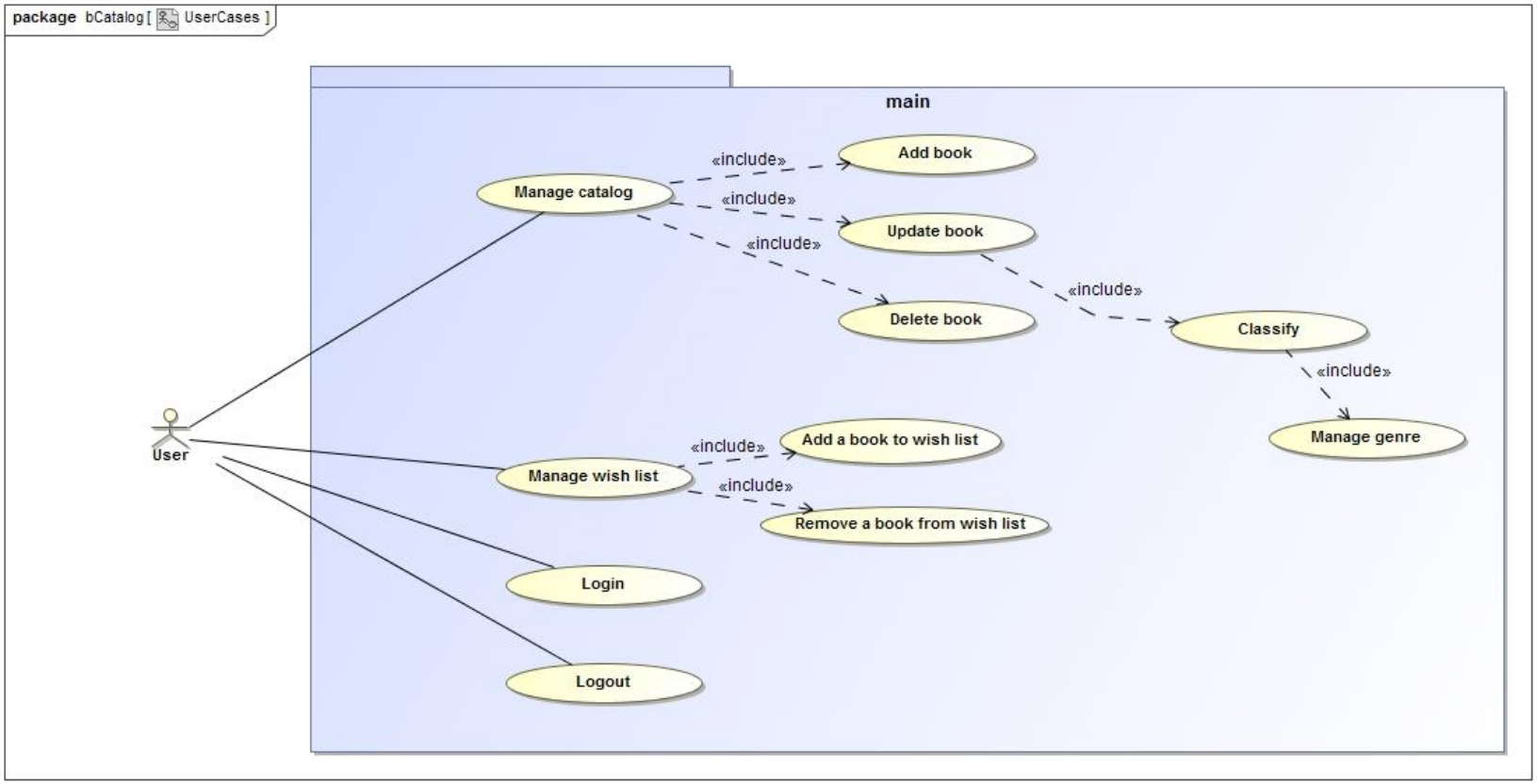


Diagrama casos d'ús



Descripció dels casos d'ús

Manage catalog		
Descripció	Gestionar el catàleg de llibres.	
Flux principal	1	L'usuari visualitza el catàleg de llibres.
	2	L'usuari realitza una acció sobre el catàleg.
Comentaris	Aquest cas d'ús inclou "*Add book*", "*Update book*" i "*Delete book*".	

Add book		
Descripció	Afegir un nou llibre al catàleg.	
Flux principal	1	L'usuari escull la creació d'un nou llibre al catàleg.
	2	L'usuari introdueix les dades del nou llibre.
	2.1	Introdueix el nom del llibre.
	2.2	Introdueix el seu ISBN.
	2.3	Introdueix una descripció.
	2.4	Introdueix el seu número de pàgines.
	3	L'usuari n'accepta els canvis.
	4	El sistema en valida les dades introduïdes.
	5	El sistema crea el nou llibre del catàleg.
	6	El sistema informa a l'usuari que els canvis s'han realitzat.
Flux alternatiu	4	Si les dades introduïdes no són correctes, s'informa a l'usuari per tal que en faci els canvis adients.
	5	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.

Update book		
Descripció	Actualitzar les dades d'un llibre ja existent al catàleg.	
Flux principal	1	L'usuari escull modificar les dades d'un dels llibres del catàleg.
	2	L'usuari en modifica les dades.
	3	L'usuari n'accepta els canvis.
	4	El sistema en valida les dades introduïdes.
	5	El sistema actualitza les dades del llibre.
	6	El sistema informa a l'usuari que els canvis s'han realitzat.
Flux alternatiu	4	Si les dades introduïdes no són correctes, s'informa a l'usuari per tal que en faci els canvis adients.
	5	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.
Comentaris	Aquest cas d'ús inclou "*Classify*".	

Classify		
Descripció	Classificar per gènere i/o tags els llibres del catàleg.	
Flux principal	1	L'usuari selecciona un llibre per a poder gestionar el seu genère i/o tags.
Comentaris	Aquest cas d'ús inclou "*Manage genre*".	

Manage genre		
Descripció	Indicar quins són els gèneres d'un llibre del catàleg.	
Flux principal	1	L'usuari escull un llibre per indicar-li gèneres.
	2	L'usuari introdueix el nom del gènere del llibre.
	3	L'usuari repeteix el pas 2 tantes vegades com gèneres vol associar al llibre.
	4	El sistema guarda les dades.
	5	El sistema informa a l'usuari que els canvis s'han realitzat.
Flux alternatiu	4	Si el gènere introduït no existeix, l'afegeix.
	4	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.

Delete book		
Descripció	Eliminar un llibre del catàleg.	
Flux principal	1	L'usuari escull un llibre del catàleg.
	2	L'usuari marca el llibre per eliminar-lo.
	3	El sistema demana confirmació de borrat a l'usuari.
	4	L'usuari confirma l'acció.
	5	El sistema elimina el llibre del sistema.
	6	El sistema informa a l'usuari que els canvis s'han realitzat.
Flux alternatiu	4	Si l'usuari no en confirma l'acció, no es realitza cap acció.
	5	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.

Manage wish list	
Descripció	Gestionar una llista de desitjos.
Flux principal	1 L'usuari escull un llibre del catàleg.
Comentaris	Aquest cas d'ús inclou "*Add a book to wish list*" i "*Remove a book from wish list*".

Add a book to wish list		
Descripció	Afegir un llibre a una llista de desitjos.	
Flux principal	1	L'usuari escull una llista de desitjos existent.
	2	L'usuari escull el llibre o llibres a incloure a la llista.
	3	L'usuari en valida l'acció.
	4	El sistema afegeix el llibre a la llista de desitjos.
	5	El sistema informa a l'usuari.
Flux alternatiu	4	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.

Remove a book from wish list		
Descripció	Eliminar un llibre d'una llista de desitjos.	
	1	L'usuari escull un llibre associat a una llista de desitjos.

Flux principal	2	L'usuari escull eliminar el llibre de la llista.
	3	El sistema realitza els canvis.
	4	El sistema informa a l'usuari.
Flux alternatiu	3	Si es produeix qualsevol error, es genera l'excepció i es reverteixen els possibles canvis fets.

Login		
Descripció	Identificar un usuari a l'aplicació.	
Flux principal	1	L'usuari introdueix el seu nom d'usuari.
	2	L'usuari introdueix el seu password.
	3	L'usuari accepta l'acció.
	4	El sistema en valida les dades.
	5	L'usuari està identificat al sistema.
Flux alternatiu	4	Si es produeix qualsevol error, es genera l'excepció i s'informa a l'usuari.

Logout		
Descripció	Sortir de l'aplicació.	
Flux principal	1	L'usuari escull sortir de l'aplicació.
	2	El sistema elimina les dades de la sessió de l'usuari.
	3	L'usuari no està identificat.
	4	El sistema mostra la pantalla d'identificació.

Manual instal·lació

Prerequisits de software

Per tal de poder configurar i desplegar el projecte desenvolupat cal tenir instal·lat el següent software:

- Java JDK 1.8 versió 1.8.0_131
- Ant versió 1.10.1
- MySql versió 5.7.18
- JBoss WildFly versió 10.1.0 Final
- Eclipse versió Neon.3 Release (4.6.3)
- Scripts d'esquema de base de dades i introducció de dades inicials executats.
- Connector Java MySql mysql-connector-java-5.1.42-bin.jar

Instal·lació del connector Java MySql al servidor d'aplicacions

- Crear l'estructura de carpetes

  modules/system/layers/base/com/mysql/main

- Crear el fitxer *module.xml* en aquesta carpeta.

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.42-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

- Copiar el JAR *mysql-connector-java-5.1.42-bin.jar* en aquesta carpeta.
- Afegir la configuració de la base de dades al fitxer

  /standalone/configuration/standalone.xml

```
            <keepalive-time time="30" unit="seconds"/>
        </thread-pool>
    </subsystem>
    <subsystem xmlns="urn:jboss:domain:bean-validation:1.0"/>
    <subsystem xmlns="urn:jboss:domain:datasources:4.0">
        <datasources>
            <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled=
            "true" use-java-context="true">
                <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
                </connection-url>
                <driver>h2</driver>
                <security>
                    <user-name>sa</user-name>
                    <password>sa</password>
                </security>
            </datasource>
            <datasource jta="false" jndi-name="java:jboss/mysqlDS" pool-name="mysqlDS" enabled="true"
            use-java-context="true" use-ccm="false">
                <connection-url>jdbc:mysql://127.0.0.1:3306/bCatalog?useSSL=false</connection-url>
                <driver-class>com.mysql.jdbc.Driver</driver-class>
                <driver>mysql</driver>
                <security>
                    <user-name>bcatalog_user</user-name>
                    <password>pfcuoc</password>
                </security>
            </datasource>
            <drivers>
                <driver name="h2" module="com.h2database.h2">
                    <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
                </driver>
                <driver name="mysql" module="com.mysql">
                    <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
                    </xa-datasource-class>
                </driver>
            </drivers>
        </datasources>
    </subsystem>
    <subsystem xmlns="urn:jboss:domain:deployment-scanner:2.0">
        <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"
```

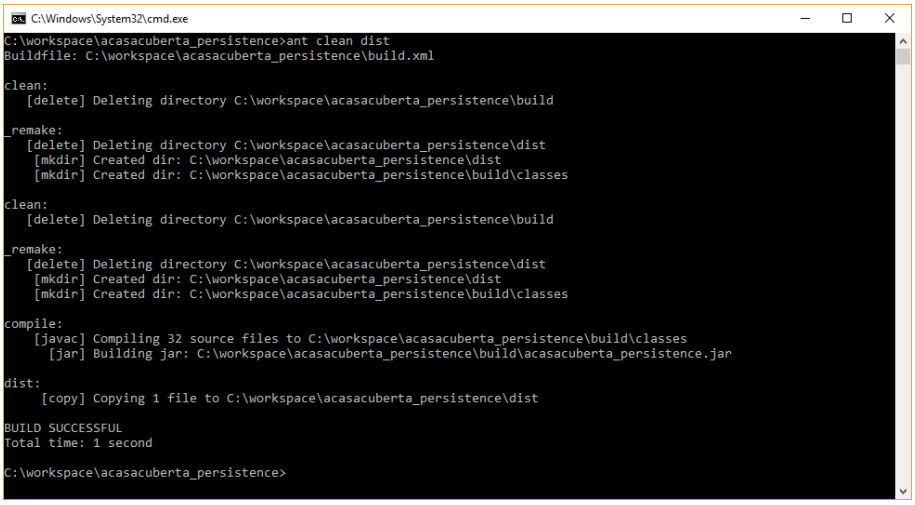
bCatalog_persistence

El framework de persistència *bCatalog_perstistence* està composat per

- build directori de sortida de les classes compilades
- config fitxers de configuració com ara log4j.xml
- dist directori amb el JAR de distribució de l'aplicació
- lib directori amb les llibreries necessàries
- src codi font del projecte

Per generar el JAR de la distribució cal executar la següent comanda des de l'arrel del projecte,

ant clean dist



bCatalog

L'aplicació *bCatalog* amb la utilització del framework de persistència generat està composat per

- build directori de sortida de les classes compilades
- dist directori amb el JAR, EAR i WAR de distribució de l'aplicació
- lib directori amb les llibreries necessàries
    - acasacuberta_persistence.jar
    - log4j-1.2.17.jar
    - mysql-connector-java-5.1.42-bin.jar
- sql scripts SQL amb l'esquema i creació de les dades necessàries
    - bCatalog_data.sql
    - bCatalog_model.sql
- src codi font del projecte
- WebContent pàgines JSP de l'aplicació

Per definir quin és el datasource que farà servir l'aplicació cal configurar el següent fitxer

/src/MET-INF/persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
    xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="bCatalog" transaction-type="JTA">
        <jta-data-source>java:jboss/mysqlDS</jta-data-source>
    </persistence-unit>
</persistence>
```

Per compilar i desplegar l'aplicació cal executar la següent comanda des de l'arrel del projecte,

ant

```
C:\Windows\System32\cmd.exe

C:\workspace\acasacuberta_bCatalog>ant
Buildfile: C:\workspace\acasacuberta_bCatalog\build.xml

clean:
   [delete] Deleting directory C:\workspace\acasacuberta_bCatalog\build
   [delete] Deleting directory C:\workspace\acasacuberta_bCatalog\dist

init:
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\build\jar
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\build\war
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\build\jar\META-INF
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\build\war\WEB-INF
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\build\war\WEB-INF\classes
    [mkdir] Created dir: C:\workspace\acasacuberta_bCatalog\dist

compileEjb:
     [copy] Copying 1 file to C:\workspace\acasacuberta_bCatalog\build\jar\META-INF
     [copy] Copying 1 file to C:\workspace\acasacuberta_bCatalog\build\jar

compileWar:
     [copy] Copying 33 files to C:\workspace\acasacuberta_bCatalog\build\war

jarEjb:
      [jar] Building jar: C:\workspace\acasacuberta_bCatalog\dist\acasacuberta_bcatalog.jar

deployClient:
      [jar] Building jar: C:\workspace\acasacuberta_bCatalog\dist\acasacuberta_bcatalog.war

ear:
     [copy] Copying 1 file to C:\workspace\acasacuberta_bCatalog\dist\META-INF
      [jar] Building jar: C:\workspace\acasacuberta_bCatalog\dist\acasacuberta_bcatalog.ear

deployear:
     [copy] Copying 1 file to C:\Users\Anna\wildfly-10.1.0.Final\standalone\deployments

all:

BUILD SUCCESSFUL
Total time: 0 seconds

C:\workspace\acasacuberta_bCatalog>
```

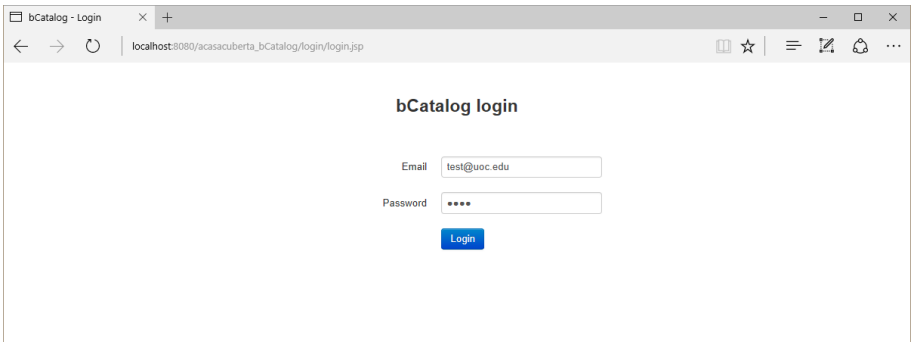
Si tenim el servidor d'aplicacions en execució ja podem validar que el desplegament de bCatalog s'ha realitzat correctament. Es mostrarà la pantalla de login.

http://localhost:8080/acasacuberta_bCatalog

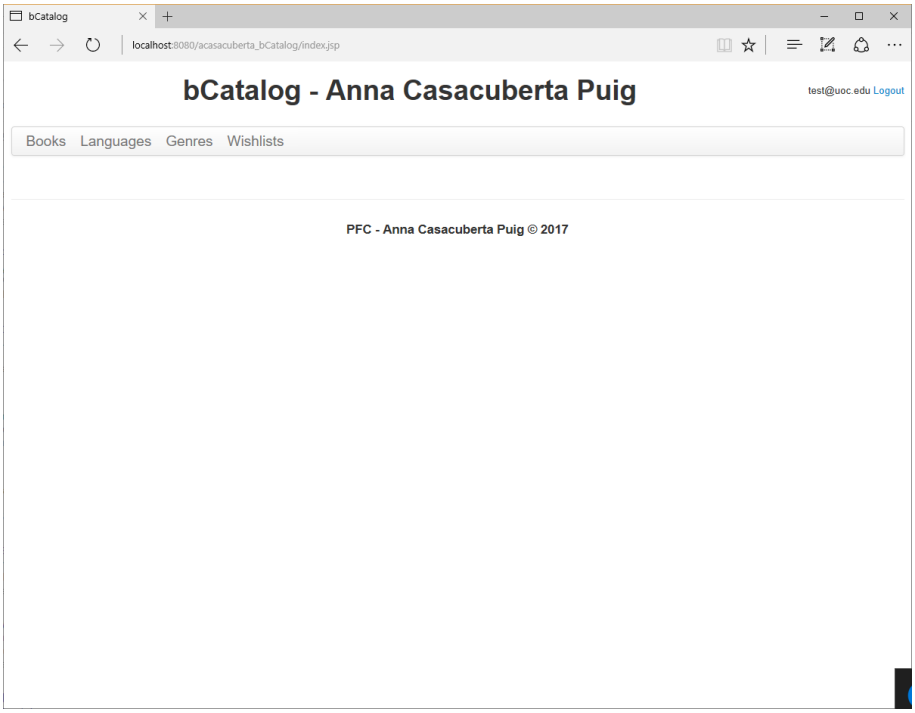
Manual d´ús

Per accedir a l'aplicació *bCatalog* cal introduir les credencials de l'usuari. Per a la demo d'aquesta aplicació he creat l'usuari *test@uoc.edu* amb contrasenya *test*.

A continuació es mostra la pantalla de login,



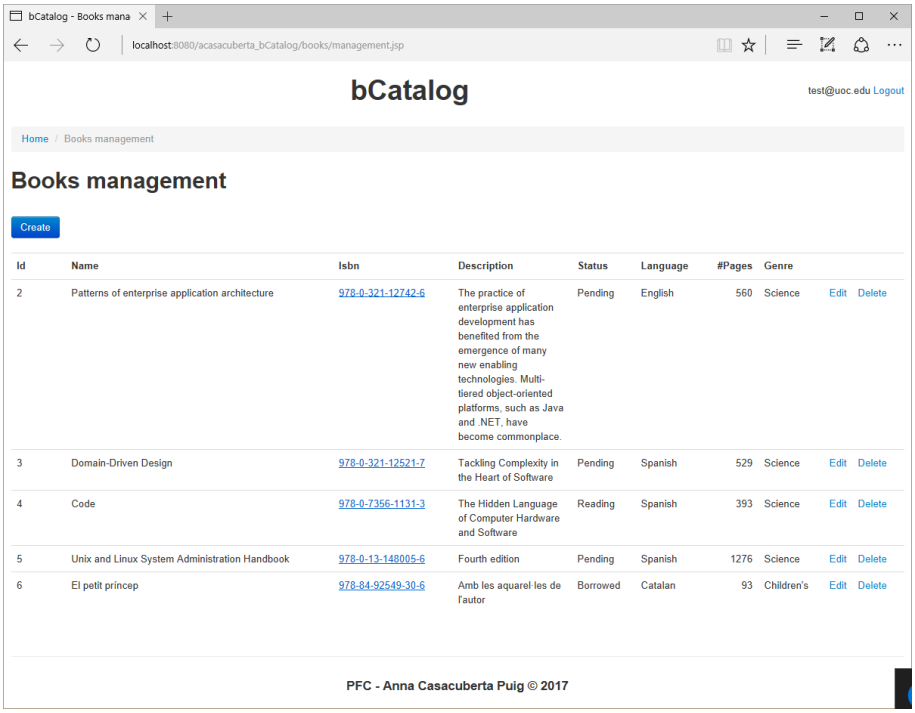
Si l'usuari és correcte s'accedeix a la pantalla principal on es poden veure els menús de l'aplicació.



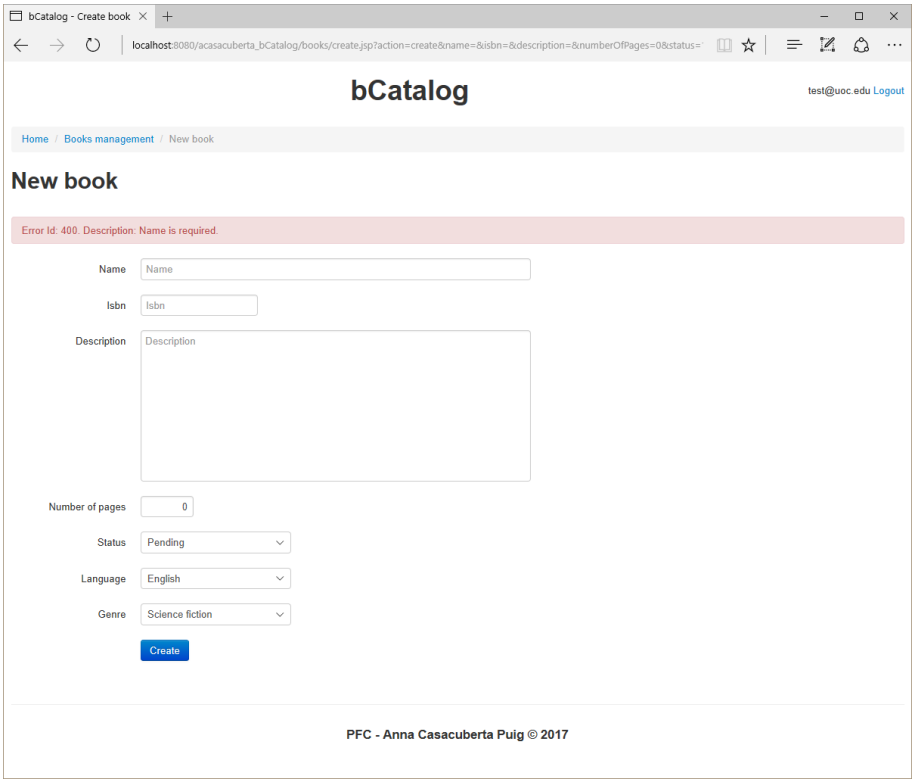
Els diferents menús: *Books*, *Languages*, *Genres* i *Wishlists*, permeten consultar la llista de registres associats, la creació d'un nou registre, la seva modificació de dades o l'eliminació definitiva. Al realitzar una operació d'actualització de dades o creació d'un registre nou es validen les dades introduïdes i en cas d'error es mostra a l'usuari. Si es vol eliminar el registre des de la llista de registres, es demana confirmació a l'usuari.

Books

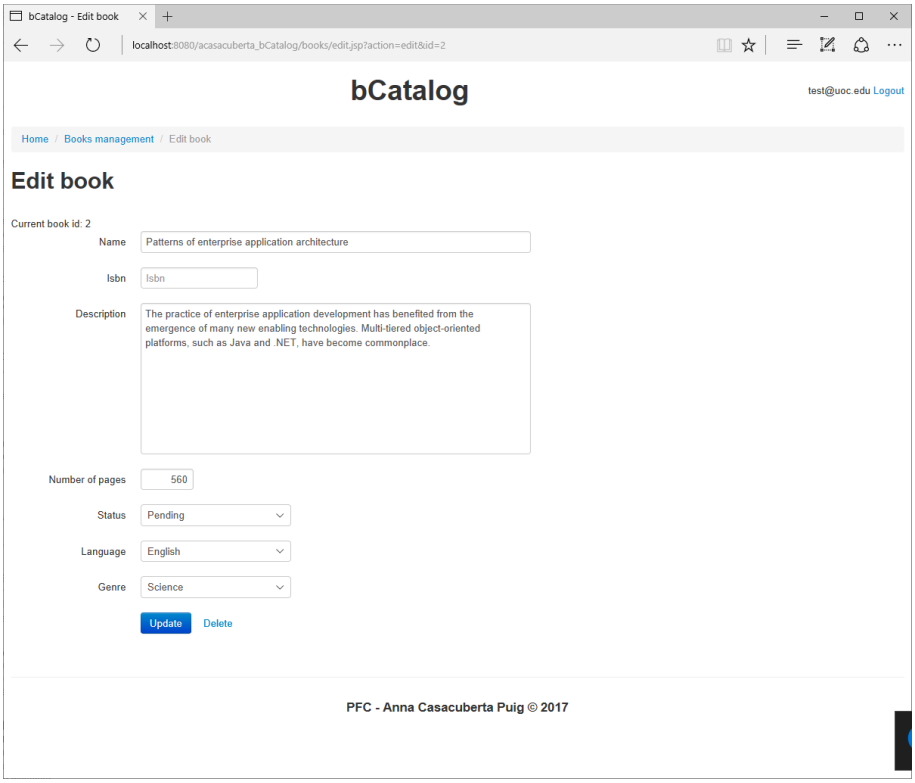
Es mostra la llista de llibres (taula Books de la base de dades) amb les opcions crear registre nou, editar ja existent i eliminar ja existent.



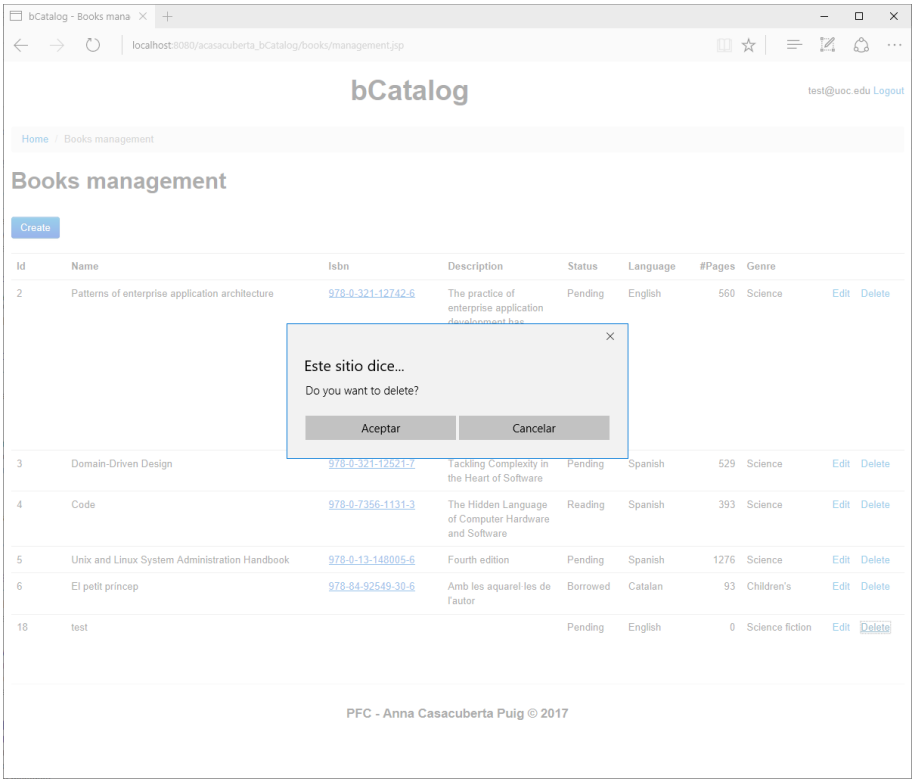
New book



Edit book

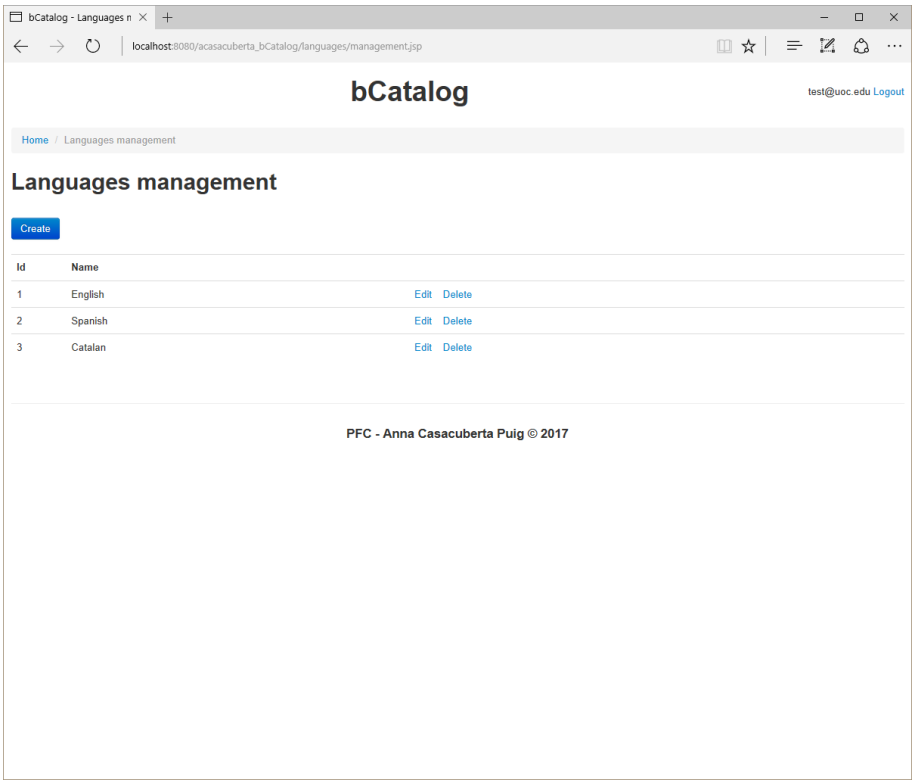


Delete book

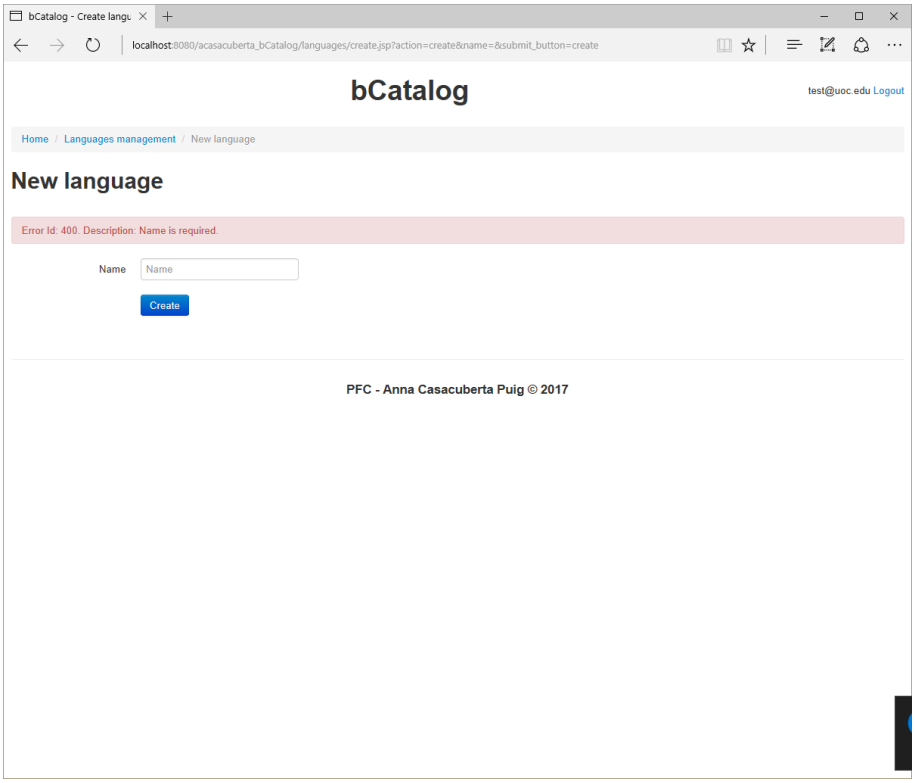


Languages

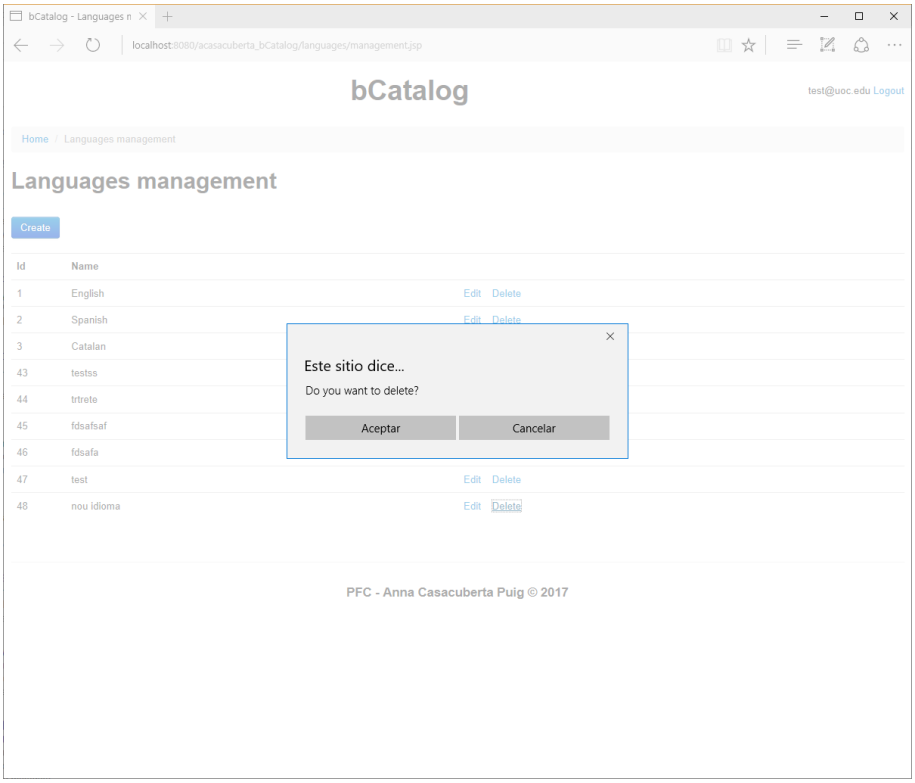
Es mostra la llista d'idiomes (taula Languages de la base de dades) amb les opcions crear registre nou, editar ja existent i eliminar ja existent.



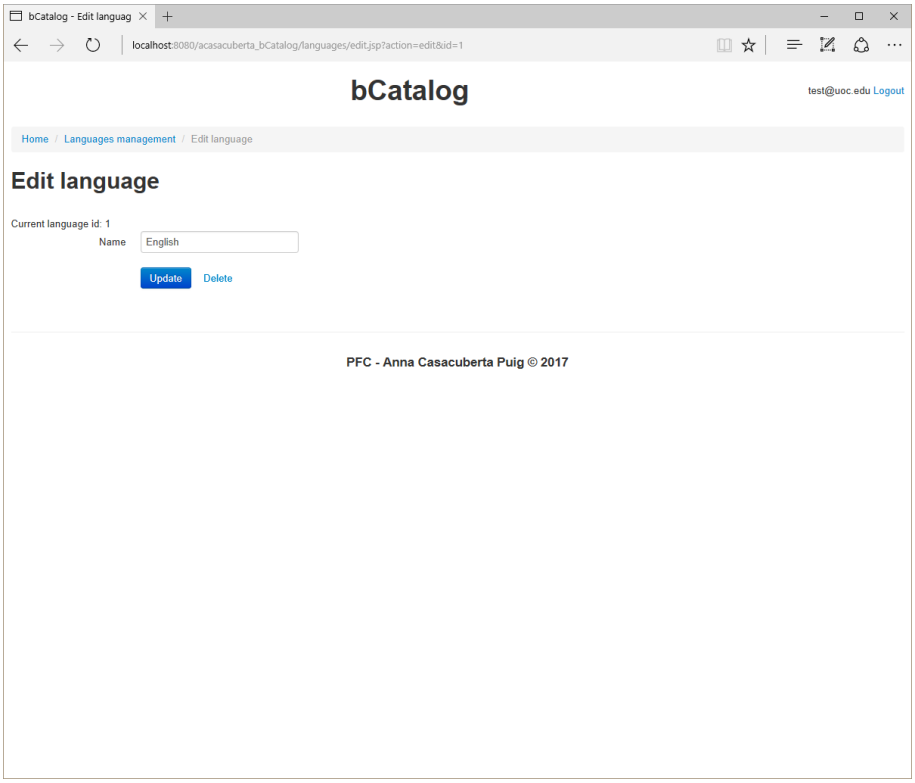
New language



Delete language

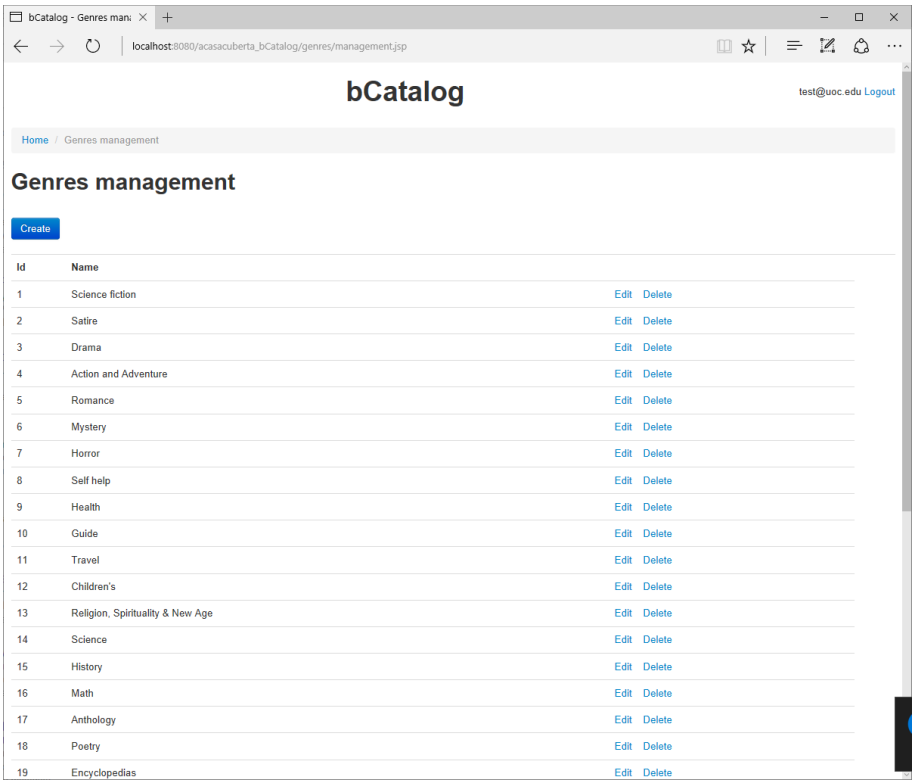


Edit language

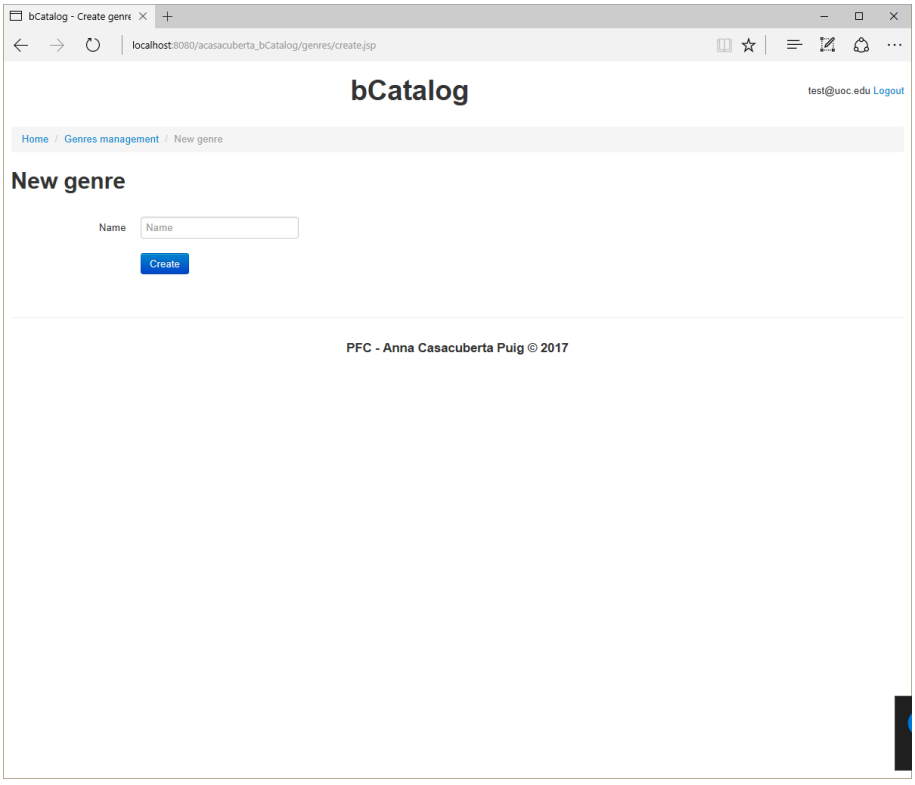


Genres

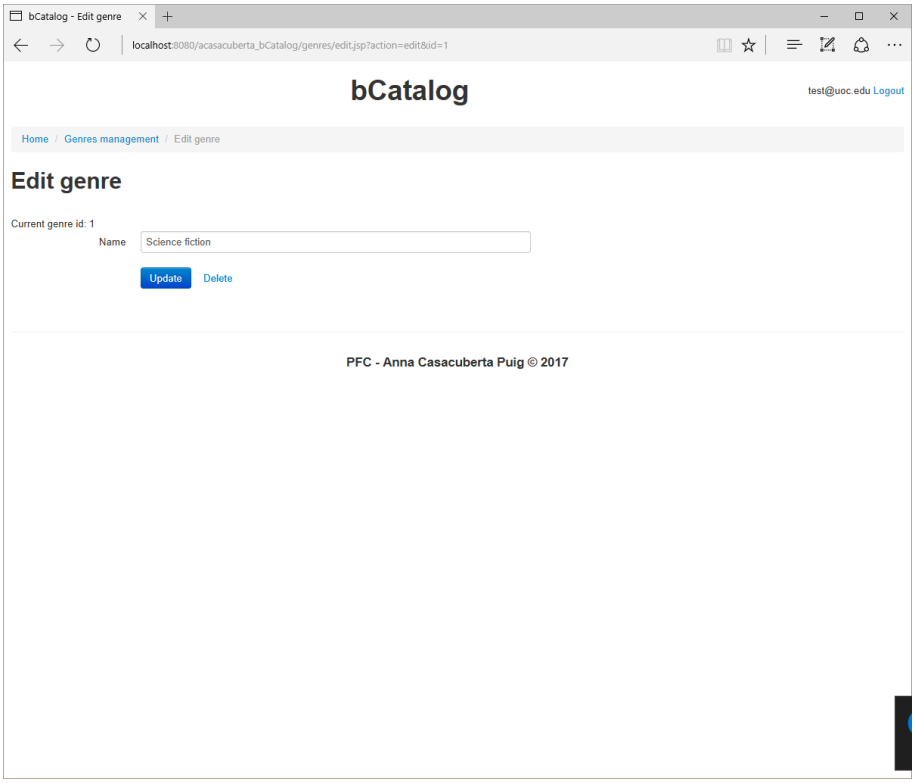
Es mostra la llista de gèneres (taula Genres de la base de dades) amb les opcions crear registre nou, editar ja existent i eliminar ja existent.



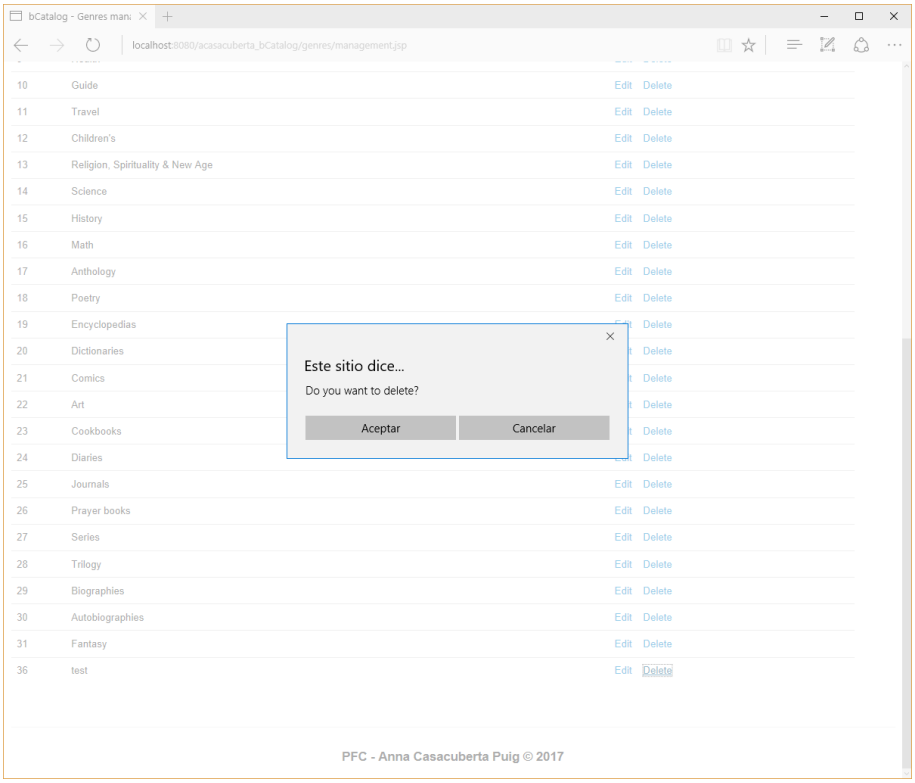
New genre



Edit genre

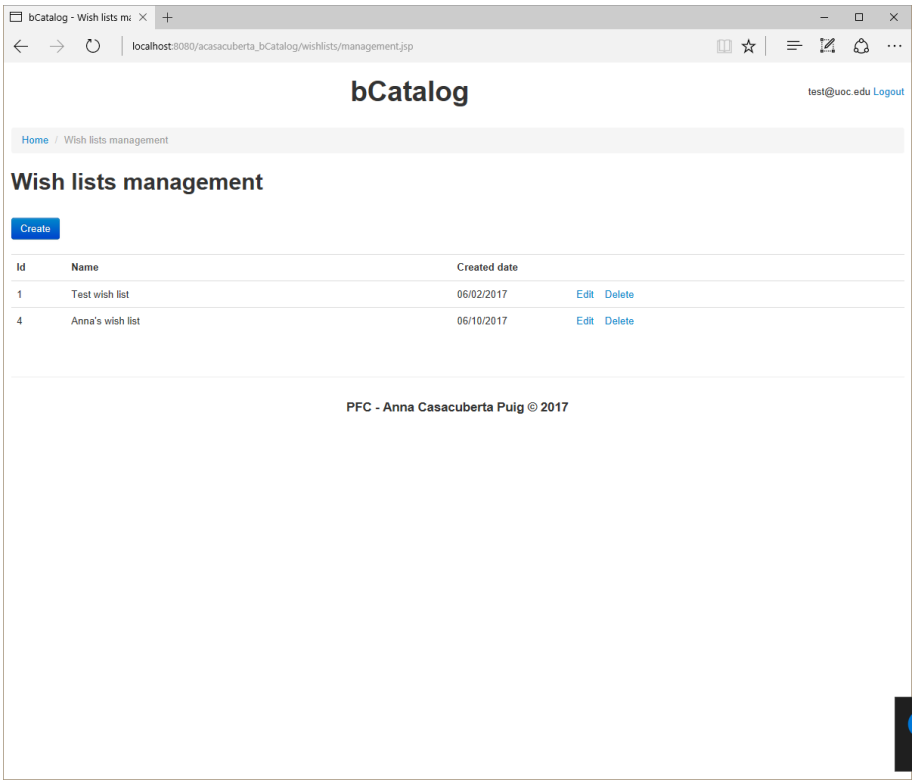


Delete genre

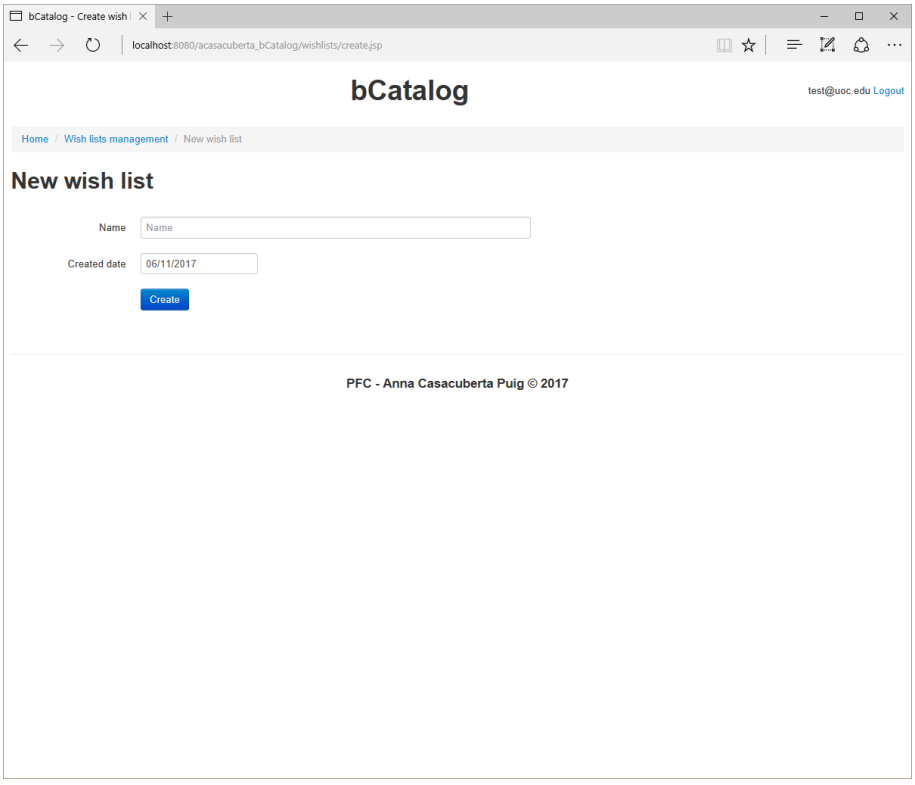


Wish lists

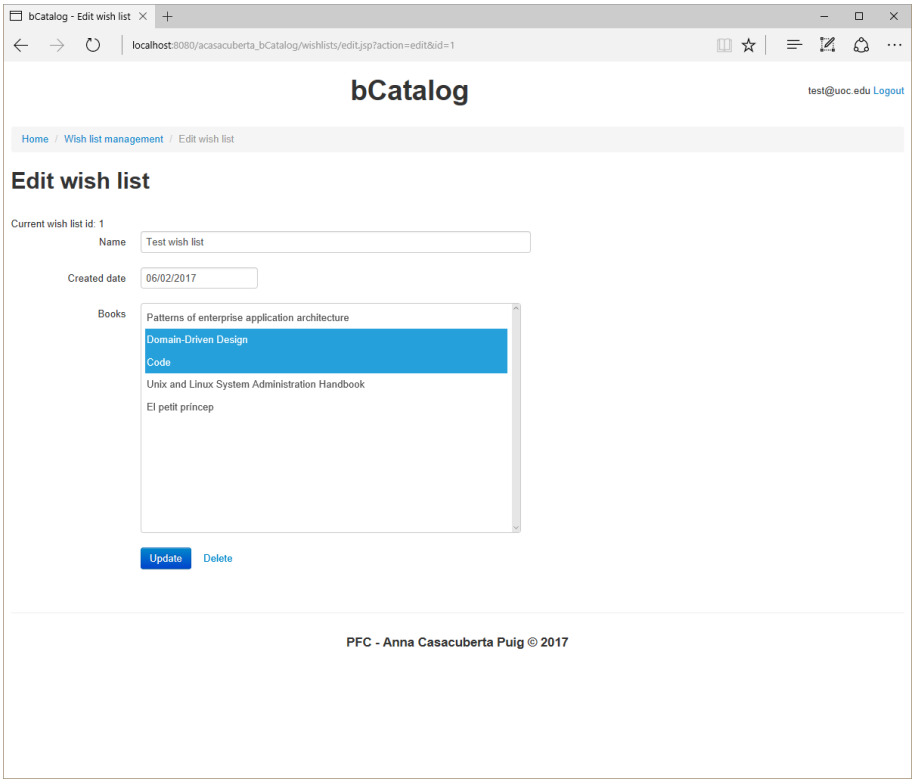
Es mostra la llista de llistes de desitjos (taula WishLists de la base de dades) amb les opcions crear registre nou, editar ja existent i eliminar ja existent.



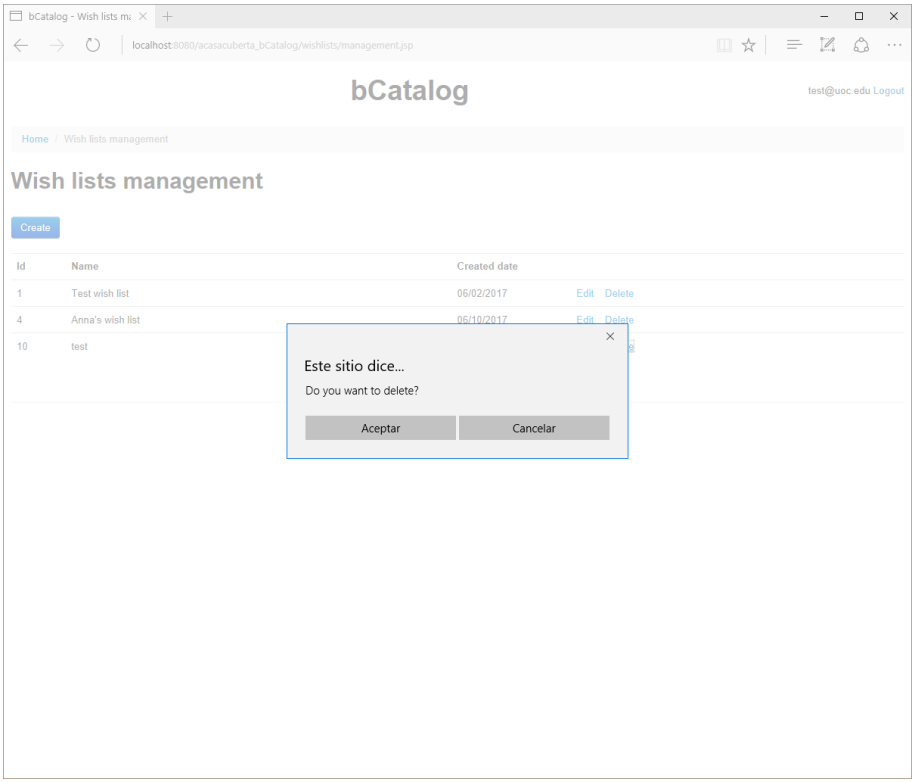
New wish list



Edit wish list



Delete wish list



Conclusions

El desenvolupament d'aquest projecte final de carrera m'ha permès aprofundir els meus coneixements de les tecnologies analitzaves i/o utilitzades. També he pogut refrescar molts conceptes estudiats durant la carrera i poder-los aplicar de nou.

Realitzar un projecte des de zero et permet posar a prova tots els coneixements apresos i saber quins són els que cal millorar. Durant la realització d'aquest projecte he hagut de buscar molta informació a la xarxa per tal de resoldre els dubtes / problemes que m'anaven sorgint.

El fet d'adquirir coneixements de noves tecnologies, les que no utilitzo habitualment a la feina, ha fet que es desperti una inquietud de coneixement (això penso que és una de les característiques de tot enginyer informàtic) i ganes de crear projectes amb aquestes tecnologies per tenir una visió molt més àmplia del mercat i les possibles solucions que podem trobar. Així com també les ganes de continuar estudiant i aprendre cada dia.

Durant tot el procés de creació del projecte, cal tenir molt clares quines són les fites i terminis de lliurament, que no sempre van ben coordinades amb la jornada laboral, per tal de poder assolir amb èxit els objectius del projecte. Això també m'ha permès ser molt més organitzada i planificar bé el temps del que disposo.

Hi ha aspectes i funcionalitats del projecte, que degut al poc temps que tenim per a fer-lo no s'han pogut implementar. Totes aquestes funcionalitat queden com a possibles millores que espero poder ampliar quan tingui una mica més de temps lliure.

Agraïments

Primer de tot, agrair al meus pares el seu suport i donar-los les gràcies per a donar-me l'oportunitat de poder ampliar els meus coneixements i la superació personal dia darrera dia.

A totes les persones del meu voltant, per entendre que el temps és molt valuós i que cal aprofitar-lo a cada moment. Ara ja puc dir, prepareu-vos per què em tornareu a aguantar. Ja disposo d'una mica més de temps lliure. 😊

I molt especialment, a tu Francesc. Per estar al meu costat sempre, per donar-me ànims durant els anys de carrera, per aguantar les èpoques d'estrès i suport moral. Torno a estar de nou amb tu.

Bibliografia

- Patrones de diseño en Java
  http://java.ciberaula.com/articulo/diseno_patrones_j2ee
- JPA Tutorial
  https://www.tutorialspoint.com/jpa/
- The Java EE 6 Tutorial
  http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html
- Java Data Objects JDO
  http://www.oracle.com/technetwork/java/index-jsp-135919.html
- Introduction to the Java Persistence API
  http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html
- Using the Java Persistence API
  http://www.objectdb.com/java/jpa/persistence
- Hibernate
  http://hibernate.org/
- Introduction to the Java 2 Platform, Enterprise Edition (J2EE)
  http://www.javaranch.com/journal/2002/10/J2EE.html
- Java EE at a Glance
  http://www.oracle.com/technetwork/java/javaee/overview/index.html
- J2EE (Java 2 Platform, Enterprise Edition)
  http://www.theserverside.com/definition/J2EE-Java-2-Platform-Enterprise-Edition
- Teorema CAP
  https://ca.wikipedia.org/wiki/Teorema_CAP
- PostreSQL
  https://ca.wikipedia.org/wiki/PostgreSQL
- Sistema de gestión de bases de datos
  https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos
- Persistence layer
  http://www.uml.org.cn/UMLApplication/pdf/persistenceLayer.pdf
- ¿Qué es un ORM?
  http://www.tuprogramacion.com/glosario/que-es-un-orm/
- ¿Por qué utilizar un ORM y una capa de abstracción?
  http://librosweb.es/libro/symfony_1_4/capitulo_8/por_que_utilizar_un_orm_y_una_capa_de_abstraccion.html
- What is an ORM?
  http://hibernate.org/orm/what-is-an-orm/

- Cayenne

https://cayenne.apache.org/

- MyBatis

http://www.mybatis.org/mybatis-3/

- Fusion Middleware Developer's Guide for Oracle TopLink

https://docs.oracle.com/cd/E17904_01/web.1111/b32441/undtl.htm#JITDG00004

- Catalog of Patterns of Enterprise Application Architecture

https://martinfowler.com/eaaCatalog/