# Easy Java Web Application Development

**Xavier Padró Sobrepera**
***ETIG***

**José Juan Rodríguez**

***18/06/2012***

# 2 Dedicatòria i agraïments

Aquest projecte el dedico especialment a la meva dona Anna, per haver aguantat durant tots aquests anys el fet de no poder dedicar-li el temps necessari i que es mereixia. Especialment els caps de setmana que m'havia de quedar tancat a casa realitzant pràctiques o estudiant pels exàmens.

Els meus agraïments també van destinats als consultors que he tingut durant la carrera, especialment els que es connectaven quasi a diari per resoldre tots els dubtes que apareixien al fòrum. Als estudiants amb qui he compartit aules i que han destinat part dels seus esforços en ajudar als companys.

Finalment agrair als meus companys de feina d'Everis pel seu suport durant aquesta etapa, especialment tenint en compte les característiques del projecte. Gràcies Manu pels teus consells a nivell d'arquitectura!. Gràcies Oscar per ajudar-me a recuperar els coneixements de C, el llibre em va treure de més d'un embolic!

# 3 Resum

El projecte que es descriu en aquest document es podria situar en la línia "Temes involucrats amb l'arquitectura J2EE en general" de l'àrea de J2EE del TFC, ja que consisteix en el desenvolupament d'una arquitectura de suport per al desenvolupament d'aplicacions web empresarials, basada en tecnologia J2EE. La idea és gestionar totes les funcionalitats de baix nivell per tal de que el desenvolupador de l'aplicació web pugui centrar-se quasi exclusivament en implementar els temes funcionals.

L'arquitectura desenvolupada és una arquitectura de 3 capes (presentació, negoci, dades), implementada principalment amb les tecnologies de JSF i Spring. A nivell de capes, l'arquitectura proporciona el següent suport:

- **Presentació**: Consisteix en una llibreria de components web (JSF) que simplifiquen la construcció de les vistes. Addicionalment, també es disposa d'una llibreria de funcions client (Javascript, Jquery) que proporcionen funcionalitats bàsiques que es poden fer en client minimitzant les crides a servidor.

- **Negoci**: Conté les estructures i mecanismes de suport (Spring) per tal de que el desenvolupador pugui construir els seus controladors centrant-se en escriure les funcionalitats de negoci, oblidant-se dels aspectes de baix nivell com són la gestió dels literals, traçabilitat o gestió d'errors. També conté la infraestructura de navegació entre les diferents vistes o estats de l'aplicació.

- **Dades**: En la darrera capa, hi ha la infraestructura necessària per invocar serveis de negoci (SCA), els quals accediran a la base de dades i recuperaran la informació que enviaran a les capes superiors de l'arquitectura.

Paraules clau:

| | | | |
|---|---|---|---|
| J2EE | Arquitectura | Suport | Web |
| Spring | SOA | SCA | Aplicacions |
| Infraestructura | Tecnologia | JSF | Hibernate |

# 4 Índex de continguts

# 5 Cos de la memòria

## *5.1 Capítol 1. Introducció*

### 5.1.1 Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC

El motiu d'haver triat aquest tipus de projecte és pel fet de que estic treballant com a programador en l'àmbit d'arquitectura J2EE i em servirà d'experiència per a futurs projectes. Naturalment, en projectes de certa grandària cada persona és assignada a una àrea acotada pel que aquest projecte em permetrà tenir una visió més global de totes les parts que formen una arquitectura J2EE.

En el context de la carrera d'Enginyeria Tècnica d'Informàtica de Gestió, i donat que es tracta de la creació d'un projecte d'arquitectura J2EE, principalment les següents assignatures permetran aplicar els coneixements adquirits:

- Enginyeria del programari
- Estructura de la informació
- Fonaments de programació
- Programació orientada a l'objecte
- Tècniques de desenvolupament del programari
- Gestió org. i projectes inf
- Bases de dades

El context del projecte es basa en la simulació d'un client el qual té instal·lades en el seu sistema, un seguit d'aplicacions web o d'escriptori. El client vol aprofitar la necessitat de la creació de noves aplicacions per a implantar un sistema que integri tant les noves aplicacions com les antigues mitjançant tecnologies modernes basades en entorn web. Per aquesta raó, el TFC consisteix en la creació d'una arquitectura J2EE que serà la infraestructura necessària per a la creació d'aplicacions i la construcció d'una aplicació web basada en aquesta arquitectura.

En una fase posterior fora de l'abast del TFC i, un cop creada l'arquitectura, el client podrà començar el procés de construcció de noves aplicacions i el procés de migració de les antigues.

### 5.1.2 Objectius del TFC

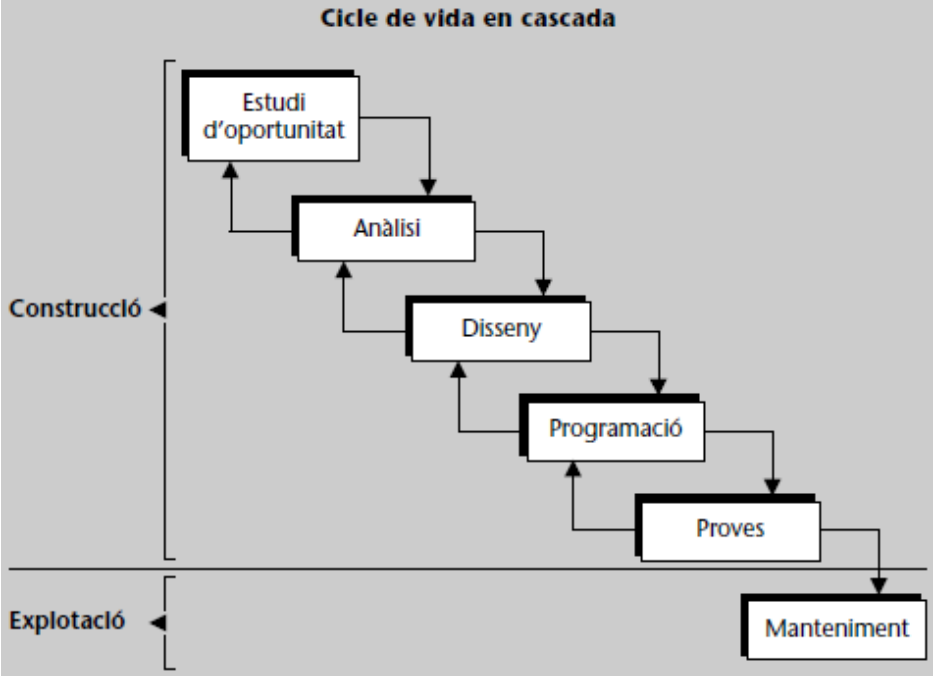
El principal objectiu de l'arquitectura, anomenada EJWAD (Easy Java Web Application Development) és el de facilitar el desenvolupament d'aplicacions web. Més concretament, EJWAD permetrà:

- Disminuir el temps de construcció: Això s'aconsegueix evitant haver de configurar temes com la traçabilitat o la integració amb la capa de serveis.
- Concentrar-se en temes funcionals: El desenvolupador podrà dedicar-se a fer el que li han demanat, oblidant-se de gestionar temes de baix nivell.
- Construir vistes més fàcilment: La llibreria de components visuals facilitada per EJWAD oferirà components que implementaran funcionalitats més complexes pintant simplement un tag.
- Homogeneïtat en les aplicacions: Al proporcionar uns mateixos estils pels components visuals, s'aconsegueix que totes les aplicacions d'un sistema tinguin la mateixa aparença.

### 5.1.3 Enfocament i mètode seguit

Aquest projecte es basa en un enfocament tecnològic de les aplicacions web en el món empresarial. Més que aportar solucions funcionals als clients, es basa en buscar les solucions tecnològiques més adients en cada situació i proporcionar una infraestructura sòlida, mantenible, reutilitzable i escalable.

Donada la estructuració de l'assignatura del TFC, aquest projecte es desenvoluparà en el sistema de cicle de vida en cascada, en el qual cada etapa té un inici i un final, produint com a resultat un conjunt d'entregables. Això implica que la següent etapa no pot començar si no ha finalitzat l'anterior.



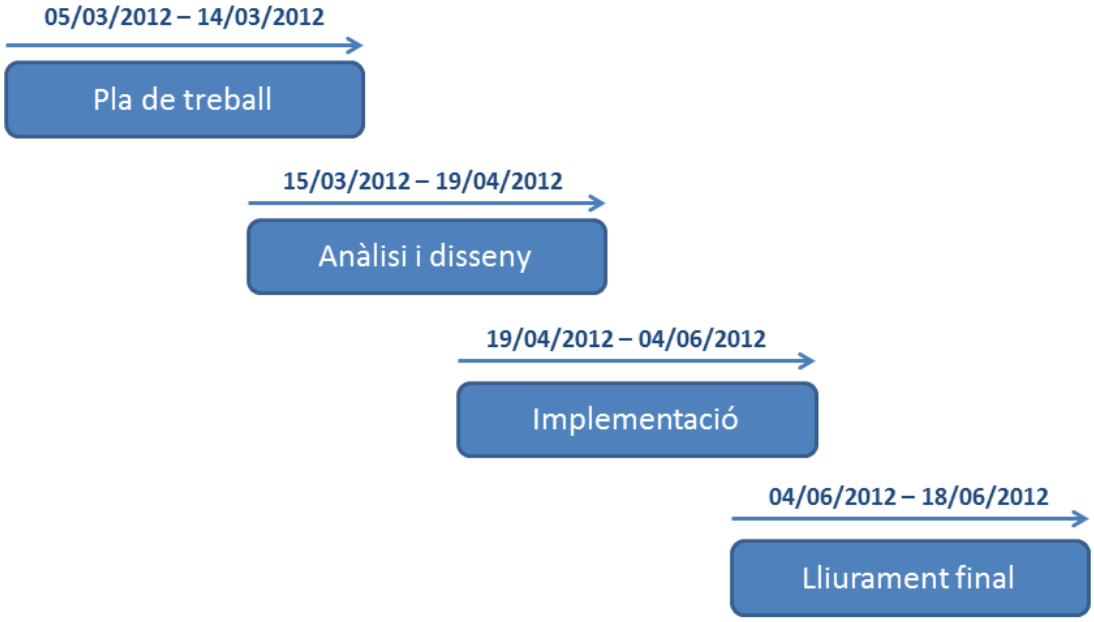

Degut també a què es tracta d'un treball individual, i a les característiques del projecte, no s'han tingut en compte les fases de presa de requeriments ni manteniment.

La fase prèvia a l'inici d'aquest cicle de vida, ha consistit en els següents aspectes:

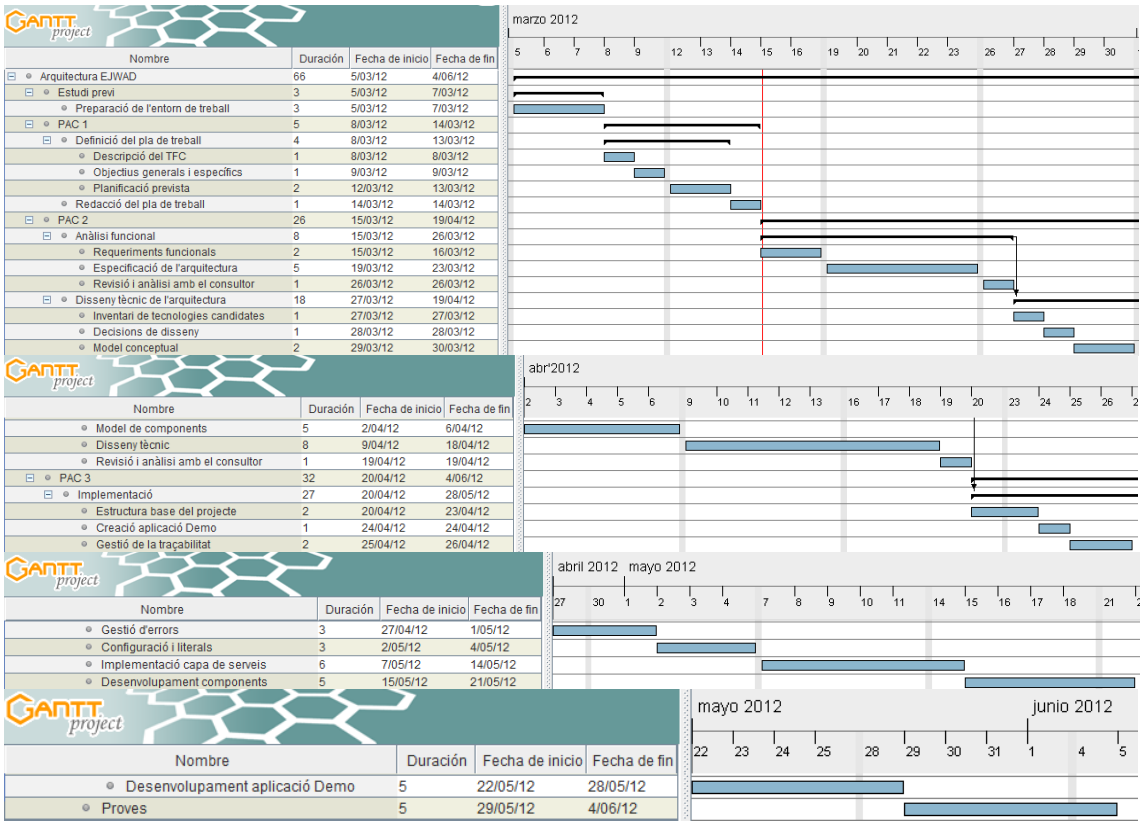
- Estudi de les tecnologies candidates a ser utilitzades en el desenvolupament de l'arquitectura.
- Procés de filtre i tria de les tecnologies a utilitzar.
- Instal·lació i preparació de l'entorn de desenvolupament

### 5.1.4 Planificació del projecte

La planificació del projecte es dividirà en diverses etapes, coincidint amb les dates d'entrega de l'avaluació contínua de l'assignatura del TFC.




Les següents imatges mostren la planificació del projecte mitjançant un diagrama de Gantt.



### 5.1.5 Productes obtinguts

- **Anàlisi funcional**: Requeriments del client sobre la nova arquitectura. Aquest document ha estat inclòs dins la memòria (apartat 5.2).
- **Disseny tècnic**: Descripció tècnica de l'arquitectura. Aquest document ha estat inclòs dins la memòria (apartat 5.3).
- **Guia d'instal·lació**: Passos necessaris per a instal·lar l'arquitectura i l'aplicació de demostració en l'equip local. Format MS Word.
- **Paquet d'instal·lació**: Llibreries i fitxers necessaris per a poder executar l'aplicació de demostració i el servidor de serveis SCA. Format comprimit .zip.
- **Presentació**: Síntesi de la memòria en format de diapositives. Format MS Powerpoint.

### 5.1.6 Breu descripció dels altres capítols de la memòria

La present memòria es troba dividida en els següents capítols:

**Capítol 2. Anàlisi funcional del projecte**

Explicació dels requeriments de l'arquitectura i especificació funcional de l'aplicació de demostració des del punt de vista del client.

**Capítol 3. Disseny tècnic del projecte**

Detall de l'estructura de l'arquitectura, incloent decisions de disseny i el model conceptual. Addicionalment inclou el disseny de la base de dades i una explicació bàsica sobre el funcionament de la capa de serveis.

**Capítol 4. Avantatges funcionals de l'arquitectura**

Explicació detallada del funcionament de cadascuna de les principals funcionalitats que ofereix l'arquitectura, indicant-ne la manera d'utilitzar-les.

**Capítol 5. Navegació**

Aquest capítol inclou una breu introducció a la tecnologia Spring Webflow explicant l'estructura bàsica d'una navegació aplicativa.

**Capítol 6. Aplicació de demostració**

Descripció de l'estructura de l'aplicació i mostra de les captures de pantalla,

**Capítol 7. Conclusions**

Conclusions extretes de la realització d'aquest projecte

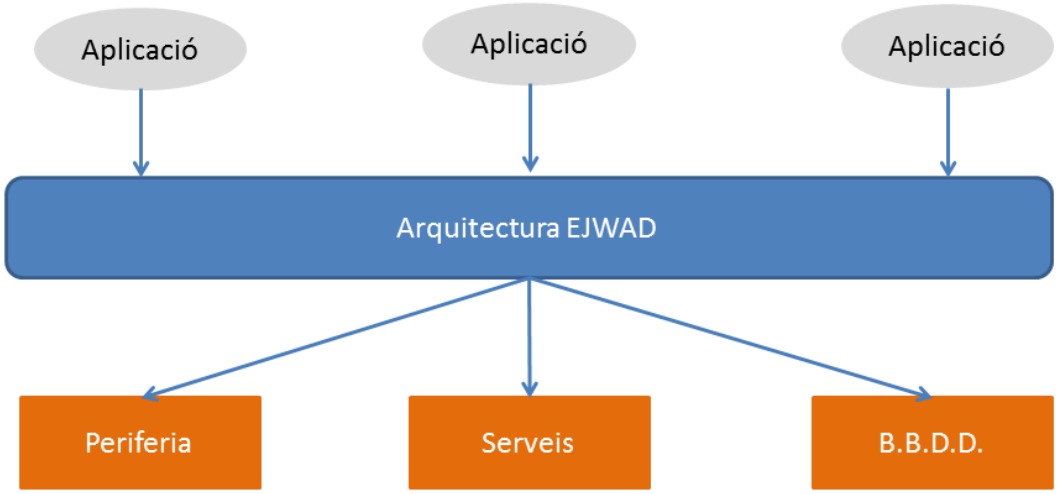
## *5.2 Capítol 2. Anàlisi funcional del projecte*

### 5.2.1 Context del projecte

Actualment, el sistema instal·lat a les oficines del client conté un conjunt d'aplicacions construïdes en diferents tecnologies (Visual Bàsic, aplicacions Web…). Aprofitant que ha sorgit la necessitat de construir noves aplicacions, es vol crear una base tecnològica unificada sobre la qual totes les aplicacions seran construïdes de la mateixa manera, facilitant el seu desenvolupament i l'adaptació dels usuaris, ja que un cop acostumats a la nova interfície, aquesta serà compartida per les altres aplicacions. Progressivament, es procedirà a la migració de les antigues aplicacions a la nova arquitectura.

Tenint en compte el cost tant del desenvolupament de les noves aplicacions com de la migració de les antigues, aquest projecte neix de la necessitat de reduir el màxim possible el temps de construcció i el cost en recursos que tindran. Per aquesta raó, es planteja la construcció d'una arquitectura sobre la qual es basaran totes les aplicacions. Aquesta arquitectura haurà de gestionar tots els temes de baix nivell per tal de que els desenvolupadors només s'hagin de preocupar dels aspectes funcionals de l'aplicació en construcció.

Com a prova pilot, es demana la construcció d'una aplicació que farà les funcions d'entrada principal a la xarxa d'aplicacions del sistema. Bàsicament consistirà en una part de noticies i una altra part que contindrà un cercador de persones. L'objectiu principal d'aquesta aplicació és provar l'eficàcia de la nova arquitectura. De cara a fases posteriors, es provarà la integració amb les altres aplicacions del sistema.




### 5.2.2 Especificació de l'arquitectura

En aquesta primera fase, la nova arquitectura haurà de proporcionar un seguit de funcionalitats que es descriuen en els següents subapartats:

- Accés a la capa de serveis
- Configuració genèrica
- Gestió de l'idioma
- Traçabilitat
- Llibreria de components visuals
- Seguretat
- Gestió d'errors

Totes les aplicacions que es construeixin sobre aquesta arquitectura gaudiran de totes aquestes funcionalitats de manera automatitzada. L'objectiu és que simplement incloent les llibreries de l'arquitectura dins l'aplicació, aquesta delegui la gestió d'aquestes funcionalitats, simplificant de manera significativa l'estructura de l'aplicació.

De cara a properes fases es consideraran altres aspectes no contemplats en aquesta fase com per exemple:

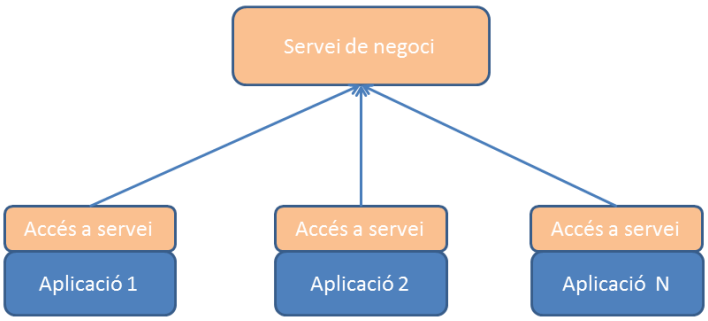
- Monitorització d'aplicacions: Es contempla la construcció d'una aplicació que monitoritzi les dades del servidor.
- Implementació de nous serveis: Per a les noves aplicacions que es construeixin o per a la migració de les antigues aplicacions.
- Ampliació de la llibreria de components: Components addicionals, plantilles…
- Gestió de la perifèria: Connexió amb els diferents components.

#### 5.2.2.1 Accés a la capa de serveis

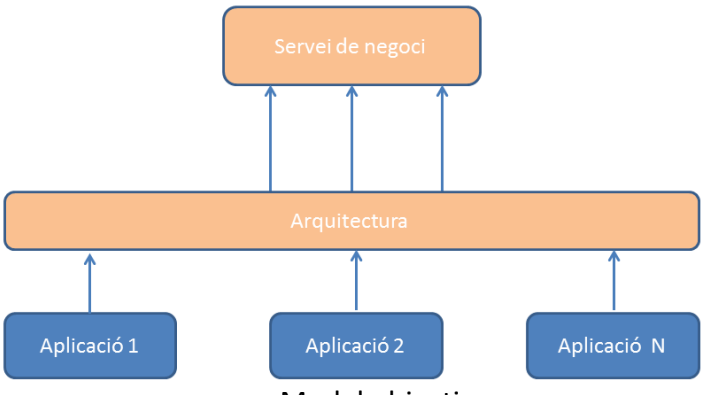
En el sistema existeixen varis serveis que són accedits per diferents aplicacions, com per exemple, el cercador de persones. Cada aplicació té integrada la configuració per la crida de cada servei, fet que complica la construcció de noves aplicacions i perjudica la reusabilitat de codi. Un canvi en l'accés al servei provocarà haver de canviar la configuració i el codi de totes les aplicacions que l'invoquin.

L'arquitectura s'haurà d'encarregar de gestionar la crida a cada servei a partir d'unes dades informades des de l'aplicació (nom del servei, operació a invocar…).

La nova arquitectura haurà de gestionar les crides a serveis reutilitzables, amagant la implementació dels mateixos de les aplicacions i deslligant-los de la capa aplicativa.




Model actual




Model objectiu

#### 5.2.2.2 Configuració genèrica

Cada aplicació té la seva pròpia configuració, quan alguns paràmetres són comuns per totes. Es vol poder canviar algun d'aquests paràmetres comuns sense haver de modificar i tornar a instal·lar l'aplicació al servidor d'aplicacions. També es desitja poder modificar paràmetres de configuració propis de l'aplicació sense necessitat de tocar l'aplicació.

#### 5.2.2.3 Gestió de l'idioma

Totes les aplicacions tindran una zona superior (menú contextual) des d'on es podrà canviar l'idioma de l'usuari. En seleccionar un altre idioma la pantalla es refrescarà mostrant la pàgina principal de l'aplicació amb el nou idioma.

Addicionalment, aquesta zona superior tindrà una part configurable on l'aplicació hi podrà incloure navegacions a altres pàgines de l'aplicació o tal i com es desitja de cara a fases posteriors del projecte, incorporar els accessos a altres aplicacions del sistema.

#### 5.2.2.4 Traçabilitat

De cara a la gestió d'incidències i a la ràpida detecció d'errors, es vol unificar la traçabilitat de totes les aplicacions, oferint un únic punt (carpeta de logs) on hi haurà tots els fitxers de traces, separats per aplicació i gestionant també els logs d'arquitectura.

#### 5.2.2.5 Llibreria de components visuals

En aquesta primera fase es demana que l'arquitectura ofereixi una llibreria amb els components necessaris per a construir les pantalles de l'aplicació pilot. Independentment dels components no visuals que es creguin necessaris per a estructurar les pantalles, es requereix els següents components:

- Taula de resultats
- Botó de navegació
- Enllaç de navegació
- Cercador
- Títol secundari
- Calendari
- Caixa de noticies

#### 5.2.2.6 Seguretat

Es desitja un control d'accés a l'aplicació, el qual en aquesta fase inicial constarà de dos nivells:

- Empleat: Accés bàsic a l'aplicació
- Directiu: Accés a funcionalitats addicionals

Aquest control de seguretat consultarà una base de dades on el client hi té emmagatzemada tota la informació dels usuaris. Per tant, l'accés a l'aplicació es realitzarà mitjançant una pantalla de login. Els usuaris que no pertanyin al sistema no podran accedir a l'aplicació.

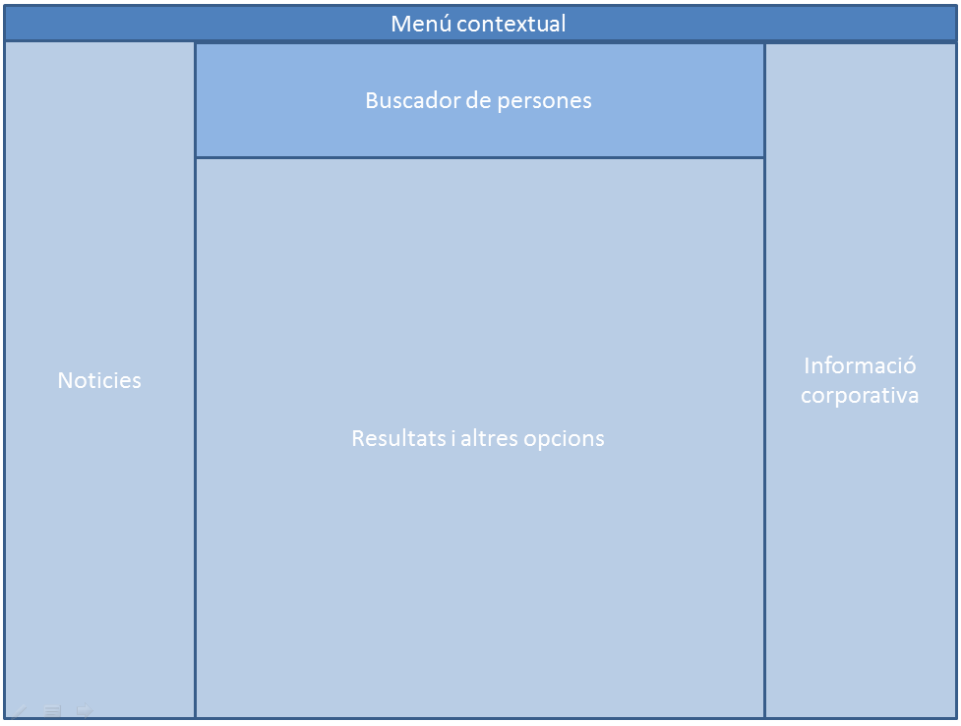
#### 5.2.2.7 Gestió d'errors

Tots els errors que puguin ocórrer en l'aplicació s'han de mostrar de la següent manera:

- **Errors controlats**: Mostrant un popup informatiu
- **Errors no controlats**: S'ha de mostrar una pantalla d'error comuna on es mostri una descripció de l'error i un codi d'error que anirà associat a la traça generada en el fitxer de lòg per tal de que els tècnics puguin detectar la causa ràpidament.

### 5.2.3 Aplicació pilot: Cercador de persones

Es tracta d'una aplicació senzilla que tindrà una pantalla principal la qual es dividirà en 3 parts, ja que el menú contextual és un component d'arquitectura que es mostrarà a totes les aplicacions. Aquestes tres parts són: apartat de notícies, cercador de persones i secció d'informació corporativa.



### 5.2.3.1 Actors

**Usuari EMPLEAT**

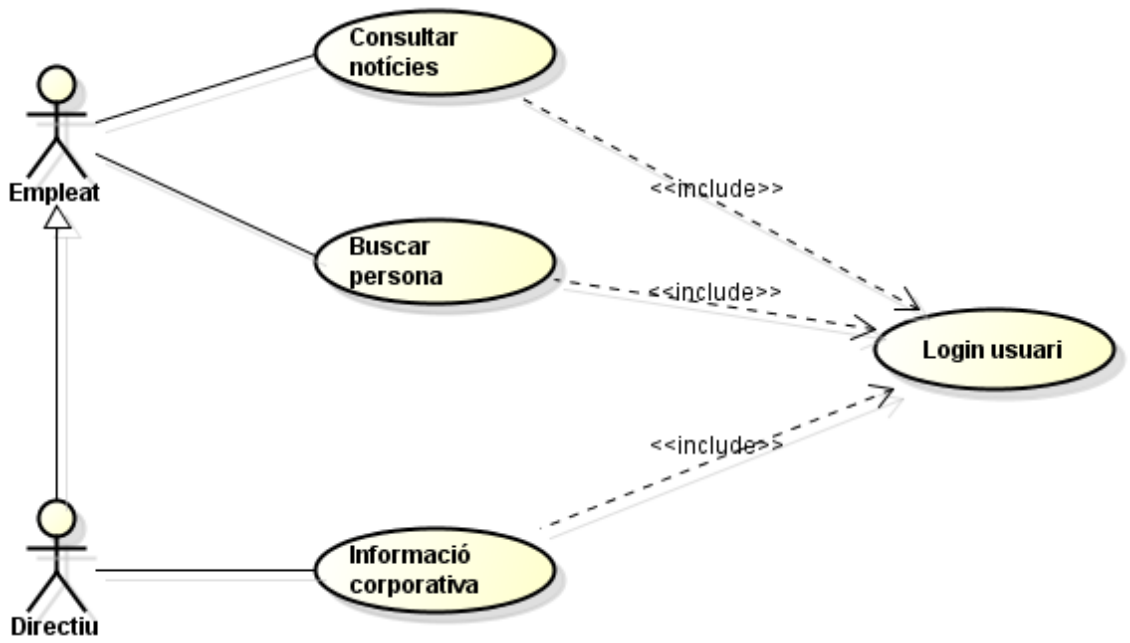
És el perfil base de les aplicacions. Té accés a l'aplicació i pot gaudir dels serveis de consulta de notícies i recerca de persones.

**Usuari DIRECTIU**

Perfil amb opcions avançades. Té accés a les funcions del perfil EMPLEAT i també al servei d'informació corporativa.

#### *5.2.3.1.1 Casos d'ús*

L'aplicació contempla uns pocs casos d'ús ja que l'objectiu d'aquest projecte no és construir una aplicació completa, sinó provar les funcionalitats de l'arquitectura construïda i validar que totes les parts estiguin ben integrades.



#### *5.2.3.1.2 Descripció textual dels casos d'ús*

| LOGIN USUARI | |
|---|---|
| **Descripció** | L'usuari s'identifica amb el sistema |
| **Actor** | Empleat, Directiu |
| **PreCondició** | L'usuari no està identificat al sistema |
| **PostCondició** | L'usuari es troba connectat al sistema |
| **Alternatiu** | L'usuari introdueix dades invàlides. Es mostra un missatge d'error i l'usuari haurà de tornar a introduir les dades |

| CONSULTAR NOTICIES | |
|---|---|
| **Descripció** | L'usuari consulta una llista amb les últimes notícies de la companyia |
| **Actor** | Empleat, Directiu |
| **PreCondició** | L'usuari s'ha d'haver identificat amb el sistema |
| **PostCondició** | Es mostra una llista de les notícies |
| **Alternatiu** | No hi ha notícies disponibles. Es mostra un missatge informatiu |

| BUSCAR PERSONA | |
|---|---|
| **Descripció** | L'usuari introdueix uns camps sobre els que buscarà altres usuaris |
| **Actor** | Empleat, Directiu |
| **PreCondició** | L'usuari s'ha d'haver identificat amb el sistema |
| **PostCondició** | Es mostra una llista amb els usuaris trobats |
| **Alternatiu** | La consulta no ha retornat resultats. Es mostra un missatge informatiu |

| INFORMACIÓ CORPORATIVA | |
|---|---|
| **Descripció** | Es mostra informació de la companyia |
| **Actor** | Directiu |
| **PreCondició** | L'usuari s'ha d'haver identificat amb el sistema i tenir perfil directiu |
| **PostCondició** | Es mostra la informació retornada pel servei |
| **Alternatiu** | La consulta no ha retornat resultats. Es mostra un missatge informatiu |

## 5.2.3.2 Parts de l'aplicació

### *5.2.3.2.1 Menú contextual*

Ha de contenir l'opció del canvi d'idioma. Aquesta opció provocarà la recarrega de l'aplicació amb l'idioma seleccionat.

Properes fases: Fora de l'àmbit d'aquesta primera fase, el menú contextual utilitzarà l'espai per contenir enllaços a les diferents aplicacions del sistema.

### *5.2.3.2.2 Notícies*

Aquest apartat contindrà les principals notícies que seran extretes mitjançant la invocació a un servei de negoci. Cada notícia es mostrarà en un format de dues columnes, a l'esquerra la foto de la noticia i a la dreta la descripció.

**Títol de la noticia**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed vehicula, mauris ornare gravida hendrerit, felis metus eleifend.

### *5.2.3.2.3 Cercador de persones*

Aquesta regió consisteix en un conjunt de camps que actuaran de filtres per a buscar una persona pertanyent al sistema. Un cop seleccionats els criteris s'invocarà al servei de negoci per a recuperar les persones sota el criteri seleccionat. Aquestes persones es mostraran en la zona de resultats.

En aquesta zona de resultats es podrà seleccionar un dels resultats per mostrar-ne el detall.

### *5.2.3.2.4 Informació corporativa*

Aquesta secció només serà visible pels usuaris identificats amb el perfil DIRECTIU.

### 5.2.4 Serveis a implementar

De cara a l'aplicació pilot, s'han identificat dos serveis necessaris per a mostrar tota la informació de la pàgina principal:

- Cercador de persones
- Recuperació de notícies
- Informació corporativa

Aquests dos serveis accediran a la base de dades per recuperar la informació demanada.

## *5.3 Capítol 3. Disseny tècnic del projecte*

### 5.3.1 Decisions de disseny

L'arquitectura es construirà utilitzant el model de 3 capes (presentació, negoci, dades). A continuació es mostren les decisions de disseny de cada capa:

#### 5.3.1.1 Capa de presentació

S'ha escollit el patró MVC (model, vista, controlador) per a desacoblar el disseny de les vistes del negoci. D'aquesta manera si es desitja canviar la implementació de la lògica de negoci es podrà fer sense afectar massa la capa de vista. Una altra avantatja és que els components de negoci es poden reutilitzar en diferents interfícies de presentació.

**Vista:**

| Tecnologia | Característiques | |
|---|---|---|
| **JSP** | Estàndard tradicional per aplicacions web | |
| | Llibreries de components | |
| | Tecnologia desfasada (JEE 1.4) | |
| | | |
| **JSF** | Estàndard actual per aplicacions web (JEE 5/6) | |
| | Model basat en components independents | |
| | Model extensible. Permet crear components personalitzats que poden utilitzar tecnologies client com CSS, JQuery… | |
| | Varietat de llibreries de components basades en l'estàndard JSF (Primefaces, MyFaces, Richfaces…) | |
| | Permet combinar diferents llibreries | |
| | Complexitat | |
| | | |
| **HTML/JQuery/CSS** | Permet combinar diferents llibreries de components | |
| | Poc suport d'arquitectura a l'utilitzar tecnologies client | |
| | Mantenibilitat de les aplicacions | |

La tecnologia escollida és l'estàndard JSF en la versió 2.1.7. Com a implementació s'utilitzarà la llibreria de components Primefaces en la versió 3.1.1.

**Controlador:**

| Tecnologia | Característiques | |
|---|---|---|
| **JSF** | Gestió de POJOs basat en el model d'inversió de control | |
| | Lògica inclosa als POJOs, reduint les dependències amb implementacions propietàries | |
| | El model de navegació ofereix poca funcionalitat | |
| | | |
| **Spring Webflow** | Gestió de POJOs basat en el model d'inversió de control | |
| | Integració amb JSF | |
| | Lògica inclosa als POJOs, reduint les dependències amb implementacions propietàries | |
| | Model de navegació avançat (subfluxes, pas de paràmetres, resolució de vistes dinàmica…) | |
| | | |
| **Struts** | Orientació al patró Command (request-response) | |
| | Utilitzat en el desenvolupament tradicional d'aplicacions web | |
| | Dependències amb el propi framework al no estar basat en un estàndard genèric. | |

La tecnologia escollida és Spring Webflow en la versió 2.3.1

**Model:**

| Tecnologia | Característiques | |
|---|---|---|
| **Spring** | Flexibilitat. Anotacions, configuració XML o configuració Java | |
| | Abstracció JDBC | |
| | Moltes de les anotacions clau de JEE estan incloses a Spring | |
| | Es pot executar en servidors web (Tomcat) | |
| | Necessitat d'incloure les llibreries amb l'aplicació | |
| | | |
| **JEE** | És l'estàndard | |
| | No necessita afegir res, ja ve inclòs en el servidor d'aplicacions | |
| | Necessitat d'utilitzar un servidor d'aplicacions | |

La tecnologia escollida és Spring.

Funcionalitat

Fortalesa

Debilitat

#### 5.3.1.2 Capa de negoci

Per a la capa de negoci s'utilitzarà la tecnologia Spring, ja que serà l'utilitzada en les altres capes de l'arquitectura (presentació, dades).

#### 5.3.1.3 Capa de dades

Tecnologies: Hibernate + SCA + Spring.

Per al desenvolupament de serveis de negoci s'utilitza la tecnologia SCA amb el suport de Hibernate per a la persistència.

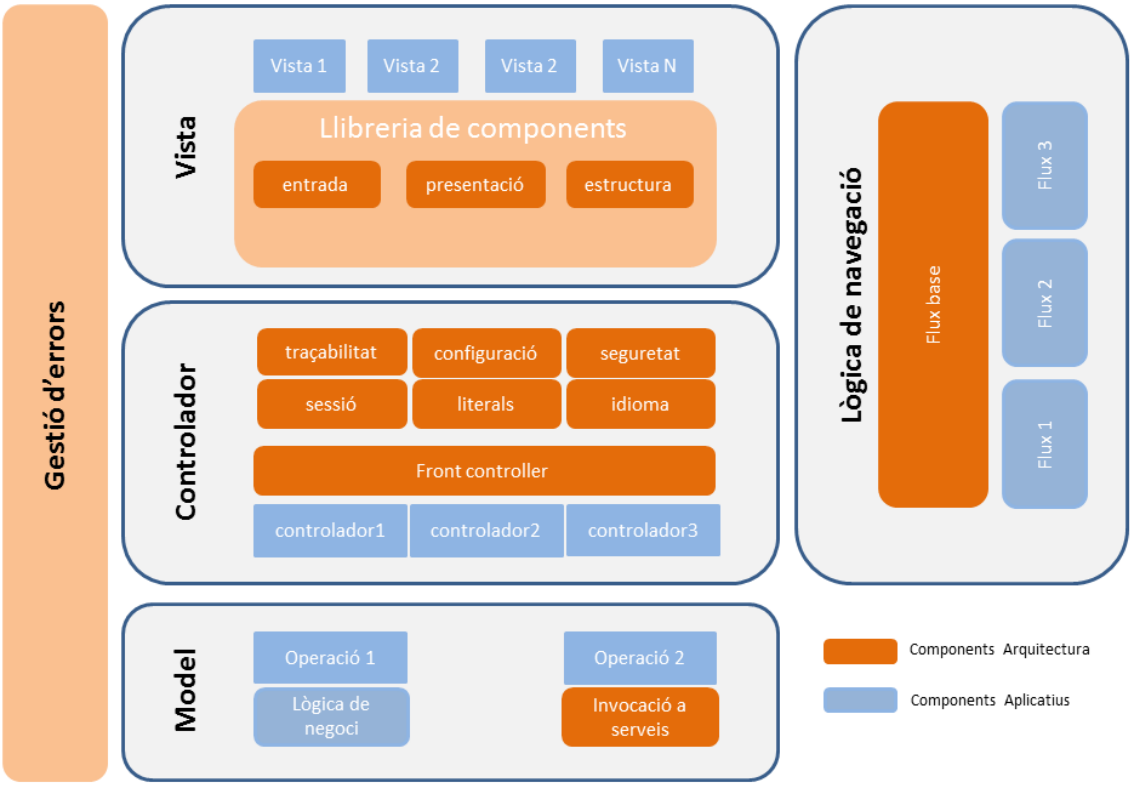
Service Component Architecture (SCA): Tecnologia de software que proveeix un model per desenvolupar aplicacions que segueixen els principis de SOA (service oriented architecture). Es tracta d'una especificació que té diverses implementacions. Les avantatges que ofereix, entre d'altres, són:

- Desacobla la lògica de negoci dels detalls de les crides que invoquen els seus serveis
- Permet cridar serveis implementats en diferents llenguatges (Java, C, …)
- Permet invocar serveis amb diferents mètodes d'accés (RMI, webservices, JMS…)

L'arquitectura utilitzarà la implementació SCA de Apache Tuscany. Els serveis es desenvoluparan en llenguatge Java i es podran invocar com a webservices.

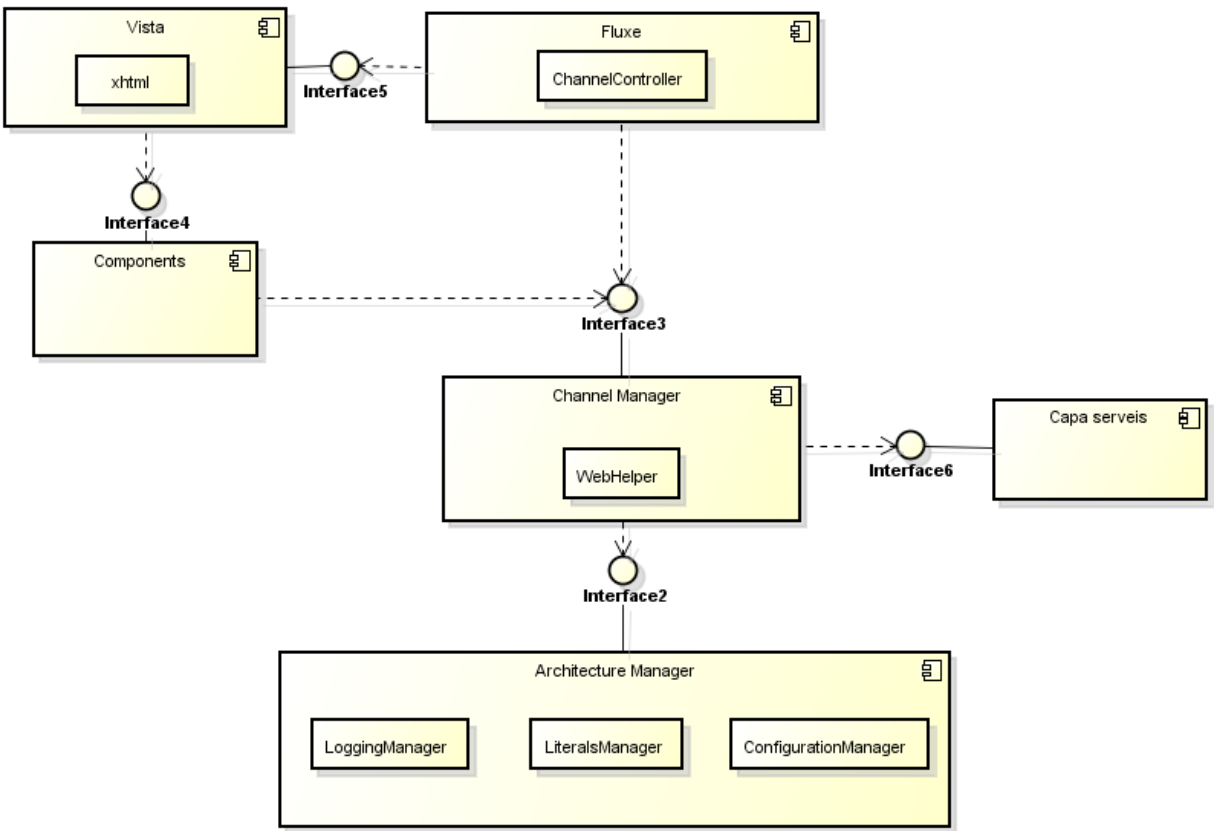
### 5.3.2 Model conceptual

El següent diagrama mostra el disseny conceptual de l'arquitectura que es vol construir:



### 5.3.3 Model de components

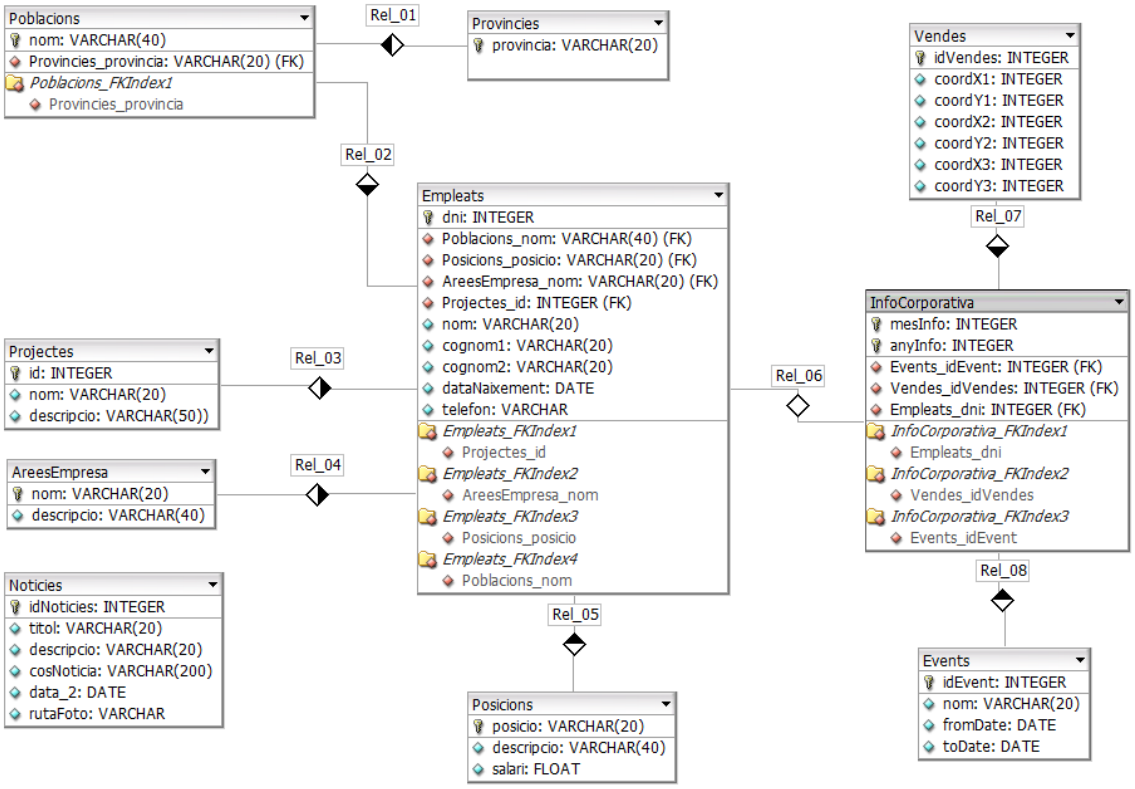
El següent diagrama mostra de manera simplificada quin serà el model de components de l'arquitectura.



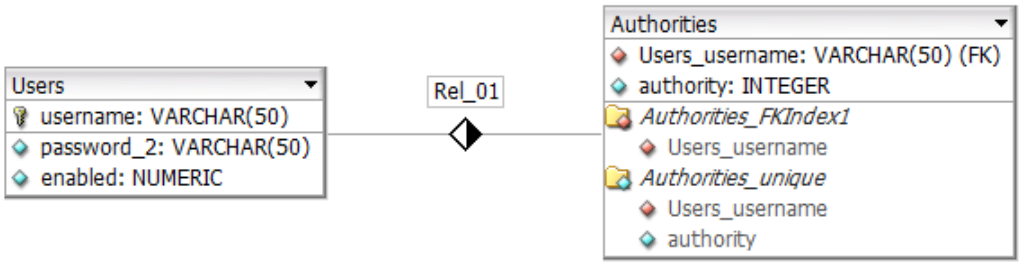
### 5.3.4 Disseny de la base de dades

El següent diagrama descriu les taules que seran necessàries en aquesta fase inicial del projecte per complir amb les funcionalitats demanades. Aquestes taules donaran la informació sol·licitada pels següents serveis:

- Cercador de persones. Taules: Empleats, Poblacions, Províncies, Projectes, AreesEmpresa, Posicions.
- Recuperació de notícies. Taules: Noticies
- Informació Corporativa. Taules: InfoCorporativa, Vendes, Events.

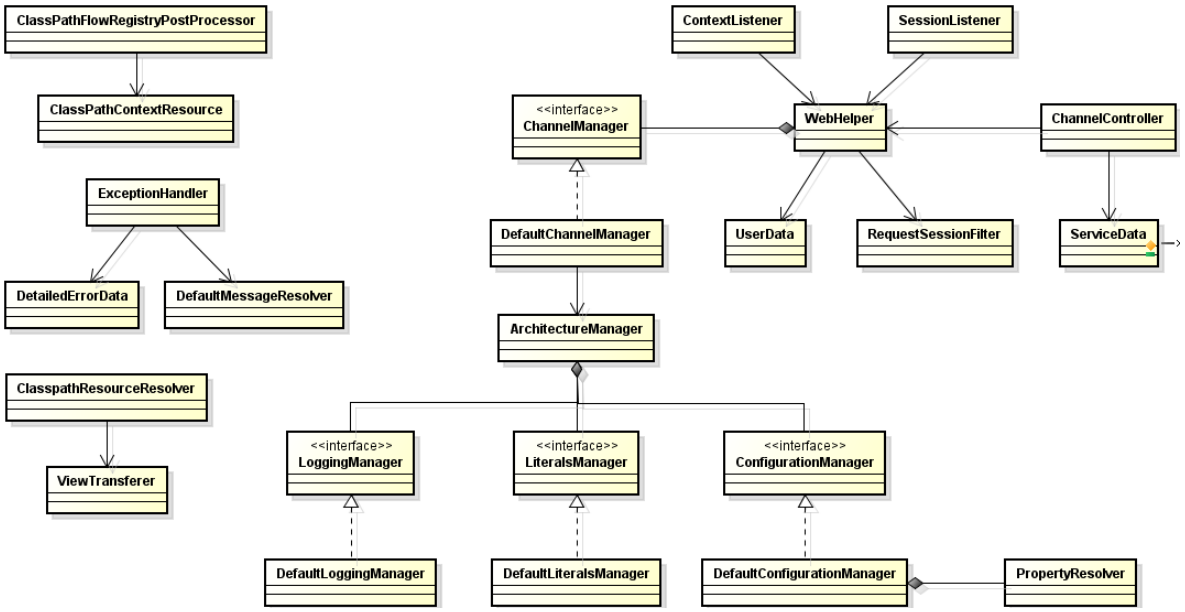



Les següents taules també seran necessàries per gestionar la identificació d'usuaris (seguretat de les aplicacions):

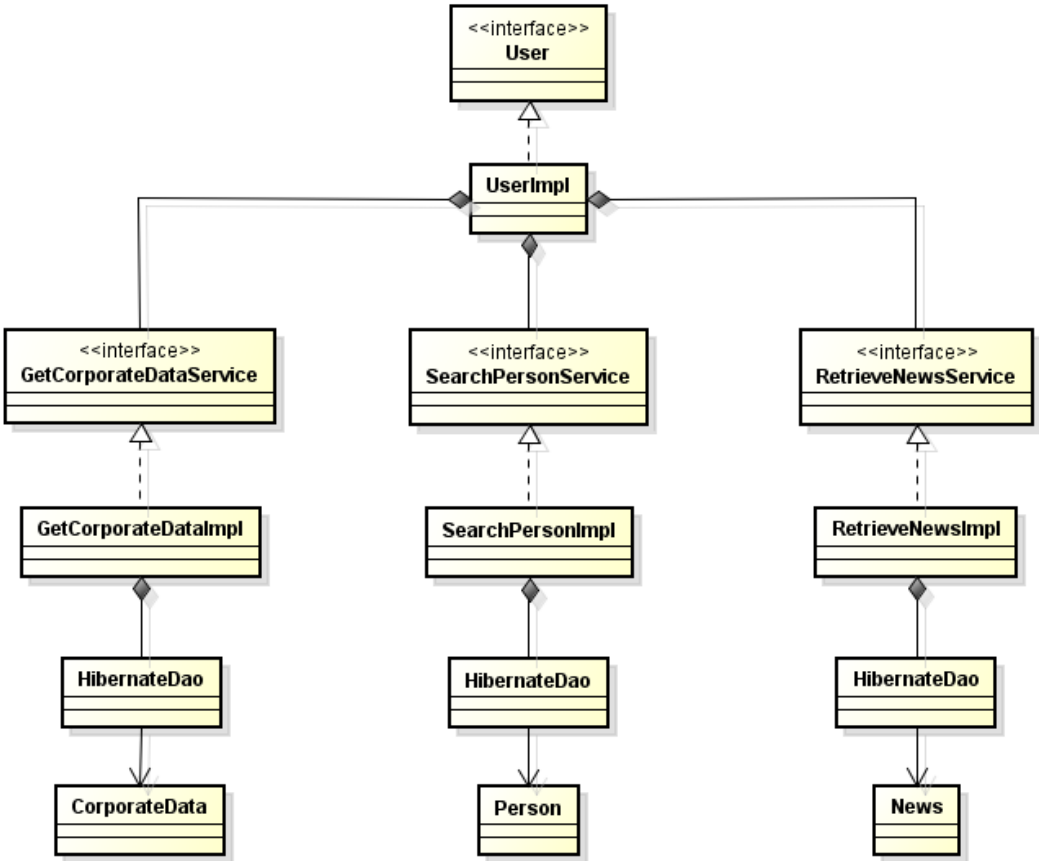


### 5.3.5 Diagrama de classes

Les següents classes formen part del nucli de l'arquitectura que es vol construir:



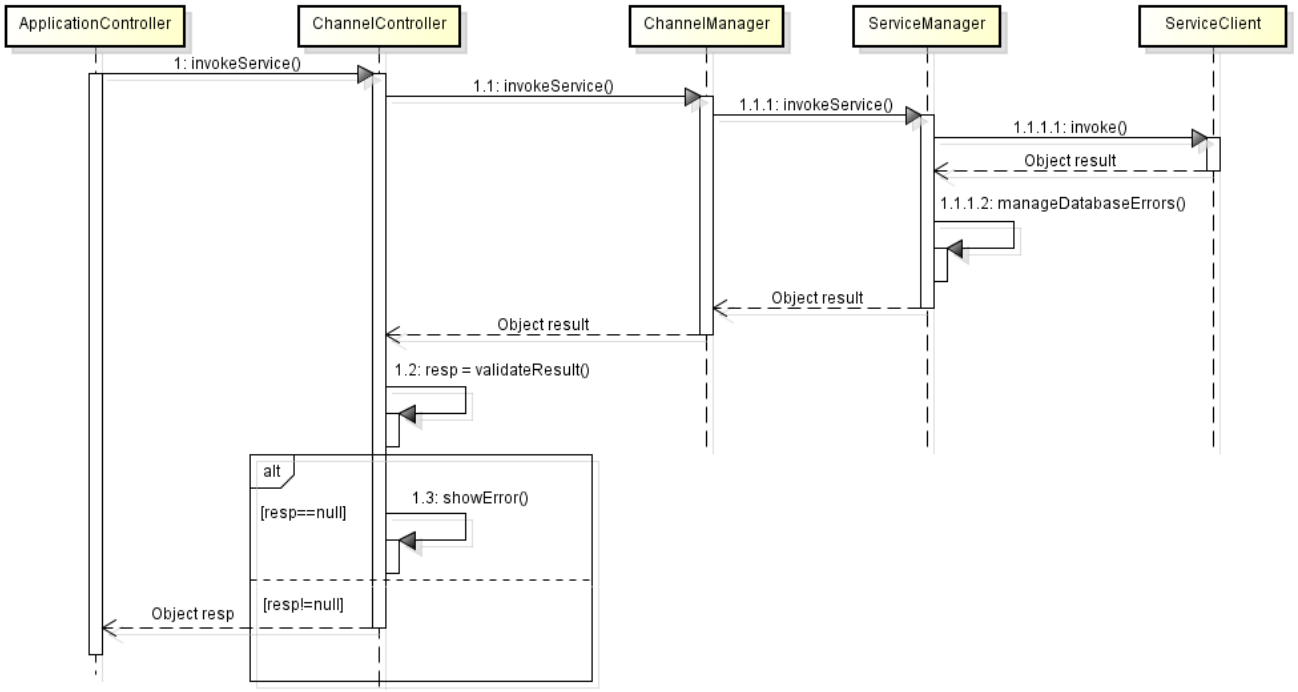

De cara a la construcció dels serveis que s'utilitzaran en la prova pilot, s'implementarà les següents classes:



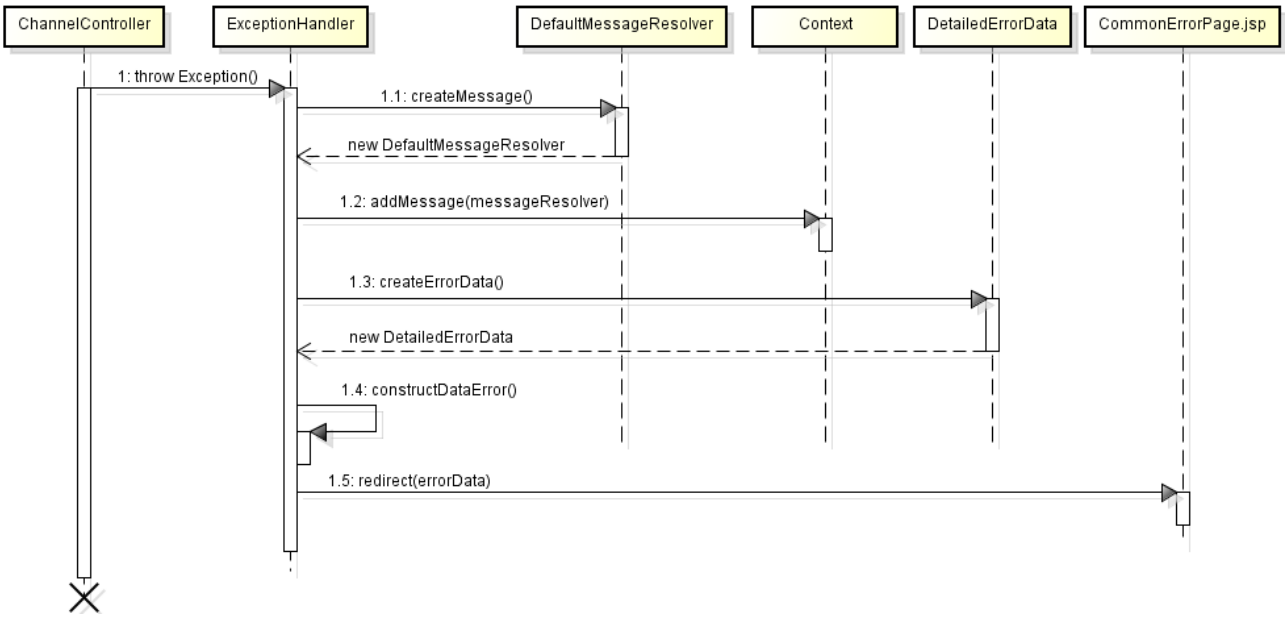
### 5.3.6 Funcionalitats de l'arquitectura

En aquest apartat es descriuen els fluxes d'algunes de les funcionalitats base de l'arquitectura.

#### 5.3.6.1 Accés a serveis

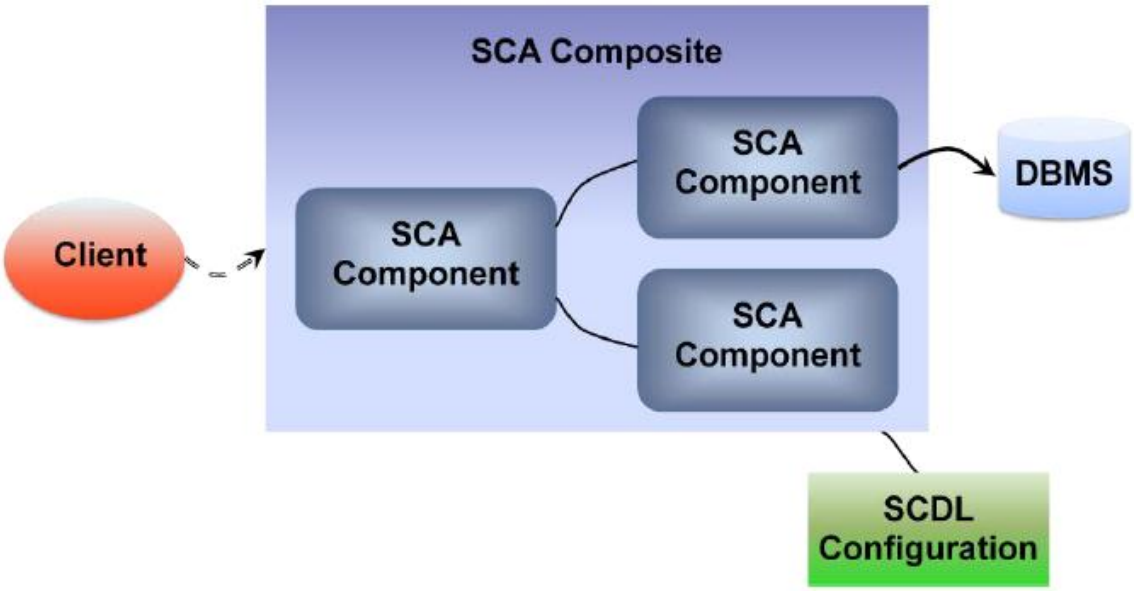



#### 5.3.6.2 Gestió d'errors no controlats



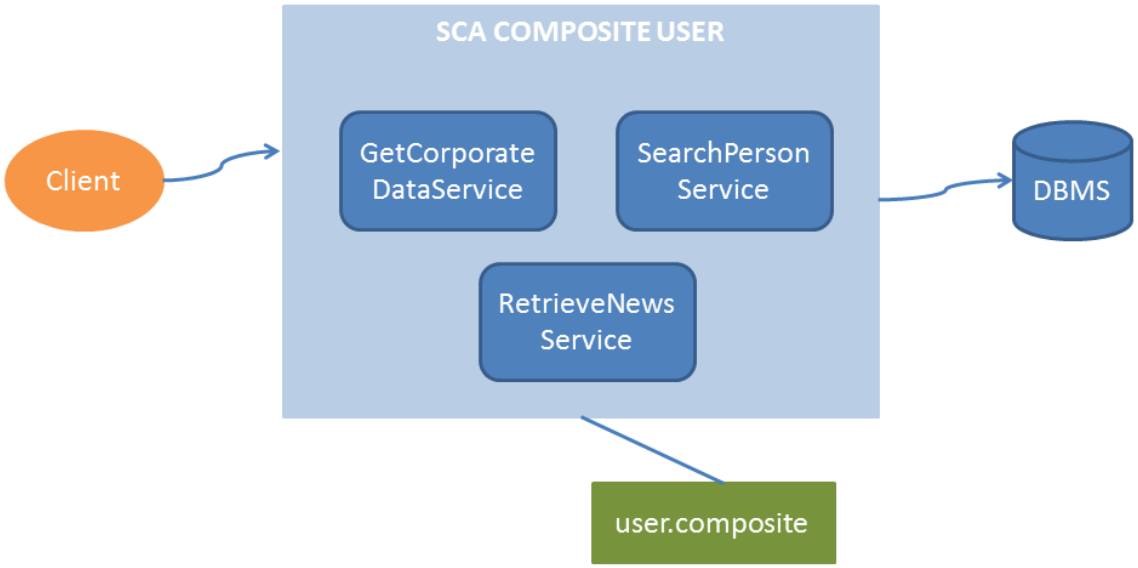
### 5.3.7 Desenvolupament dels serveis

Els serveis seran desenvolupats utilitzant la tecnologia SCA (Service Component Architecture). Aquesta tecnologia defineix cada servei com un component i una unitat lògica anomenada composite que pot contenir varis serveis (components) implementats en diferents tecnologies. D'aquesta manera el client invoca les operacions del composite sense preocupar-se de com estiguin implementats els serveis que conté.




http://www.davidchappell.com/articles/Introducing_SCA.pdf

De cara als serveis que utilitzarà l'arquitectura, es crearà un únic composite (User) que contindrà els 3 serveis que es vol utilitzar. L'estructura és la següent:




Des dels controladors d'arquitectura s'invocarà al client generat, el qual accedirà al composite i n'invocarà una de les seves operacions.

El següent exemple mostra un client accedint al composite i invocant un servei de test:

```
public class TestClient {

    public String callTest() {
        UserService us = new UserService();
        User service = us.getUserPort();

        String profile = service.getProfile("usuari");
        return profile;
    }

}
```

Per a poder invocar les operacions, el servidor de serveis haurà d'estar en funcionament. Per a arrancar el servidor amb els serveis, es disposarà d'una classe Java:

```
public class ServerLauncher {

    public static void main(String[] args) {
        SCADomain scaDomain = SCADomain.newInstance("META-INF/composites/user.composite");

        try {
            System.out.println("User server started (press enter to shutdown)");
            System.in.read();
        } catch (IOException e) {
            e.printStackTrace();
        }

        scaDomain.close();
        System.out.println("User server stopped");
    }
}
```

Internament, els components treballen amb la base de dades mitjançant Hibernate i Spring per a la configuració de Hibernate.

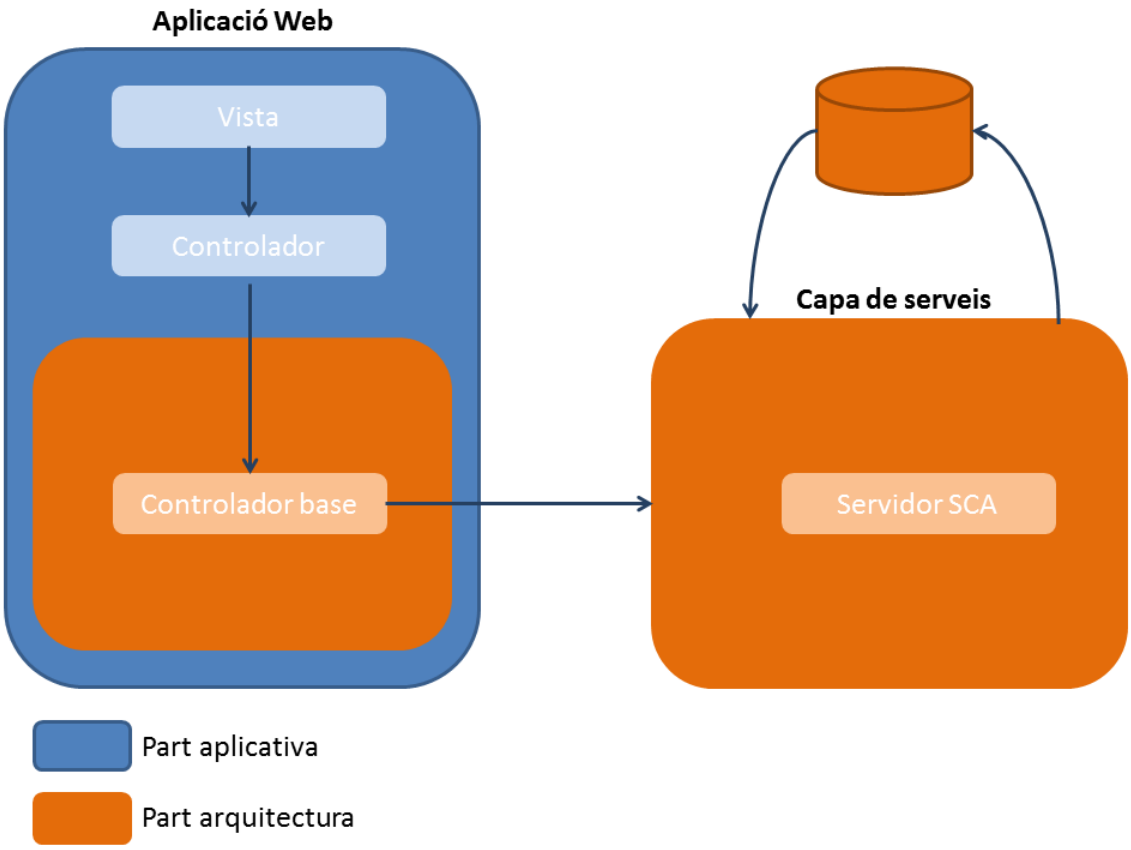
## *5.4 Capítol 4. Avantatges funcionals de l'arquitectura*

En aquest capítol s'explica les diferents avantatges que ofereix la versió actual de l'arquitectura. Cada subapartat d'aquest capítol detalla cadascuna de les funcionalitats demanades durant l'anàlisi funcional llistades a continuació:

- Accés a la capa de serveis
- Configuració genèrica
- Gestió de l'idioma
- Traçabilitat
- Gestió d'errors
- Seguretat
- Llibreria de components visuals

### 5.4.1 Invocació a serveis de negoci

La invocació a serveis de negoci es realitza en la capa intermitja de l'arquitectura, separada de la capa de presentació. EJWAD proporciona un controlador base el qual els desenvolupadors hauran d'estendre per disposar d'un seguit de funcionalitats, entre elles, una interfície d'invocació a serveis. Aquesta interfície accedeix a la capa de serveis de l'arquitectura, implementada amb les tecnologies Spring i Hibernate.




Tota la lògica d'invocació al servei, dades de connexió a la base de dades i gestió d'errors, queda amagada de l'aplicació. EJWAD proporciona la següent interfície per a la crida a un determinat servei:

```
protected Object invokeService(
            RequestContext req,
            Class<?> serviceClient,
            String operation,
            Class<?>[] paramTypes,
            Object[] paramValues) {
```

Mitjançant aquest mètode, el desenvolupador només haurà de proporcionar les dades del client (classe i operació) i els paràmetres que vol enviar:

- serviceClient: Classe del client generat del servei que es vol invocar.
- operation: Nom de l'operació que es vol invocar
- paramTypes: Llista dels tipus (classe) de cada paràmetre que es vol enviar.
- paramValues: Valors dels paràmetres que es vol enviar.

El següent és un exemple extret de l'aplicació de demostració:

```
Class[] paramTypes = {java.lang.String.class};
Object[] paramValues = {WebHelper.getUser().getLocale().getLanguage()};

List<Noticia> result = (List<Noticia>) invokeService(
        req,
        UserServiceClient.class,
        "getNews",
        paramTypes,
        paramValues);
```

L'anterior codi invoca el servei de noticies que es troba desplegat a la capa de serveis, passant-li com a paràmetre l'idioma amb el qual es vol recuperar les noticies.

La gestió d'errors es porta a terme des de l'arquitectura:

```
long initTime = System.nanoTime();
Object result = null;
try {
    result = WebHelper.getManager().invokeService(serviceData);
} catch (Exception e) {
    long endTime = System.nanoTime();;
    getLogger().info("Service invocation failed. Time: "+(endTime-initTime)+" ms");
    addMessage(req, getServiceFailedMessage(e.getMessage()), Severity.ERROR);
}

long endTime = System.nanoTime();
if (result == null) {
    getLogger().info("Service response null. Time: "+(endTime-initTime)+" ms");
    addMessage(req, getServiceFailedMessage("service response null"), Severity.ERROR);
}
else {
    getLogger().info("Service invoked. Time: "+(endTime-initTime)+" ms");
}

return result;
```

D'aquesta manera, el controlador base retorna la resposta del servei al controlador aplicatiu, mentre que si hi ha hagut algun error, la pròpia arquitectura afegeix el missatge d'error al context. Quan es torni a mostrar la vista, l'error es mostrarà per pantalla.

Normalment, l'aplicació no desitja mostrar el missatge generat a la capa de serveis, ja que acostuma a ser un missatge no entenible per l'usuari. Si el desenvolupador desitja mostrar un missatge personalitzat, només haurà d'estendre el mètode **getServiceFailedMessage** i retornar el seu propi missatge.

**Avantatges:**

- L'aplicació no necessita configurar la connexió a la base de dades.
- L'arquitectura s'encarrega de la gestió d'errors que es puguin produir durant la invocació.
- La invocació a la capa de serveis es realitza de manera transparent per l'aplicació.
- El desenvolupador només s'ha de preocupar d'indicar l'operació que vol invocar i dels paràmetres a enviar.

### 5.4.2 Configuració genèrica

Els fitxers amb les propietats de configuració de les aplicacions es gestionen de manera externalitzada en un determinat directori del sistema de fitxers de la màquina. Aquest directori contindrà els fitxers de configuració de l'arquitectura i de totes les aplicacions presents en aquest entorn. A continuació es mostra un exemple del contingut del directori:

```
configuration_root\configuration\

        ejwad_[entorn].properties
        aplicacioA_[entorn].properties
        aplicacioB_[entorn].properties
```

Els valors que pot tenir [entorn] seran: LOC, DEV, PRE i PRO.

El directori **configuration_root** és configurable per entorn, ja que aquest és definit en l'script d'arranc del servidor d'aplicacions com es veu a continuació:

```
-DconfigurationPath="D:\environment\ejwad"
-Denvironment="LOC"
```

Com es veu en l'exemple, a part dels fitxers de configuració, també hi ha present un fitxer d'arquitectura. Aquest fitxer defineix les propietats de configuració comunes a totes les aplicacions, evitant haver d'escriure una nova propietat comuna a cada fitxer de configuració.

L'accés a una propietat aplicativa des de l'aplicació es fa mitjançant la següent crida:

```
WebHelper.getApplicationProperty("app.session.timeout");
```

També és possible recuperar el valor d'una propietat d'arquitectura (propietats comunes) mitjançant la crida:

```
WebHelper.getArchitectureProperty("session.timeout");
```

Avantatges:

- Externalització de la configuració: Permet modificar valors de configuració i aplicar-los només reiniciant el servidor, sense necessitat de tornar a desplegar l'aplicació.
- Gestió de propietats comunes: Mitjançant el fitxer de configuració d'arquitectura.
- Configuració simple: Només requereix situar el fitxer en el directori indicat, no cal definir cap gestor de configuració.
- Diferenciació d'entorns: Mitjançant la variable **entorn**, es pot disposar d'un fitxer de configuració per entorn.

### 5.4.3 Internacionalització (i18n)

Aquest apartat detalla la configuració i el funcionament en aspectes d'internacionalització de les aplicacions. Aquest se centra en dos aspectes:

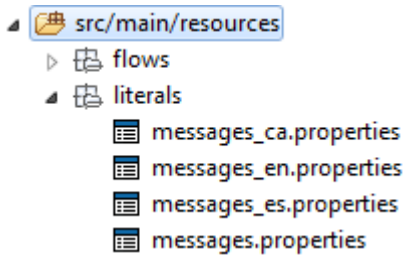
- Gestió de literals
- Gestió del canvi d'idioma

Conceptes relacionats:

**Expression Language (EL)**: És un llenguatge de scripts mitjançant el qual és possible accedir a la capa d'aplicació des de la capa de presentació (vista).

Gestió de literals

Els fitxers de literals són un dels pocs aspectes no configurables en aquesta fase inicial de l'arquitectura. Aquests han d'estar situats dins la següent estructura del projecte de l'aplicació web:




Com es pot veure, la nomenclatura ha de ser la següent: messages_[idioma].properties.

Des de l'aplicació es podrà accedir als literals aplicatius de la següent manera:

- **Des del controlador**: Accedint a la interfície que proporciona l'arquitectura mitjançant el següent mètode:

```
WebHelper.getApplicationLiteral("news.title");
```

- **Des de la vista**: Mitjançant Expression Language i el prefix "key".

```
<uoc:secondaryTitle value="#{key.literal('news.title')}" />
```

També és possible recuperar literals propis de l'arquitectura, tot i que no hauria de ser necessari accedir-hi.

- **Des del controlador**: Accedint al següent mètode:

```
WebHelper.getArchitectureLiteral("test.prop1");
```

- **Des de la vista**: Mitjançant Expression Language i la següent funció:

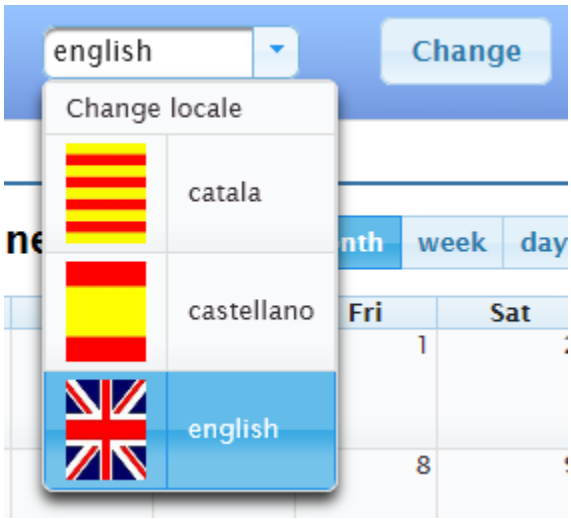
```
itemLabel="#{arch.literal('language.change')}"
```

Gestió del canvi d'idioma

L'arquitectura EJWAD proporciona a les aplicacions, una capçalera (header) des de la qual es pot accedir al mecanisme de canvi d'idioma.


En la versió actual d'EJWAD, l'arquitectura té disponibles tres idiomes

- Català
- Castellà
- Anglès




Un cop escollit l'idioma, l'aplicació es recarregarà mostrant tots els texts en el nou idioma escollit.

L'extensibilitat de l'arquitectura permetrà afegir els idiomes que es requereixin.

Avantatges:

- Permet tenir l'aplicació internacionalitzada simplement col·locant un fitxer de propietats en una carpeta determinada del projecte.
- Permet canviar l'idioma dinàmicament de manera transparent pel desenvolupador.
- Permet afegir idiomes de manera ràpida i sense afectar l'aplicació

### 5.4.4 Traçabilitat

Les aplicacions que s'executin sobre l'arquitectura EJWAD disposen d'un fitxer de configuració que hauran d'instal·lar al sistema de fitxers de la màquina (explicat amb més detall a la guia d'instal·lació). Aquest fitxer defineix com es guardaran les traces generades durant l'execució de l'aplicació.

El següent exemple mostra la definició de les traces, les quals generaran dos fitxers:

```
<appender name="ArchitectureAppender" class="org.apache.log4j.DailyRollingFileAppender">
    <param name="file" value="/environment/ejwad/logs/ejwad_architecture.log"/>
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d|%-5p|%c{1}|%m%n"/>
    </layout>
</appender>

<appender name="ChannelAppender" class="org.apache.log4j.DailyRollingFileAppender">
    <param name="file" value="/environment/ejwad/logs/ejwad_application.log"/>
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d|%-5p|%c{1}|%m%n"/>
    </layout>
</appender>
```

D'aquesta manera queden separades les traces de l'arquitectura de les de l'aplicació.

Traces d'arquitectura

Les traces d'arquitectura mostren aspectes tècnics com poden ser les navegacions aplicatives registrades o l'identificador d'usuari registrat.

```
2012-06-05 08:55:29,031|DEBUG|DefaultLoggingManager|||||Logging file initialized | /environment/ejwad/logs/log4j.xml
2012-06-05 08:55:33,984|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: main-flow
2012-06-05 08:55:34,140|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: parent-flow
2012-06-05 08:55:34,140|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: test-flow
2012-06-05 08:55:34,406|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: main-flow
2012-06-05 08:55:34,468|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: parent-flow
2012-06-05 08:55:34,468|INFO |ClassPathFlowRegistryPostProcessor|||||Registering flow: test-flow
2012-06-05 08:55:44,031|INFO |WebHelper|||||New user logged: xavi
```

Aquest fitxer també és l'encarregat d'emmagatzemar les traces d'error que puguin produir-se:

```
2012-06-05 08:38:06,421|ERROR|DefaultServiceManager|||||Error invoking service|getNews
java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
```

D'aquesta manera es pot mostrar per pantalla un missatge amigable per l'usuari mentre tenim el detall de l'error a les traces.

Traces aplicatives

Aquestes traces contenen tota la informació que el desenvolupador vulgui registrar. Normalment aquestes traces es registraran des dels controladors per registrar la invocació de serveis o per registrar lògica de negoci:

```
2012-06-05 09:37:59,203|DEBUG|GetNewsController|||||Accessing news service controller
2012-06-05 09:38:08,593|INFO |GetNewsController|||||Service invoked. Time: 9328 ms
2012-06-05 09:38:08,593|DEBUG|GetCorporateInfoController|||||Accessing corporate information service controller
2012-06-05 09:38:09,281|INFO |GetCorporateInfoController|||||Service invoked. Time: 641 ms
```

Per tal de facilitar les tasques d'enregistrament, el controlador base d'arquitectura proporciona un mètode el qual estendran els controladors aplicatius. D'aquesta manera s'afegirà una nova traça invocant el següent mètode:

```
public String process(RequestContext req) {
    getLogger().debug("Accessing news service controller");
```

Organització dels fitxers

Els fitxers queden separats per capa (arquitectura / aplicació) i per data, de manera que tenim un fitxer per cada dia:

| Nombre | Tamaño | Tipo | Fecha de r |
|---|---|---|---|
| ejwad_architecture.log | 41 KB | Documento de texto | 05/06/2012 |
| ejwad_application.log | 4 KB | Documento de texto | 05/06/2012 |
| ejwad_architecture.log.2012-06-04 | 28 KB | Archivo 2012-06-04 | 04/06/2012 |
| ejwad_application.log.2012-06-04 | 1 KB | Archivo 2012-06-04 | 04/06/2012 |
| log4j.xml | 3 KB | Documento XML | 02/06/2012 |

Si es vol afegir nous fitxers de traces, com per exemple un per cada aplicació, s'haurà d'afegir al fitxer de definició de logs (log4j.xml).

Avantatges:

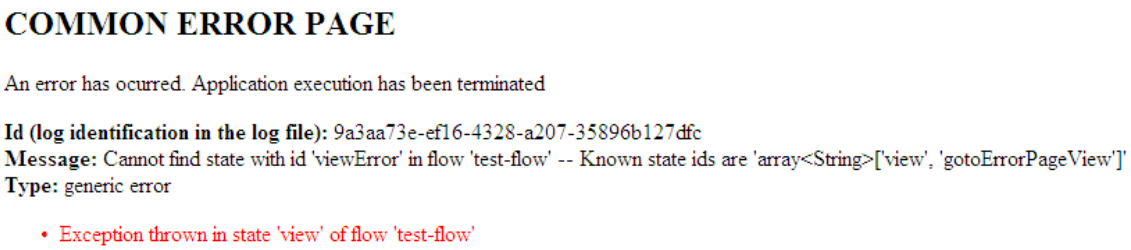
- No requereix cap configuració per habilitar les traces.
- Es disposa d'una llista de fitxers de traces separats per data i per capa (aplicacions/arquitectura).

### 5.4.5 Gestió d'errors

La gestió d'errors es divideix en dues parts:

- **Errors controlats**: Aquests errors estan prèviament identificats i estan gestionats de manera que es mostri un missatge per pantalla. Aquest missatge es mostra a la part superior dreta de la pantalla junt amb una icona d'informació. Les tipologies d'errors poden ser:
    - Error en la invocació a un servei: Ja sigui perquè ha caigut el servidor de serveis, algun error de base de dades o simplement perquè no ha retornat cap resultat.
    - Errors/Missatges de negoci: Qualsevol missatge que pugui generar la lògica de negoci i sigui important per l'usuari.


- **Errors no controlats**: Errors no previstos deguts a un error de programació o a un ús indegut dels components. En aquests casos es tracta d'un error no recuperable i per tant, es redirigeix a la pàgina d'error comuna.




Aquesta vista conté la següent informació:

- Íd: Identificador únic d'excepció. Servirà per localitzar ràpidament l'error en el fitxer de traces.
- Message: Missatge de l'excepció generada.
- Type: Tipus d'error. Actualment hi ha definits els següents tipus: genèric, de vista i de flux.

Avantatges:

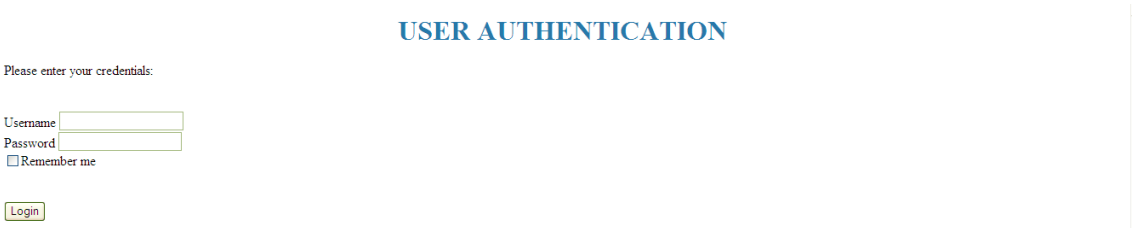
- No requereix cap tasca de configuració per part de l'usuari.
- Els errors capturats es registren als fitxers de lòg d'arquitectura.
- Totes les aplicacions mostren els missatges i errors de la mateixa manera, simplificant l'adaptació dels usuaris a les noves aplicacions.

### 5.4.6 Seguretat

L'arquitectura té integrat el sistema d'autenticació contra base de dades. Aquest consisteix en una pàgina de login com a únic accés lliure de l'aplicació. Serà necessari autenticar-se per a poder accedir a l'aplicació.

Aquesta pàgina ve inclosa dins el jar d'arquitectura, i serà copiada a l'aplicació de manera transparent per l'usuari durant la primera invocació a l'aplicació un cop desplegada al servidor d'aplicacions. En les següents invocacions ja no serà necessari efectuar aquest pas.

**Pàgina de login:**




En cas d'introduir un usuari invàlid, no se'ns permetrà accedir a l'aplicació:


Avantatges:

- No requereix configuració, només afegir el nou usuari a la base de dades.
- Gestió de permisos. Encara que actualment només gestiona dos tipus de permisos (usuari i administrador), obre la porta a tenir diferents perfils d'usuari.

### 5.4.7 Llibreria de components

Durant aquest TFC, s'ha desenvolupat una llibreria de components web reutilitzables que permeten simplificar la quantitat de codi necessari per a pintar una pàgina. Cada component consisteix en una agrupació de components simples que en formen un de més complex. Alguns altres són extensions de components ja existents de llibreries publiques.

La selecció de components, tot i ser reutilitzables, s'ha desenvolupat pensant en les necessitats de l'aplicació de demostració i són els següents:

- **Calendari d'events**: Permet la consulta d'events mitjançant un calendari que suporta internacionalització. Només requereix un bean que estengui una classe d'arquitectura i que proporcioni una llista d'events:

```
@Component("corporateInfoBean") @Scope("conversation")
public class CorporateInfoBean extends SchedulerBean implements Serializable {

    private static final long serialVersionUID = 7254969032189770423L;
    private List<ScheduleObject> events;

    @Override
    public List<ScheduleObject> getEvents() {
        return events;
    }

    @Override
    public void setEventsList(List<ScheduleObject> events) {
        this.events = events;
    }
}
```

Els events, en el cas de l'aplicació de demostració, són recuperats d'un servei. El resultat és el següent:

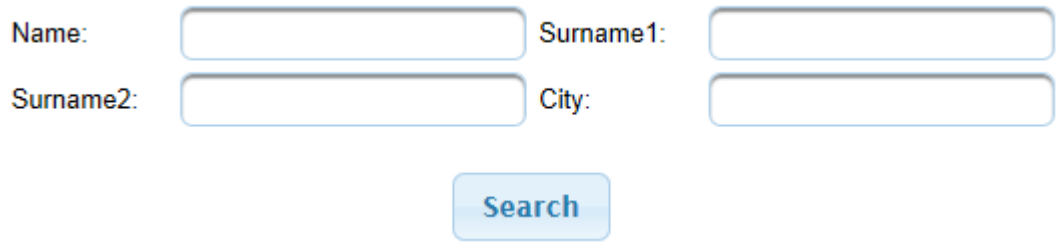
A la pàgina, el desenvolupador ha necessitat el següent codi:

```
<uoc:corporateInfo value="#{corporateInfoBean.eventModel}" model="#{corporateInfoBean}"/>
```

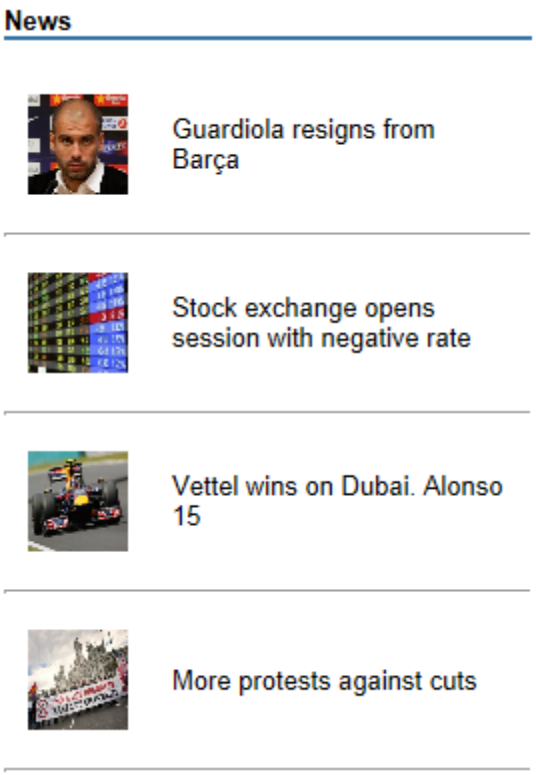
- **Caixa de filtres**: Consisteix en una caixa formulari que conté una quantitat variable de camps (d'un a quatre). Aquests camps són configurables amb els valors que introdueixi l'usuari. El codi és el següent:

```
<uoc:filterBox number="4" update=":form2"
    fieldLabel1="#{key.literal('search.name')}" fieldName1="#{searchPersonBean.name}"
    fieldLabel2="#{key.literal('search.surname1')}" fieldName2="#{searchPersonBean.surname1}"
    fieldLabel3="#{key.literal('search.surname2')}" fieldName3="#{searchPersonBean.surname2}"
    fieldLabel4="#{key.literal('search.city')}" fieldName4="#{searchPersonBean.city}"/>
```

El resultat es veu a continuació:




- **Noticies**: Mostra 4 caixes d'informació estructurades en una imatge i un titular.



- **Taula de resultats**: Mostra informació estructurada en una taula. Addicionalment permet obrir una finestra amb informació detallada en seleccionar un element de la taula:

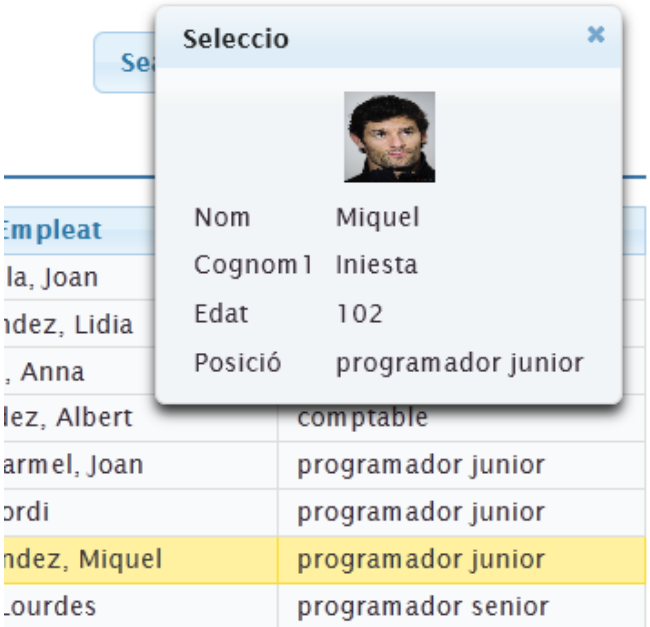
```
<uoc:resultsTable id="resultsTable" fullId="form2:resultsTable"
    model="#{personBean.dataModel}" selection="#{personBean.personSelected}" render=":form2:display"
    detailPhoto="urlFoto" detailField1="nom" detailFieldLabel1="#{key.literal('results.name')}"
    detailField2="cognom1" detailFieldLabel2="#{key.literal('results.surname1')}"
    detailField3="edat" detailFieldLabel3="#{key.literal('results.age')}"
    detailField4="posicio" detailFieldLabel4="#{key.literal('results.position')}">

    <p:column headerText="#{key.literal('results.id')}">
        #{iter.id}
    </p:column>
     <p:column headerText="#{key.literal('results.employee')}">
        #{iter.cognom1} #{iter.cognom2}, #{iter.nom}
    </p:column>
    <p:column headerText="#{key.literal('results.position')}">
        #{iter.posicio}
    </p:column>
</uoc:resultsTable>
```

El component es mostra de la següent manera:

| Id | Employee | Position |
|---|---|---|
| 6001 | Domenech Vila, Joan | general manager |
| 6002 | Garcia Fernandez, Lidia | human resources |
| 6003 | Llobet Garcia, Anna | human resources |
| 6004 | Mata Fernandez, Albert | accounting |
| 6005 | Hernandez Carmel, Joan | junior programmer |
| 6006 | Puyol Grau, Jordi | junior programmer |
| 6007 | Iniesta Fernandez, Miquel | junior programmer |
| 6008 | Villa Garcia, Lourdes | senior programmer |

1 2

En seleccionar un element de la taula es mostra informació detallada de l'element:



- **Títol secundari**: Mostra un element títol amb informació rellevant:

```
<uoc:secondaryTitle value="#{key.literal('news.title')}" />
```

**Notícies**

Mitjançant la internacionalització, tots els components s'adapten a l'idioma escollit per l'usuari, actualitzant-se en seleccionar un altre idioma des de la capçalera de l'aplicació.

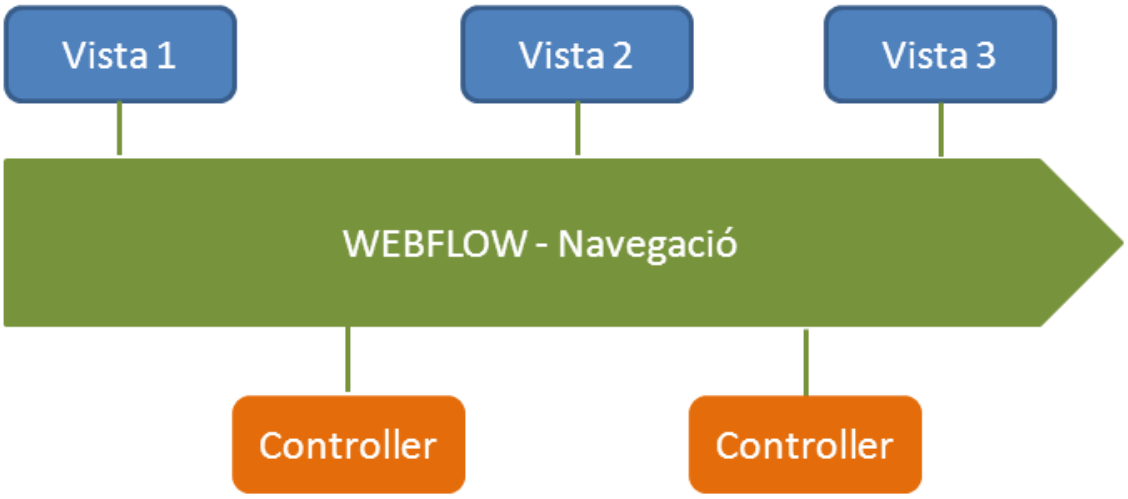
Avantatges:

- Reducció del codi necessari a escriure pel desenvolupador
- Simplificació de la pàgina: Degut a la reducció de codi, la pàgina és més neta.
- Components reutilitzables: Els components que proporciona l'arquitectura poden ser utilitzats per qualsevol aplicació que ho desitgi.

## *5.5 Capítol 5. Navegació*

### 5.5.1 Introducció

La navegació de les aplicacions està definida per Spring Webflow, fora de la tecnologia de vista (JSF). D'aquesta manera queda més separada la lògica de la presentació. El següent gràfic mostra les capes superiors de l'arquitectura:




Un cop l'usuari prem un enllaç o component de navegació, es retorna el control a Webflow, el qual continua el flux cap a la següent vista o controlador.

Definició de la navegació

La navegació d'una aplicació es defineix en un fitxer xml. Bàsicament, és una execució d'estats entre els que destaquen els següents:

- **Estat vista (view):** Conté l'enllaç a la pàgina que conté la vista i les transicions que es podran executar cap a altres estats.
- **Estat acció (action):** Conté la invocació al controlador. Aquest estat conté també les transicions que s'efectuaran, segons la resposta del controlador.

Els anteriors estats són els més utilitzats, però Webflow és molt més potent, permetent definir estats de decisió, subfluxes, etc…

### 5.5.2 Flux base - Herència

Tal i com estan definits, tots els fluxes aplicatius hereten d'un flux base (parent) el qual conté lògica d'arquitectura. Aquest flux base conté principalment la captura i gestió dels errors que es puguin produir durant l'execució d'una aplicació. En el cas concret d'EJWAD, aquest s'encarrega de recollir informació de l'error i redirigir el flux cap a una pàgina d'error comuna on es mostrarà els principals detalls.

Com es pot veure en la següent captura, el flux base d'EJWAD conté algunes transicions globals com per exemple el canvi d'idioma. També és el lloc on es defineix quin serà el bean que s'encarregarà de gestionar la captura d'errors del flux.

**Flux base**

parent-flow.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/webflow
        http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">


    <end-state id="gotoErrorPageView" view="commonErrorPage1.xhtml"/>

    <global-transitions>
        <transition on="gotoErrorPage" to="gotoErrorPageView"/>
        <transition on="archChangeLocale" to="#{conversationScope.flowSupportBean.lastView}">
            <evaluate expression="changeLocaleController"/>
        </transition>
    </global-transitions>

    <exception-handler bean="errorHandler"/>

</flow>
```

Per a heretar del flux base, el flux aplicatiu haurà de definir el pare en la seva capçalera:

```
<flow xmlns="http://www.springframework.org/schema/webflow"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schem
    parent="parent-flow">
```

La següent captura mostra un exemple de com seria una navegació aplicativa, en la qual s'inicia amb l'execució de dos controladors i es mostra una vista (view.xhtml). També s'inclou un exemple de transició, que serà generada per un component de navegació:

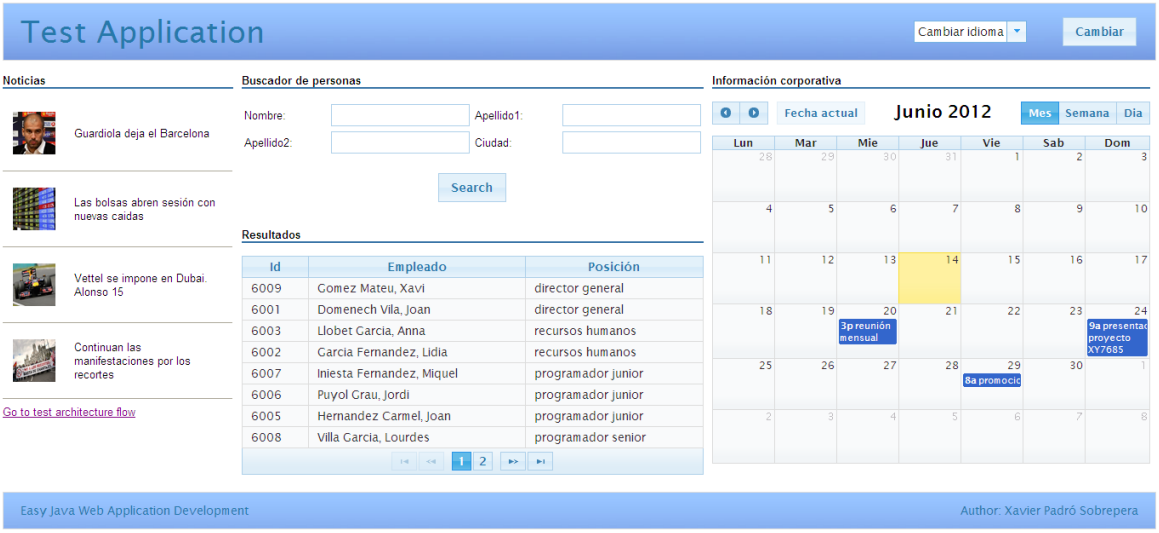
```
<!-- Main view -->
<view-state id="view" view="view.xhtml">
    <on-entry>
        <evaluate expression="getNewsController"/>
        <evaluate expression="getCorporateInfoController"/>
    </on-entry>
    <!-- test literals -->
    <transition on="testArgLiterals" to="view">
        <evaluate expression="testArgLiteralsController"/>
    </transition>
```

## *5.6 Capítol 6. Aplicació de demostració*

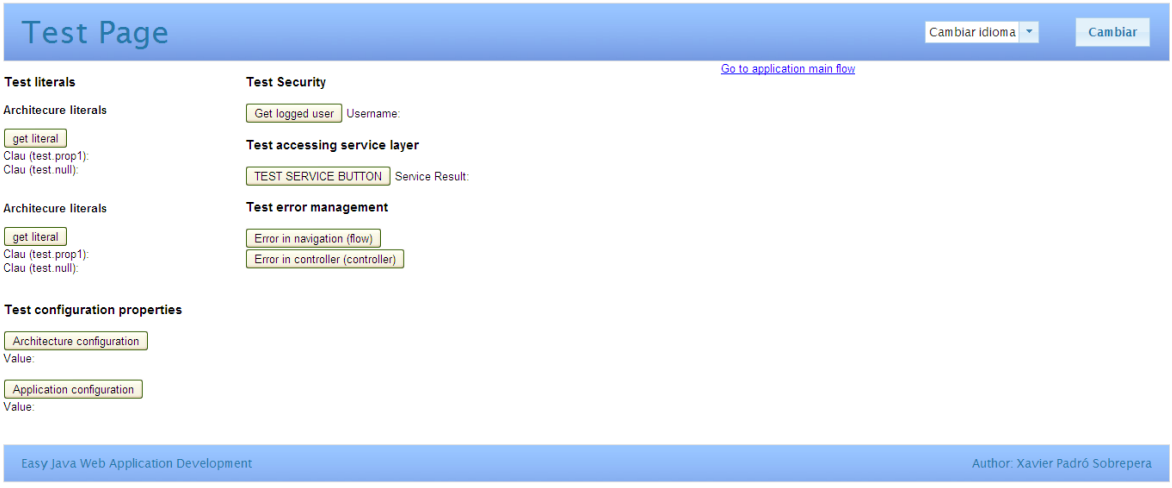
Junt amb el desenvolupament de l'arquitectura, aquest TFC inclou també la construcció d'una aplicació web muntada sobre aquesta arquitectura. L'objectiu de l'aplicació és validar la integració d'una aplicació amb l'arquitectura que la suporta, comprovar l'efectivitat de les funcionalitats proporcionades i accedir a la capa de serveis SCA.

L'aplicació es divideix en dos fluxes

- Principal (main-flow): Flux inicial de l'aplicació. Aquest flux mostra la integració de l'aplicació amb l'arquitectura, utilitzant la capçalera de canvi d'idioma i recuperant informació dels serveis de negoci mitjançant controladors.




- Test (test-flow): La pàgina principal del main-flow proporciona un enllaç que salta de flux iniciant-ne el de test. Aquest, conté una sèrie d'enllaços per provar directament les funcionalitats de l'arquitectura, com per exemple l'accés a literals (tant aplicatius com d'arquitectura), accés a propietats de configuració, consulta de l'usuari identificat, etc…



## *5.7 Capítol 7. Conclusions*

**Què ha aportat?**

La realització del present projecte m'ha permès dissenyar i desenvolupar de manera simplificada cadascuna de les capes que conformen una arquitectura J2EE. Això m'ha aportat una visió més global del que significa un projecte d'aquestes característiques, donat que en els projectes on he treballat, les persones estan especialitzades en àrees concretes de l'arquitectura. Durant la primera fase del projecte he hagut de deixar de banda la visió del programador per ampliar-la a la visió de l'arquitecte.

Aquest projecte m'ha donat l'oportunitat de provar noves tecnologies com Hibernate o l'arquitectura orientada a serveis (SOA) i d'esbrinar com s'integren i es comuniquen cadascuna de les capes. De cara a les parts que ja coneixia, he pogut utilitzar tecnologies diferents i pensar en un disseny alternatiu a l'utilitzat on treballo.

**Aspectes a millorar**

Degut al temps del que he disposat per a realitzar el projecte, hi ha varis aspectes que he hagut de descartar. L'aplicació es podria enriquir amb més tipus de perfils d'usuari, permetent una visió més dinàmica de l'aplicació. També en el cas d'haver inclòs la fase de manteniment, s'hagués pogut incloure temes de monitorització de l'arquitectura, mesurar temes de rendiment i detectar fugues de memòria.

**Futur de l'àrea seleccionada**

No hi ha dubte que a nivell empresarial, es tracta d'una àrea molt demandada, especialment les tecnologies utilitzades per a desenvolupar aquest projecte. La programació orientada a objectes és el present i el futur dels sistemes d'informació, i no té visos de canviar a curt termini.

# 6 Glossari

| | |
|---|---|
| **JSF** | És una especificació que estableix l'estàndard per construir interfícies d'usuari. Sobre aquesta especificació s'han dissenyat una sèrie d'implementacions amb l'objectiu de simplificar el desenvolupament d'aplicacions web. |
| **Spring** | Framework de desenvolupament d'aplicacions per a la tecnologia Java. |
| **SOA** | Arquitectura orientada a serveis. Defineix la utilització de serveis per donar suport als requeriments de negoci. |
| **SCA** | Service Component Architecture. Tecnologia de software que proveeix un model per desenvolupar aplicacions que segueixen els principis de SOA. |
| **Composite** | Unitat de desplegament de SCA. Conté serveis de negoci que poden ser accedits remotament. |
| **Reusabilitat** | Es refereix al disseny de funcionalitats de programari que poden tornar a ser utilitzades sense patir modificacions. |
| **MVC** | Patró de disseny model-vista-controlador. Separa la informació en tres components separats: dades (model), interfície d'usuari (vista) i lògica de negoci (controlador). |
| **Hibernate** | Eina per a tecnologia Java dissenyada per convertir dades entre una base de dades relacional i el sistema. |
| **Internacionalització (i18n)** | Procés de dissenyar programari per tal de que pugui adaptar-se a diferents idiomes i regions sense haver de modificar codi. |
| **Expression Language (EL)** | Llenguatge de scripts que permet l'accés a components Java (beans) des d'una vista (JSP). |
| **Autenticació** | És el fet de confirmar la identitat de la persona que està intentant accedir a un programa. |
| **Login** | És el procés mitjançant el qual es controla l'accés a un sistema informàtic mitjançant la identificació de credencials. |
| **Webflow** | Framework que proporciona la infraestructura per l'execució d'aplicacions web mitjançant transicions a estats. |
| **Flux** | Conjunt d'estats i transicions que conformen una navegació dins l'aplicació web. |
| **Transició** | Event llançat des d'una vista per un component de navegació i que provoca un canvi d'estat en l'execució de l'aplicació. |

# 7 Bibliografia

**Framework Primefaces**
http://www.primefaces.org/showcase-labs/ui/home.jsf

**SCA Apache Tuscany**
http://www.acis.org.co/fileadmin/Conferencias/ConfJorgeMarioCalvo.pdf
http://tuscany.apache.org/getting-started-with-tuscany.html
http://wiki.eclipse.org/images/6/61/FirstStepsWithTheSCADesigner.pdf

**Navegació (Spring Webflow)**
https://src.springframework.org/svn/spring-samples/webflow-primefaces-showcase/
http://www.redcode.nl/blog/2009/12/splitting-up-spring-web-flow-facelets-into-jars/
http://static.springsource.org/spring-webflow/docs/2.0.x/reference/htmlsingle/spring-webflow-reference.html

**Internacionalització en aplicacions**
http://www.proactiva-calidad.com/java/spring/internacionalizacion.html

**JSF 2.0 (instal·lació i preparació entorn)**
http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/JSF2-Getting-Started.pdf

**JSF 2.0**
http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/
http://www.mkyong.com/tutorials/jsf-2-0-tutorials/
http://www.slideshare.net/jimdriscoll/a-complete-tour-of-jsf-2

**JSF 2.0 (custom components)**
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=customJsfComponent
http://weblogs.java.net/blog/driscoll/archive/2009/10/09/jsf-2-custom-java-components-and-ajax-behaviors

**JSF 2.0 (fix integració JSF2 amb Webflow)**
https://jira.springsource.org/browse/SWF-1468

**Expression language**
http://docs.oracle.com/javaee/6/tutorial/doc/gjddd.html

Nota: Tots els enllaços es troben operatius en el moment de realitzar aquesta memòria.

# 8 Annexos

## *8.1 Annex 1. Material lliurat*

L'entrega del projecte inclou els següents materials:

| |
|---|
| 1.- Memòria del TFC en format .doc |
| 2.- Guia d'instal·lació de l'entorn |
| 3.- Llibreries de l'arquitectura |
| 4.- Aplicació de demostració |
| 5.- Projecte servidor amb els serveis de negoci inclosos |
| 6.- Presentació del TFC en els formats .ppt i .pps |