CONSORCIO
PARA O DESENVOLVEMENTO DE SISTEMAS
PARA XESTIÓN UNIVERSITARIA

CIXUG

Arquitectura Técnica
Anexo VI

Para a contratación do desenvolvemento das melloras funcionais requiridas no Sistema Universitario de Xestión da Investigación - Rexistro de Investigadores (SUXI-RI) – Versión 3.00

31/05/2012
V. 2012-100

Código: **PA01/2012**

UNIVERSIDADE DA CORUÑA

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Universidade Vigo

ÍNDICE

1 Introducción

O obxectivo do presente documento é describila arquitectura e estrutura básica da aplicación “Sistema unificado de xestión de investigadores. Rexistro de investigadores”, dende agora SUXI-RI.

2 Particionamento físico do sistema

O sistema esta composto de 3 nodos principais: cliente, servidor e sistemas externos.

1. **Posto Cliente:** PCs de usuario con navegador Web conectados á rede.

2. **Servidor:** O servidor é o nodo ao que se conectan os usuarios para acceder ao sistema. Dentro do nodo servidor podemos diferenciar:
    a. **Servidor Aplicacións J2EE:** executa toda a lóxica de negocio e de presentación, tamén é o encargado de atender as peticións dos clientes.
    b. **Base de datos:** mantén a información xerada polo sistema.
    c. **Proceso de sincronización de datos:** Proceso que alimenta a base de datos SUXI-RI cos datos da base de datos de persoal da Universidade. A implementación deste proceso depende de cada Institución e entre os datos implicados na sincronización figuran:
        - Datos persoais dos investigadores
        - Departamentos e areas de investigación
        - Centros e categorías profesionais.

3. **Sistemas externos:** neste nodo se engloban os sistemas externos que manterán conexión co sistema:
    a. **Servizo de autenticación**
    b. **Servizo Web de obtención de datos persoais dos investigadores e usuarios.**
    c. **Servizo Web CVN**

3 Entorno tecnolóxico do sistema

Servidor de aplicacións

- Apache Tomcat 6
- JDK 1.6

Servidor de bases de datos

- Oracle
- Informix
- SqlServer

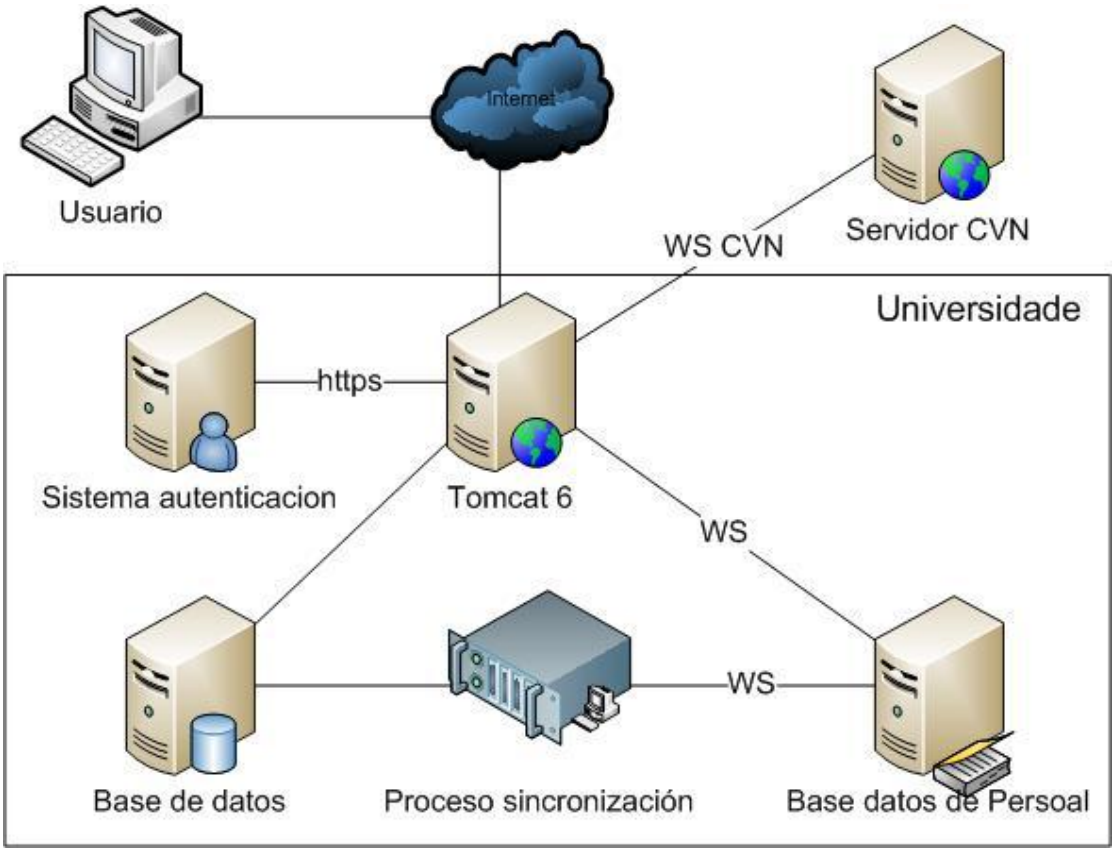
Cliente

- Internet Explorer
- Firefox

Sistemas externos

- Sistema de autenticación
  - Sistema de ticket-cas empregado pola propia Universidade.
  - Sistemas propios
- Servizo web de obtención de datos persoais de investigadores e usuarios.
- Servizo web CVN

4 Arquitectura Hardware



Sistema de autenticación

Sistema propio de autenticación de cada universidade.

Base de datos

Xestor de base de datos onde SUXI-RI explota a información.

Servidor de aplicacións (Tomcat 6)

Servidor que aloxa a aplicación SUXI-RI.

Base de datos de persoal

Base de datos propia da universidade que a aplicación explota vía WS.

Proceso de Sincronización

Proceso de sincronización entre os datos de SUXI-RI e os da base de datos de persoal. O proceso na actualidade atópase desenrolado e implantado na UDC sobre MULE. Na Universidade de Vigo empregase un listener dende a mesma aplicación. Ambos procesos execútanse diariamente.

Servidor CVN

Servidor web do CVN que proporciona o CVN-PDF a partires do CVN-XML

5 Arquitectura software

Estruturalmente a aplicación presenta un desenrolo en tres capas que promove unha especialización en cada un dos compoñentes que forman os distintos sustratos.

- **Capa de Presentación:** Na capa de presentación residen todos os compoñentes de arquitectura relacionados coa interrelación e entrega de información aos usuarios. Para a implementación desta capa utilizouse “Apache Struts” xunto co xestor de plantillas de presentación “Tiles”.

- **Capa de Lóxica de Negocio:** A capa de lóxica de negocio conten toda a lóxica e as regras de negocio necesarias para cubrir os servizos que se ofrecen aos usuarios. Esta capa esta formada polas clases do modelo que implementan as colaboracións necesarias para servir as peticións recibidas. Para a implementación desta capa utilizase o framework “Spring” e as súas comodidades co traballo con aspectos.

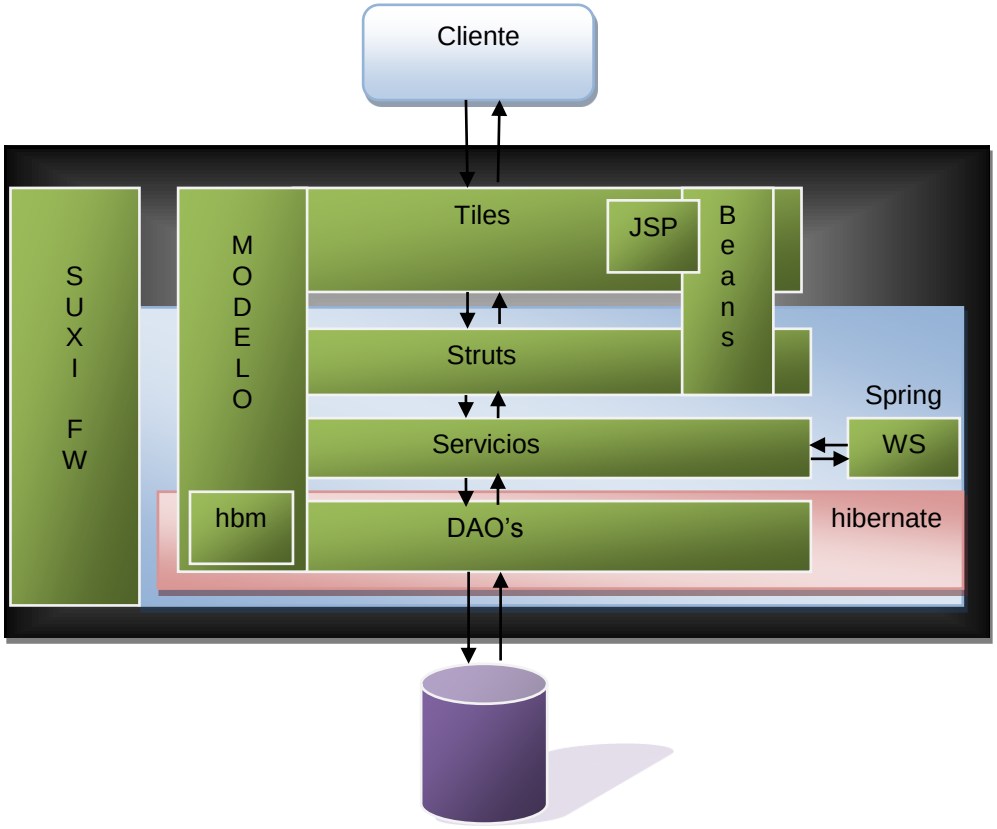
  - **Clases do modelo:** implementan as colaboracións necesarias para servir as peticións que se reciben.

  - **Servizos:** encargada de servir de punto de entrada ás peticións da capa de presentación e a invocación da capa de acceso a datos. Ofrecen unha interface coherente e ordenada. Os servizos serán os encargados da invocación dos obxectos de acceso a datos (DAO), os servizos web utilizados ou calquera outro compoñente de acceso a datos definido no aplicativo.

- **Capa de Acceso a Datos:** A capa de acceso a datos, encapsula o acceso aos datos, independentemente de onde residan os mesmos e ofrece servizos á capa de negocio para facilitar a recuperación e persistencia destes datos. Para a implementación desta capa empregaranse os seguintes compoñentes:

  - **Obxectos DAOs:** empregarase o patrón DAO co obxectivo de manexar a persistencia da información ocultando a implementación de esta persistencia. A capa de acceso a datos implementarase con xestor de persistencia Hibernate.

  - **Servizos Web:** empregase CXF para a invocación aos servizos web tanto da propia aplicación coma de aplicacións externas necesarias para o correcto funcionamento da mesma.



No apartado 9 "Ferramentas de software libre empregadas para a implementación da solución" realízase una breve descrición dos principais compoñentes de software propostos para a implantación da aplicación. Todos estes compoñentes son ferramentas de software libre.

5.1 Estrutura Interna do Proxecto

A construción da aplicación é xestionada por Maven, e empregase o xestor de perfís de Maven para xerar un arquivo de despregue por universidade para as distintas configuracións existentes. SUXI-RI defínese como un proxecto multimodular Maven, os módulos que o compoñen son:

- suxi-ri-modelo
  - Contén tódalas clases relacionadas co modelo de datos que manexa a aplicación. Así mesmo tamén dispón este módulo dos arquivos *.hbm.xml que definen a relación dos obxetos java co modelo relacional de base de datos.

- suxi-ri-logica:
  - Conten tódalas clases de actuación sobre o modelo de datos. Dende os servizos ós DAO's.

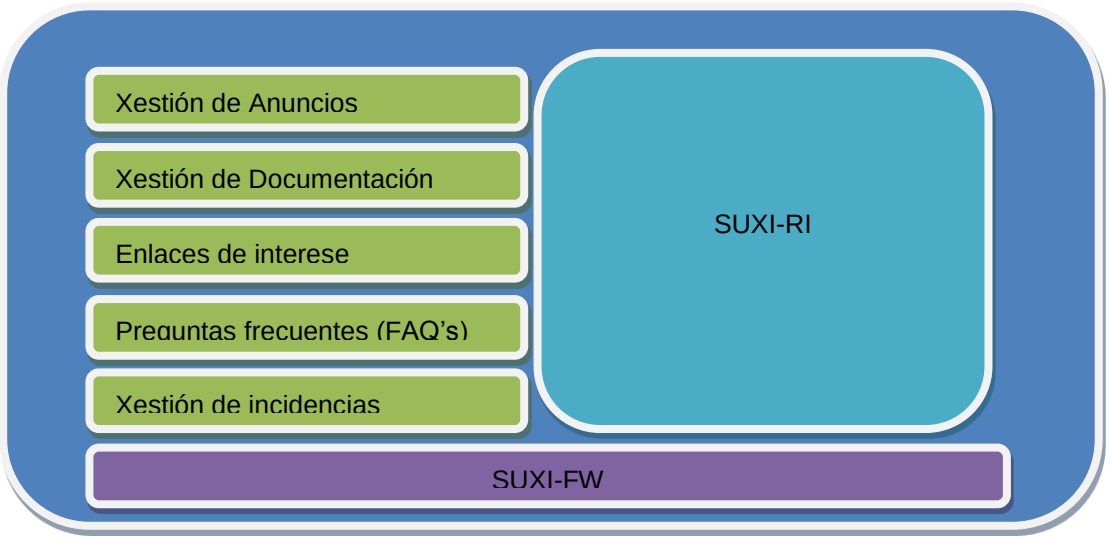
- suxi-ri-webapp
  - Conten tódalas clases de acción da aplicación ó igual que os ficheiros de presentación e arquivos de configuración da aplicación Web.

Esta división modular do código permite empregar o modelo da aplicación ou as clases de actuación sobre os datos da mesma (servizos e DAO's) de modo totalmente independente ó uso da aplicación en si, e dicir, poderían empregarse dende outras aplicacións.

As dependencias locais do proxecto son:

- suxi-fw:
  - Conten as clases básicas sobre as que se desenrola a aplicación. Os módulos dos que se atopa composto móstranse no seguinte apartado.

- suxi-Apphor
  - Proxecto multimodular maven que conten unha serie de pequenas aplicacións de ámbito xeral e que poden empregarse e integrarse con calquer desenrolo. As aplicacións das que dispón son:
    - Xestión de Anuncios
    - Xestión de Documentación
    - Enlaces de interese
    - Preguntas frecuentes (FAQ's)
    - Xestión de incidencias e asesoramento



- cixug-WS
  - Definición e cliente do servizo web que integra a aplicación co sistema de persoal da universidade. A partir deste servizo a aplicación pode obter datos persoais dos investigadores ou usuarios da aplicación, departamentos, áreas de coñecemento, centros …

Módulos do framework SUXI (suxi-fw)

A continuación se describen os módulos dos que se compón o framework suxi (suxi-fw). Cada un dos módulos presenta un empaquetado de tipo 'jar'. As versión son xestionadas por Maven o igual que as dependencias dos módulos nas aplicacións que os empregan, incluso entre as dependencias propias dos módulos entre si.

O proxecto Maven atopase definido co groupID 'SUXI' baixo o artifactID 'suxi-fw'. A continuación se enumeran e describen os módulos que o integran indicando as interdependencias.

suxi-fw-comunes

Módulo que conten tódalas clases de ámbito común do framework:

- Tratamento de ficheiros
- Comunicacións (envío de correos)
- Logs
- Excepcións
- ..

Dependencias

Non aplica.

suxi-fw-modelo

Módulo que conten as clases de entidades básicas propias da universidade.

- Centro
- Usuario
- ..

Dependencias

- *Suxi-fw-comunes*

suxi-fw-acceso

Módulo que conten as clases básicas das que han de herdar os servizos e os DAO's das aplicacións. De esta forma facilitase la elaboración dun modo común de acceso a datos e de uso nas capas máis altas das aplicacións. Os DAO's desenrolados actualmente se basean no ORM hibernate. O desenrolo de DAO's sobre outros ORM's requiriría a súa implementación respetando as interfaces básicas.

Dependencias

- *Suxi-fw-comunes*
- *Suxi-fw-modelo*

suxi-fw-auth

Módulo que conten as clases de autenticación do framework. Conten dende as entidades a os servizos que obteñen os permisos.
Neste módulo albérganse as distintas formas de autenticación das universidades que integran el consorcio e será mediante configuración a selección dunha ou outra.

Dependencias

- suxi-fw-acceso
- *suxi-fw-modelo*

suxi-fw-navegacion

Módulo que conten as clases básicas para as capas máis altas de la aplicación como poden ser:

- As accións e os bean's das que han de herdar as aplicacións (sobre struts).
- Accións comúns como la de la páxina de inicio ou o cambio de idioma.
- Clases de validación propias

- Clases referentes ó control da navegación
- ActionServlet propio das aplicacións SUXI.
- ...

Dependencias

- suxi-fw-auth
- *suxi-fw-navegacion2*

suxi-fw-navegación2

Módulo que estende as clases de la librería displayTag para adaptalas o funcionamento da aplicación.

Dependencias

- suxi-fw-comunes

6 Modelo de Dominio

A continuación darase unha pequena descrición do modelo co cal traballa a aplicación SUXI-RI. Divídese o modelo xeral en pequenas partes relacionadas:

- Datos do investigador
- Colectivos
- Méritos e soportes
- Mestres e criterios de clasificación e valoración

6.1 Datos do investigador

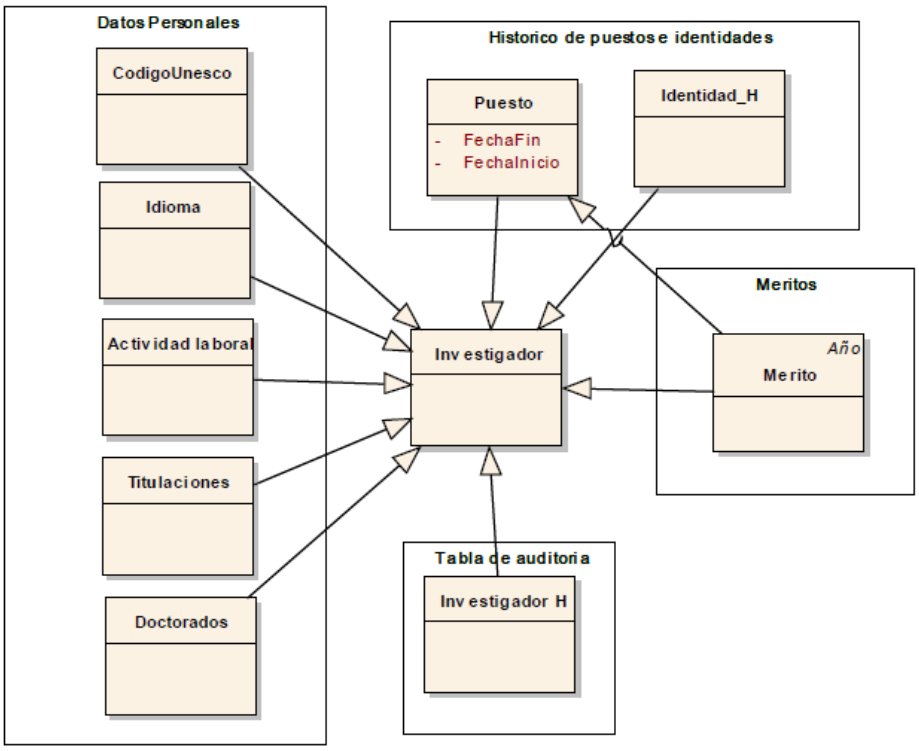
O Investigador é o núcleo onde se basea a estrutura da aplicación. Ó redor del vincúlanse os seus datos persoais e a súa produción científica.

Entre os seus datos persoais figuran:

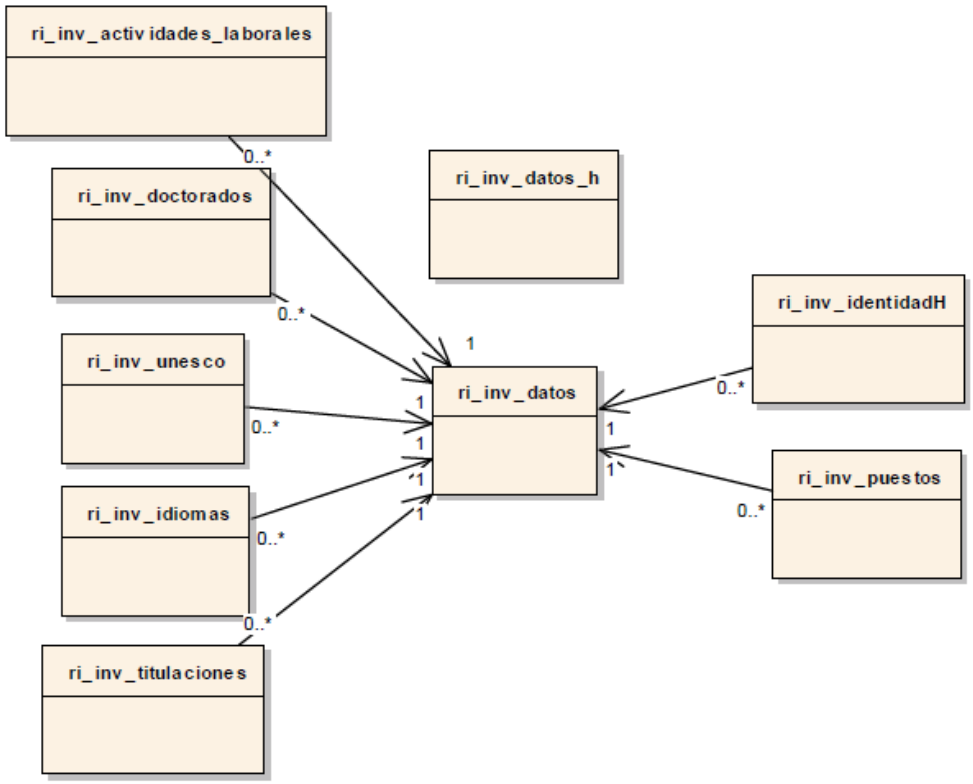
- Os idiomas que coñece
- Os seus doutorados
- As súas titulacións
- As súas actividades laborais
- Os Códigos Unesco cos que se relaciona

A aplicación mantén para cada obxeto dos que xestiona unha táboa de auditoría, mantendo unha copia de cada modificación do investigador. Pero a maiores para os investigadores a aplicación detecta cada cambio de posto ou identidade mantendo así unha listaxe de datos que si emprega de forma activa a aplicación, por exemplo na creación dun mérito.

Modelo de Dominio



Modelo de base de datos

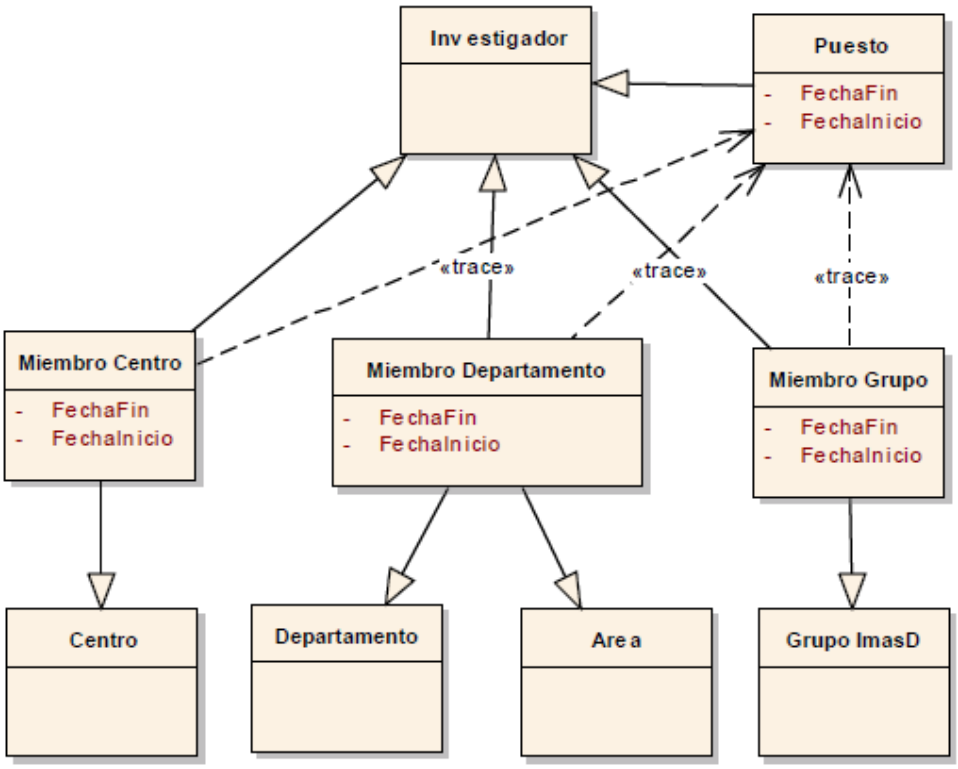


6.2 Membros de colectivos

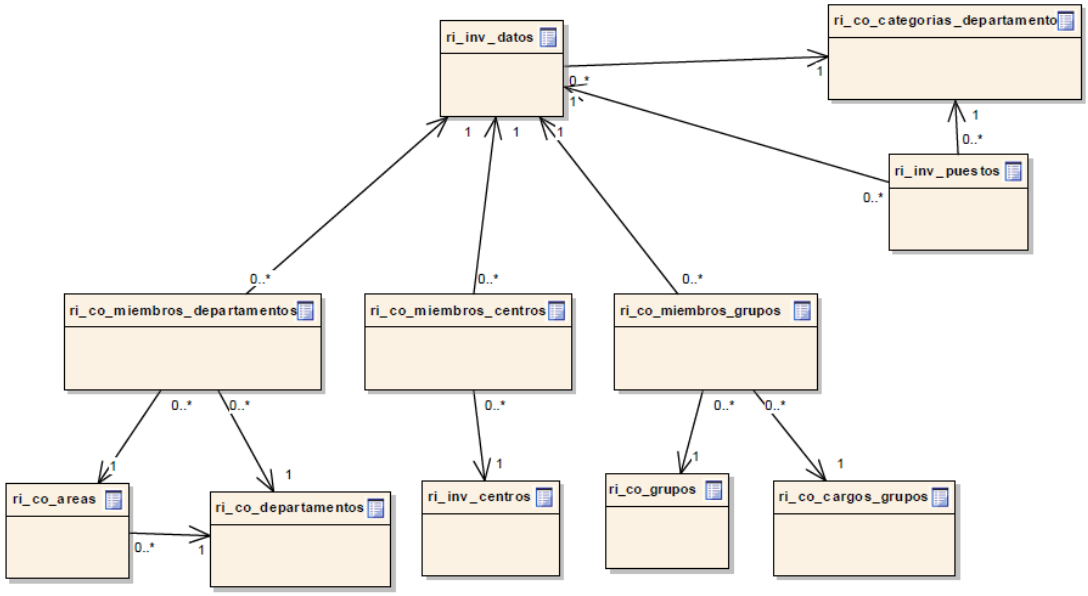
Na aplicación mantéñense datos de pertenza dos investigadores a Departamentos, Centros e Grupos de Investigación.

A continuación mostrase o modelo de dominio desta relación representando a relación entre a pertenza a un colectivo e o posto pola coincidencia entre as datas dos rexistros.

Modelo de Dominio



Modelo de Base de datos



6.3 Méritos

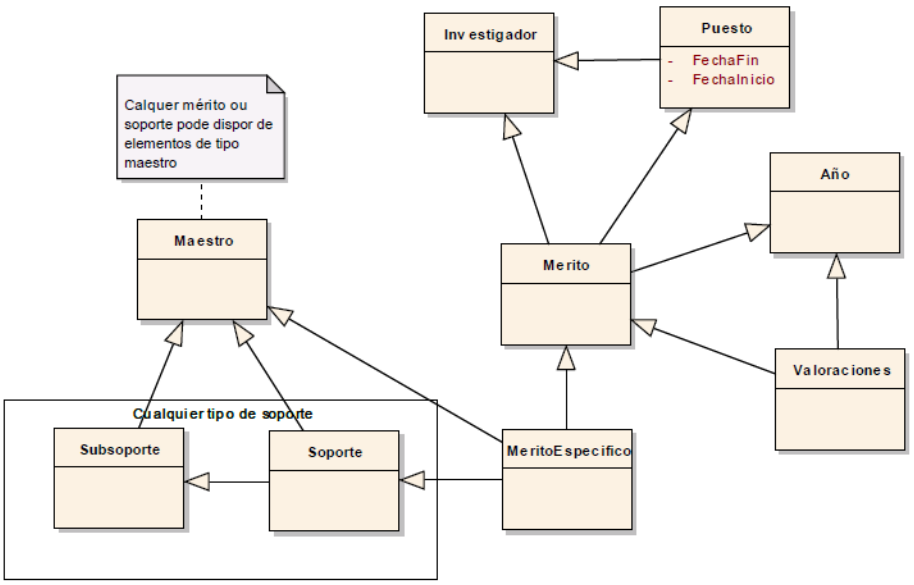
A continuación mostrase o modelo simplificado da estrutura dun mérito xenérico na aplicación. Un mérito pode dispor ou non de soporte (Libro, Capitulo, Congreso…), e a súa vez un soporte pode dispor de subsoporte (por exemplo un capitulo dispón do subsoporte Libro). Tamén tanto os méritos coma os soportes poden albergar datos xerais coma cidades, países, organismos, editoriais…

Un mérito asociase sempre a un Investigador e a un dos postos polos que pasou na Universidade, a un departamento, un área de coñecemento e a un grupo de investigación. Cada mérito atópase asociado a un ano, pero pode valorarse, tal e como se representa no modelo de dominio, en varios anos con diversas puntuacións.

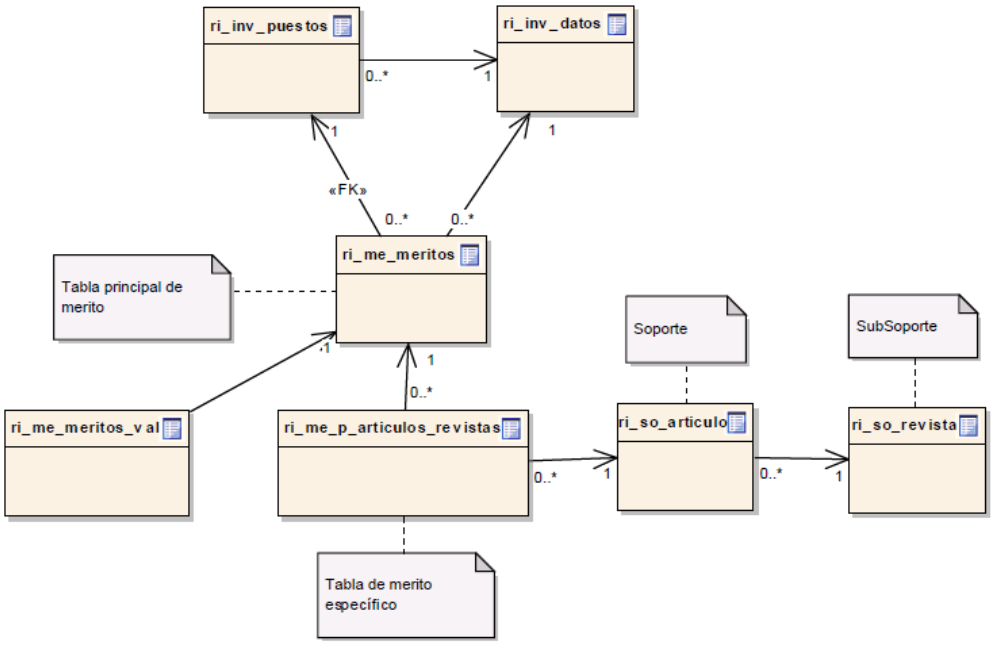
A aplicación mantén os seguintes tipos de méritos:

Tipo de Mérito	Soporte	Subsoporte
Participación en Proxectos o contratos de I+D	Proxecto ou contrato de I+D	
Participación en revistas	Revista	
Participación en Artículos en revistas	Articulo de revista	Revista
Participación en Libros	Libro	
Participación en Capítulos de libros	Capitulo de libro	Libro
Participación en Documentos científico-técnicos	Documento científico-tecnico	
Participación en Patentes	Patente	
Estancias en centros de investigación		
Participación en congresos	Congreso	
Participación en Comunicacións en congresos	Comunicación en congreso	Congreso
Participación en Tesis doctorales	Tesis doutoral	
Participación en Tesis de licenciatura / proyectos de fin de carrera	Tesis de licenciatura/pfc	
Xestión en actividades de I+D		
Participación en comités	Comité científico	
Participación en exposicións	Exposición	
Participación en concursos/certámenes	Concurso	
Premios a la investigación		
Quinquenios		
Sexenios de investigación		
Participación na Creación de empresas de base tecnológica	Empresa de base tecnolóxica	
Outros méritos		

Modelo de dominio



Modelo de base de datos

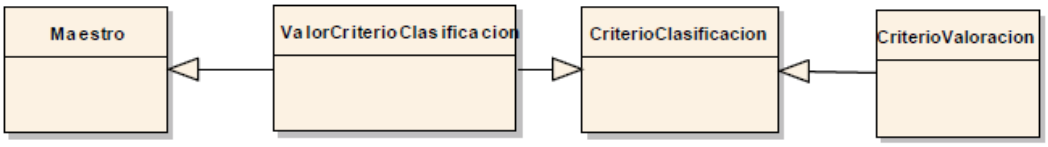


6.4 Mestres e criterios de clasificación e avaliación

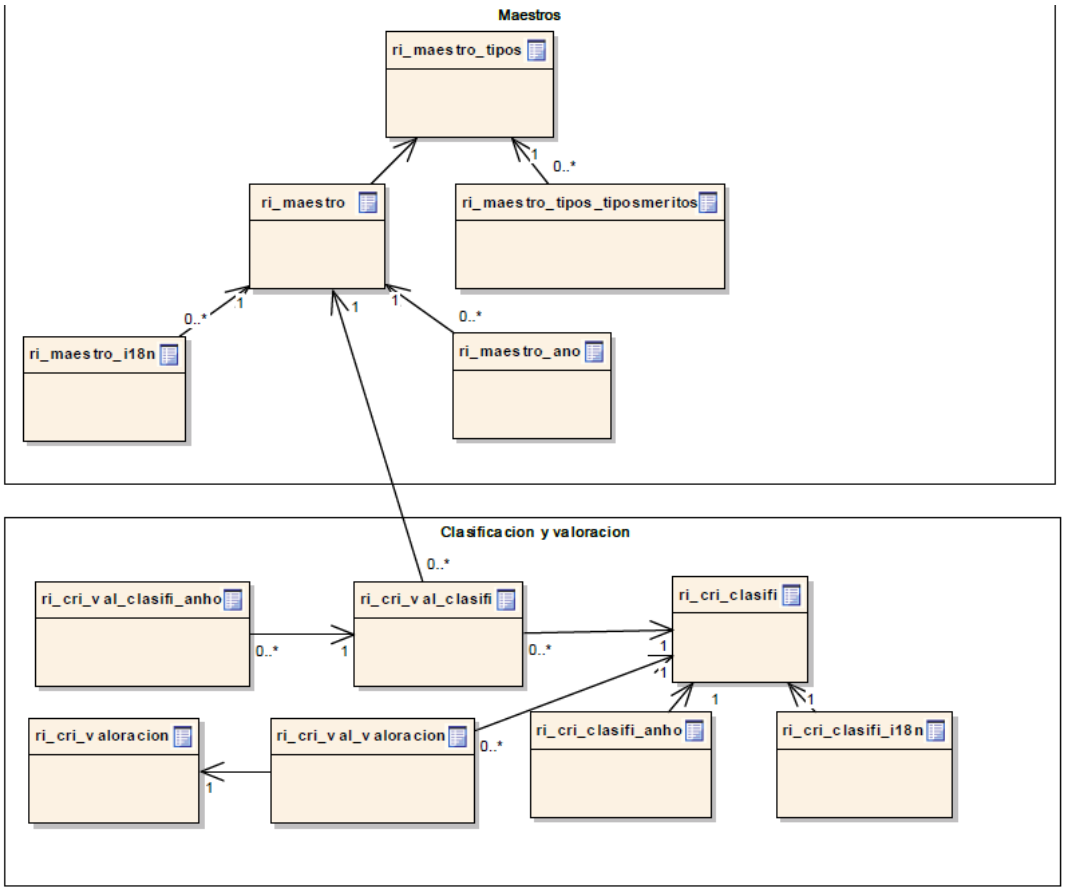
Cada mérito ou soporte pode dispor entre os seus datos de determinados campos tabulados na táboa mestre. Esta táboa é de ámbito xeral e pode dispor de datos tipificados de diversa índole e os seus valores poden asociarse a determinados anos.

A partir destes datos tabulados a aplicación pode autoclasificar ós méritos, e a partires de determinados criterios de clasificación (agrupación de valores da táboa de mestres) asignar determinadas valoracións ós méritos sen a necesidade de clasificar mérito a mérito.

Modelo de dominio



Modelo de base de datos



7 Autenticación e autorización

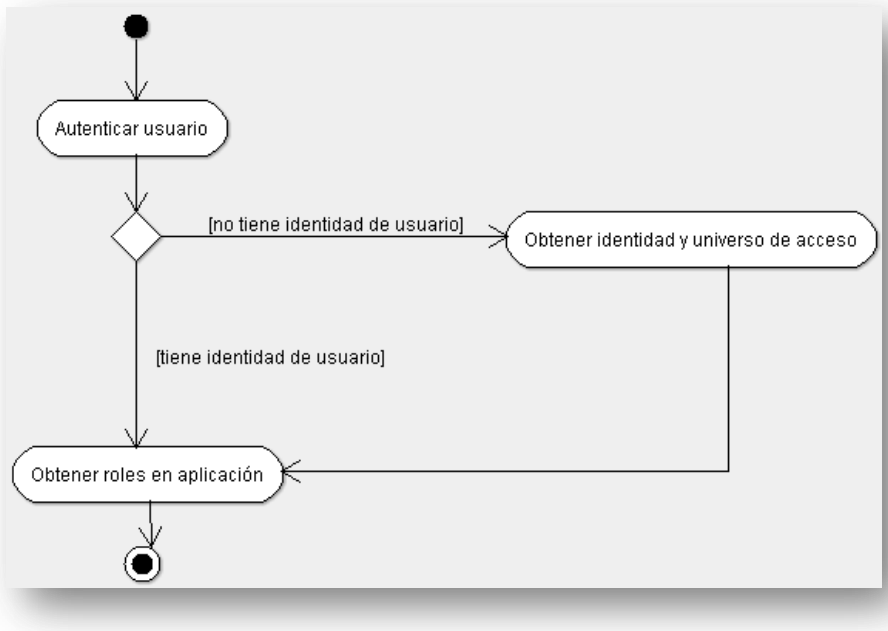
A autenticación reside nos sistemas dispostos polas universidades para tal efecto. Na actualidade existen 3 escenarios independentes por parte dos tres membros do CIXUG:

- UDC:
  - Ofrece os perfiles e departamentos accesibles polo usuario no momento da autenticación como datos dentro dun "Tícket CAS" devolto nese proceso.
- UVIGO:
  - Dispón dun sistema propio de autenticación
- USC:
  - Ticket CAS con algún dato persoal pero sen datos de autorización.

Ante esta casuística a aplicación permite múltiples escenarios implementados todos eles sobre o API de Spring Security. Para cada Universidade disponse dun mecanismo de autenticación que, analiza a resposta do sistema contra o cal se autentica e obtén os perfís do usuario. A partir destes perfís a aplicación obterá uns roles internos que serán os que definan o acceso as distintas unidades funcionais da aplicación.

Distínguense dous casos claros representados no seguinte diagrama:

- sistema de autenticación aporta información do perfil do usuario
- sistema de autenticación **non** aporta información do perfil do usuario.

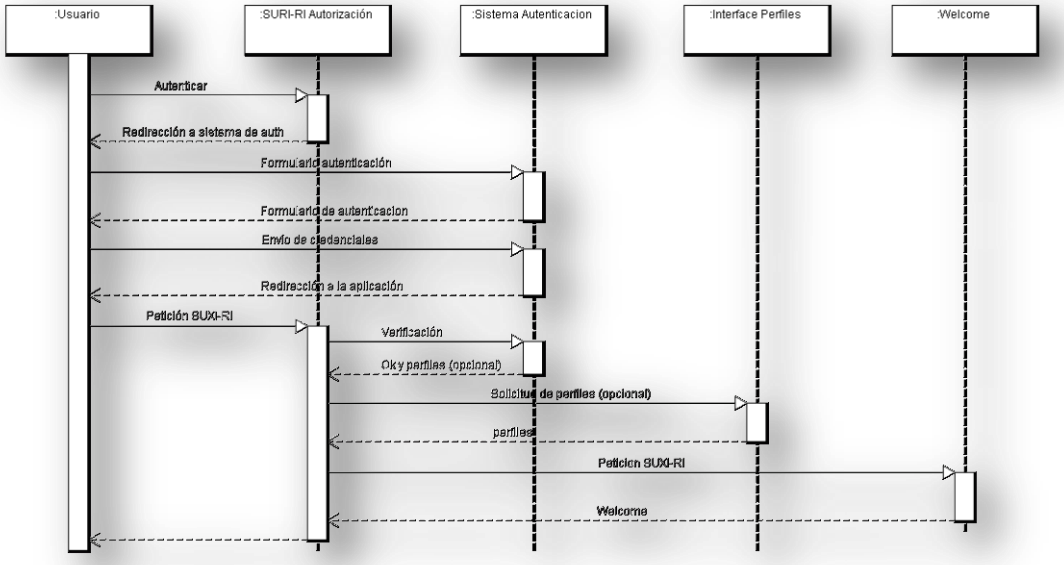


Se os datos necesarios (Identidades e Universo de acceso) non son ofrecidos no momento da autenticación obteranse a partir de peticións despois da mesma.

As peticións faranse contra unha interface común deseñada para tal efecto. Dita interface disporá de múltiples implementacións segundo a universidade sobre a que se atope instalada a aplicación. Seleccionarase unha implementación ou outra dependendo da universidade e si se dá o caso sobre o entorno sobre o que se realiza o despregamento.

A implementación desta interface será opcional no caso de que os perfiles sexan facilitados no "Tícket CAS" de autenticación, tal e como sucede actualmente co "Tícket CAS" da UDC.

A continuación preséntase un diagrama de secuencia de como sería o proceso dunha petición de autenticación.



1. O usuario realiza unha petición a SUXI-RI de autenticación ou para que necesita estar autenticado.
2. O módulo de autorización o redirixe ao sistema de autenticación propio da universidade.
3. O usuario cubre o formulario e autenticase.
4. O sistema de autenticación o devolve á aplicación.
5. A aplicación verifica que a autenticación é válida e recibe e analiza os datos da mesma. Este é un dos puntos onde se poden obter os perfiles do usuario (por exemplo para a UDC).
6. O módulo de autorización da aplicación chama á implementación dispoñible da interface de obtención de perfiles para obter os datos necesarios (perfiles e universo de acceso se non se dispón deles xa, esta sería unha chamada opcional).
7. O módulo de autorización permite a entrada á aplicación finalmente.

Unha vez obtidos os perfís do usuario autenticado a aplicación cotexa estes perfís cos roles internos da aplicación, obtendo así unha listaxe de unidades funcionais ás que pode acceder. As relacións entre os perfís do usuario e os roles internos da aplicación son administrables pola mesma, pero as unidades funcionais ás que da acceso albérganse nun arquivo XML, que é modificable e parametrizable por universidade se fose preciso.

Os roles internos que soporta a aplicación son:

- Investigador
- Coordinador de grupo
- Director de departamento
- Persoal dos servizos de investigación
- Persoal da Vicerreitoría de investigación
- Administrador do sistema

8 Xestión da configuración

Como xa se comentou SUXI-RI é un proxecto multimodular Maven. Maven encargase de xestionalas dependencias da aplicación e de dar forma ó proxecto. Cada un dos módulos dispón do seu propio tipo de empaquetado, concretamente o modelo e a lóxica dispoñen dun empaquetado “jar” e o módulo webapp dispón dun empaquetado “war” que depende dos módulos anteriores.

No momento de xerar o arquivo de despregue, labor que realiza Maven, debe indicarse o perfil para o cal se xera. Disponse dos seguintes perfís:

- udc
- usc
- uvigo

Con estes perfís pódese parametrizar calquera propiedade da aplicación coma o nome dos datasources, o nivel de log, indicar para que universidade se realiza o despregue, parametrizar o sistema de autenticación …

Ademais diso tamén é factible a parametrización dos recursos da aplicación, tanto recursos web como arquivos de configuración propios dunha aplicación web. Deste xeito permítese dispor de distintas plantillas de imaxe para as universidades, informes ca imaxe corporativa e configuracións totalmente diferentes entre os sistemas.

Obtense así unha solución flexible para o mantemento dunha mesma aplicación en entornos diversos coma é o caso de SUXI-RI.

9 Ferramentas de software libre empregadas para a implementación da solución

9.1 Introducción

Un dos alicientes fundamentais do acceso ás tecnoloxías da información é software e, a pesar disto, o acceso ao software vese restrinxido por políticas diversas en forma de contratos, licencias ou patentes que limitan o uso e discriminan a algúns sectores da cidadanía discriminando o prohibido o acceso ao software en condicións legais e cunhas garantías mínimas de calidade.

Dende fai anos existe un movemento internacional de persoas e entidades que desenrolan software baixo unhas licencias que defenden que, cando se adquire un programa, o usuario ten dereito a estudalo, modificalo e distribuílo libremente. O software que garante estes dereitos é coñecido como “software libre”.

O software libre permite que todos os seus usuarios e usuarias poidan axudar a súa evolución e mellora. Son software libre o sistema GNU/Linux, diferentes navegadores Web como “Firefox”, paquetes ofimáticos como o OpenOffice.org o Koffice, os servidores que utiliza Google e outros moitos programas de gran calidade.

No software libre hai millóns de voluntarios/as, e tamén grandes empresas multinacionais como IBM, HP, Sun, etc.

Utilizar software libre ten moitas vantaxes:

- Quén o compra non ten que ser cliente indefinidamente do seu provedor xa que pode contratar a quen elixa libremente para realizar as modificacións ou adaptacións segundo as súas necesidades. Se existe un erro na aplicación, non se depende da empresa en cuestión para solucionalo.
- Non discrimina a ninguén. Ata os provedores de software privado poden ofrecer software libre, cambiando a licenza do seu produto ou adaptando o software libre existente.
- A maioría do software libre é gratuíto ou de baixo custo, ao contrario que o software privado.
- acceso ao código permite a súa análise para, por exemplo, comprobar que non contén código espía ou malicioso.
- Nos permite participar en proxectos internacionais punteiros, e polo tanto potenciar unha industria local de desenvolvemento de software.

A continuación faise unha breve descrición das principais ferramentas de software libre empregadas para a desenvolvemento do sistema.

9.2 Implementación do modelo vista-controlador: Struts

Para a implementación de modelo vista-controlador utilizase o "FrameWork" JAVA Struts[1].

"Struts" é un "FrameWork" para construír aplicacións Web, baseado no patrón de deseño MVC e implementado coa tecnoloxía Java e J2EE. Este patrón de deseño, levado ao dominio das aplicacións Web, é coñecido tamén como "Modelo 2".

Dentro da arquitectura implementa a parte do controlador e ofrece unha serie de compoñentes e librerías de "tags" para facilitar a creación das vistas mediante "JSP's". No obstante, permite o uso de calquera alternativa na presentación. Ademais, non presupón nada acerca do modelo. Isto é desexable, xa que o modelo da aplicación debe ser independente do cliente que se vaia a utilizar.

A utilización de "Struts" é recomendable por:

- É un "FrameWork" desenrolado por expertos.
- É estable e maduro.
- Ten unha curva de aprendizaxe suave e manexable.
- É "Open Source", o cal significa que o 100% do código fonte está a nosa disposición.
- Existe una ampla documentación.

Dende o punto de vista do programador, "Struts" ofrece:

- Unha funcionalidade rica en características e servizos para o desenvolvemento de unha aplicación Web, o que simplifica o tratamento dos problemas derivados deste tipo de aplicacións, e permite fixar o foco no negocio, no problema real que hai que resolver.
- É software libre, tanto para o desenvolvemento como para o despregue da aplicación.
- Está soportado por multitude de ferramentas de terceiros.
- Integración coas tecnoloxías J2EE.
- Ampla comunidade de usuarios.
- Flexibilidade, posibilidade de personalización e de extensión.
- Eficiencia e rendemento.

Entre as características mais importantes de "Struts" podemos citar:

- Implementación do patrón MVC. Modelo 2 en arquitectura de aplicacións Web.
- Soporte para a internacionalización de aplicacións.
- Ampla librería de "Tags" para "JSP's".
- Baseado en "JSP", "Servlet", Java e XML.
- Filosofía "Write Once, Run Anywhere" de Java.
- Soporta diferentes implementacións do modelo ("JavaBeans", "EJB", etc.)
- Soporta diferentes implementacións para a presentación (JSP, XML/XSLT, etc.)

A interface de usuario se dividirá nun conxunto de unidades funcionais, desta forma será posible adaptar esta interface ás accións que o usuario poderá realizar sobre o sistema, evitando mostrar funcionalidades e accións que non estean permitidas para un determinado usuario.

[1]Para máis información pode consultar http://struts.apache.org/

9.3 Rexistro de logs:Log4J

Para o tratamento de mensaxes de monitorización do sistema lanzados dende a aplicación utilizase a API Log4J[2] do proxecto Jakarta, que ten as seguintes características:

- Altamente configurable con arquivos XML
- Saída múltiple non excluinte:
  - Arquivos
  - Base de Datos vía JDBC
  - Correo
- Formato de saída configurable mediante expresións estándar. Algunhas destas expresións son:
  - d : Fecha
  - -nr: XXXX Hora
  - t : threath
  - p :Prioridade
  - c :Categoría
  - m : Mensaxe
  - (%F:%L) Liña
- Distintas categorías de erro. Por defecto DEBUG, INFO, WARN, ERROR, FATAL con posibilidade de definir funciones propias.
- Pode ser invocado a nivel de servizo de servidor de aplicacións e a nivel de aplicación Web, sendo as eu comportamento configurable en cada caso.
- O uso desta API e os seus arquivos de configuración permitirá ao programador invocar o "log" sobre o que quere actuar e escribir nel as mensaxes necesarias dentro de la aplicación.
- Dado que as mensaxes deben ser posteriormente tratados e a API o permite, as mensaxes se cargarán de forma automática en Base de Datos en tempo de execución, lo que posibilitará o acceso e traspaso das mesmas a históricos.

[2]Para máis información pode consultar http://jakarta.log4j.org/

9.4 Xeración de informes: Jasper Reports

Jasper Resports[3] é un poderoso "framework open source" para a xeración de informes presentando gran flexibilidade na organización e presentación de contido. Permite a xeración dinámica de informes en diferentes formatos como PDF, HTML, XLS, CSV e XML.

Ademais, esta ferramenta converteuse no últimos tempos en practicamente un estándar de programación o que provocou o nacemento de numerosas ferramentas de software libre encamiñadas a facilitar as labores de deseño de informes, permitindo a seu mantemento por persoal sen unha forte compoñente técnica.

Funcionamento

O deseño dun informe con esta ferramenta, incluíndo a localización dos campos para ser cumprimentados con valores dinamicamente, está baseado nun arquivo en formato XML que permite definir:

- Textos estáticos.
- Imaxes.
- Liñas.
- Formas xeométricas.
- Campos que serán cumprimentados dinamicamente a partir dunha fonte de datos.

Diferentes obxectos de "Jasper Reports" son usados para representar as etapas do proceso de xeración do informe:

- JasperDesing: representa a definición del informe.
- JasperReport: representa un "JasperDesing" compilado.
- JasperPrint: representa un informe creado.

Fontes de datos para a xeración de informes

Con esta ferramenta se permite la xeración de informes (JasperPrint) apartir dun deseño creado (JasperReport) extraendo os datos de diferentes fontes:

- Bases de datos.
- Arquivos XML.
- Obxectos Java.
- Etc.

9.5 Aplicacións multiidioma:i18n

i18n[4] é unha librería de "tags" que permite manexar a complexidade de creación e mantemento de aplicacións internacionalizadas.

A través da utilización desta librería se conseguen os seguintes obxectivos:

[3]Para máis información pode consultar http://jasperforge.org/sf/projects/jasperreports
[4]Para máis información pode consultar http://jakarta.apache.org/taglibs/doc/i18n-doc/intro.html

- Independizamos a internacionalización da aplicación da lóxica de negocio: ao utilizar os “tags” dende os propios “jsps” encargados da presentación.
- Contamos con mecanismos automáticos de formateo de mensaxes para construír datas, números, etc.
- A fonte de datos poderá estar en diferentes fontes de datos: arquivos de propiedades, base de datos, etc, aínda que o máis habitual é xerar arquivos de propiedades onde se manterán os textos internacionalizados.
- Esta librería está integrada dentro do “framework” de desenrolo “Struts”.

9.6 Hibernate

Hibernate[5] é un marco de traballo de mapeo O/R Open Source que evita a necesidade de utilizar a API JDBC.

Hibernate soporta a maioría dos sistemas de bases de datos SQL. O “*Hibernate Query Language”*, deseñado como unha extensión mínima, orientada a obxectos de SQL, proporciona unha ponte elegante entre os mundos obxecto e relacional.

Adicionalmente ofrece facilidades para recuperación e actualización de datos, control de transaccións, repositorios de conexións a bases de datos, consultas programáticas e declarativas, e un control de relacións de entidades declarativas.
Hibernate é menos invasivo que outros marcos de traballo de mapeo O/R. Se utilizan “Reflection” e a xeración de bytecodes en tempo de execución, e a xeración do SQL ocorre no momento da arrancada. Isto nos permite desenrolar obxectos persistentes seguindo a linguaxe común de Java: incluíndo asociación, herdanza, polimorfismo, composición e o marco de traballo Collections de Java. Os obxectos de negocio das aplicacións son POJO e non necesitan implementar ningún interface específico de Hibernate.

[5]Para máis información pode consultar http://www.hibernate.org/

9.7 Spring Framework

Spring[6] é un framework de aplicacións Java/J2EE desenrolado usando licenza OpenSource que ten un conxunto de características que o fan o seu uso moi interesante en aplicacións baseadas en arquitectura J2EE como as que se describen a continuación:

- Esta baseado nunha configuración a base de "javabeans" bastante simple.
- É potente en canto á xestión do ciclo de vida dos compoñentes e facilmente ampliable.
- Destaca a funcionalidade que permite o uso de programación orientada a aspectos (IoC).
- Ten modelos que permiten un uso máis fácil de *Hibernate*, iBatis, JDBC..., e se integra "de fábrica" cos Frameworks Quartz, Velocity, Freemarker, Struts, Webwork2.
- Dispón dun plugin para utilizar co IDE de desenrolo Eclipse.
- Ofrece un contenedor de bean para os obxectos da capa de negocio, *DAOs* e repositorio de *Datasources JDBC* e sesións *Hibernate*. Mediante un *xml pódese definir* o contexto da aplicación sendo unha potente ferramenta para manexar obxectos Singleton ou "factorías" que necesitan a súa propia configuración.
- obxectivo de *Spring* é non ser intrusivo, aquelas aplicacións configuradas para usar beans mediante *Spring* non necesitan depender de interfaces ou clases de *Spring*, pero obteñen así a configuración a través das propiedades dos seus "beans". Este concepto pode ser aplicado a calquera entorno, dende unha aplicación J2EE a un *applet*. Como exemplo podemos pensar en conexións a base de datos ou de persistencia de datos, como *Hibernate*.

Coa utilización de Spring no desenvolvemento da aplicación conseguirase a separación dos accesos a datos e los aspectos relacionados coas transaccións, para permitir obxectos da capa de negocio reutilizables que non dependan de ningunha estratexia de acceso a datos con transaccións.

Spring ofrece una maneira simple de implementar DAOs baseados en *Hibernate* sen necesidade de manexar instancias de sesión de *Hibernate* ou participar en transaccións. Non necesita bloques "try-catch", innecesario para o chequeo de transaccións. Pódese acadar un método de acceso simple a *Hibernate* con una soa liña de código fonte.

Que proporciona

Spring proporciona:

- Unha potente xestión de configuración baseada en JavaBeans, aplicando os principios de Inversión de Control (*IoC*). Isto permite que a configuración de aplicacións sexa rápida e sinxela ao non ser necesario ter "*singletons"* nin arquivos de configuración. Estas definicións de "*beans"* se realizarán no propio contexto da aplicación.
- Unha capa xenérica de abstracción para a xestión de transaccións, permitindo xestores de transacción engadibles e facendo sinxela a demarcación de transaccións sen tratalas a baixo nivel.
- Unha capa de abstracción JDBC que ofrece una significativa xerarquía de excepcións (evitando a necesidade de obter de SQLException os códigos que cada xestor de base de datos asigna aos erros), simplifica o manexo de erros e reduce considerablemente a cantidade de código necesario.
- Integración con *Hibernate*, JDO e iBatis SQL Maps en termos de soporte a implementacións DAO e estratexias con transaccións. Ten un soporte especial a *Hibernate* engadindo características de *IoC*, e solucionando moitos dos problemas comúns problemas de integración de *Hibernate*. Todo isto cumprindo coas transaccións xenéricas de *Spring* e a xerarquía de excepcións DAO.

[6] Para máis información pode consultar *http://www.springsource.org/*

- Funcionalidade AOP, totalmente integrada na xestión de configuración de *Spring*. Pódese aplicar AOP a calquera obxecto xestionado por *Spring*, engadindo aspectos como xestión de transaccións declarativa.
- Unha capa modelo realizada con *Spring* pode ser facilmente utilizada con unha capa web baseada en Struts.

Toda esta funcionalidade pode usarse en calquera servidor J2EE, e la maioría dela nin sequera require a seu uso. O obxectivo principal de *Spring* é permitir que obxectos de negocio e de acceso a datos sexan reutilizables, non atados a servizos J2EE específicos. Estes obxectos poden ser reutilizados tanto en entornos J2EE (Web ou EJB), aplicacións "standalone", entornos de probas, etc.… sen ningún problema.

¿Qué é Ioc?

Spring baséase en IoC que consiste en:

- Un "Contenedor" que manexa obxectos polo programador.
- contenedor xeralmente controla a creación destes obxectos. Por dicilo de algunha maneira, o contenedor fai os "new" das clases java para que non os realice o programador.
- contenedor resolve dependencias entre os obxectos que conten.

9.8 Xestión de plantillas: Tiles

O "framework" Tiles[7] proporciona un mecanismo que permite separar a estrutura da páxina (de agora en diante "layout") do contido, proporcionando a habilidade de establecer unha estrutura e dinamicamente inserir os contidos das páxinas nesta estrutura de páxina (layout) no momento da xeración da pantalla (execución). Este mecanismo é moi poderoso para a adaptación do aspecto ou para a internacionalización.

O "framework" Tiles proporciona:

- Capacidades de plantilla.
- Construción e carga dinámica de páxinas.
- Definición de pantallas.
- Reutilización dos compoñentes dos layouts.
- Internacionalización

Un Tile é un área ou rexión dunha paxina web. A rexión pode ser a páxina web enteira, ou a páxina pode ser dividida en varias rexións.

Una paxina JSP consta de varias rexións ou Tiles. Non hai nada especial acerca da páxina salvo que está pensada para usarse co "framework" de Tiles e fai uso da librería de "tags" desta ferramenta.

A vantaxe fundamental do uso dun Tile é que se trata de un compoñente JSP que é reutilizable. Un layout en Tiles é o que comunmente entendemos como plantilla. Un layout se constitúe por un grupo de Tiles ou áreas para constituír una páxina. De feito a figura anterior é un layout, composto de varios Tiles (cabeceira, menú,....).

[7] Para máis información pode consultar http://struts.apache.org/1.x/struts-tiles/

O Layout mostrado na figura anterior non coñece nada da páxina que o contén. Desta forma, podería reempregarse este mesmo layout noutra área (ou tile). O non estar ligado o layout ao contido, este pode ser reempregado en múltiples paxinas.

Como se desprende dos exemplos anteriores, o uso desta ferramenta permite:

- Definicións de páxinas.
- Centralización das declaracións de páxinas
- Utilizar a herdanza ou extensión das definicións, o que evita a redundancia ou repetición de paxinas similares.
- Evitar a creación de compoñentes intermedios para o paso de parámetros.
- Usar definicións como vistas de struts.
- Usar o nome de definicións como un parámetro para un compoñente.
- Sobrescribir definicións de atributos.
- Internacionalización.

9.9 Invocación de Servizos Web: CXF

Apache CXF[8] é un framework de servizos de software libre. Facilita a construcción e desenrolo de servizos empregando API's de programación coma JAX-WS y JAX-RS. Ditos servizos poden comunicarse en diferentes protocolos coma SOAP, XML/HTTP, RESTful HTTP ou CORBA e traballar sobre varios protocolos de trasporte coma HTTP, JMS ou JBI.

A súa integración con Spring é moi sinxela e permite a invocación e publicación de servizos Web de modo moi cómo e sinxelo para o programador.

[8] Para máis información pode consultar http://cxf.apache.org/index.html