Proyecto Fin de Carrera

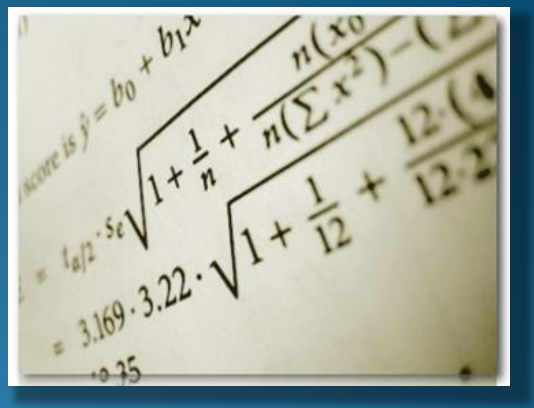
Diseño e Implementación de un Framework de Presentación

Autor: Daniel Rodríguez Simó

Tutor: Óscar Escudero Sánchez

UOC, 14 de Enero de 2013

Índice



- Objetivos – Planificación.
- Justificación.
- Patrones de Diseño: *Patrón MVC.*
- Estudio/Comparativa de Frameworks actuales.
- Framework de Presentación "*FUOC*".
- Aplicación Web "*Club Ciclista UOC*".
- Resultados / Productos obtenidos.
- Mejoras / Evoluciones.

Planificación

- ***Fase1 → Propuesta del proyecto*** (*Pec1*):
  - Plan de Trabajo, fijando objetivos generales y específicos.
  - Definición de las subtareas a realizar y Tiempos.

- ***Fase 2 → Análisis y Diseño*** (*Pec2*):
  - Visión general de los Frameworks del Mercado.
    - Struts2 / Tapestry / Grails.
  - Comparativa y conclusiones.

- ***Fase 3 → Implementación*** (*Pec3*):
  - Análisis y Diseño de la aplicación.
  - Implementación (Framework y Aplicación).

- ***Fase 4 → Entrega Final:***
  - Productos obtenidos.
  - Conclusiones y Mejoras

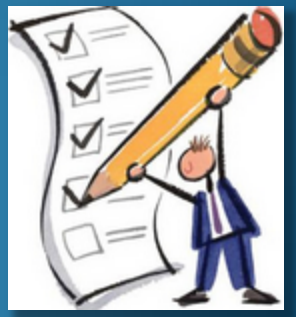
Principales Objetivos

- Conocer y analizar los Frameworks de Presentación actuales del mercado.
- Diseñar e implementar un Framework de Presentación.
- Construir y desarrollar una Aplicación Web que lo contenga.
- Analizar los resultados, mejoras y conclusiones obtenidas.

Frameworks Actuales ***Framework FUOC*** ***Aplicación Club Ciclista*** ***Resultados Obtenidos***





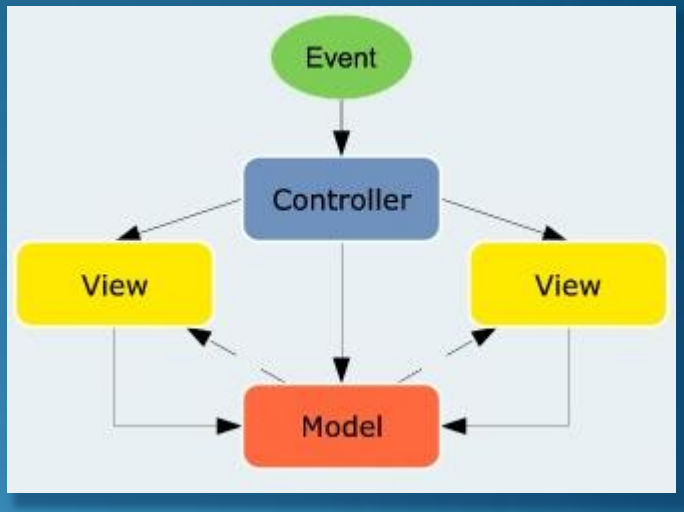


Justificación

- La Capa de Presentación tiene una importancia muy relevante en las Aplicaciones Web.
- J2EE es uno de los estándares más empleados del mercado.
- Ofrece un marco de trabajo y un conjunto de servicios sobre los cuales desarrollar aplicaciones en arquitecturas multicapa.
- Se incorporan el uso de patrones de diseño a la Ingeniería del Software (*Core J2EE Patterns, Gang of Four –GoF-, etc.*).
- El empleo de patrones proporcionará mayor agilidad en el diseño y desarrollo, estructurando el código, cediendo el resto de tareas al Servidor de Aplicaciones.

Patrón MVC

- El empleo de Patrones de Diseño ofrece soluciones a problemas comunes a los que se enfrenta la programación orientada a objetos durante el diseño de toda aplicación.



- Principales componentes:
  1. La Interfaz de Usuario.
  2. Lógica de Negocio.
  3. Los Datos.

- Separa la lógica del negocio de la interfaz de usuario, con lo que se consigue:

  ✓ Proporcionar un software mantenible.
  ✓ Facilitar la evolución por separado de ambos componentes (modular y poco acoplado).
  ✓ Incrementar la flexibilidad y aumentar su reutilización.

Framework de Presentación

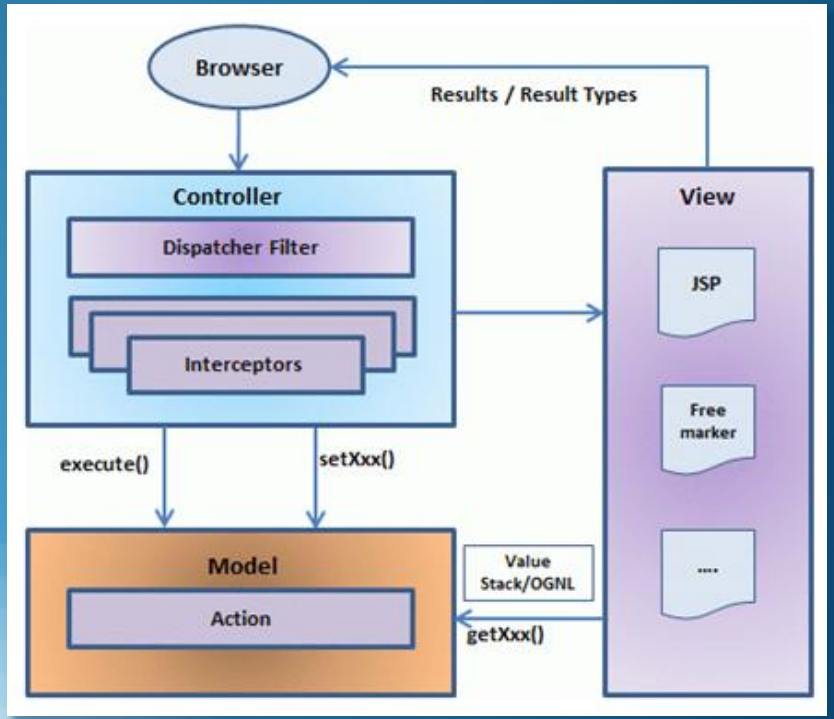
Struts 2

- Según el patrón MVC, el controlador lo implementa el Dispatcher Servlet Filter, encargado de recibir las peticiones y decidir la acción a ejecutar.

- El modelo se implementa el patrón Command, mediante clases Action, que ejecutando la lógica correspondiente y actuando como Transfer Object de los datos entre el cliente y la aplicación.

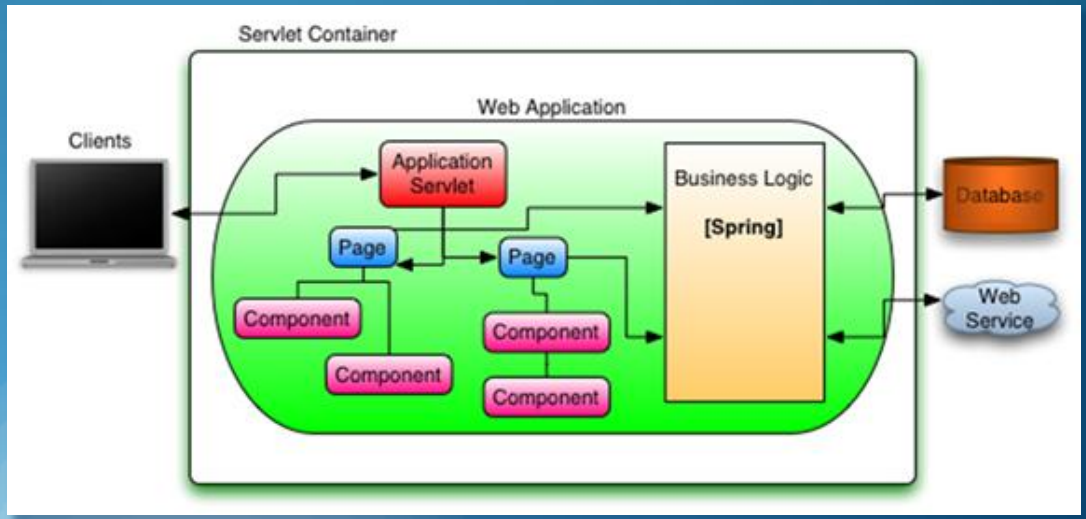
- La vista se construirá en función del Action ejecutado, contando con JSP, XSLT, Velocity, Free Marker, Tiles, etc.

- Configuración mediante:
  - Descriptores de Despliegue.
  - Anotaciones Java
  - Ficheros XML



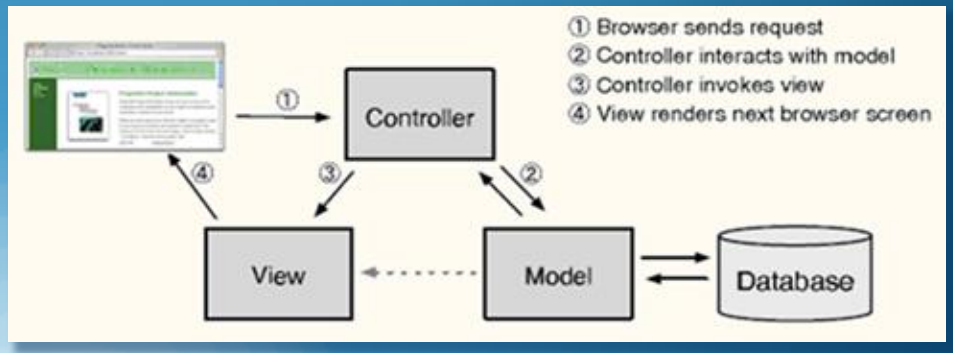
Framework de Presentación

- Complementa y construye desde el estándar Java Servlet API, funcionando también en cualquier servidor contenedor de servlets o contenedor de aplicaciones.
- Trabaja en términos de componentes y objetos, métodos y propiedades.
- Convierte los parámetros de la URL al tipo adecuado y los deja en una propiedad del componente de ese tipo.
- Implementa la arquitectura MVC, siguiendo un modelo basado en componentes y en el patrón de diseño FrontController.
- Integración con Hibernate, JPA, Spring, Lucene, etc.



Framework de Presentación

- Basado también en el patrón Modelo Vista Controlador (*MVC*).
- Los modelos son tratados como clases de dominio que permitirán a la aplicación mostrar los datos en la vista.
- El Dispatcher Servlet se empleará como Front Controller; Será un Servlet apoyado en uno o N patrones Mapper para determinar a qué Controller ha de llevarse cada petición del usuario.
- Se emplean, entre otros, los patrones IoC (*Inversión of Control*) y la Inyección de dependencias: Las dependencias no las gestionará el componente, de tal forma que sólo contenga la lógica necesaria.
- El controlador permitirá gestionar las peticiones a la aplicación y organizar los servicios proporcionados.



Comparativa de Frameworks

Struts 2	Tapestry	Grails
Basado en Framework J2EE (WebWork)	Basado en Framework J2EE	Basado en Framework J2EE
Orientado a acciones, URLs	Componentes	Acciones y Componentes
Config: XMLs y anotaciones Java	Config: Anotaciones Java)	Config: Ficheros groovy.property
Trabaja con URLs y parámetros	Trabaja con Componentes y Objetos	Trabaja con URLs y parámetros
Inyección de Dependencias	Inyección de Dependencias	Inyección de Dependencias
Controlador Servlet Filter	Controlador Servlet	Controlador Servlet
Vista: JSP, Velocity, FreeMarker, XSLT	Vista: En base a Plantillas	Vista: Ficheros GSP
Validación: En Cliente y Servidor	Validación: En Cliente y Servidor	Validación: En Cliente y Servidor
Autenticación: Interceptor	Autenticación: Spring Security	Autenticación: Spring Security
Arquitectura sencilla y extensible	Curva de aprendizaje alta.	Tareas de debugueo y trazas de error pobres
No componentes de interfaces usuario ni gestión de eventos para crear interfaces ricas	Gestión eficiente de recursos. Optimización en cuanto a CPU y Memoria	DRY (*Don´t Repeat Yourself*). Permite alta reutilización de código

Framework *FUOC*

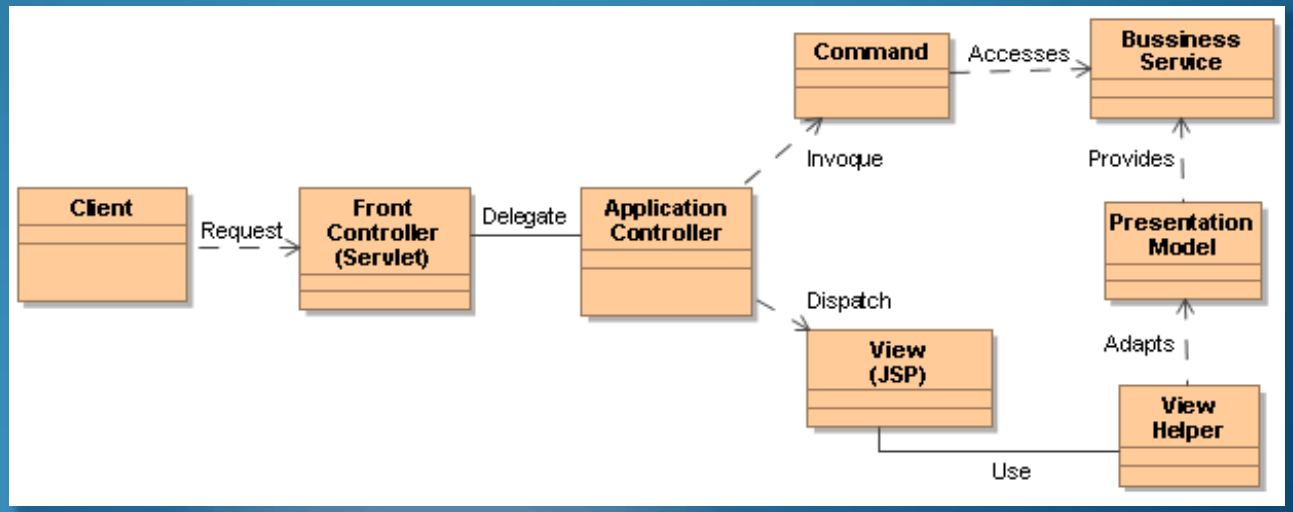
- Funcionalidades
- Patrones empleados
- Diseño
- Proceso de Inicialización/Configuración
- Mapeado de las Acciones
- Flujo de una Petición
- Estructura de Clases/Paquetes

FUOC: Funcionalidades

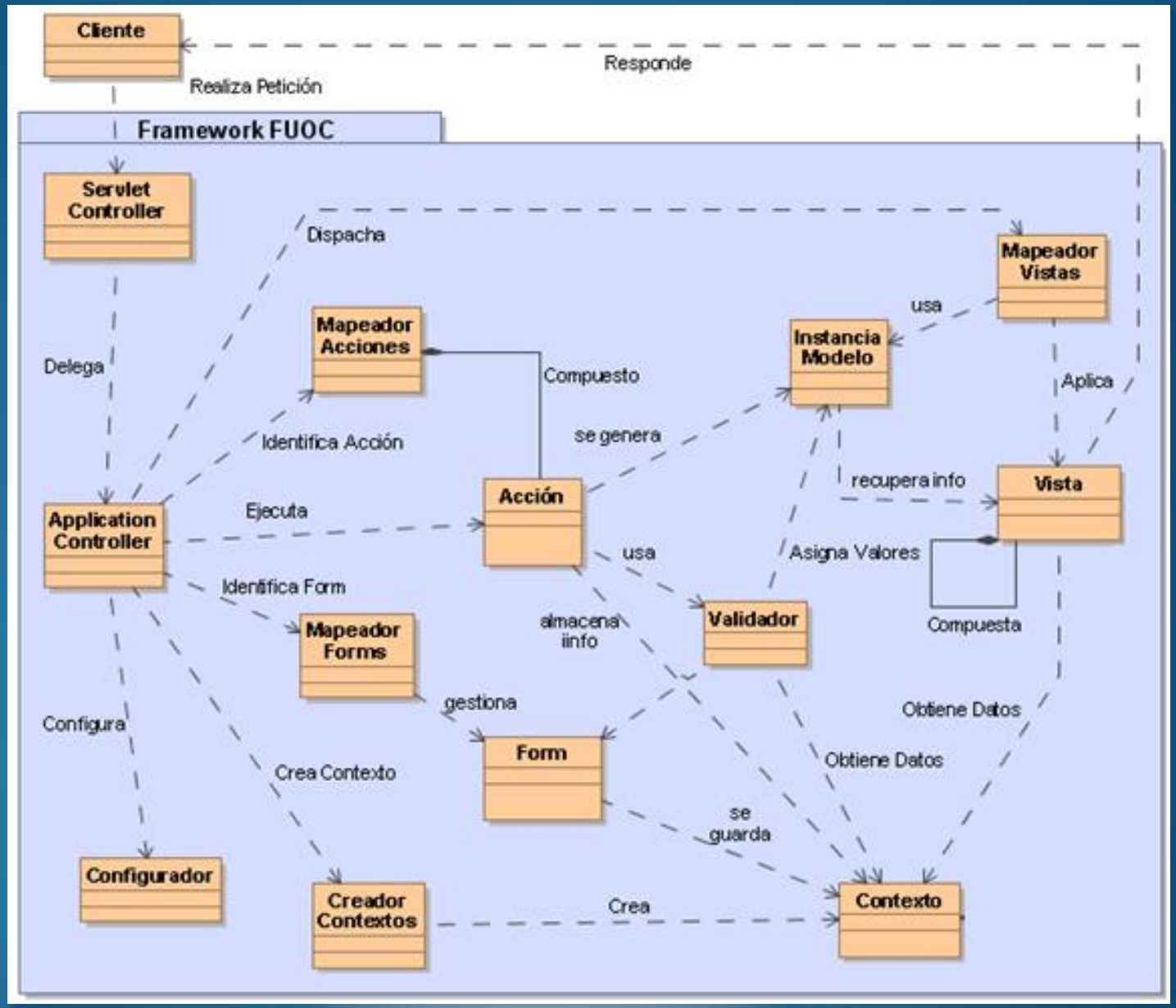
- Control del flujo de forma declarativa.
- Servicio de validación de datos.
- Servicio de internacionalización (multiidioma).
- Gestión de excepciones/errores.
- Gestión de Logs.
- Sesiones.
- Navegación.

FUOC: Patrones Empleados

- Patrón arquitectónico central del PFC: ***MVC***
- Otros Patrones destacables:
  - Patrón Front Controller
  - Patrón Application Controller
  - Patrón Service to Worker:
    - FrontController
    - Application Controller
    - View Helper
  - Patrón Command
  - Patrón Context Object
  - Patrón Singleton
  - Patrón Composite View

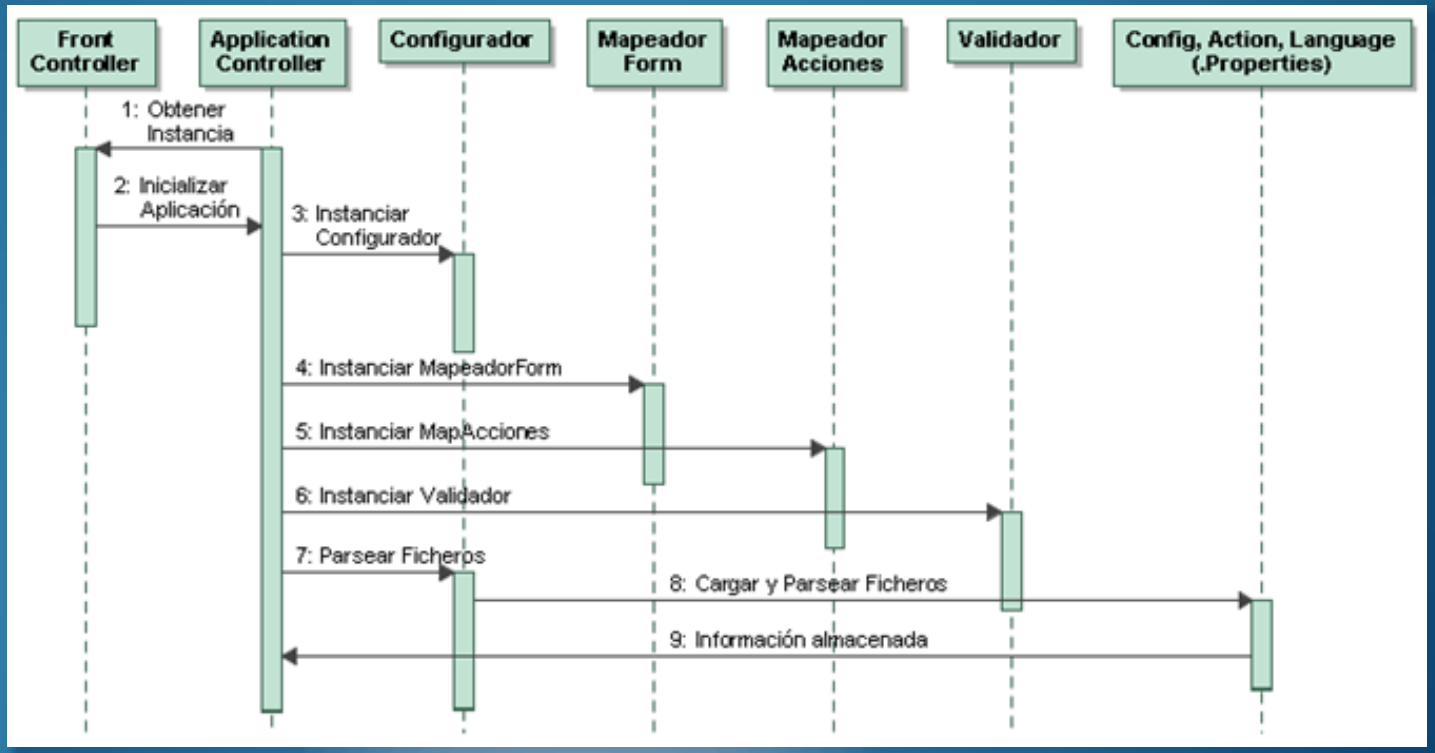


FUOC: Diseño *(Diagrama de Clases)*



FUOC: Inicialización / Configuración

- Front Controller delega todo el proceso de inicialización y configuración al Application Controller.
- Se cargan/gestionan los ficheros Init *actionsFile* y *configFile.properties*:
  - Acciones de la Aplicación (Action + Form + Scope + Next Ok + Next Error).
  - Servicio de Trazas ("*modo verbose*" on/off).
  - Idioma



FUOC: Mapeado de Acciones

- Información proveniente del parseo de *ActionFile.properties:*

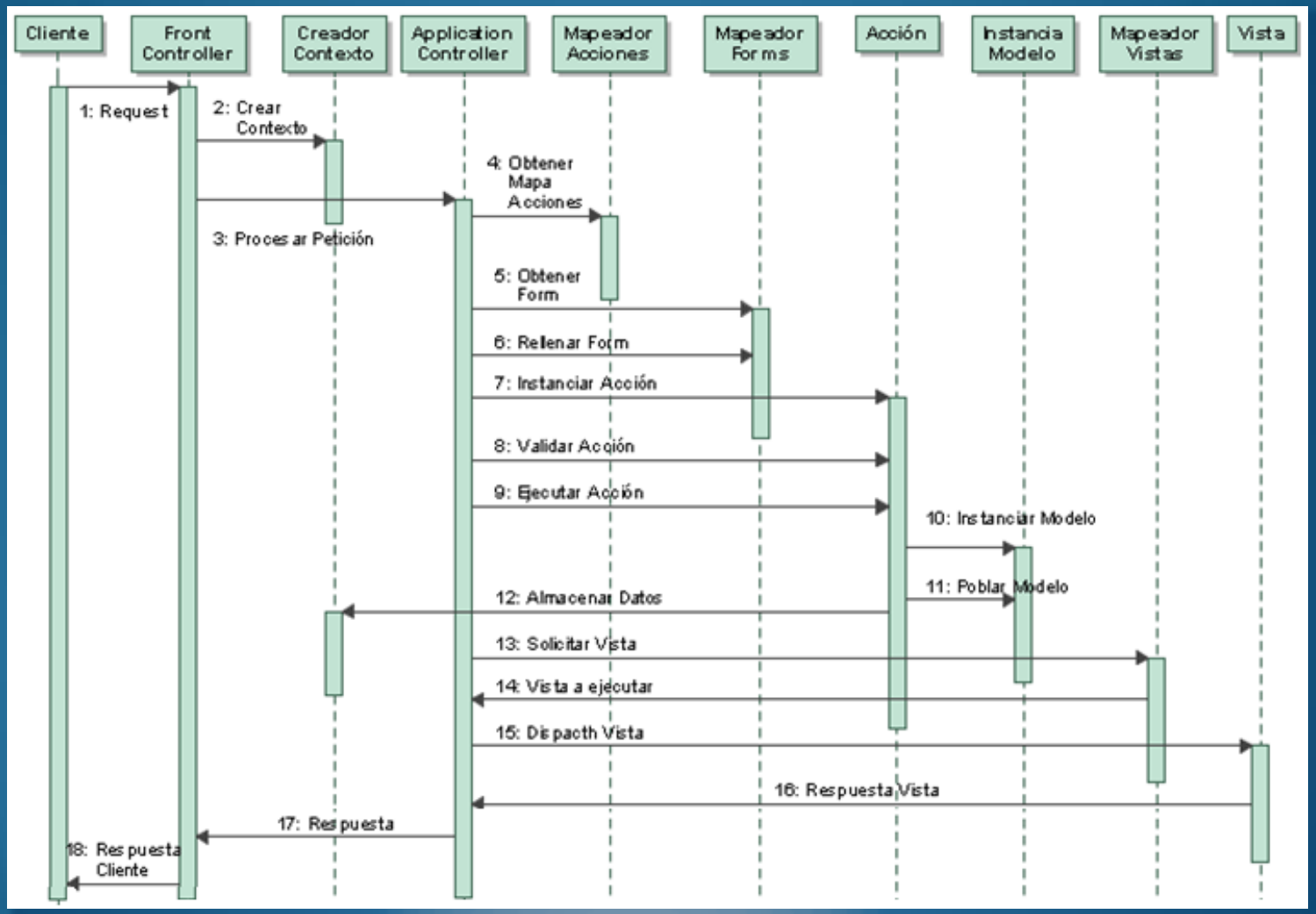
```
# ------------------------------------------------------------- #
# Acción/Página de 'Login de Usuario':
loginusuario        = edu.uoc.pfc.drsimo.ccuoc.acciones.LoginAction
loginusuario.form   = edu.uoc.pfc.drsimo.ccuoc.forms.LoginForm
loginusuario.next   = view-homeUsuario
loginusuario.error  = view-login
loginusuario.scope  = session
# ------------------------------------------------------------- #
```

- *Patrón URL de Petición:* ***http://<server>:<puerto>/<context>/<accion>.htm***
  - ***<Accion>*** → Clase Action a mapear (*Abstract Action*) → "*LoginAction*"
  - ***<Form>*** → Clase Form a generar (*Form Bean*) → "*LoginForm*"
  - **<Next>** → Siguiente Acción/Vista a ejecutar → Vista: "*homeUsuario*"
  - **<Error>** → En caso de error, Acción/Vista a ejecutar → Vista: "*homeUsuario*"
  - **<Scope>** → Ámbito a asociar a la Acción (*Request / Session*) → "*Session*"

  - *Ejemplo*: http://localhost:8080/clubciclistauoc/***loginusuario.htm***

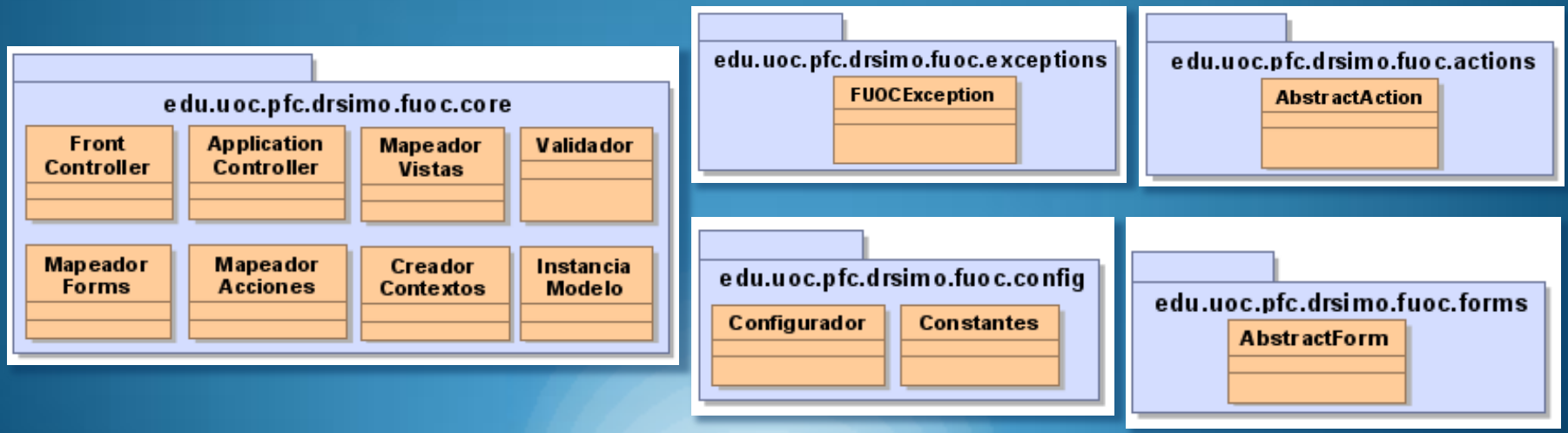
FUOC: Flujo de una Petición

- *Front Controller* recibe la petición (con un determinado contexto) y delega su ejecución al *Application Controller* (éste orquestará toda su posterior gestión):



FUOC: Estructura de Clases/Paquetes

- *edu.uoc.pfc.drsimo.fuoc.core:*
  Clases principales "core" que soportarán las principales funcionalidades.
- *edu.uoc.pfc.drsimo.fuoc.actions:*
  AbstractAction, clase abstracta a partir de la cual heredarán todas las Actions.
- *edu.uoc.pfc.drsimo.fuoc.config:* Configuración inicial del Framework, etc.
- *edu.uoc.pfc.drsimo.fuoc.exceptions:* FUOCExcepcion (gestión excepciones).
- *edu.uoc.pfc.drsimo.fuoc.forms:* AbstractForm, clase abstracta a partir de la cual heredarán todos los Forms.

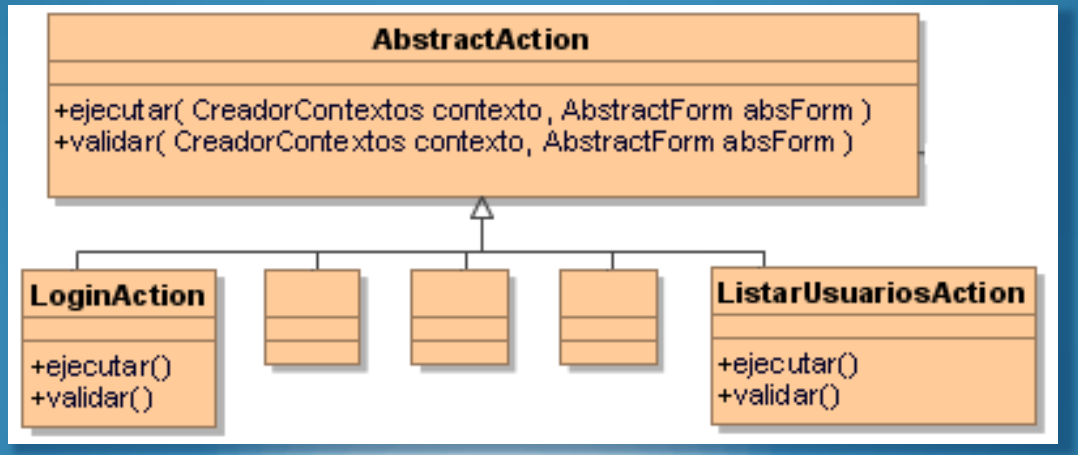


Aplicación Web: "Club Ciclista UOC"

- Análisis:
  - Clase Action
  - Clase Form
- Arquitectura
- Diagrama de Estados/Navegación
- Interfaz de Usuario:
  - Validación de Campos
  - Usabilidad/Accesibilidad/Dinamicidad
  - Dependencias Externas
- Productos Obtenidos
- Mejoras/Evoluciones

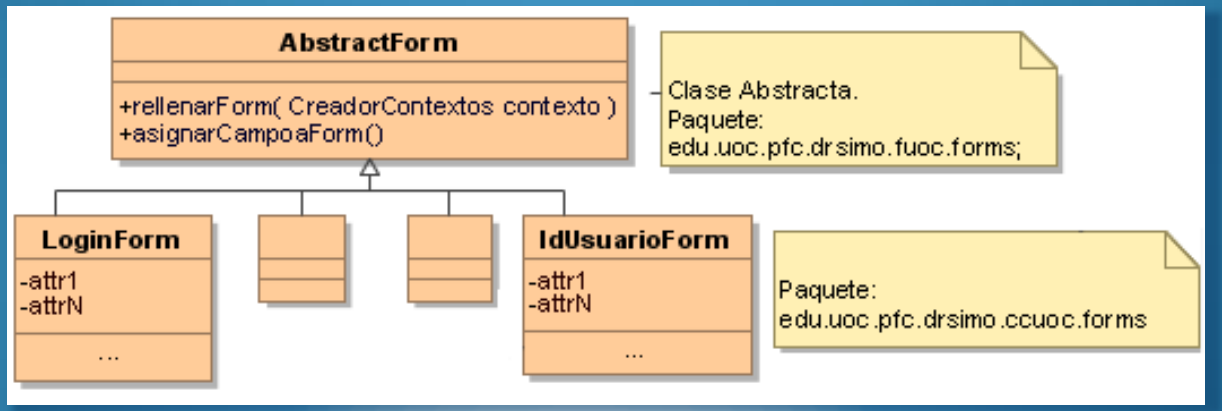
Clase Action

- ***Paquete***: ***edu.uoc.pfc.drsimo.ccuoc.acciones***
- Todas las peticiones se mapean con Clases *Action* (*Patrón Command*).
- Heredan de *AbstractAction* redefiniendo los métodos "*ejecutar*" y "*validar*" e implementándolos según la necesidad.
- Todos los Action reciben:
  - ***Contexto*** (*Creador Contextos*), que será el contexto del Servlet con toda la información de la Request.
  - ***Form*** (*Abstract Form*) que identificará el formulario generado para la ejecución de la acción y contendrá los datos necesarios para la misma.
- *Navigation Action* (no realiza ninguna operación, empleada para navegación).



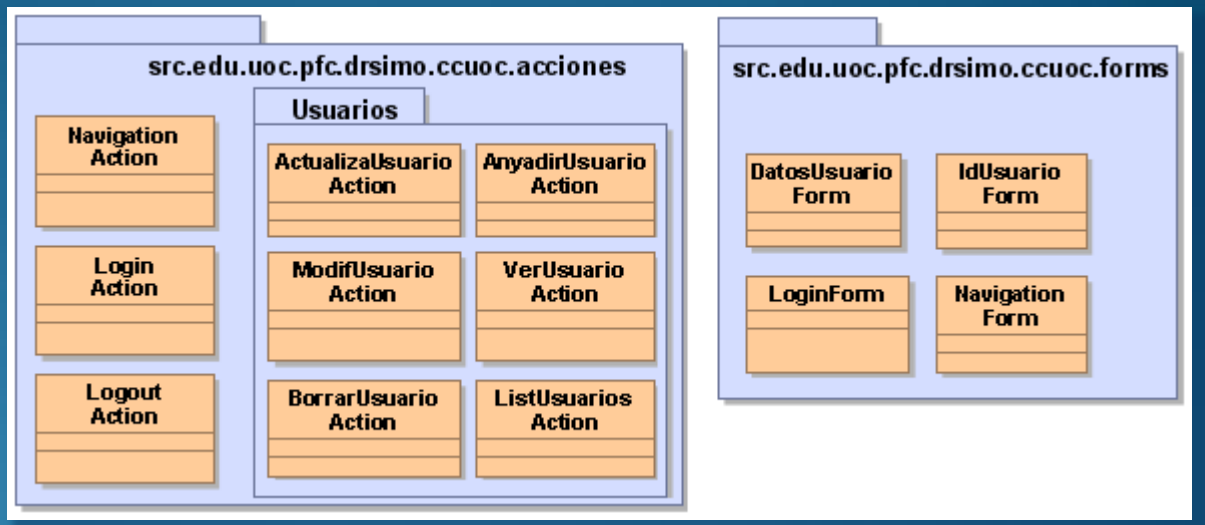
Clase Form

- ***Paquete***: ***edu.uoc.pfc.drsimo.ccuoc.forms***
- Encargados de albergar los diferentes parámetros que pueden llegar desde los formularios de entrada a la aplicación.
- Heredarán de *AbstractForm*, definiendo cada uno sus respectivas características y atributos.
- Tales clases se instanciarán con el mismo número y tipo de parámetros que se esperan de entrada y además irán asociados a una acción concreta, de tal forma que han de existir todos los parámetros necesarios.
- Al llegar una petición, *Application Controller* instancia y rellena las clases *Form* y la *Action* adecuadas de forma dinámica, mapeándose los campos del formulario y los atributos de la clase de forma automática.



Diseño de la Aplicación

- Capa de Presentación:
  - *Paquete Actions.*
  - *Paquete Forms.*



- Capa de Negocio y Datos:

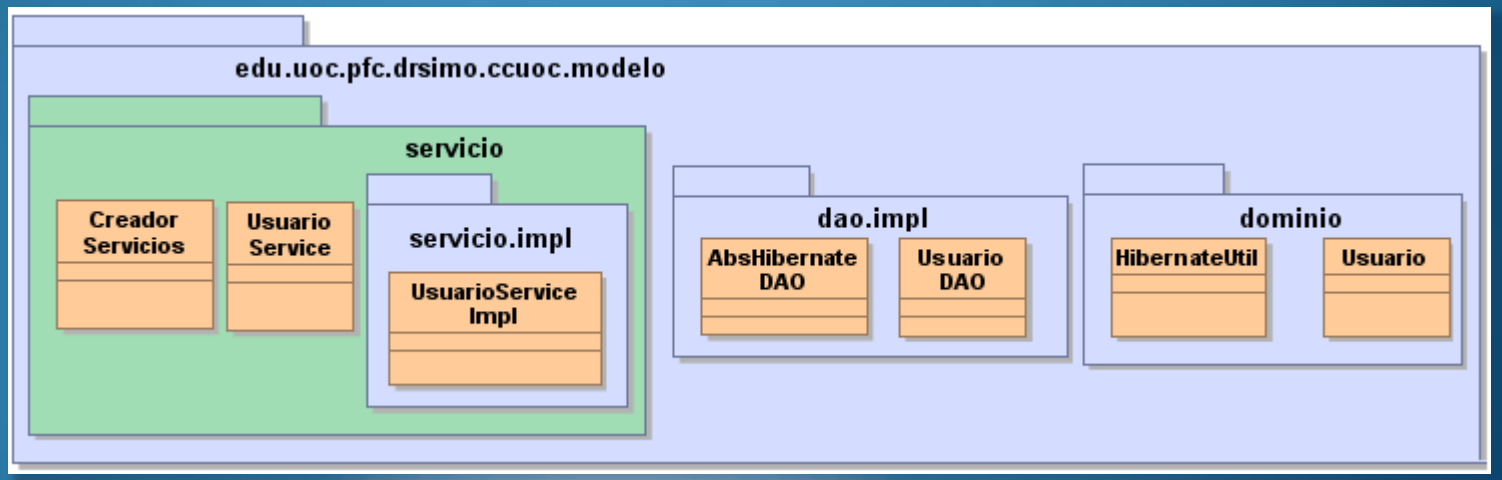
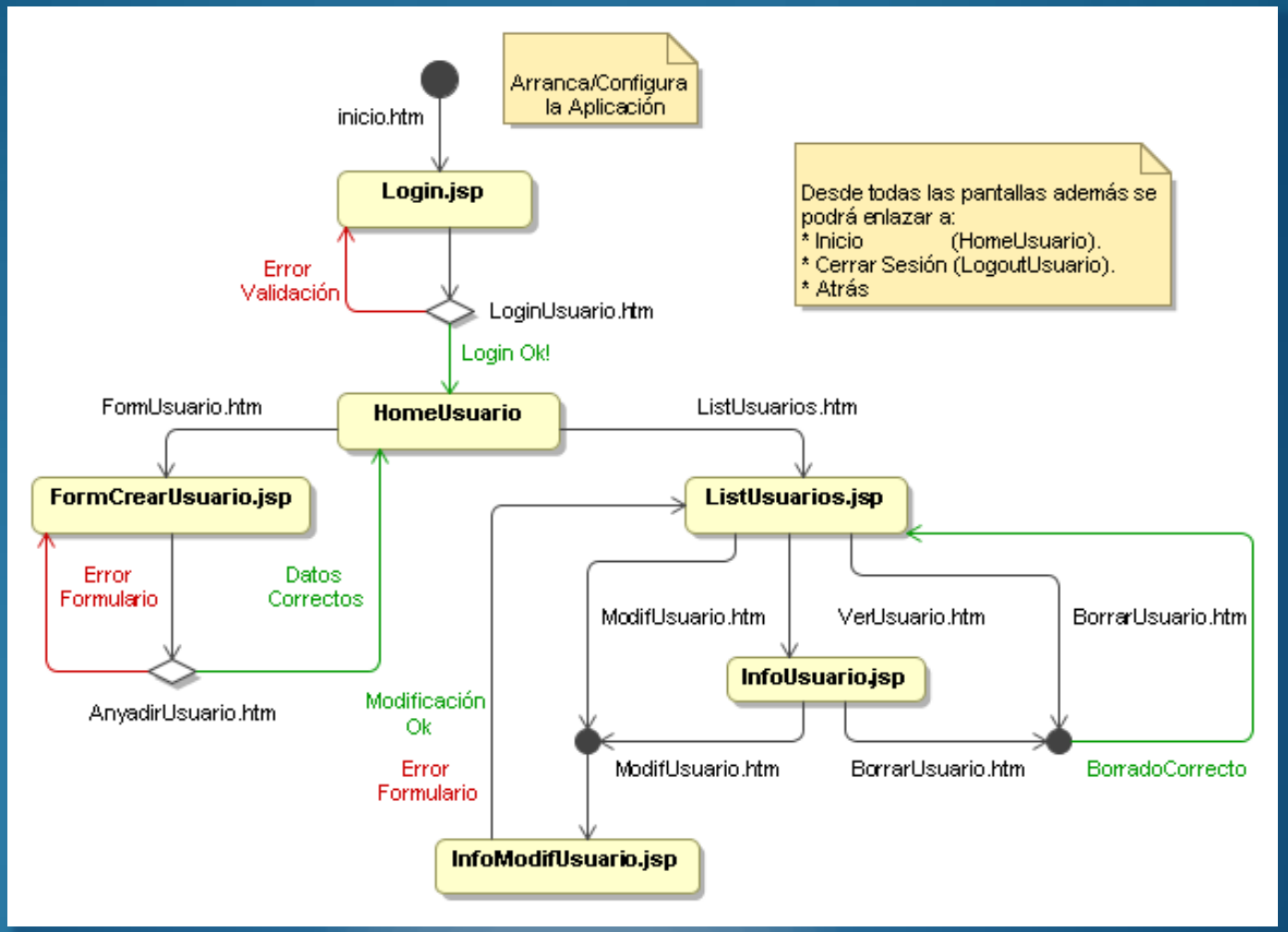


Diagrama de Estados / Navegación

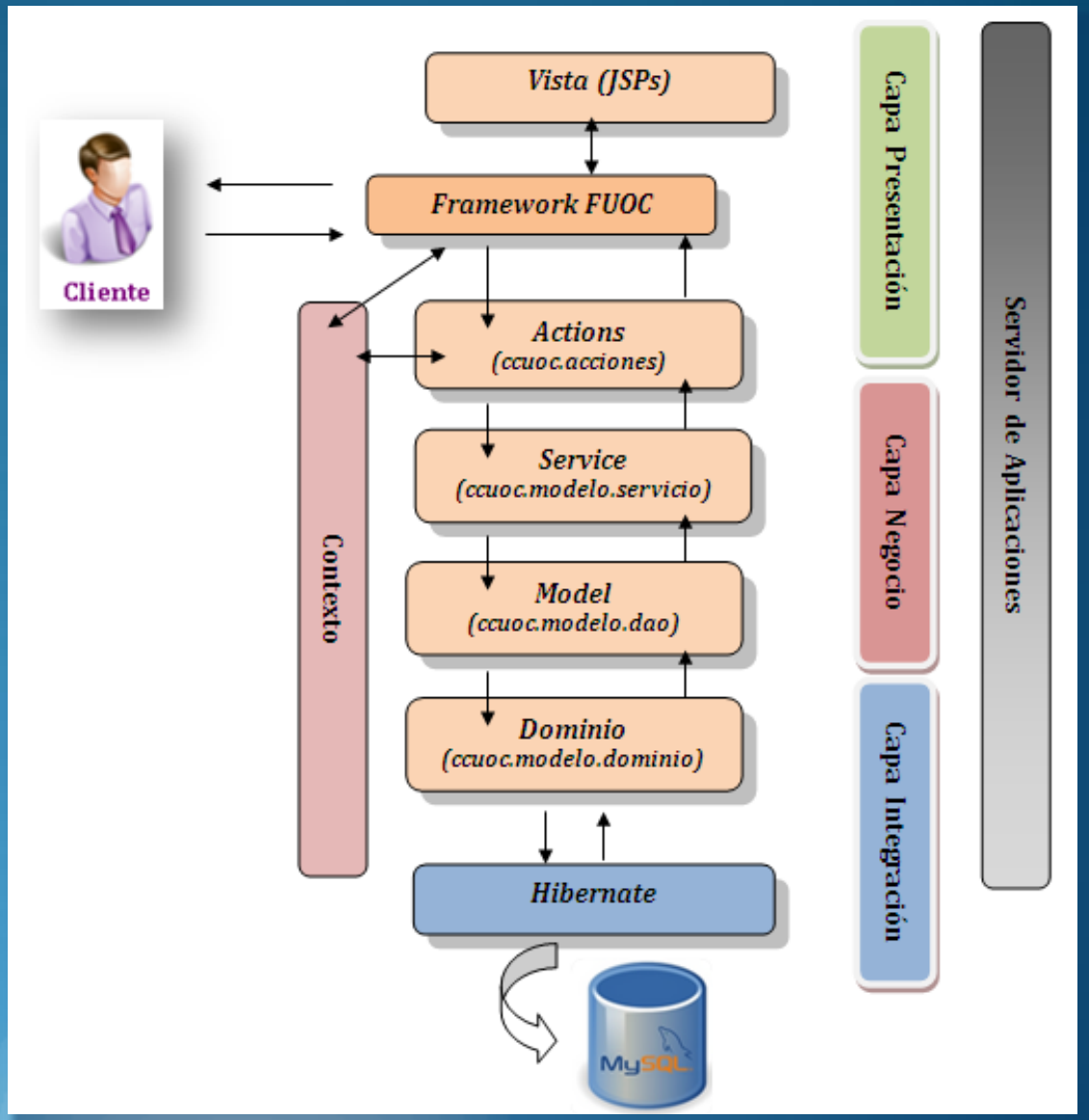
- Flujo de las principales Acciones/Estados por los que transita la Aplicación:



Arquitectura

Tecnologías Empleadas:

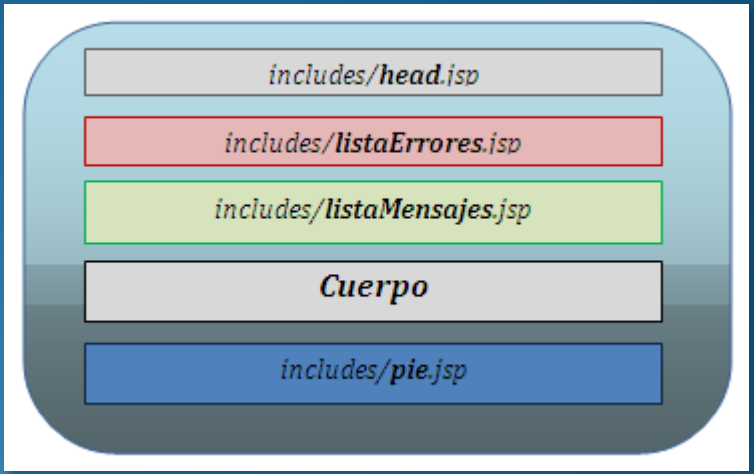
- Java JDK 6 *(Update 37)*
- Servidor Aplicaciones:
  - JBoss *6.1.0*
- Hibernate *3.6.10*
- Base de Datos:
  - MySQL *5.5.28*
  - Conector MySQL *5.1.22*
- Ant *1.8.4*



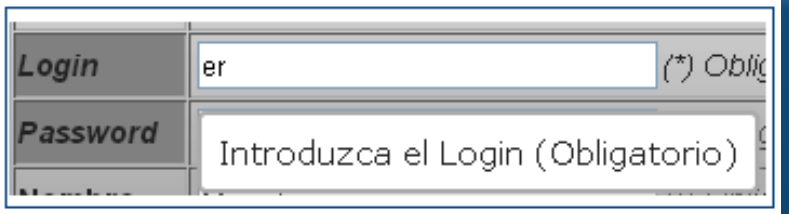
Interfaz de Usuario

- Vistas Modularizadas:

  (software mejor estructurado, fomentando la reutilización y mantenibilidad)

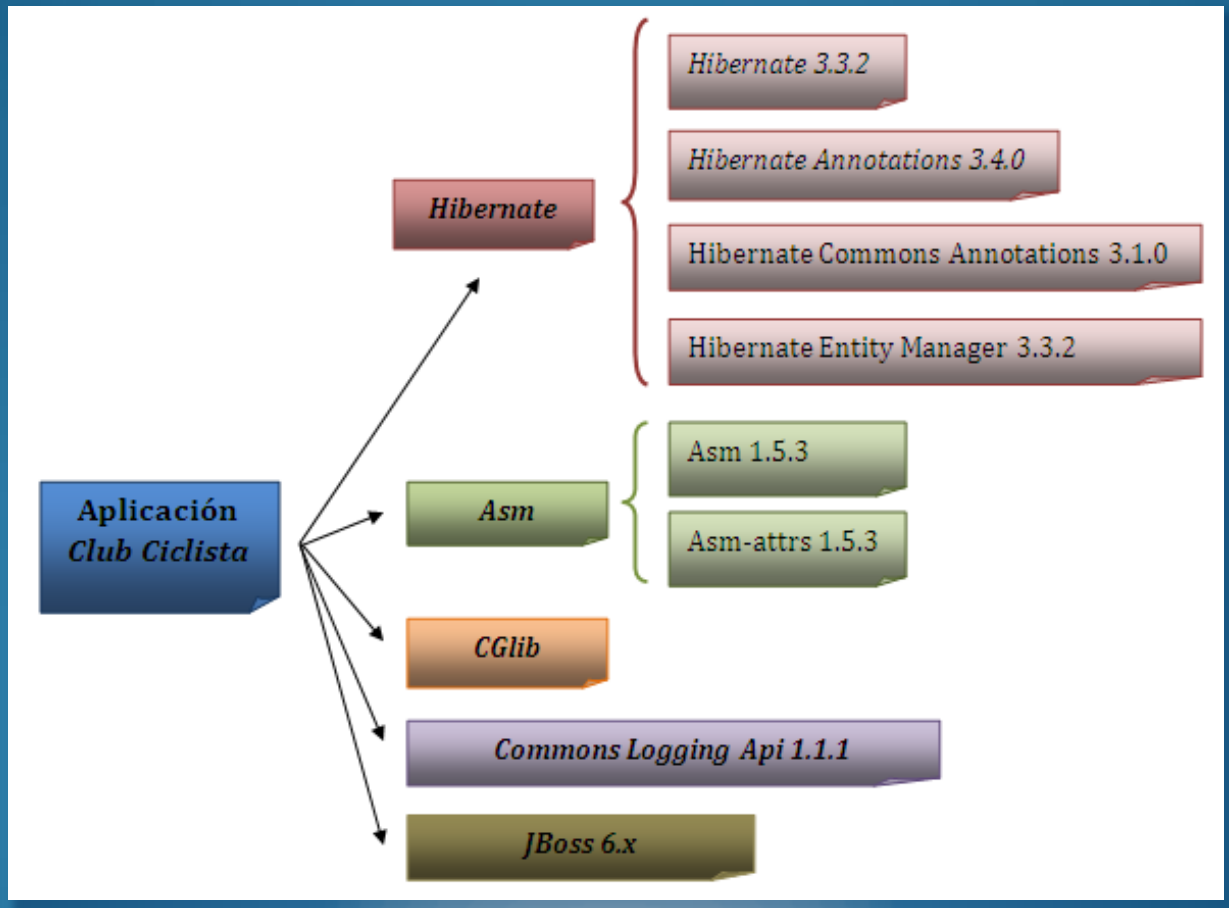


- Empleo de Jquery, proporcionando:
  - Mayor "Dinamicidad" a la Aplicación
  - Validación de campos en Cliente
  - Usabilidad



Librerías / Dependencias Externas

- La Aplicación Web "***Club Ciclista***" requerirá de la funcionalidad de las siguientes librerías externas (*además del propio "**FUOC-Framework**"*):



Resultados/Productos Obtenidos

- ***Framework de Presentación***: "***FUOC-Framework-0.1***":
  - Código Fuente.
  - Librería generada *(/dist/lib/FUOC-Framework-0.1.jar).*
  - Documentación Javadoc.
  - Proyecto Eclipse.
  - Herramienta para generación de entregables *(Ant).*
- ***Aplicación Web***: "***Club Ciclista UOC***":
  - Código Fuente.
  - Entregable generado *(/dist/clubciclistauoc.war).*
  - Documentación Javadoc.
  - Proyecto Eclipse.
  - Herramienta para generación del desplegable *(Ant).*
  - Archivos configuración para la BBDD y JBoss *(/Configuraciones).*
- ***Documentación:***
  - Memoria.
  - Presentación.
  - Anexos (*guía instalación, configuración, ejecución, despliegue, metodología pruebas, etc*.)

Mejoras / Evoluciones

Tanto el ***Framework*** como la ***Aplicación Web*** implementan gran abanico de funcionalidades, pero también deja la puerta abierta a otras muchas, como son:

- Implementación de roles/control de acceso.
- Servicio de Logs/Trazas (*niveles, tipos de salidas, etc.*)
- Creación de Forms de forma dinámica.
- Jquery/Ajax (*más funcionalidades, mayor gestión desde el lado del Cliente*).
- Web Responsive (*con vistas al despliegue en dispositivos móviles*).
- Sistema RESTful (*como medida de robustez y escalabilidad, interfaz ligera*).

- Otros:
  - Implementación de más mecanismos de validación en la parte cliente.
  - Añadir más idiomas al servicio de internacionalización.
  - Implementar cacheo para la aplicación Web (*suponiendo un gran volumen de visitas*).

Gracias por la Atención

Estudio e Implementación de un Framework de Presentación

Universitat Oberta de Catalunya.

Curso 2012/13