DP1 2020-2021

Documento de Diseño del Sistema - Entrega Final

Futvilla

https://github.com/gii-is-DP1/dp1-2020-g3-08.git

Miembros:

- Gonzalo Fernández Jiménez (Project Manager)
- Javier Grosso Gómez de Terreros
- Juan Luis Muñoz Navarro
- Marcos Rodríguez Osorio
- Ignacio Sanabria Alonso de Caso
- Bogdan Marian Stefan

Tutor: Irene Bedilia Estrada Torres

GRUPO G3-08

09/02/2020

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
08/01/2020	V1.0	• Creación del documento	3
09/01/2020	V1.1	• Diagrama de dominio/diseño	3
10/01/2020	V1.2	• Patrones de diseño y arquitectónicos aplicados • Decisiones de diseño	3
09/02/2020	V2	• Añadido el diagrama capas corregido • Añadido el diagrama de datos corregido • Añadido decisión de diseño • Modificación patrones de diseño	4

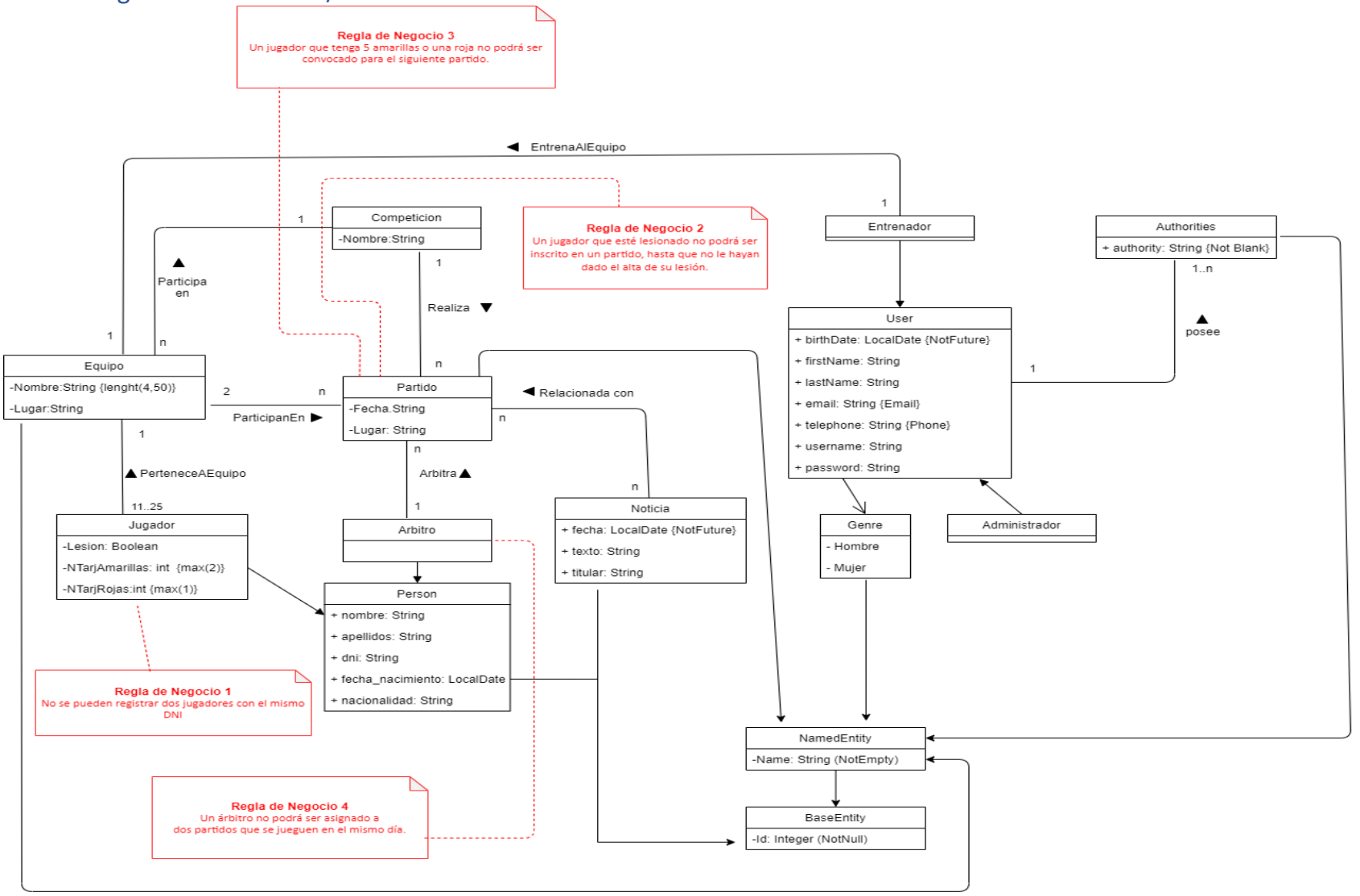
Contenido

1. Introducción

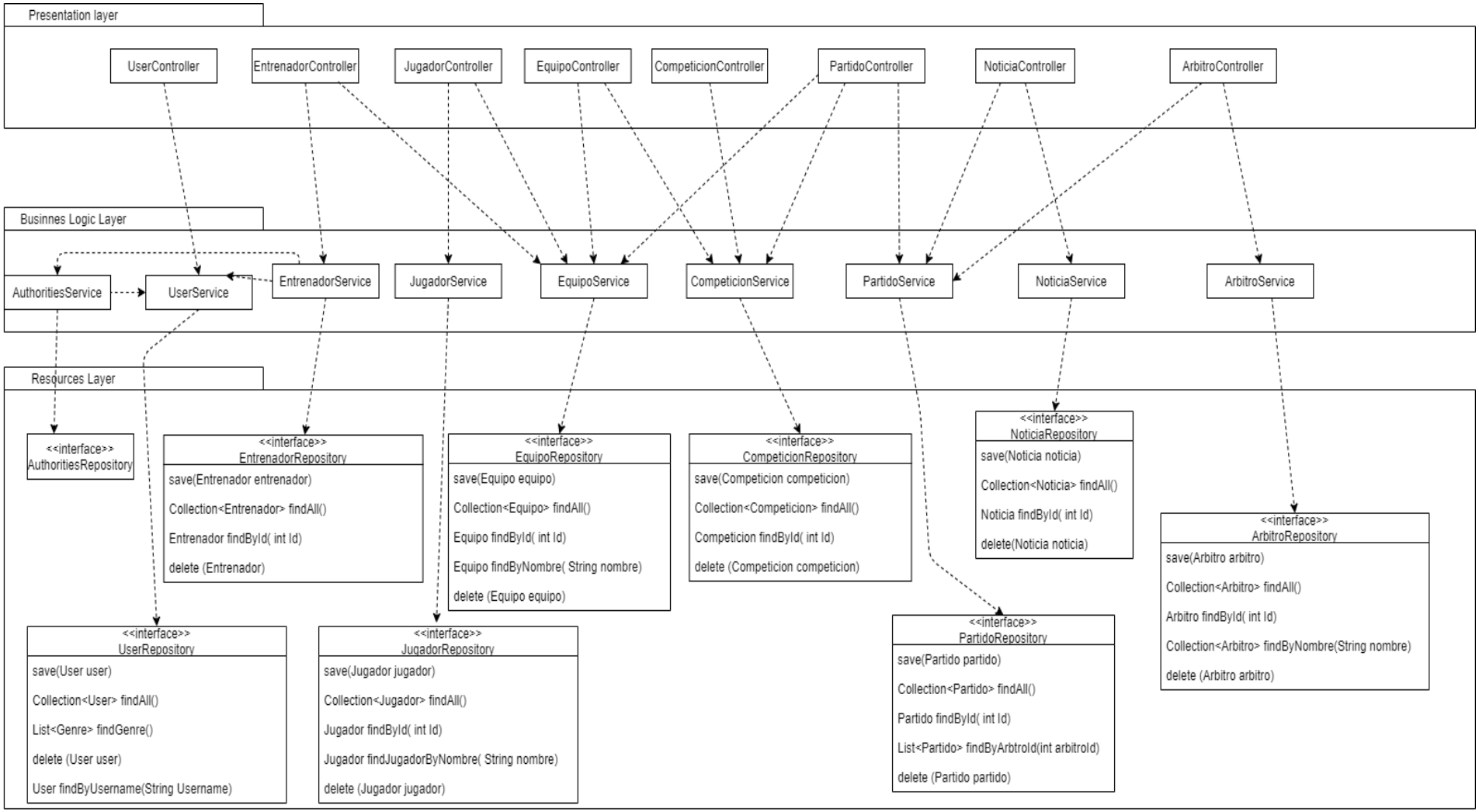
El proyecto se basa en administrar competiciones de fútbol y publicar noticias relacionadas. Cada competición tiene una lista de equipos formado por un número de jugadores que disputan partidos. Un partido está configurado por un árbitro y los dos equipos que se enfrentan. Los usuarios del sistema son administrador, entrenador y usuario registrado. Los objetivos del proyecto son una mejor organización en las competiciones y proporcionar una interfaz de usuario que permita a los usuarios navegar y acceder a datos (jugadores de un equipo, árbitros, ligas…) de forma fácil y eficaz.

2. Diagrama(s) UML:

2.1. Diagrama de Dominio/Diseño



2.2. Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios)



3. Patrones de diseño y arquitectónicos aplicados

En esta sección se especifica el conjunto de patrones de diseño y arquitectónicos aplicados durante el proyecto. Para especificar la aplicación de cada patrón puede usar la siguiente plantilla:

3.1. Patrón: Estilo arquitectónico por capas.

Tipo: Arquitectónico

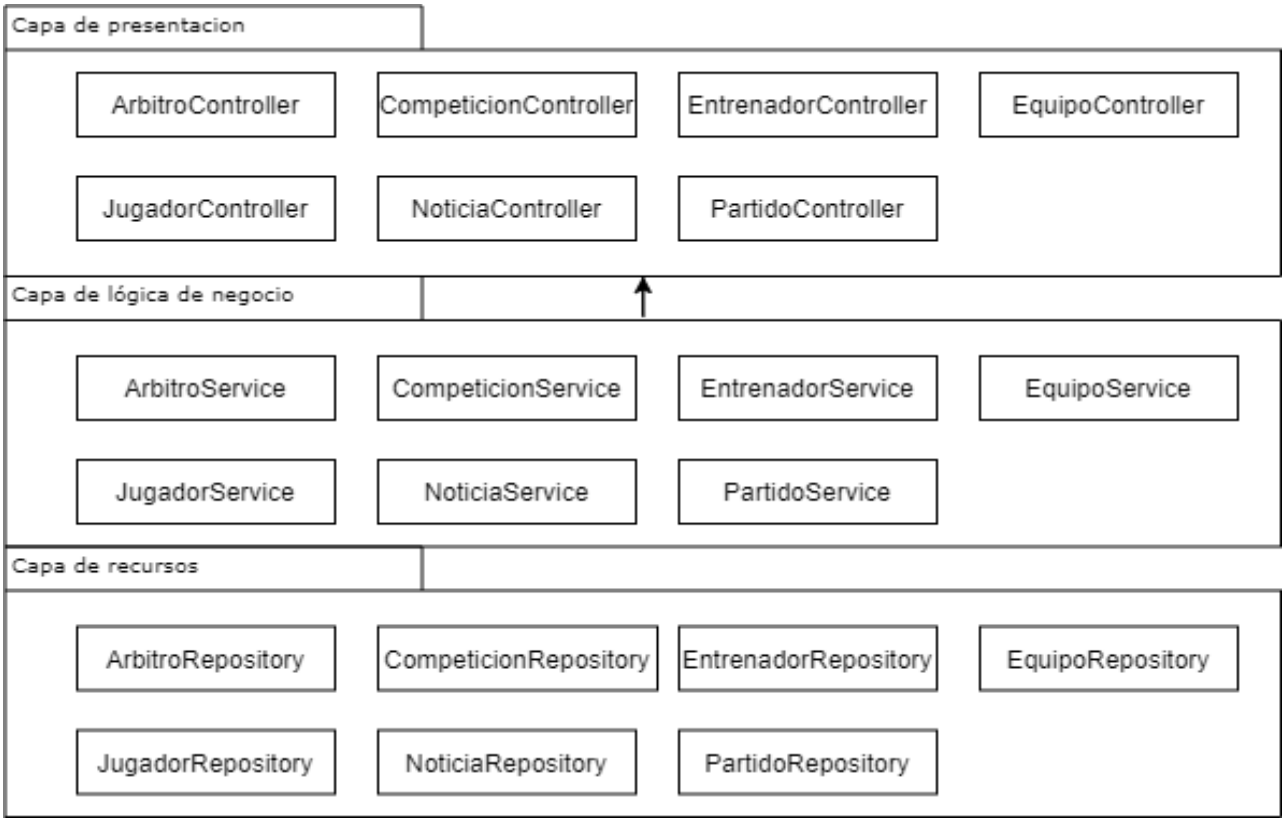
La funcionalidad del sistema se organiza en capas. Cada grupo de componentes de una capa ofrece una funcionalidad ofrecida por la capa inmediatamente inferior a ella.

Contexto de Aplicación

El estilo arquitectónico por capas realiza la separación de responsabilidades separándolas en 3 capas, la capa de presentación, la capa de la lógica de negocio y la capa de recursos.

En nuestra aplicación, en la capa de presentación se encuentran todos los controladores, en la capa de lógica de negocio, se encuentran los servicios y en la capa de recursos los repositorios.

Clases o paquetes creados



Ventajas alcanzadas al aplicar el patrón

Este estilo es aconsejable para el desarrollo de una aplicación web ya que favorece;

- Alta cohesión
- Bajo acoplamiento
- Separación de responsabilidades.

3.2. Patrón: MVC (Modelo, Vista y Controlador)

Tipo: Diseño

El patrón Modelo, Vista y Controlador es un patrón arquitectónico que separa la lógica de negocio de su representación, para ello tiene 3 componentes, el Modelo, que son los datos y el tratamiento de ellos, la Vista, que es la representación del modelo con la que el usuario puede interactuar y el Controlador, que básicamente redirecciona los datos desde la vista al modelo.

Contexto de Aplicación

Hemos aplicado el patrón Modelo, Vista y Controlador en nuestra aplicación relacionada con la creación de competiciones de fútbol.

Para el **Modelo** hemos utilizado los siguientes paquetes:

- org.springframework.samples.model

- org.springframework.samples.repository

- org.springframework.samples.service

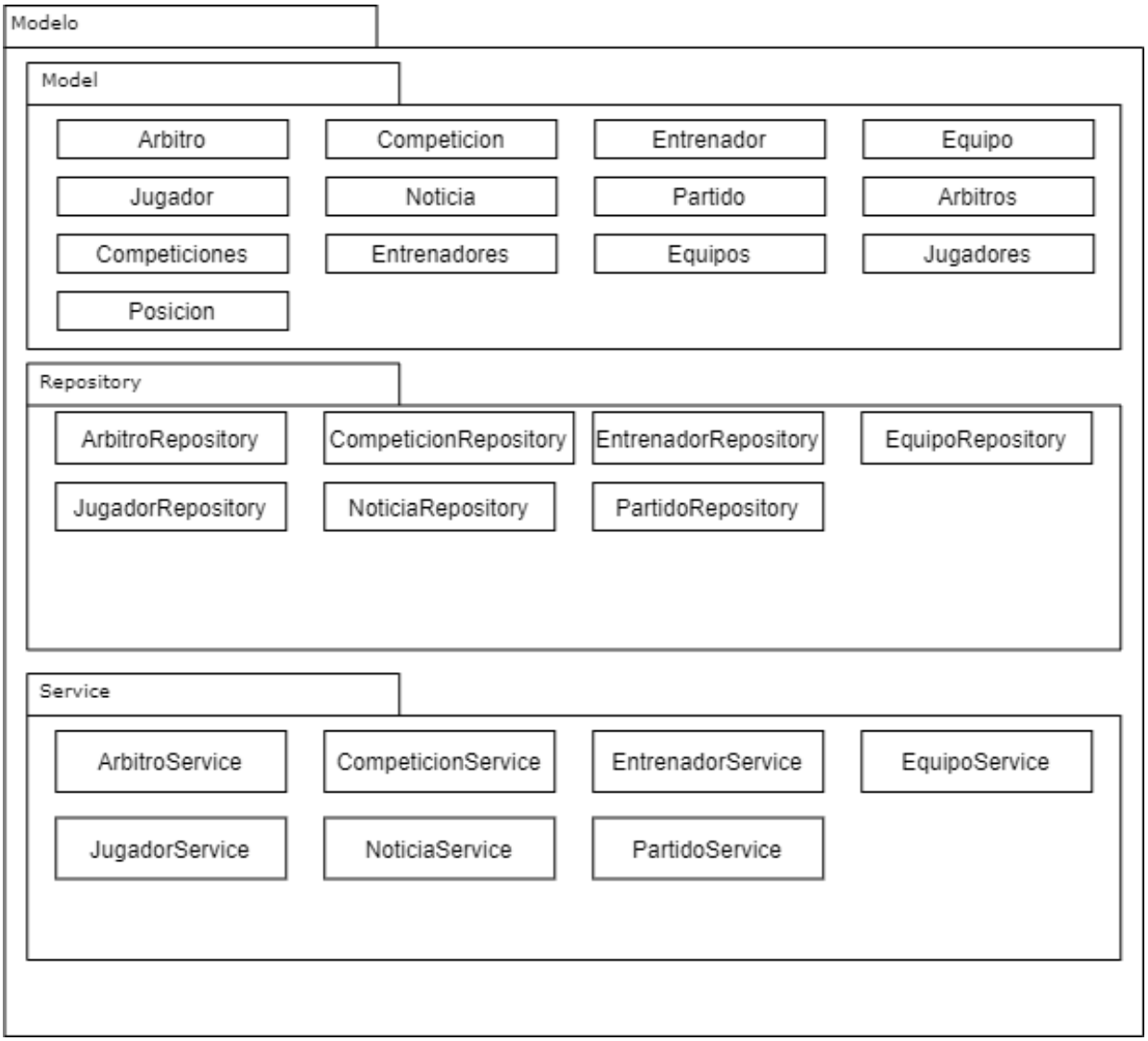
Para la **Vista**, debemos seguir la siguiente ruta:

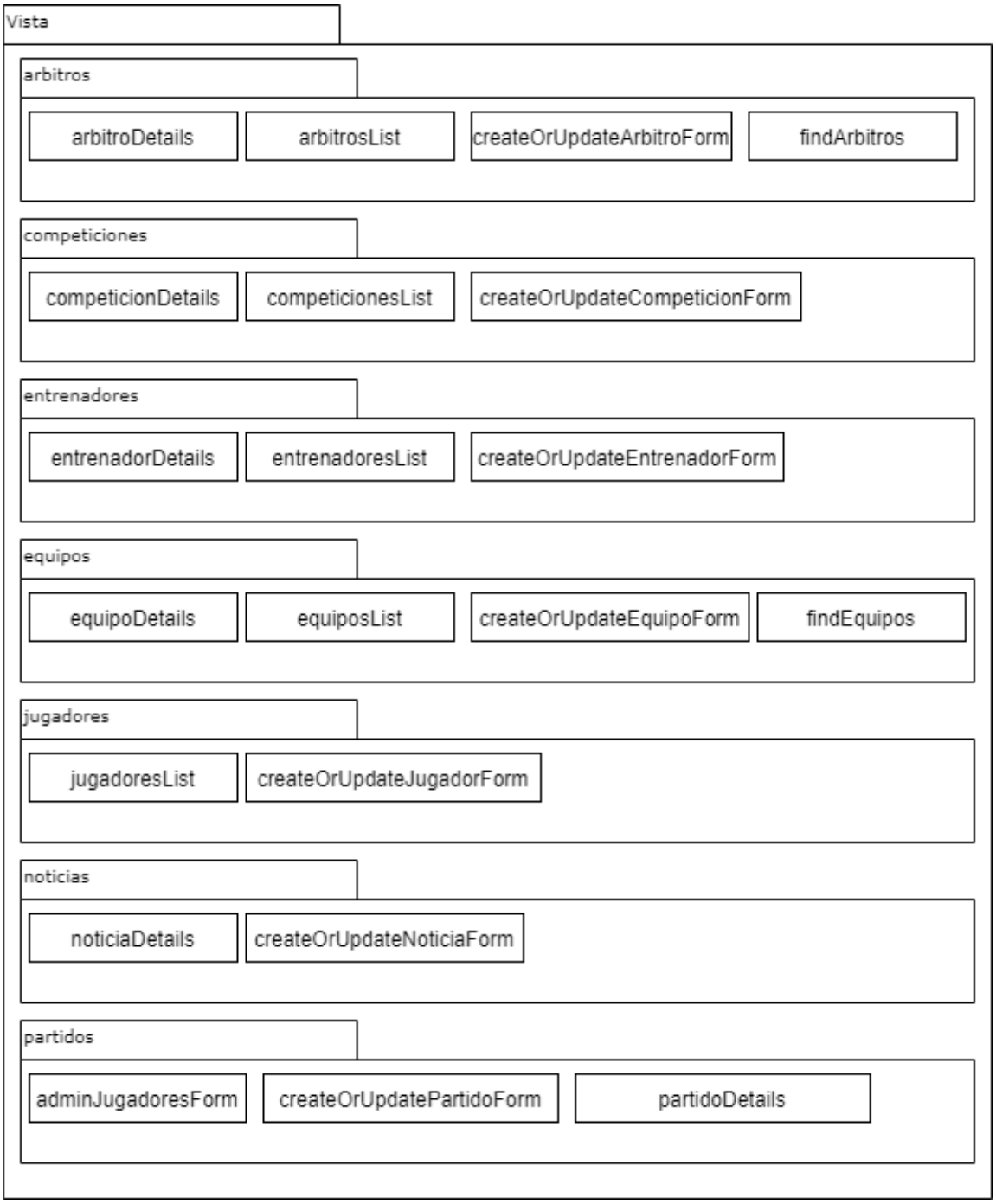
src->main->webapp->WEB-INF, y aquí podemos ver todas las carpetas con sus correspondientes archivos .jsp o .tag que hacen referencia a las vistas de nuestra aplicación.

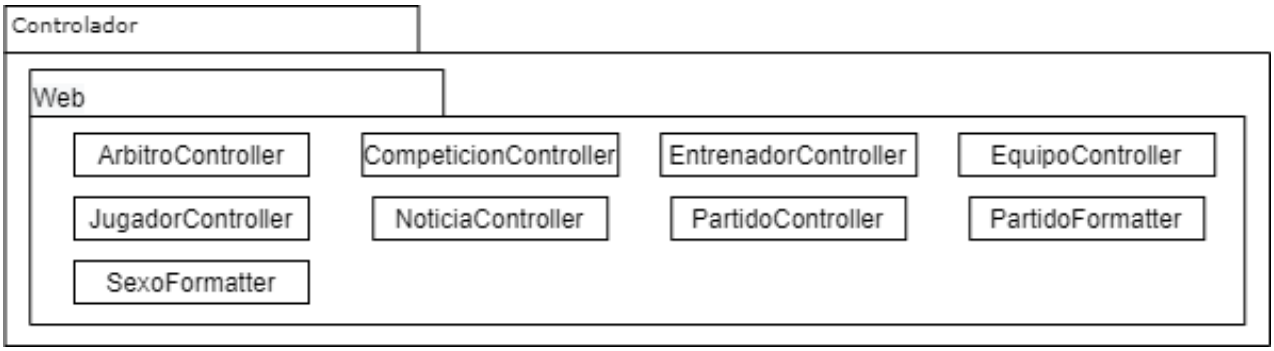
Por último, tenemos el **Controlador**, los podemos ver en el siguiente paquete:

- org.springframework.samples.web.

Clases y paquetes creados




Vista
arbitros
arbitroDetails
arbitrosList
createOrUpdateArbitroForm
findArbitros
competiciones
competicionDetails
competicionesList
createOrUpdateCompeticionForm
entrenadores
entrenadorDetails
entrenadoresList
createOrUpdateEntrenadorForm
equipos
equipoDetails
equiposList
createOrUpdateEquipoForm
findEquipos
jugadores
jugadoresList
createOrUpdateJugadorForm
noticias
noticiaDetails
createOrUpdateNoticiaForm
partidos
adminJugadoresForm
createOrUpdatePartidoForm
partidoDetails



Ventajas alcanzadas al aplicar el patrón

Las dos grandes ventajas de aplicar este patrón son las siguientes:

- Soporta múltiples vistas para los datos.
- Favorece alta cohesión, bajo acoplamiento y separa responsabilidades.

Tiene una desventaja:

- Requiere el uso de un Framework.

4. Decisiones de diseño

En esta sección describiremos las decisiones de diseño que se han tomado a lo largo del desarrollo de la aplicación que van más allá de la mera aplicación de patrones de diseño o arquitectónicos.

Decisión 1

Descripción del problema: La clase Entrenador no funcionaba como Usuario

Decidimos meter la clase Entrenador como Usuario, pero nos surgió problemas en cuanto a la compatibilidad de Petclinic, ya que no podía haber varias clases de usuario.

Alternativas de solución evaluadas:

a) Crear la clase Entrenador como una entidad aparte, y relacionarla con la clase User: esta alternativa era viable, pero se necesitaba aumentar el número de relaciones en el proyecto.

Justificación de la solución adoptada

Como consideramos que la entidad Entrenador era necesaria en nuestra visión del proyecto, decidimos implementar la alternativa A de crear la clase Entrenador como una entidad aparte, creando aparte una clase más y relacionándola con Usuario.

Decisión 2:

Descripción del problema: Dificultades en la historia de usuario H5: relacionar noticias con otras entidades

La historia de usuario H5 en un principio conllevaba que la noticia se pudiera relacionar con multitud de entidades como partido, equipo, competición, jugador, etc.

Alternativas de solución evaluadas:

a) Prescindir de las relaciones de Noticias con otras entidades.

b) Escoger el evento principal (Partidos) para las relaciones con noticias.

c) Hacer múltiples relaciones N:N no necesarias en la visión del proyecto y que únicamente aumentan la complejidad.

Justificación de la solución adoptada

Decidimos implementar la alternativa B, dado que podía observar un claro interés relacionar Noticias con alguna entidad de relevancia en la información de los Partidos, pero no era una función tan necesaria como para aumentar la complejidad sin motivo alguno metiendo relaciones de más.

Así que, se decidió que sólo habría relación con Partidos (además de que, Partidos tiene relación con Equipo y Árbitro, así que igualmente la información aparecerá al seleccionar el partido).

Decisión 3:

Descripción del problema: Selección sobre donde poner logs.

La decisión surge sobre la necesidad de utilizar los logs y decidir en qué lugar tendrían que aparecer dichos logs,

Justificación de la solución adoptada

Para ello hemos introducido logs al principio de todos los servicios y controladores, en caso de que los controladores pudieran llevar a error se han añadido dos tipos de logs uno si ha entrado correctamente y otro cuando da error.

Todos estos logs que hemos implementado en nuestra aplicación han sido de tipo “info”, y los que pasan por los errores son de tipo “warning”.