

DP1 2020-2021

Documento de Diseño del Sistema

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

Versión 1

17/11/2020

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
08/01/2020	V1.0	<ul style="list-style-type: none">● Creación del documento	3
09/01/2020	V1.1	<ul style="list-style-type: none">● Diagrama de dominio/diseño	3
10/01/2020	V1.2	<ul style="list-style-type: none">● Patrones de diseño y arquitectónicos aplicados● Decisiones de diseño	3

Contenido

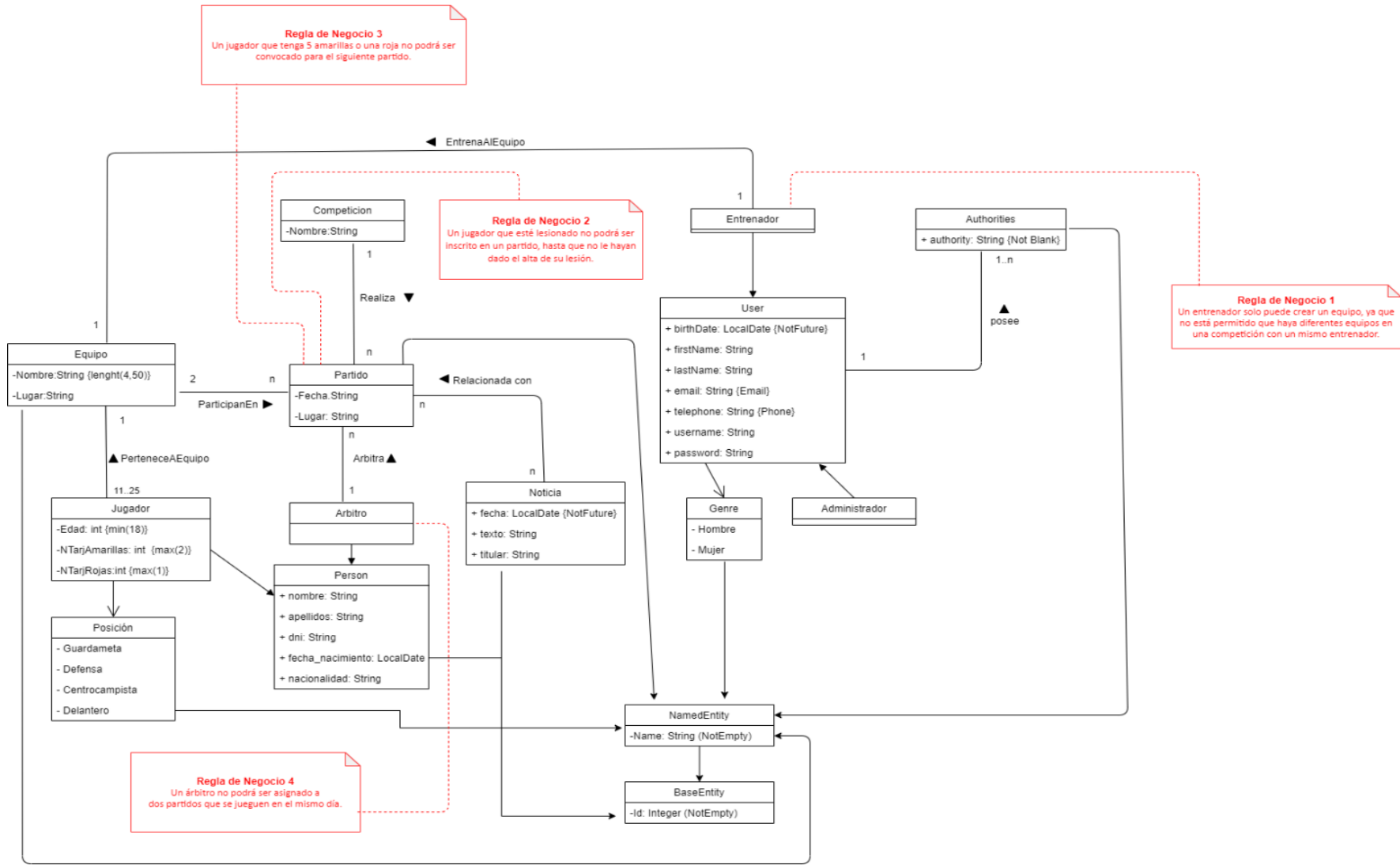
1. Introducción	4
2. Diagrama(s) UML:	1
2.1. Diagrama de Dominio/Diseño	1
2.2. Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios)	2
3. Patrones de diseño y arquitectónicos aplicados	1
3.1. Patrón: MVC (Modelo, Vista y Controlador)	1
4. Decisiones de diseño	6

1. Introducción

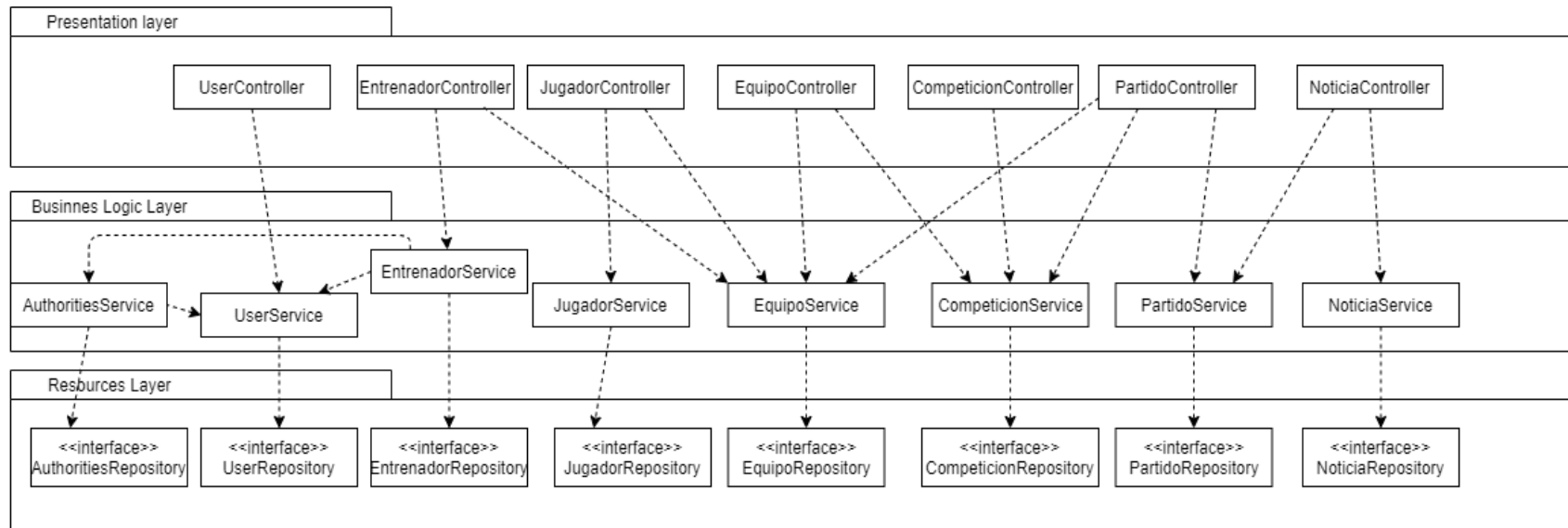
El proyecto se basa en administrar competiciones de fútbol y publicar noticias relacionadas. Cada competición tiene una lista de equipos formado por un número de jugadores que disputan partidos. Un partido está configurado por un árbitro y los dos equipos que se enfrentan. Los usuarios del sistema son 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: MVC (Modelo, Vista y Controlador)

Tipo: Arquitectónico | de 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

MODELO		
PAQUETES		
Model	Repository	Service
Arbitro	ArbitroRepository	ArbitroService
Arbitros	CompeticionRepository	CompeticionService
Competicion	EntrenadorRepository	EntrenadorService
Competiciones	EquipoRepository	EquipoService
Entrenador	JugadorRepository	JugadorService
Entrenadores	NoticiaRepository	NoticiaService
Equipo	PartidoRepository	PartidoService
Equipos		
Jugador		

Jugadores		
Noticia		
Partido		
Posicion		

VISTA	
PAQUETES	
Entidad	Archivos .jsp
arbitros	arbitroDetails arbitrosList createOrUpdateArbitroForm findArbitros
competiciones	competicionDetails competicionesList createOrUpdateCompeticionForm

entrenadores	entrenadorDetails entrenadoresList createOrUpdateEntrenadorForm
equipos	equipoDetails equiposList createOrUpdateEquipoForm findEquipos
jugadores	jugadoresList createOrUpdateJugadorForm
noticias	createOrUpdateNoticiasForm noticiasDetails
partidos	adminJugadoresForm createOrUpdatePartidoForm partidoDetails

CONTROLADOR
PAQUETE
Web
ArbitroController
CompeticionController
EntrenadorController
EquipoController
JugadorController
NoticiaController
PartidoController
PartidoFormatter
SexoFormatter

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) Prescindir de la clase Entrenador: esta alternativa no era viable, dado que era una clase necesaria en nuestros planes del proyecto.
- b) 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 B 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.

El problema surgió al intentar implementar tantas relaciones a la vez, para una función que realmente no era necesaria con tantas entidades.

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).