

Informe individual de actividades del proyecto

Datos generales

URL del Repositorio de GitHub: <https://github.com/gii-is-DP1/dp1-2020-g3-08.git>

Nombre de usuario en GitHub: javgrogom

Rama del usuario en Github: javgrogom

Participación en el proyecto

Historias de usuario en las que he participado

He desarrollado junto a mi compañero de equipo Gonzalo Fernandez las siguientes historias de usuario:

- H4: Publicar Noticias.
- H10: Visualización de Noticias
- H5: Asignar noticias a partidos
- H16: Editar perfil
- H9: Asignar y eliminar jugadores a un partido

Además de haber participado en las siguientes reglas de negocio junto a mi compañero Gonzalo Fernandez:

- RN3: número de tarjetas amarillas/rojas
- RN4: arbitrar dos partidos a la misma hora

Funcionalidad implementada

Junto a mi compañero Gonzalo Fernandez he creado la entidad Noticia y la entidad Equipo junto a sus repositorios.

También he implementado junto a Gonzalo Fernandez el controlador NoticiaController.

.

Implemente junto a Gonzalo Fernandez los métodos de los servicios NoticiaService, EquipoService, y los métodos necesarios para la tarea H9 en PartidoService.

También implemente junto a mi compañero Gonzalo Fernandez el validador PartidoValidator, necesario para la RN3 y la RN4

Y también he creado junto a mi compañero Gonzalo Fernandez las vistas de las entidades Noticia y Partido, situadas en la carpeta jsp.

Pruebas implementadas

Pruebas unitarias

He creado de forma individual el test unitario para 1 controlador (EquipoController).

En equipo he realizado junto a Gonzalo Fernandez 2 tests unitarios para Servicios (UsuarioService, NoticiaService)
Eso hace un total de 3 clases de test unitarios con un total de 19 métodos anotados con @Test.

Pruebas de Controlador

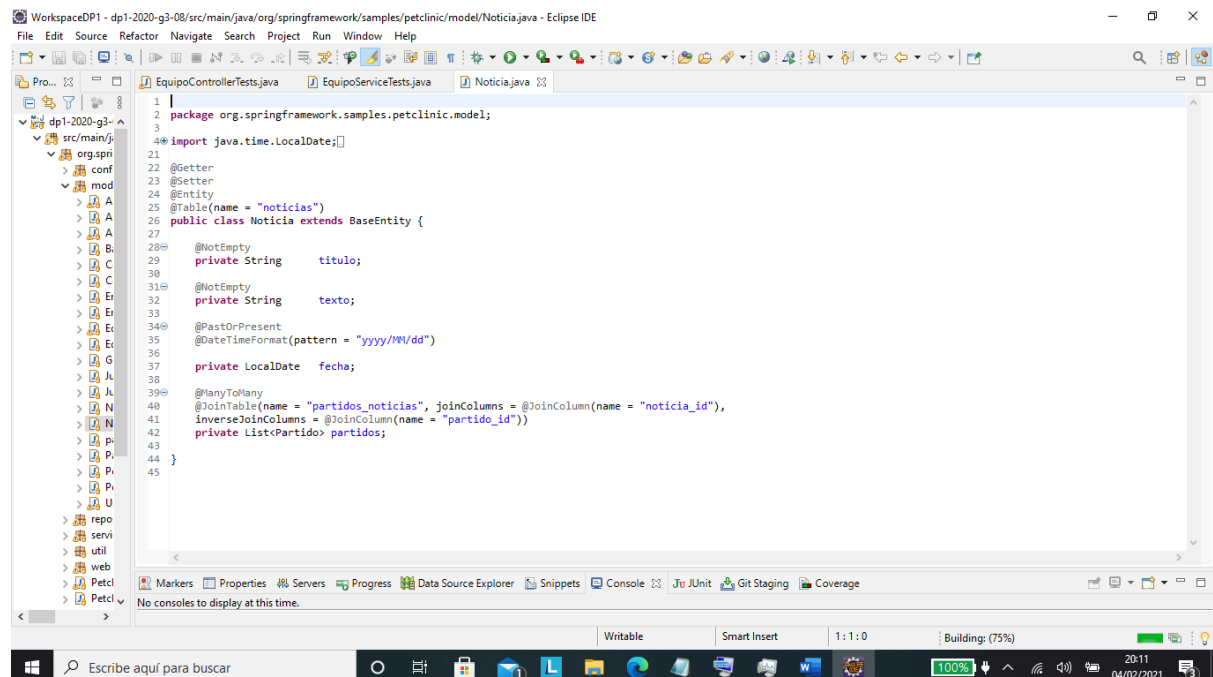
He creado 8 casos de prueba positivo y 2 negativos de controlador para TestEquipoController

Ejemplos de funcionalidades implementadas

Entidades

Entidad Noticia, esta en la ruta src/main/java/org.springframework.samples.petclinic/model
Realizada en colaboración con Gonzalo Fernandez

Consiste en una entidad que implementa en su código una relación many to many con la entidad partidos (debido a la historia de usuario H5 donde debemos de poder relacionar las noticias con los partidos), y posee un parámetro para el título de la noticia (no puede estar vacío), otro parámetro para el desarrollo de la noticia (tampoco puede estar vacío), y un parámetro para la fecha de publicación de la noticia (que solo puede ser fechas pasadas o el presente y esta formateada de la forma año/mes/día).

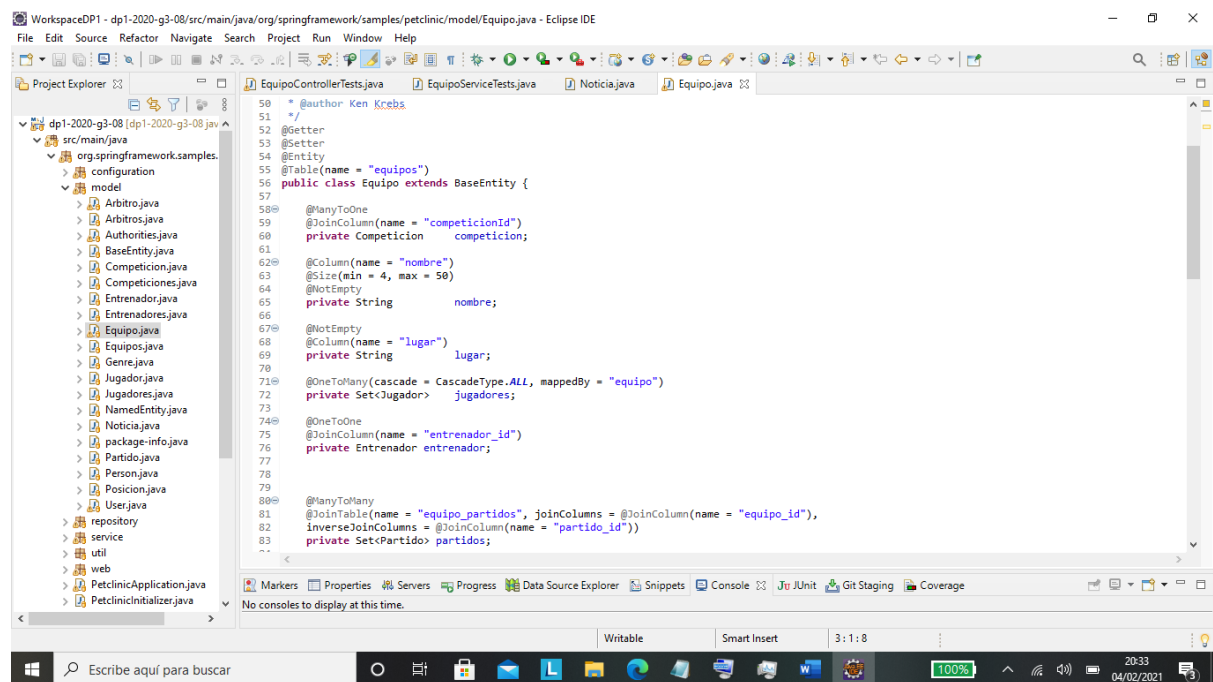


```
1 package org.springframework.samples.petclinic.model;
2
3 import java.time.LocalDate;
4
5 @Entity
6 @Table(name = "noticias")
7 public class Noticia extends BaseEntity {
8
9     @NotEmpty
10     private String titulo;
11
12     @NotEmpty
13     private String texto;
14
15     @PastOrPresent
16     @DateTimeFormat(pattern = "yyyy/MM/dd")
17     private LocalDate fecha;
18
19     @ManyToMany
20     @JoinTable(name = "partidos_noticias", joinColumns = @JoinColumn(name = "noticia_id"),
21         inverseJoinColumns = @JoinColumn(name = "partido_id"))
22     private List<Partido> partidos;
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

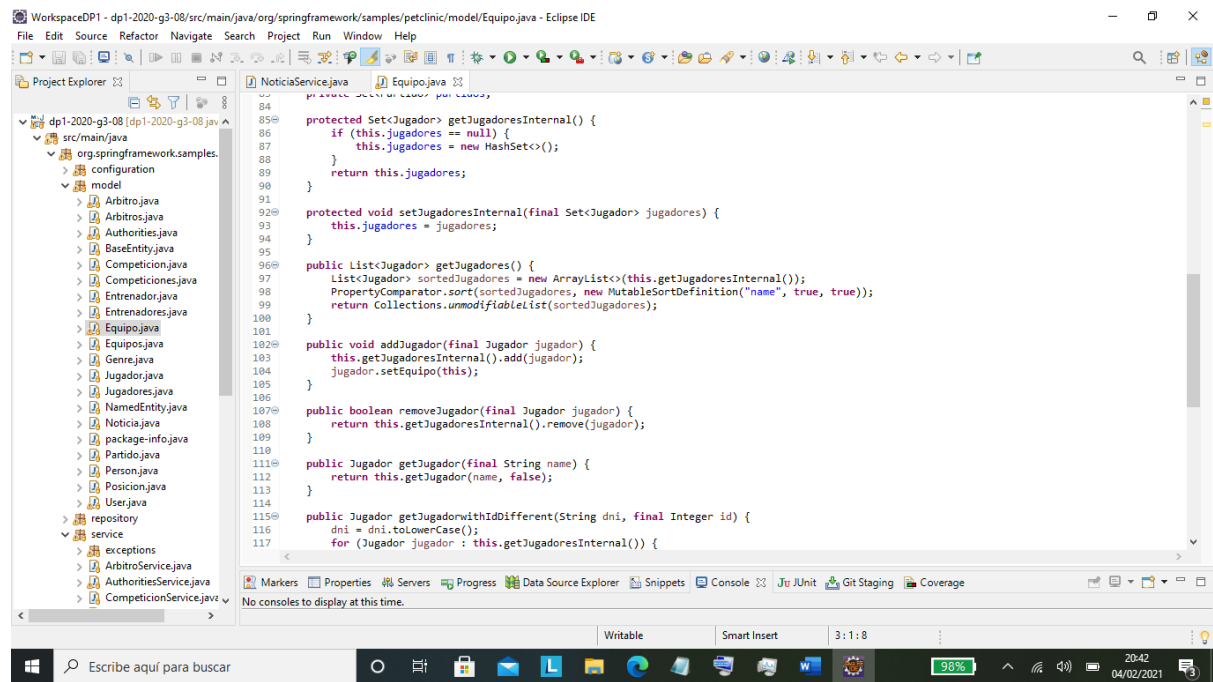
Entidad Equipo, también realizada en colaboración con Gonzalo Fernandez, se encuentra en la misma ruta que la entidad Noticia (src/main/java/org.springframework.samples.petclinic/model)

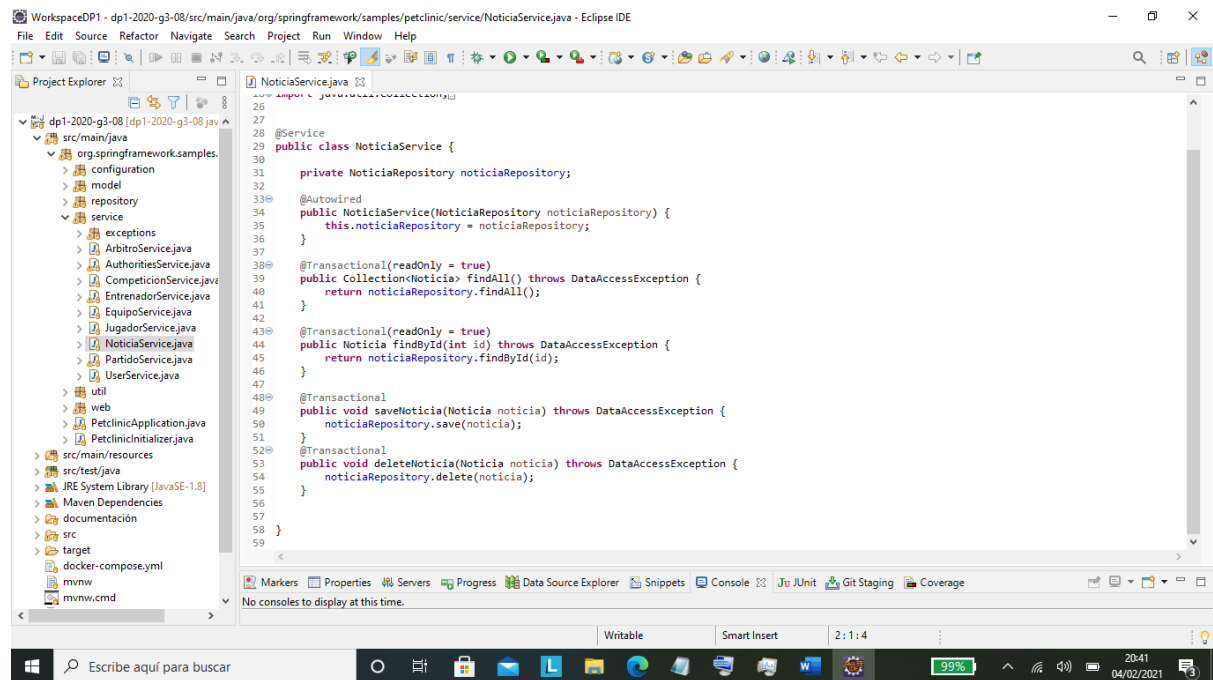
Estamos ante una entidad que implementa en el código una relación Many To Many con otra entidad (la entidad partidos), una relación Many To One (con competición) y una relación One to One (con entrenador).

La entidad también posee varios parámetros con restricciones (ninguno puede estar vacío), que son: nombre (indica el nombre del equipo y tiene que ser de entre 4 y 50 letras), y lugar (región o ciudad de procedencia del equipo).



La entidad también implementa diferentes funciones como getJugadores (obtiene la lista de jugadores del equipo), addJugador (añade un jugador al equipo), removeJugador (elimina un jugador del equipo), getJugador (obtiene un jugador del equipo según el nombre) y getJugadorwithIdDifferent (obtiene jugadores que no coincidan con el dni dado a la función)



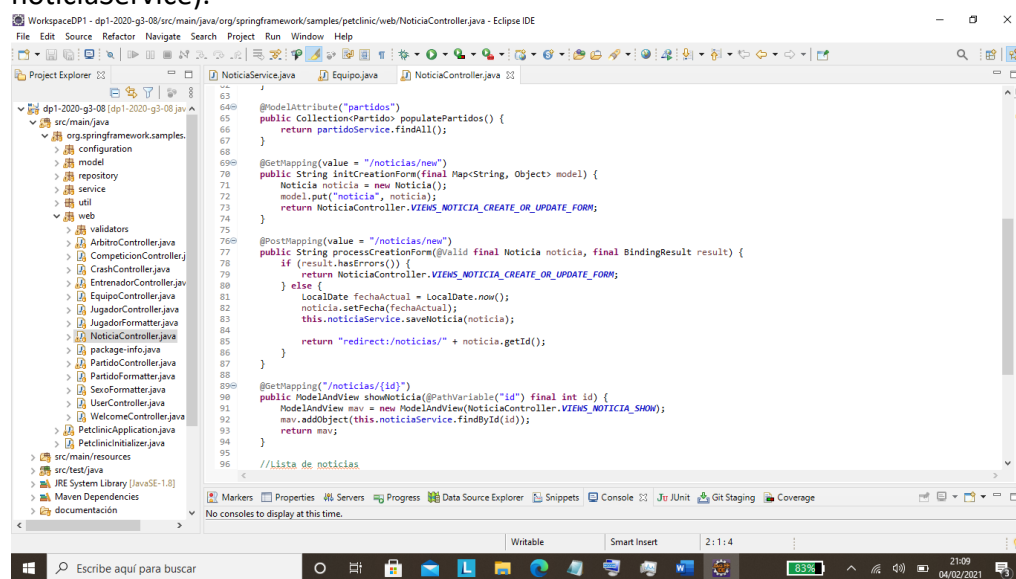


Controlador

Controlador NoticiaController, localizado en la ruta src/main/java/org.springframework.samples.petclinic/web. Realizado en colaboración con Gonzalo Fernandez.

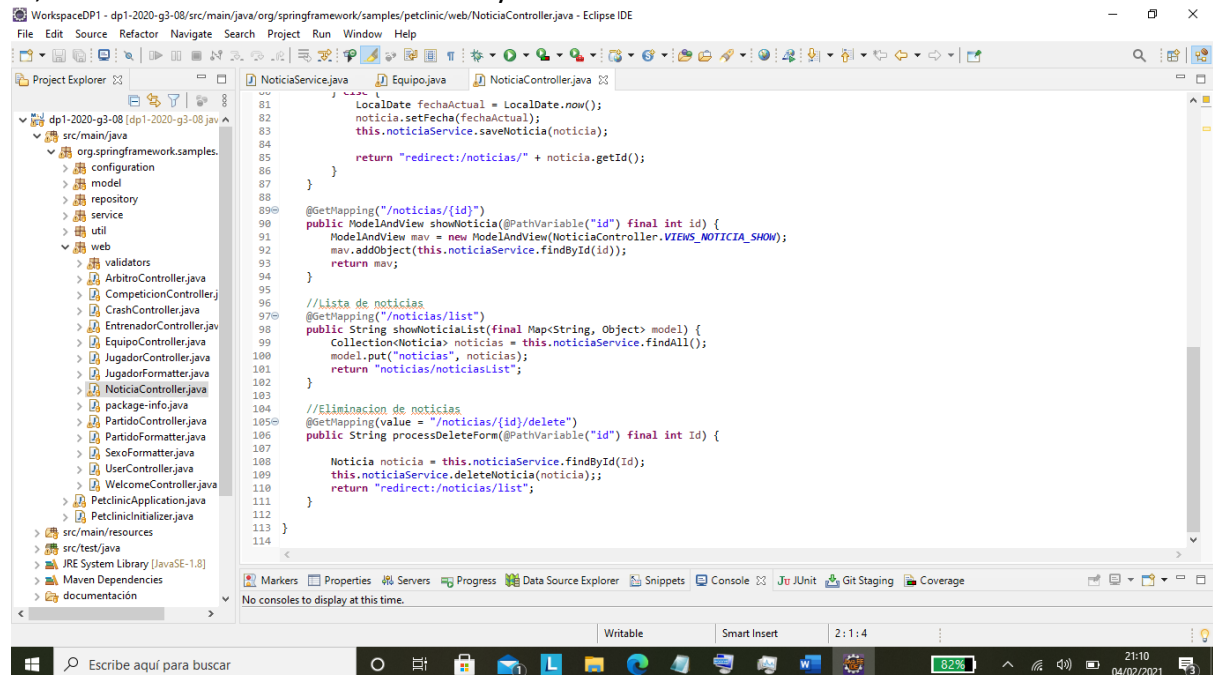
NoticiaController se encarga de responder a los eventos que se producen en la vista de Noticia, invocando cambios en el modelo, y mapeando las peticiones HTTP.

Por ejemplo, podemos ver que implementa mapeo para el Get y el Post de cuando creas una nueva noticia (initCreationForm y processCreationForm) y de como invoca cambios en el modelo (añadiendo la noticia al repositorio por medio del servicio que ofrece noticiaService).



Al crear la noticia, en el processCreationForm, podemos ver como usamos LocalDate.now para introducir como fecha de la noticia, la fecha actual.

Podemos ver que también se implementan peticiones Get de mostrar una noticia usando su id, la de mostrar una lista de noticias y la de borrar una noticia usando su id.



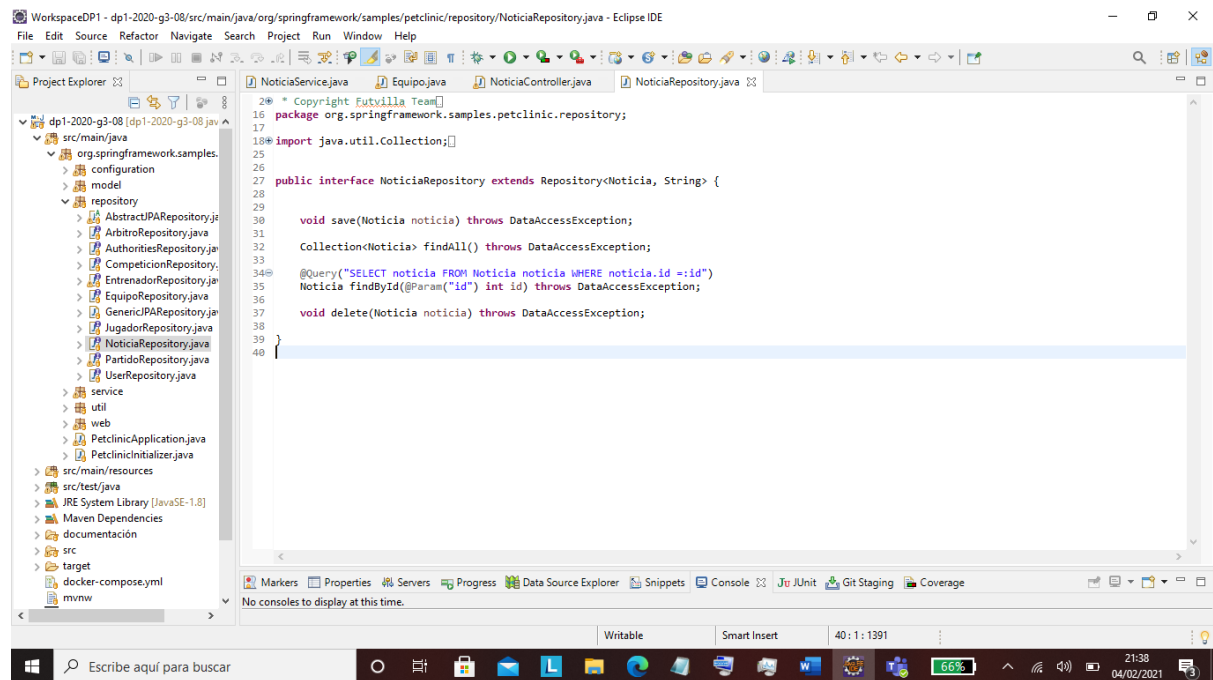
Repositorio

NoticiaRepository, localizado en la ruta

src/main/java/org.springframework.samples.petclinic/repository

NoticiaRepository es una clase que contiene la lógica requerida para acceder a los datos del repositorio de Noticia en la base de datos, haciendo las consultas a esta de forma transparente.

Tiene implementada la lógica necesaria para poder obtener todas las noticias, para poder buscar en la base de datos noticias según el Id, y para poder borrar una noticia.



Validador y anotación asociada

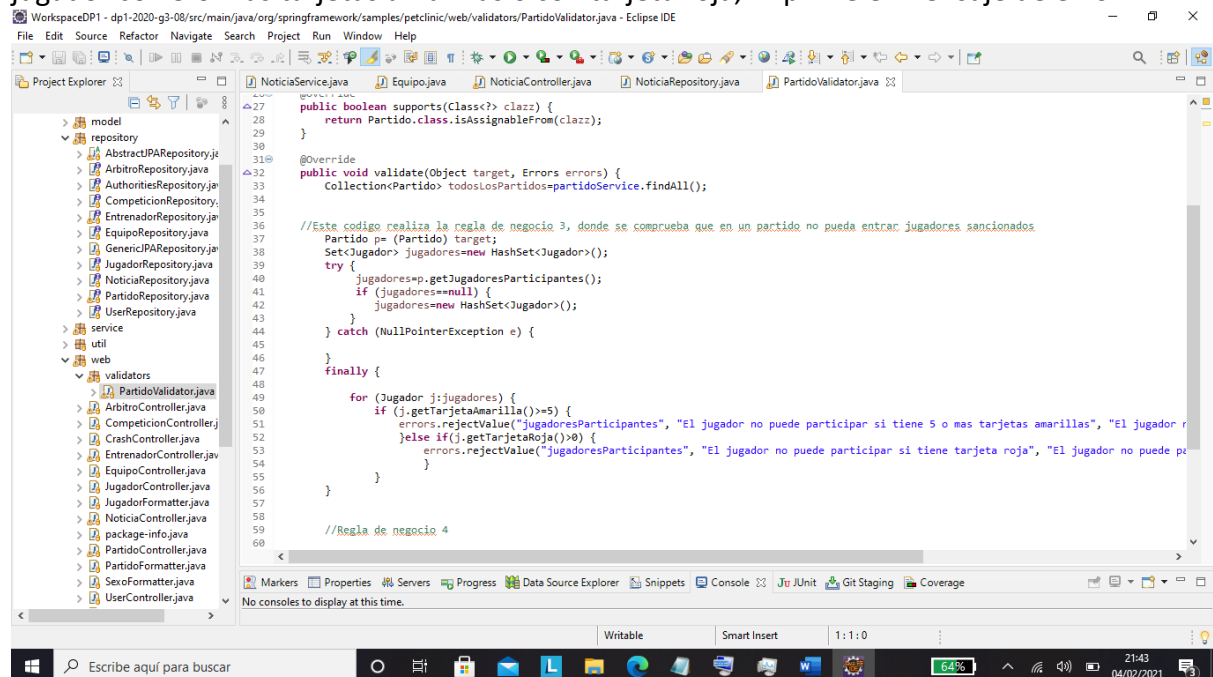
PartidoValidator, realizado junto a mi compañero Gonzalo Fernandez.

Situado en la ruta src/main/java/org.springframework.samples.petclinic/web/validators

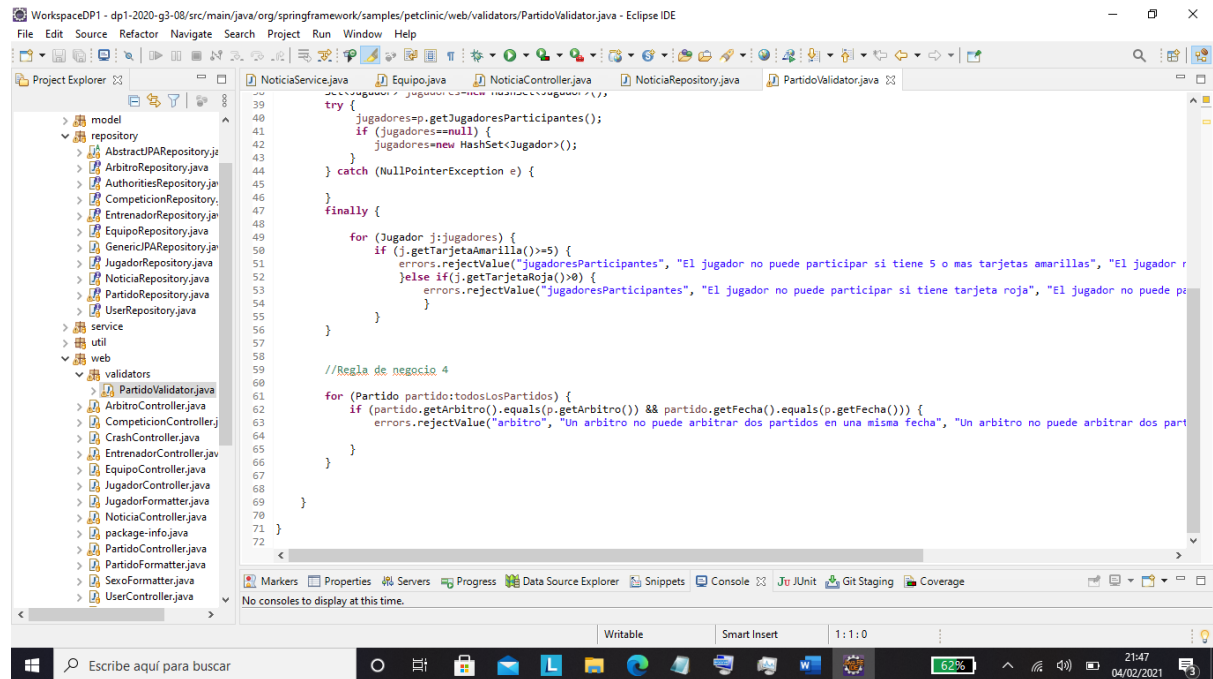
Es un validador necesario para los requisitos de negocio RN3 y RN4

La primera validación consiste en comprobar que en un partido no juega un jugador sancionado.

Esto se comprueba haciendo un bucle for con todos los jugadores del partido, y si hay algún jugador con 5 o mas tarjetas amarillas o con tarjeta roja, imprime el mensaje de error.



La segunda validación consiste en comprobar que un arbitro no esta en 2 partidos en una misma fecha, esto se hace haciendo un bucle for con todos los partidos y comparándolos con el partido que estamos validando, si coinciden tanto el entrenador como la fecha en ambos, se lanza un mensaje de error indicando que un arbitro no puede estar presente en 2 partidos que se disputan a la vez.



Ejemplos de pruebas implementadas

Pruebas unitarias (máximo de dos ejemplos)

Test 1: realizado a NoticiaService dentro del archivo NoticiaServiceTest en la ruta src/test/java/org.springframework.samples.petclinic/service

Realizado en colaboración con Gonzalo Fernandez

Este test consiste en crear una noticia, guardarla en la base de datos por medio del servicio, y comprobar que se ha guardado bien en la base de datos.

Para ello, primero obtenemos el tamaño de la lista de todas las noticias guardadas, luego creamos una nueva noticia, para a continuación guardarla usando los métodos de NoticiaRepository. Por ultimo, comprobamos que la nueva lista de noticias su tamaño sea igual al de la lista original pero incrementado en 1 (debido a la nueva noticia que le añadimos).

@Test

```

@Transactional
public void shouldInsertNoticias() {

```



```

    Collection<Noticia> noticias = this.noticiaService.findAll();
    int found = noticias.size();

    Noticia not = new Noticia();
    not.setFecha(LocalDate.now());
    not.setTexto("Hola soy un ejemplo");
    not.setTitulo("Primicia: el ejemplo");

    this.noticiaService.saveNoticia(not);
    ;
    Collection<Noticia> noticias2 = this.noticiaService.findAll();
    int found2 = noticias2.size();

    Assertions.assertThat(found2).isEqualTo(found + 1);
}

```

Test 2: realizado a EquipoService dentro del archivo EquipoServiceTest en la ruta src/test/java/org.springframework.samples.petclinic/service
Realizado en colaboración con Gonzalo Fernandez.

Este test consiste en comprobar que al eliminar un equipo de la base de datos usando el servicio, efectivamente esta acción se produzca y se elimine el equipo de la base de datos.

Para ello, primero buscamos un equipo presente en la base de datos usando el id y posteriormente lo borramos de la base de datos.

Luego volvemos a buscar el equipo en la base de datos, y comprobamos si el resultado es null (si es null, el equipo se borro, funciona correctamente, si no es null, el equipo no se ha borrado por lo que hay bugs).

```

@Test
@Transactional
public void shouldDeleteEquipo() throws Exception {
    Equipo equipo = this.equipoService.findEquipoById(1);
    this.equipoService.deleteEquipo(equipo);
    Equipo equipoNew = this.equipoService.findEquipoById(1);
    Assertions.assertThat(equipoNew).isEqualTo(null);
}

```

Pruebas de controlador

Proporciono 2 test realizados en EquipoControllerTest (creado por mi de forma individual) en la ruta src/test/java/org.springframework.samples.petclinic/controller

Caso positivo: comprueba si al rellenar los datos correctamente de un equipo en el formulario de creación de un equipo, la pagina se redirige a la direccion url correspondiente (que seria en la que se muestra las competiciones)

```

@WithMockUser(value="admin1")
@Test
void testProcessNewEquipoFormSuccess() throws Exception {

```

```

        mockMvc.perform(post("/competiciones/{competicionId}/equipos/new",
TEST_COMPETICION_ID)
            .param("nombre", "Nombre")
            .with(csrf())
            .param("lugar", "Lugar"))
            .andExpect(status().is3xxRedirection())
            .andExpect(view().name("redirect:/competiciones/{competicionId}"));
    }

```

Caso negativo: comprueba que al rellenarse incorrectamente el formulario de crear un nuevo equipo, se genere el error y la pagina se redirija de nuevo al formulario de crear equipo.

```

@WithMockUser(value="admin1")
@Test
void testProcessNewEquipoFormHasErrors() throws Exception {
    mockMvc.perform(post("/competiciones/{competicionId}/equipos/new",
TEST_COMPETICION_ID)
        .param("nombre", "")
        .with(csrf())
        .param("lugar", "Lugar"))
        .andExpect(model().attributeHasErrors("equipo"))
        .andExpect(status().isOk())
        .andExpect(view().name("equipos/createOrUpdateEquiposForm"));
}

```

Principales problemas encontrados

Ninguno

Otros comentarios

En el proyecto, a parte de lo anteriormente mencionado, también he participado en:

- Hacer los diferentes Mockups (diagrama de dominio, y Modelo de Datos) de en equipo con mi compañero Gonzalo Fernandez.
- Participacion en los documentos junto al resto de los compañeros (decidir entidades, escribir y detallar historias de usuario, los documentos de diseño del sistema, ect etc)
- Implementar la opción de registrarse como entrenador al usuario, en equipo con Gonzalo Fernandez.
- Correcciones en el código de funciones implementadas por otros compañeros, o correcciones ortográficas de lo que se muestra al usuario en la capa de aplicación.