



Tecnologías Web

Curso 2019 – 2020

Creación de un portal para
la gestión de reuniones
corporativas

Óscar Montes Díaz
72131486J

omontes10@alumno.uned.es

1. Introducción

La práctica está desarrollada usando como base el framework *Spring Boot*, que facilita mucho el comienzo de un desarrollo proporcionando las librerías base para comenzar cualquier proyecto Java.

La finalidad de la práctica es construir un desarrollo web siguiendo el patrón de arquitectura de software Modelo Vista Controlador (MVC), este patrón propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Es por ello que la configuración del proyecto es siguiendo el modelo Spring MVC, teniendo como dependencia en el pom de *Maven* “spring-boot-starter-web”, con esta dependencia ya tendremos las librerías de “spring-webmvc” y “spring-web”.

Para la parte de persistencia se trabaja con *Java Persistence API (JPA)* apoyándonos en Spring, por lo que tengo la dependencia que ya me lo trae todo de *Spring Boot* “spring-boot-starter-data-jpa”. Como base de datos utilizo *H2* en su modo en memoria, la ventaja que he tenido con *H2* es que puede crear la estructura de las tablas basándose en las entidades del proyecto y con un simple script (data.sql) hago la carga de datos inicial de la aplicación.

Para la parte frontal de la aplicación me he apoyado en el gestor de plantillas *Thymeleaf*, del cual tiene su propia dependencia “starter” Spring Boot (spring-boot-starter-thymeleaf). Aunque los motores de plantillas no es algo nuevo y han existido desde los principios de Java no siempre han sido las tecnologías más usadas. La competencia con JSP y JSTL siempre ha existido. Sin embargo, me gusta mucho más la idea de trabajar con ficheros HTML en vez de con JSP, para mí es algo mucho más sencillo y aunque es la primera vez que trabajo con *Thymeleaf*, me pude adaptar con facilidad.

Para terminar de decorar la web, utilicé dos veteranos ya en el desarrollo web, las bibliotecas *Bootstrap* para los estilos CSS y *jQuery* para algunas funciones Javascript que necesité implementar, como por ejemplo alguna validación de campos.

Spring Boot me ha ayudado mucho con esta práctica, entiendo que ahora la mayoría de los nuevos desarrollos (sobre todo los enfocados a microservicios) estén basados en él. El núcleo central del desarrollo en la parte *back* no me llevó más de dos o tres días, en ese poco tiempo ya podía arrancar la aplicación con una base de datos en memoria y probar los *tests* con la mayoría de los servicios ya completados.

Thymeleaf ha sido mano de santo para mí, nunca me he sentido del todo cómodo en un desarrollo web y esta librería ha sido todo un descubrimiento para mí, el código queda muy limpio en mis páginas y facilita mucho el traslado de información del *front* al *back* y esto pudo acelerar mi primera toma de contacto con los formularios de creación de Evento y mostrar los diferentes listados. Ya podemos olvidarnos del engorroso trabajo de JSTL, en mi opinión, en un desarrollo de puro MVC, *Thymeleaf* deberá ser la primera solución.

Todos los formularios de la aplicación están validados con *Java Bean Validation*, un *framework* que de forma muy sencilla nos valida los campos, es un excelente compañero

para olvidarnos de extraños algoritmos de validación y que nuestro desarrollo sea mucho más robusto ante datos no deseados.

La primera parte del desarrollo fue tener una base de back estable, es donde más seguro me siento y estuve haciendo todas las entidades, repositorios y servicios necesarios, poniendo una cobertura de *tests* sobre los métodos de los servicios y asegurándome de que no había ningún problema para recuperar o persistir los datos.

Cuando ya tenía una base de servicios importante, empecé a planificar el *front* y a desarrollar los controladores.

La parte final del desarrollo fue mejorar el diseño, hacer pruebas en la máquina de AWS y documentación.

Sobre la parte de las pruebas en la máquina AWS, fue interesante usar un entorno *cloud* para las pruebas, pero han sido bastante tortuosas, con un poco de uso del Tomcat dejaba de responder y tenía que pararlo y volver a arrancarlo, en alguna ocasión no era suficiente y tuve que parar la maquina completa y volverla a arrancar, parece que se quedaba sin recursos con facilidad. Por otro lado esto me hizo recapacitar sobre alguno de mis diseños, por ejemplo para evitar una sobre carga de datos cambié las relaciones *@OneToMany* y *@ManyToOne* que tenía en un primer momento como “Eager” a “Lazy” y dejar de cargar toda la información que tenían en base de datos, me era útil para facilitar el desarrollo, pero a la larga podía ser un problema de memoria. También en AWS tuve problemas con algunas rutas, por tener una mala configuración, así que mejoré mi desarrollo gracias a tener que desplegar en este entorno.

2. Composición del grupo y roles

He estado solo en este equipo, pero eso no me ha impedido repartirme el trabajo en roles en distintas fases del desarrollo.

Empecé siendo plenamente un desarrollador *back* y *QA*, montando las entidades, servicios y pruebas unitarias con Junit.

Con alrededor de 50 *tests* funcionando, empecé a ser diseñador web, esta faceta no es con la que me encuentro más cómodo, pero si tengo claro que me gustan las páginas web sencillas y claras, busqué algunos diseños con pestañas que me gustaran y apliqué uno con *JQuery* y *Font-Awesome* que me parece interesante.

Una vez decidido el diseño me puse en el rol de desarrollador web, utilizando *Bootstrap*, *JQuery* y *Thymeleaf*. La elección de *Thymeleaf* la tuve clara, queriendo estar firme en el modelo *MVC*, *Thymeleaf* se acompla a la perfección en este modelo, es una librería muy sencilla de aprender y sin duda me aceleró y simplificó la parte de desarrollo web.

Otro rol importante es el de *QA*, trabajo que he desempeñado en la parte final probando la aplicación en Tomcat 9 y en el entorno *AWS* y descubriendo pequeños problemas no detectados durante el desarrollo.

Mi último rol ha sido el de documentalista, escribiendo este documento y refinando el *Javadoc*.

3. Plan de trabajo

El plan de trabajo se compone de cinco semanas, dando por comienzo el 4 de mayo y finalizando el 7 de junio.

La primera semana será del 4 al 11 de mayo, la segunda semana del 12 al 18 de mayo, la tercera semana del 19 al 25 de mayo, la cuarta semana del 25 de mayo al 1 de junio y la última semana del 2 al 7 de junio.

Cada semana tiene una serie de hitos que debo cumplir para poder llegar a la fecha de finalización con todo realizado.

3.1 Primera semana 04/05/2020 – 11/05/2020

El objetivo de la primera semana es tener una versión *alfa* que cubra de punta a punta el desarrollo, pudiendo dar de alta eventos y usuarios y que estos se persistan en base de datos.

04/05/2020

Diseño del modelo de datos

Se diseña un primer modelo de datos a partir del enunciado de la práctica, se deciden entidades como *Event* para Eventos, *User* para Usuario, *Rol* para el Rol de usuario, *Room* para salas y *RoomTechResource* para los medios técnicos de la sala.

Montar entorno de desarrollo

El primer objetivo fue montar un desarrollo con *Spring Boot 2.2.6*, usando *H2* como base de datos en memoria y teniendo como repositorio uno privado en *GitLab*.

Se crean las primeras entidades con sus repositorios y servicios (*User*, *Rol*, *Room*, *RoomTechResource*) y se insertan los primeros “inserts” para estas entidades en *data.sql*, el script que generará nuestros datos en memoria cada vez que arranque la aplicación.

05/05/2020

Modelado y primeras entidades en el proyecto

Con el deseo de probar el funcionamiento de la base de datos en memoria y la generación de tablas de forma automática, se termina de modelar las entidades *Event*, *EventEnroll* (entidad donde se guardan los usuarios suscritos al evento) y *EventWaitingList* (para la lista de espera) con sus repositorios y servicios y se crean tests unitarios (JUnit) de todos los métodos de los servicios que existen hasta este momento, comprobando que *Spring Boot* arranca, genera las tablas a partir de las entidades y hace los “inserts” desde *data.sql*.

Se encuentra en común el identificador de todas las entidades (Long ID), por lo que se saca a una clase externa (*EntityIdentifiable*) de la cual extienden todas las entidades.

Debo refactorizar todo el código para juntar las clases por entidades, tendremos el paquete *event*, el paquete *room* y el paquete *user*.

Se completa la parte básica del back, teniendo todos los Service, Repository y Entidades, en los service están los métodos básicos de CRUD y sus respectivos tests.

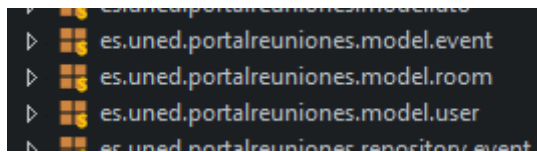


Figura 3.1 Paquetería de las entidades.

Se toma la decisión de trabajar con *Thymeleaf* para comunicar la información del *front* con el back, se añade su dependencia de la versión con *Spring Boot* y se comienza con el desarrollo de los dos primeros controladores, *EventController* y *UserController*, teniendo un sencillo formulario de alta de usuarios y de eventos.

06/05/2020

Se hace una primera versión de votación de horas por parte de los usuarios invitados del evento.

Se trabaja ya a nivel *full stack*, creando los métodos en los servicios necesarios para trabajar con el *front*, como recuperar las horas que ha votado un usuario.

07/05/2020

Se añaden más elementos en las entidades, como el token en evento y su generación cuando se crea por parte del administrador/jefe de proyecto.

09/05/2020

En vista de tener un *front* más ordenado, se empiezan a refactorizar sus carpetas y mover los ficheros HTML dependiendo de su funcionalidad, la parte de usuario del *front* en la carpeta "user", la parte de evento en "event" y la parte del administrador en "admin".

11/05/2020

Se comienza a trabajar en el panel del usuario, mostrando una lista de los eventos a los que ha sido invitado.

3.2 Segunda semana 12/05/2020 – 18/05/2020

Hasta ahora no ha habido una gestión de usuarios y una pantalla de *login* para saber qué usuario está conectado, se desea en esta segunda semana tener una pantalla de *login* y una mínima gestión de roles.

En esta semana se espera mejorar el *front* añadiendo *Bootstrap*, *Font-Awesome* y *JQuery* y se hará una prueba de despliegue, en vez de AWS en un Tomcat 9 local.

15/05/2020

En los test de Junit de borrado se detecta que no se puede borrar el evento por las muchas dependencias que tienen a nivel de base de datos, es por eso que se añade el borrado en cascada en *Event* de todas las entidades que dependen del evento.

```
@OneToMany(mappedBy = "event", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE)
private Set<EventFile> files;

@OneToMany(mappedBy = "event", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE)
private Set<EventEnroll> enrolls;

@OneToMany(mappedBy = "event", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE)
private Set<EventWaitingList> waitingList;

@OneToMany(mappedBy = "event", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
private Set<EventPreferenceSchedule> preferenceSchedule;

@OneToMany(mappedBy = "event", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
private Set<EventInvited> userInvited;
```

Figura 3.2 Algunas de las dependencias de la entidad Event.

Se añade la entidad *EventFile* para poder guardar en BBDD los ficheros que se adjuntan en el evento, se crea su repositorio y servicio y se llama a este guardado desde *EventController*.

Después de varias pruebas de seguridad con Spring Security, se decide prescindir de esta seguridad y crear un *login* más sencillo, donde en su controlador *LoginController* guardaremos en sesión los datos de usuario y con estos datos gestionaremos la seguridad y los permisos.

Se hace una primera versión del chat del evento, se utiliza *WebSocket* de *Spring Boot* para el back y las librerías de JavaScript *Sockjs* y *Stomp*.

16/05/2020

Se añaden las dependencias en el *pom* de *jQuery* y *Bootstrap* aprovechando las librerías de *org.webjars*.

```
</dependency>
<!-- Bootstrap -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>3.3.6</version>
</dependency>
<!-- jQuery -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>jquery</artifactId>
  <version>1.9.1</version>
</dependency>
```

Figura 3.2.1 Las dependencias de org.webjars en el pom de Maven

Se empiezan a aplicar algunos estilos al *front*.

17/05/2020

Desarrollo de la actualización de usuarios y se empieza a integrar el chat en la ventana de la celebración del evento.

18/05/2020

Se añade a la entidad del evento el día y la hora en la que se celebrará y el email al usuario.

Se empiezan a mostrar los ficheros subidos en la ventana de la celebración del evento.

En las pruebas de despliegue en el Tomcat 9 se detecta que no carga bien *Bootstrap* y *jQuery* desde las librerías de *webjars*, se decide eliminar estas dependencias y se añade *Bootstrap* y *jQuery* de forma manual en la carpeta *static* del proyecto.

3.3 Tercera semana 19/05/2020 – 25/05/2020

En esta tercera semana se desea trabajar en el aspecto del *front*, se busca un aspecto a nivel de pestañas donde se pueda reunir mucha información en una sola página.

Se desea tener acababa ya casi la mayor parte de la lógica de la celebración de un evento.

21/05/2020

Se empieza a modificar el panel del usuario para que se empiecen a mostrar las secciones solo del administrador, esto es enviando una variable al *front* booleana sobre si se puede mostrar o no.

El administrador comienza a ver en su panel el listado de eventos que faltan por votar los usuarios.

22/05/2020

Se comienza el desarrollo del detalle del evento.

Se modifica la ventana de la celebración para que muestre los usuarios inscritos en él (una nueva pestaña).

Se desarrolla la lógica en el *back* de la votación, saliendo el día y la hora más votados.

23/05/2020

Se hace la parte del borrado de ficheros del evento por parte del organizador del evento.

Se crea una pestaña nueva en la celebración del evento donde veremos todos los mensajes que se han escrito, para así poder ver la conversación completa y que el organizador del evento pueda borrar algún mensaje.

Se hace la lógica del borrado de mensajes del chat por parte del organizador.

24/05/2020

Se empiezan a validar los datos que se introduce en los formularios, para ellos se utiliza “Java Bean Validation”, por lo que se desarrollan una serie de *beans* para los formularios que se quieren validar, sustituyendo estos a los entity.

Se decide utilizar *DataTables* con *Bootstrap* para la búsqueda de usuarios en la creación del evento, se encuentran muchos problemas para no perder a los usuarios seleccionados cuando cambias de página, por lo que cada usuario seleccionado se guarda en un campo oculto en la página y después se guarda en sesión, el funcionamiento parece bastante correcto.

25/05/2020

Se añade en la creación del evento la lista de espera, es una búsqueda con una tabla igual que la de usuarios.

En la configuración del evento podemos trasladar usuarios de la lista de espera a invitados y viceversa.

Con todo esto, añadido un contador donde se muestra cuanto aforo disponible queda en la sala.

3.4 Cuarta semana 26/05/2020 – 01/06/2020

En esta semana quiero dar por concluido el desarrollo, por eso me centraré sobre todo en validaciones y pulir pequeñas cosas.

26/05/2020

No se había valorado que el organizador desee meter varias fechas cuando crea el evento y que los usuarios en la votación decidan qué fecha es la mejor, se cambia el *front* para poder añadir varias fechas y a nivel de back se recupera un array de fechas para trabajar con estos datos.

Se hace la validación con la que un evento no puede superar en asistentes el tamaño de una sala.

27/05/2020

Se añade en la ventana de configuración del evento que el organizador pueda cambiar de sala antes de que finalice la votación.

En la configuración del evento se deshabilita el botón de configurar hasta que no hayan votado todos los usuarios invitados.

Hago la parte de validación para no poder coger salas que ya están reservadas en otro evento.

Se añade la funcionalidad de usuarios no registrados, creando una nueva entidad *UserNotRegistered*, lo cual vincula ese evento con el usuario, este usuario creado se guarda en la tabla *User* con un rol “0” de no registrado, no tendrá contraseña ni correo, únicamente un nombre. Solo podrá entrar al evento al que se le ha asignado en

UserNotRegistered, cuando sea invitado a otro evento, volverá a introducir su nombre y se generará otro usuario no registrado.

Desde el panel de usuario, en la bandeja de entrada de eventos, añadido un botón para que el usuario se pueda dar de baja del evento.

29/05/2020

Me faltaba por mostrar en el evento las características técnicas de la sala, ahora en una pestaña nueva, podremos ver cuales tiene.

Todavía podíamos escoger salas que no estuvieran disponibles por su configuración horario, añadido una validación nueva para impedir este caso.

Se añade un nuevo tipo de excepción para controlar que un usuario no pueda entrar a un evento donde no está suscrito o para que un usuario no pueda entrar al detalle de un evento finalizado si a este no está invitado o suscrito. En caso de tener prohibida la entrada a este evento, le llevará a una pantalla donde le informará que no puede visitarlo.

Se añade el control de no poder escribir más mensajes ni subir ficheros si el organizador del evento lo finaliza, esto te sacará a una pantalla donde te informará de que el evento ha finalizado.

30/05/2020

Se realizan las primeras pruebas de despliegue en AWS

Se comienza a escribir la memoria de la práctica.

3.5 Quinta semana 02/06/2020 – 07/06/2020

En esta semana entramos en fase de pruebas y documentación, dando por finalizada la práctica.

Sobre todo se prueban todos los formularios con distintos valores, encontrando bastantes problemas, dejando campos con espacios, enviando letras en lugar de números... Después de escribir una lista con todos los problemas detectados se van solucionando y probando de nuevo en el entorno AWS.

Se mejoran mensajes y avisos, por ejemplo se le proporciona el enlace al evento para que lo pueda visitar una vez finalizado en la ventana de finalización de eventos.

4. Especificación de Requerimientos

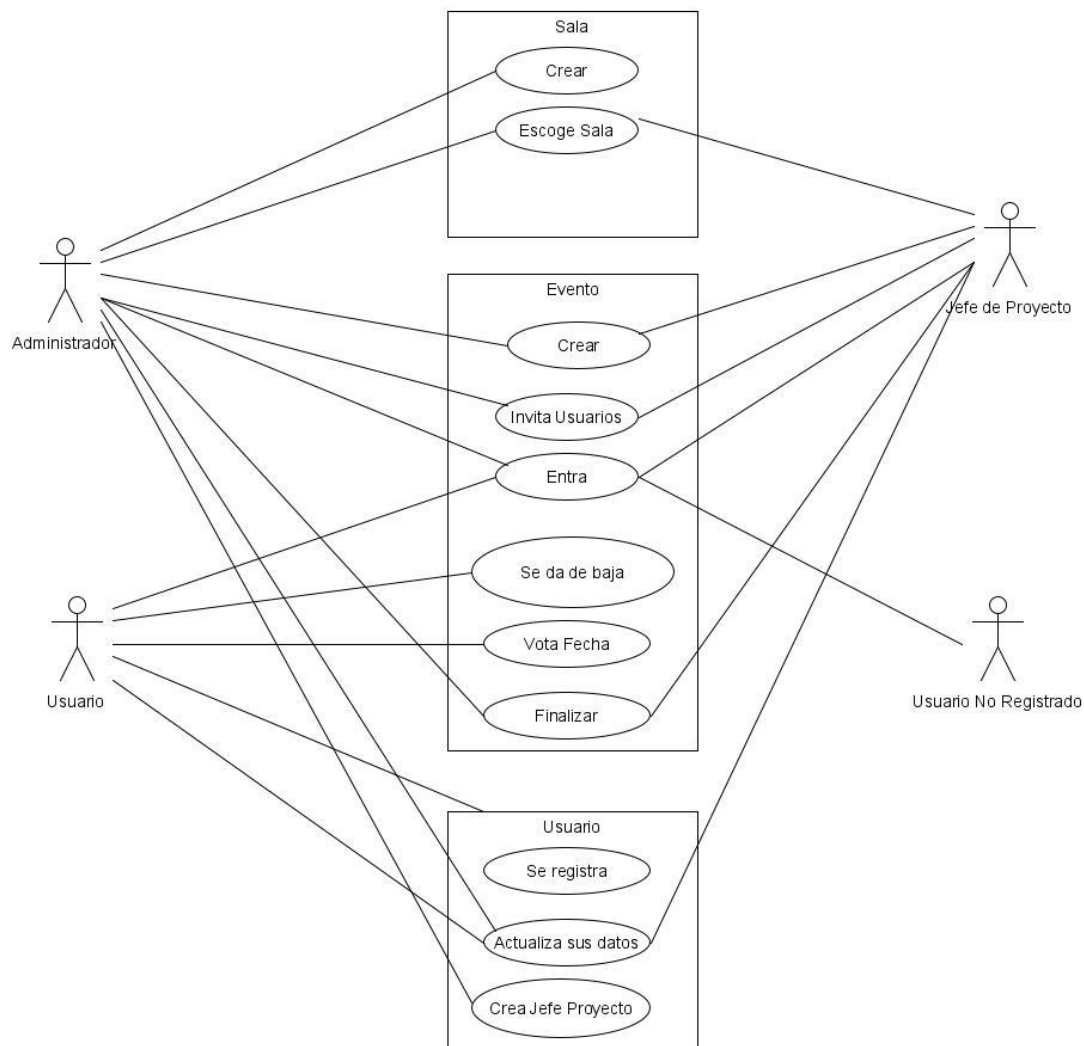


Figura 4.1 Actores y acciones que pueden hacer

Caso 1. Alta de nuevos usuarios

Los nuevos usuarios pueden venir por dos caminos, un administrador puede dar de alta nuevos jefes de proyecto y una persona puede registrarse en el sistema como nuevo usuario, este usuario nunca se dará de alta como administrador ni como jefe de proyecto.

En ningún caso se puede dar de alta un usuario con un nick que ya tenga otro usuario.

Caso 2. Alta de nuevas salas

Un administrador puede crear nuevas salas, escogiendo su nombre, recursos técnicos, capacidad y disponibilidad horaria.

Nunca se podrán crear dos salas con el mismo nombre.

Caso 3. Crear evento

Un administrador y un jefe de proyecto pueden dar de alta nuevos eventos. En estos eventos deben seleccionar una sala, elegir una preferencia de fechas y horas, invitar a usuarios tanto registrados como no registrados y crear una lista de espera de usuarios.

Solo se podrán escoger las salas que estén disponibles en ese horario.

El organizador del evento verá el evento creado en el apartado “Eventos pendientes” mientras se está realizando la votación y después en “Mis Eventos” cuando ya esté organizado.

Caso 4. Actualización de datos personales

Los administradores, jefes de proyecto y usuarios pueden actualizar sus datos personales, el *nickname* y el rol no pueden ser cambiados en ningún momento.

Caso 5. Recibir invitación al evento

Cualquier perfil de usuario pueden recibir una invitación para acudir a un evento, los administradores, jefes de proyecto y usuarios registrados podrán participar en la votación de la fecha y hora, los usuarios no registrados únicamente podrán acudir al evento a través de un enlace con un token que les proporcionará el creador del evento.

Los usuarios registrados verán las invitaciones a eventos en el apartado de invitaciones de la aplicación y podrán entrar a votar la fecha y horas pulsando sobre el evento.

El organizador del evento verá los eventos que ha creado para poder entrar en el apartado “Mis Eventos”.

Caso 6. Votación de día y fecha del evento

Los administradores y jefes de proyecto serán los organizadores del evento y configuran unas fechas posibles y unas horas del evento. Los usuarios registrados invitados al evento deberán votar sobre esas fechas y horas la elección que mejor les convenga, solo podrán escoger un día, pero si podrán elegir entre varias horas de las seleccionadas por el organizador del evento.

Cuando terminen todos los invitados de votar, el organizador podrá acceder a la votación desde el apartado “Eventos pendientes” y ver lo que ha votado cada usuario y la fecha y hora más votadas, aún así podrá cambiar esta fecha y hora si le conviene en ese momento, si no cambia nada, el evento se dará de alta con lo más votado.

Caso 7. Cambiar la sala del evento

Mientras no se haya concertada la fecha y hora definitivas del evento, el organizador podrá cambiar la sala, siempre que esta esté disponible para esa hora y tenga el aforo suficiente para que puedan entrar todos los invitados. Este cambio se hará en la misma ventana donde el organizador sigue la votación.

Caso 8. Utilizar la lista de espera

Mientras no se haya concertada la fecha y hora definitivas del evento, el organizador podrá retirar de la lista de invitados y añadir a ella usuarios que estén en la lista de espera, siempre que estos cambios no superen el aforo de la sala del evento.

Caso 9. Entrar al evento

Tanto el organizador como los invitados (ya sean registrados o por token) podrán entrar al evento al menos cinco minutos antes de la hora y día del evento.

En el caso de usuarios registrados verán este evento en su bandeja de entrada de próximos eventos desde el mismo momento que el organizador lo haya configurado con la fecha y hora votadas, entrarán pulsando sobre el evento.

En el caso de los usuarios no registrados, el organizador del evento les proporcionará un enlace para poder suscribirse al evento y entra a él al menos cinco minutos antes de la hora y día del evento.

Caso 10. Escribir en el chat del evento

Mientras el evento se esté celebrando, todos los participantes del evento podrán escribir en el chat en el apartado correspondiente para ello y ahí podrán mantener una conversación con el resto de usuarios asistentes.

En otro apartado podrán ver la conversación completa, incluido lo que se ha hablado en su ausencia.

Caso 11. Borrar mensajes del chat del evento

Mientras el evento se esté organizando, el organizador del evento podrá borrar mensajes del chat en la ventana donde se puede ver la conversación completa, estos mensajes desaparecerán y no podrán ser recuperado.

Caso 12. Subir ficheros al evento

Mientras el evento se esté organizando, todos los participantes podrán subir ficheros en el apartado de ficheros del evento, verán tanto sus ficheros como los que están subiendo todos los participantes

Caso 13. Borrado de ficheros del evento

Mientras el evento se esté organizando, el organizador podrá eliminar uno a uno los ficheros que se estén subiendo al evento por parte de todos los participantes del evento.

Caso 14. Finalizar Evento

El organizador del evento podrá finalizar el mismo en el momento que crea oportuno, para ello tendrá un botón para cerrarlo.

El resto de participantes recibirán un aviso de evento cerrado cuando intenten escribir en el chat o subir un fichero impidiendo que hagan nada más en el evento.

Los usuarios registrados una vez finalizado, podrán entrar a ver los ficheros y la conversación del evento en el apartado de eventos finalizados del panel de usuario.

Los no registrados obtendrán el enlace para visitar el evento finalizado una vez se finalice.

Caso 15. Invitado al evento, pero no asistente

En caso de recibir una invitación al evento, pero que el horario no encaje con la votación, o sea, que escoja otro día o que no coincida ni en el día ni en ninguna de las horas posibles, no se le registrará como asistente al evento y no podrá entrar en él, pero si podrá ver el evento una vez finalizado en el apartado “Eventos finalizados”.

5. Esquema de base de datos

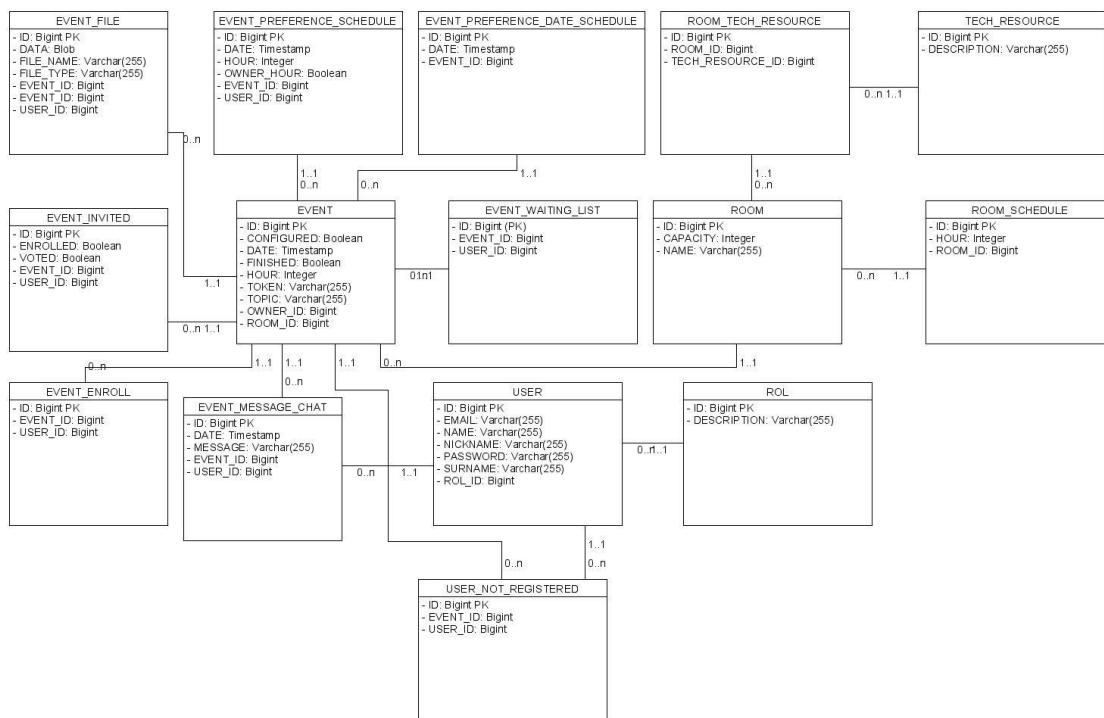


Figura 5.1 Esquema de la base de datos de la práctica

Las tablas de esta base de datos son:

Para los eventos: EVENT, EVENT_FILE, EVENT_PREFERENCE_SCHEDULE,
EVENT_PREFERENCE_DATE_SCHEDULE, EVENT_INVITED, EVENT_WAITING_LIST,
USER NOT REGISTERED, EVENT MESSAGE CHAT, EVENT ENROLL

Para las salas: ROOM, ROOM SCHEDULE, ROOM TECH RESOURCE, TECH RESOURCE

Para los usuarios: USER, ROL.

- EVENT: Guardar los eventos.
- EVENT_FILE: Guardar los ficheros del evento
- EVENT_PREFERENCE_SCHEDULE: Guardar las preferencias en la votación de las horas.
- EVENT_PREFERENCE_DATE_SCHEDULE: Guardar las preferencias en la votación de las fechas.
- EVENT_INVITED: Invitados al evento.
- EVENT_WAITING_LIST: Usuarios en lista de espera.
- USERT_NOT_REGISTERED: Usuarios invitados no registrados.
- EVENT_MESSAGE_CHAT: Mensajes de chat del evento.
- EVENT_ENROLL: Usuario registrados inscritos al evento.
- ROOM: Las salas.
- ROOM_SCHEDULE: Horario de uso de la sala.
- ROOM_TECH_RESOURCE: Medios técnicos de la sala.
- TECH_RESOURCE: Medios técnicos existentes.

6. Arquitectura

Para la realización de la práctica se ha usado el clásico esquema Modelo Vista Controlador.

El Modelo

El modelo van a ser nuestras propias entidades y en algunas ocasiones un *bean* para añadirle validaciones de datos desde la vista, *Spring* nos ayuda mucho en este aspecto, para poder enviar los objetos a la vista y que esta pueda trabajar con ellos nos ofrece *Model*, *ModelMap* y *ModelAndView*. Yo he utilizado *Model* para todas estas tareas.

```
@RequestMapping("/panel")
public String showUserPanel(HttpServletRequest request, Model model) {
    try {
        checkSession(request);
        Long idUser = (Long) Sessions.getValue(request, Session.ID_USER);
        logger.info("Entramos a su panel el usuario con ID:: {}", idUser);
        model.addAttribute("eventsInvited", eventInvitedService.findEventInvitedByIdUser(idUser));
        model.addAttribute("eventsEnrolled", eventEnrollService.findByIdUser(idUser));
        model.addAttribute("myEvents", eventService.findByIdOwner(idUser));
        model.addAttribute("myEventsFinished", eventService.findFinishedByIdOwner(idUser));
        model.addAttribute("pendingEvents", eventService.findPendingEventByIdOwner(idUser));
        model.addAttribute("finishedEvents", eventInvitedService.findEventInvitedFinishedByIdUser(idUser));
        model.addAttribute("idUser", idUser);
        model.addAttribute("user", userService.findById(idUser));
        if (!model.containsAttribute(USER_UPDATE DTO)) {
            model.addAttribute(USER_UPDATE DTO, userService.findUserDTOById(idUser));
        }
        Sessions.removeValue(request, Session.USERS_SELECTED);
        setSucessMessage(request, model);
        completeModelSession(request, model);
    } catch (ElementNotFoundException e) {
        logger.error("No se encontró el elemento {}", e.getMessage());
        return GO_LOGIN;
    } catch (SessionNotFoundException e1) {
        logger.warn("No hay sesión {}", e1.getMessage());
        return GO_LOGIN;
    }
    return USER_PANEL;
}
```

Figura 6.1 Ejemplo de una llamada en el controlador donde se ve el uso de Model.

La parte del modelo lo cubren las entidades, representando las tablas de la base de datos y los repositorios, donde se han ido escribiendo las distintas consultas SQL que han sido necesarias.

La Vista

Para la vista utilizamos *Thymeleaf*, por los que las páginas son ficheros html en vez de jsp.

Thymeleaf es bastante simple de utilizar y hacer que interactúe con el controlador es tan sencillo como agregar unas expresiones nuevas a nuestras etiquetas HTML de toda la vida.


```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<div th:replace="common/libraries :: libraries"></div>
<title>Portal</title>
</head>
<body>
<div th:replace="common/header :: header"></div>
<div class="container login-container">
<div class="row d-flex justify-content-center">
<div class="col-md-6 login-form-1">
<h3 th:text="#{login.welcome}"></h3>
<form action="#" th:action="@{/login}" th:object="${login}"
method="POST">
<div class="alert alert-danger" th:if="${#fields.hasErrors('*')}">
<p th:each="err : ${#fields.errors('*')}" th:text="${err}"></p>
</div>
<div class="form-group">
<input type="text" class="form-control"
th:placeholder="#{login.user}" th:field="${nickname}" />
</div>
<div class="form-group">
<input type="password" class="form-control"
th:placeholder="#{login.password}" th:field="${password}" />
</div>
<div class="form-group mb-3">
<button type="submit" class="btnSubmit" th:text="#{login.button.enter}"></button>
<a th:href="@{/user/newUser}" class="col-sm-3" th:text="#{login.register}"></a>
</div>
</form>
<div class="form-group form-group-center">
<a th:href="@{/doc/Memoria.pdf}" target="_blank" th:text="#{login.document.download}"></a>
</div>
<div class="form-group form-group-center">
<a th:href="@{/javadoc/index.html}" target="_blank" th:text="#{login.javadoc}"></a>
</div>
</div>
</div>
<div th:replace="common/footer :: footer"></div>
</body>
</html>

```

Figura 6.1 Pantalla de login donde se pueden ver las expresiones de Thymeleaf

El Controlador

El controlador es el punto de unión entre el modelo y la vista y como su nombre indica se encarga de controlar la vista que tiene que mostrar, los datos que se incluirán en esta y de recopilar los datos enviados desde la vista para actuar en consecuencia (guardarlos, hacer una consulta, etc.).

Entre el controlador y el modelo tendremos la parte de la lógica de la aplicación, que son los servicios, donde se harán la mayoría de las funcionalidades.

```

/**
 * Gestión de los roles de usuario
 *
 * @author omonetes
 *
 */
public interface RolService {

    /**
     * Se recuperan todos los roles del sistema
     *
     * @return
     */
    List<Rol> findAll();

    /**
     * Recuperar un rol por su identificador
     *
     * @param idRol
     * @return
     * @throws ElementNotFoundException
     */
    Rol findById(Long idRol) throws ElementNotFoundException;
}

```

Figura 6.3 Interfaz del servicio del Rol de usuario

Entre el modelo y los servicios tenemos la parte de los repositorios, la parte de la aplicación donde se harán todas las consultas SQL necesarias.

```

@Repository
public interface RoomRepository extends JpaRepository<Room, Long> {

    @Query("SELECT r FROM Room r, Event e WHERE r.id = e.room.id AND r.id = :idRoom AND e.date = :date AND e.hour = :hour AND e.id != :idEvent")
    List<Room> findByDateHour(@Param("idRoom") Long idRoom, @Param("idEvent") Long idEvent, @Param("date") Date date,
        @Param("hour") Integer hour);

    @Query("SELECT r FROM Room r WHERE r.name = :name")
    Room findByName(@Param("name") String name);
}

```

Figura 6.4 Repositorio pertenecientes a las salas.

A continuación voy a hablar de la paquetería de Event, tanto de las entidades como de los repositorios, servicios y controladores.

Diagrama para Event

Por motivos de diseño se decidió que todas las entidades tuvieran como identificador un campo "id" de tipo Long autonumérico, cada vez que hay una inserción se suma uno al identificador, es por esto que tenemos una clase abstracta "EntityIdentifiable" de la cual todos los objetos de nuestro modelo de datos extiende de ella.

```

@MappedSuperclass
public abstract class EntityIdentifiable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    private Long id;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
}

```

Figura 6.5 Todas las entidades extienden de EntityIdentifiable

Este es el caso de todas las entidades relacionadas directamente con Event.

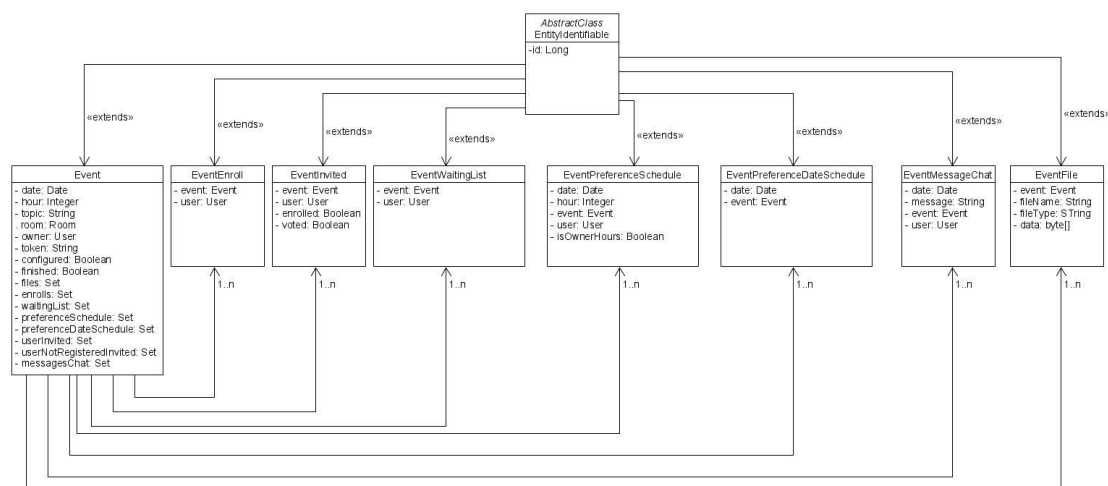


Figura 6.6 Diagrama de las entidades relacionadas directamente con Event.

Los eventos se guardan en la entidad "Event" y en ella están relacionados todos los objetos que ella maneja.



Figura 6.7 Entidad Event.

En Event tenemos tanto la fecha como la hora del evento, su topic (asunto), sala en la que se celebra, el usuario que la ha creado, su token para invitaciones a usuarios no registrados, si ya se ha terminado su configuración (que ya todos votaron y el Administrador/Jefe de proyecto la ha convocado en al hora y fecha convenida) o si ya se ha finalizado.

Además en ella están en una relación *OneToMany* todas las entidades que la referencian.

- enrolls: Listado de usuarios suscritos al evento.
- waitingList: Lista de espera de usuarios.
- preferenceSchedule: Las horas votadas de los usuarios invitados al evento, además de los del creador.
- preferenceDateSchedule: Las fechas votadas para el evento, incluidas las del creador.
- userInvited: Usuarios registrados invitados al evento.
- User NotRegisteredInvited: Usuarios no registrados invitados al evento.
- messagesChat: Los mensajes de chats escritos durante la celebración del evento.

Todas estas relaciones tienen una configuración a nivel de JPA de borrado en cascada, para que en el caso de que fuera un evento borrado, no haya ningún problema, ya que deben borrarse todos los registros en todas las tablas que hagan referencia a ese evento.

```

@OneToMany(mappedBy = "event", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
private Set<EventFile> files;

@OneToMany(mappedBy = "event", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
private Set<EventEnroll> enrolls;

@OneToMany(mappedBy = "event", fetch = FetchType.LAZY, cascade = CascadeType.REMOVE)
private Set<EventWaitingList> waitingList;
  
```

Figura 6.8 Algunas de las dependencias de Event, todas con un borrado de tipo cascada

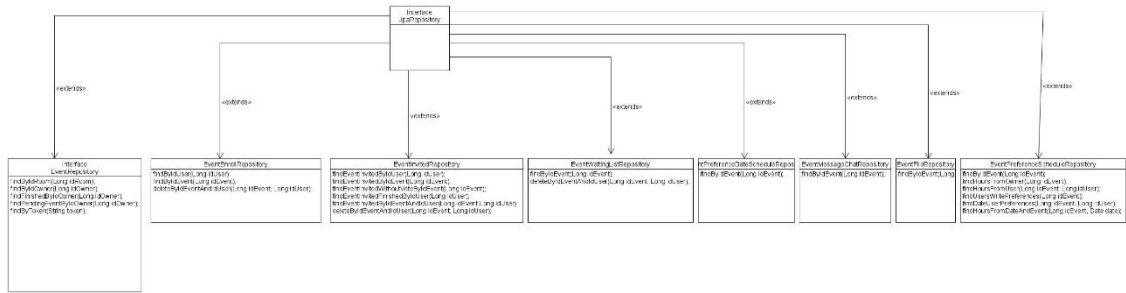


Figura 6.9 Todos los repositorios de la paquetería de Event.

Cada una de las entidades tiene su propio repositorio, el cual extiende de la interfaz *JpaRepository* de Spring, con esto ya tenemos soporte para los métodos más habituales para trabajar con la entidad, como encontrar un elemento por su identificador, guardar una entidad nueva o eliminarla.

Además de estos métodos fui añadiendo nuevos según necesidades.

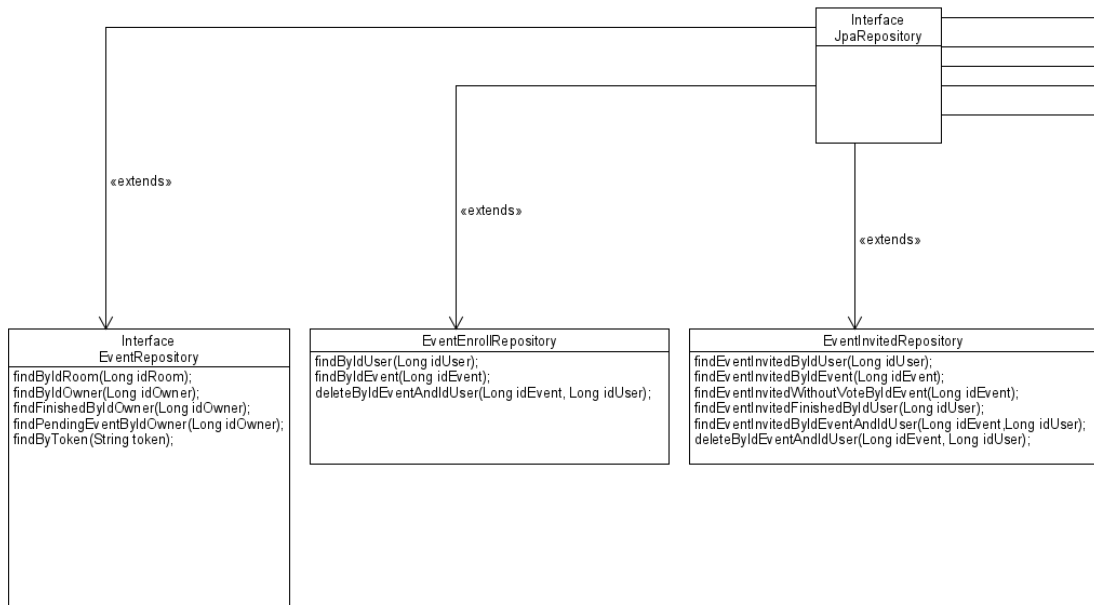


Figura 6.10 Repositorios de la paquetería de Event al detalle.

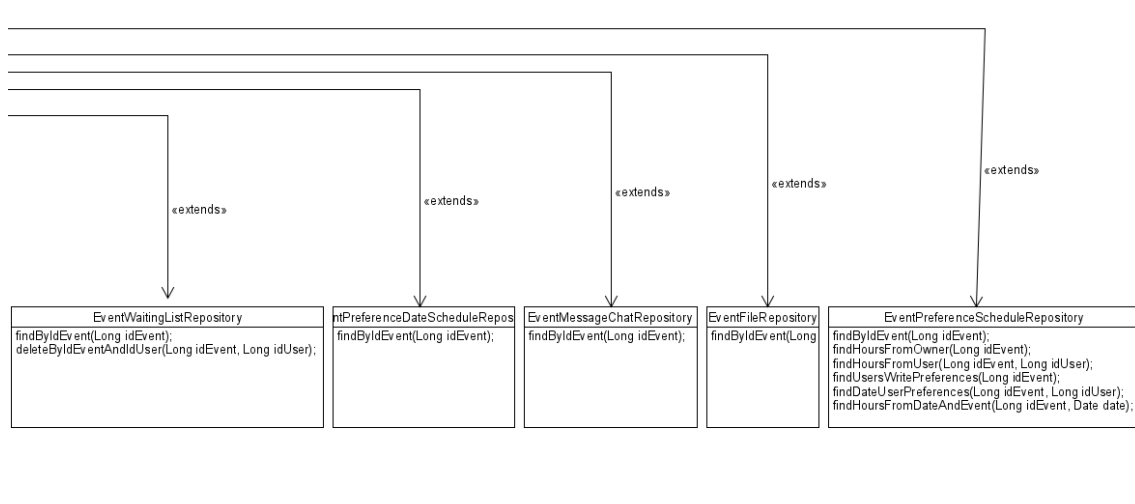


Figura 6.11 Repositorios de la paquetería de Event al detalle.

Los más habituales es encontrar los elementos a partir del id del evento, como por ejemplo buscar todos los invitados a un evento.

```
@Query("SELECT t FROM EventInvited t where t.event.id = :idEvent")
List<EventInvited> findEventInvitedByIdEvent(@Param("idEvent") Long idEvent);
```

Figura 6.12 Ejemplo de Query en el repositorio EventInvitedRepository.

Para trabajar la lógica de la aplicación, a cada repositorio le correspondió su propia cada de servicios.

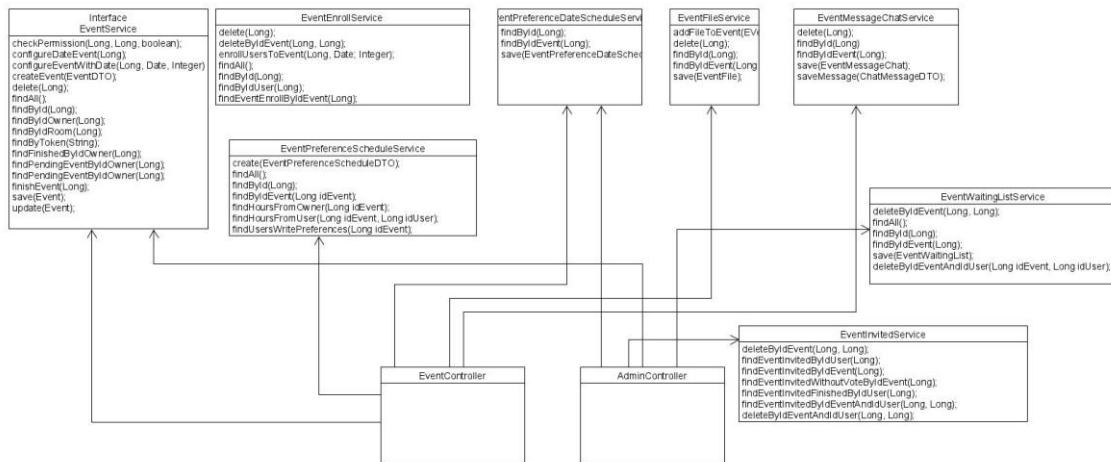


Figura 6.13 Servicios de la paquetería de Event, todos ellos relacionados con su repositorio y entidad.

Los servicios relacionados directamente con los eventos se llaman desde dos controladores, *EventController* y *AdminController*.

En *EventController* se trabaja con toda la parte de los servicios de la parte frontal que no tienen que ver con el rol de administrador y jefe de proyecto, mostrar el panel del usuario, hacer la votación por parte del usuario de las fechas y horas de los eventos, entrar al evento a su debida hora.

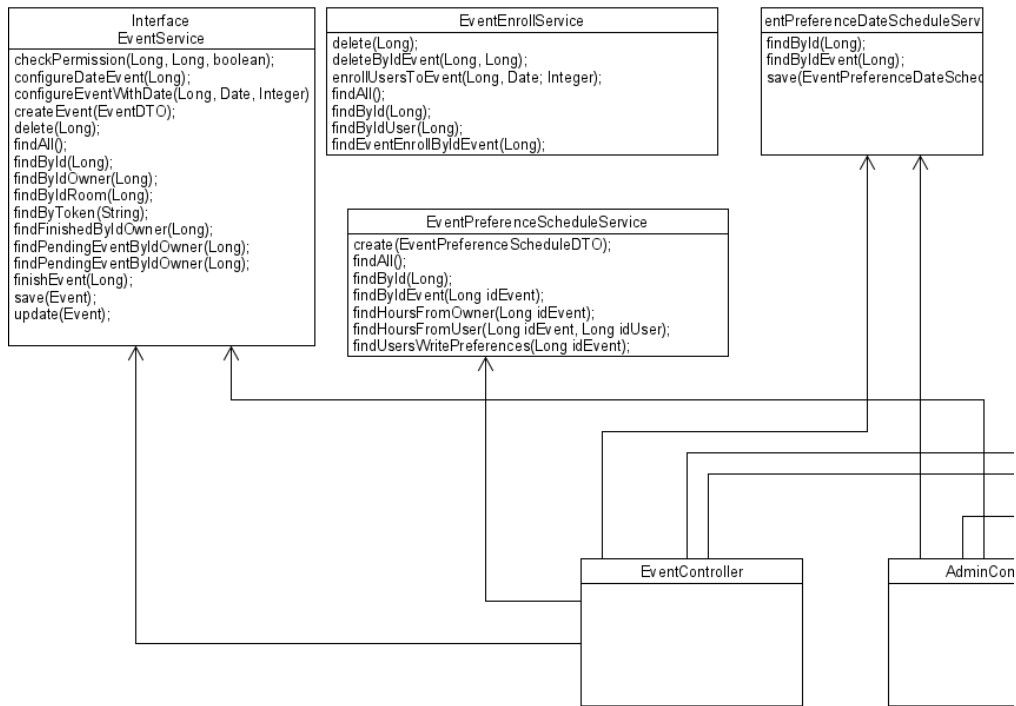


Figura 6.14 Servicios de la paquetería de Event al detalle.

Desde *AdminController* se trabaja con todas las tareas relacionadas con los administradores y jefes de proyecto, desde ese controlador llamaremos a los servicios para dar de alta un jefe de proyecto, un evento, aceptar las horas que han votado los usuarios, borrar mensajes del chat o ficheros del evento o finalizarlo.

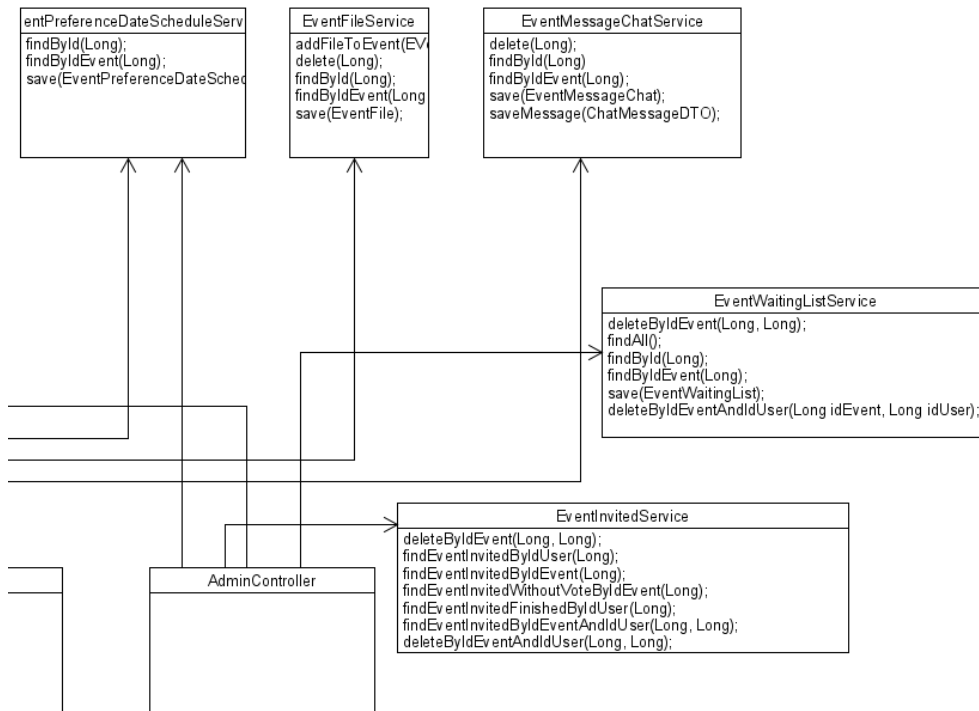


Figura 6.15 Servicios de la paquetería de Event al detalle.

Desde estos dos controladores llamaremos a los servicios encargados de la lógica de cada una de estas funciones.

- **EventService**

Gestión de todo lo directamente relacionado con la entidad *Event*, sus principales funciones son la recuperación de un evento, su creación, guardado, actualización y finalización. Permite buscar un evento por su identificador, por el creador, por la sala o por el token.

- **EventEnrollService**

Servicio para gestionar los inscritos a un evento, la entidad *EventEnroll*. Sus principales funciones son recuperar los inscritos a un evento, inscribir usuarios al evento o su borrado del evento.

- **EventFileService**

Desde aquí trabajamos con los ficheros adjuntos del evento, podremos encontrar un fichero en concreto o los ficheros adjuntos del evento, así como su creación y eliminación.

- **EventInviteService**

La gestión de los invitados a un evento se hace desde este servicio, con la entidad *EventInvite* recuperaremos los invitados a un evento, guardaremos la invitación de un usuario a un evento y además se actualiza la entidad *EventInvite* para saber cuándo un usuario invitado votó ya en la encuesta para la fecha y la hora y también saber que ese usuario ya fue inscrito al evento. Esta parte es para discernir de los usuarios invitados al evento, cuales finalmente fueron inscritos, quedando estos usuarios no inscritos únicamente como invitados para después poder consultar el evento una vez finalizado.

- **EventMessageChatService**

Servicio para cubrir la lógica de los mensajes de chat, a través de la entidad *EventMessageChat*, podremos guardar nuevos mensajes, recuperar los que se han escrito durante el evento o eliminar alguno.

- **EventPreferenceDateScheduleService**

Con la entidad *EventPreferenceDate* se realizará la votación de la fecha en la que se celebrará el evento, desde este servicio podremos recuperar las fechas que se han votado y guardar las fechas que votan los usuarios y el creador del evento.

- **EventPreferenceScheduleService**

En este servicio se trabajará con las horas en las que se puede celebrar el evento, aquí se guardará las horas que ha elegido el creador del evento y también las horas que deciden los usuarios invitados.

- EventWaitingListService

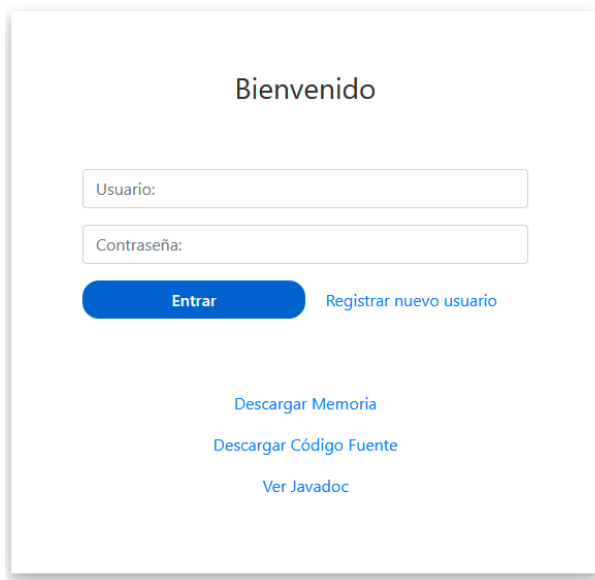
Un evento puede tener lista de espera, en este servicio se gestiona esta lista a través de la entidad *EventWaitingList*. Se podrá recuperar la lista de espera de un evento, guardar a un usuario en esta lista o hacer gestiones como traspasar a un usuario invitado a esta lista de espera.

Además de estas clases tenemos más para el resto de funciones, separadas por paquetes y con esta misma filosofía, si desea conocer más en profundidad todas estas en la pantalla de entrada del usuario además dispone del Javadoc de la aplicación, donde podrá conocer más en detalle todas las clases que participan y su descripción.

7. Guía de Usuario

7.1 Manual de Usuario

7.1.1 Registro de usuarios



Bienvenido

Usuario:

Contraseña:

[Entrar](#) [Registrar nuevo usuario](#)

[Descargar Memoria](#)

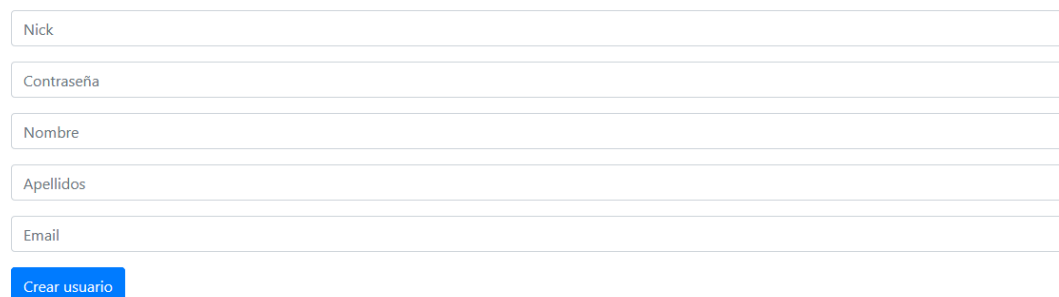
[Descargar Código Fuente](#)

[Ver Javadoc](#)

Figura 7.1.1.1 Pantalla de login de usuario

En la pantalla de *login* se debe pulsar en Registrar nuevo usuario.

Alta de usuario



Nick

Contraseña

Nombre

Apellidos

Email

[Crear usuario](#)

© 2020 Registro Horario. UNED.

Figura 7.1.1.2 Pantalla de nuevo usuario.

Se deben rellenar todos los datos del formulario, el Nick debe ser único (se mostrará un mensaje de error en caso de que ya exista) y el Email debe respetar el formato “[nombre@dominio.com](#)”, en caso contrario se mostrará un error.

Alta de usuario

El formato del email no es válido

El apellido es obligatorio

El nombre es obligatorio

La contraseña es obligatoria

El usuario es obligatorio

Nick

Contraseña

Nombre

Apellidos

prueba

Crear usuario

© 2020 Registro Horario. UNED.

Figura 7.1.1.3 Pantalla de nuevo usuario.

7.1.2 Entrar en la aplicación

Bienvenido

Usuario:

Contraseña:

Entrar

Registrar nuevo usuario

Descargar Memoria

Descargar Código Fuente

Ver Javadoc

Figura 7.1.2.1 Pantalla de entrada de usuario.

Con el usuario y contraseña que ha escogido o con el que le ha dado de alta el administrador, debe rellenar los campos de Usuario y Contraseña y pulsar en el botón de *login*, donde entrará en su panel de usuario.

7.1.3 Panel de usuario

Bandeja de próximos Eventos			
Día	Hora	Sala	Organizador
10/05/2020	10:00	Futurama	Óscar Montes

© 2020 Registro Horario. UNED.

Figura 7.1.3.1 Panel de usuario de alguien que tiene el rol de Usuario.

En este panel la primera pestaña que verá será la bandeja de los próximos eventos a los que debe acudir, estos eventos ya están con fecha y hora y solo queda entrar en él pulsando sobre la fila del evento, con un botón al final en caso de que quiera darse de baja de ese evento.

Si pulsa en el evento y todavía no se ha iniciado, verá una pantalla de aviso.

El evento no comenzará hasta cinco minutos antes de las 10:00 del 10/05/2020 y finalizará a las 11:00

© 2020 Registro Horario. UNED.

Figura 7.1.3.2 Mensaje de aviso para un evento que aún no ha comenzado.

Eventos invitado			
Topic	Sala	Organizador	
Evento de prueba 4	Stargate	Óscar Montes	

© 2020 Registro Horario. UNED.

Figura 7.1.3.3 Pestaña de eventos invitado en el panel de usuario.

La segunda pestaña son los eventos a los que está invitado, en caso de que tenga algún evento nuevo le aparecerá un número, ya que tiene pendiente realizar la votación sobre qué fecha es la mejor para que acuda a ese evento. Deberá pulsar en él para decidir qué día y qué hora es la mejor para usted.

Eventos Finalizados

Día	Hora	Topic	Sala	Organizador
28/05/2020	10:00	Evento de prueba 6	Stargate	Óscar Montes

© 2020 Registro Horario. UNED.

Figura 7.1.3.4 Pestaña de eventos finalizados en el panel de usuario.

La tercera pestaña son los eventos que ya han finalizado, aquí verá los eventos en los que ha estado suscrito o que le han invitado y ya finalizaron, pulsando sobre podrá ver la conversación que hubo en ese evento y qué ficheros se adjuntaron.



Settings

czamora

Clara

Zamora

czamora@mail.com

Actualizar

© 2020 Registro Horario. UNED.

Figura 7.1.2.5 Pestaña de actualización de datos del usuario.

La cuarta pestaña es para modificar sus datos personales, únicamente no podrá modificar el usuario. Cuando quiera modificar algún dato puede cambiarlo y pulsar en el botón actualizar.

7.1.4 Entrar al evento



Día	Hora	Sala	Organizador
10/05/2020	10:00	Futurama	Óscar Montes

© 2020 Registro Horario. UNED.

Figura 7.1.4.1 Pestaña de Bandeja de próximos eventos

Pulsando en el evento al que quiere acudir, entrará al evento si ya se está celebrando. El entorno arranca cinco minutos antes de la hora del evento.

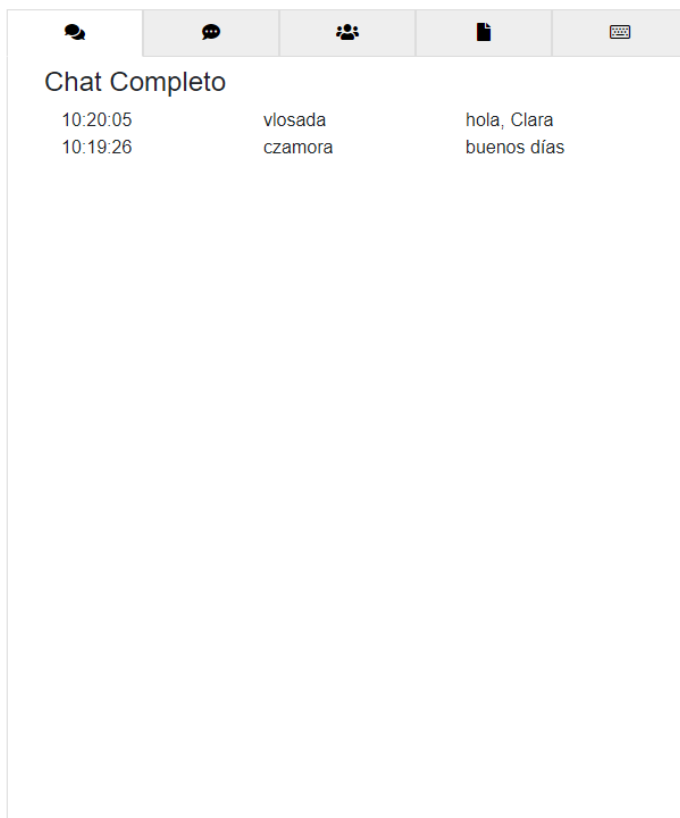


Figura 7.1.4.2 Pestaña del chat completo en la ventana del evento.

La primera pestaña es la del chat completo, podrá ver la conversación que se ha producido en su ausencia.

En la segunda pestaña podrá participar en esta conversación.

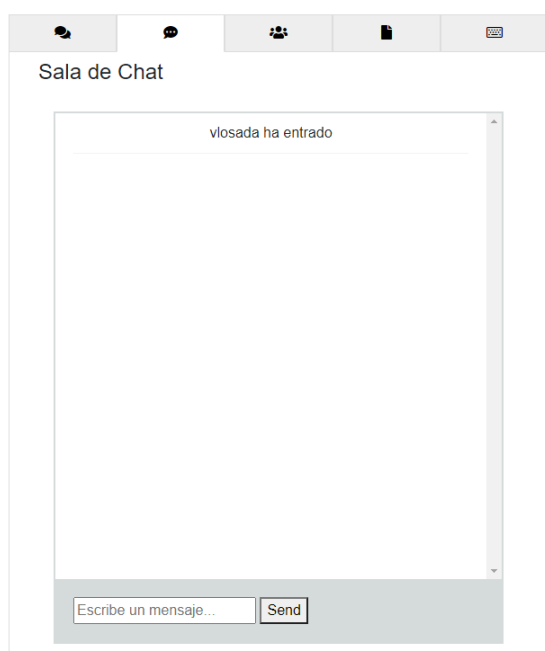


Figura 7.1.4.3 Pestaña del chat en la ventana del evento.

Escribiendo el mensaje en la parte inferior y pulsando en “Send” o en el “intro” del teclado, este mensaje será enviado al *chat* y podrá verlo junto a los demás en la primera pestaña.

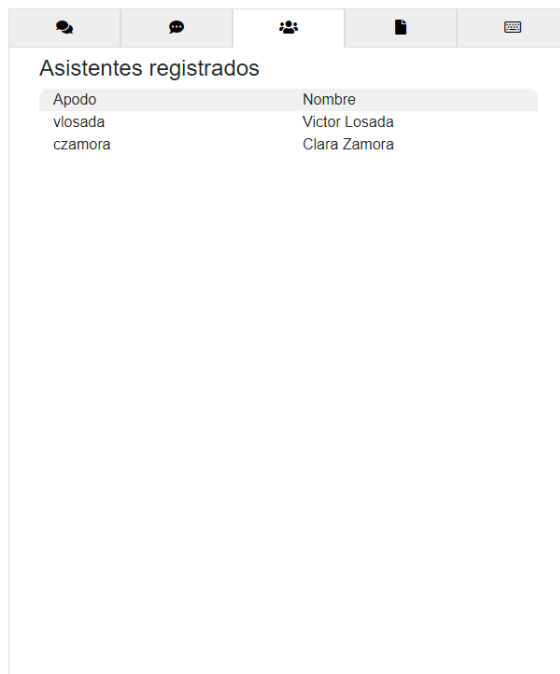


Figura 7.1.4.4 Pestaña de asistentes al evento.

En la tercera pestaña puede ver los usuarios que están suscritos al evento.

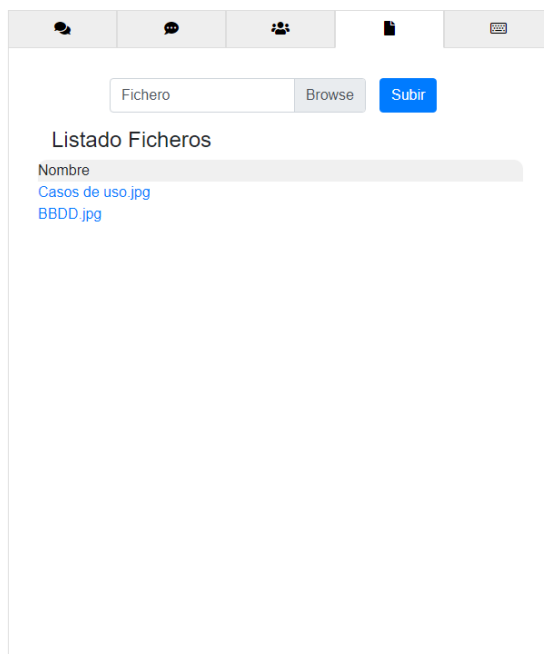


Figura 7.1.4.5 Pestaña de subida y muestra de ficheros del evento.

En la cuarta pestaña podrá subir un fichero, pulsando en “Browse” y eligiendo uno de su ordenador y posteriormente pulsando en Subir.

Verá en la lista de abajo este fichero y el resto de ficheros subidos.

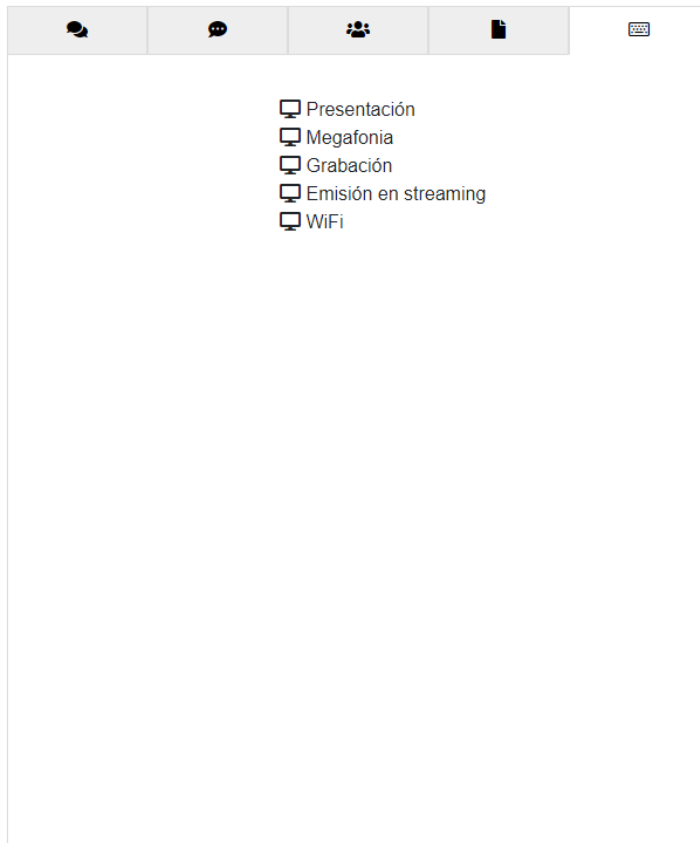
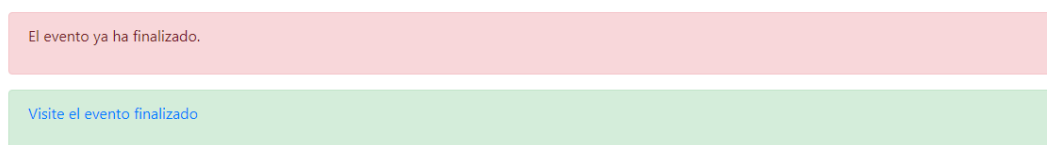


Figura 7.1.4.6 Pestaña de recursos técnicos de la sala del evento.

La última pestaña es para conocer de qué recursos técnicos dispone en la sala.

Cuando el organizador del evento lo finalice, si interactúa con el evento recibirá el siguiente mensaje.



© 2020 Registro Horario. UNED.

Figura 7.1.4.7 Mensaje avisando de que el evento ha finalizado.

Si desea volver a su panel de usuario, puede pulsar en su nombre en la cabecera.

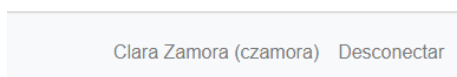


Figura 7.1.4.8 Pulsando en el nombre de la cabecera volvemos al panel del usuario.

7.1.5 Eventos invitado

En la segunda pestaña del panel de usuario recibirá los avisos de los eventos a los que le invitan, además verá un número indicándole que tiene eventos a los que le han invitado y no ha votado aún.

	2		
Eventos invitado			
Topic	Sala	Organizador	
Evento de prueba 4	Stargate	Óscar Montes	
Nuevo evento de prueba	Star Trek	Óscar Montes	

© 2020 Registro Horario. UNED.

Figura 7.1.5.1 Pestaña de eventos a los que le han invitado en el panel de usuario.

Pulsando sobre el evento, podrá ir a la ventana para realizar la votación del día y la fecha que más le convenga.

Seleccionar horas de disponibilidad

Fecha

Horas

10	11	13
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

© 2020 Registro Horario. UNED.

Figura 7.1.5.2 Ventana de votación de fecha y hora.

En el combo de fecha deberá seleccionar el día que mejor le convenga y en las horas elegir qué horas son mejores para usted, debe escoger alguna hora obligatoriamente o saldrá un mensaje de error.

Seleccionar horas de disponibilidad

Debe escoger alguna hora

Fecha

Horas

10	11	12
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

© 2020 Registro Horario. UNED.

Figura 7.1.5.3 Aviso de error en la ventana de votación de fecha y hora.


Una vez marque alguna hora y seleccione la fecha, pulse en el botón “Elegir horas” y se guardarán sus preferencias.


Se han guardado correctamente sus preferencias.


			
Bandeja de próximos Eventos			
Día	Hora	Sala	Organizador
10/05/2020	10:00	Futurama	Óscar Montes
			
© 2020 Registro Horario. UNED.			


Figura 7.1.5.4 Mensaje de grabación correctamente en el panel de usuario.

7.1.6 Visualizar eventos finalizados









Eventos Finalizados

Día	Hora	Topic	Sala	Organizador
28/05/2020	10:00	Evento de prueba 6	Stargate	Óscar Montes

© 2020 Registro Horario. UNED.

Figura 7.1.6.1 Pestaña de eventos finalizados en el panel de usuario.

En la tercera pestaña verá los eventos finalizados en los que ha participado o ha sido invitado. Pulsando sobre el evento irá a una ventana donde podrá visualizar información sobre el evento.



Figura 7.1.6.2 Pantalla de los asistentes en el detalle del evento.

En la primera pestaña verá la fecha, hora, el topic del evento, en qué sala estuvo o quienes asistieron.

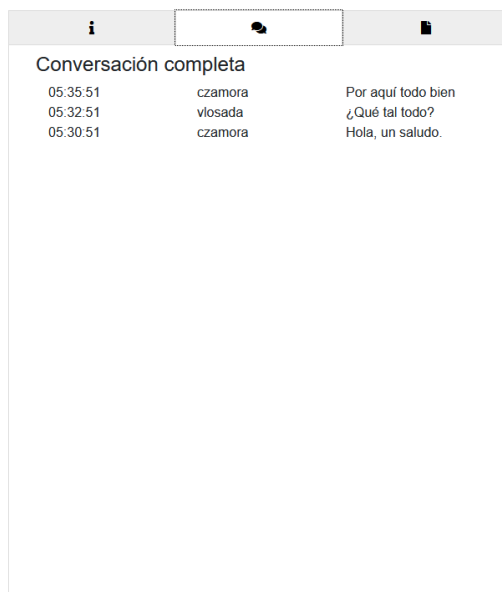


Figura 7.1.6.3 Pestaña con la conversación del chat del evento finalizado.

En la segunda pestaña verá el chat completo del evento.

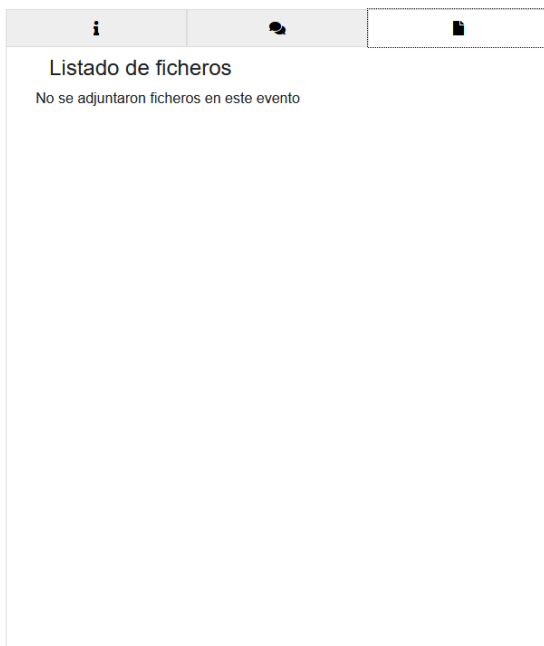


Figura 7.1.6.4 Pestaña con los ficheros compartidos en el evento finalizado.

En caso de que se hayan adjuntado ficheros, podrá verlos en la tercera pestaña.

7.1.7 Salir de la aplicación

En caso de que quiera salir cerrando su sesión, deberá pulsar el botón “Desconectar” que está situado en la cabecera a la derecha, junto a su nombre.

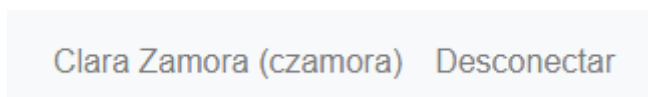


Figura 7.1.7.1 Botón desconectar

Esto le devolverá a la pantalla de login.

7.2 Manual de Jefe de Proyecto

En su panel de usuario, el jefe de proyecto tiene más apartados que un usuario registrado normal.



Figura 7.2.1 Panel de usuario de un jefe de proyecto.

En la tercera pestaña encontramos “Mis Eventos”.

1

Mis Eventos

Día	Hora	Topic	Sala
10/05/2020	10:00	Evento de prueba 1	Futurama
15/05/2020	11:00	Evento de prueba 2	Los Simpsons
16/05/2020	12:00	Evento de prueba 3	Los Simpsons
20/05/2020	9:00	Evento de prueba 5	Stargate

© 2020 Registro Horario UNED.

© 2020 Registro Horario. UNED.

Figura 7.2.2 Pestaña Mis Eventos.

Aquí tenemos los eventos que hemos creado nosotros y que no han finalizado, su funcionalidad es parecida a la de la “Bandeja de próximos Eventos”, pero aquí solamente estarán los que hemos creado nosotros. Cinco minutos antes de la hora podrá entrar al evento y participar en él como un usuario más, si el evento no se ha iniciado recibirá un aviso.

El evento no comenzará hasta cinco minutos antes de las 10:00 del 10/05/2020 y finalizará a las 11:00

© 2020 Registro Horario. UNED.

Figura 7.2.3 Aviso de que el evento aún no ha empezado.

La cuarta pestaña son los eventos pendientes.

Eventos pendientes

Topic	Sala	Enlace
Evento de prueba 4	Stargate	Invitación <div></div>

© 2020 Registro Horario. UNED.

Figura 7.2.4 Pestaña de eventos pendientes.

En ella estarán los eventos que hemos creado y que todavía están en fase de votación, si copia el enlace de la invitación, puede darle este enlace a usuarios no registrados para que se apunten al evento. Pulsando en la flecha podrá entrar a ver cómo va la votación.

En la sexta pestaña tenemos nuestros eventos finalizados.

Mis Eventos Finalizados						
Día	Hora	Topic	Sala			
28/05/2020	10:00	Evento de prueba 6	Stargate			

© 2020 Registro Horario. UNED.

Figura 7.2.5 Pestaña para mis eventos finalizados.

Aquí tendremos eventos que hemos creado y que ya finalizaron, pulsando sobre el evento entraremos al detalle de este para ver qué conversación se mantuvo, quienes estaban invitados o qué archivos se compartieron.

Fecha 28/05/2020 Hora 10:00

Evento "Evento de prueba 6" situado en la sala **Stargate**

Asistentes registrados

Usuario	Nombre
vlosada	Victor Losada
czamora	Clara Zamora
brosell	Bilal Rosell

Asistentes no registrados

Nombre
Sergi
Gonzalo
Faemino

Figura 7.2.6 Ventana de detalle del evento.

Ver sección del manual de usuario para conocer al detalle la pantalla de detalle de Eventos.

7.2.1 Nuevo evento

Para crear un nuevo evento, debe pulsar en “Nuevo Evento” en la cabecera del programa.

Portal de reuniones [Nuevo Evento](#)

Figura 7.2.1.1 Enlace para Nuevo Evento en la cabecera de la aplicación.

Llegando al formulario para crear eventos.

Alta de Eventos

+ 10/05/2020

Topic

Usuarios disponibles

Mostrar 10 registros Buscar:

	Nombre	Apellidos	Email
<input type="checkbox"/>	Óscar	Montes	omontes@mail.com
<input type="checkbox"/>	Bilal	Rosell	brosell@mail.com
<input type="checkbox"/>	Victor	Losada	vlosada@mail.com
<input type="checkbox"/>	Clara	Zamora	czamora@mail.com
<input type="checkbox"/>	Dolores	Blasco	dblasco@mail.com
<input type="checkbox"/>	Gregoria	Jara	gjar@mail.com
<input type="checkbox"/>	Pedro Miguel	Candela	pmcandela@mail.com
<input type="checkbox"/>	Unax	Espinosa	uespinosa@mail.com
<input type="checkbox"/>	Fernando	Torres	ftorres@mail.com
<input type="checkbox"/>	Ramón	García	rgarcia@mail.com

Mostrando registros del 1 al 10 de un total de 23 registros

Anterior 1 2 3 Siguiente

Usuarios para lista de espera

Mostrar 10 registros Buscar:

	Nombre	Apellidos	Email
<input type="checkbox"/>	Óscar	Montes	omontes@mail.com
<input type="checkbox"/>	Bilal	Rosell	brosell@mail.com

Figura 7.2.1.2 Formulario de creación de un nuevo evento.

Lo primero que tiene es la posibilidad de añadir cuantas posibles fechas quiera, pulsando en el botón “+” que está junto a la fecha.

+ 10/05/2020

Figura 7.2.1.3 Detalle de la fecha en la pantalla de creación de nuevo evento.

Cada vez que lo pulse, aparecerá una fecha nueva.

Alta de Eventos

+ 10/05/2020 dd/mm/aaaa - dd/mm/aaaa -

Figura 7.2.1.4 Podemos añadir cuantas fechas queramos al evento.

Pulsando en el botón “-” que está a la derecha de la nueva fecha puede eliminarla. Deberá completarla eligiéndola del calendario.

Alta de Eventos

10/05/2020 × dd/mm/aaaa dd/mm/aaaa

Topic

Usuarios disponibles

Mostrar 10 registros

Nombre

<input type="checkbox"/>	Óscar
<input type="checkbox"/>	Bilal

Rosell

Figura 7.2.1.5 Tenemos un calendario para la selección de la fecha.

Pulse sobre el día deseado para elegirlo.

Después tiene una tabla con todos los usuarios del sistema.

Usuarios disponibles

Mostrar 10 registros

Buscar: clara

	Nombre	Apellidos	Email
<input type="checkbox"/>	Clara	Zamora	czamora@mail.com

Mostrando registros del 1 al 1 de un total de 1 registros (filtrado de un total de 23 registros)

Anterior 1 Siguiente

Usuarios para lista de espera

Figura 7.2.1.6 Ejemplo de búsqueda de un usuario.

Puede utilizar el buscador para encontrar al usuario que quiera y para seleccionarlo pulse en el *checkbox*. Puede seleccionar cuantos usuarios quiera.

Después tiene una tabla similar con los usuarios que quiera introducir en la lista de espera.

Usuarios para lista de espera

Mostrar 10 registros

Buscar:

	Nombre	Apellidos	Email
<input type="checkbox"/>	Óscar	Montes	omontes@mail.com
<input type="checkbox"/>	Bilal	Rosell	brosell@mail.com
<input type="checkbox"/>	Víctor	Losada	vlosada@mail.com
<input type="checkbox"/>	Clara	Zamora	czamora@mail.com
<input type="checkbox"/>	Dolores	Blasco	dblasco@mail.com
<input type="checkbox"/>	Gregoria	Jara	gjara@mail.com
<input type="checkbox"/>	Pedro Miguel	Candela	pmcandela@mail.com
<input type="checkbox"/>	Unax	Espinosa	uespinosa@mail.com
<input type="checkbox"/>	Fernando	Torres	ftorres@mail.com
<input type="checkbox"/>	Ramón	García	rgarcia@mail.com

Mostrando registros del 1 al 10 de un total de 23 registros

Anterior 1 2 3 Siguiente

Usuarios disponibles

Figura 7.2.1.7 Tabla de usuarios para la lista de espera.

Estos usuarios no recibirán ningún aviso mientras están en esta lista, solo cuando pasen a la lista de invitados recibirán un aviso para realizar la votación de fechas y horas.

Delores Blasco dblasco@gmail.com

Futurama (Aforo: 10) Disponibilidad: [8:00, 9:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00]

Los Simpsons (Aforo: 5) Disponibilidad: [10:00, 11:00, 12:00, 13:00, 16:00]

Stargate (Aforo: 6) Disponibilidad: [9:00, 10:00, 16:00, 17:00]

Expediente X (Aforo: 8) Disponibilidad: [16:00, 17:00, 18:00, 19:00]

Seinfeld (Aforo: 10) Disponibilidad: [10:00, 11:00, 12:00, 13:00]

Doctor Who (Aforo: 7) Disponibilidad: [10:00, 11:00, 12:00]

Lost (Aforo: 7) Disponibilidad: [8:00, 9:00, 10:00, 11:00, 12:00, 13:00, 14:00]

Fringe (Aforo: 9) Disponibilidad: [15:00, 16:00, 17:00, 18:00, 19:00, 20:00]

Star Trek (Aforo: 6) Disponibilidad: [10:00, 11:00, 12:00, 13:00, 14:00]

Westworld (Aforo: 8) Disponibilidad: [12:00]

Futurama (Aforo: 10) Disponibilidad: [8:00, 9:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00]

Figura 7.2.1.8 Combo de salas.

Después tenemos un combo de salas donde además vemos el aforo y su disponibilidad horaria, debe seleccionar salas con aforo igual o superior a los usuarios invitados, en caso de superar el aforo recibirá un aviso de error a la hora de crear el evento y no le dejará crearla.

Horas

8	9	10	11	12	13	14	15	16	17	18	19	20
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Invitar

Figura 7.2.1.9 Selección de horas.

Por último tiene las horas para invitar, desde las 8 de la mañana a las 20 horas de la tarde, puede escoger las que quiera, pero debe tener en cuenta la disponibilidad de la sala para esas horas, en caso de no estar disponible recibirá un aviso cuando cree el evento y deberá escoger otra sala o cambiar las horas.

Cuando finalmente cree el evento pulsando en el botón “Invitar”, todos los usuarios invitados recibirán un aviso en su bandeja de “Eventos invitados” y podrán empezar a votar. Usted tendrá un nuevo registro en la pestaña de “Eventos pendientes” y podrá seguir la votación desde ahí.

7.2.2 Configuración del evento

Configuración

Evento "Evento de prueba 4" situado en la sala **Stargate**

Aforo total: 6 aforo restante: 4

Stargate Disponibilidad: [9:00, 10:00, 16:00, 17:00] ▼ Cambiar sala

Voto Usuarios

¡Todavía no ha votado ningún usuario!

Usuarios que faltan por votar

Nombre	Email	
Bilal Rosell	brosell@mail.com	↓
Clara Zamora	czamora@mail.com	↓

Día del evento

03 / 06 / 2020 ⊗

Hora del evento

Hora

Asignar Hora

© 2020 Registro Horario. UNED.

Figura 7.2.2.1 Configuración del evento con las votaciones.

A través de la cuarta pestaña del manual de usuario tenemos el listado de “Eventos Pendientes”, pulsando en la flecha del evento deseado llegamos a la configuración, ahí podremos ver qué usuarios han votado y quienes faltan por votar, según lleguen las votaciones veremos como va apareciendo el Día del evento y la hora más votados. Cuando hayan terminado todos de votar se habilitará el botón “Asignar Hora” y ya podremos pulsar en él, quedando ya configurado el evento con esa fecha y esa hora, todos los usuarios invitados recibirán la invitación en su bandeja de entrada y usted en la tercera pestaña “Mis Eventos” del panel de usuario.

En caso de que desee cambiar de sala, puede escoger otra del combo y pulsar en el botón “Cambiar Sala”, en caso de que el aforo de la sala sea inferior al de los usuarios invitados no podrá escogerla y saldrá un mensaje de aviso.

7.2.3 Finalizar evento.

Dentro de los eventos que ha organizado, durante la celebración puede finalizar el evento cuando lo desee.

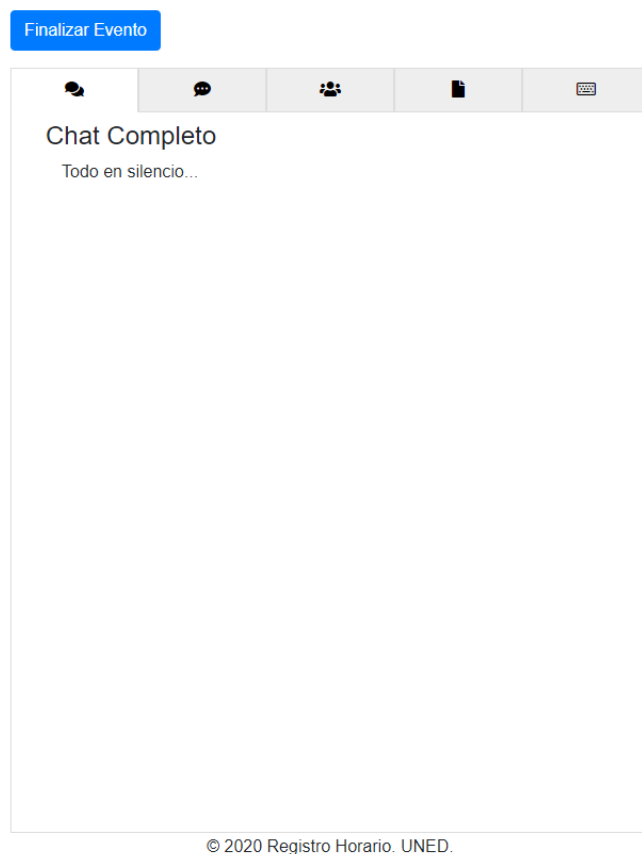


Figura 7.2.3.1 Ventana del evento donde podemos ver el botón de finalizar.

Pulsando en el botón “Finalizar Evento” saldrá del evento y el resto de usuarios ya no podrán escribir más en el chat ni subir ficheros. Desde ese momento podrá ver el evento en la pestaña “Mis Eventos Finalizados”.

7.3 Manual de Administrador

El panel de usuario del administrador es igual al del jefe de proyecto, los apartados nuevos para un administrador están en la cabecera.

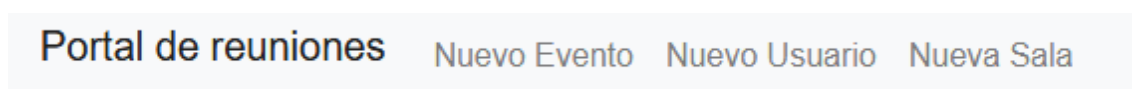
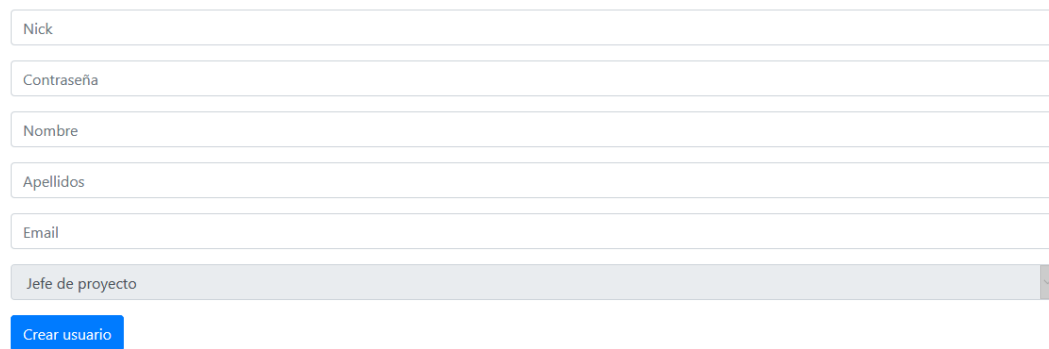


Figura 7.3.1 Menú en la cabecera del administrador.

7.3.1 Nuevo Usuario

Pulsando en la cabecera el enlace “Nuevo Usuario” el administrador puede crear nuevos jefes de proyecto.

Alta de usuario



Formulario de creación de usuario con los siguientes campos:

- Nick
- Contraseña
- Nombre
- Apellidos
- Email
- Jefe de proyecto (menú desplegable)
- Botón: Crear usuario

© 2020 Registro Horario. UNED.

Figura 7.3.1.1 Pantalla de creación de jefe de proyecto.

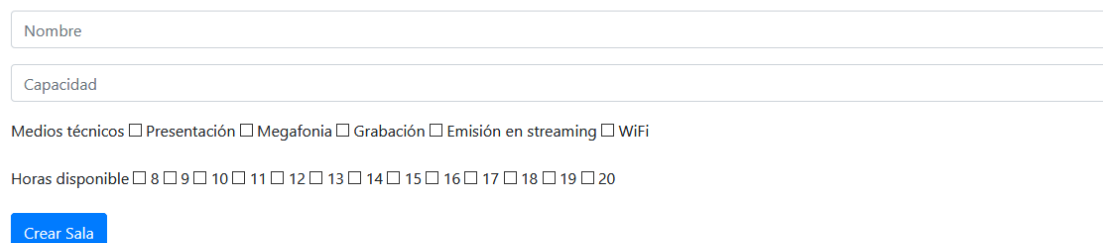
Como puede ver tiene un combo de roles de usuario donde no se puede cambiar el valor de Jefe de Proyecto.

Se deben rellenar todos los datos del formulario, el Nick debe ser único (se mostrará un mensaje de error en caso de que ya exista) y el Email debe respetar el formato "[nombre@dominio.com](#)", en caso contrario se mostrará un error.

7.3.2 Nueva Sala

El administrador puede crear nuevas salas pulsando en el botón de la cabecera "Nueva Sala".

Alta de salas



Formulario de creación de sala con los siguientes campos:

- Nombre
- Capacidad
- Medios técnicos: ☐ Presentación ☐ Megafonia ☐ Grabación ☐ Emisión en streaming ☐ WiFi
- Horas disponible: ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐ 13 ☐ 14 ☐ 15 ☐ 16 ☐ 17 ☐ 18 ☐ 19 ☐ 20
- Botón: Crear Sala

© 2020 Registro Horario. UNED.

Figura 7.3.2.1 Pantalla de creación de salas

Debe rellenar todos los campos, en caso contrario recibirá avisos de error.

Alta de salas

Debe rellenar el nombre

Debe marcar alguna hora disponible

Debe rellenar la capacidad

Debe marcar al menos un medio técnico

Nombre

Capacidad

Medios técnicos ☐ Presentación ☐ Megafonía ☐ Grabación ☐ Emisión en streaming ☐ WiFi

Horas disponible ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐ 13 ☐ 14 ☐ 15 ☐ 16 ☐ 17 ☐ 18 ☐ 19 ☐ 20

Crear Sala

© 2020 Registro Horario. UNED.

Figura 7.3.2.2 Pantalla con mensajes de error en la pantalla de creación de la sala.

Una vez completados todos los campos y pulsado en Crear Sala, tendrá una nueva sala disponible para los Eventos.

7.4 Manual de Usuario no registrado

El organizador del evento puede enviarle un enlace con un token a un usuario no registrado para poder asistir a un evento, con ese enlace el invitado llegará a la siguiente pantalla.

Nombre

Entrar

© 2020 Registro Horario. UNED.

Figura 7.4.1 Formulario para inscribirse en un evento como usuario no registrado.

Deberá introducir su nombre y pulsar en el botón “Entrar”, llegando a la pantalla que le confirma que está inscrito al evento.

Se ha suscrito correctamente al evento.

Acceda al siguiente enlace o guárdelo para cuando comience el evento

[Enlace al evento](#)

© 2020 Registro Horario. UNED.

Figura 7.4.2 Aviso de que ya está suscrito al evento

El invitado debe pulsar en el enlace para ir al evento, siempre que este ya haya comenzado, si no ha comenzado puede recibir dos avisos, en caso de que todavía se estén votando fechas.

El evento está todavía en proceso de votación.

© 2020 Registro Horario. UNED.

Figura 7.4.3 Aviso de que en el evento todavía se está votando por día y fecha.

O en caso de que todavía no haya llegado la hora del evento, recibirá el siguiente aviso.

El evento no comenzará hasta cinco minutos antes de las 16:00 del 06/06/2020 y finalizará a las 17:00

© 2020 Registro Horario. UNED.

Figura 7.4.4 Aviso de que todavía no ha comenzado el evento.

Dentro del evento, el usuario no registrado podrá participar en el chat y subir ficheros al igual que el resto de usuarios registrados.

8. Conclusiones

Estamos ante una práctica muy idónea para el mundo actual de desarrollo web, el poder utilizar *frameworks* que ahora mismo te puedes encontrar en cualquier empresa de software, como *Spring Boot*, me ha dado una libertad para poder desarrollar una aplicación de una manera muy cercana a como lo haría en un trabajo real.

He quedado muy satisfecho con el resultado final y el enfoque con el que la he afrontado, la falta de tiempo me ha hecho no ir por caminos que me apetecían mucho, como por ejemplo tener una parte frontal aún mucho más atractiva y aprovechar las bondades de seguridad que nos ofrece Spring Security, aunque esto segundo lo intenté, finalmente no quise invertir más tiempo en esto y lo que hice fue hacer una seguridad algo más rústica, guardando en base de datos al usuario con su contraseña, validando su existencia y guardando este en memoria, en mi opinión es la parte que más descontento me ha dejado, aunque a nivel funcional cumple con su cometido.

La práctica no me ha parecido para nada pequeña y creo que no está bien estimada en las 48 horas que tenemos en el plan de trabajo, no soy capaz de calcular las horas que he invertido, pero me atrevo a decir que he multiplicado por dos esas horas.

Pensando en qué partes mejorar, además de la seguridad y que está claro que había que respetar el patrón MVC, hoy en día se está caminando hacia un desarrollo más de servicios Rest y desacoplar totalmente la parte del back con la vista, de esta forma todos los controladores debería exponer estos servicios en vez de trabajar con el modelo directamente cony en su lugar tener una vista desarrollada con soluciones como React o el framework Angular.

9. Anexo

9.1 Usuarios

Hay más de 20 usuarios disponibles en la aplicación, los más importantes son:

- Administrador: Usuario: **fsimon** Contraseña: **1234**
- Jefe de proyecto: Usuario: **omontes** Contraseña: **1234**
- Usuario registrado: Usuario: **czamora** Contraseña: **1234**
- Usuario registrado: Usuario: **vlosada** Contraseña: **1234**
- Usuario registrado: Usuario: **dblasco** Contraseña: **1234**

Si se desea utilizar más usuarios, en la pantalla de nuevo evento sale el listado de todos, ahí podrá ver el usuario y todos tienen como contraseña: 1234.

9.2 Tecnologías, librerías y frameworks

- Java 11: <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>
- Spring Boot 2.2.6: <https://spring.io/projects/spring-boot>
- H2 Database 1.4.200: <https://www.h2database.com>
- Thymeleaf: 3.0.11: <https://www.thymeleaf.org>
- JQuery 3.5.1: <https://jquery.com>
- Bootstrap 4.5.0: <https://getbootstrap.com>
- DataTables 1.10.21: <https://datatables.net>
- SockJS 1.4.0: <http://sockjs.org>
- StompJS 5: <https://stomp-js.github.io>