Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior

Trabajo Fin de Grado

ESTUDIO Y DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB PARA LA GESTIÓN DE ABONOS DE CARRERA OFICIAL

Alumno: David López Cano

Tutor: José Ramón Balsas Almagro
Dpto: Informática

Junio, 2016

“A la Gloriosissima Virgen María…
con el título de los Dolores
que se venera en su Capilla de Sor. Sn. Juan
de esta Ciudad de Jaén”

AGRADECIMIENTOS

A Nerea, por ser mi pilar fundamental, por escucharme, por apoyarme, por transmitirme tu fortaleza, por volcarte tanto como yo para que este proyecto sea una realidad. Gracias cariño.

A mis padres, por enseñármelo todo en esta vida, por vuestros consejos, por formarme como persona, y darme la oportunidad de formarme como profesional. Muchísimas gracias.

A María, por tus ánimos, por tus visitas constantes mientras trabajaba, por tu cariño. Ahora te toca a ti.

A mis abuelos y mi tía, por ser mis segundos padres, por su estima e interés siempre. Gracias.

A José Ramón, por tener siempre las puertas de su despacho para consultar tantas y tantas dudas que me han surgido en la realización de este proyecto. Gracias por tú paciencia y dedicación en la dirección de este TFG.

A Antonio José y José Manuel, por vuestra amistad sincera, por esas conversaciones jartibles en mis momentos de descanso. Gracias.

A Villi, por regalarme siempre tu sonrisa. Gracias.

A la Agrupación de Cofradías de Jaén, darme la idea sobre la que se fundamenta este trabajo, por confiar en mí a sabiendas de mi nula experiencia.

TABLA DE CONTENIDO

RELACIÓN DE TABLAS

TABLA DE ILUSTRACIONES

1 INTRODUCCIÓN

1.1 Propósito

El propósito de este Trabajo Fin de Grado es analizar, diseñar y desarrollar una aplicación web que gestione los abonos de una carrera oficial de Semana Santa.

Esta aplicación web nos proporcionará una gestión integral de los abonados a la carrera oficial. Nos permitirá desde dar de alta a un abonado, a modificar sus datos o darlo de baja del sistema. También nos mostrará de forma sencilla todos los datos personales del abonado además de poder visualizar todas las sillas que tiene adquiridas. Otra de las funcionalidades de la aplicación es la gestión de palcos. el usuario podrá crear, modificar o borrar un palco del sistema. También tendrá la opción de poder visualizar y gestionar el estado en el que se encuentran las sillas de un palco, realizando la venta de sillas a un abonado o cancelando esta venta en caso de error.

La plataforma dispondrá solamente de un tipo de usuario, un administrador que será miembro de la Agrupación de Cofradías.

1.2 Objetivos

Los objetivos de nuestro proyecto son:

- Realizar un estudio del sistema de gestión y necesidades de una Agrupación de Cofradías para organizar los abonados de las sillas de Semana Santa en la carrera oficial. Este estudio consistirá en un análisis de las técnicas. relacionadas con la venta de entradas y de las plataformas ya existentes en el mercado actualmente.

- Diseñar una aplicación web para gestionar los abonados y las sillas de la carrera oficial. Este proceso se realizará tras elaborar un exhaustivo análisis de los requerimientos de la aplicación.

- Implementar un prototipo utilizando el *framework* para desarrollo web Spring MVC.

1.3 Metodología

Una decisión muy importante a la hora de empezar a trabajar en un proyecto es elegir cuidadosamente con qué metodologías vamos a trabajar en él, ya que estas no permiten definir los patrones que se deben seguir en cada una de las etapas de desarrollo de software.

Existen un gran número de metodologías que pueden aplicarse a la gestión de proyectos. Una de ellas es la metodología de desarrollo tradicional, que es la metodología de construcción de productos físicos. Esta metodología de desarrollo se caracteriza por:

- Se basa en ciclos de vida de desarrollo de software en cascada que organiza los proyectos de forma secuencial.

- Las etapas se ejecutan una sola vez, hasta que no finaliza una etapa de desarrollo no se puede pasar a la siguiente.

El problema de esta metodología es que no se adapta a lo que ocurre cuando nos enfrentamos a un proyecto real ya que los requisitos software son altamente variables, y la planificación y estimación de recursos inicial es muy imprecisa y normalmente suele desbordar las estimaciones.

En el lado opuesto nos encontramos las metodologías de desarrollo ágil [2], el término ágil aplicado al desarrollo de software nace en una reunión celebrada en 2001 en Utah (EEUU). En esta reunión se elaboró el manifiesto ágil [3], este manifiesto pone los pilares sobre lo que conocemos como metodologías de desarrollo ágil. Los principios derivados de este manifiesto son:

1. Satisfacción del cliente como prioridad.
2. Los cambios son bienvenidos.
3. Entregar frecuentemente software que funcione.
4. Personal de la empresa y desarrolladores trabajando juntos.
5. Motivación del equipo.
6. Comunicación cara a cara.
7. El software que funciona es la mejor medida de progreso.
8. Ritmo de desarrollo sostenible.
9. Atención continua a la calidad técnica y al buen diseño.
10. Maximizar el trabajo no hecho. No implementar más de lo acordado.
11. Equipos auto-organizados.
12. Autoevaluación.

En nuestro proyecto vamos a realizar una aplicación real, con unos requisitos impuestos por el cliente y un tiempo y coste determinados. Es por ello que la metodología que más se adapta a nuestras necesidades es una metodología de desarrollo ágil, debido a la gran aceptación a cambios en los requisitos que esta tiene. En particular vamos a utilizar la metodología Scrum, esta metodología se explica detalladamente en el punto 2.4 de esta memoria.

1.4 Estructura del documento

El presente documento se ha organizado de la siguiente manera:

En el primer apartado se indica el propósito y los objetivos de nuestro proyecto, así como las metodologías que se utilizaran para que se lleven a cabo.

Continuaremos con el segundo apartado en el que analizaremos cual es el problema al que tiene que dar solución nuestra aplicación. Se realizará una breve exposición sobre qué es la Semana Santa, además de examinar cuáles son los costes de este proyecto y qué plazos tenemos que cumplir.

Pasaremos al tercer apartado de esta memoria en el que pasaremos a diseñar la arquitectura y los objetos que formarán parte de nuestro proyecto. Igualmente se dará forma a la manera en que los objetos se almacenarán en la base de datos y a la interfaz con la que se comunicará el usuario.

En el apartado cuarto comentaremos la arquitectura que forma nuestro sistema, así como los principales detalles de la implementación del código para que el lector sea capaz de comprender la forma en la que se relacionan los distintos elementos del sistema.

En el punto cinco de la memoria se desarrollará la conclusión obtenida tras el desarrollo del proyecto y posibles propuestas de mejora del mismo.

El apartado seis está relacionado con la bibliografía utilizada para la elaboración de esta memoria.

Para finalizar esta memoria encontramos un conjunto de anexos en los que podemos encontrar un manual de usuario, y la descripción de los contenidos suministrados en el CD junto a esta memoria.

2 ANÁLISIS

Para comenzar la parte de análisis realizaremos una introducción sobre qué es la Semana Santa y qué supone una carrera oficial dentro de esta, para que así cualquier persona que lea esta memoria pueda comprender cuál es el problema que se expone y que nos ha llevado a obtener los requisitos del sistema desarrollado.

Estudiaremos las aplicaciones similares que existen en el mercado, y analizaremos sus ventajas y desventajas, y a partir de ahí realizaremos una propuesta de solución para nuestro problema. Más tarde realizaremos una descomposición de esta propuesta de solución, dando lugar a las historias de usuario y requisitos. Finalmente se mostrará el proceso de desarrollo que vamos a utilizar, una planificación de las tareas y una estimación del coste final del proyecto.

2.1 Análisis preliminar

2.1.1 Aproximación a la Semana Santa.

Para entender la utilidad que tiene nuestra aplicación web es necesario tener en cuenta la repercusión de la Semana Santa para la sociedad tanto a lo largo de la historia como en la actualidad.

En sus orígenes [4], la iglesia católica conmemora cíclicamente la Pascua del Señor, a partir de la asamblea eucarística convocada el primer día de la semana, día de la resurrección del Señor (dominicus dies) o domingo. Y, muy pronto, apenas en el siglo II , comenzó a reservarse un domingo particular del año para celebrar este misterio salvífico de Cristo. Las primeras manifestaciones de representaciones de la Pasión y Muerte de Cristo datan de la segunda mitad del siglo XV y del siglo XVI, viviendo su culminación en el siglo XVII. Se trata de unas fechas que coinciden en el tiempo con el Concilio de Trento (1545-1563) y la Contrarreforma, la respuesta que la Iglesia Católica dio a la Reforma Protestante de Martín Lutero en el Siglo XVI por la que se tomó la decisión exteriorizar la fe. En esta época hay indicios de un Semana Santa muy parecida a la que hoy conocemos.

Hoy en día entendemos por Semana Santa la conmemoración anual de la Pasión, Muerte y Resurrección de Jesús de Nazaret. Por eso, es un período de intensa actividad litúrgica dentro de las diversas confesiones cristianas. Da comienzo el Domingo de Ramos y finaliza el Domingo de Resurrección. La fecha de la celebración depende del calendario lunar. La Semana Santa va precedida

por la Cuaresma, que finaliza en la Semana de Pasión donde se celebra la eucaristía en el Jueves Santo, se conmemora la Crucifixión de Jesús el Viernes Santo y la Resurrección en la Vigilia Pascual durante la noche del Sábado Santo al Domingo de Resurrección. Durante la Semana Santa tienen lugar numerosas muestras de religiosidad popular a lo largo de todo el mundo, destacando las procesiones y las representaciones de la Pasión [5].

Esta celebración tiene lugar en muchas partes del mundo, todas ellas conmemoran la Pascua. En casi toda Europa, la celebración es muy parecida. En Alemania, Suiza o los países nórdicos, la tradición dice que hay una liebre que pinta y esconde los llamados 'huevos de Pascua'. En Italia se vive con mucha devoción. Los italianos, a parte de los intensos días de procesiones y pasteles tradicionales, cuentan con la cita más importante: la bendición papal del Urbi et Orbi en la plaza de San Pedro del Vaticano. En Francia, el Jueves y Sábado Santo las campanas del país no repican. Un silencio por la muerte de Jesús. En Australia, como en Estados Unidos donde la religión católica es minoritaria, no se celebran grandes liturgias, pero los comercios sí utilizan estas fechas para vender, al igual que en Centroeuropa, dulces y huevos de chocolate. En Sudamérica, casi 3 millones de mexicanos se congregan en el Cerro de la Estrella de Iztapalapa, en México D.F, donde más de 6.000 personas recrean la Pasión de Cristo por las calles de la ciudad. En Filipinas, recrean el mismo dolor que dicen sufrió Jesús, 10 personas son crucificadas y otras tantas pasean por la ciudad dándose latigazos hasta sangrar. Es en Jerusalén donde más sentido tiene la Semana Santa, y en la que con más fervor se celebra. Allí, los visitantes que se desplazan hasta la Ciudad Santa, pueden recorrer los lugares que se citan en la Biblia, y asistir a numerosas misas, que les harán transportarse 2016 años atrás [6].

En España se paralizan numerosas ciudades durante una semana, principalmente las que cuentan con celebraciones consideradas de Interés turístico Nacional: Sevilla, Granada, León, Murcia, Valladolid, Cáceres, Zamora…hasta cuarenta y una ciudades. Esta semana tiene un gran impacto económico, más allá de su interés religioso.

La Semana Santa en España no se celebra de la misma manera, cada provincia y cada pueblo tiene sus costumbres, tradiciones y gastronomía propia de esta fiesta. Andalucía tiene una serie de valores culturales apreciados hoy y ausentes en otras regiones y naciones. Son valores admirados y codiciados por otros; de aquí que se dedique a comercializarlos. No solo vive estos valores para sí misma, sino que se dedica a producirlos para venderlos y ganar así dinero y admiración. La Semana Santa se está desarrollando o resucitando en muchos sitios por la influencia de la imitación de pueblos o ciudades valorados en el conjunto de Andalucía y en los que la semana santa funciona como elemento de refuerzo de ese valor. En este sentido, diríamos que la feria de Sevilla, su semana santa y la romería del Rocío están sirviendo de término de referencia o de modelo

para muchos pueblos y ciudades de Andalucía, que se lanzan a aprender a bailar sevillanas, a llevar los pasos y acudir a las procesiones de semana santa o a fundar su hermandad del Rocío [7].

La Semana Santa promueve la economía y constituye un recurso turístico de primer orden que moviliza los mayores gastos de consumo en una ciudad, por parte de los residentes y turistas que asisten como espectadores a las procesiones de Semana Santa. Los agentes que generan el gasto directo son las Cofradías y Hermandades, la asociación de cofradías de la ciudad, y por supuesto los participantes e integrantes de la Semana Santa.

Si nos centramos en ejemplos reales, según un estudio, la Semana Santa de Palencia [8] tiene en la ciudad un impacto económico que supera los 2,4 millones de euros. Una cantidad que puede parecer insignificante si los comparamos con los 280 millones de euros que generó la Semana Santa de Sevilla el pasado año 2015, esta cifra representa el 1,26% del producto interior bruto de Sevilla [9].

2.1.2 ¿Qué es una carrera oficial?

El historiador Juan Carrero Rodríguez, en su Diccionario cofradiero define a la carrera oficial como [10]:
"*Calles de la ciudad, fijadas por la autoridad, por las que han de pasar obligatoriamente todas las cofradías en su anual recorrido para cumplir la estación penitencial, y donde en ciertos lugares fijos hay establecidos ciertos controles por el Consejo General para hacer cumplir el horario durante el tránsito*."

Este recorrido, normalmente engalanado al efecto, se realiza por calles emblemáticas de la localidad, por lo que suele ser lugar privilegiado para ver el paso de las cofradías, a veces incluso desde palcos y sillas de alquiler. Suele estar dotado de la llamada tribuna oficial, desde la que presiden las autoridades locales. Normalmente, el recorrido trae consigo el paso de la estación de cada cofradía por la iglesia mayor de la localidad (catedral o colegiata), desde donde emprenden el camino de regreso a su sede [11].

La sucesión de calles que componen la carrera oficial suele ser muy estable, incluso tradición fijada y difícilmente alterable. Por su parte, el horario de paso que le corresponde en ella a cada cofradía se fija anualmente por una asamblea de las corporaciones que van a concurrir a la misma. Esta asamblea se reúne en el seno de la institución que agrupa y gobierna a las corporaciones penitenciales (y normalmente de gloria) de la localidad: denominada unas veces consejo general (Sevilla) y otras agrupación (Córdoba, Jaén, o Almería) o federación (Granada).

Esta asociación tiene a su cargo hacer públicos los horarios correspondientes a cada día de la semana, vigilar su cumplimiento por parte de las respectivas cofradías, e incluso sancionar el incumplimiento de los mismos. Al efecto, la correspondiente asociación suele disponer a lo largo de la carrera oficial uno o varios puntos de control de paso (palquillos) donde los representantes de la cofradía en cuestión y de la asociación fiscalizadora firman conjuntamente un acta recogiendo el horario de paso efectivo de aquella que luego ha de surtir los efectos pertinentes.

De este modo, donde la asociación de hermandades disfruta de la concesión municipal para explotar económicamente el alquiler de sillas y palcos, dispuestos para presenciar el paso de las cofradías, los beneficios derivados de ello se distribuyen equitativamente entre todas las hermandades que hacen acto de presencia en la carrera, salvo que un incumplimiento del horario y los perjuicios que esto conlleva para las otras cofradías concurrentes, dé lugar a una penalización económica de la hermandad contraventora.

Por ejemplo, en Cádiz el el beneficio alcanzado con la venta de sillas del año 2015 fue de 150.000 euros [12], cifra que se que queda muy pequeña si comparamos, otra vez, el beneficio que generó la carrera oficial de Sevilla que fue de 3,78 millones de euros [13]. Estamos por tanto antes la “gallina de los huevos de oro” de la Semana Santa, tanto es su potencial que incluso existe un mercado negro de reventa de sillas que mueve en Sevilla miles de euros de forma fraudulenta [14]. Debido a la gran cantidad de dinero que genera esta venta de sillas en todas las ciudades en la que se implanta es necesario que la gestión de la carrera oficial sea lo más eficiente posible, y que no dé lugar a que se cometan errores. Pero como podremos comprobar en el siguiente apartado de esta memoria esto no es así ya que aún en pleno siglo XXI la venta de estos abonos sigue manteniendo un proceso muy tradicional.

Ilustración 2.1 - Carrera oficial de Jaén antes del paso de una cofradía. Foto: M.Quesada Titos 2015.

2.1.3 Necesidades en la gestión de una carrera oficial en Semana Santa

En el mes de mayo de 2015, miembros de la Agrupación de Cofradías de Jaén se ponen en contacto con el equipo de desarrollo para intentar poner una solución informática para el problema que es expone a continuación.

El proceso de venta de abonos que se han llevado a cabo tradicionalmente es el siguiente:

- Primero se notifica por correo ordinario a los abonados del año anterior que el proceso de renovación de sus abonos ha comenzado. Para su renovación deben acudir en persona a la sede de la Agrupación de Cofradías en las fechas y horas indicadas en la carta. Una vez concluido el proceso de renovación, si un abonado no ha acudido a renovar su abono, pierde las sillas que ocupó el año anterior.

- Finalizado este primer periodo, se abre el plazo de venta libre de abonos en el que cualquier persona puede adquirir una silla de las que no hayan sido ocupadas en el periodo anterior.

La forma de venta de los abonos es la misma para los dos periodos: una persona que esté interesada tanto en renovar como en adquirir un nuevo abono debe acudir personalmente a la sede de la Agrupación de Cofradías y allí un miembro de la misma procederá a la venta. Tanto los datos del cliente como las sillas que ha adquirido quedan apuntados en una libreta de papel. Aunque esta

acción se realiza con la mayor atención y cuidado posible conlleva a un gran cúmulo de fallos que pasarán factura en Semana Santa.

Uno de los problemas más importantes que ha acarreado esta forma de venta se hizo presente en la Semana Santa del año 2015. Una nueva Junta de Gobierno había entrado a gestionar la Agrupación ese mismo año [15], por lo tanto todos los miembros de la Agrupación encargados de gestionar la carrera oficial eran inexpertos ya que era su primera vez. El Domingo de Ramos cuando la cruz de guía de la Cofradía de la Borriquita se acercaba a la calle Bernabé Soriano, los miembros de la Agrupación se afanaban por acomodar a un gran número de abonados que esperaban impacientemente. La sorpresa para estos acomodadores fue que en el proceso de venta de los abonos se habían vendido sillas que no existían en la configuración real de la carrera oficial. Palcos con una capacidad de 50 sillas se habían tomado con una capacidad de 54 sillas vendiendo 4 sillas que no existían, por ejemplo.

Este problema supuso un esfuerzo titánico para los miembros de la Agrupación que tuvieron que reacomodar a las personas con algún fallo en su abono. La situación desembocó en el enfado de muchos abonados perjudicados que no volvieron a comprar su abono al año siguiente, además de un estrés y preocupación en los miembros de la Agrupación totalmente innecesario.

La situación vivida puede evitarse en los años venideros cuidando con todo detalle el proceso de venta de abonos. Para eso se desarrollará una aplicación robusta que gestione de forma eficiente los palcos y abonados de la carrera oficial y que disponga de una interfaz clara y fácil de usar ya que la aplicación la gestionan personas de mediana edad con unos conocimientos de informática básicos.

2.1.4 Estado del arte. Aplicaciones similares.

Tras realizar un estudio sobre el proceso que realizan las asociaciones de cofradías para la venta de los abonos de Semana Santa la conclusión que hemos obtenido nos muestra que estas asociaciones realizan un proceso de venta muy tradicional en el que las nuevas tecnologías no tienen demasiada presencia.
Si queremos encontrar aplicaciones web destinadas a la venta de abonos tenemos que trasladarnos al ámbito del cine, teatro o encuentros deportivos.

2.1.4.1 Consejo de Cofradías de Sevilla

Después de analizar el proceso de venta de abonos de las ocho capitales de provincia de Andalucía, únicamente el Consejo General de Hermandades y Cofradías de Sevilla dispone de una simple plataforma para realizar distintas solicitudes [16].

Ilustración 2.2 - Logo de la aplicación de gestión de palcos del Consejo de Cofradías de Ilustración Sevilla

Esta plataforma permite realizar la renovación, pago, cesión y solicitud de nuevos abonos a través de internet, en las fechas indicadas en su página web. Para poder obtener estos servicios, el usuario debe introducir su DNI y una referencia que ha debido de recibir mediante correo postal postal, por lo tanto no permite el registro de nuevos usuarios desde la plataforma.

Ilustración 2.3 - Página de inicio de la aplicación de gestión de palcos del Consejo de Cofradías de Sevilla

Ilustración 2.4 - Página para la renovación de abonos de la aplicación de gestión de palcos del Consejo de Cofradías de Sevilla

Ventajas:

- Permite al usuario realizar gestiones básicas sobre su abono.

Inconvenientes:

- Se trata de un sistema propietario hecha a medida para esta institución con muchas restricciones debido a la complejidad de la carrera oficial de Sevilla. Esto puede suponer un problema para intentar reutilizar la aplicación.

2.1.4.2 Unión Cine Ciudad

La venta de entradas online de cine es un campo que guarda mucha similitud con nuestra aplicación. Para estudiar un caso en particular hemos elegido la empresa Unión Cine Ciudad [17]. Unión Cine Ciudad es una empresa que dispone de salas de cine distribuidas por toda España y que dispone de un gran soporte web para realizar la compra de entradas.

Ilustración 2.5 - Logo de Unión Cine Ciudad

El proceso de compra es el siguiente, primero el usuario debe de seleccionar el cine al que va a asistir. Una vez seleccionado el cine aparecen todas las películas con las distintas sesiones en las que esta se emite.

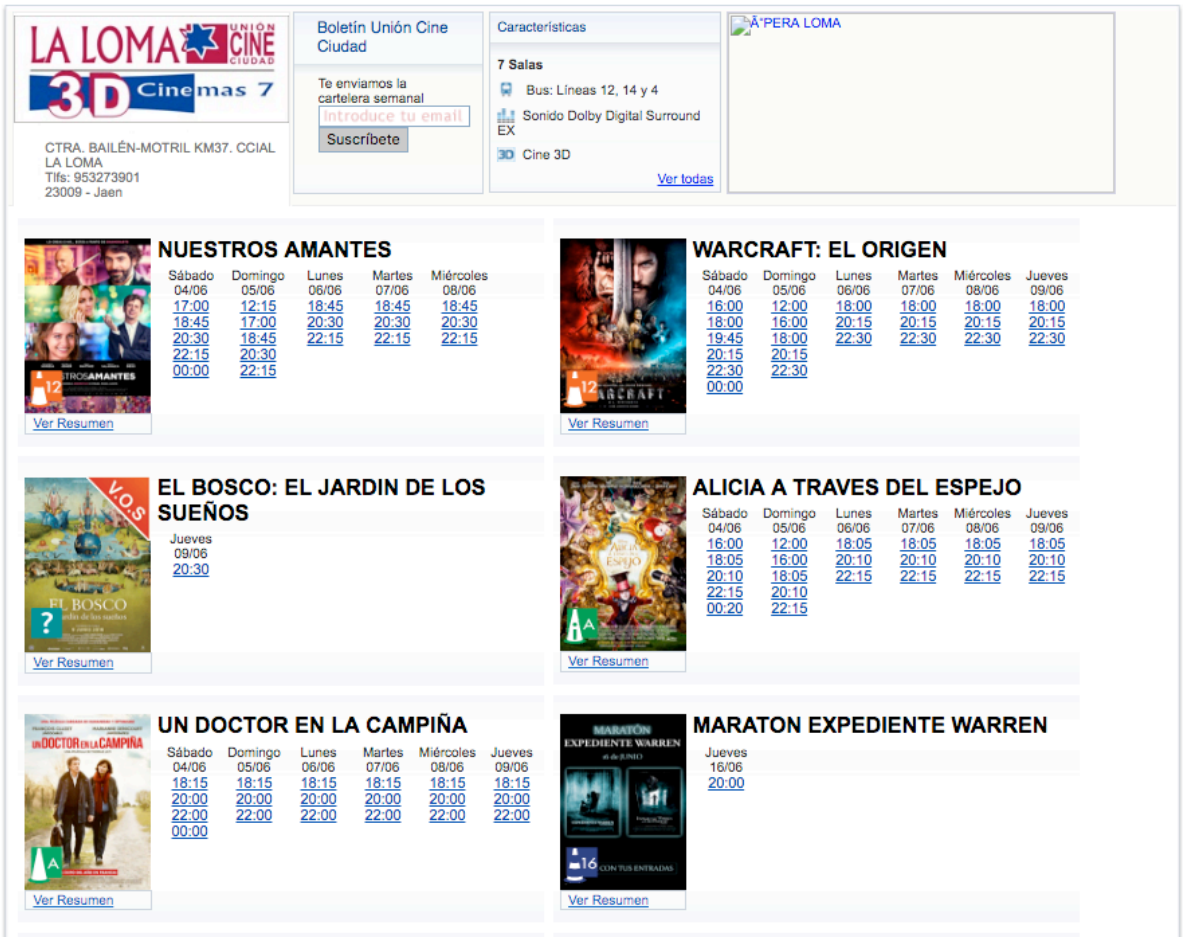

Ilustración 2.6 - Cartelera de películas de la web de Unión Cine Ciudad.

Tras seleccionar la sesión a la que vamos a acudir el sistema nos reenviará a una web externa que es la que se encarga de realizar el proceso de venta [18]. En esta página aparece un plano de la sala de cine en el que las butacas de color verde indican que la silla está libre, y las sillas de color rojo indica que está ocupada. Tras seleccionar las sillas que queremos adquirir y pulsar sobre el botón de comprar hay que proceder al registro, y una vez que el usuario esté registrado puede pagar sus entradas mediante una plataforma de pago segura.

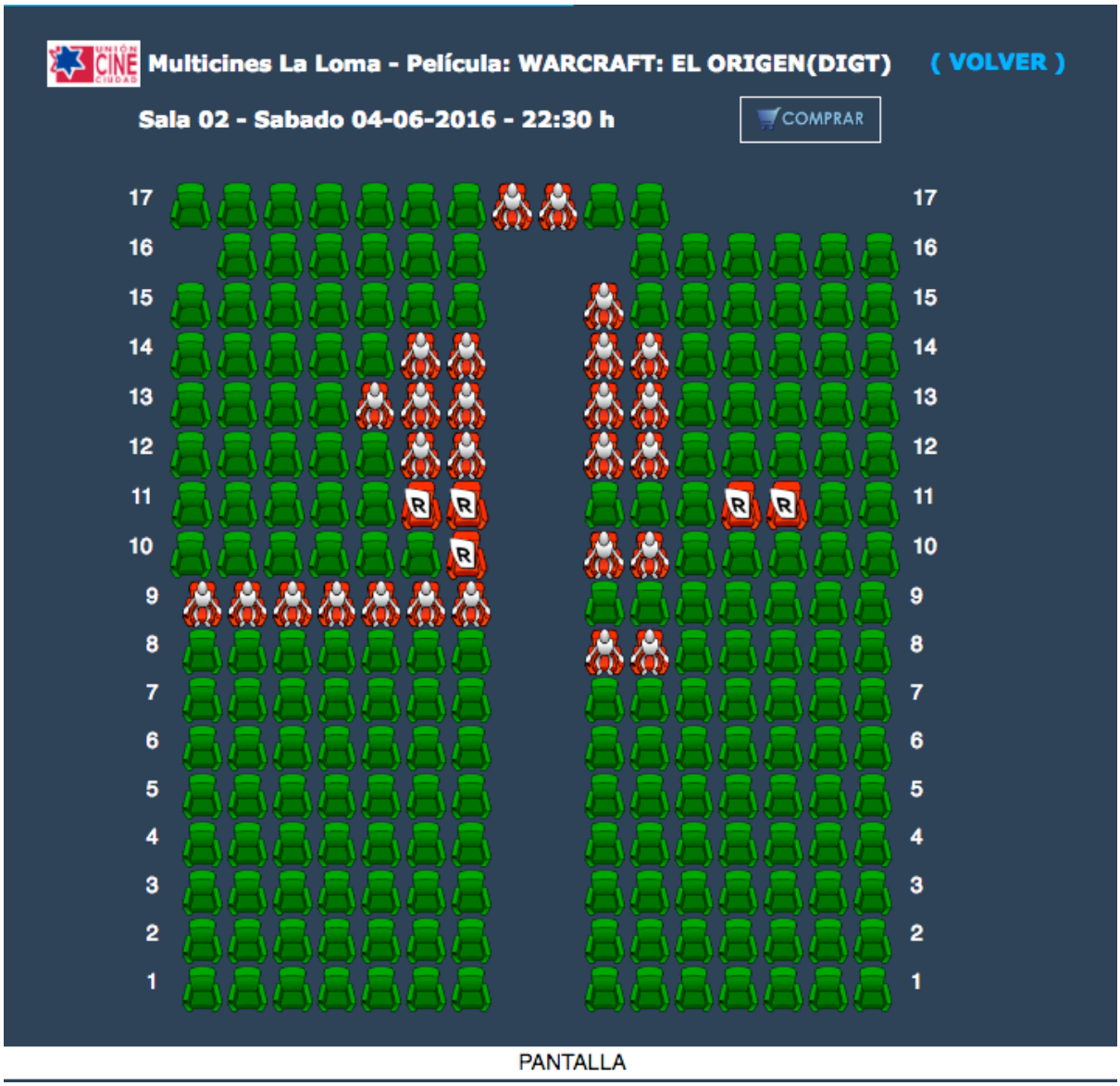

Ilustración 2.7 - Página de compra de entradas de la web de compraentradas.com

Ventajas

- Permite al usuario elegir las sillas que desea comprar libremente.
- Permite el pago de estas mediante plataforma de pago segura.

Inconvenientes

- Esta aplicación es poco configurable, y enfocada a espectáculos que se organizan por sesiones.

2.2 Propuesta de solución

Una vez que hemos realizado un análisis preliminar de nuestro problema y tras decidir que lo que más se ajusta a nuestras necesidades sería el **desarrollo de una aplicación web** que permita al usuario gestionar de forma integral tanto los abonados de la carrera oficial como los palcos.

A la hora de definir el diseño de nuestra aplicación, utilizaremos técnicas que favorezcan el mantenimiento y escalado de la aplicación, definiendo una arquitectura modular donde estén separadas la competencias de cada módulo, para ello hemos optado por utilizar el **patrón Modelo Vista Controlador**. El patrón MVC es un patrón arquitectónico que divide a la aplicación en tres tipos de componentes: modelos, vistas y controladores. La aplicación de este patrón arquitectónico garantiza un bajo acoplamiento, dado que los cambios de la interfaz no afectan a las clases del modelo, y una alta cohesión debido a que el modelo se centra en los procesos de negocio despreocupándose de los detalles de las vistas. El patrón MVC se detalla en el punto 4.1.2 de esta memoria.

Para facilitar la implementación de este patrón vamos a utilizar un *framework* de desarrollo. En nuestro caso el *framework* elegido ha sido **SpringMVC** [19], debido a que es muy flexible e implementa toda su estructura como interfaces y la facilidad a la hora de su uso y configuración.

Si pasamos a ver como vamos a desarrollar la parte del cliente, se ha optado por utilizar el *framework* de software libre **Twitter Bootstrap** [20]. Bootstrap nos permite crear interfaces de usuario web con CSS [21] y JavaScript [22] que adaptan la web al tamaño del dispositivo en el que se visualice, por lo tanto se trata de una solución perfecta para "*responsive design*". Además incluye diseños predefinidos y un gran número de plantillas que nos facilitarán considerablemente el diseño de la interfaz.

Con respecto a la metodología de desarrollo que vamos a utilizar, se ha creído conveniente elegir la metodología de desarrollo ágil **Scrum**[23], al tratarse de un proyecto real. Esta metodología es tolerante a cambios en los requisitos, en caso de que el cliente lo solicitara y además una de sus prioridades es la plena satisfacción del cliente. Para ello se establece una estrecha relación con el cliente en la que prima la comunicación y en la que se realizaran entregas frecuentes de software en etapas tempranas.

2.3 Historias de usuario

Las historias de usuario son un instrumento usado en metodologías ágiles para obtener los requisitos de una aplicación a partir de la información obtenida por los usuarios [24].

Las historias de usuario evolucionan a lo largo de la vida del proyecto, adaptándose a los cambios en los requisitos. Se trata de peticiones concretas con un tamaño pequeño adecuado para el trabajo iterativo. Deben de ser lo suficientemente detallados como para poder empezar a trabajar, los detalles se añadirán en fase de desarrollo [25].

Las historias de usuario están formadas por un id, titulo, prioridad, estimación, descripción y dependencias, en caso de que una historia de usuario dependa de otra.

Nuestras historias de usuario son las siguientes:

ID	TÍTULO	PRIORIDAD	PUNTOS ESTIMADOS	DESCRIPCIÓN
1	Configuración del entorno	Alta	10	Como desarrollador quiero configurar el entorno para que se pueda iniciar el proceso de desarrollo.
2	Crear abonado	Alta	5	Como usuario quiero ser capaz de crear “abonados” para poder gestionar sus características.
3	Buscar abonado	Alta	3	Como usuario quiero ser capaz de buscar un abonado para poder visualizar sus datos.
4	Visualizar abonado	Alta	2	Como usuario quiero ser capaz de visualizar un abonado para conocer todas sus características
5	Visualizar palco	Alta	5	Como usuario quiero visualizar los datos para conocer todas sus características
6	Visualización de las sillas	Alta	5	Como usuario quiero visualizar las sillas de un palco en forma de matriz, ordenadas por filas, las sillas libres de color verde y las

				sillas ocupadas de color rojo, para poder conocer de un vistazo el estado de ocupación de un palco.
7	Reserva de sillas	Alta	10	Como usuario quiero ser capaz de realizar la reserva de silla/as a un abonado. Las sillas que se van a reservar se seleccionarán de la matriz de sillas.
8	Modificar un abonado	Alta	3	Como usuario quiero poder modificar los datos de un abonado.
9	Borrar un abonado	Alta	3	Como usuario quiero poder borrar un abonado.
10	Imprimir un palco	Alta	2	Como usuario quiero poder imprimir un listado con todos los detalles de un palco.
11	Autenticarse en el sistema	Media	5	Como usuario quiero poder autenticarme en el sistema para poder trabajar con la aplicación.
12	Crear un palco	Media	7	Como usuario quiero ser capaz de crear “palcos” para poder gestionar sus características
13	Modificar un palco	Media	3	Como usuario quiero poder modificar un palco.
14	Borrar un palco	Media	3	Como usuario quiero poder dar de baja un palco.
15	Herramienta de cambio de año	Media	7	Como usuario quiero ser capaz de crear “palcos” para poder gestionar sus características.
16	Cerrar sesión	Baja	3	Como usuario quiero poder cerrar la sesión de forma segura.

Tabla 2.1 - Historias de usuario

2.4 Requisitos del sistema

Definimos requisito como: una condición o capacidad exigida por el usuario para solucionar un problema o alcanzar un objetivo.

Según la clasificación realizada por H. Pohl [26] podemos clasificar los requisitos en tres tipos que son los siguientes:

Requisitos funcionales: en ellos se especifica la funcionalidad que el sistema debe proporcionar a los usuarios. Los requisitos funcionales de nuestra aplicación son:

1. **Identificación de usuario:** el usuario debe poder identificarse en la plataforma, introduciendo su nombre de usuario y contraseña.
2. **Añadir un nuevo abonado:** un usuario debe poder crear un nuevo abonado introduciendo de forma manual sus datos personales.
3. **Modificar datos de un abonado:** un usuario debe poder modificar los datos de cualquier abonado de la plataforma.
4. **Borrar abonado:** un usuario debe tener la opción de eliminar todos los datos de un abonado.
5. **Visualizar los datos de un abonado:** un usuario debe tener la opción de visualizar todos los datos de un abonado registrado en la plataforma.
6. **Buscar a un abonado por su nombre y apellidos:** un usuario debe tener la opción de poder buscar un abonado registrado en la plataforma introduciendo su nombre y/o sus apellidos.
7. **Añadir un nuevo palco:** un usuario debe poder crear un nuevo palco introduciendo de forma manual todos sus datos.
8. **Modificar los datos de un palco:** un usuario debe poder modificar los datos de un palco.
9. **Borrar un palco:** un usuario debe tener la opción de eliminar todos los datos de un palco.
10. **Ver un listado de todos los palcos**: un usuario debe tener la posibilidad de consultar un listado con todos los palcos del sistema con así lo desee.
11. **Filtrar los palcos por año**: un usuario debe tener la posibilidad de filtrar el listado de palcos por año.
12. **Visualizar los datos de un palco:** un usuario debe poder visualizar los datos principales de un palco.
13. **Visualizar los datos de un palco de forma detallada:** un usuario debe poder visualizar los principales datos de un palco además de una tabla en la que se indique las sillas del palco, así como el nombre del abonado que la ha comprado y la fecha de la venta.

14. **Imprimir los datos de un palco:** un usuario debe poder imprimir en papel los datos que muestra la vista del palco detallada.
15. **Realizar la venta de silla/as a un abonado:** un usuario debe poder realizar la venta de sillas a un abonado.
16. **Visualizar los datos de un abonado sobre una silla vendida:** un usuario debe poder visualizar los datos de un abonado que ha comprado una silla sobre el plano de las sillas de un palco.
17. **Liberar una silla vendida**: un usuario debe tener la opción de eliminar una venta de una silla a un abonado, dejando a esa silla libre.
18. **Configurar el sistema para el comienzo de un nuevo año:** un usuario debe poder preparar la plataforma automáticamente para una nueva Semana Santa.
19. **Cierre de sesión:** el usuario debe de poder cerrar su sesión y salir de la aplicación.

Requisitos de calidad: requisitos no funcionales relacionados con la calidad del sistema en desarrollo.

1. **Usabilidad:** La aplicación deberá de tener una gran facilidad de uso. El grado de esfuerzo del usuario para interpretar las entradas o salidas del sistema deberá de ser muy bajo.
2. **Disponibilidad**. El sistema deberá de tener un alto porcentaje de tiempo durante el cual estará plenamente operativo y disponible para usarlo.
3. **Confiabilidad**: La probabilidad de que el sistema funcione sin fallos debe ser alta.

Restricciones: limitan el proceso de desarrollo o las propiedades del sistema a desarrollar.

- Las funcionalidades más importantes deben de estar implementadas antes del mes de enero de 2016.
- El sistema estará disponible con todas sus funcionalidades antes del 14 de junio de 2016.

2.5 Planificación de tareas.

En este apartado vamos a describir cómo a partir de las historias de usuario[27] se ha realizado un proceso de identificación de las tareas necesarias para que se puedan llevar a cabo. Más tarde realizaremos una estimación del tiempo que nos llevará poder cumplir todas las historias de usuario utilizando la metodología ágil Scrum. Finalmente se expone la forma en la que se ha desarrollado las iteraciones estimadas en realidad.

2.5.1 Descomposición de tareas

En la siguiente tabla se muestran las historias de usuario del apartado 2.4 divididas en subtareas, a cada subtarea se le ha asignado unos puntos de historia determinados según una estimación, también se detalla la persona del equipo encargada de realizar cada tarea. Los puntos de historia son la forma que tenemos en Scrum de estimar el coste de una tarea. Es una estimación relativa ya que estimamos en función de la diferencia de coste entre una historia u otra.

ID	TAREA	ESTIMACIÓN (en puntos de historia)	PERSONAL ENCARGADO
1	**CONFIGURACIÓN DEL ENTORNO**	**10**	
1.1	Instalación y configuración de SpringMVC	3	Analista
1.2	Análisis y diseño de la interfaz y arquitectura de la aplicación	4	Analista
1.3	Creación controladores	1	Programador
1.4	Creación de la base de datos	2	Programador
2	**CREAR ABONADO**	**5**	
2.1	Análisis y diseño de la funcionalidad	2	Analista
2.2	Configuración del controlador	1	Programador
2.3	Configuración del DAO	1	Programador
2.4	Configuración de las vistas	1	Programador
3	**BUSCAR ABONADO**	**3**	
3.1	Configuración del controlador	1	Programador
3.2	Configuración del DAO	1	Programador
3.3	Configuración de las vistas	1	Programador
4	**VISUALIZAR UN ABONADO**	**2**	
4.1	Configuración del controlador y DAO	1	Programador
4.2	Configuración de las vistas	1	Programador
5	**VISUALIZAR PALCO**	**5**	
5.1	Análisis y diseño de la funcionalidad	2	Analista
5.2	Configuración del controlador	1	Programador

5.3	Configuración del DAO	1	Programador
5.4	Configuración de las vistas	1	Programador
6	**VISUALIZACIÓN DE LAS SILLAS**	**5**	
6.1	Análisis y diseño de la funcionalidad	1	Analista
6.2	Configuración del controlador	1	Programador
6.3	Configuración del DAO	1	Programador
6.4	Configuración de las vistas	2	Programador
7	**RESERVA DE SILLAS**	**10**	
7.1	Análisis y diseño de la funcionalidad	4	Analista
7.2	Configuración del controlador	3	Programador
7.3	Configuración del DAO	1	Programador
7.4	Configuración de las vistas	2	Programador
8	**MODIFICAR UN ABONADO**	**3**	
8.1	Configuración del controlador	1	Programador
8.2	Configuración del DAO	1	Programador
8.3	Configuración de las vistas	1	Programador
9	**BORRAR ABONADO**	**3**	
9.1	Configuración del controlador	1	Programador
9.2	Configuración del DAO	1	Programador
9.3	Configuración de las vistas	1	Programador
10	**IMPRIMIR PALCO**	**2**	
10.1	Configuración del controlador	1	Programador
10.2	Configuración de las vistas	1	Programador
11	**AUTENTICARSE EN EL SISTEMA**	**5**	
11.1	Análisis y diseño de la funcionalidad	2	Analista
11.2	Configuración del realm	1	Programador
11.3	Configuración del DAO	1	Programador
11.4	Configuración de las vistas	1	Programador
12	**CREAR PALCOS**	**7**	

12.1	Análisis y diseño de la funcionalidad	3	Analista
12.2	Configuración del controlador	2	Programador
12.3	Configuración del DAO	1	Programador
12.4	Configuración de las vistas	1	Programador
13	**MODIFICAR PALCO**	**3**	
13.1	Configuración del controlador	1	Programador
13.2	Configuración del DAO	1	Programador
13.3	Configuración de las vistas	1	Programador
14	**BORRAR PALCO**	**3**	
14.1	Configuración del controlador	1	Programador
14.2	Configuración del DAO	1	Programador
14.3	Configuración de las vistas	1	Programador
15	**HERRAMIENTA DE CAMBIO DE AÑO**	**7**	
15.1	Análisis y diseño de la funcionalidad	4	Analista
15.2	Configuración del controlador	1	Programador
15.3	Configuración del DAO	1	Programador
15.4	Configuración de las vistas	1	Programador
16	**CERRAR SESIÓN**	**3**	
16.1	Análisis y diseño de la funcionalidad	2	Analista
16.2	Configuración del controlador	1	Programador

Tabla 2.2 - Descomposición de tareas

Tras realizar esta división de tareas, al analista de nuestro proyecto le han sido asignados 27 puntos de historia, mientras que al programador se le han asignado 49 puntos de historia.

2.5.2 Scrum

Scrum [28] es un proceso ágil en el que se aplican un conjunto de buenas prácticas para trabajar colaborativamente y obtener el mejor resultado para un proyecto.

En scrum los proyectos avanzan mediante un conjunto de *sprints* que suelen tener una duración de entre 2-4 semanas. Al finalizar un *sprint* se obtiene

software que funciona y se decide si liberarlo o seguir mejorándolo. Durante un *sprint* el producto es diseñado, codificado y probado. No se permiten cambios durante la realización de un *sprint* [29].

La secuencia de trabajo de un *sprint* es la siguiente:

1. Se realiza el análisis de los requisitos. Para cada requisito principal se crea un bloque de trabajo al que llamamos historia.
2. El cliente ordena las historias en una pila de trabajo, *product backlog*, según prioridad.
3. El equipo selecciona un grupo de historias, *sprint backlog*, con las que trabajará durante un *sprint*.
4. Al terminar el *sprint* se entrega el resultado del trabajo. Se vuelve al punto 2 hasta que la pila de trabajo quede vacía.

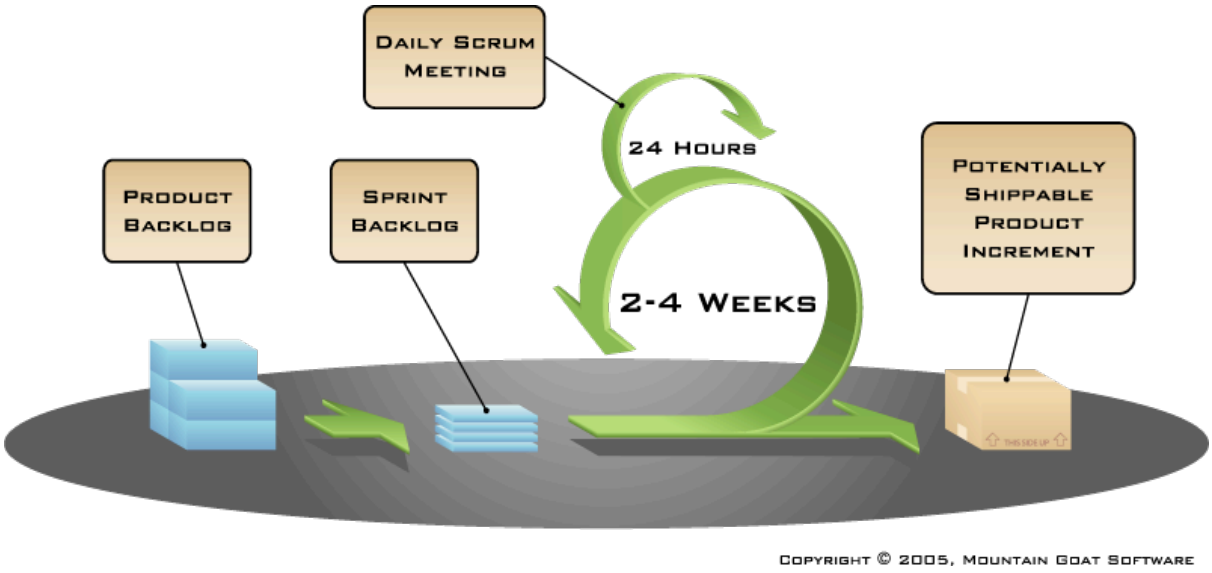

Ilustración 2.8 - Secuencia de trabajo de Scrum

Un equipo de scrum está formado por los siguientes roles:

- *Product owner:* es la persona responsable del producto, la que se encarga de comunicarse con el cliente.
- *Scrum master*: persona responsable de que scrum funcione correctamente.
- *Team*: equipo responsable del desarrollo.

También se realizan varios tipos de reuniones entre los miembros del equipo pero debido a que nuestro equipo está formado únicamente por una persona, no entraremos en más detalles sobre los roles que puede tomar cada una, y las reuniones en las que participarán.

2.5.3 Planificación de las Iteraciones

Según la estimación realizada en los apartados anteriores, la suma de todos los puntos de historia es de 76 puntos. El equipo de trabajo está formado únicamente por una persona, que realizará los roles de analista y programador.

Dado que la duración de un *sprint* debe estar entre 2 y 4 semanas se estima conveniente que la duración de cada *sprint* sea de 3 semanas naturales, lo que corresponde a 15 días hábiles.

Nuestro proyecto cuenta con dos principales fechas a tener en cuenta:

- Últimos días de diciembre de 2015: las historias de usuario que cuentan con una prioridad alta deben estar implementadas para esa fecha. Debido a que los primeros días del mes de enero la Agrupación de Cofradías y Hermandades de Jaén comienza su proceso de venta de abonos. Para que no se vuelvan a repetir los problemas ocasionados en años anteriores, estas funcionalidades deben de estar implementadas.
- Primeros días de junio de 2016: debido a que el día 14 de junio de dos mil dieciséis finaliza el plazo de entrega de trabajos fin de grado, nuestro prototipo deberá de estar finalizado en las primeras semanas del mes de junio.

Se acuerda con el cliente que en el periodo que transcurre desde el comienzo de la venta de abonos hasta el final de la Semana Santa se realizará una pausa en el desarrollo del proyecto.

Para comenzar a trabajar se realiza una estimación, se cree conveniente que el número máximo de puntos de historia por sprint será de 10. La estimación es la siguiente:

Sprint	Fecha inicio	Fecha fin	Puntos completados	Días de trabajo	Velocidad	ID Historias a realizar
1	7-9-2015	25-9-2015	10	15	66%	1
2	28-9-2015	16-10-2015	10	15	66%	2,3,4
3	19-10-2015	6-11-2015	10	15	66%	5,6
4	9-11-2015	27-11-2015	10	15	66%	7
5	30-11-2015	18-12-2015	8	15	53%	8,9,10
6	4-4-2016	22-4-2016	10	15	66%	12,13
7	25-4-2016	13-5-2016	10	15	66%	14,15
8	16-5-2016	3-6-2015	8	15	53%	11,16

Tabla 2.3 - Estimación del proyecto

Según lo visto en la tabla anterior todas las historias de usuario estarán realizadas en un total de 8 *sprints*. Si el primer *sprint* comienza el 7 de septiembre de 2015, en 5 *sprint* se llegaría a la primera fecha marcada en nuestro calendario y el cliente obtendría un producto funcional en el que se incluyen todas las historias de usuario marcadas con una prioridad alta.

La Semana Santa de 2016 finalizó el día 27 de marzo, Domingo de Resurrección. En los días posteriores se mantendrá una reunión con el cliente para que nos muestre su opinión tras utilizar nuestra aplicación durante estos meses.

Si todo marcha como esperamos, el día 4 de abril de 2016 dará comienzo el 6 *sprint*, el octavo y último *sprint* finalizará el día 3 de junio, quedando una holgura de dos semanas para la entrega del proyecto por si ocurriera algún inconveniente.

Esta estimación se ha realizado con una velocidad baja a conciencia, no dándole toda la carga de trabajo posible a cada *sprint*. Para calcular la velocidad del *sprint* debemos de dividir los puntos completados en el *sprint* entre las jornadas de trabajo [30].

Sprint 1
Puntos de historia completados:10
Jornadas de trabajo: 15
Velocidad: 10 / 15 = 0,666666
El equipo ha trabajado en el 1º sprint con una velocidad de un 66%

Esto significa que el equipo de trabajo ha podido dedicar el 66% de su jornada laboral a acciones concretas de desarrollo programadas para el *sprint*. El otro 44% restante ha sido invertido en realizar otras acciones, tales como acudir a clases en la universidad, visitar al tutor del proyecto para consultar alguna duda o corregir algún error sobre la marcha. Al planificar los *sprint* con este colchón de un 44%, en este caso, aportamos al equipo una herramienta que permite asumir el trabajo previsto sin que cause un impacto negativo en la planificación.

2.5.4 Desarrollo de las iteraciones.

Tras realizar esta primera estimación y temporización del proyecto pasamos a ver cómo se ha desarrollado realmente.

Tal y como podemos ver en la tabla 2.6 hasta la finalización del *sprint* 5 se cumplen todos los plazos y las historias de usuario que indicamos en la tabla 2.3.

Entre el 5º y 6º *sprint* se produce la reunión con el cliente para mostrar su opinión sobre el primer periodo de prueba. La reunión en cuestión tiene lugar el jueves 31 de marzo de 2016. En ella la Agrupación de Cofradías de Jaén muestra su satisfacción obtenida tras el uso de la aplicación indicando que no se ha producido el error que venían acumulando durante tantos años atrás, la venta de sillas inexistentes en los palcos. No todo fueron elogios, también indicaron ciertos cambios que dieron lugar a la siguiente historia de usuario.

ID	TÍTULO	PRIORIDAD	PUNTOS ESTIMADOS	DESCRIPCIÓN
17	Modificaciones a realizar en el proyecto	Alta	3	Modificaciones a realizar en el proyecto tras la reunión mantenida con el cliente

Tabla 2.4 - Historia de usuario 17

ID	TAREA	ESTIMACIÓN (en puntos de historia)	PERSONAL ENCARGADO
17	**MODIFICACIONES A REALIZAR**	**3**	
17.1	Cambio de posición de distintos botones	1	Programador
17.2	Modificación de la interfaz de usuario general	3	Programador
17.3	Modificación de la vista para la compra de abonos	3	Programador

Tabla 2.5 - Descomposición de la historia 17 en tareas

Al añadir una nueva historia de usuario el proyecto sufre una re-estimación. Es por ello que queda de la siguiente manera:

Sprint	Fecha inicio	Fecha fin	Puntos completados	Días de trabajo	Velocidad	ID Historias a realizar
1	7-9-2015	25-9-2015	10	15	66%	1
2	28-9-2015	16-10-2015	10	15	66%	2,3,4
3	19-10-2015	6-11-2015	10	15	66%	5,6
4	9-11-2015	27-11-2015	10	15	66%	7
5	30-11-2015	18-12-2015	8	15	53%	8,9,10
6	4-4-2016	22-4-2016	8	15	53%	17,11
7	25-4-2016	13-5-2016	10	15	66%	12,13
8	16-5-2016	3-6-2016	10	15	66%	14,15
9	6-6-2016	10-6-2016	3	5	60%	16

Tabla 2.6 - Re-estimación del proyecto

De esta re-estimación surge un nuevo *sprint* que duró únicamente una semana debido a que la historia de usuario que faltaba tenía solo 3 puntos de historia.

El siguiente diagrama es un diagrama *burn down* o diagrama de evolución, que indica de forma gráfica el trabajo que queda por hacer en un proyecto en el

tiempo. En él se puede ver claramente cómo van descendiendo el número de puntos de historia a realizar a medida que van avanzando el número de *sprints*. Podemos observar como el diagrama nos muestra la re-estimación que hemos tenido que realizar al finalizar el *sprint* número 5, esto ha implicado el añadir un *sprint* más.

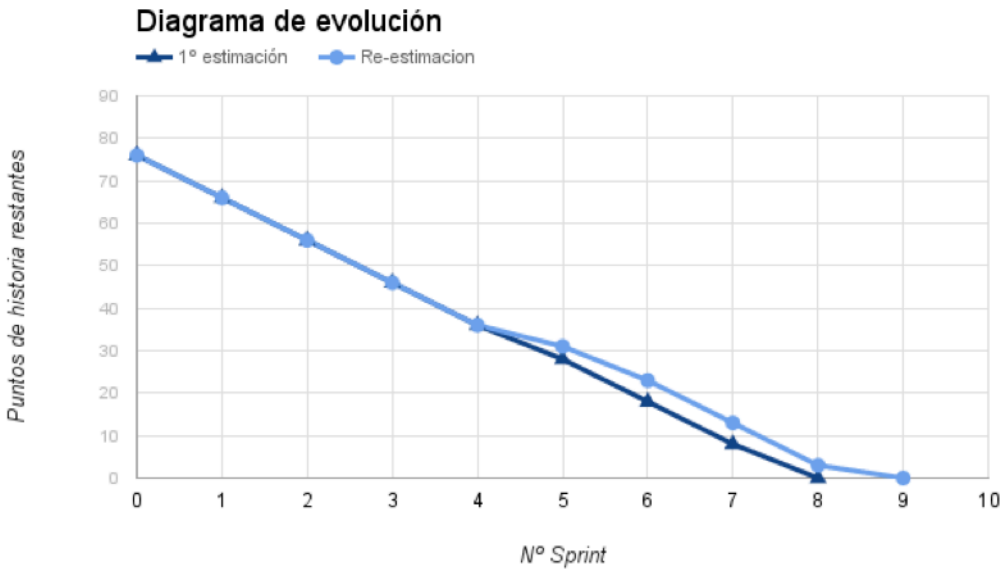

Ilustración 2.9 - Diagrama de evolución

A pesar de los cambios ocurridos respecto a la planificación podemos concluir que el retraso incluido no ha sido excesivo, y se ha podido llegar sin problema a las fechas marcadas con todas las historias de usuario completas.

2.6 Estudio viabilidad

En este apartado vamos a analizar los costes que va a acarrear el desarrollo de nuestro proyecto.

HERRAMIENTA SOFTWARE	Precio	Duración del proyecto	Precio ponderado
Paquete Microsoft Offfice 365 Hogar	10,00 €/mes	6 meses	60,00€
Chrome v50.0.2661	Free	6 meses	-
Netbeans 8.0.1	Free	6 meses	-
Visual Paradimg Comunity Edition	Free	6 meses	-
Xampp 5.6.12	Free	6 meses	-
SublimeText 2	Free	6 meses	-
Avast Mac Security 2015	Free	6 meses	-
Apache Tomcat 8.0.3	Free	6 meses	-

Tabla 2.7 - Gastos en herramientas software

La tabla 2.7 representa todos los gastos relacionados con productos software estos gastos no son demasiado elevados debido a que casi todos los programas utilizados poseen licencias gratuitas.

EQUIPOS INFORMÁTICOS	Precio	Vida estimada	Precio ponderado
MacBook Pro 15'' 2012	1823 €	7 años	130,21€

Tabla 2.8 - Gastos en herramientas hardware

En la tabla anterior observamos los gastos ocasionados por el uso de equipos informáticos. Estos equipos tienen una vida útil determinada, por eso, se han cargado a los gastos del proyecto el precio ponderado por el uso de este equipo durante el tiempo que ha durado el proyecto.

SUELDOS	Salario/año	Salario/mes	Salario/hora	Días trabajo	Horas día	Total horas	Precio ponderado
Analista	26.088,30 €	2.174,02€	13,58€	43	8	344	4.671,52€
Programador	20.941,34 €	1.745,11€	10,90€	82	8	664	7.237,60€

Tabla 2.9 - Gastos en nóminas de trabajadores

Para el desarrollo de este proyecto necesitamos a dos tipos de profesionales distintos, un analista y un programador. Según la carga de trabajo que hemos asignado a cada rol en el apartado 2.5.1 de esta memoria, se ha realizado una estimación de las retribuciones que recibirán estos profesionales por realizar su trabajo. El salario/mes ha sido obtenido del BOE según el convenio colectivo de Telefónica en el siguiente enlace:
https://www.boe.es/boe/dias/2016/01/21/pdfs/BOE-A-2016-541.pdf

GASTOS	PRECIO
Gasto en software	60,00€
Gasto en hardware	130,21€
Gasto en nóminas	11.909,12 €
TOTAL GASTOS	12.099,33€

Tabla 2.10 - Gastos totales del proyecto

COSTE	PRECIO
Gastos	12.099,33€
Beneficios(15% de los gastos)	1.814,89€
TOTAL COSTE	**13.914,22€**

Tabla 2.11 - Coste final del proyecto

Para calcular el coste total del proyecto se ha añadido un incremento de un 15% sobre el total de gastos del proyecto. Este 15% es el beneficio que obtiene la empresa encargada de firmar el contrato.

2.7 Modelo de dominio

Una vez que hemos terminado el análisis del problema es necesario pasar a diseñar el sistema, pero antes proponemos una solución de diseño preliminar que será la que nos determinará el diseño final. Se trata de un diagrama de clases conceptuales [31] que está creado desde el punto de vista del análisis y no desde el punto de vista del diseño. Es por esto que algunas relaciones podrán ser modificadas o algunos elementos o entidades podrán ser eliminados.

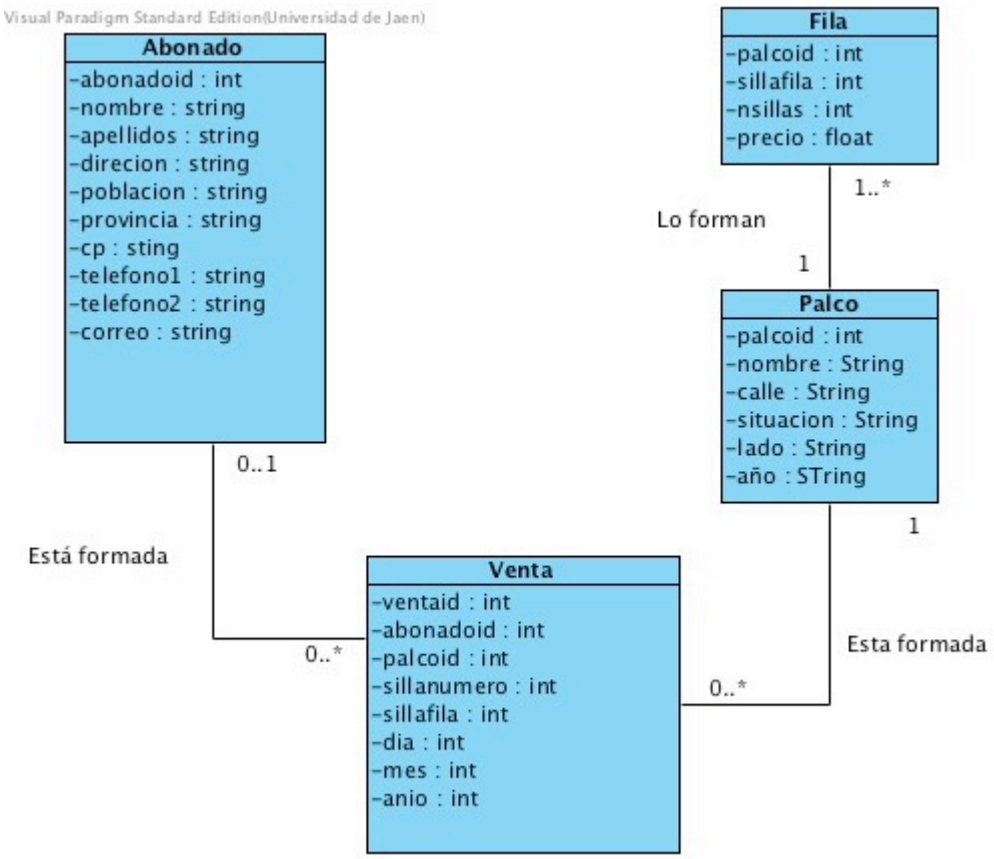

Ilustración 2.10 - Modelo de dominio

Como podemos ver en la ilustración 2.10 la clase conceptual venta es la que relaciona un abonado con el palco al que pertenece. En la clase venta se almacena tanto la fila que ocupa la silla tanto el número que tiene la silla en la fila.

3 DISEÑO

Una vez finalizada la fase de análisis de nuestro proyecto, comenzamos la fase del diseño de los objetos. Las clases de análisis reflejadas en el modelo de dominio, son el origen de las clases de diseño. No hay que perder de vista que el diseño de objetos es un proceso iterativo, que se va refinando a medida que avanza el proyecto, pero por motivos didácticos en esta memoria se desarrolla de forma secuencial.

3.1 Diagrama de clases

Dado que en nuestro proyecto estamos utilizando el patrón arquitectónico Modelo Vista Controlador, los tres principales componentes de este patrón quedarán claramente diferenciados. El modelo vista controlador se explica con más detenimiento en el apartado 4.1.2 de esta memoria.

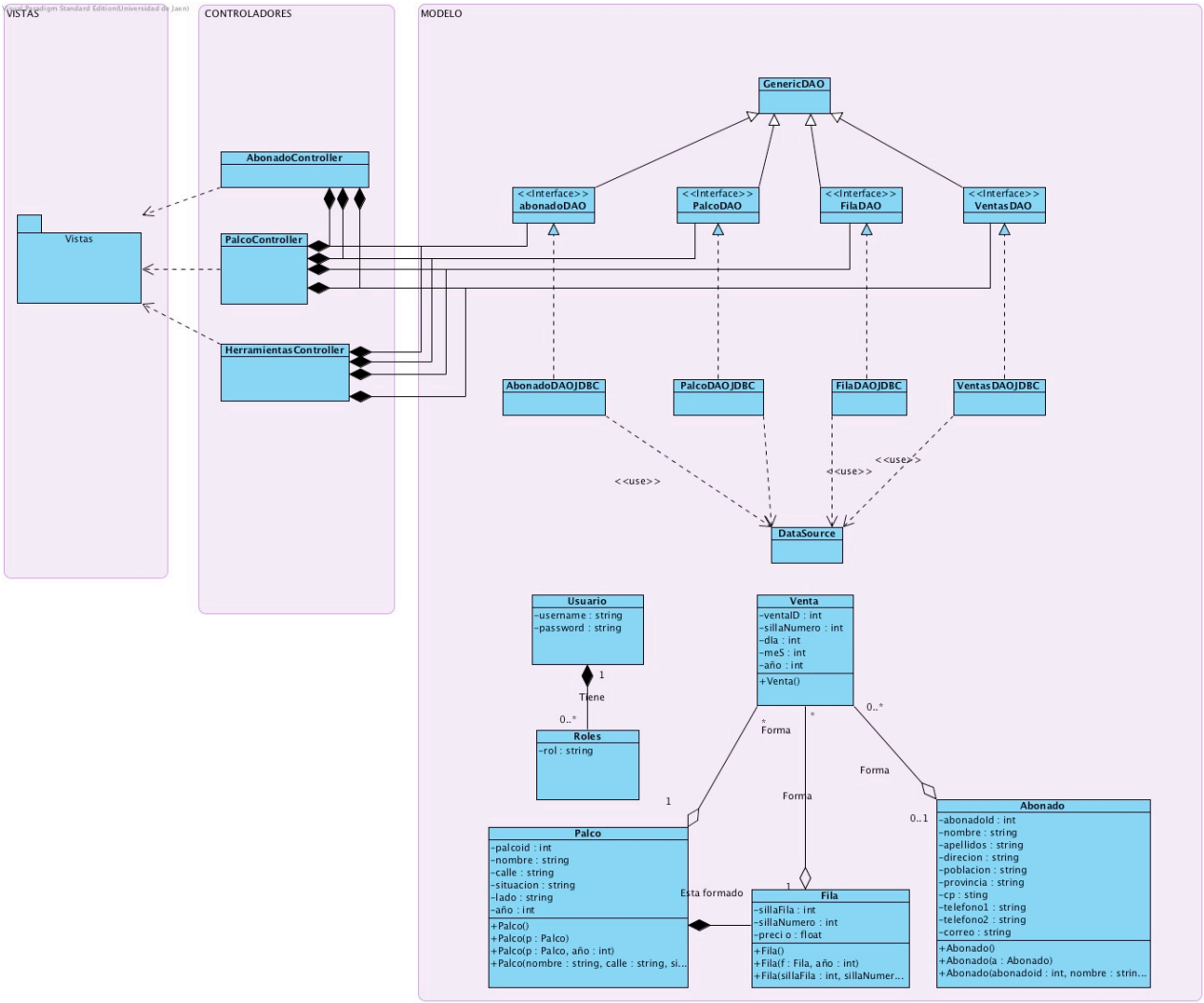

Ilustración 3.1 - Diagrama de clases.

Nuestra aplicación necesita una interfaz para poder comunicarse con la base de datos, por ello que utilizaremos el patrón DAO para suministrar esta interfaz. Es el encargado de relacionar la lógica de negocio con la capa de persistencia. Por cada tabla de una base de datos relacional existirá un DAO [32].

Respecto a los DAO´s, hemos implementado una interfaz de acceso a cada uno de estos para que exista una independencia entre el DAO y el sistema de almacenamiento utilizado. Además también tenemos una interfaz genérica (*GenericDAO*) para homogeneizar todos los DAO con operaciones básicas.

Con objeto de no sobrecargar el diagrama con relaciones que no nos aportan demasiada información se han obviado algunas de estas, como por ejemplo, las dependencias que existen entre las clases del modelo y los objetos DAO´s y los controladores. Como es lógico debido a los nombres de los controladores y de los DAO´s se puede sobrentender que estos utilizan las clases del modelo que coinciden con su nombre.

En el diagrama tampoco se han considerado a las vistas como clases propiamente dichas, aunque una página JSP [33] se transforma en servlet la primera vez que se accede a ella. Nuestra aplicación cuenta con un total 17 páginas JSP.

3.2 Diagramas de secuencia

A continuación se muestran los principales diagramas de secuencia que representan las principales funcionalidades de nuestro proyecto y que ayudarán a comprender la interacción de los diferentes componentes del diagrama de clases.

Crear abonado

En este diagrama se muestra la forma en la que se crea un nuevo abonado. El controlador carga la vista con el formulario para introducir los datos del abonado, y una vez que el usuario pulsa el botón de enviar en la vista, el controlador manda el nuevo objeto al método del DAO encargado de almacenar ese objeto en la base de datos.

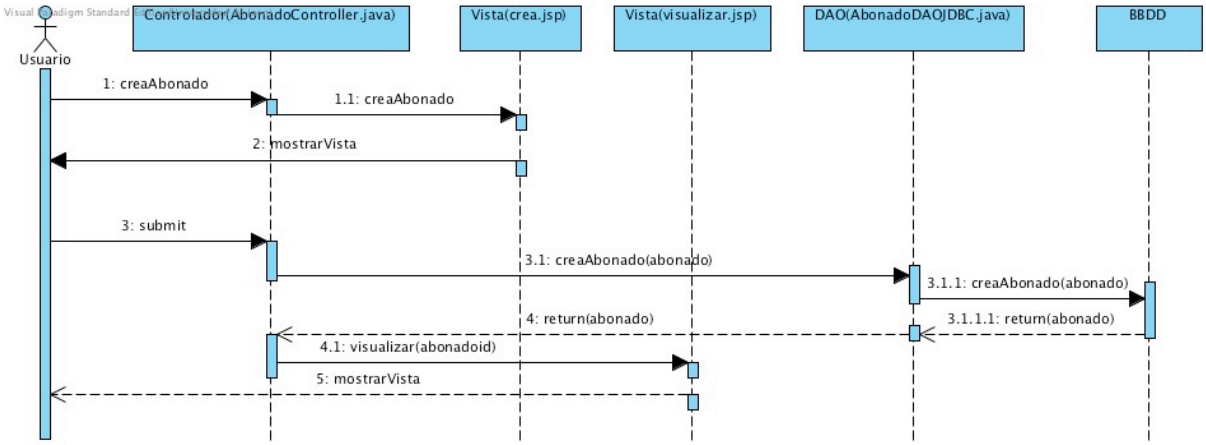

Ilustración 3.2 - Diagrama de secuencia. Crear abonado.

Modificar abonado

El proceso de modificar un abonado similar al proceso de crearlo. La única diferencia es que antes de que el controlador muestre la vista con el formulario para editar los datos, este tiene llamar al método correspondiente del DAO que nos devuelve el objeto abonado que queremos editar.

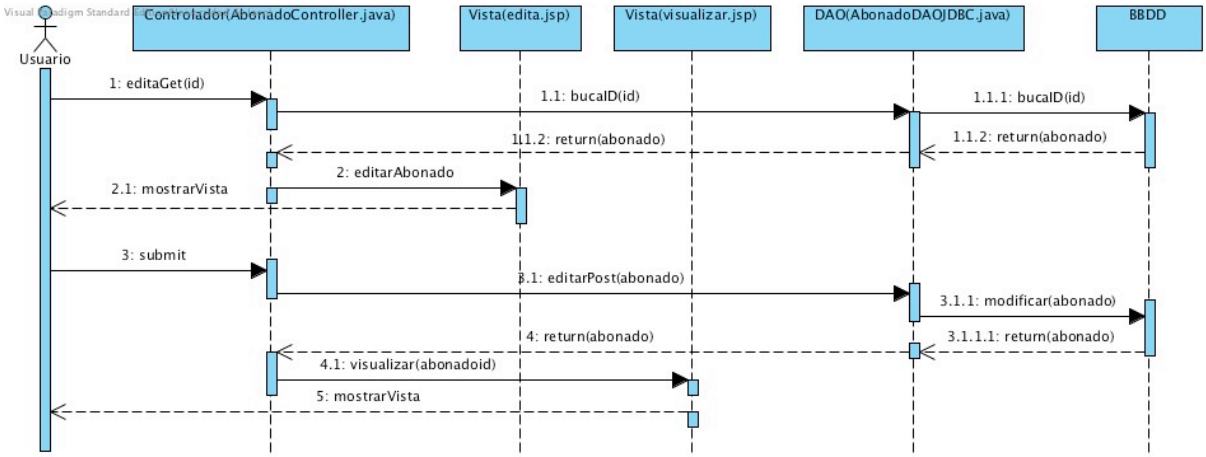

Ilustración 3.3 - Diagrama de secuencia. Editar abonado.

Eliminar palco

Para eliminar del sistema un palco tenemos que llamar al método adecuado del controlador indicando el id del palco a borrar en cuestión. El controlador será el encargado de llamar a los métodos adecuados de los DAO's para eliminar de la base de datos las ventas y filas asociadas a este palco, así como el mismo palco en sí.

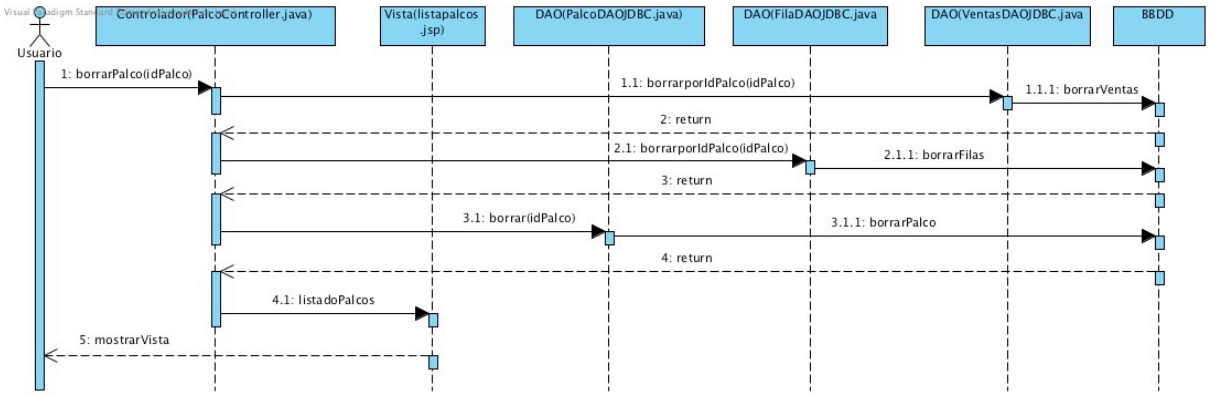

Ilustración 3.4 - Diagrama de secuencia. Eliminar palco.

Venta de sillas

Este proceso es el que da sentido a la aplicación, ya que es la que relaciona un abonado y un palco, dando lugar una venta.

Su funcionamiento es el siguiente: llegan al controlador una lista con los *idventas* que se van a comprar. El controlador tras realizar distintas operaciones nos muestra una vista en la que aparecen las sillas seleccionadas por el usuario y un formulario para realizar las búsqueda del abonado que va a realizar la compra. Cuando el usuario rellena el formulario y presiona el botón de enviar el controlador, el controlador llama al método adecuado de *AbonadoController* que nos devuelve una lista de abonados que su nombre y apellidos coinciden con la cadena que ha introducido el usuario. El controlador nos envía a la misma vista que estábamos antes pero esta cargará el listado de abonados que coinciden con nuestra búsqueda. Tras seleccionar el abonado al que queremos asignarle las sillas, el controlador actualiza los datos de los objetos ventas con el id del abonado en cuestión y llamará al método vender() de *VentasDAO* tantas veces como ventas a modificar tengamos.

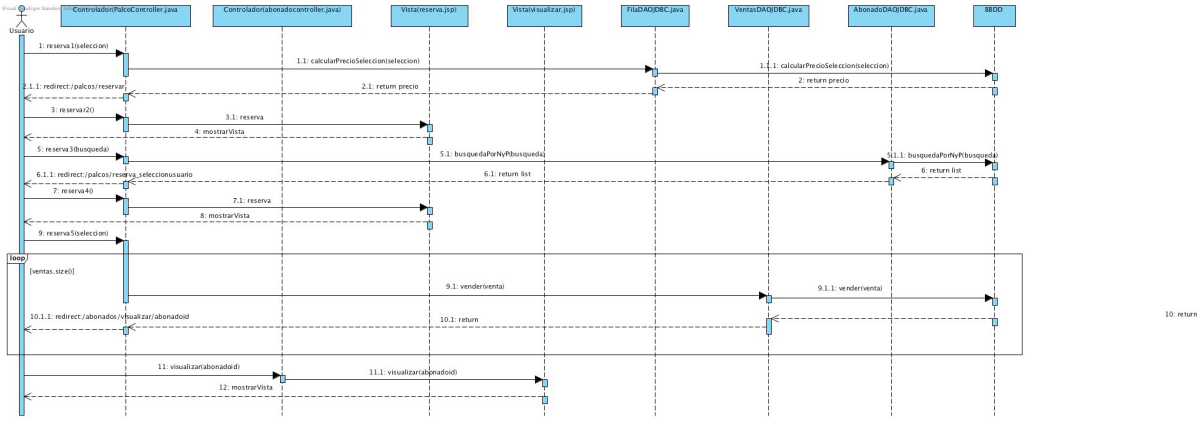

Ilustración 3.5 - Diagrama de secuencia. Venta de sillas.

Cambio de año

Esta funcionalidad prepara la aplicación para poder empezar a trabajar un nuevo año, para poder empezar a trabajar, al llamar al método adecuado del controlador este nos manda la vista *cambioanio* que contiene un formulario en el que introduciremos el nuevo año en el que queremos empezar a trabajar. Al presionar el boton del formulario, llamaremos al método *cambioAnioPost()* del controlador pasándole como parámetro el año que hemos introducido en el controlador. El controlador consultará a *PalcoDAO* si existen en la base de datos palcos del año anterior al que hemos introducido. Si existen, el controlador será el encargado de llamar a los distintos DAO´s para realizar una copia de todos los palcos del año anterior, asignándole el año nuevo que hemos introducido. Las ventas pertenecientes a estos nuevos palcos no están asignados a ningún abonado, es decir libres.

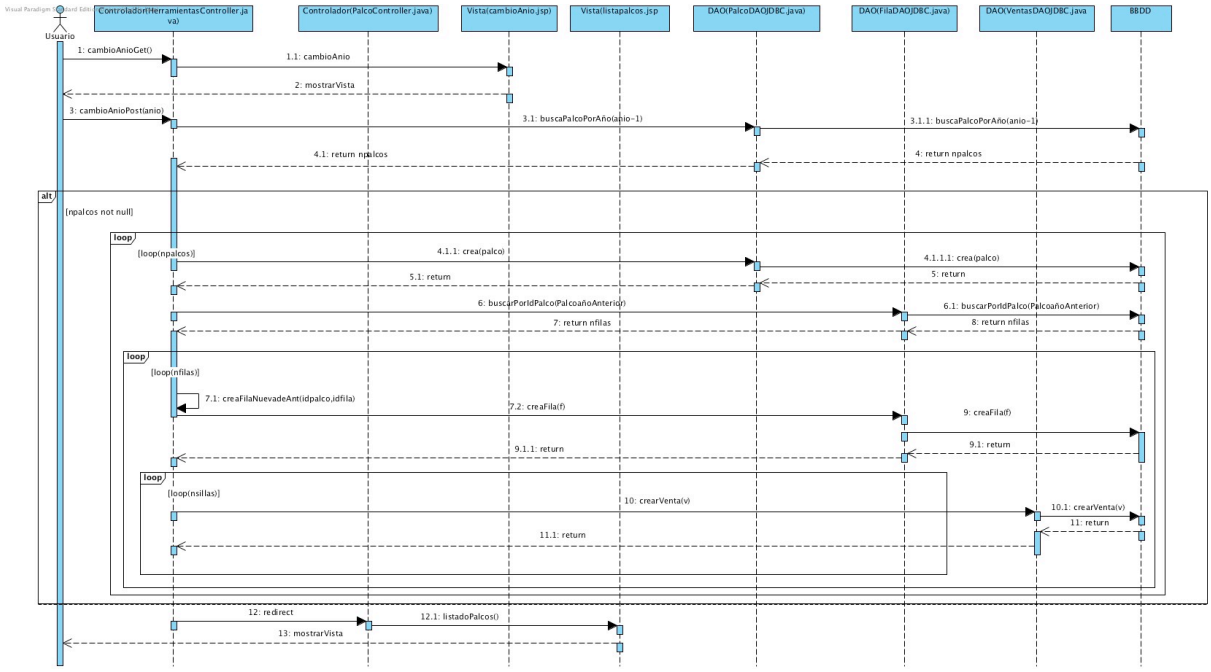

Ilustración 3.6 - Diagrama de secuencia. Cambio de año.

3.3 Modelo entidad relación

Algunos objetos que se utilizan en una aplicación informática son objetos transitorios, son objetos que existen en memoria y se desechan cuando la aplicación termina.

Sin embargo, otros objetos deben mantenerse de una ejecución a otra de la aplicación, estos objetos se denominan objetos persistentes.
En nuestro caso, estos objetos persistentes los almacenaremos en una base de datos relacional. Las bases de datos relacionales almacenan los datos en tablas. Las tablas constan de filas de datos organizados en columnas.

3.3.1 Tablas

A continuación se muestran las tablas que componen la base de datos de nuestra aplicación.

TABLA: *usuarios*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
usuario	Cadena	Primaria	Identificador de usuario
clave	Cadena	-	Contraseña de acceso

Tabla 3.1 - Base de datos. Tabla usuarios.

TABLA: *roles*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
usuario	Cadena	Foránea	Identificador de usuario
rol	Cadena	-	Tipo de usuario. Indica sus privilegios.

Tabla 3.2 - Base de datos. Tabla roles.

TABLA: *abonado*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
abonadoid	Entero	Primaria	Identificador del abonado
nombre	Cadena	-	Nombre de pila del abonado.
apellidos	Cadena	-	Primer y segundo apellido del abonado.
dirección	Cadena	-	Calle y número de la vivienda del abonado.
población	Cadena	-	Localidad de residencia del abonado.
provincia	Cadena	-	Provincia de residencia del abonado.
cp	Cadena	-	Código postal de la localidad.
telefono1	Cadena	-	Teléfono de contacto número uno.
telefono2	Cadena	-	Teléfono de contacto número dos.
correo	Cadena	-	Correo electrónico de contacto.

Tabla 3.3 - Base de datos. Tabla abonado.

TABLA: *palco*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
palcoid	Entero	Primaria	Identificador del palco
nombre	Cadena	-	Nombre del palco.
calle	Cadena	-	Nombre de la calle en la que se encuentra el palco.
situacion	Cadena	-	Altura de la calle en la que se encuentra el palco.
lado	Cadena	-	Lado de la calle en el que se encuentra el palco.
año	Cadena	-	Año al que pertenece el palco.

Tabla 3.4 - Base de datos. Tabla palco.

TABLA: *fila*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
palcoid	Entero	Primaria - Foránea	Identificador del palco
sillafila	Entero	Primaria	Número de la fila
nsillas	Entero	-	Número de sillas de la fila
precio	Real	-	Precio unitario

Tabla 3.5 - Base de datos. Tabla fila.

TABLA: *ventas*

CAMPO	TIPO	CLAVE	DESCRIPCIÓN
ventaid	Entero	Primaria	Identificador del palco
abonadoid	Entero	Foránea	Identificador del abonado
palcoid	Entero	Foránea	Identificador del palco
sillafila	Entero	-	Fila en la que se encuentra la silla.
sillanumero	Entero	-	Número de la sillas dentro de la fila
día	Entero	-	Día en el que se realiza la venta.
mes	Entero	-	Mes en el que se realiza la venta.
año	Entero	-	Año en el que se realiza la venta.

Tabla 3.6 - Base de datos. Tabla ventas.

3.3.2 Diagrama Entidad - Relación

Una vez que hemos visto la estructura que tienen los datos de nuestra aplicación vamos a ver de qué manera se relacionan entre sí. Para ello utilizaremos un diagrama denominado modelo Entidad-Relación(E-R).

El modelo E-R, se basa en la percepción del mundo real a través de unos objetos básicos a los que llamamos entidades, y de relaciones entre esos objetos. El modelo entidad relación que hemos diseñado para nuestra aplicación se encuentra normalizado, es decir, en tercera forma normal.

Existen tres niveles en cuanto a la normalización de una base de datos [34], son los siguientes:

- Primera forma normal (1FN): Una tabla está en 1FN si todos los atributos que no son clave primaria dependen funcionalmente de esta. Es decir, no existen grupos repetitivos para un valor clave.
- Segunda forma normal(2FN): Una tabla está en 2FN si y sólo si está en 1FN y cada uno de sus atributos no clave dependen funcionalmente de forma completa de la clave primaria.
- Tercera forma normal(3FN): Una tabla está en 3FN si y sólo si está en 2FN y cada atributo depende solo de la clave principal y no de ningún otro atributo no clave.

Esta normalización evita grandes problemas que pueden ser ocasionados por un mal diseño de la base de datos, como son: información duplicada, pérdida de información en el proceso de diseño de la base de datos o imposibilidad de representar algún tipo de información.

El resultado de nuestro diagrama entidad relación es el siguiente:

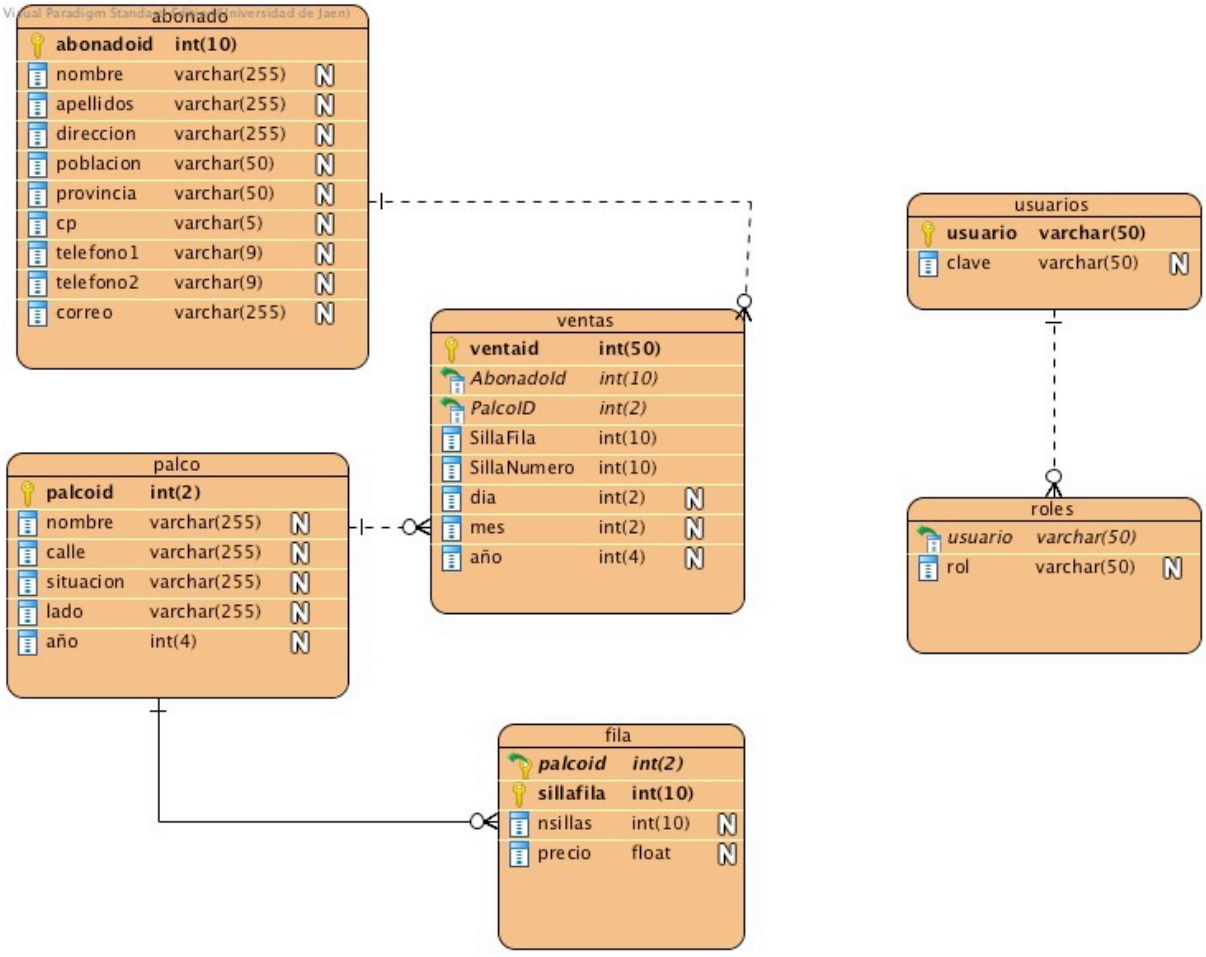

Ilustración 3.7 - Diagrama entidad relación.

Todas las tablas a excepción de roles disponen de una clave primaria, por ello no se podrán almacenar tuplas que tengan la misma clave primaria en una tabla. Los atributos *abonadoid* y *palcoid* son claves foráneas [35] en la tabla ventas, la clave foránea identifica una columna o grupo de columnas en una tabla que se refiere a una columna o grupo de columnas en otra tabla. Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada. El atributo *palcoid* también es una clave foránea de la tabla fila.

Los campos *abonadoid* de la tabla abonado, *palcoid* de la tabla palco y *ventaid* de la tabla ventas se calculan de forma automática al insertar un objeto nuevo en la base de datos.

3.4 Diseño de la interfaz

En este apartado se realiza un boceto del aspecto que tendrá la interfaz gráfica que representaremos en nuestra aplicación web. Se trata de un proceso muy importante dado que será la parte de la aplicación con la que va a interactuar el usuario. Esta interfaz debe carecer de complejidad y ser muy intuitiva.

La herramienta elegida para la realización del prototipo es Axure RP [36]. Axure RP es una aplicación de escritorio que permite diseñar *wireframes* y prototipos básicos de forma fácil, principalmente dirigido a aplicaciones web y de escritorio, aunque también permite el diseño de prototipos para aplicaciones móviles. Axure es una herramienta de pago, pero que permite disfrutar de 30 días de prueba de la aplicación, en este periodo de prueba gratuito se ha realizado el prototipo de la aplicación.

Al tratarse nuestra aplicación de un proyecto real, el prototipo realizado fue presentado en una reunión al cliente, la Agrupación de Cofradías de Jaén. El prototipo fue realizado según las indicaciones que el cliente indicó en reuniones anteriores, entre las que podemos destacar:

- La sencillez. Fue una de las cuestiones a la que más importancia se le dio.
- La visualización de las sillas del palco se realizará en forma de matriz. Las sillas libres se representarán en color verde y las sillas ocupadas en color rojo.

A primera vista causó una gran impresión por la vistosidad del mismo pero surgieron dudas sobre cómo utilizarlo y sobre todo cómo se iba a realizar el proceso de venta de abonos. Tras explicarles el flujo de trabajo por el que se procedería a la reserva se realizaron algunas preguntas más sobre el mismo que no tienen relevancia como para reflejarlas en esta memoria.

Procedemos a analizar las vistas más representativas del prototipo:

Pantalla de identificación de usuarios.

Ilustración 3.8 - Interfaz. Identificación de usuario.

Esta será la primera que se encontrará el usuario al entrar a la aplicación. En ella deberá introducir su nombre de usuario y su contraseña para identificarse.

Pantalla de inicio

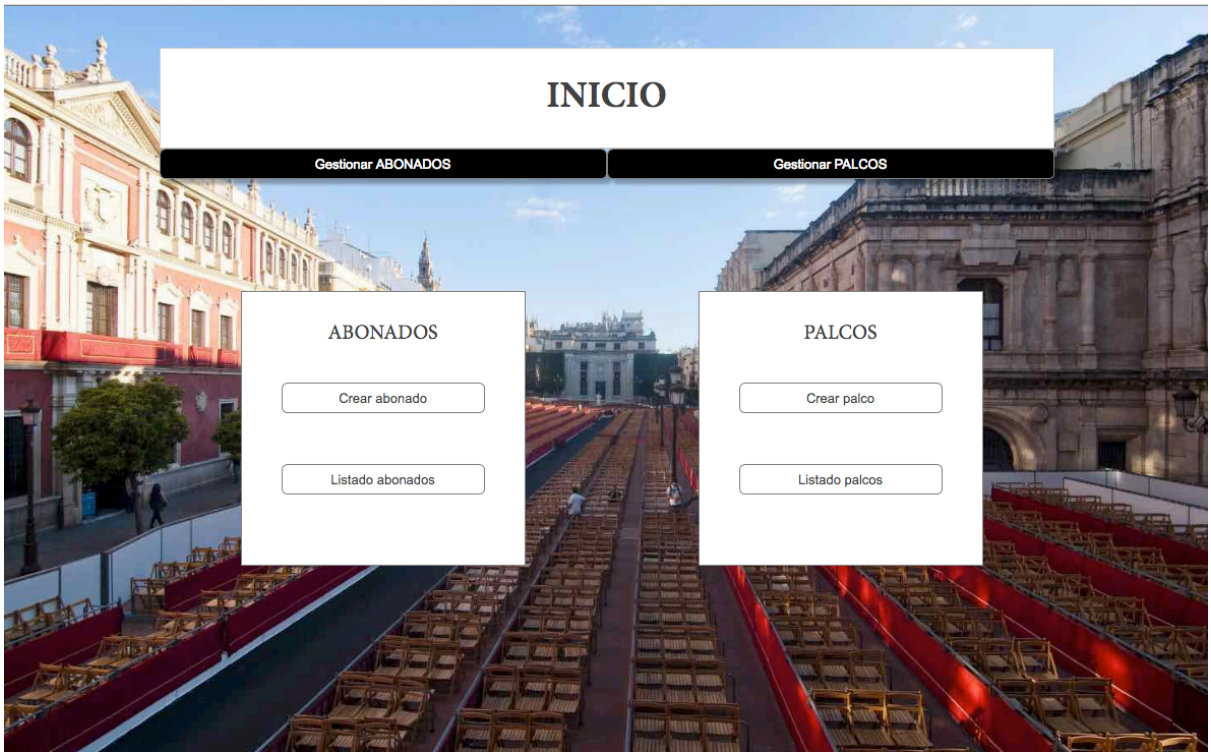

Ilustración 3.9 - Interfaz. Inicio.

En esta pantalla observamos tres secciones claramente diferenciadas:

- **Barra superior**: se encuentra en la parte superior de la pantalla y aparecerá en todas las demás vistas de la aplicación. En ella se observa el nombre de la vista en la que nos encontramos actualmente y dos enlaces, uno al apartado de gestionar abonados y otro al apartado de gestionar palcos.
- **Gestión de abonados**: aparece en la parte izquierda de la pantalla, y nos permite acceder a las vistas de “Crear abonado” y “Listado de abonados”.
- **Gestión de palcos**: aparece en la parte derecha de la pantalla, y nos permite acceder a las vistas de “Crear palco” y “Listado de palcos”.

Pantalla para dar de alta un abonado

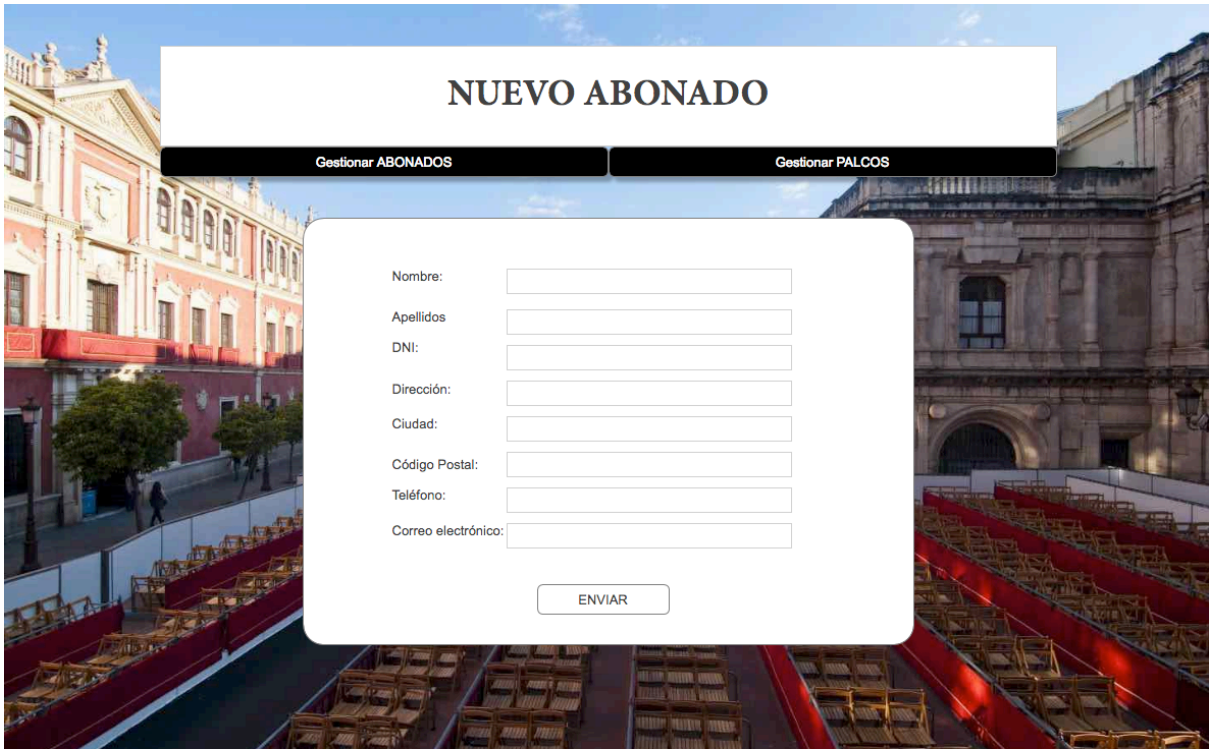

Ilustración 3.10 - Interfaz. Alta de abonado

En esta pantalla aparece un formulario en el que se podrán introducir los datos requeridos para poder crear un nuevo abonado.

Visualizar datos de un abonado

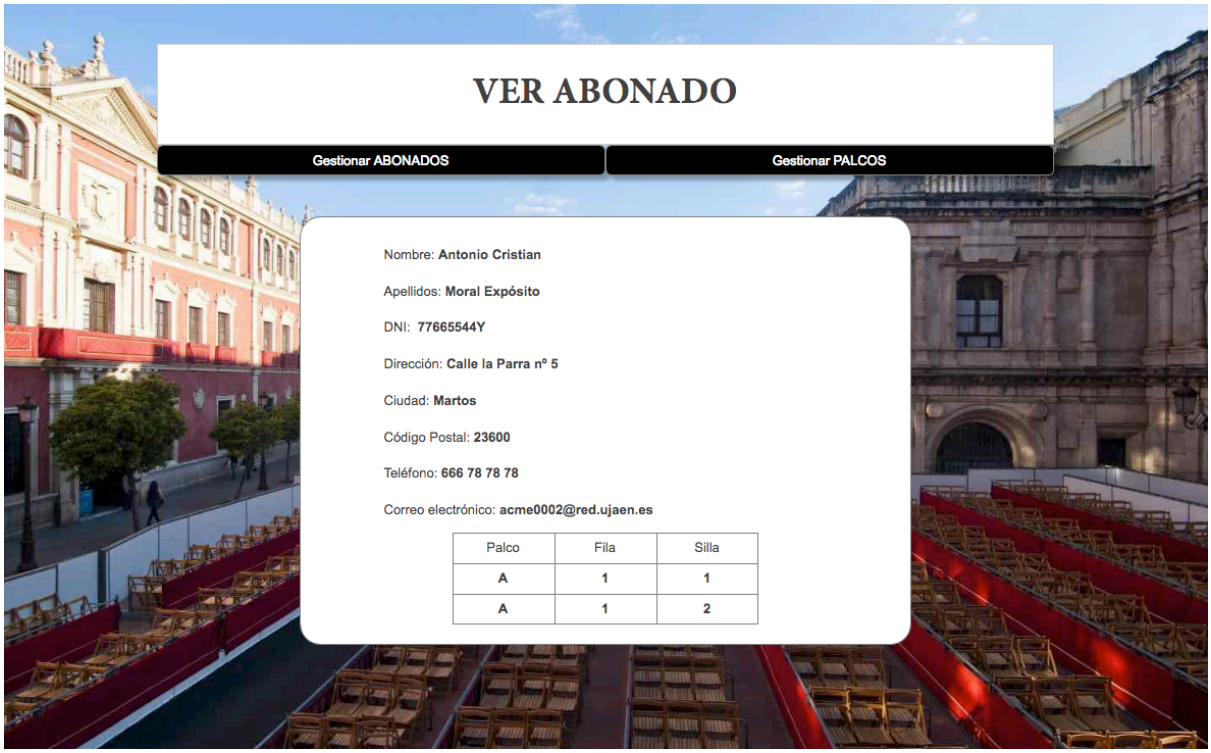

Ilustración 3.11 - Interfaz. Visualizar datos de abonado.

Pantalla que muestra una ficha con los datos personales de un abonado, además también aparecerán las sillas que ese abonado tiene adquiridas para la carrera oficial.

Listado de los palcos

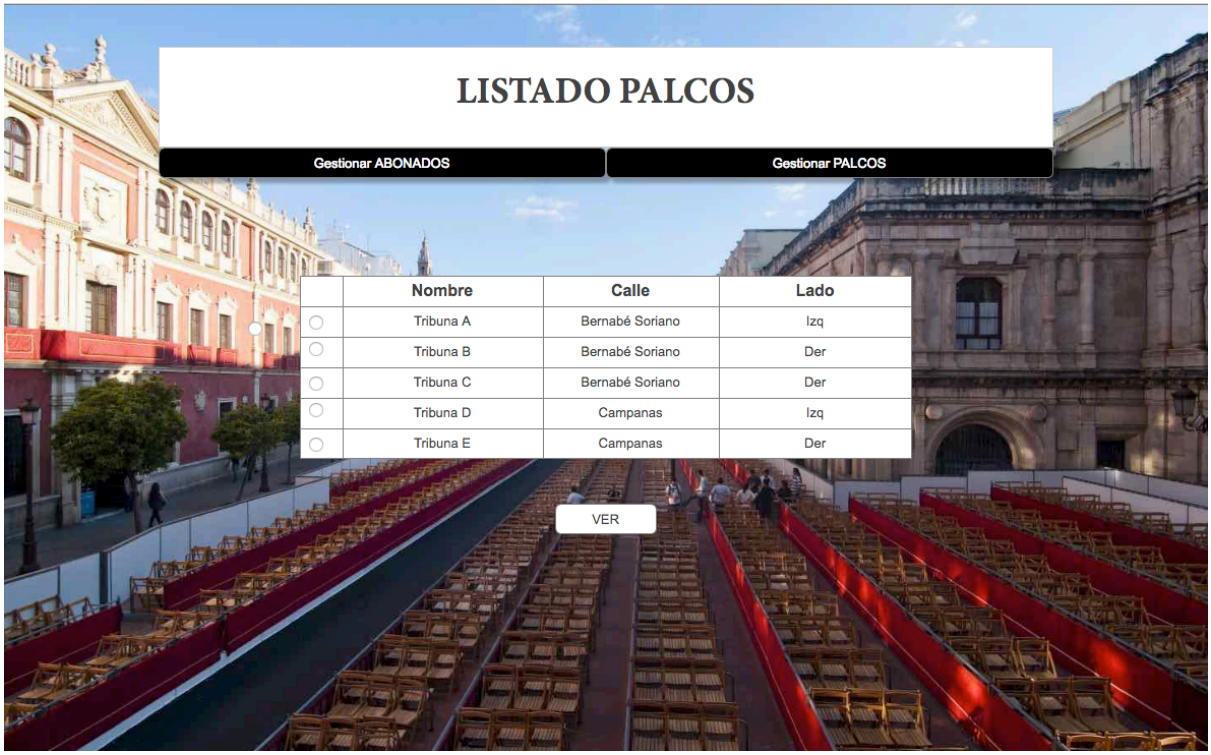

Ilustración 3.12 - Interfaz. Listado de palcos

Como se puede observar en el centro de la pantalla aparece una tabla que nos muestra todos los palcos registrados en el sistema. En esta tabla aparecen los principales datos para situar un palco. Si seleccionamos un *chekbox* y pulsamos sobre el botón enviar podremos visualizar la ficha del palco.

Visualizar datos de un palco

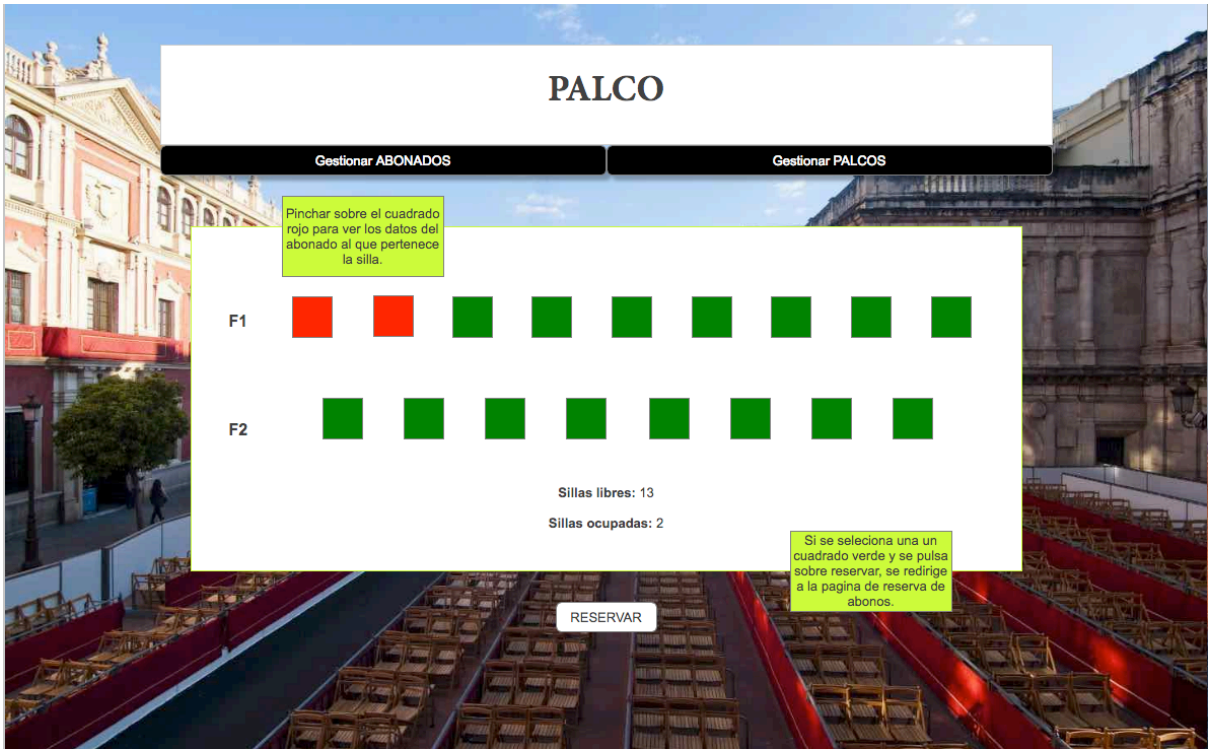

Ilustración 3.13 - Interfaz. Visualizar datos de palco.

Esta pantalla es la más importante de la aplicación. En ella aparecen las filas y las sillas de las que dispone un palco. Las sillas se representan en dos colores distintos:

- Verde: silla que no está ocupada.
- Rojo: silla que está ocupada.

Si pulsamos sobre una silla que está ocupada el sistema nos enviará a la ficha del abonado que ha comprado la silla.

Si seleccionamos una o mas sillas verdes, y pulsamos sobre el botón de reservar, se iniciará el proceso de venta de las sillas.

Esta decisión fue impuesta por parte del cliente, en la concepción inicial que este tenía sobre el sistema, la forma en la que se comprarían las sillas debía de ser lo más parecido posible a una aplicación de venta de entradas de cine o de teatro.

4 IMPLEMENTACIÓN

En este apartado mostraremos una visión general sobre la arquitectura de la aplicación, en las que se exponen las partes que las componen y las relaciones que hay entre ellas.

Posteriormente se analizarán algunos detalles importantes sobre la implementación de nuestra aplicación, para comprender su funcionamiento y la interacción de los elementos constituyentes.

4.1 Arquitectura

En este apartado vamos a tratar la arquitectura de nuestra aplicación, esta incluye dos arquitecturas totalmente diferenciadas.

- **Cliente-servidor**: se trata de la arquitectura que ven los clientes, es la arquitectura tradicional de una aplicación web.
- **Modelo vista-controlador:** este patrón se aplica en la arquitectura interna del servidor.

4.1.1 Arquitectura cliente servidor.

Esta arquitectura se compone por dos elementos, tal y como indica su nombre, el cliente y el servidor. El cliente es el encargado de hacer peticiones al servidor, y el servidor se encuentra a la espera de recibir peticiones y servirlas.

En el caso de una aplicación web es el cliente, a través del navegador, el que realiza una petición introduciendo una dirección URL, que mediante protocolo HTTP llega al servidor, para este servirle el recurso solicitado.

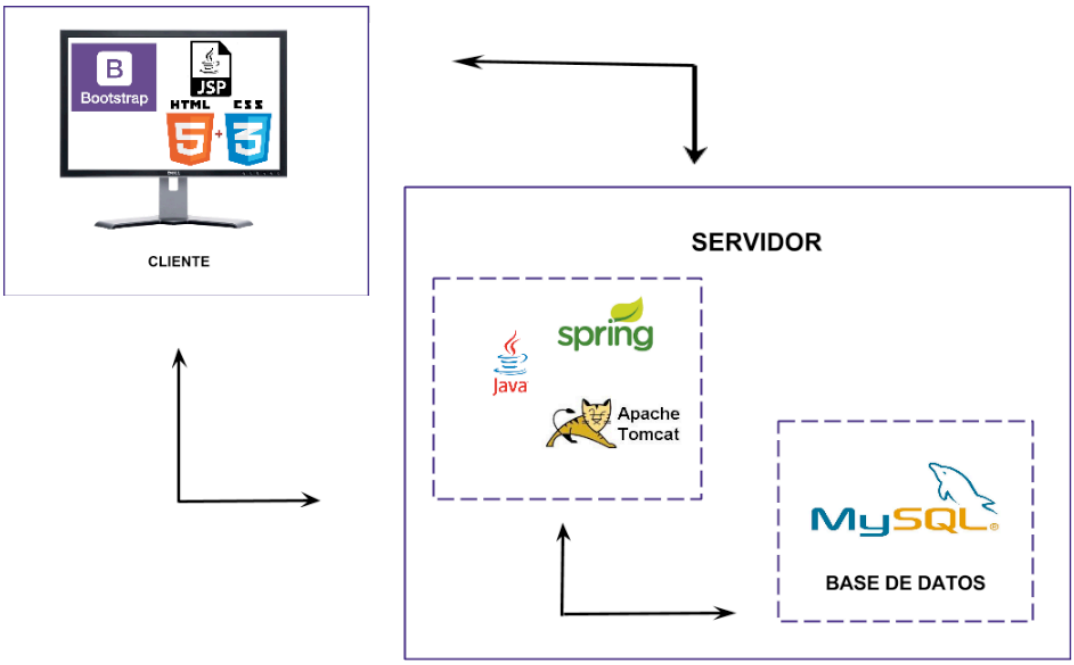

Ilustración 4.1 - Diagrama cliente servidor

En la ilustración anterior se muestra un esquema cliente-servidor con los logos de todas las tecnologías que han intervenido en nuestro proyecto en ambas partes.

4.1.2 Modelo vista controlador (MVC)

El modelo vista controlador es un patrón de arquitectura software que estructura un sistema en tres capas. Las tres capas de este patrón son:

- **Modelo de datos**: representación de los diferentes datos almacenados en el sistema y los mecanismos para acceder a estos datos.
- **Vistas**: representan el conjunto de interfaces con las que interactúa el usuario.
- **Controladores**: son los mecanismos encargados de interactuar entre el modelo de datos y las vistas.

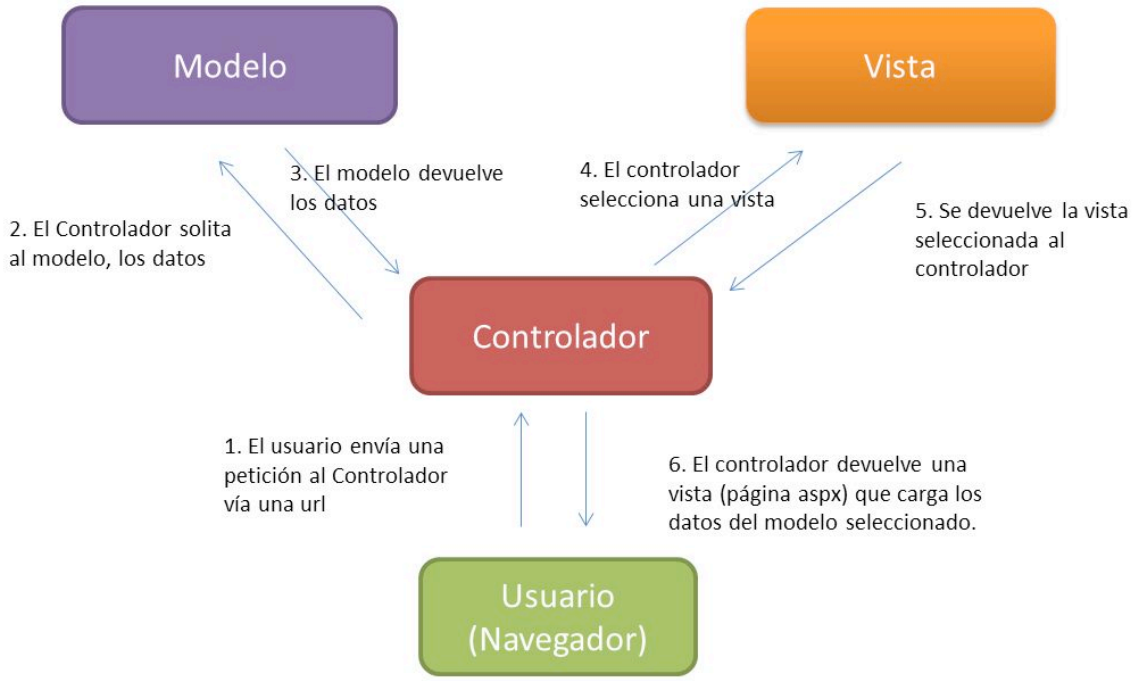

Ilustración 4.2 - Diagrama MVC

En la ilustración 4.2 se explica el funcionamiento básico del modelo vista controlador.

Existen múltiples *frameworks* que utilizan este patrón, como por ejemplo: Ruby on Rails que usa Ruby como lenguaje de programación, Django para Python, o Spring MVC para Java. Este último será el *framework* que utilizaremos para desarrollar nuestra aplicación.

Las vistas de nuestra aplicación han sido desarrolladas con la tecnología JSP [37], además de usar JSTL [38] para procesar los datos que se han recibido del controlador. El controlador está formado por un conjunto de clases java con métodos que implementan las acciones del usuario. El modelo de datos también está formado por clases Java que representan la lógica de negocio. Para acceder a los datos de la base de datos se ha aplicado el patrón DAO, que nos ofrece una interfaz entre el modelo y la base de datos.

Una de las ventajas para utilizar el modelo vista controlador es la flexibilidad que nos ofrece a la hora de realizar cambios, debido a su modularidad de sus tres capas. Ya que este proyecto es un proyecto real, y está sometido a las necesidades del cliente puede sufrir cambios en los requisitos, por lo tanto se trata de una arquitectura idónea.

La naturaleza del protocolo HTTP nos obliga a hacer algunas adaptaciones sobre el patrón MVC original. Por eso surge Model 2 [39], que es la adaptación del

patrón MVC a la tecnología *JavaServer Pages.* Este enfoque híbrido aprovecha los puntos fuertes tanto de los *Servlet* como de JSP ya que JSP se utiliza para generar las vistas, y los servlet se destinan para las tareas de procesado de datos.

4.2 Detalles sobre la implementación

En este apartado se presentan algunos detalles y ejemplos de uso extraídos del código de la aplicación para conocer los detalles de funcionamiento de las tecnologías utilizadas. Son los siguientes:

4.2.1 Spring MVC

Como ya indicamos en el apartado 2.2 de esta memoria, para facilitar la implementación del modelo vista controlador vamos a utilizar el *framework* Spring MVC [40]. Una de las ventajas que nos proporciona SpringMVC es que una vez conocida esta tecnología, el proceso de desarrollo es mucho más rápido. También y gracias al modelo MVC permite ordenar el código en grupos como servicios o repositorios. Otras de las facilidades que nos ofrece Spring MVC es el principio de programación IoC [41], que permite pasar el control del proceso de interconexión de objetos de la fase de compilación a la fase de ejecución. La inyección de dependencias es la principal técnica del principio IoC.

El flujo de procesamiento de una petición HTTP es el siguiente:

- Todas las peticiones HTTP llegan al controlador principal, o también llamado *front controller*. En el caso particular de Spring nuestro *front controll*er es la clase *DispatcherServlet*.

- El *front controller* averigua a partir de la URL a qué *Controller* hay que llamar para servirle la petición.

- El *Controller* determinado obtiene los resultados y los devuelve al *servlet*, encapsulados en un objeto del tipo del modelo en cuestión. También se devuelve el nombre lógico de la vista que se va a mostrar.

- Un *ViewResolver* se encarga de averiguar el nombre físico de la visa.

- Finalmente el *front controller* redirige la petición hacia la vista que muestra los resultados obtenidos.

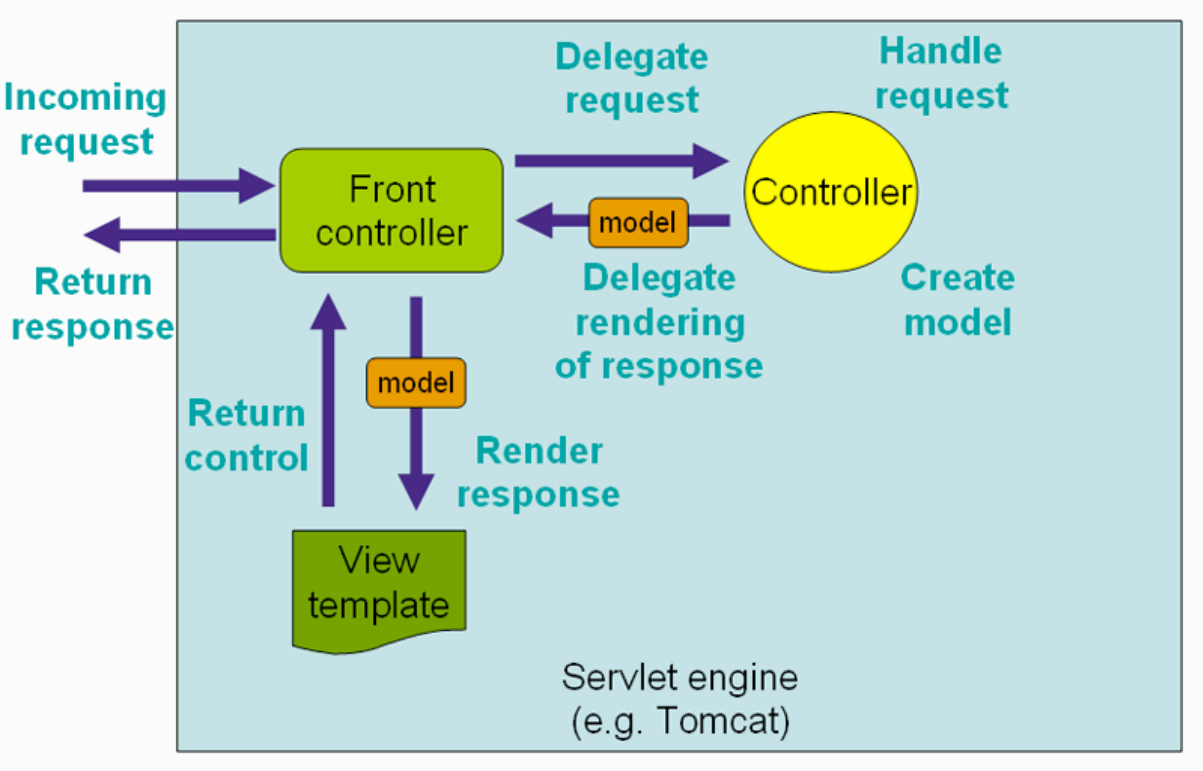

Ilustración 4.3 - Procesamiento de petición de Spring. Fuente: *Spring Framework Reference Documentation*

Una vez conocido el funcionamiento de Spring, ¿como indicamos le indicamos a nuestra aplicación cuál es su *front controller* y cual su *ViewResolver*?

En el fichero /WEB-INF/web.xml indicamos que nuestro *front controller* es *SpringDispatcher* (1)y que tiene que buscar los controladores a partir de las rutas que comienzan por /main (2) tal y como podemos ver en la ilustración 4.4.

```
<servlet>
  1 <servlet-name>SpringDispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  2 <servlet-name>SpringDispatcher</servlet-name>
    <url-pattern>/main/*</url-pattern>
</servlet-mapping>
```

Ilustración 4.4 - Spring. Configuración del front controller.

Después hay que modificar el fichero de configuración del controlador principal de la aplicación web, /WEB-INF/SpringDispatcher-servlet.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <mvc:annotation-driven /> 1
    <context:component-scan base-package="tfg" /> 2

    <bean id="viewResolver" 3
          class="org.springframework.web.servlet.view.InternalResourceViewResolver"
          p:prefix="/WEB-INF/jsp/"
          p:suffix=".jsp" />

</beans>
```

Ilustración 4.5 - Spring. Configuración del ViewResolver.

Según los números que aparecen en la ilustración superior, en la anotación 1 indicamos a Spring que vamos a utilizar anotaciones de mvc, en la anotación 2 se indica la jerarquía de paquetes en la que Spring puede encontrar las clases que estarán implicadas. Posteriormente configuramos el *ViewResolver*, tal y como vemos en la anotación 3, nuestras vistas puede encontrarlas en /WEB-INF/jsp y serán todas las que terminan por .jsp.

4.2.2 Controladores

Hemos considerado que nuestra aplicación esté formada por tres controladores:

- ***AbonadoController***: es el encargado de gestionar la creación, edición, borrado y visualización de los abonados.

- ***PalcoController***: es el gran controlador de la aplicación, se encarga de la visualización de los palcos, además de su edición, borrado, también es el encargado de realizar la venta de sillas entre otras varias utilidades.

- ***HerramientasController***: dado que *PalcoController* era una clase muy extensa, hemos creído conveniente realizar este tercer controlador que se

encargará de gestionar la creación de palcos, así como el sistema de cambio de año de la aplicación.

Para crear un controlador seguiremos la filosofía de Spring de evitar convenciones de la arquitectura, por lo tanto nuestro controlador será una clase Java que añadiremos al paquete *tfg.controller*. Para que el *DispatcherServlet* pueda localizarlas y enviarles peticiones debemos de añadir las anotaciones *@Controller* y *@RequestMapping.*

```
@Controller
@RequestMapping("/abonados")
public class AbonadoController {
```

Ilustración 4.6 - Controlador. Anotación controller

A continuación vamos a ver dos métodos que se encargaran de: preparar y mostrar el formulario para crear un abonado; y de recibir, validar los datos que ha mandado el usuario y si es posible crear el abonado, respectivamente.

Utilizaremos la notación @RequestMapping para indicar la ruta a la que responde nuestra función y también se indicará el tipo de petición HTTP a la que tiene responder. La primera función responde a una petición de tipo GET, por lo tanto solamente prepara el modelo, y nos envía a la vista /abonados/crea.jsp. La segunda función está asociada a una petición de tipo POST. Esta petición recoge los datos enviados por el usuario en el formulario, comprueba que los datos son correctos, abonadoDAO se encarga de almacenar esos datos en la base de datos. Finalmente nos redirige a la url que indica la variable view.

```
@RequestMapping(value = "/crea", method = RequestMethod.GET)
public String creaAbonado(ModelMap model) {
    model.addAttribute("abonado", new Abonado());
    return "abonados/crea";
}

@RequestMapping(value = {"/crea"}, method = RequestMethod.POST)
public String creaAbonado(@ModelAttribute("abonado") @Valid Abonado abonado, BindingResult result, ModelMap model) {

    String view;
    //He omitido la parte de valoracion
    if (!result.hasErrors()) {
        abonadoDAO.crea(abonado);
        Abonado a = abonadoDAO.buscaSinId(abonado);
        view = "redirect:visualizar/";
        model.clear();
        view = view + a.getAbonadoid();
    } else {
        view = "abonados/crea";
    }
    return view;

}
```

Ilustración 4.7 - Controlador. Peticiones.

4.2.3 Conexión y acceso a base de datos

El gestor de bases de datos que vamos a utilizar es MySql [42], dado que es un gestor muy potente y se trata de software libre. Además nos aportará muchas facilidades a la hora de desplegar la aplicación en un *PaaS* [43], plataforma como servicio. Un PaaS nos proporciona una API de servicios en la nube para implementar aplicaciones.

Abrir y cerrar una conexión a una base de datos es un proceso muy costoso y lento, es por ello que para solucionar este problema usaremos un *pool* de conexiones, ya que nos proporciona a los usuario de la base de datos un conjunto de conexiones que se reutilizan continuamente. Esto mejora el consumo de recursos del proceso de conexión y desconexión de la base de datos.

La aplicación conectará con el contenedor de *servlet*, en nuestro caso Tomcat [44], para acceder al *pool* de conexiones mediante un objeto *DataSource*. Nuestro pool de conexiones está definido en el fichero /META-INF/context.xml y es el siguiente:

```
<Resource auth="Container"
          driverClassName="com.mysql.jdbc.Driver"
          maxActive="8"
          name="jdbc/bdtfg"
          password="admin"
          type="javax.sql.DataSource"
          url="jdbc:mysql://localhost:3306/bdtfg"
          username="admin"/>
```

Identificador del pool

Ilustración 4.8 - Pool de conexiones.

En él definimos parámetros como el *driver* de conexión, número máximo de conexiones simultáneas activas o el identificador que tendrá el *pool* de conexiones y que permitirá a la aplicación localizarlo.

Para continuar con el proceso de conexión hay que indicarle a Spring que debe instanciar el objeto *DataSource* asociado al *pool* de conexiones. Así un objeto *DataSource* cuyo nombre JNDI es *java:comp/env/jdbc/bdtfg* se convertirá en un *bean* de Spring y por lo tanto será inyectable. Eso se realizará en el fichero *applicationContext.xml.*

```
<jee:jndi-lookup id="dataSourceBean" jndi-name="java:comp/env/jdbc/bdtfg" />
```

Ilustración 4.9 - Instancia del objeto asociado al pool de conexiones.

Para poder utilizar los DAO´s hay que configurarlos, para ello tenemos que añadir la anotación *@Repository* sobre el nombre de la clase. Dado que vamos a utilizar más de una clase DAO hay que añadir un identificador a la anotación para que Spring no tenga duda del DAO que tiene que instanciar e inyectar en el controlador.

```
@Repository("abonadoDAOJdbc")
public class AbonadoDAOJDBC implements AbonadoDAO {

        @Autowired
        private DataSource ds;
```

Ilustración 4.10 - DAO anotación Repository y Autowired

Por último añadimos la anotación *@Autowired* y *@Qualifer* al atributo que haga referencia al DAO en el controlador para realizar la inyección de dependencias. Ya que nos entramos con más de un DAO, hay que indicar a *@Qualifer* a cual estamos haciendo referencia en cada momento.

```
public class AbonadoController {

    @Autowired
    @Qualifier("abonadoDAOJdbc")
    private AbonadoDAO abonadoDAO;

    @Autowired
    @Qualifier("ventasDAOJdbc")
    private VentasDAO ventasDAO;

    @Autowired
    @Qualifier("palcoDAOJdbc")
    private PalcoDAO palcoDAO;
```

Ilustración 4.11 - Controlador anotación Qualifier y Autowired

4.2.4 Autenticación y autorización de usuarios

Nuestra aplicación solo cuenta con un tipo de usuario, el usuario administrador. El volumen de usuarios de la aplicación no será muy grande. Lo normal será que exista una cuenta de administrador y que los miembros de la agrupación de cofradías conozcan la contraseña.

Para implementar la autenticación de nuestra aplicación definiremos un *realm* donde el Tomcat consultará la información necesaria para autenticar a los usuarios, este *realm* lo definiremos en */META-INF/context.xml*. Ya que nuestro prototipo de aplicación está pensado para entrar en fase de producción utilizaremos tablas de la base de datos para almacenar las credenciales de los usuarios, y los roles a los que estos pertenecen. Para almacenar estas credenciales se crearán dos tablas, una tabla *Usuarios* que almacenará el nombre de usuario y la contraseña, y otra tabla *Roles*, que almacenará el nombre de usuario y el rol.

```
<Realm className="org.apache.catalina.realm.DataSourceRealm"
       dataSourceName="jdbc/bdtfg"  1
       localDataSource="true"
       roleNameCol="rol"
       userCredCol="clave"
  2    userNameCol="usuario"
       userRoleTable="Roles"
       userTable="Usuarios"/>
```

Ilustración 4.12 - Definición Realm

En el realm indicamos el nombre del pool de conexiones para conectar a la base de datos(1), así como los nombres de las tablas y de las columnas en los que se encuentran los datos(2).

Una vez definido el *realm* hay que definir el uso de las credenciales, esto se realizará en el fichero /WEB-INF/web.xml.

```
<security-constraint>
    <display-name>Constraint1</display-name>
    <web-resource-collection>
        <description/>
        <url-pattern>/*</url-pattern> 1
    </web-resource-collection>
    <auth-constraint>
        <role-name>administrador</role-name> 2
    </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>FORM</auth-method> 3
    <form-login-config>
        <form-login-page>/WEB-INF/secure/login.jsp</form-login-page> 4
        <form-error-page>/WEB-INF/secure/login.jsp</form-error-page> 5
    </form-login-config>
</login-config>
```

Ilustración 4.13 - Definición de credenciales

Ya que vamos a utilizar autenticación por formulario (3), tenemos que indicar la página en la que se encuentra el formulario (4), y la página a la que nos reenviará en caso de error (5), en nuestro caso hemos optado en los dos casos por la misma, *login.jsp*, así en caso de producirse un error en la autenticación la aplicación nos mostrará de nuevo el formulario para poder volver a introducir el nombre y la contraseña. Además también tenemos que indicar que solo los usuarios del rol *administrador* (2) pueden acceder a las distintas carpetas de la aplicación (1).

4.2.5 *Bean Validation*

Spring MVC nos permite validar de forma automática los datos que recibe de un formulario. Para ello hay que añadir la etiqueta @*Valid* al parámetro del controlador que representa el *bean* que queremos validar (1).Además tenemos que añadir un parámetro de tipo *BindingResult* para comprobar si han ocurrido errores de validación antes de pasar el mensaje al DAO para su almacenamiento en la base de datos (2).

```
@RequestMapping(value = {"/crea"}, method = RequestMethod.POST)
public String creaAbonado(@ModelAttribute("abonado") @Valid Abonado abonado, BindingResult result, ModelMap model) {
                                                      1                       2
```

Ilustración 4.14 - Controlador anotación Valid

El validador que vamos a utilizar es el nuevo estándar JEE de validadores JSR-303/349 [45]. Para definir las restricciones es necesario el uso de anotaciones en las propiedades del bean a validar. Es el entorno el encargado de verificar que estas se cumplen en tiempo de ejecución.

En el caso de la clase abonado, las propiedades nombre y apellidos no pueden ser nulas. Si el usuario al utilizar un formulario no cumpliera con esta restricción se mostraría el mensaje personalizado que hemos indicado junto a la anotación con la propiedad *message*., tal y como se muestra en la siguiente ilustración.

```
import org.hibernate.validator.constraints.NotEmpty;

/**
 *
 * @author David
 */
public class Abonado {

    private int abonadoid;

    @NotEmpty(message = "El campo nombre no puede ser nulo")
    private String nombre;
    @NotEmpty(message = "El campo apellidos no puede ser nulo")
    private String apellidos;
    private String direccion;
    private String poblacion;
    private String provincia;
    private String cp;
    private String telefono1;
    private String telefono2;
    private String correo;
```

Ilustración 4.15 - Modelo anotación NotEmpty

Para mostrar los mensajes de error que hemos indicado anteriormente es necesario modificar el formulario en la vista. Basta con añadir una etiqueta *<form:error* indicando la propiedad en cuestión.

```
<div class="row">

    <div class="col-md-5">
        <div class="form-group">

        <form:errors path="nombre" cssClass="alert alert-danger"/>
    </div>
</div>

<div class="col-md-7">
    <div class="form-group">

        <form:errors path="apellidos" cssClass="alert alert-danger"/>
    </div>

</div>
</div>
```

Ilustración 4.16 - Mensaje de error en vistas

5 CONCLUSIONES

El desarrollo de este proyecto ha supuesto para mi un gran reto personal. Nunca en mis años de estudiante me había enfrentado a tener que desarrollar una aplicación para un cliente real, en este caso la Agrupación de Cofradías de Jaén. Esto ha supuesto la necesidad de tener que entrevistarme con ellos en más de una ocasión, tanto para la adquisición de requisitos, como para la entrega de las versiones tras la finalización de los distintos *sprints*. Trabajar bajo la presión de tener que cumplir unas historias de usuario para una fecha determinada me ha aportado una gran experiencia que seguro que podré poner a prueba en el mercado laboral.

Hemos conseguido cumplir los 3 objetivos que planteamos en el punto 1.2 de esta memoria. La solución informática que se ha desarrollado ha conseguido solucionar el problema que nos planteó la Agrupación de Cofradías de Jaén, la venta errónea de sillas que no existían en la disposición real de los palcos de la carrera oficial. En esta pasada Semana Santa no se recibió ninguna queja de ningún abonado con respecto a este tema, el funcionamiento de la carrera oficial transcurrió con total normalidad una gran satisfacción en todos los miembros de este colectivo.

Se ha implementado una interfaz sencilla y una aplicación fácil de utilizar, debido a que las personas que van a utilizar nuestra aplicación, en principio, son personas de una mediana edad, que no tienen demasiados conocimientos de informática, y que tendrán un proceso lento de adaptación a la aplicación.

En cuanto a la metodología utilizada, Scrum, me parece que es un metodología ideal para trabajar en proyectos reales. Scrum está enfocado al trabajo por equipos no demasiado grandes, debido a que mi equipo únicamente estaba formado por mi mismo no hemos podido exprimir al máximo esta metodología. La gran ventaja que Scrum nos ha aportado ha sido su facilidad para adaptarse a los cambios en los requerimientos, ya que no hubo ningún problema en introducir una nueva historia de usuario en mitad del desarrollo del proyecto, cosa que utilizando una metodología de desarrollo tradicional hubiera sido más difícil adaptarse a este cambio.

El framework SringMVC también nos ha aportado grandes ventajas para la realización de este proyecto. Aunque uno de los problemas que he encontrado en este proyecto fue adaptarme a utilizar este *framework*, debido al aprendizaje lento que tiene este *framework* tuve que leer mucho, consultar bibliografía y preguntar a mi tutor hasta que conseguí comprender el funcionamiento y la manera de configurar y utilizar este *framework*.

También durante este periodo he puesto en práctica muchos de los conocimientos y habilidades adquiridos durante mis años de formación en la Universidad de Jaén, acudiendo en más de una ocasión a los apuntes de varias asignaturas para recordar cuestiones olvidadas que en el momento que estudié dije: ¿utilizaré esto alguna vez? la respuesta es sí.

Para finalizar, al analizar el problema que ha dado lugar a este proyecto descubrí que existe un gran nicho de negocio en este mercado. No se conoce ninguna aplicación similar a la que hemos desarrollado, únicamente la aplicación del Consejo de Cofradías de Sevilla se asemeja en parte. Todas las grandes ciudades de Andalucía poseen carrera oficial en su Semana Santa, y cada vez mas, pequeños municipios y localidades optan por instalar este lugar de paso obligatorio para sus hermandades debido al beneficio económico que estos obtienen con su venta. Se abre por tanto una gran puerta para comercializar nuestra aplicación y poder obtener un beneficio económico además de una gran satisfacción personal.

5.1 Mejoras y trabajos futuros

Cuando he contado a mis amigos y familiares sobre que iba a tratar mi trabajo de fin de grado, la respuesta mayoritaria que siempre me he encontrado ha sido: “¿Entonces eso es para que yo pueda meterme en la página web y pueda comprar las sillas por internet no?” a lo que yo respondía: “Más o menos”.

Permitir a los usuario poder registrarse en la aplicación, y que estos puedan comprar sus sillas de carrera oficial y realizar el pago mediante una plataforma de pago segura aportaría a nuestra aplicación unas funcionalidades muy interesantes para su comercialización. Estas funcionalidades no estaban dentro de los requisitos principales del sistema, que fueron impuestos por la Agrupación de Cofradías de Jaén, pero debido a que el diseño de nuestra arquitectura está realizado según el modelo vista controlador se ha obtenido un sistema modular que nos permitirá sin ningún problema adaptar el sistema para dos tipos distintos de usuarios.

Ya que el sistema sería usado por muchos usuarios es conveniente disponer de un historial en el que queden reflejados todos los movimientos en torno a las reservas, quien, que y cuando ha realizado una reserva.

Para la asociación de cofradías encargada de la carrera oficial sería de gran ayuda un sistema de estadísticas en el que una parte de las estadísticas se centre en el número de ventas, ocupación de los palcos y dinero obtenido con las

ventas y otra parte de las estadísticas que estudie la demanda que tiene cada palco con objeto de poder ajustar precios en los años venideros.

6 BIBLIOGRAFÍA

[1] S. McConnell, Desarrollo de proyectos informáticos, Madrid: McGraw Hill, D.L., 2000.

[2] P. González, Apuntes de la asignatura: Desarrollo Ágil, Jaén: Universidad de Jaén, 2014.

[3] Agilemanifesto.org. *Manifesto for Agile Software Development*. [online] Available at: http://agilemanifesto.org [Accessed 4 Sep. 2015].

[4] www.europapress.es. (2015). *Cuál es el origen de las procesiones de Semana Santa*. [online] europapress.es. Available at: http://www.europapress.es/sociedad/noticia-cual-origen-procesiones-semana-santa-20150330145107.html [Accessed 1 May 2016].

[5] Es.wikipedia.org. (2016). *Semana Santa*. [online] Available at: https://es.wikipedia.org/wiki/Semana_Santa [Accessed 1 May 2016].

[6] COLUMNAZERO. (2013). *ASÍ SE CELEBRA LA SEMANA SANTA POR EL MUNDO - COLUMNAZERO*. [online] Available at: http://columnazero.com/asi-se-celebra-la-semana-santa-por-el-mundo/ [Accessed 1 May 2016].

[7]R. Briones Gómez, "La Semana Santa andaluza", *Ugr.es*, 1983. [Online]. Available: http://www.ugr.es/~pwlac/G02_01Rafael_Briones_Gomez.html. [Accessed: 01- May- 2016].

[8]"Un estudio revela que el impacto económico de la Semana Santa de Palencia supera los 2,4 millones de euros : Ayuntamiento de Palencia", *Aytopalencia.es*, 2016. [Online]. Available: http://www.aytopalencia.es/node/1261. [Accessed: 13- Apr- 2016].

[9]"Una Semana Santa de 280 millones", *ELMUNDO*, 2015. [Online]. Available: http://www.elmundo.es/andalucia/2015/04/05/55215ace22601dc8288b456b.html. [Accessed: 09- Apr- 2016].

[10]J. Carrero Rodriguez, *Diccionario Cofradiero*. Sevilla, 1996.

[11]J. Gómez Palas, "Abierto el plazo de renovación de sillas y palcos", *elcorreoweb.es*, 2016. [Online]. Available: http://elcorreoweb.es/maspasion/abierto-el-plazo-de-renovacion-de-sillas-y-palcos-JX1246281. [Accessed: 04- May- 2016].

[12]F. Márquez, "La Semana Santa de 2015 se salda con un beneficio de 125.000 euros", *Cádiz Directo - Noticias de Cádiz y Provincia - Información Cádiz - Sucesos y Opinión*, 2015. [Online]. Available: http://www.cadizdirecto.com/la-semana-santa-de-2015-se-salda-con-un-beneficio-de-125-000-euros.html. [Accessed: 06- Mar- 2016].

[13]J. Cretario, "Así se reparte el dinero del Consejo de Cofradías - Pasión en Sevilla", *Pasión en Sevilla*, 2014. [Online]. Available: http://sevilla.abc.es/pasionensevilla/actualidad/noticias/el-consejo-recaudo-38-millones-de-euros-por-las-sillas-1404857033.html. [Accessed: 11- Apr- 2016].

[14]A. Morente, "Tráfico de sillas en la Carrera Oficial: un mercado negro que mueve miles de euros", *elcorreoweb.es*, 2016. [Online]. Available: http://elcorreoweb.es/sevilla/trafico-de-sillas-en-la-carrera-oficial-un-mercado-negro-que-mueve-miles-de-euros-GA1570686. [Accessed: 04- Jun- 2016].

[15]J. Plaza, "Francisco Latorre, nuevo presidente de la Agrupación de Cofradías | Pasión en Jaén", *Pasionenjaen.com*, 2014. [Online]. Available: http://pasionenjaen.com/francisco-latorre-nuevo-presidente-de-la-agrupacion-de-cofradias/. [Accessed: 24- May- 2015].

[16]"Reactivada la Web de Gestión de Abonos de Sillas y Palcos para la Semana Santa de 2016", *Hermandades-de-sevilla.org*, 2016. [Online]. Available: http://www.hermandades-de-sevilla.org/noticias/803-reactivada-la-web-de-gestion-de-abonos-de-sillas-y-palcos-para-la-semana-santa-de-2016. [Accessed: 13- Mar- 2016].

[17]"Unión Cine Ciudad - Compra por Internet tu entrada de cine, estrenos, promociones", *Cineciudad.com*, 2016. [Online]. Available: http://www.cineciudad.com/. [Accessed: 07- Jun- 2016].

[18]"Compraentradas", *Compraentradas.com*, 2016. [Online]. Available: http://www.compraentradas.com. [Accessed: 07- Jun- 2016].

[19]"21. Web MVC framework", *Docs.spring.io*, 2016. [Online]. Available: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. [Accessed: 22- Oct- 2015].

[20]a. Mark Otto, "Bootstrap · The world's most popular mobile-first and responsive front-end framework.", *Getbootstrap.com*, 2016. [Online]. Available: http://getbootstrap.com/. [Accessed: 23- Oct- 2015].

[21]"CSS Snapshot 2015", *W3.org*, 2015. [Online]. Available: https://www.w3.org/TR/CSS/#css. [Accessed: 13- Jan- 2016].

[22]"JavaScript", *Es.wikipedia.org*, 2016. [Online]. Available: https://es.wikipedia.org/wiki/JavaScript. [Accessed: 13- Jun- 2016].

[23]"What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance", *Scrumalliance.org*, 2016. [Online]. Available: https://www.scrumalliance.org/why-scrum. [Accessed: 26- Oct- 2015].

[24]M. Cohn and K. Beck, *User stories applied: For agile software development*, 12th ed. Boston: Addison-Wesley Educational Publishers, 2004

[25]M. Cohn, "User Stories and User Story Examples by Mike Cohn", *Mountain Goat Software*, 2016. [Online]. Available: http://www.mountaingoatsoftware.com/agile/user-stories. [Accessed: 15- Oct- 2015].

[26]K. Pohl, *Requirements engineering: Fundamentals, principles, and techniques*. Heidelberg: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2010.

[27]"Estimando en Puntos de Historia," 2011. [Online]. Available: http://geeks.ms/devnettips/2011/02/02/estimando-en-puntos-de-historia/. Accessed: Nov. 15, 2015.

[28]M. Cohn, "User Stories and User Story Examples by Mike Cohn", *Mountain Goat Software*, 2016. [Online]. Available: http://www.mountaingoatsoftware.com/agile/user-stories. [Accessed: 15- Oct- 2015].

[29]A. Álvarez, R. de las Heras, and C. Lasa, *Métodos Ágiles y Scrum (Manuales Imprescindibles)*, Anaya, Ed.

[30]D. Álvarez and V. perfil, "Be My Scrum Master: El cálculo de la velocidad", *Bemyscrummaster.blogspot.com.es*, 2016. [Online]. Available: http://bemyscrummaster.blogspot.com.es/2013/11/el-calculo-de-la-velocidad.html. [Accessed: 02- Nov- 2015].

[31]I. Sommerville, *Software engineering (9th edition)*, 9th ed. Boston: Addison-Wesley Educational Publishers, 2010.

[32]S. Bennett, S. McRobb, and R. Farmer, *Object-oriented systems analysis and design using UML*, 3rd ed. London: McGraw Hill Higher Education, 2007.

[33]*JavaServer Pages Technology (2010). The Java EE 5 Tutorial. Oracle. cap 5.*, 2007. [Online]. Available: http://docs.oracle.com/javaee/5/tutorial/doc/. Accessed: Jun. 13, 2016.

[34]A. Silberschatz, H. Korth and S. Sudarshan, *Database system concepts*, 4th ed. Boston: McGraw-Hill, 2002, p. Capitulo 7.

[35]"Clave foránea," in *Wikipedia*, Wikimedia Foundation. [Online]. Available: https://es.wikipedia.org/wiki/Clave_for%C3%A1nea. Accessed: Jun. 13, 2016.

[36]A. S. Solutions, "Prototypes, specifications, and diagrams in One tool," 2002. [Online]. Available: http://www.axure.com/. Accessed: Jun. 13, 2016.

[37]Oracle, "JavaServer pages technology,". [Online]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. Accessed: Jun. 13, 2016.

[38]O. Corporation, "JSP standard tag library," 2013. [Online]. Available: https://jstl.java.net/. Accessed: Jun. 13, 2016.

[39]G. Seshadri, "Understanding JavaServer pages model 2 architecture," JavaWorld, 1999. [Online]. Available: http://www.javaworld.com/article/2076557/java-web-development/understanding-javaserver-pages-model-2-architecture.html. Accessed: Jun. 13, 2016.

[40]"21. Web MVC framework", *Docs.spring.io*, 2016. [Online]. Available: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. [Accessed: 22- Oct- 2015].

[41]M. Fowler, "Inversion of control containers and the dependency injection pattern," martinfowler.com, 2004. [Online]. Available: http://martinfowler.com/articles/injection.html. Accessed: Jun. 13, 2016.

[42]"MySQL," 2016. [Online]. Available: http://www.mysql.com. Accessed: Jun. 13, 2016.

[43]"¿Qué es PaaS?,". [Online]. Available: http://www.interoute.es/what-paas. Accessed: Jun. 13, 2016.

[44]A. T. Project, "Apache Tomcat® - welcome!," 1999. [Online]. Available: http://tomcat.apache.org/. Accessed: Jun. 13, 2016.

[45]H. Ferentschik and G. Morling, "Hibernate Validator JSR 349 reference implementation reference guide 5.0.3.Final," 2014. [Online]. Available: http://docs.jboss.org/hibernate/validator/5.0/reference/en-US/pdf/hibernate_validator_reference.pdf. Accessed: Jun. 13, 2016.

ANEXO I - MANUAL DE INSTALACIÓN.

En este anexo explicaremos la forma de configurar, subir, y desplegar nuestra aplicación en www.openshift.com utilizando el sistema operativo MAC OSX el Capitán versión 10.11.1.

1 Registro en www.openshift.com

Para poder empezar a instalar nuestra aplicación es necesario tener una cuenta en www.openshift.com, en caso contrario puedes crear una rellenado el formulario que aparece en la web.

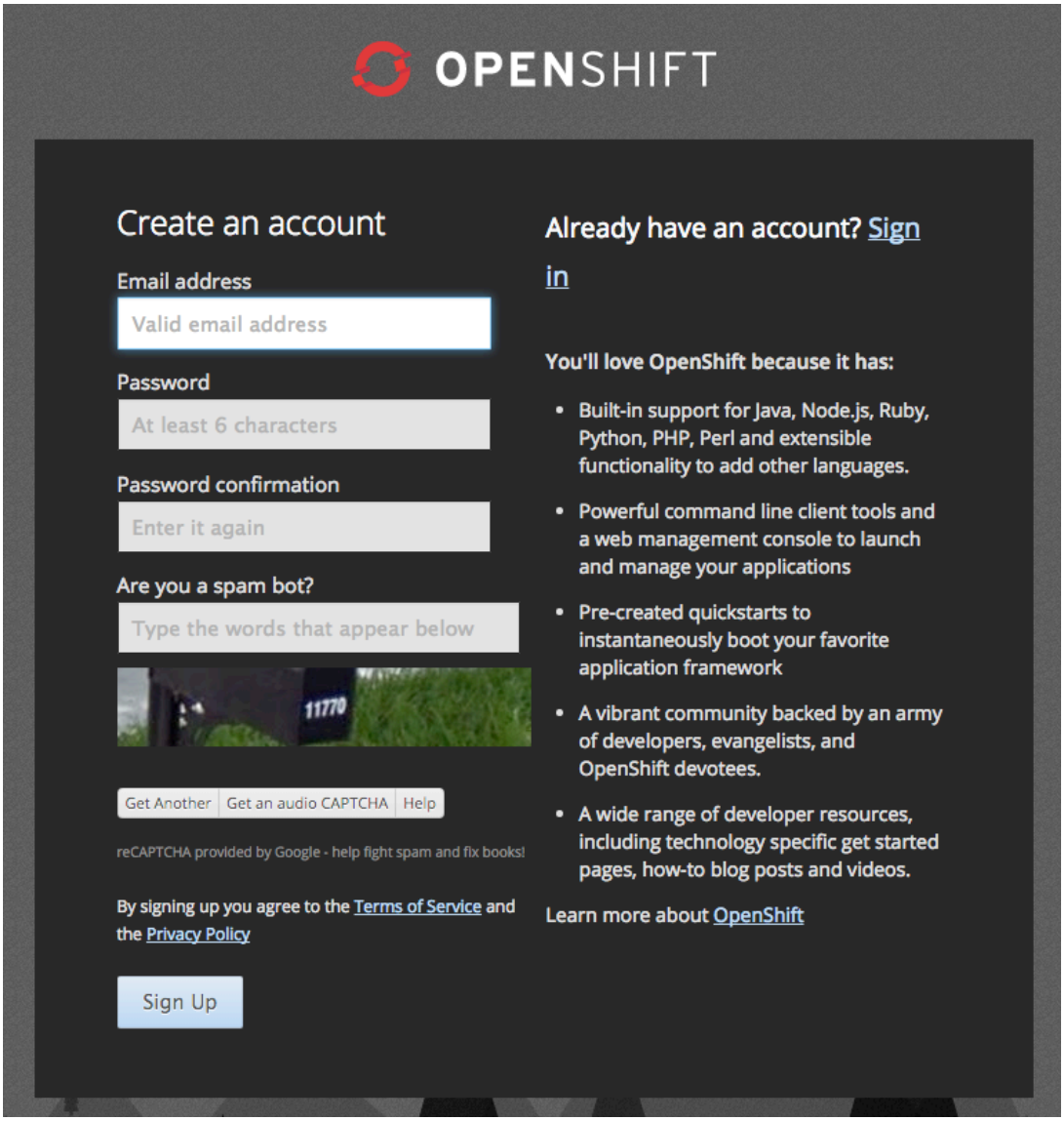

Ilustración A2.1 - Registro en openshift.com

2 Instalación y configuración de RHC Tools.

La instalación de las herramientas del cliente de openshift nos servirán para crear y administrar las aplicaciones de Openshift. Para poder instalar RHC es necesario tener instalado Ruby y Git.

En OS X, Ruby ya viene instalado por defecto. Para instalar GIT es necesario ejecutar el archivo DMG que podemos descargarnos en este enlace: https://sourceforge.net/projects/git-osx-installer/

Una vez instalado GIT y Ruby procedemos a instalar RHC ejecutando en la consola:

```
$ sudo gem install rhc
```

Ahora tenemos que configurar RHC. Ejecutamos:

```
$ rhc setup
```

Y tendremos que introducir nuestras credenciales de usuario:

```
Login to openshift.redhat.com: user@example.com
Password: password
```

A continuación se pregunta si desea generar un *token* de autenticación, la duración de este *token* es de un día.

```
OpenShift can create and store a token on disk which allows to you to
access the server without using your password. The key is stored in your
home directory and should be kept secret. You can delete the key at any
time by running 'rhc logout'.
Generate a token now? (yes|no) yes
Generating an authorization token for this client ... lasts about 1 day
```

Ahora tenemos que configurar las claves SSH para que el sistema pueda conectarse de forma remota a las aplicaciones.

```
Your public ssh key must be uploaded to the OpenShift server to access
code.
Upload now? (yes|no) yes

Since you do not have any keys associated with your OpenShift account,
your new key will be uploaded as the 'default' key

Uploading key 'default' from C:\Users\User1\.ssh\id_rsa.pub ... done
```

El siguiente paso es elegir el *namespace* que será incluido en todas las aplicaciones que creemos en openshift.

```
Checking for a domain ... none

Your domain is unique to your account and is the suffix of the public
URLs we assign to your applications. You may configure your domain here
or leave it blank and use 'rhc domain create' to create a domain later.
You will not be able to create applications without first creating a
domain.

Please enter a domain (letters and numbers only) |<none>|: MyDomain
Your domain name 'MyDomain' has been successfully created
```

Tras realizar este paso rhc ya está configurado para poder crear nuestras aplicaciones.

3 Creando la aplicación

En la pantalla de inicio de openshift pulsamos sobre *Add Application*

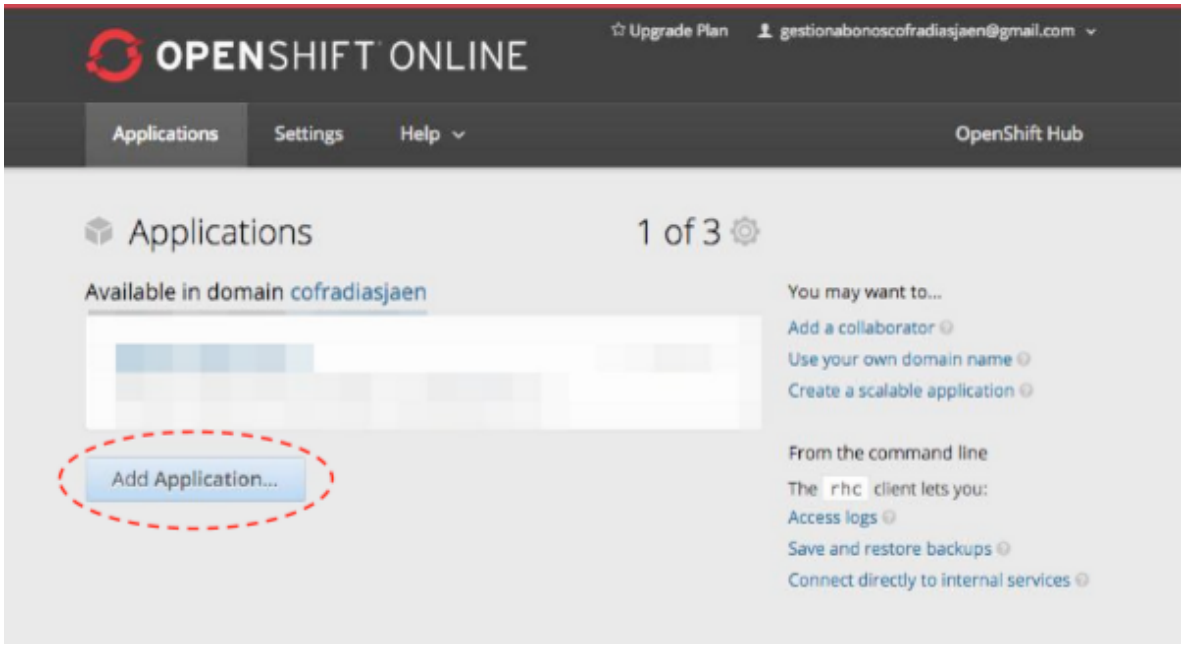

Ilustración A2.2 - Añadir nueva aplicación

Ahora debemos de seleccionar cual es el tipo de aplicación que vamos a crear. En nuestro caso es Tomcat 7 (JBoss EWS 2.0).

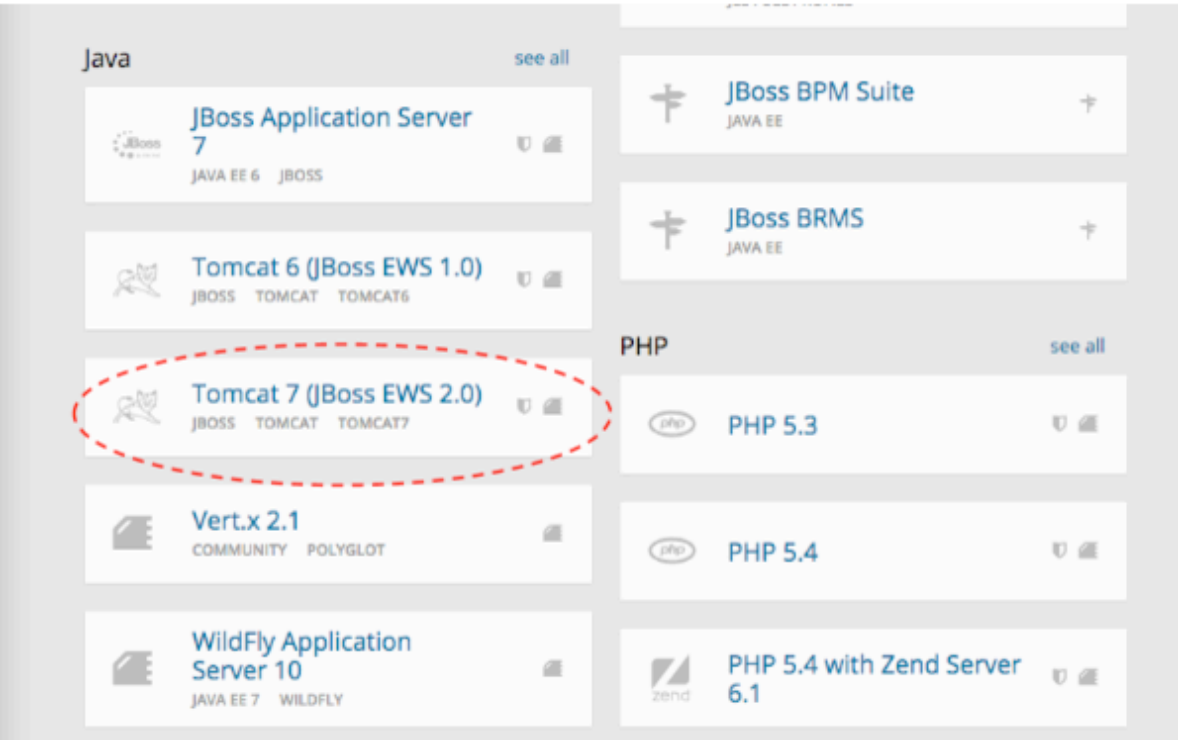

Ilustración A2.3 - Selección del tipo de aplicación.

El sistema nos llevará a otra ventana en la que podemos configurar distintos parámetros de la aplicación. Entre otras cosas, cuál será la URL de la aplicación.

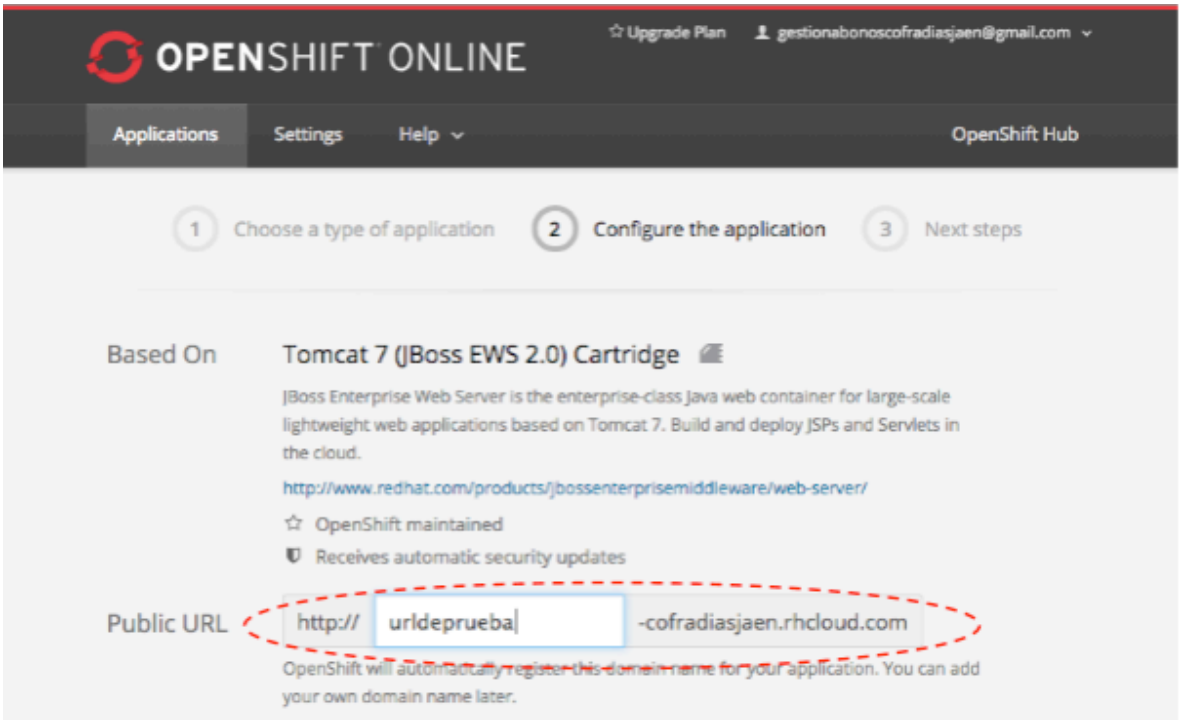

Ilustración A2.4 - Seleccionar nombre de la URL de la aplicación.

Tras crear la aplicación tenemos que usar GIT para clonar el repositorio en nuestro ordenador. Se creará una carpeta con el código fuente actual de la aplicación, que openshift crea por defecto. En esta carpeta solo necesitamos otra carpeta que se llame webapps, y ahí introduciremos el fichero .war de nuestra aplicación.

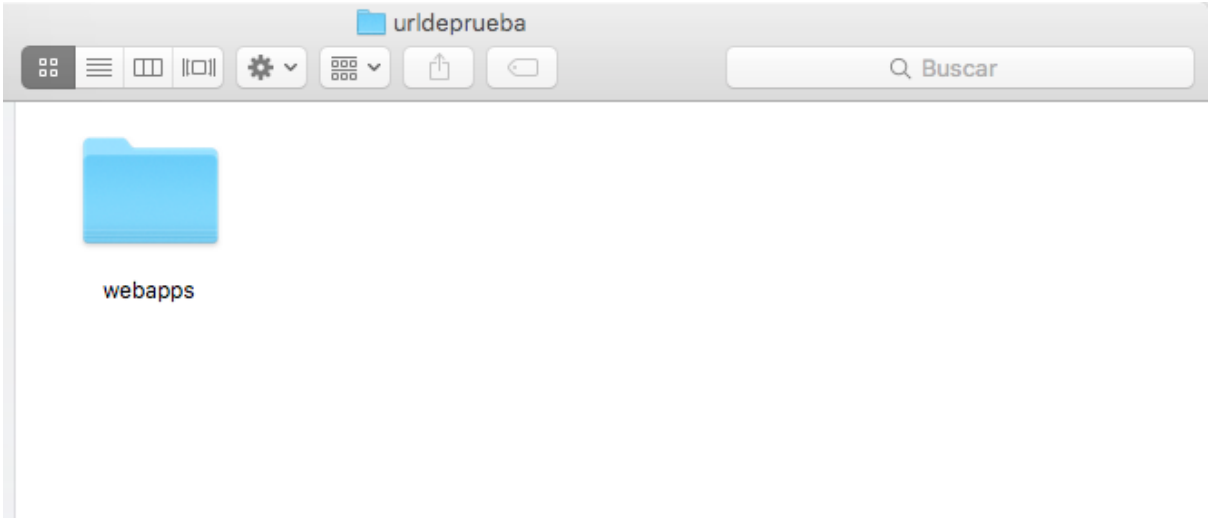

Ilustración A2.5 - Aspecto del repositorio clonado.

Tras introducir el fichero .war en la carpeta ejecutaremos en la consola los siguientes comandos de git.

```
git add .
git commit -m "Descripción de los cambios"
git push
```

Al realizar esto, se subirán los cambios al servidor, y se desplegará la aplicación en la ruta indicada para poder utilizarla.

ANEXO II MANUAL DE USUARIO

El presente manual de usuario le ayudará a gestionar de manera ágil y sencilla nuestra aplicación web para la gestión de abonos de una carrera oficial.

1 Entrada a la aplicación

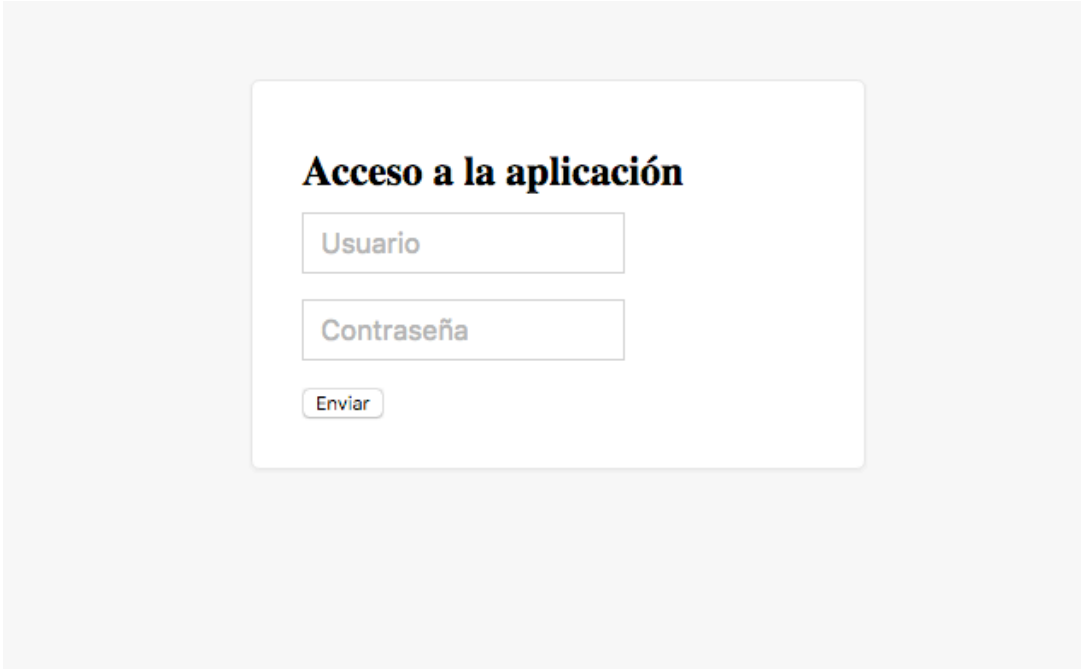

Ilustración A1.1 - Identificación de usuario.

Una vez introducidos el nombre de usuario y la contraseña, pulsar el botón de enviar para acceder a la aplicación.

2 Panel de control

Ilustración A1.2 - Panel de control de la aplicación.

En el panel de control se podrán realizar operaciones relacionadas los abonados, así como, la gestión de los palcos y acceder a las herramientas del sistema. Al pulsar sobre *Log Out* se produce el cierre de sesión.

2.1 Alta abonado

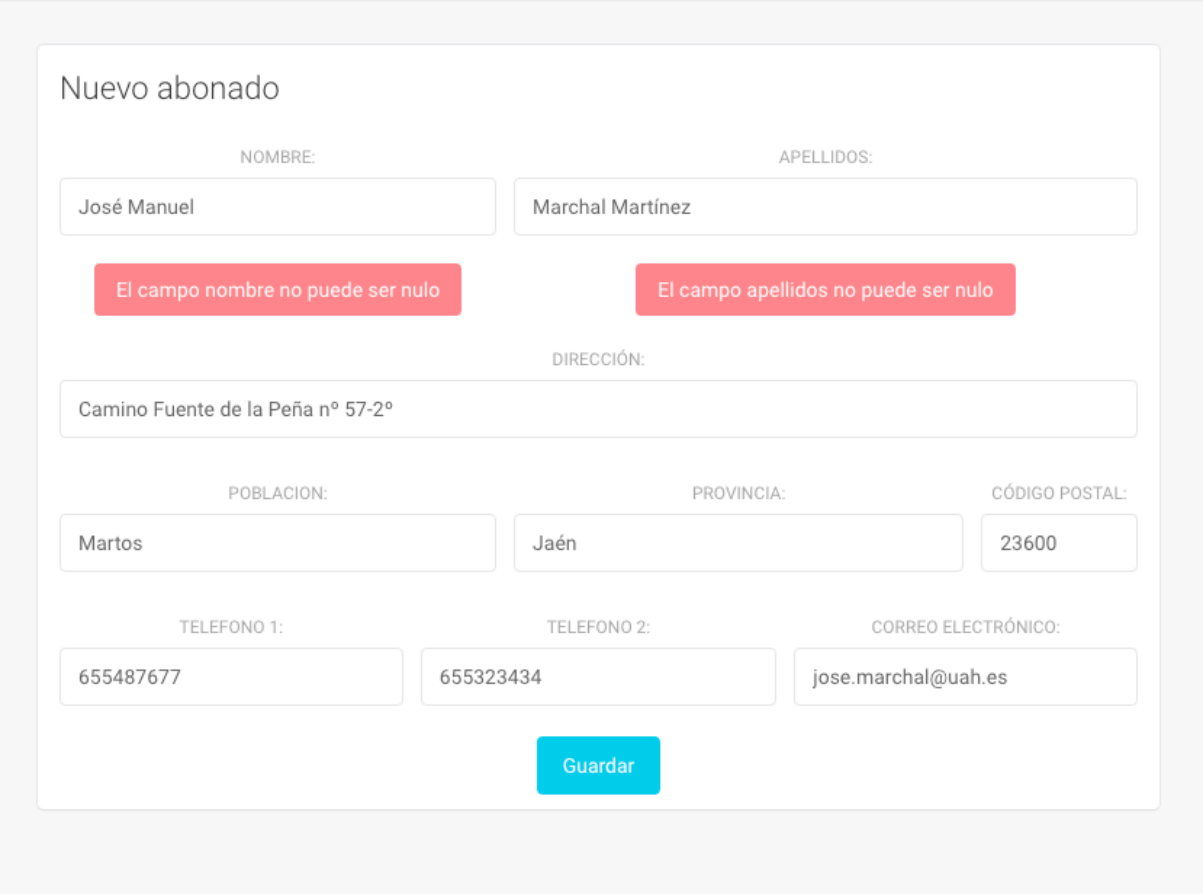

Ilustración A1.3 - Formulario de alta de abonado.

Para registrar un nuevo abonado en la aplicación rellene el formulario y envíe la solicitud. Los campos *nombre* y apellidos no pueden ser nulos.

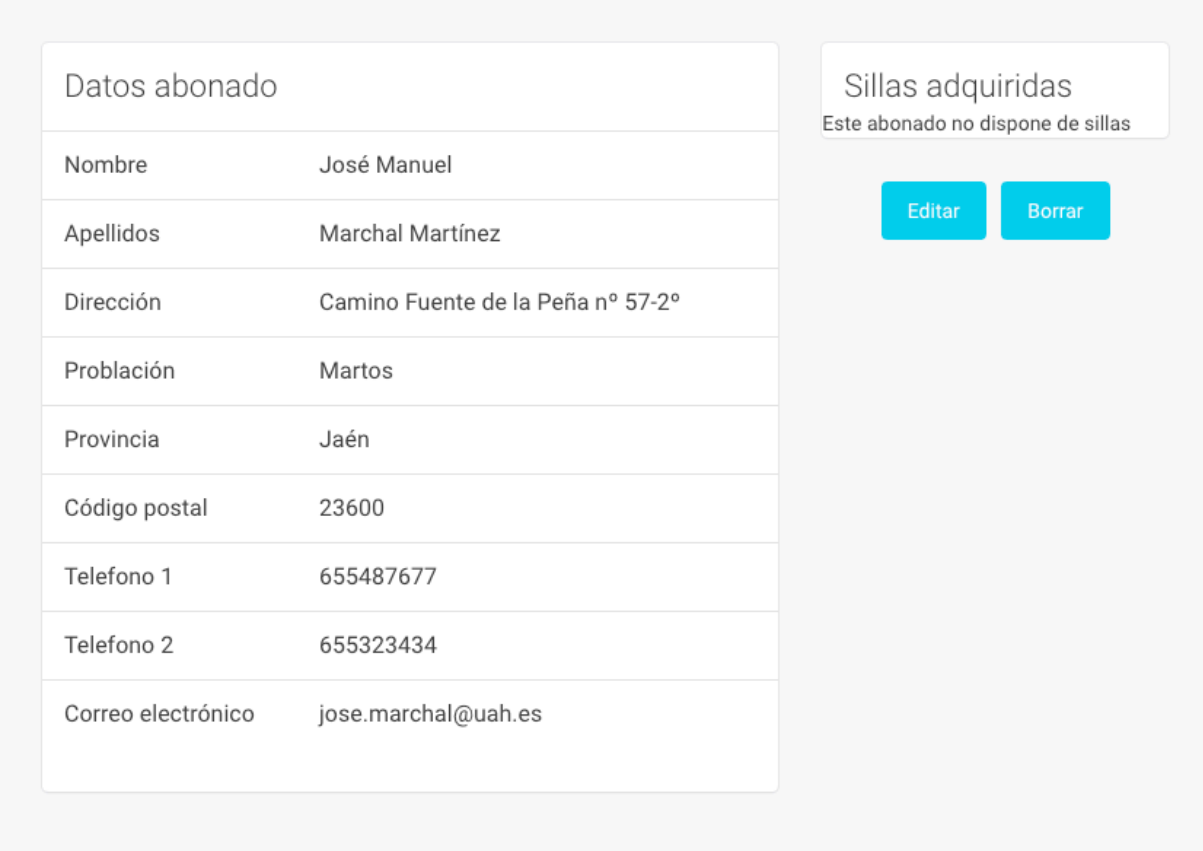

Ilustración A1.4 - Ficha con los datos de un abonado.

Tras pulsar el botón *Guardar* el sistema nos enviará a la ficha del abonado donde poder visualizar todos sus datos, además de las sillas que tiene adquiridas. También se pueden realizar las siguientes acciones:

- Editar: para modificar algún dato del abonado se modifica los datos del formulario que nos muestra el sistema y pulsamos el botón de Guardar.

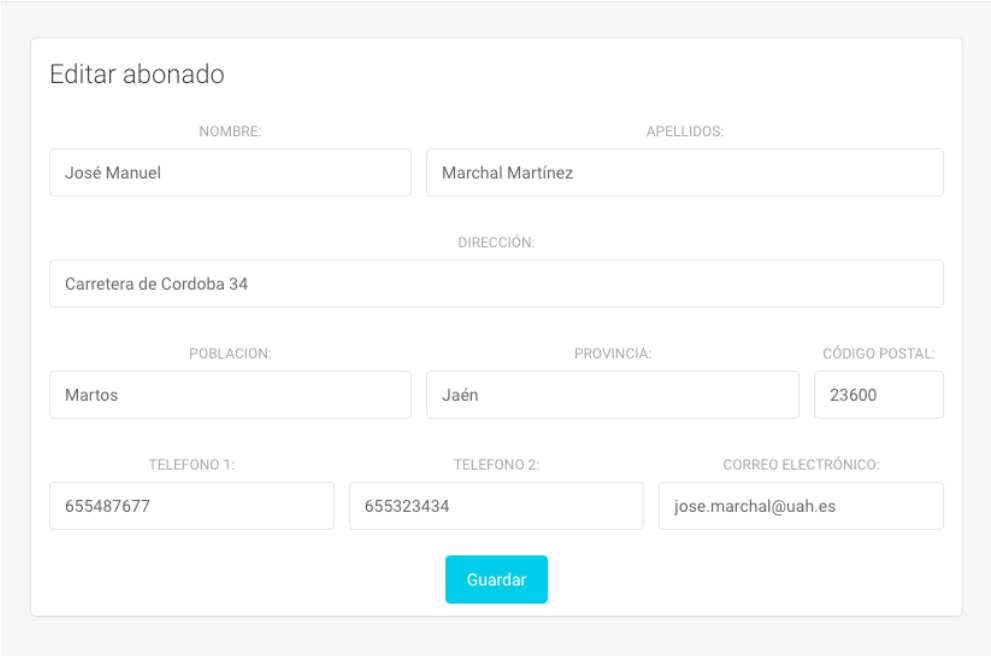

Ilustración A1.5 - Formulario para editar un abonado.

- Borrar: si pulsamos sobre el botón *Borrar*, el abonado se eliminará automáticamente del sistema.

2.2 Buscar abonado

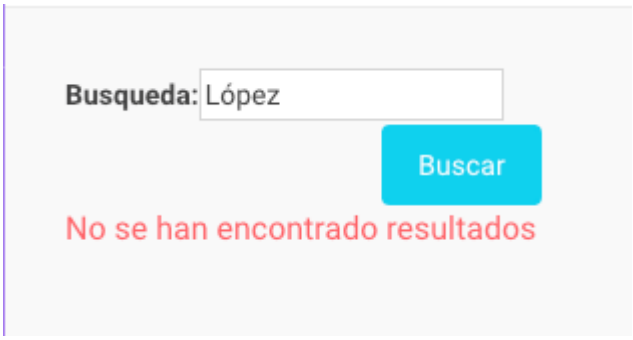

Ilustración A1.6 - Búsqueda de un abonado.

Para realizar la búsqueda de algún abonado es necesario introducir en el formulario de forma parcial o completa o su nombre, o sus apellidos, o ambos y pulsar en el botón *Buscar*.

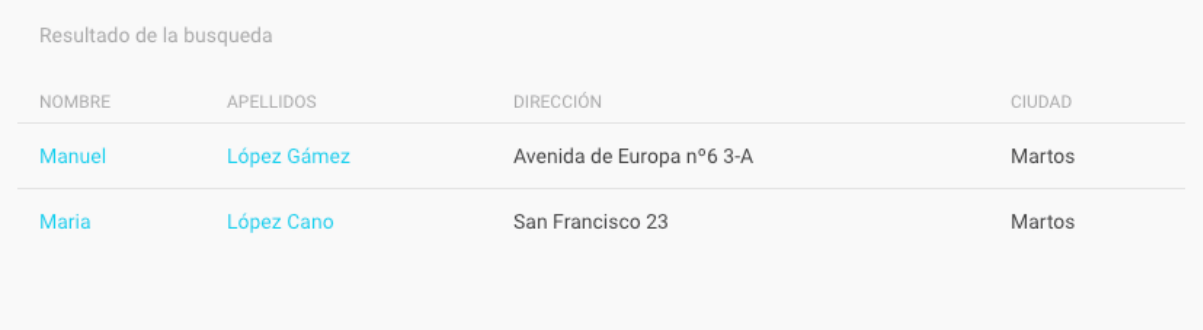

Ilustración A1.7 - Resultado de la búsqueda de un abonado.

El sistema nos mostrará un listado de todos los abonados que tengan coincidencias con la cadena introducida en el formulario. Para visualizar la ficha del usuario es necesario pulsar sobre su *nombre* o *apellidos*.

2.3 Listado de palcos

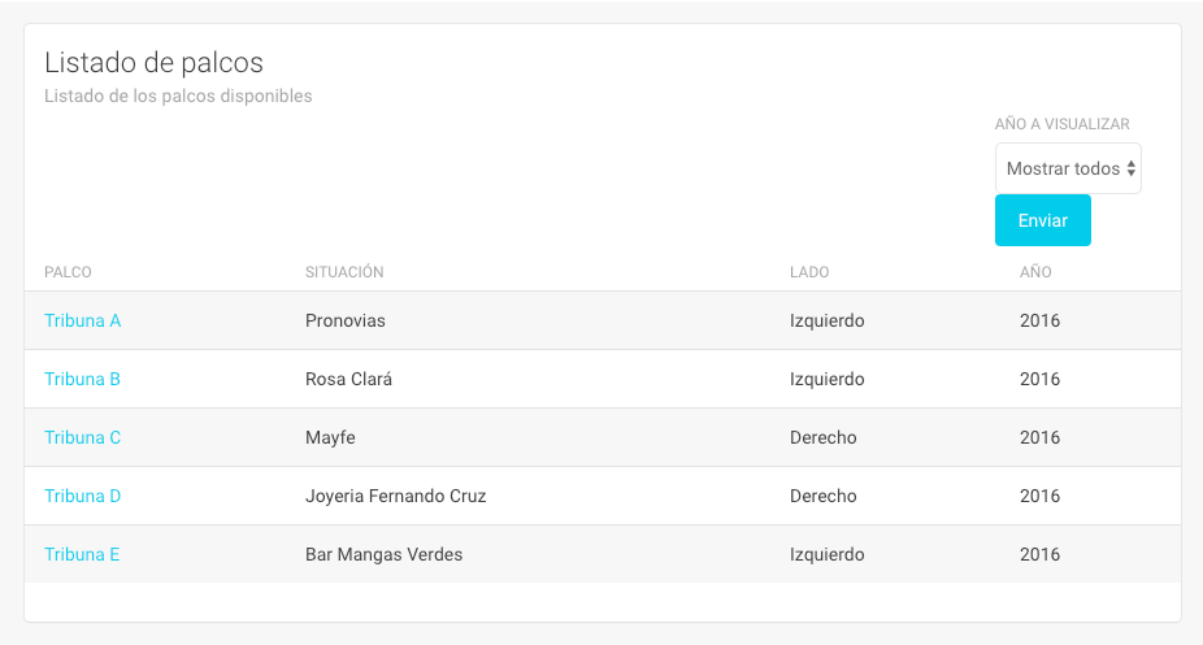

Ilustración A1.8 - Listado de los palcos disponibles.

El sistema nos mostrará todos los palcos de los que dispone ordenados por año. Si pulsamos sobre la lista desplegable, nos permite filtrar el listado por años. Al pinchar sobre el nombre de un palco el sistema nos enviará a la ficha de este palco donde poder visualizar todos los datos.

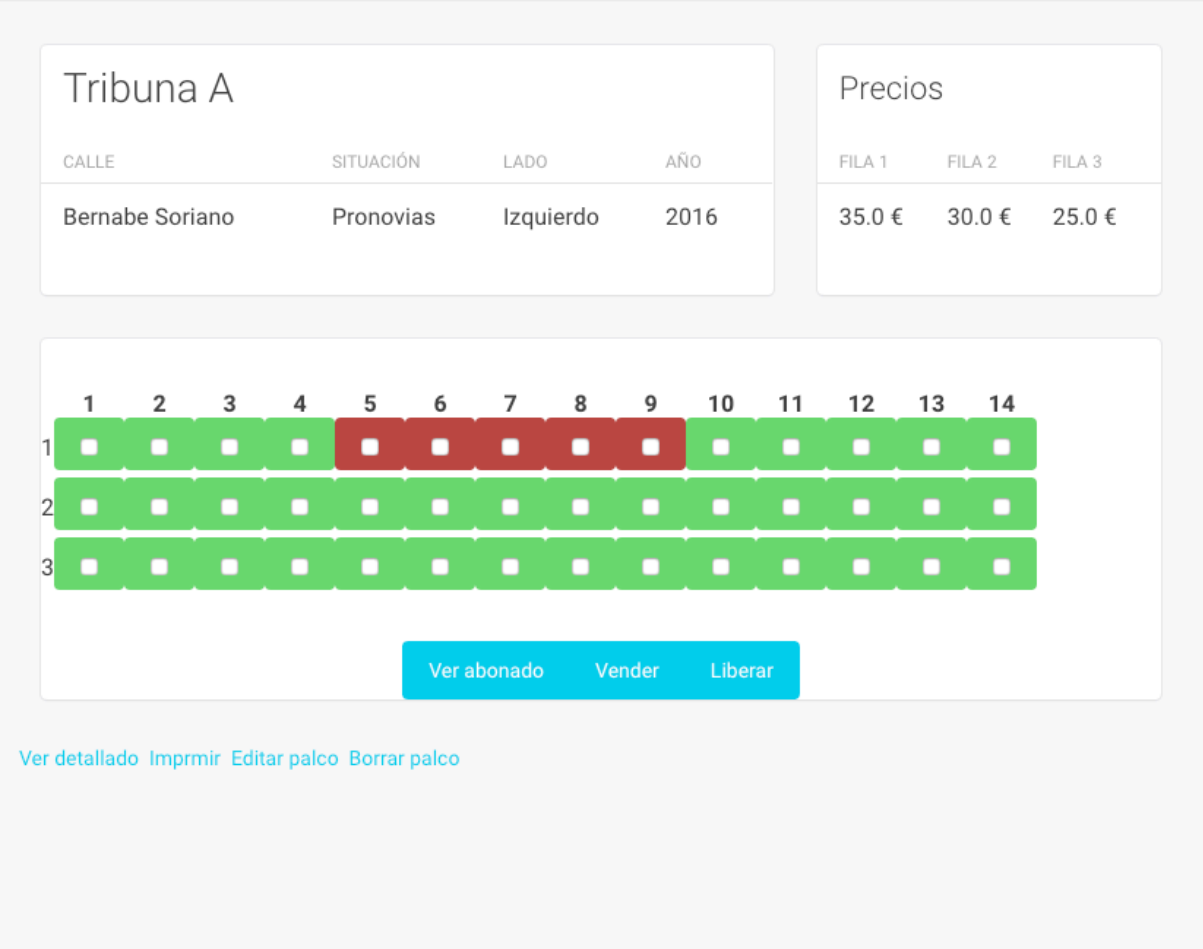

Ilustración A1.9 - Ficha con los datos de un palco.

En la ficha del palco aparece un mapa de la ocupación de dicho palco. Las sillas de color verde indica que esa silla está libre. Las sillas de color rojo indica que esa sillas se encuentra ocupada.

Las acciones disponibles dentro de la ficha del palco son:

- Ver abonado: al seleccionar una silla en rojo y pulsar sobre el botón *Ver abonado*, el sistema nos enviará a la ficha del abonado.
- Vender: al seleccionar una o un conjunto de sillas y pulsar sobre el botón *Vender* se inicia el proceso de venta de abonos.
- Liberar sillas: al seleccionar una o un conjunto de sillas ocupadas y pulsar sobre el botón *Liberar,* el sistema elimina automáticamente la compra de las sillas por parte de un abonado, quedando estas libres.
- Ver detallado: al pulsar sobre el botón *Ver detallado* se muestra un listado detallado del palco, mostrando la relación entre abonados y sillas.
- Imprimir: al pulsar sobre el botón *Imprimir,* el sistema mostrará un listado detallado del palco preparado para imprimir.
- Editar palco: al pulsar el botón *Editar palco* el sistema nos permitirá poder modificar los principales parámetros de un palco

- Borrar palco: al pulsar sobre el botón *Borrar palco* se eliminará el palco del sistema

2.3.1 Vender

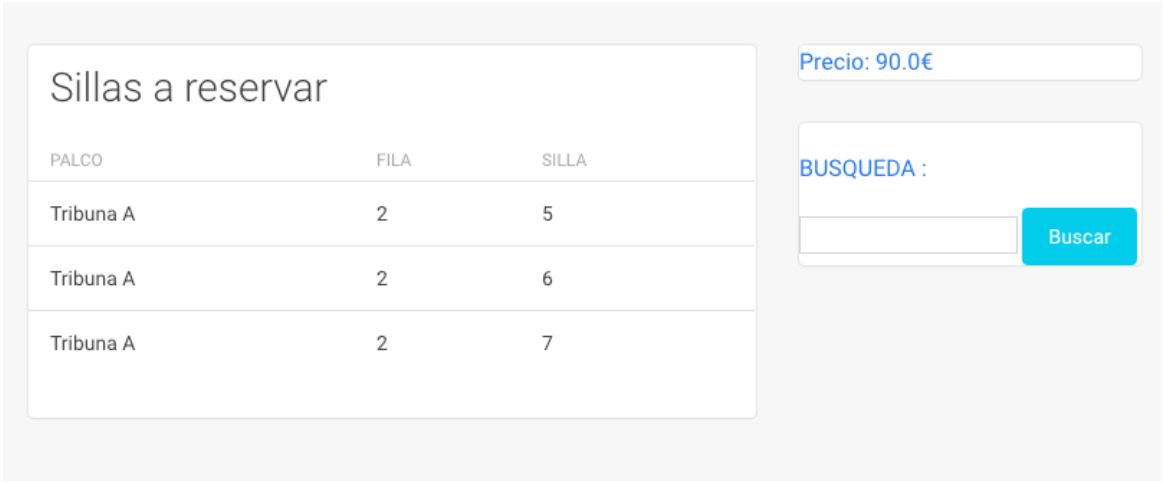

Ilustración A1.10 -Proceso de compra de abonos I.

Tras seleccionar las sillas a reservar es necesario seleccionar al abonado que va a comprar las sillas. Para eso introducir en el formulario del apartado búsqueda de forma parcial o completa o su nombre, o sus apellidos, o ambos y pulsar en el botón buscar.

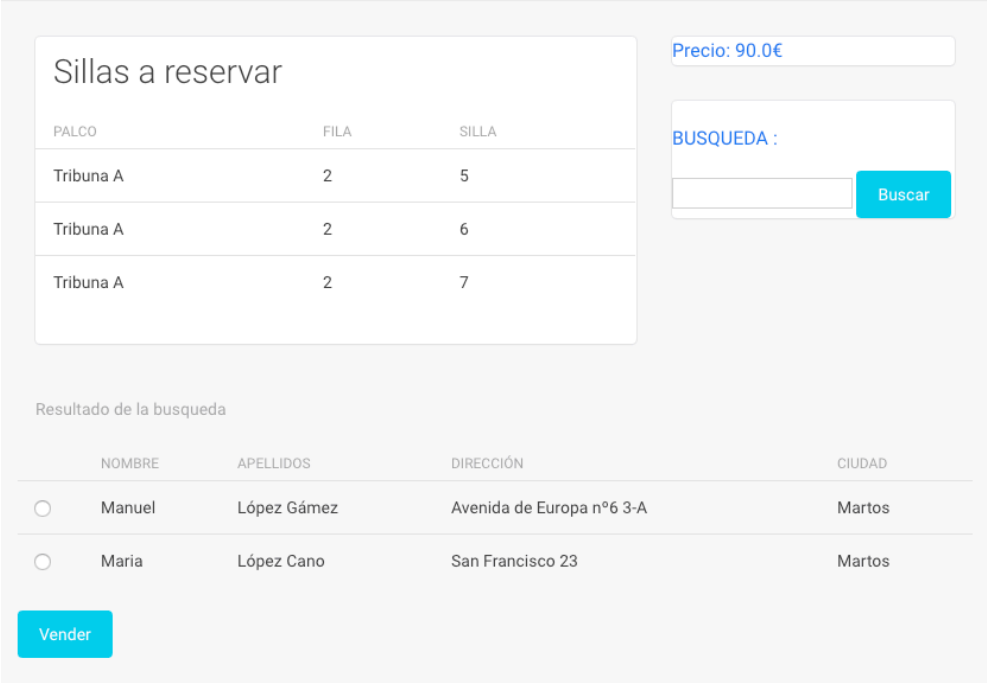

Ilustración A1.11 - Proceso de compra de abonos II.

Tras seleccionar el abonado que quiere realizar la compra y pulsar sobre el botón *Vender* se habrá realizado la compra.

2.3.2 Ver detallado

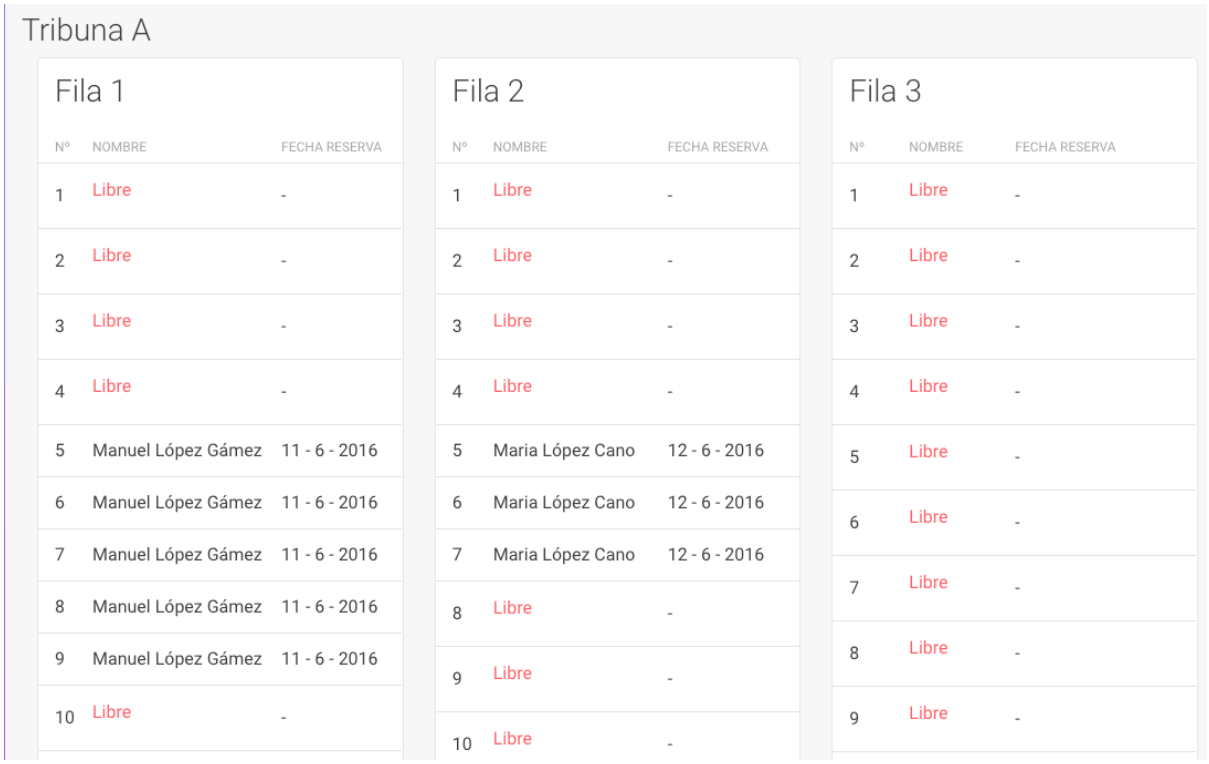

Ilustración A1.12 - Listado detallado de un palco.

El sistema nos muestra un listado donde aparecen las distintas filas del palco, además de todas sus sillas y el nombre del abonado al que pertenecen, además de la fecha en la que se realizó la reserva.

2.3.3 Editar palco

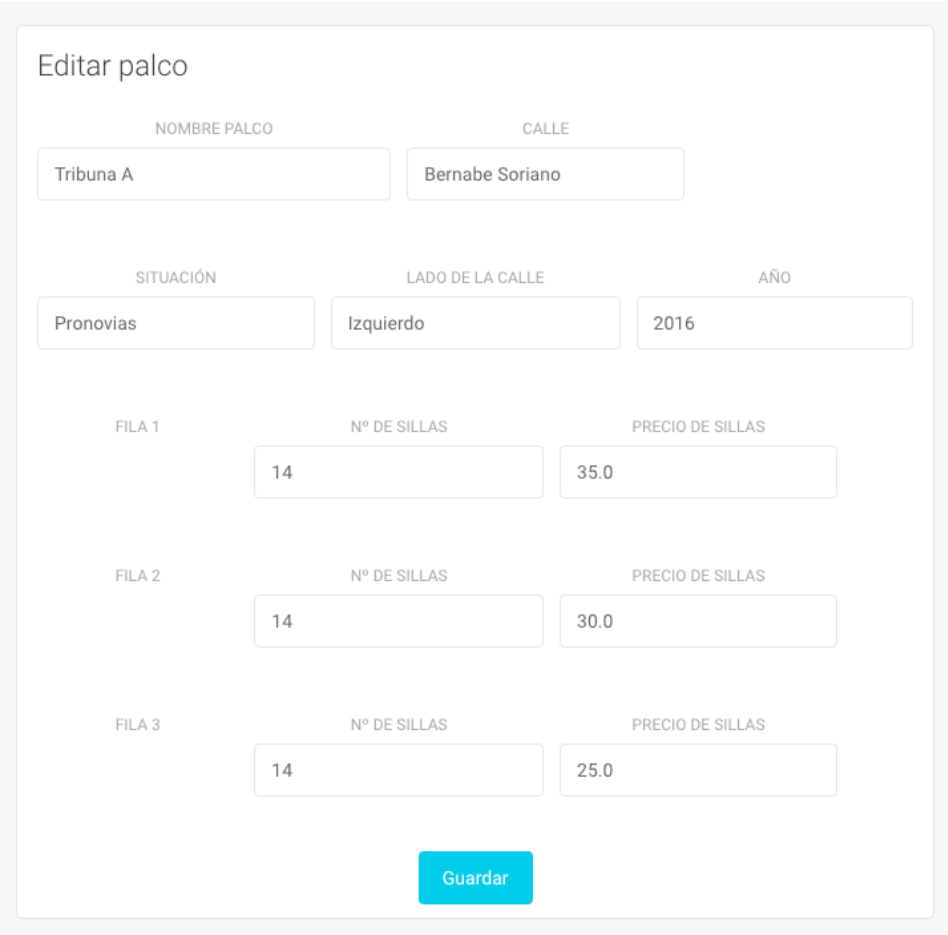

Ilustración A1.13 - Formulario para editar un palco.

El sistema nos muestra un formulario que podemos editar y cambiar los parámetros del palco. Al pulsar el botón *Guardar* se aplicarán los cambios realizados.

2.4 Herramientas

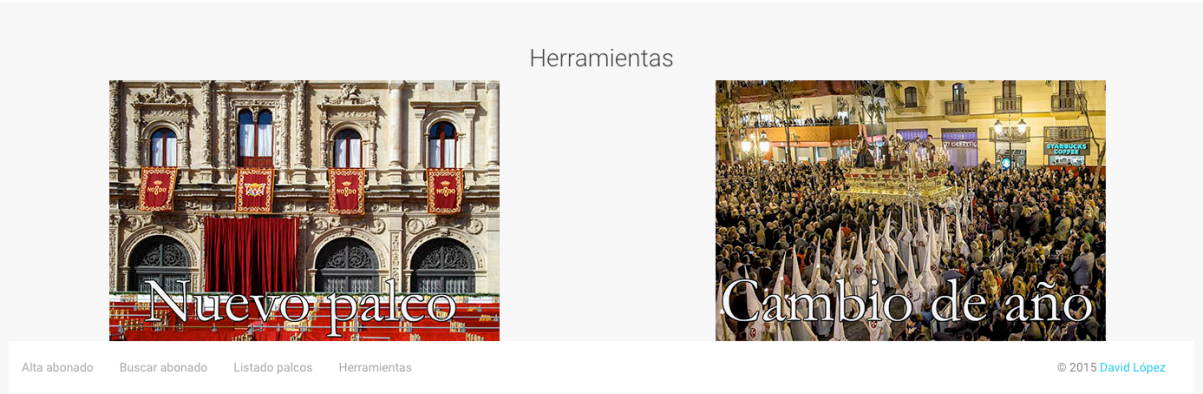

Ilustración A1.14 - Herramientas de la aplicación.

Las acciones disponibles son:

- Nuevo palco: al pulsar sobre la imagen con la leyenda *Nuevo palco* el sistema iniciará el proceso para la creación de un palco.
- Cambio de año: al pulsar sobre la imagen con la leyenda *Cambio de año* el sistema iniciará el proceso adaptar la plataforma para un nuevo año.

2.4.1 Nuevo palco

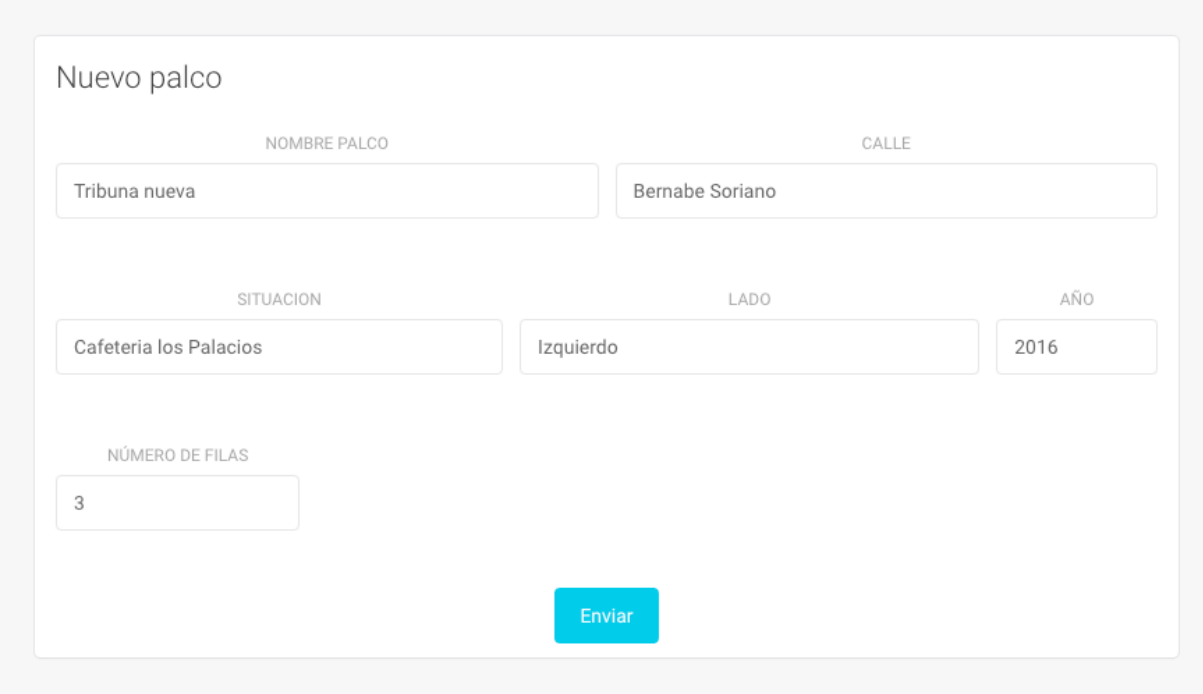

Ilustración A1.15 - Formulario de alta de un palco I.

Primero hay que indicar los principales datos sobre el palco, y el número de filas con las que cuenta.

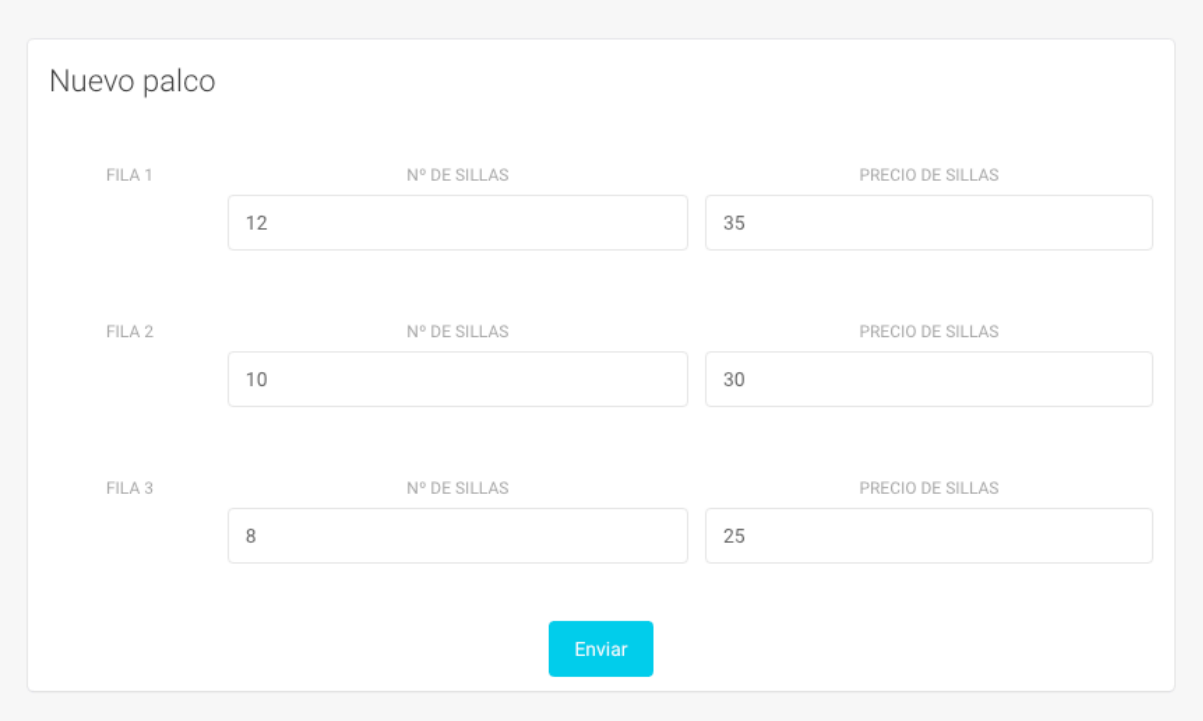

Ilustración A1.16 - Formulario de alta de un palco II.

Después de indicar el número de filas con las que cuenta, tenemos que indicar de cuantas sillas dispone cada fila y el precio de cada silla.

2.4.2 Cambio de año

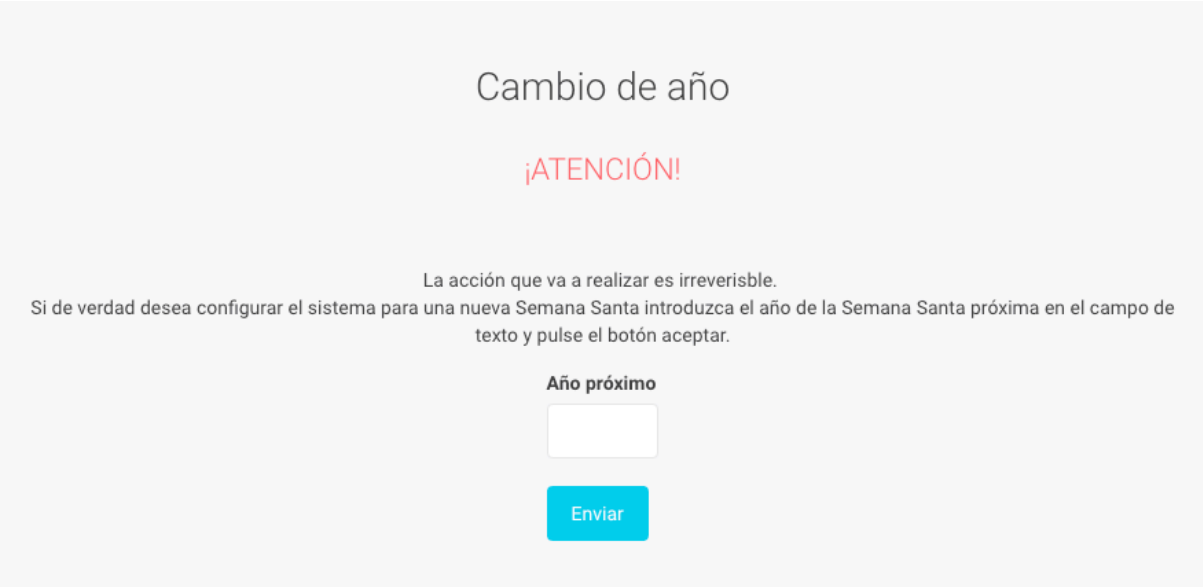

Ilustración A1.17 - Formulario para realizar cambio de año.

Para preparar la plataforma para una nueva Semana Santa tenemos que indicar el año de la Semana Santa para la que queremos configurar la aplicación.

Al pulsar el botón *Enviar* el sistema realizará una copia de todos los palcos del año anterior pero con ninguna de sus sillas ocupadas.

ANEXO III DESCRIPCIÓN DE CONTENIDOS SUMINISTRADOS.

El CD que se entrega contiene:

- **Memoria_DavidLopezCano**: memoria del TFG en formato PDF.
- **Codigo_DavidLopezCano**: código fuente del proyecto.
- **SQL_DavidLopezCano**: archivos para crear la base de datos.
- **Diagramas_DavidLopezCano**: fichero de VisualParadimg en el que se encuentran todos los diagramas que aparecen en la memoria.