Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Plataforma flexible para la venta de entradas a eventos de alta demanda

Estudiante: Alejandro Vázquez Calaza
Dirección: Alejandro Cortiñas Álvarez
Dirección: Miguel Ángel Rodríguez Luaces

A Coruña, agosto de 2020.

Dedico este Trabajo de Fin de Grado a mis padres y a mi hermano que me ayudaron y siempre estuvieron ahí en los momentos más difíciles y que sin ellos no hubiera sido posible terminar la carrera.

Agradecimientos

Quiero dar las gracias a mis tutores del Trabajo de Fin de Grado por el tiempo dedicado a ayudarme y guiarme en su realización, a todos los profesores que he tenido durante estos cuatro años de carrera por las cosas que me han enseñado, a mis compañeros y a los amigos que he conocido en la carrera, que me han hecho más amenos estos años. También quiero agradecer a mis padres y a mi hermano por toda la ayuda y el apoyo que me dieron durante la carrera que fue vital para conseguir sacarla adelante.

Resumen

El objetivo de este Trabajo de Fin de Grado es desarrollar una aplicación de venta de entradas de eventos con un proceso de venta de entradas altamente configurable, ofrecer funcionalidades útiles que den valor a la aplicación y proporcionar información y estadísticas interesantes a los usuarios.

Para alcanzar este objetivo fue necesario en primer lugar realizar un estudio de la tecnología que se iba a utilizar en el desarrollo de la aplicación. En segundo lugar se hizo un análisis preliminar de la aplicación, realizando modelos de diseño como son el modelo de datos o prototipos de pantallas. En tercer lugar se definió la planificación en el desarrollo de la aplicación. Por último lugar se procedió al desarrollo y prueba de la aplicación.

Durante el desarrollo se creó un servidor con Java, utilizando el framework Spring para implementar la lógica de la aplicación. El servidor implementa un servicio REST que es consultado por la aplicación cliente para la gestión de los datos. El cliente se programó con el framework VueJS, usando los lenguajes HTML, JS y CSS. Para la persistencia de los datos se utilizó un SGBD PostgreSQL.

El trabajo de fin de grado se gestionó siguiendo una metodología basada en Scrum para el desarrollo software.

Abstract

The objective of this end-of-degree project is to develop an application for event management, with a highly configurable selling of tickets process, offer useful functionalities that give value to the application and provide interesting information and statistics to the users.

In order to achieve this goal, it was necessary first of all to make a study of the technology that was used for the application development, secondly a preliminary analysis of the application was performed, making design models, like data model or mock-ups. Then it was defined the application development planning. Finally it was done the application development.

During the development it was created a server using Java, utilizing the Spring framework to implement the application logic. The server implements a REST service that is called by the client application for the data management. The client is programmed with the VueJS framework, using the languages HTMl, JS and CSS. For the data persistence it was used a PostgreSQL DBMS.

The end-of-degree work was managed following a Scrum based methodology for software development.

Palabras clave:

- Aplicación de venta de entradas de eventos
- Página web
- Eventos de alta demanda
- Altamente configurable
- Configuración de eventos personalizable
- Reventa de entradas
- Estadísticas de usuario

Keywords:

- Application for event management
- Web page
- High demand events
- Highly configurable
- Customizable event settings
- Tickets resale
- User statistics

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1 Motivación

La venta de entradas para eventos a través de Internet está a la orden del día. Hay varias plataformas conocidas dedicadas a ello, y además hay plugins de aplicaciones más genéricas, como Wordpress u otros CMSs, que permiten vender entradas de eventos a pequeña escala. Esto suele ser suficiente para la mayor parte de los casos. Sin embargo, cuando el evento a organizar es muy popular, la mayor parte de alternativas existentes presentan varios problema. Por ejemplo, incluso las plataformas más conocidas tienen problemas de escalabilidad en los eventos más multitudinarios, no permiten demasiada configuración a la hora de decidir el método de venta de entradas, o tienen unas comisiones bastante altas a las que hay que sumar gastos por la devolución de las entradas en caso de cancelación del evento. Otra alternativa es utilizar una plataforma de venta de entradas de código abierto, como Attendize[5], pero entonces nos encontramos con que las opciones de configuración para cada evento son bastante limitadas.

A la hora de vender entradas, es posible que nos interese que las entradas se pongan a la venta a una hora determinada y que se vendan en orden de llegada. O quizás preferimos tener un periodo de preinscripción y un sorteo posterior para decidir el orden de venta. O un sistema de colas que sólo permita a un número de usuarios acceder a la página de compra, mientras el resto se quedan esperando en orden. Por otro lado podemos querer entradas nominales o no. También es posible que el precio de la entrada deba variar en función del número de entradas vendidas, o de plazos temporales. Todas estas son opciones que cualquier promotor querría poder configurar a la hora de crear un evento, en función de las expectativas del mismo, ya que pueden solucionar o aliviar problemas especialmente para los eventos más populares. Sin embargo, no es habitual tener unas opciones de configuración tan avanzadas en las plataformas o plugins existentes. Por último, otro problema habitual que debería tener fácil solución es la reventa de entradas, que podría ser controlada por el propio promotor,

facilitando la reventa "legítima", aún con entradas nominales, y prohibiendo cualquier otro tipo de reventa.

Este Trabajo de Fin de Grado consiste en el desarrollo de una plataforma para la venta de entradas que permita opciones y funcionalidades avanzadas, como las que hemos mencionado en el párrafo anterior, y que permitirá a los promotores decidir el método de venta de entradas entre un abanico de opciones, así como permitir la reventa legal de entradas del evento.

1.2 Objetivos

El objetivo de este Trabajo de Fin de Grado es el desarrollo de una aplicación de venta de entradas altamente configurable en el proceso de venta de entradas, este objetivo principal se puede dividir en los siguientes objetivos más concretos:

- Ofrecer tres tipos de usuarios, que pueden realizar diferentes funcionalidades: usuario registrado, usuario promotor y usuario administrador.
- Proporcionar una aplicación en la que se pueda, tanto comprar entradas de eventos (usuario registrado), como promover eventos (usuario promotor de eventos), así como administrar la aplicación (usuario administrador).
- Facilitar a los promotores de eventos una aplicación con un proceso de venta de entradas altamente configurable. Éstos pueden elegir el método de venta de entradas de sus eventos dependiendo de las características que tiene su evento.

  También pueden elegir parámetros importantes como la fecha en la que se abre la venta de las entradas o el número de entradas máximo que puede comprar cada usuario.
- Poner a disposición de los usuarios registrados la opción de revender sus entradas compradas, directamente desde la aplicación.
- Proveer a los usuarios registrados de la posibilidad de guardar sus eventos favoritos en un página donde pueden acceder a ellos cuando quieran.
- Visualización de una página propia de cada usuario donde se muestra información sobre sus entradas compradas (usuario registrado) o sobre sus eventos (usuario promotor).
- Disponibilidad de una página donde se muestran unas estadísticas sacadas de la actividad en la aplicación, tanto para los usuarios registrados como para los usuarios promotores.
- Posibilidad de administrar la aplicación, pudiendo gestionar eventos, administrar usuarios, o administrar recintos donde se celebran los eventos.

Capítulo 2

Fundamentos tecnológicos

2.1 Estado del arte

Se han buscado aplicaciones existentes que cumplan los objetivos descritos en el apartado anterior, y no hay ninguna que los reúna todos. Algunas de ellas tienen funcionalidades similares: Ticketea, Ticketbud y Attendize.

A continuación se explica brevemente las características básicas de cada una de estas aplicaciones:

Ticketea (Eventbrite) [6]: es una aplicación que permite a los usuarios buscar, crear, promover eventos y comprar entradas para los mismos. En el caso de que los eventos promovidos no sean gratuitos se cobra una tarifa. Una de sus ventajas es la opción de elegir entre una variedad de herramientas promocionales: widgets a enlaces personalizados, invitaciones, redes sociales, etc. También ofrece un buscador de eventos muy completo que permite adaptar los eventos que ven los usuarios según sus preferencias. En la Figura 2.1 se puede ver la página de búsqueda de eventos.

Las principales desventajas de la aplicación son: la ausencia de ningún tipo de estadística de usuario y la falta de personalización de los eventos en términos de formas de venta de entradas.

Ticketbud [7]: es una aplicación de venta de entradas y promoción de eventos similar a Ticketea. Tiene un amplio soporte de funcionalidades para organizar eventos de manera sencilla, cobra un porcentaje por las entradas vendidas. Una de las ventajas con respecto a Ticketea es que el ingreso del dinero por la venta de las entradas se hace automáticamente cada vez que se venden las entradas, así las ganancias que se hayan recaudado

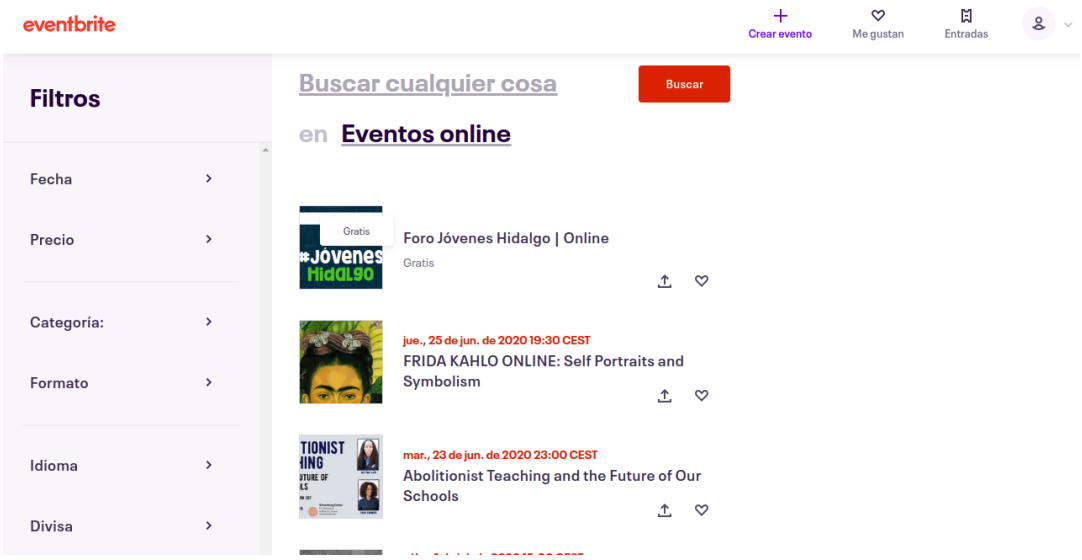

Figura 2.1: Página de búsqueda de eventos

se obtienen antes del comienzo del evento. En la Figura 2.2 se puede ver la página de búsqueda de eventos.

Las principales desventajas de esta aplicación son las siguientes: tiene un buscador de eventos muy simple, el cual solo permite buscar eventos por nombre, no ofrece unas estadísticas muy completas a los usuarios y no permite la configuración de ningún método de venta de entradas en la creación de los eventos.

Attendize [5]: es una aplicación open-source de venta de entradas y promoción de eventos. Tiene una amplio soporte de funcionalidades para organizar eventos de manera sencilla. Ofrece estadísticas a los usuario promotores de eventos que pueden ser útiles. Una de las ventajas con respecto a Ticketea y Ticketbud es que es open-source y no hay que pagar comisiones, además de que estas anteriores no ofrecen ninguna estadística a los usuarios. En la Figura 2.3 se puede ver la página de creación de eventos.

La principal desventaja de la aplicación es la falta de configuración de métodos de venta de entradas.

Una vez vistas las funcionalidades de estas aplicaciones, se puede concluir que ninguna de ellas cumple los objetivos del proyecto, pero sin embargo sí podemos tomarlas como referencia para aprovechar sus bondades. Por ejemplo, intentaremos basarnos en la funcionalidad de búsqueda de eventos de Ticketea, ya que destaca por ser un buscador bastante completo en cuanto a parámetros de búsqueda, de Attendize tomaremos como referencia las estadísticas que ofrecen a los usuarios. Además se añadirán opciones de configuración de métodos de

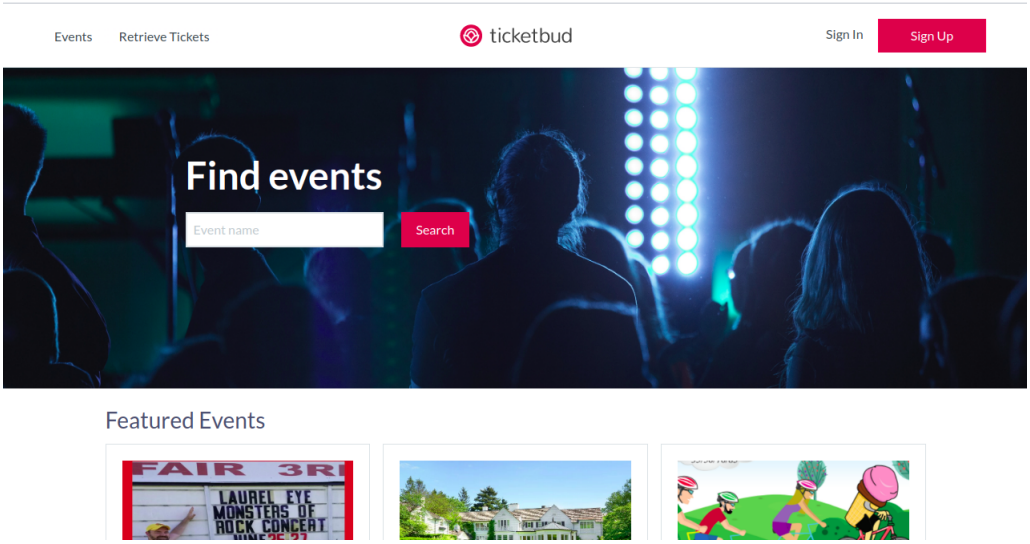

Figura 2.2: Página de búsqueda de eventos

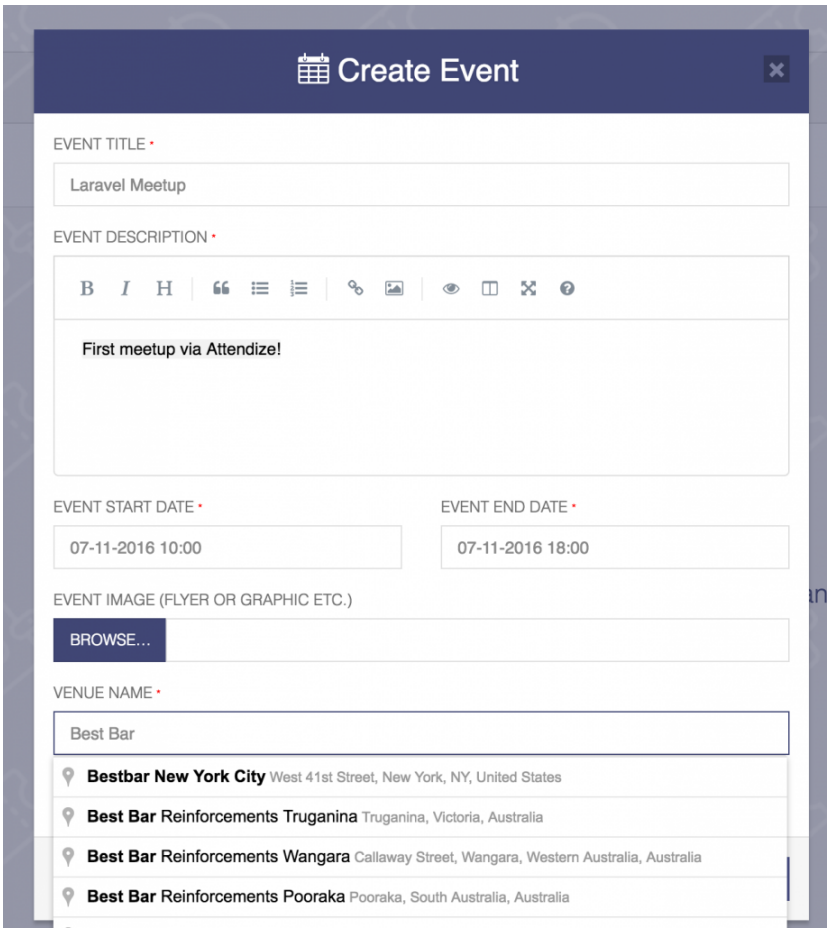

Figura 2.3: Página de creación de eventos

venta de entradas de eventos así como mejorar y crear alguna funcionalidad nueva que estas aplicaciones no contemplan y que pueden ser provechosas.

2.2 Tecnologías utilizadas

Las tecnologías que se han utilizado en el desarrollo de la aplicación son las siguientes:

Vue.js [8, 9]. Es un framework progresivo para construir interfaces de usuario. Su núcleo es bastante pequeño y se escala a través de plugins. Engloba en un mismo archivo HTML, CSS y JavaScript.

Spring Boot [10, 11]. Es un framework que elimina la mayor parte del trabajo de configurar las aplicaciones basadas en Spring. Permite compilar nuestras aplicaciones Web como una aplicación Java normal, integrando el servidor de aplicaciones y levantándolo cuando iniciamos la aplicación. De esta forma, podemos distribuir nuestras aplicaciones de una forma mucho más sencilla.

JavaScript [12]. Es un lenguaje de programación ligero e interpretado, orientado a objetos. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Hibernate [13]. Es un framework que permite el mapeo objeto-relacional en el lenguaje de programación Java. Realiza el mapeo de objetos Java a una base de datos relacional, gracias a una serie de anotaciones incluidas en las clases que indican como se deben establecer estas relaciones.

PostgreSQL [14]. Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

Spring [15, 16]. Es un framework para el desarrollo de aplicaciones Java. Spring proporciona soporte de infraestructura a nivel de aplicación, permitiendo que los usuarios se centren en la lógica de negocio de las aplicaciones.

Vuetify [17]. Es un framework para el diseño de interfaces web con Vue. Facilita componentes reutilizables ya listos para utilizar directamente en la aplicación.

HTML (HyperText Markup Language) [18]. Es un lenguaje de marcas que se utiliza para la creación de páginas web.

CSS (Cascading Style Sheets) [19]. Es un lenguaje de diseño gráfico para páginas web, se usa conjuntamente con HTML.

Heroku [20]. Es una plataforma PaaS (Plataforma como servicio) que permite a los desarrolladores construir, ejecutar y operar aplicaciones en la nube programadas en diferentes lenguajes.

Capítulo 3

Metodología y planificación

3.1 Metodología de desarrollo

Durante el desenvolvimiento de la aplicación se ha seguido una metodología incremental guiada por funcionalidades, basada en Scrum [21]. El desarrollo incremental consiste en dividir el proyecto en iteraciones en las que se implementan una serie de funcionalidades. Algunas de las razones por las que se escogió esta metodología son: la flexibilidad ante cambios durante el desarrollo del proyecto, la posibilidad de proporcionar un producto funcional al final de las iteraciones, mejor estimación del tiempo que va a llevar el proyecto al dividir las funcionalidades en iteraciones, etc.

Scrum consiste en un marco de trabajo para el desarrollo ágil de software. Se basa en un desarrollo incremental formado por *Sprints*. Un *Sprint* consiste en un período de tiempo en el que se implementan una serie de funcionalidades que posteriormente son revisadas y finalmente producen un producto funcionable que se puede entregar al cliente.

Las fases de Scrum que se han seguido durante el desarrollo del proyecto se explican a continuación. En la Figura 3.1 se puede ver el marco de trabajo de Scrum.

- **Planificación del *Sprint***: Etapa en la que se acuerda con el equipo de Scrum las funcionalidades que se van a implementar en dicho *Sprint*, además de los plazos de tiempo para finalizarlo.
- ***Sprint* o etapa de desarrollo**: Tiempo en el que se lleva a cabo la implementación del trabajo acordado en la fase anterior.
- **Revisión del *Sprint***: Reunión del equipo de Scrum para revisar y analizar el trabajo del *Sprint* finalizado.
- **Retrospectiva del *Sprint***: Lecciones aprendidas durante el desarrollo del *Sprint*. Los miembros del equipo dejan sus impresiones sobre el *Sprint* finalizado para recalcar las cosas bien hechas y las que se pueden mejorar.

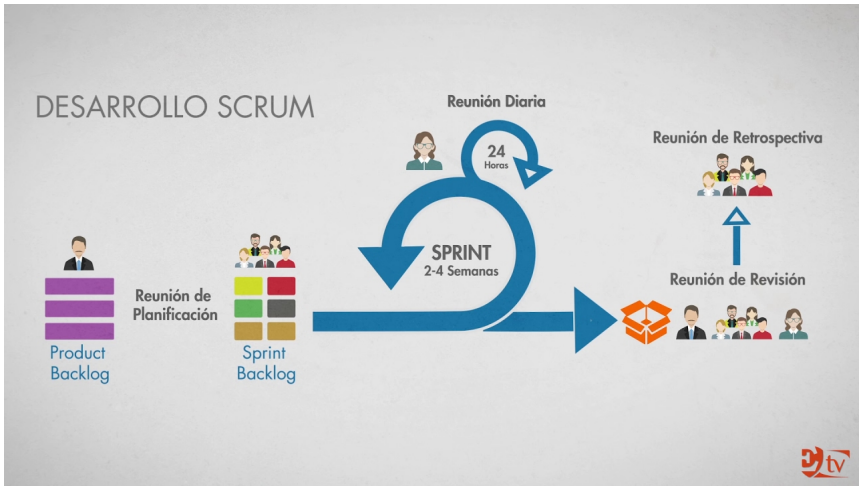

Figura 3.1: Marco de trabajo de Scrum [1]

Los actores o roles de Scrum que se han jugado durante el desarrollo de la aplicación se enumeran posteriormente. En la Figura 3.2 se pueden ver los diferentes roles que existen en Scrum.

- **Product Owner (Dueño de producto)**: El Product Owner es el encargado que el equipo de Scrum trabaje de forma adecuada desde la prespectiva de negocio. Marca los objetivos de los *Sprints* y las funcionalidades que se tienen que implementar en cada uno de ellos priorizando el orden según su importancia. Este rol es jugado por los directores del Trabajo de Fin de Grado.
- **Scrum Master**: El Scrum Master se dedica a gestionar el proceso Scrum, es la persona que examina que no haya impedimentos a la hora del desarrollo del proyecto. En este caso se corresponde con los directores del Trabajo de Fin de Grado.
- **Equipo de Desarrollo**: El Equipo de Desarrollo es el que desarrolla el producto final. En este caso el equipo de desarrollo estuvo formado por un único miembro, el autor del trabajo.

3.2 Planificación y seguimiento

En este apartado se van a contar las fases del proyecto que se concretaron durante la planificación y a continuación se estimará el tiempo de trabajo en el proyecto y se mostrarán las desviaciones del mismo.

En primer lugar, las tareas que se definieron durante la planificación y que se siguieron durante el desarrollo del proyecto son las siguientes:

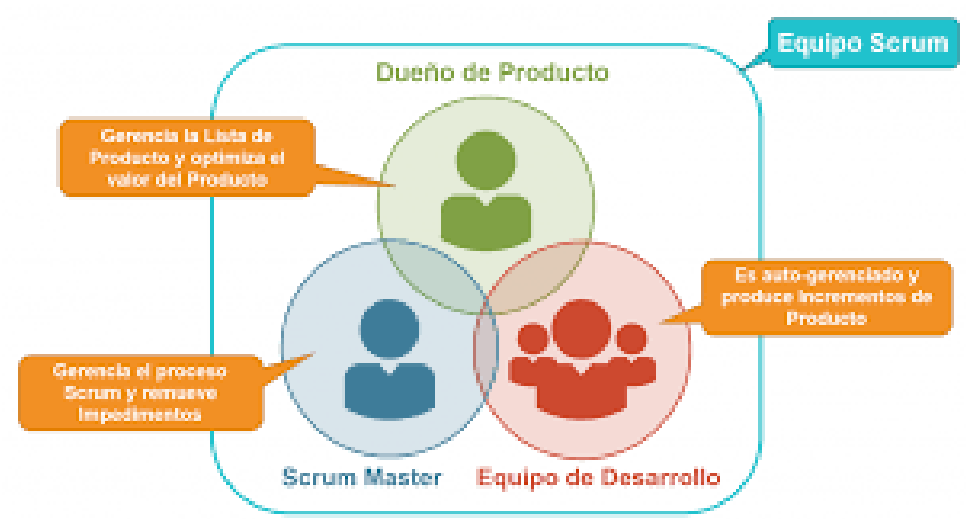

Figura 3.2: Roles de Scrum [2]

- **Análisis preliminar**: Reuniones en las que se decidió lo que se iba a hacer, qué herramientas se iban a usar en el desarrollo y concretar el alcance del proyecto. Dentro de esta fase de realizaron diferentes diagramas:
  - **Diagrama de UML con el modelo de datos**: se diseñó la estructura de la base de datos, haciendo un diagrama entidad relación con los datos que iba a manejar la aplicación.
  - **Prototipos de pantallas**: se crearon maquetas de la pantallas que iba a tener la aplicación.
  - **Definir las funcionalidades**: se concretaron las funcionalidades de la aplicación o historias de usuario de la misma. Una historia de usuario es un requisito de la aplicación escrito en lenguaje coloquial del cliente. Más tarde, con estas funcionalidades, se organizaron las iteraciones que iba a tener el desarrollo.
  - **Definir el servicio REST**: se determinaron las operaciones que iban a ser necesarias en el servicio REST.
- **Estudio de la tecnología**: En primer lugar se realizó un estudio de las tecnologías existentes para tener conocimiento de su funcionamiento. Posteriormente, se decidió usar una serie de alternativas, que finalmente resultaron en las tecnologías que se usaron en el desarrollo del proyecto (definidas en el Apartado 2.2).
- **Planificación**: Se decidió la metodología de desarrollo que se utilizaría durante el proyecto, contada en el Apartado 3.1, también se planificó el número de iteraciones que

iba a tener el proyecto. Decidimos dividir el tiempo del que se disponía para realizar el proyecto. Se consideró la duración de cada *Sprint* en 3 semanas y dado que se disponía de 18 semanas hasta la finalización del proyecto, se dividieron las semanas disponibles entre la duración de cada *Sprint*, dando el resultado de 6 *Sprints* de 3 semanas. Una vez calculado esto, se dividieron las historias de usuario, las cuales eran 68, entre el numero de *Sprints*, obteniendo así 12 historias de usuario por *Sprint* para desarrollar.

Una vez reunidas todas las historias de usuario, se hizo uso de Gitlab (servicio web de control de versiones y desarrollo de software que utiliza Git) [22], para realizar la planificación. Para organizar el trabajo usamos un artefacto propio de Gitlab llamado *Issue*. Una *Issue* es elemento utilizado en Gitlab para planificar el trabajo a realizar en un proyecto, cada *Issue* es una funcionalidad o tarea que se debe hacer. Se crearon tantas *Issues* como historias de usuario hubiera, y se agruparon todas las *Issues* en *Milestones*. Un *Milestone* es un elemento utilizado en Gitlab al igual que las *Issues*. Se usa para hacer el seguimiento de un conjunto de *Issues* que tienen que ser implementadas en un cierto período de tiempo. Se crearon tantos *Milestones* como Sprints, de esta manera se agruparon varias *Issues* en un *Milestone* y se le estableció una fecha de inicio y de fin, para la cual deben estar finalizadas dichas *Issues*, dividiendo de esta manera los *Sprints*.

- **Desarrollo**: Período de tiempo en el que se implementaron las funcionalidades de la aplicación, clasificadas en *Sprints*, que se definieron en la fase de **Planificación**. La implementación de las funcionalidades de estos *Sprints* se hizo siguiendo el marco de trabajo Scrum, el cual fue descrito en el Apartado 3.1.

- **Seguimiento**: Se apuntaron las horas invertidas en el desarrollo de cada historia de usuario, para poder obtener todas las horas que se dedicaron a la realización del proyecto. Al final de cada *Sprint* se hicieron revisiones del trabajo realizado en dicho *Sprint*.

  En la Figura 3.3 y en la Figura 3.4 se muestra una tabla con las fases del proyecto, su duración estimada, la duración real y con todas las tareas que se realizaron en dichas fases, con el tiempo de trabajo en cada una de ellas. También se muestra un pequeño Diagrama de Gantt en la Figura 3.5, mostrando gráficamente la duración real y la estimada de las fases.

- **Documentación del trabajo realizado**: Fase que se corresponde con la redacción de esta memoria donde se documenta el trabajo realizado durante el proyecto.

Fases del proyecto	Fecha Inicio	Fecha Fin	Duración Real(días)	Duración Estimada(días)	Duración Estimada – Real(días)	Tareas	Duración(horas)
						Diagrama de UML con el modelo de datos	40
Análisis Preliminar	25/11/19	15/01/20	17.5	27	9.5	Mockups o prototipos de pantallas	56
						Definir las funcionalidades	28
						Definir el servicio REST	16
Estudio tecnología	25/11/19	15/01/20	5	10	5	Estudio de la tecnología	40
Planificación	15/01/20	20/01/20	0.75	2	1.25	Planificación	6
						Login como administrador	4
						Administrar recintos	2
						Crear recintos	2
						Borrar recintos	2.5
						Editar información de un recinto	2
						Crear zonas en un recinto	6
Desarrollo Sprint 1	20/01/20	09/02/20	6.4	14	7.6	Borrar zonas de un recinto	3
						Editar zonas de un recinto	6
						Crear configuraciones de recinto	3
						Añadir zonas a una configuración de un recinto	10.7
						Quitar zonas a una configuración de un recinto	3.5
						Borrar configuraciones de recintos	0.5
						Editar configuraciones de recinto	5
						Logout como administrador	1
						Registrarse como usuario Registrado o Promotor	4
						Login como usuario Registrado o Promotor	6
						Cancelar eventos	2
						Crear un evento	16
						Configurar las entradas en un recinto	12
						Elegir el método de venta de mis entradas	8
Desarrollo Sprint 2	09/02/20	29/02/20	11.75	14	2.25	Administrar eventos	4
						Borrar eventos	4
						Editar eventos	16
						Administrar los tipos de entradas de un evento	8
						Configurar los tipos de entradas	4
						Borrar los tipos de entradas de un evento	4
						Editar los tipos de entradas de un evento	6
						Elegir un evento, para poder ver su información	4
						Ver información de tipos de entradas de un evento	4
						Ver un mapa del recinto del evento	2
						Registrarse como usuario Registrado	2
						Poder guardar los eventos en favoritos	4
Desarrollo Sprint 3	29/02/20	20/03/20	8.25	14	5.75	Comprar mis entradas de eventos	10
						Comprar entradas en reventa	6
						Revender entradas	14
						Ver información sobre las entradas compradas	4
						Ver información de los eventos a los que voy a asistir	2
						Puntuar un evento al que he asistido	10
						Ver los eventos que tengo guardados en favoritos	4

Figura 3.3: Tabla de seguimiento primera parte

Fases del proyecto	Fecha Inicio	Fecha Fin	Duración Real(días)	Duración Estimada(días)	Duración Estimada – Real(días)	Tareas	Duración(horas)
						Como usuario Administrador, buscar los eventos por nombre	8
						Como usuario Administrador, buscar los eventos por usuario	8
						Como usuario Administrador, buscar los eventos por recinto	8
						Como usuario Administrador, buscar los eventos por lugar	8
						Como usuario Administrador, buscar los eventos por fecha	8
Desarrollo Sprint 4	20/03/20	09/04/20	10.5	14	3.5	Como usuario Administrador, administrar los usuarios	16
						Como usuario Administrador, buscar usuarios por nombre	4
						Como usuario Administrador, buscar usuarios por e-mail	4
						Como usuario Administrador, buscar usuarios por tipo	4
						Como usuario Administrador, poder borrar usuarios	6
						Como usuario Administrador, poder editar los usuarios	6
						Como usuario Administrador, poder buscar recintos por nombre	4
						Como usuario Administrador, poder buscar recintos por lugar	4
						Cualquier tipo de usuario puede entrar en la página web	3
						Como usuario Anónimo, quiero ordenar los eventos por fecha	4
						Como usuario Anónimo, quiero ordenar los eventos por precio	4
						Como usuario Anónimo, quiero ordenar los eventos por porcetaje ventas	4
						Como usuario Anónimo, quiero buscar los eventos por localización	2
						Como usuario Anónimo, quiero buscar los eventos por fecha	2
						Como usuario Anónimo, quiero buscar los eventos por tipo de evento	4
						Como usuario Anónimo, quiero buscar los eventos por recinto	4
Desarrollo Sprint 5	09/04/20	29/04/20	8.625	14	5.375	Como usuario Anónimo, quiero buscar los eventos por nombre	2
						Como usuario Registrado, quiero visualizar mis datos personales	4
						Como usuario Registrado, quiero actualizar mis datos personales	4
						Como usuario Registrado, quiero ver las estadísticas de mi actividad	10
						Como usuario Promotor, quiero acceder a mi perfil	4
						Como usuario Promotor, quiero actualizar mis datos personales	4
						Como usuario Promotor, quiero ver las entradas de mis eventos	4
						Como usuario Promotor, quiero ver la información de mis eventos	4
						Como usuario Promotor, quiero ver las estadísticas de mi actividad	6
						Sistema de venta de entradas cola	16
Desarrollo Sprint 6	29/04/20	19/05/20	12	14	2	Sistema de venta de entradas inscripcion sorteo	36
						Sistema de venta de entradas lista de espera entradas agotadas	32
						Sistema de venta de entradas variacion precio entrada	12
Memoria	19/05/20	01/09/20	12	53	41	Redacción de la memoria del Trabajo de Fin de Grado	96

Figura 3.4: Tabla de seguimiento segunda parte

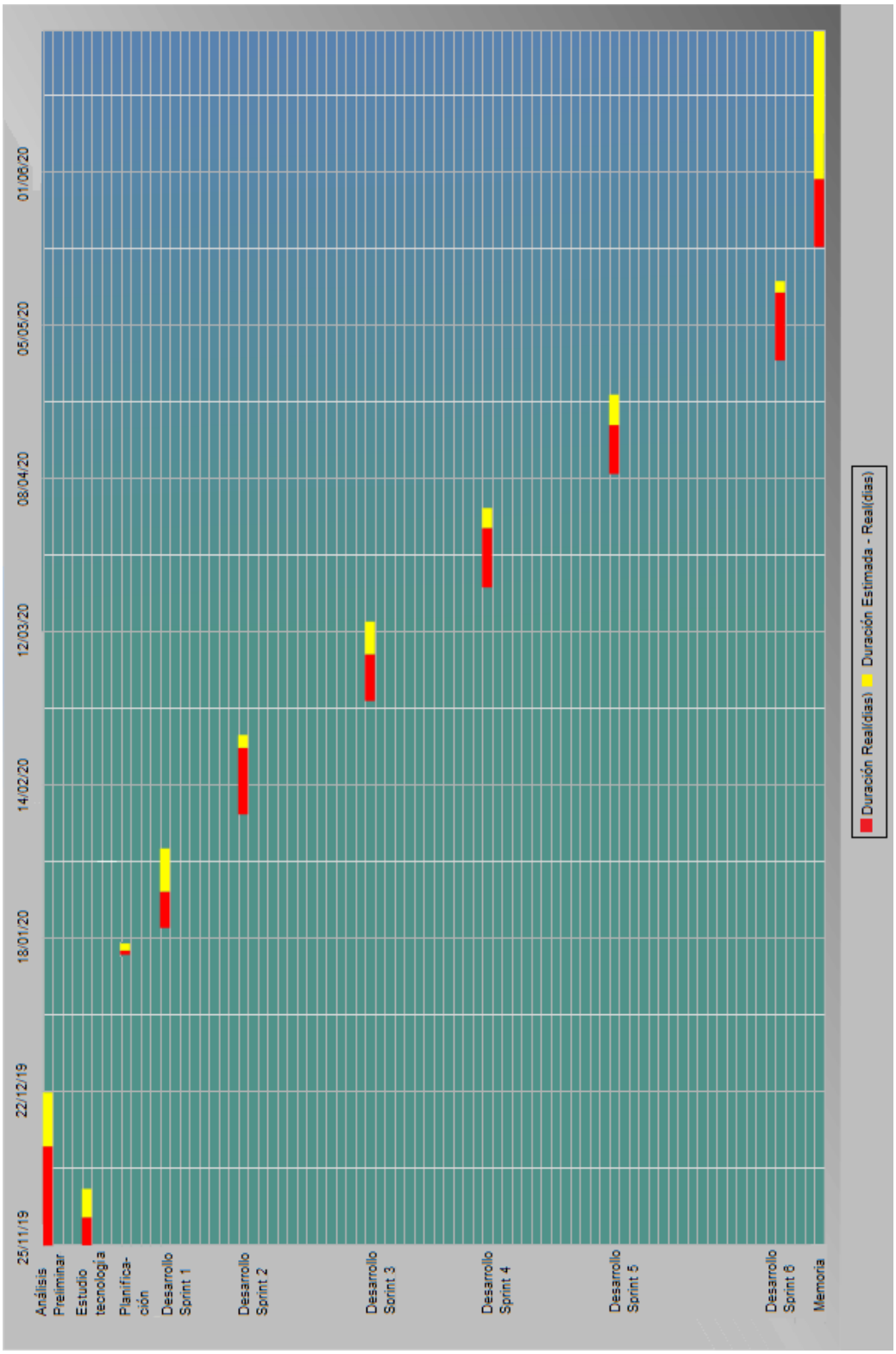

Figura 3.5: Diagrama de Gantt

Capítulo 4

Análisis

4.1 Tipos de usuarios

La aplicación soporta diferentes tipos de usuarios que realizan diferentes funcionalidades. En este apartado se describen los tipos de usuarios que tiene el proyecto:

- **Usuario No Registrado**: Usuario anónimo no autenticado en la aplicación que puede acceder, desde la página principal de la aplicación, a los distintos eventos a través del buscador de los mismos. Además, puede registrarse y hacer login en la aplicación tanto como promotor de eventos como usuario normal para comprar entradas.
- **Usuario Registrado**: Usuario registrado de la aplicación que puede compra entradas para eventos. Además de la compra de entradas también puede acceder a su perfil, poner entradas compradas en reventa, ver sus estadísticas, etc.
- **Usuario Promotor**: Usuario registrado en la aplicación como promotor de eventos. Puede gestionar sus propios eventos, crearlos, cancelarlos, etc. También puede acceder a una página donde se le muestran unas estadísticas de su actividad en la aplicación. Este tipo de usuario no puede comprar entradas para eventos.
- **Usuario Administrador**: Usuario administrador de la aplicación encargado de administrar la misma. Tiene acceso a todos los datos de la aplicación: usuarios, recintos, eventos, configuraciones de recintos, etc. Puede editar o borrar todos estos datos, y en el caso de los recintos, puede crear nuevos.

4.2 Historias de usuario

En este apartado se van a enumerar las historias de usuario de la aplicación, clasificándolas por el tipo de usuario que lleva a cabo la funcionalidad de la historia de usuario:

4.2.1 Historias de usuario del Usuario No Registrado

1. Quiero poder entrar en la página web, para poder acceder a las funcionalidades que ofrece la misma.

2. Quiero poder ordenar los eventos por fecha, para poder visualizar los eventos más recientes.

3. Quiero poder ordenar los eventos por precio, para poder visualizar los eventos más baratos.

4. Quiero poder ordenar los eventos por porcentaje de entradas vendidas hasta el momento, para poder ver los eventos más demandados.

5. Quiero poder buscar los eventos por la localización donde se van a llevar a cabo, para poder ver los eventos que más me interesan por localización.

6. Quiero poder buscar los eventos por fecha, para poder ver los eventos que se celebran en una fecha determinada.

7. Quiero poder buscar los eventos por el tipo de evento del cual se trata, para poder ver los eventos del tipo que me interesan.

8. Quiero poder buscar los eventos por el nombre del recinto en el cual se va a llevar a cabo, para poder ver todos los eventos que se van a llevar a cabo en dicho recinto.

9. Quiero poder buscar los eventos por nombre, para poder buscar directamente el evento que me interesa.

10. Quiero poder elegir un evento, para poder ver su información.

11. Quiero poder ver un mapa del recinto con las diferentes zonas a las que da acceso los diferentes tipos de entradas.

12. Quiero poder ver las entradas de un evento, para poder ver información de los diferentes tipos de entradas.

13. Quiero poder registrarme como usuario Registrado o Promotor.

14. Quiero hacer login como usuario Registrado o Promotor.

4.2.2 Historias de usuario del Usuario Registrado

15. Quiero poder comprar las entradas de un evento, requiriendo esto registrarse como usuario Registrado.
16. Quiero poder guardar los eventos en favoritos, para poder acceder a ellos de manera más sencilla.
17. Quiero comprar mis entradas de eventos publicados en la aplicación y poder elegir el tipo de entradas que me interesa.
18. Quiero comprar entradas puestas en reventa por otros usuarios.
19. Quiero poder revender las entradas que ya había comprado.
20. Quiero poder visualizar la información sobre las entradas que he comprado.
21. Quiero visualizar mis datos personales.
22. Quiero poder actualizar mis datos personales.
23. Quiero poder ver información de los eventos a los que voy a asistir.
24. Quiero poder poder puntuar un evento al que he asistido de manera numérica.
25. Quiero poder ver los eventos que tengo guardados en favoritos.
26. Quiero poder ver la información del evento que tengo guardado en favoritos.
27. Quiero poder ver una serie de estadísticas de las entradas y los eventos a los que he asistido como el número de entradas que he comprado, dinero gastado en la compra de entradas, tipos de eventos a los que más he asistido o lugar de celebración de eventos al que más he acudido.
28. Quiero poder hacer logout, para poder salir del usuario del cual estaba logueado.

4.2.3 Historias de usuario del Usuario Promotor

29. Quiero crear mis propios eventos, requiriendo esto registrarme como usuario Promotor.
30. Quiero poder acceder a mi perfil, para ver la información que tengo guardada en él.
31. Quiero poder actualizar mis datos personales de Mi perfil.
32. Quiero poder ver la información de las entradas de los eventos que he creado.
33. Quiero poder ver la información de los eventos que he creado.

34. Quiero poder ver una serie de estadísticas de mis eventos creados y de las entradas que he vendido como el número de entradas vendidas y el dinero recaudado por cada evento, el dinero ganado total, el número de entradas vendidas en total o el tipo de evento con el que más dinero he ganado.

35. Quiero poder cancelar los eventos, para que su celebración no se lleve a cabo.

36. Quiero poder crear un nuevo evento, pudiendo elegir el nombre del mismo, asignarle una imagen, escribir una texto descriptivo del evento, configurarle las fechas de inicio de venta de entradas así como las de inicio y fin del evento.

37. Quiero poder asignar al evento un recinto de entre los disponibles en la aplicación, y posteriormente, una configuración del recinto con unas zonas determinadas, en las que se pueda configurar los diferentes tipos de entradas con diferentes precios.

38. Quiero poder elegir el método de venta de mis entradas, pudiendo ser por orden de llegada, usando una cola en la que esperan los usuarios mientras otros hacen sus compras, poder inscribirse en un sorteo previo a la apertura de la venta de las entradas, en el que se elige el orden en el que los usuarios entrarán en la compra.

39. Quiero poder hacer logout, para poder salir del usuario del cual estaba logueado.

4.2.4 Historias de usuario del Usuario Administrador

40. Quiero poder loguearme como usuario Administrador, para poder administrar la página web.

41. Quiero administrar los eventos.

42. Quiero poder borrar eventos.

43. Quiero poder editar eventos.

44. Quiero poder administrar los tipos de entradas de un evento.

45. Quiero poder configurar los tipos de entradas de una configuración de un recinto en concreto.

46. Quiero poder borrar los tipos de entradas de un evento.

47. Quiero poder editar los tipos de entradas de un evento.

48. Quiero poder administrar los usuarios.

49. Quiero poder buscar usuarios por nombre.

50. Quiero poder buscar usuarios por e-mail.

51. Quiero poder buscar usuarios por tipo de usuario.

52. Quiero poder borrar usuarios.

53. Quiero poder editar la información de los usuarios.

54. Quiero poder administrar recintos.

55. Quiero poder buscar recintos por nombre.

56. Quiero poder buscar recintos por lugar.

57. Quiero poder crear un nuevo recinto, para poder celebrar un evento en él, pudiendo asignar el recinto al evento en la creación de este último.

58. Quiero poder borrar un recinto.

59. Quiero poder editar la información de un recinto.

60. Quiero poder crear zonas en un recinto.

61. Quiero poder borrar zonas de un recinto.

62. Quiero poder editar las zonas de un recinto.

63. Quiero poder crear configuraciones de recinto, en las que dicha configuración de recinto esté compuesta por un subconjunto de zonas del recinto que se pueden asignar a un evento en la creación del mismo.

64. Quiero poder añadir zonas a una configuración de un recinto.

65. Quiero poder quitar zonas a una configuración de un recinto.

66. Quiero poder borrar configuraciones de recinto.

67. Quiero poder editar configuraciones de recinto.

68. Quiero poder hacer logout, para poder salir del usuario del cual estaba logueado.

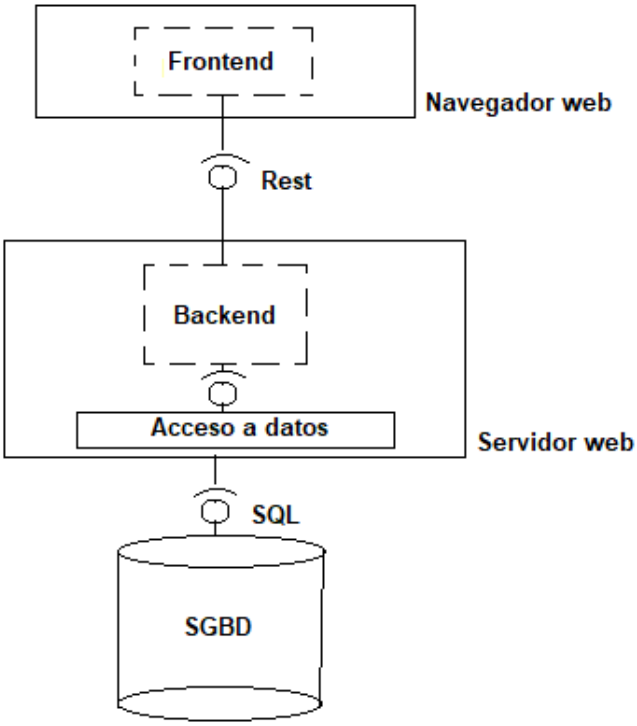

Figura 4.1: Aquitectura del sistema

4.3 Arquitectura del sistema

La arquitectura del sistema consiste principalmente en dos capas, la capa cliente y la capa servidor. En la Figura 4.1 se puede ver la misma. A continuación describe la arquitectura del sistema.

En primer lugar, en la capa más baja, tenemos una base de datos para la persistencia de los datos de la aplicación. Un nivel más arriba nos encontramos con la capa servidor, encargado de procesar los datos y almacenarlos en la base de datos. Esta capa se comunica con la base de datos mediante consultas en el lenguaje SQL para poder escribir y leer datos en ella. En el siguiente nivel se encuentra la capa cliente, encargada de mostrar la interfaz gráfica de la aplicación con la que interactúa el usuario y de comunicarse con la capa servidor para procesar los datos. Esta comunicación con la capa servidor se hace mediante el protocolo REST que está formado por recursos a los que están asignadas unas URL que son llamadas por la capa cliente para efectuar operaciones sobre los recursos.

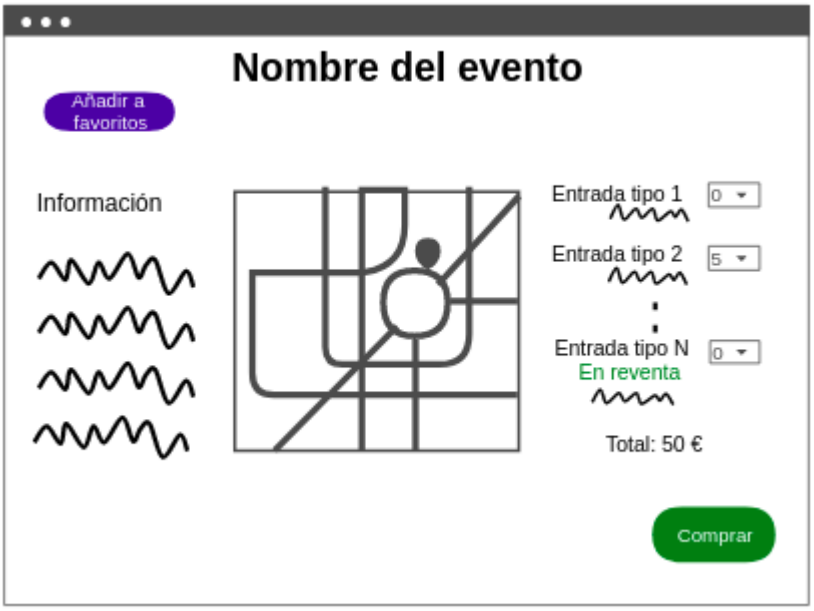

Figura 4.2: Compra de entradas

4.4 Interfaz de usuario

La estructura general de las pantallas de la aplicación se describe en este apartado. En concreto se enumeran y se describe la estructura de navegación entre las pantallas de la aplicación.

- En la Figura 4.2, se ve el proceso de compra de entradas. El usuario selecciona el número de entrada que quiere comprar, pulsa sobre el botón "Comprar" y por último se procede a cubrir información necesaria para la compra de las entradas.

- En la Figura 4.3, el usuario promotor accede a la página de sus eventos donde puede crear alguno nuevo, ver la información de los eventos que tiene creados, así como un historial de los que ya han finalizado. En concreto podemos ver como el usuario procede a la cancelación de uno de sus eventos.

- En la Figura 4.4, se observa el proceso de creación de un nuevo evento.

- En la Figura 4.5, se muestra como el administrador entra en la página de administración de eventos y edita uno de los eventos.

- En la Figura 4.6, el administrador entra en la página de administración de usuarios donde edita un usuario.

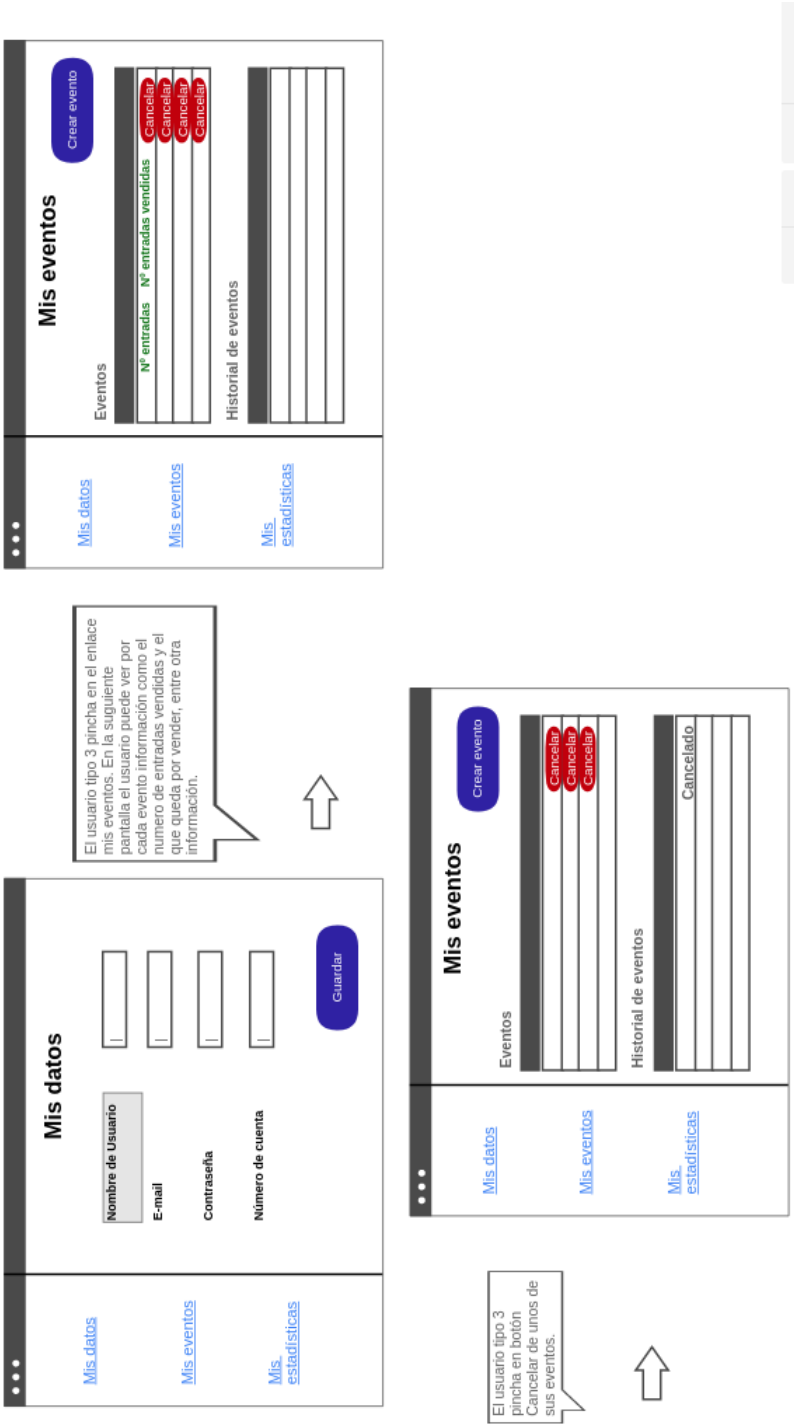

Figura 4.3: Cancelación de un evento por usuario promotor

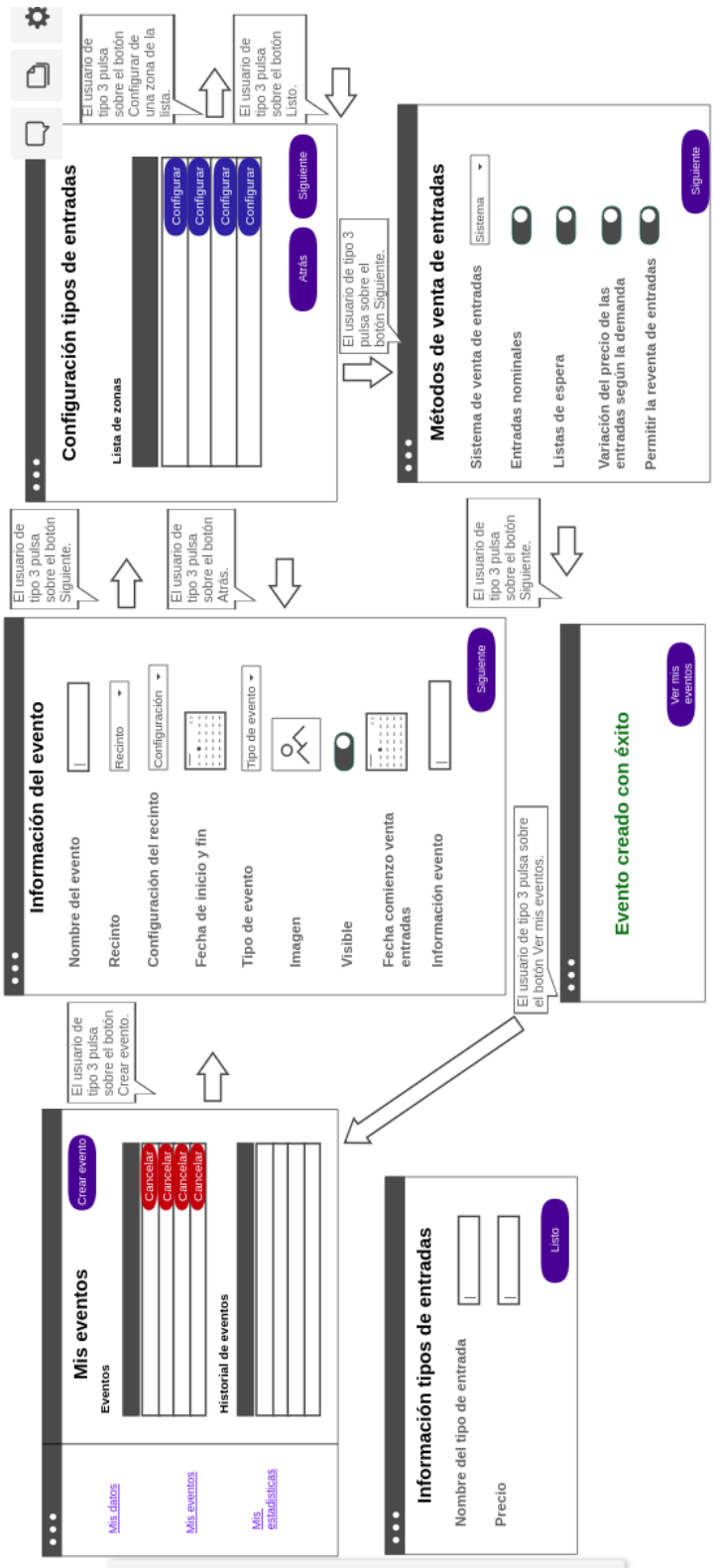

Figura 4.4: Creación de un evento

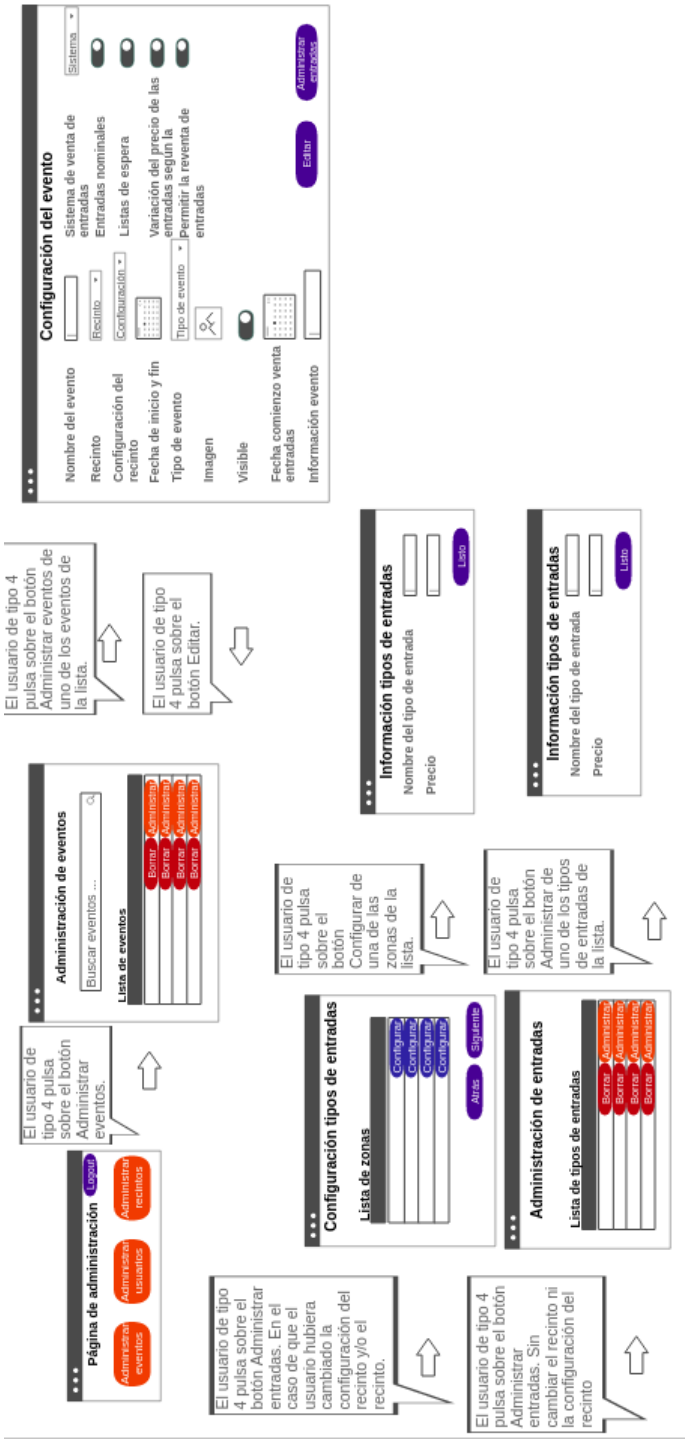

Figura 4.5: Página de administración de eventos

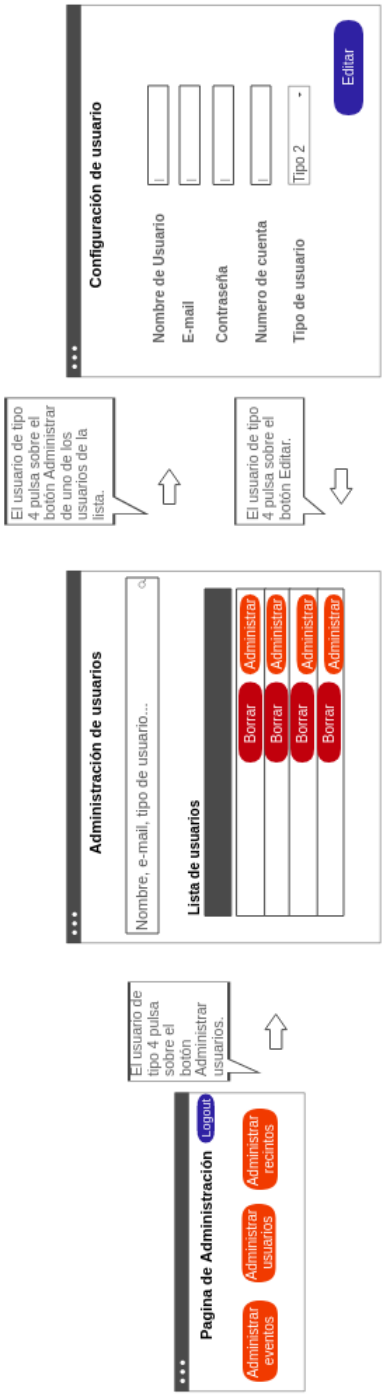

Figura 4.6: Página de administración de usuarios

- En la Figura 4.7, el administrador entra en la página de administración de recintos donde puede crear un nuevo recinto.
- En la Figura 4.8, el administrador entra en la página de administración de recintos donde se edita un recinto existente.

Se han creado más diseños para el resto de funcionalidades de la aplicación, como el login en la aplicación o la visualización de la pantalla de estadísticas, pero por cuestiones de espacio se muestran en el Apéndice A.

4.5 Modelo conceptual de datos

En esta sección se describe el modelo de datos de la aplicación, es decir, la estructura de la base de datos, las entidades y sus atributos. En la Figura 4.9 se muestra el diagrama de clases de los datos persistentes.

Usuario: Entidad representativa de todos los tipos de usuarios de la aplicación.

- *idUsuario*: identificador único de la entidad.
- *password*: contraseña de la cuenta del usuario.
- *nombre*: nombre de usuario de su cuenta.
- *email*: correo electrónico del usuario.

TipoUsuario: Entidad que en la que se guardan todos los tipos de usuario que existen.

- *idTipoUsuario*: identificador único de la entidad.
- *tipo*: tipo de usuario que representa la entidad.

Evento: Entidad que contiene los eventos de la aplicación.

- *idEvento*: identificador único de la entidad.
- *estado*: estado en el que se encuentra ese evento en ese momento
- *numMaxEntradasUsuario*: número de entradas máximo del evento que puede comprar un usuario.
- *numVotos*: número de votaciones de puntuación realizadas por los usuario en ese evento.
- *puntuacion*: suma de las puntuaciones dadas al evento por los usuarios.
- *nombre*: nombre del evento.

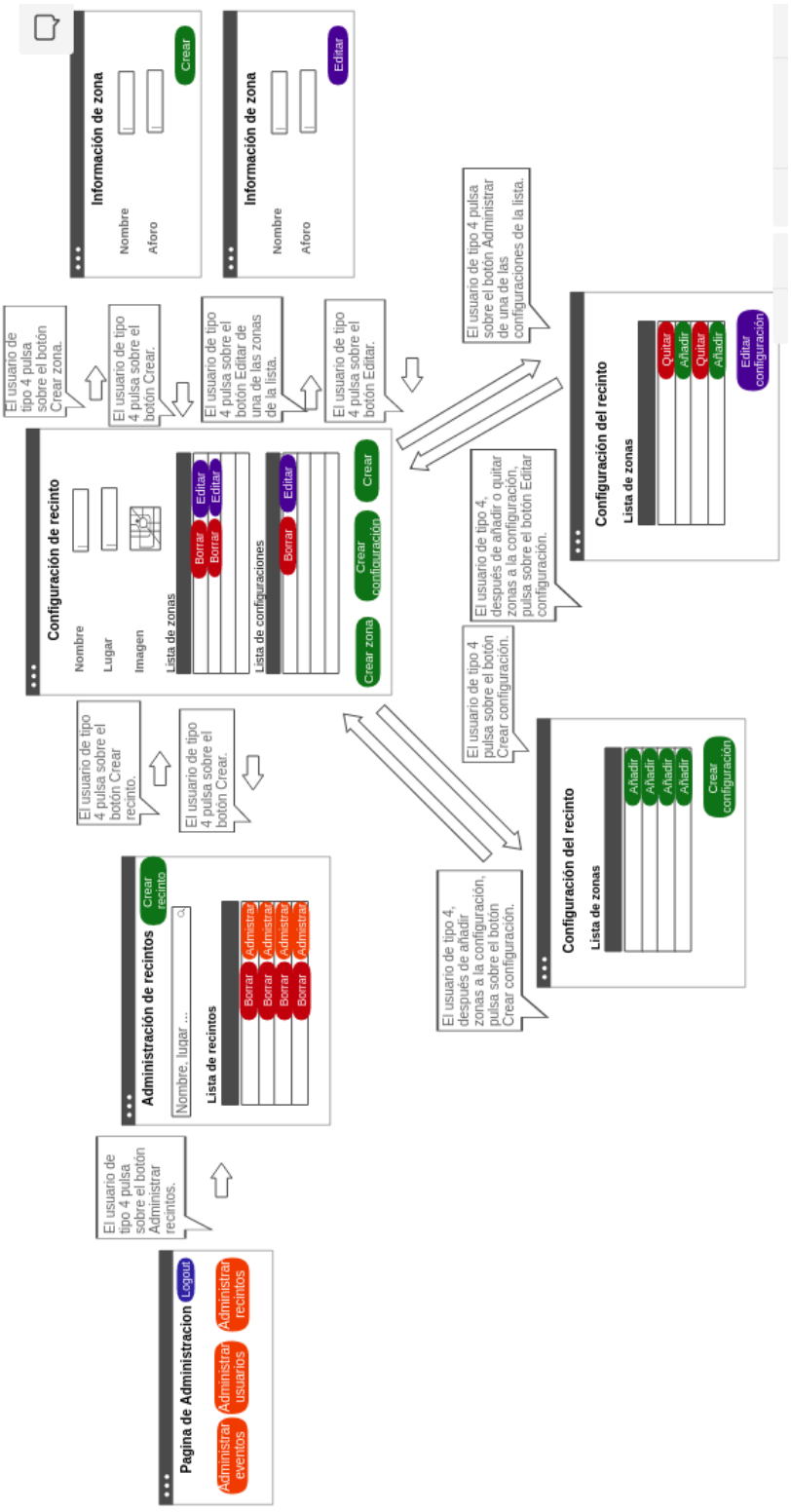

Figura 4.7: Creación de un recinto en la página de administración de recinto

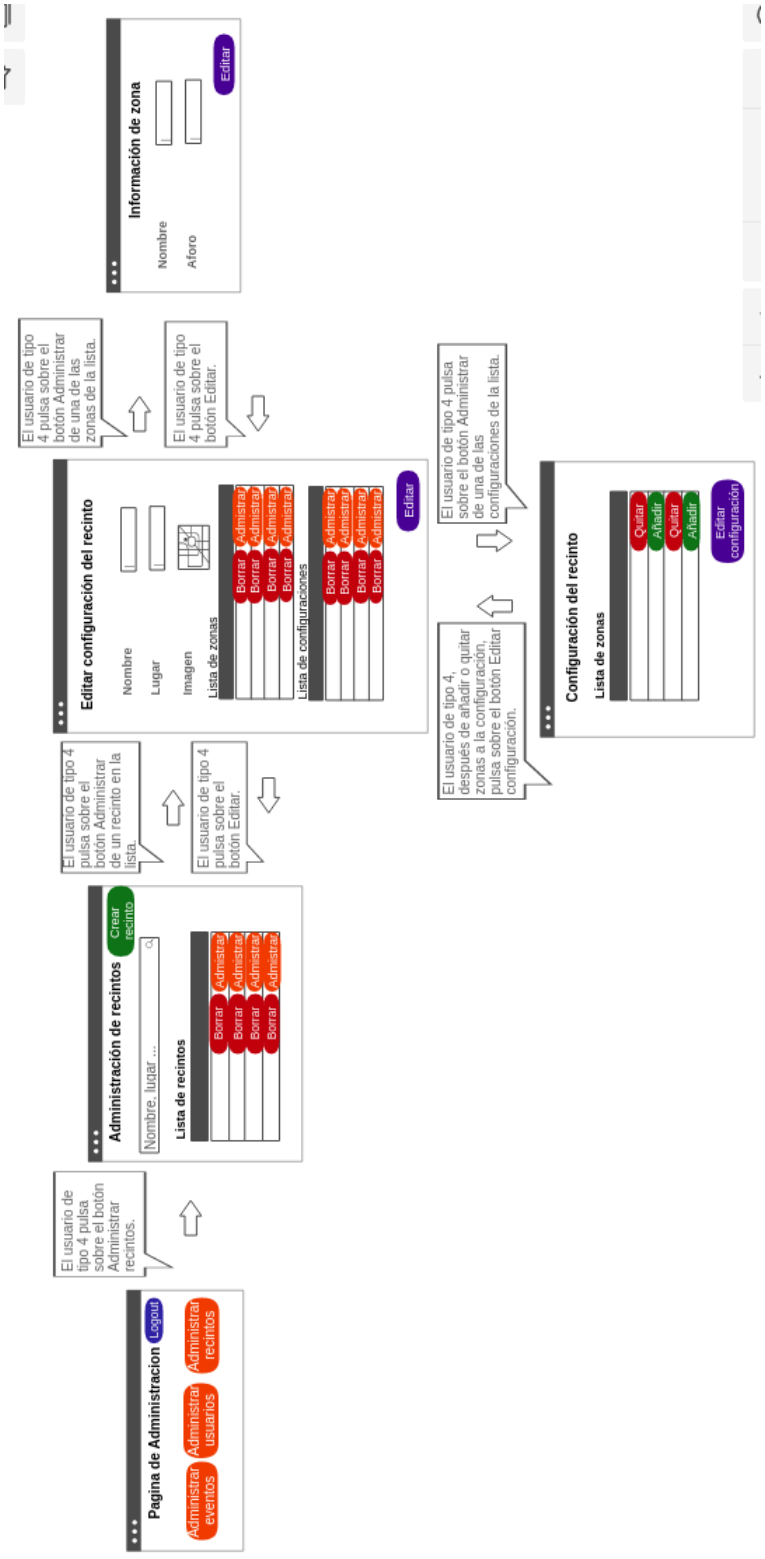

Figura 4.8: Edición de un recinto en la página de administración de recinto

- *visible*: valor booleano(puede tener dos valores verdadero/falso) que indica si el evento es visible para los demás usuarios.
- *info*: información general del evento.
- *fechaDeInicio*: fecha en la que comienza el evento.
- *fechaDeFin*: fecha de finalización del evento.
- *fechaVentaEntradas*: fecha en la que se abre el período de venta de entradas del evento.
- *imagen*: imagen representativa del evento.

TipoEvento: Entidad que muestra los tipos de eventos existentes en la aplicación.

- *idTipoEvento*: identificador único de la entidad.
- *nombre*: nombre del tipo de evento en cuestión.

Recinto: Entidad que simboliza los recintos de la aplicación.

- *idRecinto*: identificador único de la entidad.
- *nombre*: nombre del recinto.
- *localizacion*: lugar donde se encuentra el recinto.

ConfRecinto: Representa las configuraciones de recinto existentes.

- *idConfRecinto*: identificador único de la entidad.
- *imagen*: imagen que muestra el recinto con la configuración del recinto en cuestión.
- *nombre*: nombre de la configuración de recinto

Zona: Entidad que contiene las zonas de un recinto

- *idZona*: identificador único de la entidad.
- *nombre*: nombre de la zona del recinto.
- aforo: número del aforo de la zona.

Entrada: Representa las entradas de los eventos.

- *idEntrada*: identificador único de la entidad.
- *nombre*: nombre del usuario dueño de la entrada.

- *apellido*: apellidos del usuario dueño de la entrada.
- *dni*: dni del usuario dueño de la entrada.
- *puntos*: puntuación otorgada al evento de la entrada por el usuario dueño de la entrada.
- *precio*: precio de la entrada.

TipoEntrada: Entidad que contiene los tipos de entradas que existen en la aplicación.

- *idTipoEntrada*: identificador único de la entidad.
- *nombre*: nombre del tipo de entrada.

EntradaReventa: Representa las entradas que están en reventa.

- *idEntradaReventa*: identificador único de la entidad.
- *precio*: precio al que se vende la entrada en reventa.

ConfSisVentaEntrada: Entidad que recoge los sistema de venta de entradas habilitados en un evento.

SisVentaEntradas: Representa las entradas de los sistemas de venta de entradas que habilitarse en los eventos.

- *idSisVentaEntrada*: identificador único de la entidad.
- *nombre*: nombre del sistema de venta de entradas.

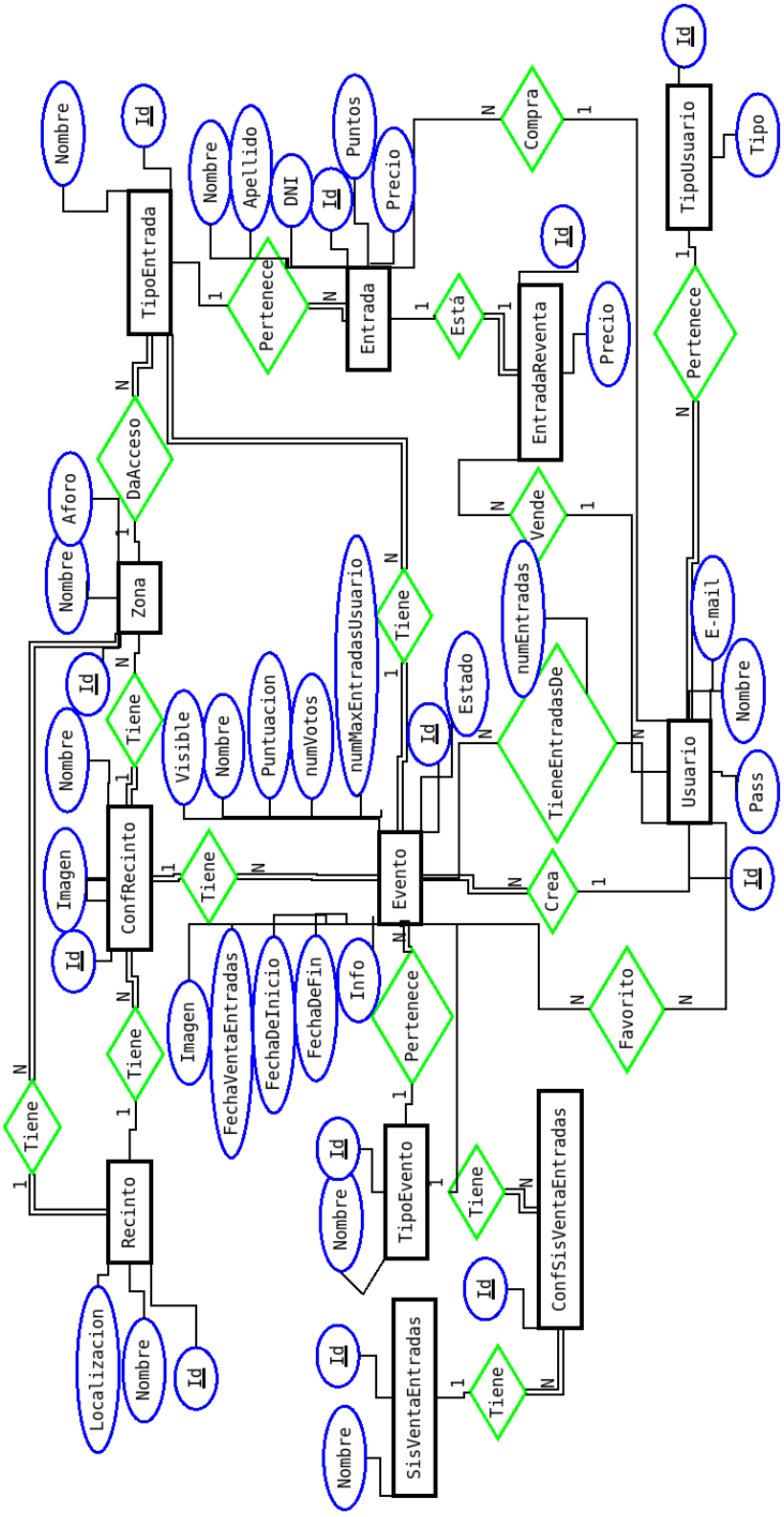

Figura 4.9: Modelo de datos

Capítulo 5

Diseño

5.1 Arquitectura tecnológica del sistema

En esta sección se va a explicar el diseño de la arquitectura tecnológica del sistema, describiendo las tecnológicas utilizadas en cada componente:

- En la capa de persistencia se utilizó **PostgreSQL** como sistema de gestión de base de datos.
- En la capa servidor se hizo uso del framework **Spring boot** y el lenguaje Java para su programación.
- **Hibernate** es la tecnología utilizada para la comunicación y el mapeo objeto-relacional entre la capa de persistencia y la capa servidor.
- En la capa cliente se hizo uso del framework **Vue** para programar la interfaz de la aplicación, utilizando en conjunto **JavaScript, HTML y CSS**.
- Para la comunicación, la manipulación y el intercambio de datos entre la capa servidor y la capa cliente se usó el protocolo **REST**.

En la Figura 5.1 se muestra una imagen de la arquitectura del sistemas con las tecnologías descritas anteriormente.

5.2 Diseño de la aplicación

Los detalles del diseño más complejos de la aplicación, tanto del lado servidor como del lado cliente se exponen seguidamente:

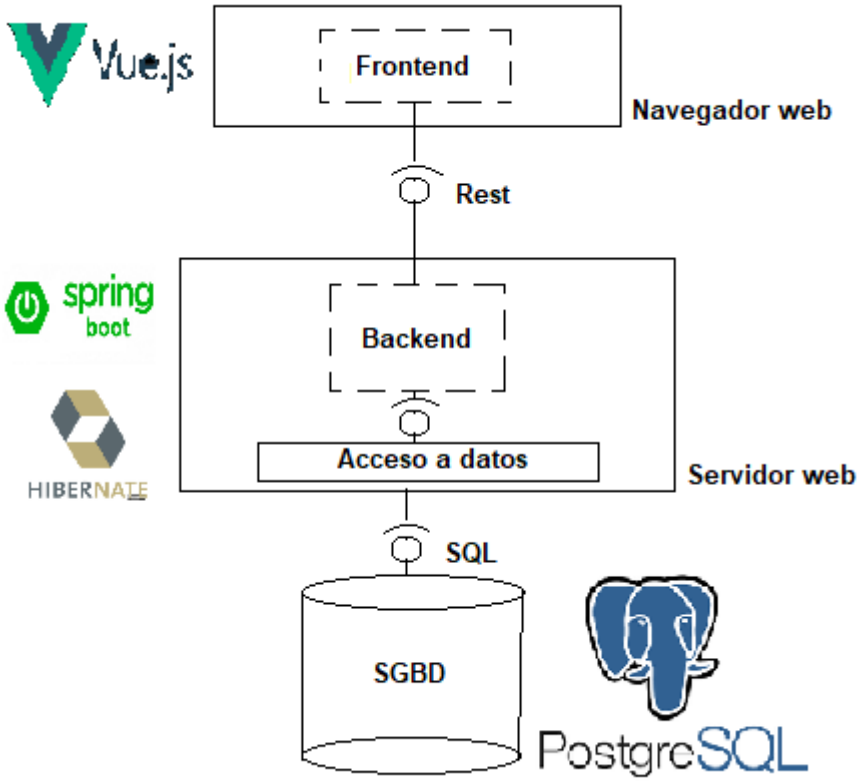

Figura 5.1: Aquitectura tecnológica del sistema

5.2.1 Servidor

En esta sección se muestran aspectos del diseño de la capa servidor de la aplicación.

Diagrama de componentes

El diagrama de componentes de la capa servidor se puede ver en la siguiente Figura 5.2. En este apartado se explica cada uno de sus elementos, a continuación se pondrá un ejemplo de comunicación entre sus componentes realizando una operación y finalmente se mostrará un ejemplo de patrón de diseño utilizado:

En la Figura 5.2 se puede ver la capa servidor en la que se encuentran las entidades en el componente "Entidades". Una entidad es la representación de un objeto persistente de la aplicación, cada componente de "Entidades" corresponde con una tabla en la base de datos. Las "Entidades" son usadas por los componentes "Controllers". Los "Controllers" son las clases que proporcionan los endpoints encargados de procesar las peticiones HTTP que llegan desde el "API" del cliente, una vez procesan los datos, hacen uso de las "Entidades" para realizar los cambios pertinentes en la base de datos. Los componentes de "Controllers" hacen uso de los servicios localizados en "Servicios". Los "Servicios" proporcionan funcionalidades adicionales a los controladores, usando al igual que estos las "Entidades" para realizar cambios en la base de datos.

A continuación en la Figura 5.3 se muestra un ejemplo de interacción entre los componentes, al recibir una petición desde el "API" del cliente de finalización de compra de un usuario, usando el método de venta de entradas con cola:

La primera operación "finCompraUsuario" llega desde un componente del "API" del cliente llamado "usuarioAPI" y es recibido por el componente ubicado en "Controllers", llamado "UsuarioController". A continuación "UsuarioController" manda una petición a un componente de "Servicios", en concreto a "TiempoCompraService", con el objetivo de que el contador del tiempo de compra del usuario se pare. Una vez hecho esto "UsuarioController" vuelve a enviar otra petición a un componente en "Servicios", en este caso a "ColaUsuariosService", esta vez el propósito de la operación es sacar al usuario que finalizó su compra de la cola de usuarios. De este modo el componente "ColaUsuariosService" hace uso del componente en "Entidades", llamado "ColaUsuarios", para actualizar la cola de usuarios en la base de datos sacando de ella el usuario que finalizó su compra. Finalmente, una vez hecho esto, "UsuarioController" manda un mensaje de respuesta al componente "usuarioAPI" situado en el "API" del cliente indicándole que la operación se realizó correctamente.

Uno de los principales patrones que se utilizan en la capa servidor es el patrón **Fachada**. Se puede ver un esquema de este patrón en la Figura 5.4.

El patrón Fachada consiste principalmente en ocultar la complejidad proporcionando una

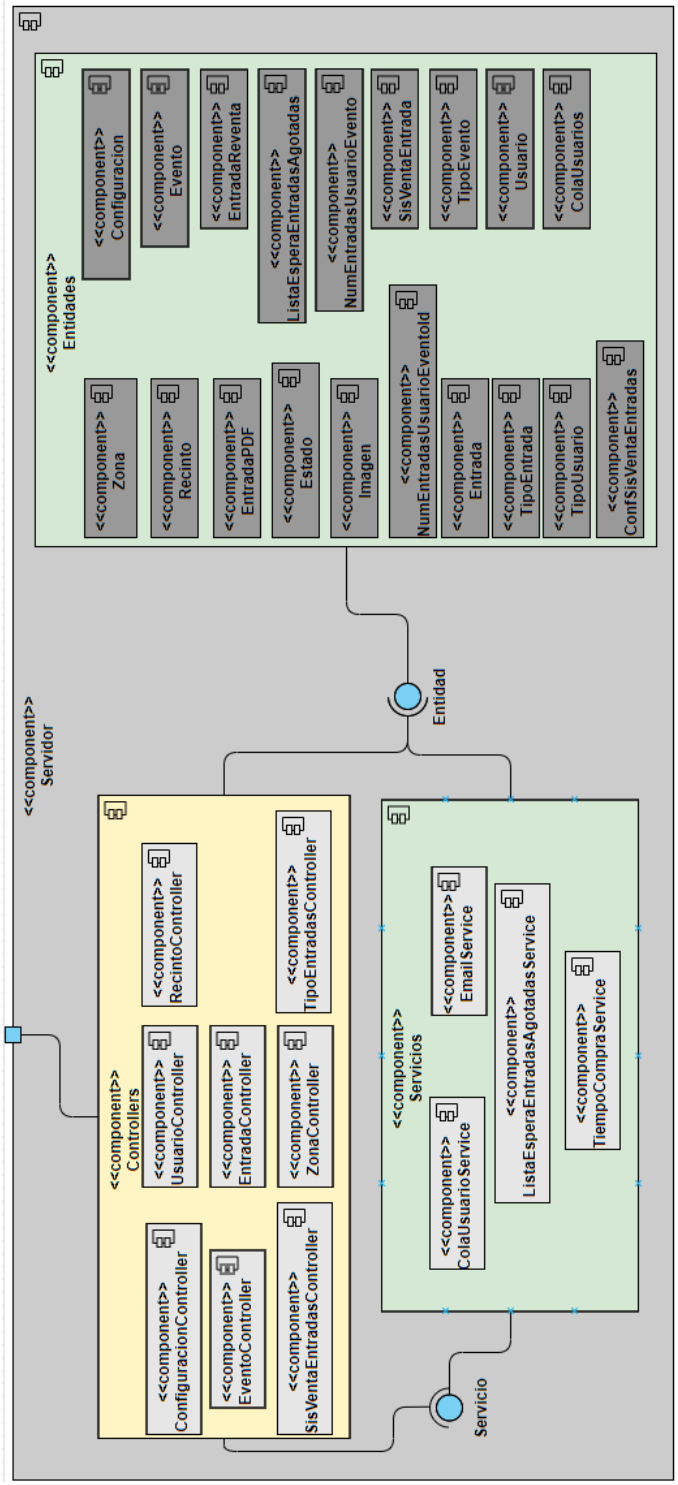

Figura 5.2: Diagrama de componentes del servidor

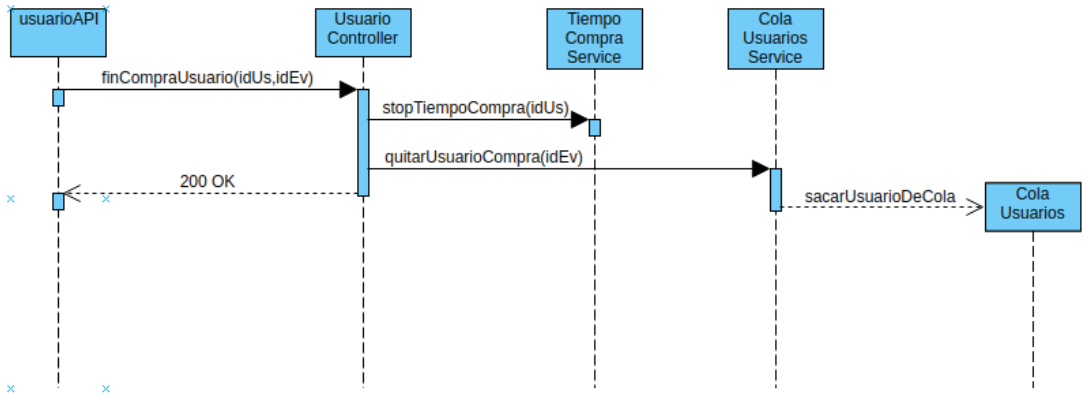

Figura 5.3: Diagrama de secuencia entre componentes de la capa servidor

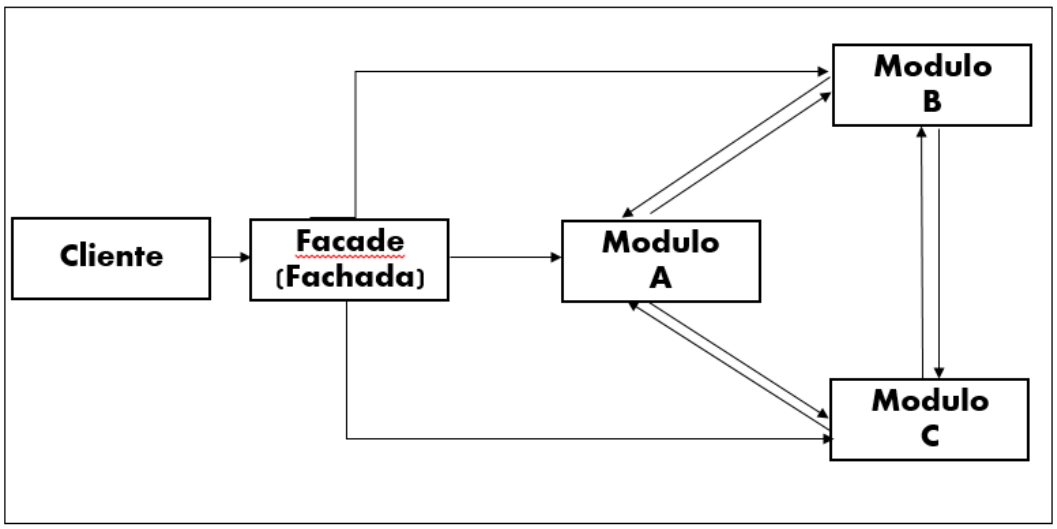

Figura 5.4: Diagrama del patrón Fachada [3]

interfaz, la cual se encarga de realizar la comunicación con todos los subsistemas minimizando las comunicaciones y dependencias entre estos. La interfaz o fachada que se proporciona son los componentes “Controllers”, que como se había dicho antes, son los encargados de recibir la peticiones que llegan desde el “API” del cliente y comunicarse con todos los subsistemas, que en este caso, son los componentes “Entidades” y “Servicios”.

Servicio REST

El API REST que ofrece el servidor, se compone de diferentes recursos sobre los que actúa, representados por una URL la cuál los identifica de manera única. Sobre cada uno de los recursos es posible ejecutar cuatro posibles operaciones: POST, GET, PUT y DELETE.

En este apartado se expone la interfaz del servicio web. Cada método REST está clasificado

por el recurso sobre el que actúa. En concreto se muestra el nombre de la operación, el método de invocación REST y la URL a la que se llama desde el cliente para realizar la operación:

En la siguiente tabla se muestra una tabla con los métodos REST que actúa sobre el **recurso Evento**. En la URL de la operación "updateEvento" el parámetro "vi" puede ser true o false dependiendo de si se quiere que el evento sea visible o no.

Operación	Método de invocación	URL
crearEvento	POST	api/eventos
añadirEvento - ConfiguracionSistema	POST	api/eventos/{idEv}/ confs_sis_venta_entradas_evento/ {idConfSis}
crearTipoEntradaEvento	POST	api/eventos/{idEv}/tipos_entrada_evento
añadirTipoEventoEvento	POST	api/eventos/{idEv}/tipo_evento/ {idTipoEvento}
getConfiguracionEvento	GET	api/eventos/{id}/configuracion_evento
getEventos	GET	api/eventos
getEvento	GET	api/eventos/{id}
getTipoEntradasEvento	GET	api/eventos/{id}/tipos_entrada
getTipoEventoEvento	GET	api/eventos/{id}/tipo_evento
getSistemasVenta - EntradasEvento	GET	api/eventos/{id}/ confs_sis_venta_entradas_evento
getUsuarioEvento	GET	api/eventos/{id}/creador_evento
getEntradasVendidasEvento	GET	api/eventos/{id}/entradas_vendidas
hayEntradasEnVenta	GET	api/eventos/{id}/entradas/en_venta
getInfoEvento	GET	api/eventos/{idEvento}/info
getInfoEventoAdmin	GET	api/eventos/info-admin
updateEvento	PUT	api/eventos/{id}/visible/?vi=VerDetalle
updateDataEvento	PUT	api/eventos/{id}
cancelarEvento	PUT	api/eventos/{id}/cancelar
updateImagenEvento	PUT	api/eventos/{id}/imagen
removeEvento	DELETE	api/eventos/{id}
removeConfVenta - EntradaFromEvento	DELETE	api/eventos/{idEv}/ confs_sis_venta_entradas_evento/ {idConf}

En la siguiente tabla se encuentran los métodos REST del **recurso Usuario**. En la URL de la operación "updateNumEntradasUsuarioEvento" el parámetro "num" permite actualizar el número de entradas compradas por un usuario de un evento, permitiendo indicar el número de entradas para actualizar. En la URL de la operación "iniciarTiempoCompra" el parámetro "conf" permite iniciar el tiempo de compra de un usuario, permitiendo especificar el método de compra (Cola o sorteo). En la URL de la operación "updateUsuario" el parámetro "tip" permite actualizar el usuario indicandole el tipo de usuario.

Operación	Método de invocación	URL
añadirEventoUsuario	POST	api/usuarios/{idUsuario}/ eventos/{idEvento}
añadirEvento - FavoritoUsuario	POST	api/usuarios/{idUsu}/ eventos_favoritos/{idEv}
comprarEntradaReventa	POST	api/usuarios/{idUsu}/ entradas_reventa/{idE}
añadirNumeroEntradas - UsuarioEvento	POST	api/usuarios/{idUsu}/ eventos/{idE}/ num_entradas_usuario_evento
getAll	GET	api/usuarios
getTipoUsuario	GET	api/usuarios/{id}/ tipoUsuario
getTipoUsuario2	GET	api/usuarios/ {id}/tipoUsuario
getUsuario	GET	api/usuarios/{id}
getEventosUsuario	GET	api/usuarios/{id}/ eventos_usuario
getEventosFav	GET	api/usuarios/{id}/ eventos_favoritos
getEntradasUsuario	GET	api/usuarios/{id}/ entradas_usuario
getEntradas - ReventaUsuario	GET	api/usuarios/{id}/ entradas_reventa
getSePuedeComprar	GET	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ check_compra

Operación	Método de invocación	URL
getEsTurno	GET	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ es_turno
getEsTurnoSorteo	GET	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ es_turno_sorteo
getTiempoCompra	GET	api/usuarios/{id}/ tiempo_compra
getUsuarioEnCuentaAtras	GET	api/usuarios/{id]/ en_compra
getNumeroEntradas - UsuarioEvento	GET	api/usuarios/ {idUsu}/eventos/ {idE}/ num_entradas_usuario_evento
getInfoUsuariosAdmin	GET	api/usuarios/info-admin
updateUsuario	PUT	api/usuarios/{id}/ update?tip=VerDetalle2
salirCompra	PUT	api/usuarios/eventos/ {idEvento}/fin_compra
cancelarEsperaCompra	PUT	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ espera_cancelada
inscribirEnSorteo	PUT	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ inscripcion
finalizaCompraSorteo	PUT	api/usuarios/ eventos/{idEvento}/ compra_terminada
añadirUsuarioListaEspera	PUT	api/usuarios/ {nombreUsuario}/ eventos/{idEvento}/ lista_espera

Operación	Método de invocación	URL
iniciarTiempoCompra	PUT	api/usuarios/{id}/ eventos/{idEvento}/ inicio_tiempo_compra ?conf=VerDetalle8
finTiempoCompra	PUT	api/usuarios/{id}/ fin_tiempo_compra
updateNumEntradas - UsuarioEvento	PUT	api/usuarios/{idUsu}/ eventos/{idE}/ num_entradas_usuario_evento ?num=VerDetalle9
removeEvento - FavoritoUsuario	DELETE	api/usuarios/ {idUsu}/ evento_favorito/{idEv}
removeUsuario	DELETE	api/usuarios/{id}
removeEvento - FromUsuario	DELETE	api/usuarios/{idUsu}/ eventos/{idEv}

En la tabla inferior se pueden ver los métodos REST del **recurso Entrada**. En la URL de la operación "añadirEntradaReventa" el parámetro "pre" permite poner a la venta entradas indicándole el precio.

Operación	Método de invocación	URL
añadirUsuarioEntrada	POST	api/entradas/{idEntrada}/ usuario_comprador/{idUsu}
añadirEntradaReventa	POST	api/entradas/{idEntrada}/ entrada_reventa_entrada/ ?pre=VerDetalle3
getPdf	GET	api/entradas/{id}/exportPDF
getTipoEntradaEntrada	GET	api/entradas/{id}/tipo_entrada_entrada
getEntradaReventaEntrada	GET	api/entradas/{id}/entrada_reventa
getEntradas	GET	api/entradas
getEntradasCompletas	GET	api/entradas/usuario/{id}/completo
getEntradasUsuario - TipoEntrada	GET	api/entradas/usuario/{idUs}/ tipo_entrada/{idTe}

getEstadisticasReg	GET	api/entradas/usuario/{idUs}/ estadisticas-reg
getEstadisticasPro	GET	api/entradas/usuario/{idUs}/ estadisticas-pro
getEstadisticasProEventos	GET	api/entradas/usuario/{idUs}/ estadisticas-pro-eventos
updateEntrada	PUT	api/entradas/{id}
removeEntrada	DELETE	api/entradas/{id}
removeEntrada - FromReventa	DELETE	api/entradas/{idEntrada}/ entrada_reventa_entrada

En la siguiente tabla están representados los métodos REST del **recurso Recinto**. En la URL de la operación "crearConfiguracionRecinto" el parámetro "nom" permite crear una configuración de recinto indicándole el nombre.

Operación	Método de invocación	URL
crearRecinto	POST	api/recintos
crearZonaRecinto	POST	api/recintos/{idRecinto}/ zonas_recinto_recinto
crearConfiguracionRecinto	POST	/api/recintos/{idRecinto}/ configuraciones_recinto/ ?nom=VerDetalle4
getRecintos	GET	api/recintos
getRecinto	GET	api/recintos/{id}
getZonasRecinto	GET	api/recintos/{id}/zonas_recinto
getConfiguracionesRecinto	GET	api/recintos/{id}/configuraciones
updateRecinto	PUT	api/recintos/{id}
removeRecinto	DELETE	api/recintos/{id}

A continuación se muestra una tabla con los métodos REST del **recurso Zona**.

Operación	Método de invocación	URL
añadirTipoEntradaZona	POST	api/zonas/{idZona}/ tipos_entrada/{idEntrada}
getRecintoZona	GET	api/zonas/{id}/recinto_zona
updateZona	PUT	api/zonas/{id}
removeZona	DELETE	api/zonas/{id}

En la tabla inferior se observan los métodos REST del **recurso Configuracion**.

Operación	Método de invocación	URL
añadirZonasConfiguracion	POST	api/configuraciones/{idConfiguracion}/ zonas_configuracion/{idZona}
añadirEventoConfiguracion	POST	api/configuraciones/{idConfiguracion}/ eventos/{idEvento}
getZonasConfiguracion	GET	api/configuraciones/{id}/ zonas_configuracion
getRecintoConfiguracion	GET	api/configuraciones/{id}/ recinto_configuracion
updateConfiguracion	PUT	api/configuraciones/{id}
removeConfiguracion	DELETE	api/configuraciones/{id}
removeEvento - FromConfiguracion	DELETE	api/configuraciones/{idConf}/ eventos/{idEv}
removeZona - FromConfiguracion	DELETE	api/configuraciones/{idConfiguracion}/ zonas_configuracion/{idZona}

En la siguiente tabla podemos ver los métodos REST del **recurso EntradaReventa**.

Operación	Método de invocación	URL
getEntradaEntradaReventa	GET	api/entradasReventa/{id}/ entrada_entrada_reventa
getEntradasReventa	GET	api/entradasReventa

Seguidamente se encuentra la tabla con los métodos REST del **recurso ConfSisVentaEntrada**.

Operación	Método de invocación	URL
getSistemasVentaEntradasConfSis	GET	api/confSisVentaEntradas/{id}/ sistema_venta_entradas
removeConfVentaEntrada	DELETE	api/confSisVentaEntradas/{idConf}

En la tabla inferior se muestran los métodos REST del **recurso SisVentaEntrada**.

Operación	Método de invocación	URL
añadirSistema - ConfiguracionSistema	POST	api/sisVentaEntradas/{idSis}/ confs_sistema_venta_entradas
getSistemasVentaEntradas	GET	api/sisVentaEntradas
removeConfVentaEntrada - FromSisVentaEntrada	DELETE	api/sisVentaEntradas/{idSis}/ confs_sistema_venta_entradas/ {idConf}

En la tabla inferior se exponen los métodos REST del **recurso TipoEntrada**. En la URL de la operación "updateEntradasTipoEntrada" el parámetro "precio" permite actualizar todas la entradas de un tipo de entrada con un precio diferente. En la URL de la operación "añadirEntradasTipoEntrada" el parámetro "num" y "precio" permite crear entradas de un tipo de entrada determinado, indicándole el número de entradas de ese tipo de entrada que se quiere crear y el precio de cada una.

Operación	Método de invocación	URL
añadirEntradas - TipoEntrada	POST	api/tipoEntradas/{idTipo}/ entradas_tipo_entrada/ ?num=VerDetalle5 ?precio= VerDetalle5
getZonaTipoEntrada	GET	api/tipoEntradas/{id}/zona_entrada
getZonaEntrada	GET	api/tipoEntradas/{id}/zona_entrada
getEntradasTipoEntrada	GET	api/tipoEntradas/{id}/entradas_tipo

getEventoTipoEntrada	GET	api/tipoEntradas/{id}/evento_entrada
getEntradasVenta	GET	api/tipoEntradas/{idTipoEntrada}/ entradas/venta
getEntradasVentaNumero	GET	api/tipoEntradas/{idTipoEntrada}/ entradas/ventaNumero
updateTipoEntrada	PUT	api/tipoEntradas/{id}
updateEntradas - TipoEntrada	PUT	api/tipoEntradas/{id}/ entradas_tipo_entrada/ ?precio=VerDetalle6
removeTipoEntrada	DELETE	api/tipoEntradas/{id}
removeAllEntradas - TipoEntrada	DELETE	api/tipoEntradas/{id}/entradas_tipo_entrada

En la siguiente tabla vemos los métodos REST del **recurso TipoEvento**.

Operación	Método de invocación	URL
getTipoEventos	GET	api/tipoEventos

A continuación se exhibe ua tabla con los métodos REST del **recurso TipoUsuario**.

Operación	Método de invocación	URL
crearUsuario	POST	api/tipoUsuarios/{idTipoUsuario}/usuarios
getTipoUsuarios	GET	api/tipoUsuarios

5.2.2 Cliente

En esta sección se muestran aspectos del diseño de la capa cliente de la aplicación.

Diagrama de componentes

El diagrama de componentes de la capa cliente se puede ver en la Figura 5.5. Se va a explicar cada uno de sus elementos, posteriormente se pondrá un ejemplo de comunicación entre los componentes realizando un operación y finalmente se pondrá un ejemplo de un patrón de diseño utilizado.

En el diagrama de componentes se puede ver la capa cliente en la que se encuentran los

componentes pertenecientes a la aplicación cliente. Los "Components" que se ven a la derecha de la imagen, son los ficheros organizados en paquetes según la funcionalidad que cumplen. Dentro de estos ficheros se encuentra el código HTMl, JavaScript, y CSS necesario para generar una página de la aplicación que contiene diferentes funcionalidades. Los "Components" se comunican con los componentes localizados en "API" mandándoles peticiones para que estas lleguen a la capa servidora, donde se procesan. Cuando llegan al "API", se envían al servicio REST de la capa servidor, localizado en los controladores. Una vez reciben la respuesta del servidor se la mandan al componente en "Components" adecuado, y finalmente, generan la página que verá el usuario de la aplicación.

A continuación en la Figura 5.6 se muestra un ejemplo de interacción entre los componentes de la capa cliente, al pulsar el usuario sobre el botón de salir de la compra de entradas de un evento:

En primer lugar vemos como el usuario pulsa sobre el botón salir de la compra, esta operación se envía al componente apropiado en "Components", en concreto al componente "info-entradas-evento", este a su vez envía una petición al componente "usuarioAPI", localizado en "API", luego este componente se encarga de reenviar la petición al controlador de la capa servidor que se encarga de esta realizar esta operación, en este caso "UsuarioController". Finalmente el controlador realiza la operación y responde al "API" con el código de éxito, este se lo envía al componente "info-entradas-evento" que genera la página conveniente para el usuario.

Uno de los principales patrones utilizados en la capa cliente es el patrón **MVVM (Model–view–viewmodel)**, en la siguiente Figura 5.7 se puede ver el esquema de este patrón. Consiste en organizar la arquitectura software de manera que se separe la interfaz de usuario (la vista) de la lógica de negocio (el modelo), con el objetivo de que la vista no sea dependiente de ningún modelo específico. Los elementos de este patrón son los siguientes:

- **Modelo**: Representa la capa de datos, se corresponde con el cliente REST.
- **Vista**: Representa la interfaz de usuario, se corresponde con el código HTML y CSS.
- **Modelo de vista**: Es el actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. Se corresponde con el código JavaScript.

Enrutamiento

Cada uno de los componentes de los que se habló en la sección anterior son llamadas por el componente Vue Router, utilizado por Vue para enrutar las páginas a las que accede el usuario navegando a través de la aplicación.

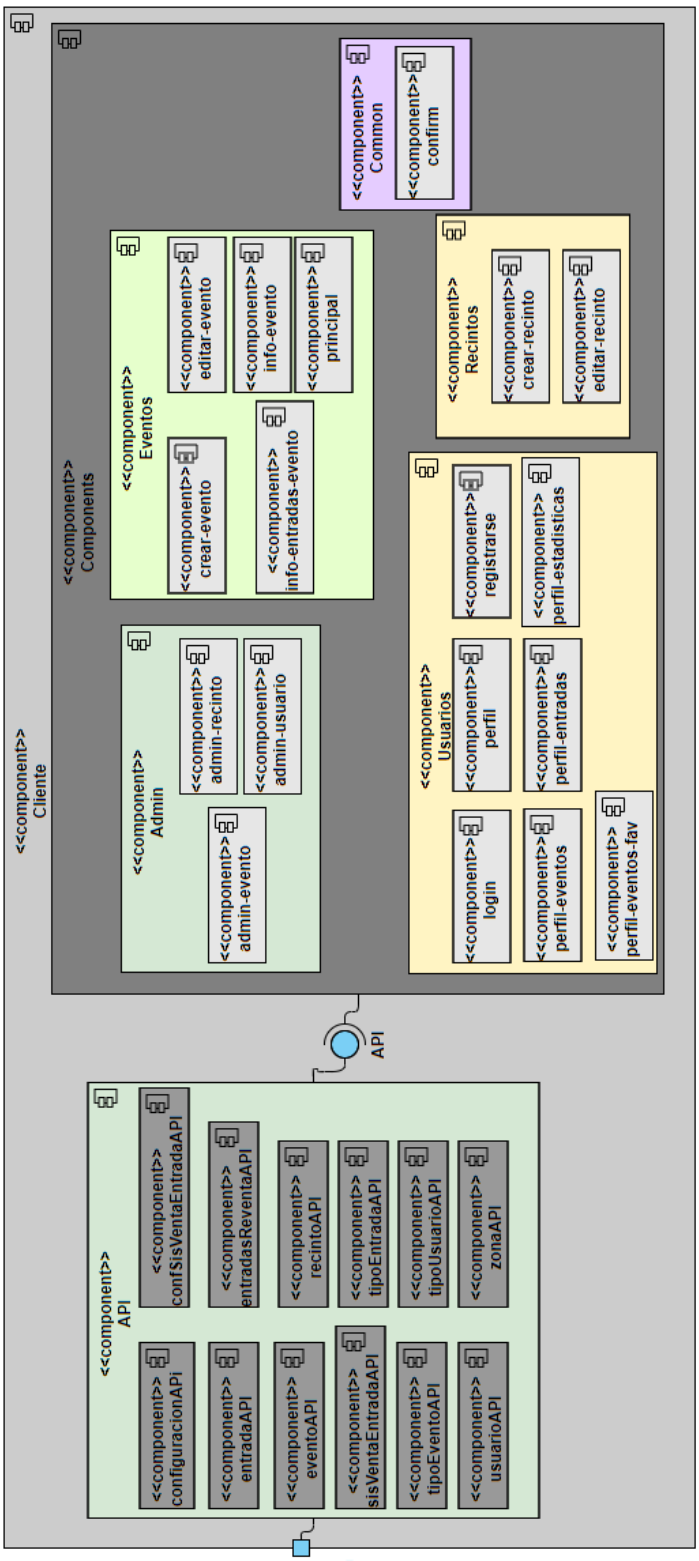

Figura 5.5: Diagrama de componentes del cliente

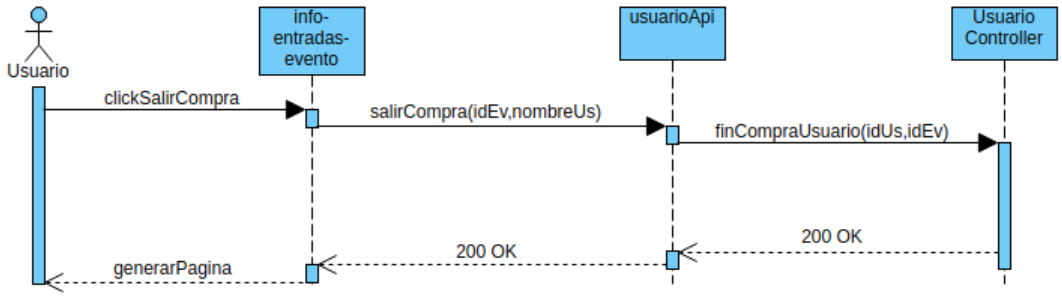

Figura 5.6: Diagrama de secuencia entre componentes de la capa cliente

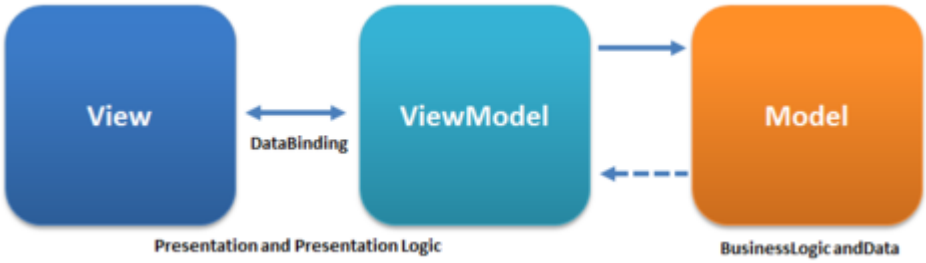

Figura 5.7: Patrón MVVM [4]

Posteriormente se muestran las rutas de cada uno de los componentes junto con su nombre, clasificados por la funcionalidad que cumple cada componente.

A continuación se muestra la tabla con las **rutas de los componentes de administración**.

Componente	Nombre	Ruta
admin-recinto.vue	admin-recinto	/admin-recinto
admin-evento.vue	admin-evento	/admin-evento
admin-usuario.vue	admin-usuario	/admin-usuario

En la tabla inferior se exponen las **rutas de los componentes de evento**.

Componente	Nombre	Ruta
crear-evento.vue	crear-evento	/crear-evento
editar-evento.vue	editar-evento	/editar-evento/:id
info-entradas-evento.vue	info-entradas-evento	/info-entradas-evento/:id/ :colaActivada/:sorteoActivado
info-evento.vue	info-evento	/info-evento/:id
principal.vue	principal	/principal

En la siguiente tabla se enseñan las **rutas de los componentes de usuario**.

Componente	Nombre	Ruta
login.vue	login	/login
perfil.vue	perfil	/perfil
perfil-entradas.vue	perfil-entradas	/perfil-entradas
perfil-estadisticas.vue	perfil-estadisticas	/perfil-estadisticas
perfil-eventos.vue	perfil-eventos	/perfil-eventos
perfil-eventos-fav.vue	perfil-eventos-fav.	/perfil-eventos-fav
registrarse.vue	registrarse	/registrarse

A continuación se pueden ver las **rutas de los componentes de recinto**.

Componente	Nombre	Ruta
crear-recinto.vue	crear-recinto	/crear-recinto
editar-recinto.vue	editar-recinto	/editar-recinto/:id

Capítulo 6

Implementación y pruebas

6.1 Implementación

En esta sección se van a exponer los detalles complejos de la aplicación, se describen los pasos y se muestran junto con un diagrama de secuencia. Por último se explicarán las pruebas que se llevaron a cabo.

6.1.1 Método de venta de entradas por sorteo

En la Figura 6.1 se muestra el diagrama de secuencia del método de venta de entradas por sorteo, el cuál se explicará posteriormente:

1. El usuario entra en un evento con la configuración de venta de entradas "Sorteo" y pulsa sobre el botón "Inscribirse en sorteo". El cliente manda al servidor que añada al usuario a una lista para sortear el turno de compra. El servidor contesta con un mensaje de éxito o de error en el caso de que el usuario ya estuviera inscrito en el sorteo.

2. Cuando llega la hora de la apertura del plazo de compras de entradas del evento, se realiza en el servidor el sorteo de los turnos de compra entre los participantes inscritos en el sorteo, y a continuación, se envía un correo al usuario con el puesto en el que ha quedado y la hora a la que puede comprar entradas.

3. El usuario entra en el evento y pulsa sobre el botón "Ver entradas". El cliente manda un mensaje al servidor para que compruebe si es el turno del usuario para comprar entradas, el servidor responde con el resultado de la operación. En caso de que sea su turno se le permite comprar entradas, en caso contrario se le muestra un mensaje de error.

4. Cuando el usuario está comprando y finaliza la compra, o sale de la misma, el cliente le comunica al servidor que este usuario terminó de comprar. El servidor saca al usuario

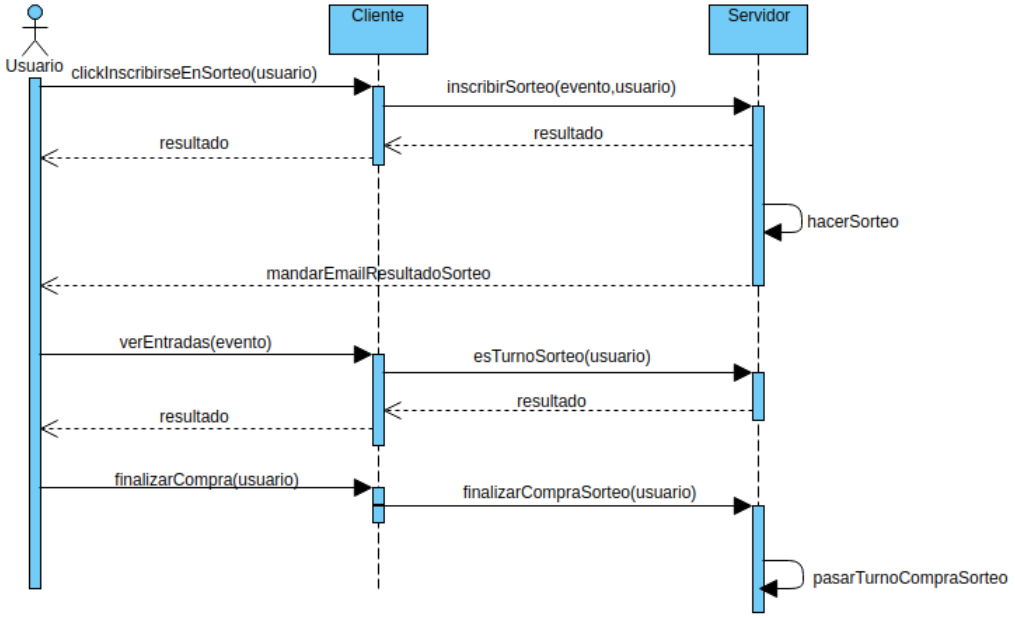

Figura 6.1: Diagrama de secuencia del sorteo de entradas

de la cola, finalizando así su compra.

5. En el momento que pasa el turno de compra, se ejecuta en el servidor automáticamente una función que saca a los usuarios del turno anterior de la cola e introduce a los del siguiente turno.

6.1.2 Reventa de entradas en la aplicación

A continuación en la Figura 6.2 se va a explicar el proceso por el cuál se lleva a cabo la funcionalidad de reventa de entradas en la aplicación:

1. El usuario en su página “Mis entradas”, hace click sobre el botón “revender” de un tipo de entrada de las que tiene compradas.

2. El cliente manda un mensaje al servidor pidiéndole las entradas de ese tipo y que tiene en su posesión el usuario.

3. El cliente recibe la lista de entradas del servidor y se las muestra al usuario.

4. El usuario pulsa sobre el botón “Revender” de una de las entradas de la lista, mostrándole un formulario para que introduzca el precio al que quiere poner a la reventa la entrada.

5. Una vez introduce el precio el cliente manda un mensaje al servidor para que ponga en la reventa esa entrada.

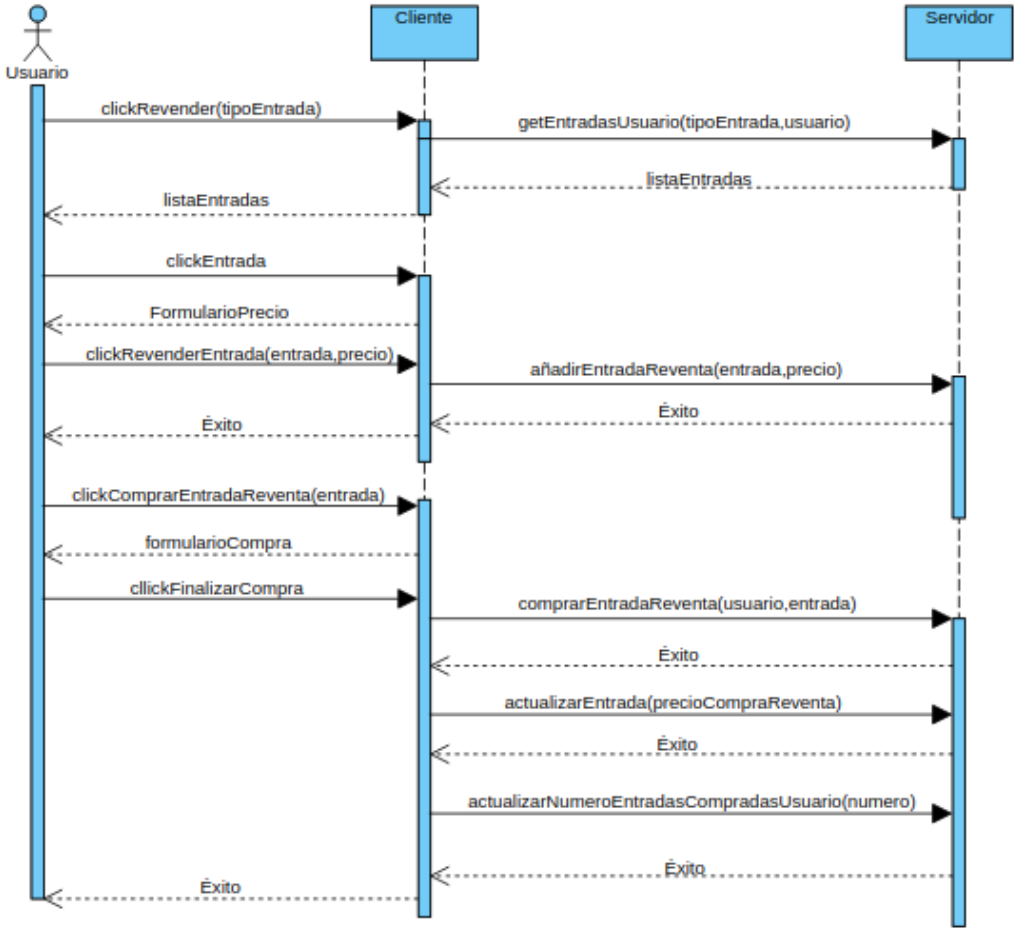

Figura 6.2: Diagrama de secuencia de reventa de entradas

6. Un usuario diferente entra en la compra de entradas del evento al que pertenece la entrada que puso a la venta el usuario anterior. Selecciona la entrada en reventa y pulsa sobre el botón “Comprar”.

7. El cliente le muestra un formulario que tiene que cubrir con los datos necesarios para la compra.

8. El usuario rellena el formulario y pulsa en “Finalizar compra”.

9. El cliente manda un mensaje al servidor para que éste realice el proceso de compra de la entrada en reventa.

10. Una vez hecho esto, el cliente envía otro mensaje pidiendo que actualice el precio de la entrada comprada al precio en el que se compró en la reventa. Esto se hace para que el usuario comprador de la entrada en reventa no se lucre volviendo a revender la entrada en más precio de la que la compró.

11. Finalmente se actualiza el número de entradas compradas por el usuario en ese evento.

6.1.3 Despliegue en Heroku

La aplicación se desplegó en Heroku, una plataforma PaaS (Plataforma Como Servicio). El servidor y cliente se desplegaron en dos URLs diferentes.

A continuación se muestra el proceso que hizo posible el despliegue de la aplicación en una plataforma cloud.

1. Lo primero que fue necesario es registrarse en la página de Heroku [20].

2. A continuación, en la página de Heroku, se crea una nueva aplicación, añadiendo ésta a una nueva pipeline Figura 6.3.

3. Una vez tenemos creada la aplicación, instalamos el cliente de Heroku en la terminal ejecutando lo siguiente: “sudo snap install –classic heroku”, luego una vez instalado hacemos login en nuestra cuenta de Heroku: “heroku login”.

4. Ya logueados en nuestra cuenta nos dirigimos al directorio donde se encuentra la aplicación: “cd directorio-aplicacion” y ejecutamos: “git init”, y posteriormente: “heroku git:remote -a app-nombreapp”. Con esto lo que hacemos es crear un repositorio heroku para subir nuestro código para poder desplegar la aplicación.

5. Por último subimos nuestro código al repositorio: “git add .”, hacemos commit: “git commit -m “Subido código””, y finalmente push: “git push heroku master”. Una vez hecho esto la aplicación estará desplegada en Heroku.

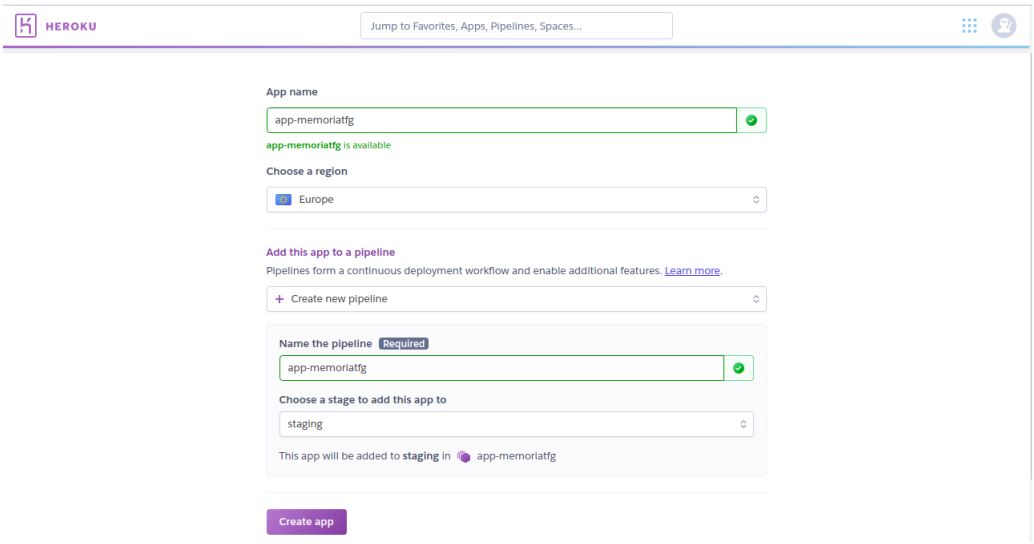

Figura 6.3: Creación de la aplicación en Heroku

6.2 Pruebas

Se creó un programa cliente para insertar datos en el servidor web y así poder probar que funciona correctamente. Concretamente lo que hace es llamar al servicio REST que se está ejecutando en el servidor, en la URL donde está desplegado el servico REST, para probar la mayor parte de las funcionalidades de la aplicación, sirviendo como pruebas de integración. Se hizo uso de la librería RestTemplate [23] para realizar las llamadas al servicio REST.

A continuación se muestra un ejemplo de llamadas a métodos REST utilizados en el programa:

- En el Listado 6.1, se puede ver un ejemplo de llamada al método POST, en el que se crean dos recintos nuevos.
- En el Listado 6.2, se observa como se hace llama al método GET y se obtienen los sistemas de venta de entradas que posteriormente se introducen en un array.
- En el Listado 6.3, se observa como se hace llama al método PUT para actualizar el número de entradas compradas de un evento por un usuario.

```
1 //Se crean los recintos
2 ResponseEntity<Recinto> recinto1 =
      restTemplate.postForEntity(urlCrearRecintos, r1, Recinto.class);
3 String idRecinto1 = recinto1.getBody().getId_recinto().toString();
4 ResponseEntity<Recinto> recinto2 =
      restTemplate.postForEntity(urlCrearRecintos, r2, Recinto.class);
```

```
String idRecinto2 = recinto2.getBody().getId_recinto().toString();
```

Listing 6.1: Ejemplo de llamada a método POST

```
//Se cogen los sistemas de venta de entradas posibles para un evento
ResponseEntity<Map> sisVenta =
    restTemplate.getForEntity(urlComun+"sisVentaEntradas/total",
    Map.class);
mapaSisVentaEntradas = (HashMap<String, Long>) sisVenta.getBody();
List<String> listaSisVentaEntradas = new
    ArrayList(sisVenta.getBody().keySet());
for(String s: listaSisVentaEntradas) {
  if(s.equals("COLA") || s.equals("ORDEN_LLEGADA") ||
    s.equals("INSCRIPCION_SORTEO")) {
      sisVentaEntradaUnico.add(s);
  }
  else {
      sisVentaEntradaMultiple.add(s);
  }
}
```

Listing 6.2: Ejemplo de llamada a método GET

```
int numeroEntradasTotal = numEntradas + numeroEntradasComprar;
restTemplate.put(urlComun+"usuarios/"+idUsuarioAleatorio+"/eventos/"
+idEventosEntradas.get(idTipoEntradaAleatorio)
+"/num_entradas_usuario_evento?num="+numeroEntradasTotal, null);
```

Listing 6.3: Ejemplo de llamada a método PUT

El programa crea recintos, zonas, configuraciones, usuarios, eventos, realiza compras de entradas, etc. La estructura principal del programa se puede ver en el siguiente Listado 6.4. En el se puede ver el flujo del programa: en primer lugar se crean los recintos, luego los usuarios y posteriormente se crean eventos finalizados, comprando entradas de los mismos. Finalmente se crean los eventos sin finalizar y se compran algunas de sus entradas.

```
public static void main(String[] args) throws IOException {
  para = new Parametros();
  restTemplate = new RestTemplate();
  randnum = new Random();
  idUsuariosPromotores = new ArrayList<Long>();
  idUsuariosRegistrados = new ArrayList<Long>();
  idConfiguracionesZonas = new HashMap<Long,List<Long>>();

  //Se crean los recintos
```

```
10    Prueba.crearRecintos();
11
12    //Se crean los usuarios
13    Prueba.crearUsuarios();
14
15    //Se crean los eventos pasados
16    HashMap<Long,Long> idEventosEntradasEventosPasados =
        crearEventos(true);
17    //Se compran entradas de los eventos pasados
18    comprarEntradas(idEventosEntradasEventosPasados,
        "numero.tipos.entradas.compradas.evento.pasado");
19
20    //Se crean los eventos
21    HashMap<Long,Long> idEventosEntradas = crearEventos(false);
22    //Se compran entradas de los eventos
23    comprarEntradas(idEventosEntradas,
        "numero.tipos.entradas.compradas");
24  }
```

Listing 6.4: Estructura principal del programa

Es parametrizable, se puede indicar el número de eventos, precio de las entradas, numero de entradas de cada evento, etc. En el Listado 6.5 se muestran los parámetros que se pueden personalizar.

```
1  #Configuracion del programa de insercion de datos
2  numero.eventos=4
3  #No mayor que 5
4  numero.sis.venta.entradas.evento=3
5  precio.maximo.tipos.entradas=50
6  numero.tipos.entradas.compradas=5
7  numero.maximo.entradas.tipos.entradas=50
8
9  numero.eventos.pasados=2
10 #No mayor que 5
11 numero.sis.venta.entradas.evento.pasado=3
12 precio.maximo.tipos.entradas.evento.pasado=50
13 numero.tipos.entradas.compradas.evento.pasado=5
14 numero.maximo.entradas.tipos.entradas.evento.pasado=50
```

Listing 6.5: Parámetros del programa de prueba

Los datos que crea el programa son variables, ya que se utilizan valores aleatorios para la generación de los mismos. Por ejemplo en la creación de un evento, el usuario creador del mismo, se escoge de manera aleatoria de la siguiente manera: se genera un identificador aleatorio de un usuario promotor, y a continuación se le vincula el evento.

Capítulo 7

Solución desarrollada

En este capítulo se van a mostrar las características principales de la aplicación desarrollada. Lo que se va a hacer es realizar una visita guiada por las principales funcionales de la aplicación acompañada de capturas de pantalla de la aplicación final. El resto de pantallas de la aplicación se encuentran en el Apéndice B

Se va a clasificar las funcionalidades de la aplicación en base al tipo de usuario que las realiza.

7.1 Usuario registrado

- El usuario registrado pulsa sobre el botón en la esquina superior derecha para acceder a su página de entradas. En esta página puede ver tres listas diferentes, como podemos ver en la Figura 7.1:

  - En la primera se encuentran las entradas compradas de eventos que aún no se han celebrado. Pulsando sobre los botones que hay en cada entrada se pueden poner en la reventa (En el caso de que la configuración del evento de las entradas tenga habilitada tal función), ver la información y descargar en PDF cada una de las entradas, o tener la información del evento al que pertenecen las entradas.
  - En segundo lugar hay una lista para las entradas que ha puesto en reventa el usuario. Cada entrada dispone de un botón para ver su información y quitarlas de la reventa.
  - Por último lugar una lista de las entradas de los eventos a los que ha asistido. Cuentan con dos botones, uno de ellos para ver la información de las entradas, desde el que se pueden puntuar los eventos, y otro para ver la información del evento.

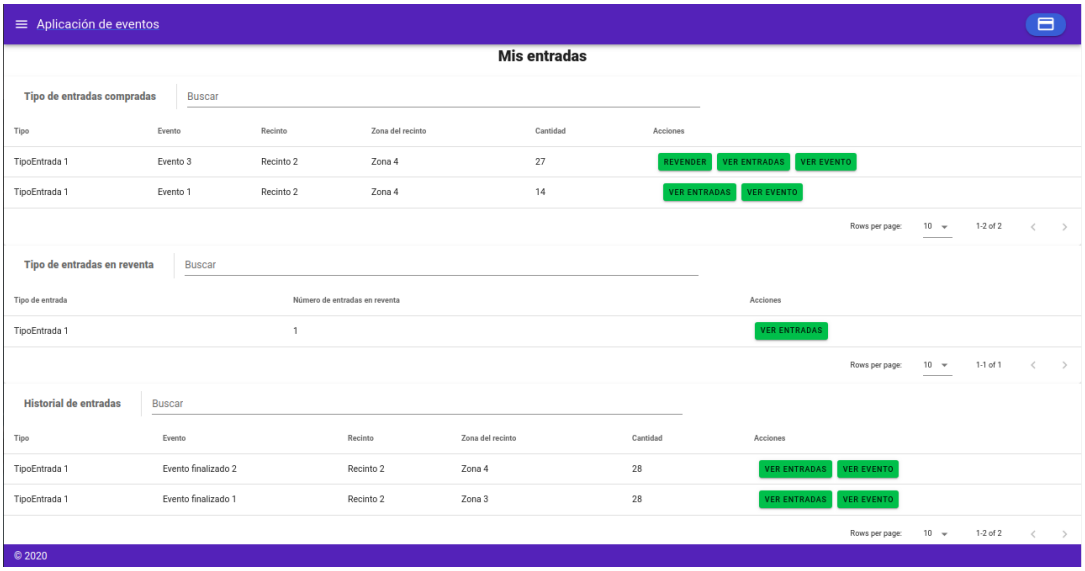

Figura 7.1: Pantalla de entradas

- El usuario pulsa sobre el menú en la parte superior izquierda de la pantalla de la aplicación (Véase la Figura 7.2), en ella podemos ver los siguientes apartados:
  - "Mi perfil", donde se muestran los datos de la cuenta del usuario, con la posibilidad de editarlos.
  - "Mis eventos favoritos", se muestra una lista de eventos que el usuario tiene guardados en favoritos.
  - "Mis estadísticas", se encuentra un listado de estadísticas de usuario.
  - "Logout usuario", sale de su cuenta de usuario.
- El usuario pulsa sobre un evento de la lista de eventos de la página principal, accede a la información del evento que ha elegido y pulsa sobre el botón "Ver entradas". El usuario entra en la página de compra de entradas del evento elegido (Véase la Figura 7.3), selecciona para comprar dos entradas y pulsa sobre el botón "Comprar". Aparece un formulario con la información de compra que debe rellenar y pulsa sobre "Finalizar compra".

7.2 Usuario promotor

- En la Figura 7.4, vemos como el usuario promotor pulsa sobre el botón de la esquina superior derecha para acceder a la su lista de eventos creados. En ella se pueden ver dos listas:

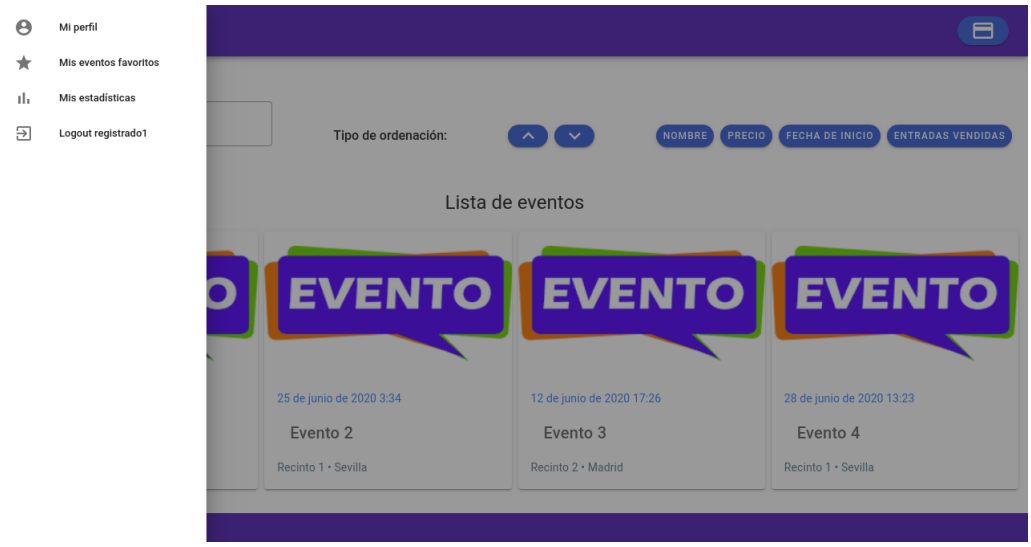

Figura 7.2: Pantalla de menú del usuario registrado

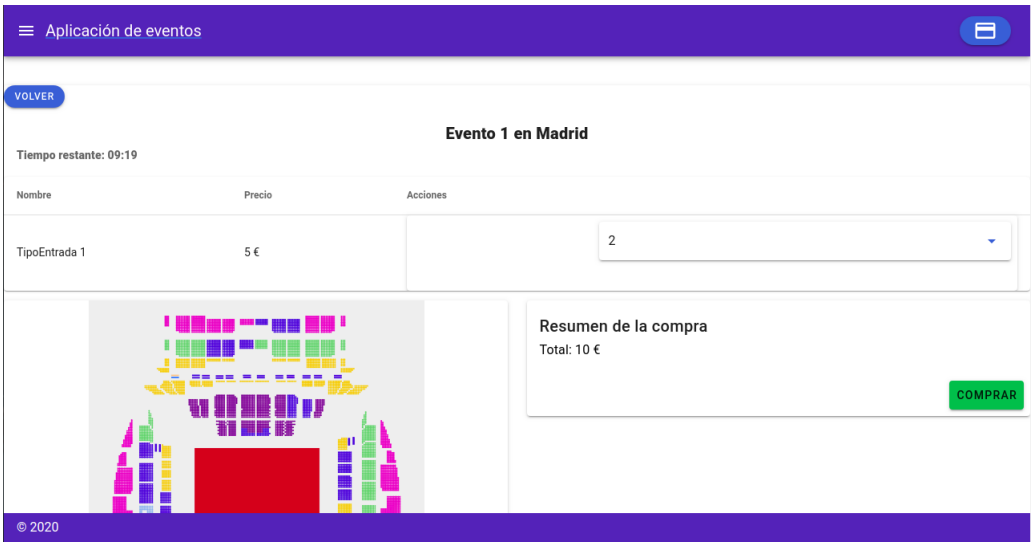

Figura 7.3: Pantalla de compra de entradas

  - En la primera de ellas se muestran los eventos creados que aún no han sido celebrados. Las acciones posibles sobre estos eventos son editar, ver su información o cancelarlos.
  - La segunda lista enseña los eventos creados por el usuario que ya han finalizado. La acción posible es ver su información

- Como podemos ver en la Figura 7.4, el usuario pulsa sobre el botón "Crear evento" de la esquina superior derecha de la página "Mis eventos" para crear un nuevo evento(Véase la Figura 7.5 y la Figura 7.6). En la siguiente Figura 7.5, observamos la primera de las pantallas de creación de evento, vemos información general que el usuario debe rellenar. En la Figura 7.6, el usuario configura parámetros del evento como son el recinto donde se celebrará el evento, sistemas de venta de entradas, etc.

- El usuario pulsa sobre el menú situado en la esquina superior izquierda de la Figura 7.7, en ella podemos ver los siguientes apartados:

  - "Mi perfil": en donde se muestra una página donde se ven los datos de la cuenta del usuario, lugar en el que se pueden editar, al igual que el usuario registrado.
  - "Mis estadísticas": accede a una página de estadísticas del promotor. En ella se puede ver:
    * Una lista con todos los eventos que creó el usuario, con estadísticas por evento, junto con un gráfico mostrando el número de entradas vendidas en cada fecha, y unas estadísticas globales.
    * También se muestra una tabla mostrando las puntuaciones medias de sus eventos finalizados.
  - "Logout usuario": sale de su cuenta de usuario.

7.3 Usuario Administrador

- En la Figura 7.8, el usuario administrador entra en su cuenta, apareciendo en la página principal de administración de eventos. En esta página puede visualizar y editar la información de los eventos, además de borrar eventos.

- El usuario pulsa sobre el menú situado en la esquina superior izquierda de la Figura 7.9, en ella se pueden ver los siguientes apartados:

  - "Administrar eventos", que se corresponde con la página principal del usuario, descrita anteriormente.

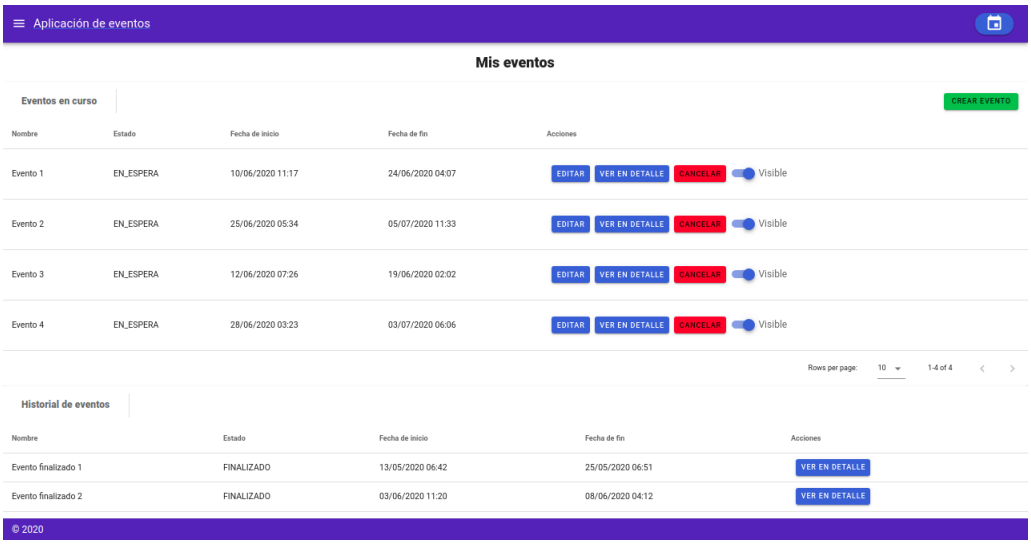

Figura 7.4: Pantalla de eventos

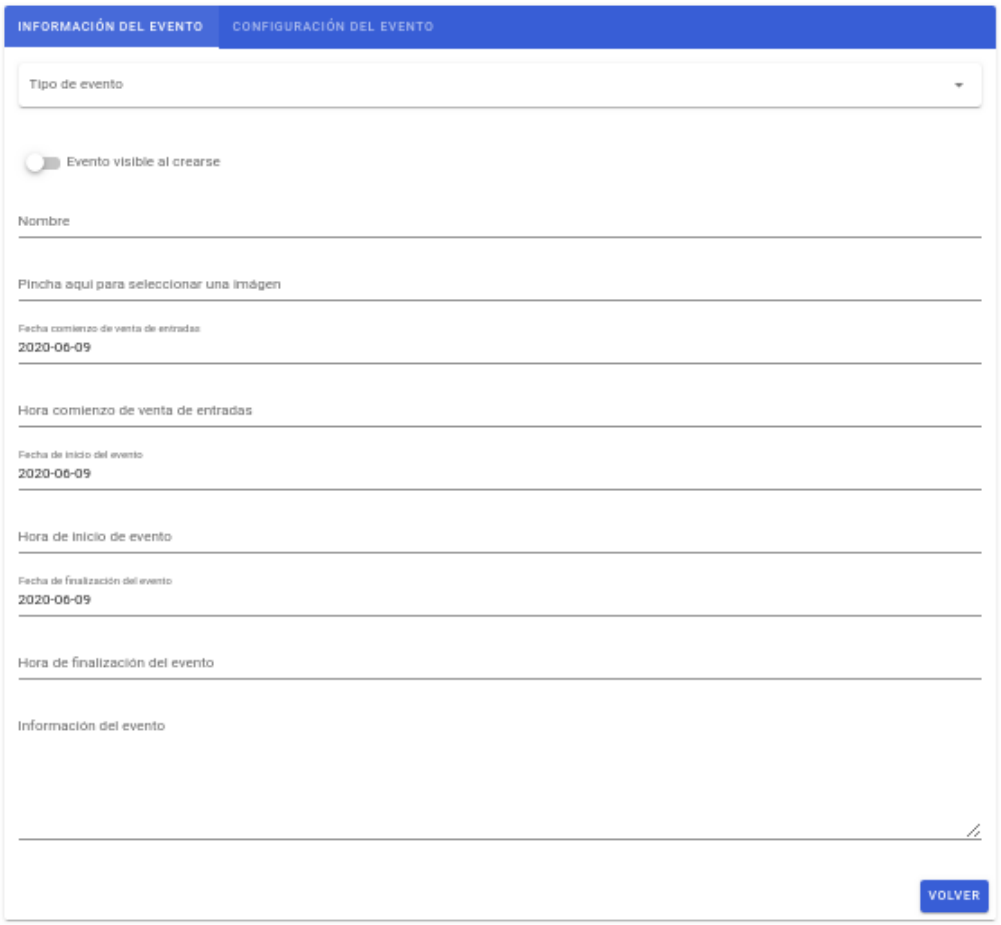

Figura 7.5: Pantalla de información de creación del evento

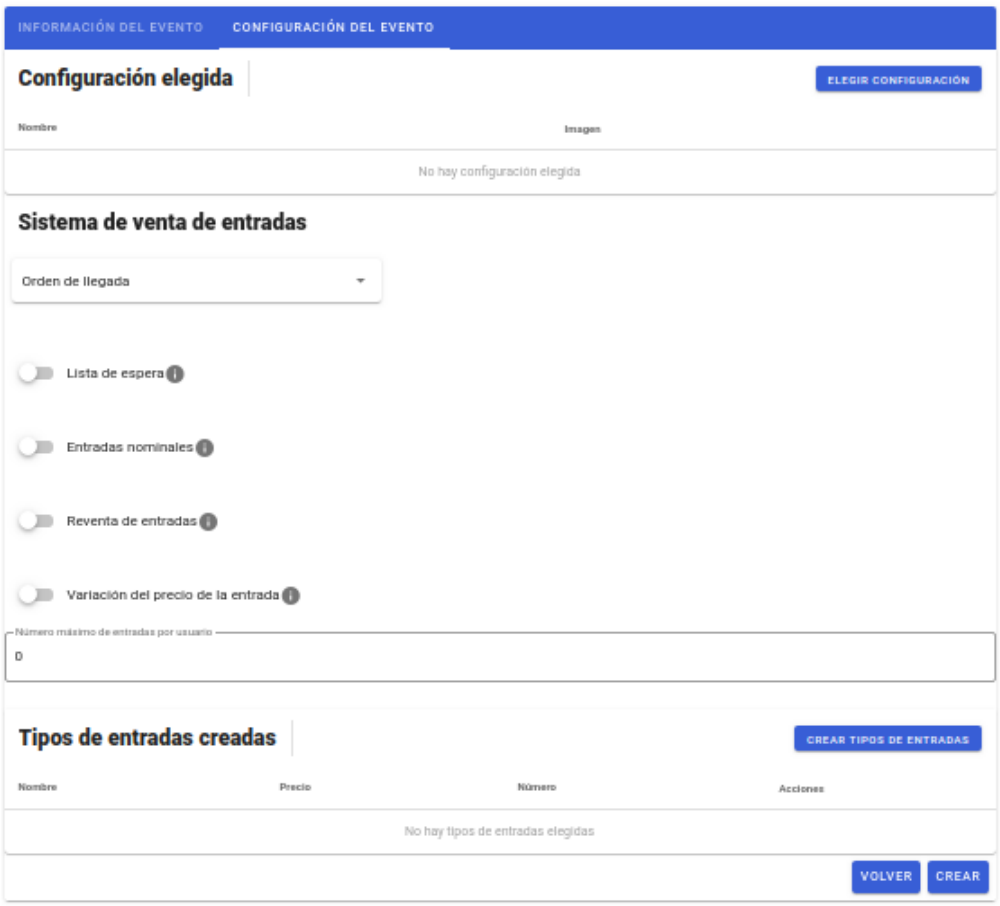

Figura 7.6: Pantalla de configuración de creación del evento

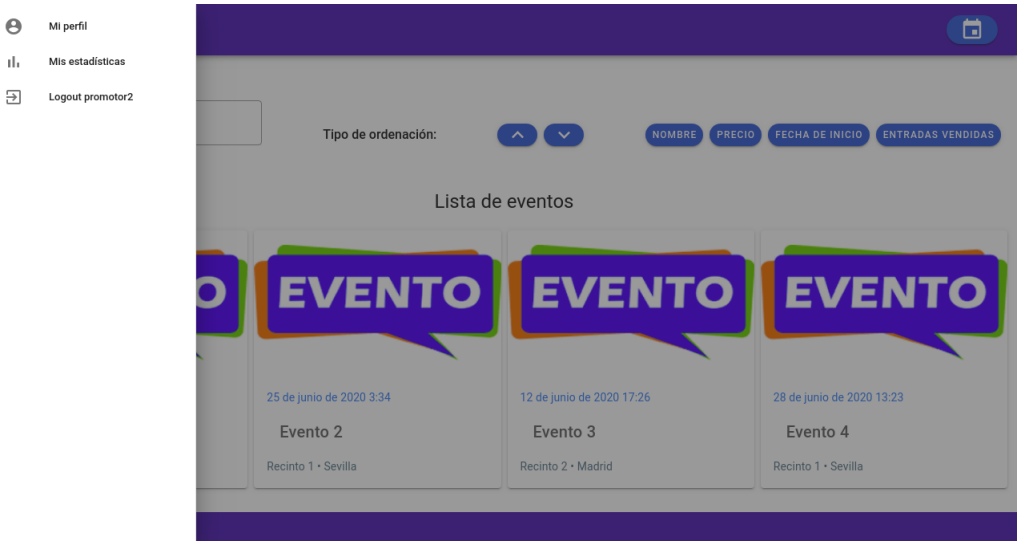

Figura 7.7: Pantalla de menú del usuario promotor

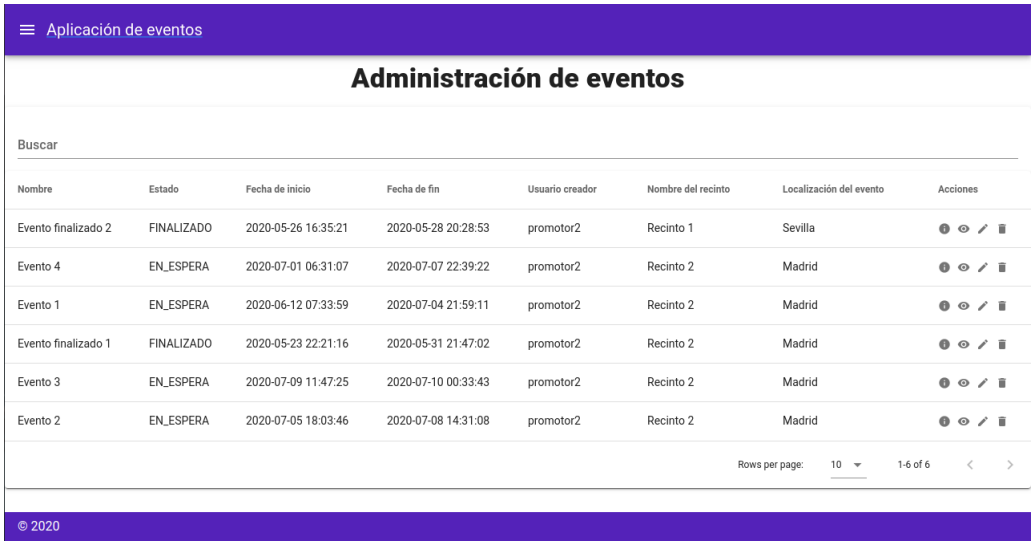

Figura 7.8: Pantalla principal de administración

- "Administrar recintos", en la Figura 7.10 el usuario crea, edita y borra recintos. Si el usuario pulsa sobre el botón "Crear recinto", le aparece la siguiente pantalla (Véase la Figura 7.12), donde rellena la información necesaria para la creación de un recinto.
- "Administrar usuarios", en esta Figura 7.11 se puede ver la página donde el usuario puede editar y borrar usuarios.

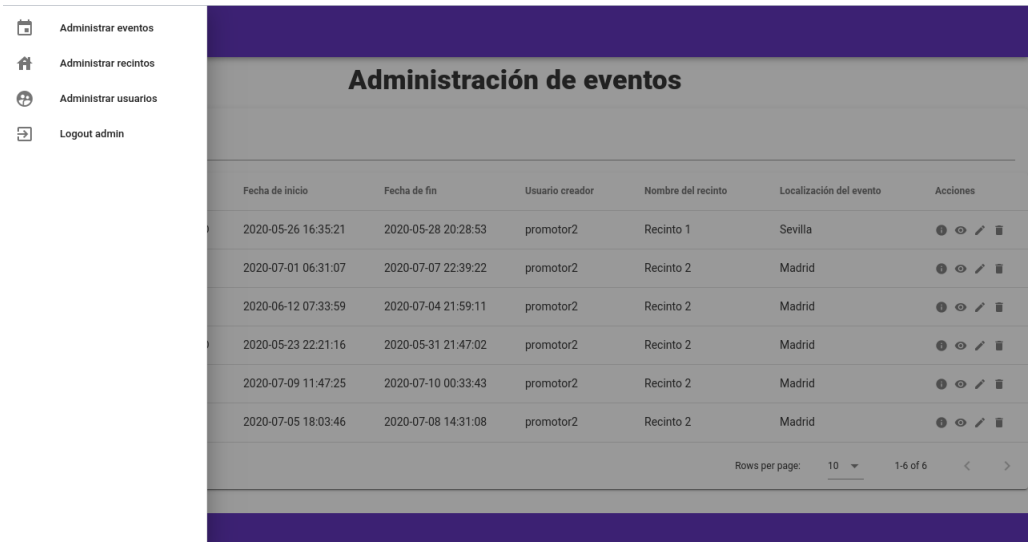

Figura 7.9: Pantalla de menú del usuario administrador

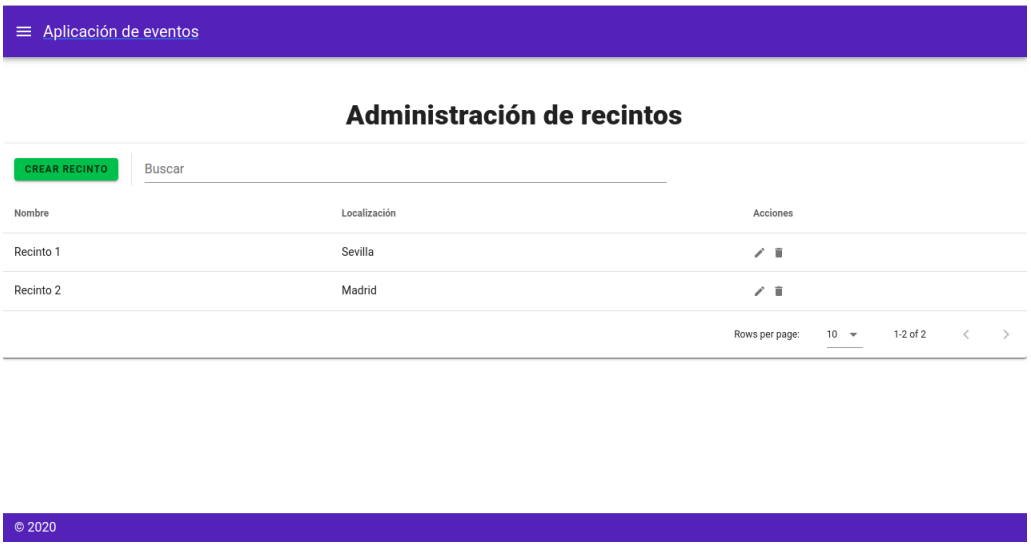

Figura 7.10: Pantalla de administración de recintos

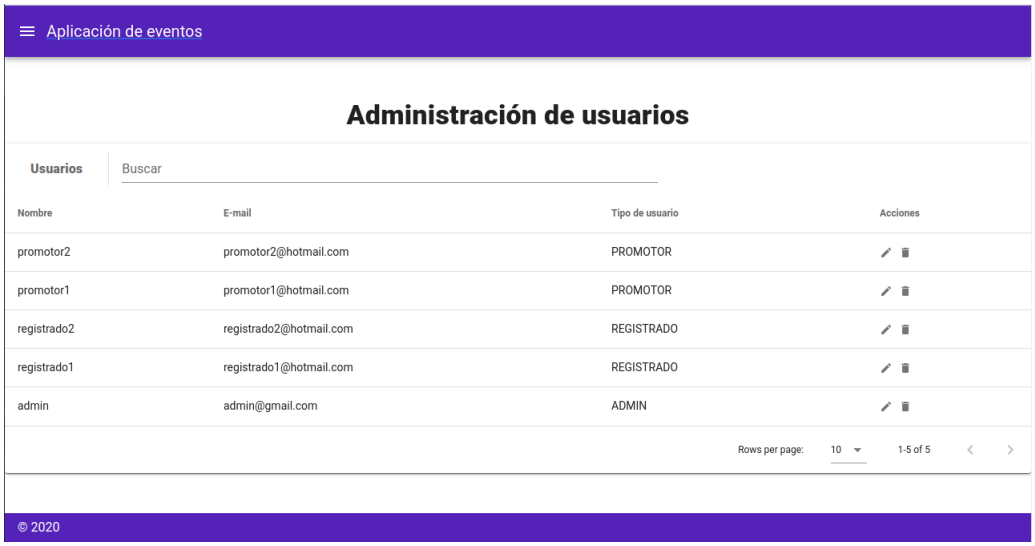

Figura 7.11: Pantalla de administración de usuarios

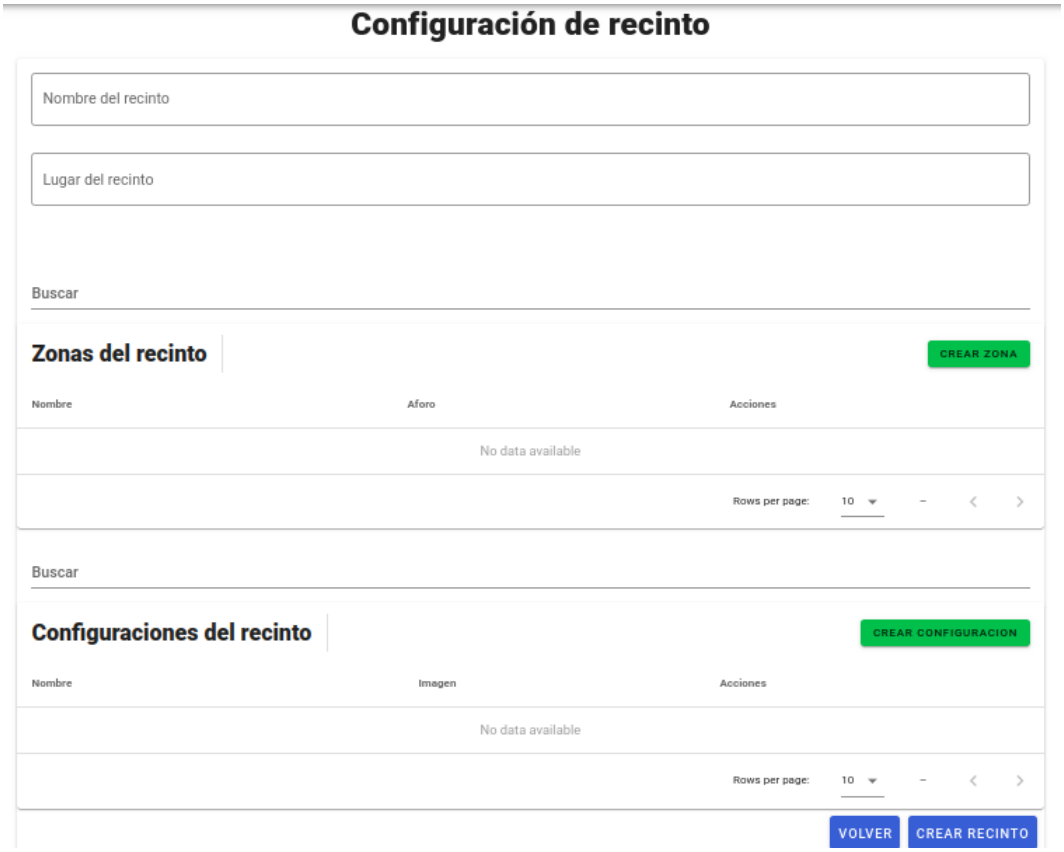

Figura 7.12: Pantalla de creación de recintos

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conclusiones

Se ha desarrollado una aplicación de venta de entradas de eventos altamente configurable que ofrece funcionalidades útiles que dieran valor a la aplicación, proporcionar información y estadísticas interesantes a los usuarios y funcionalidades nuevas, como es la reventa de entradas, visualización de estadísticas interesantes o soporte para gran cantidad de usuarios comprando entradas.

Al no estar familiarizado el autor del trabajo con todas las tecnologías que se utilizaron en el proyecto, hizo su desarrollo más complicado, sobre todo al principio mientras no se consiguió dominar todas las herramientas.

Durante el desarrollo del proyecto, el desarrollador del trabajo utilizó algunas herramientas que durante la carrera no había utilizado, como: Spring Boot, Vue, JavaScript, etc. Eso le sirvió para ampliar sus conocimientos sobre la Ingeniería del Software, mención que no había cursado. En relación con la mención que cursó, Tecnologías de la Información, se desplegó la aplicación en una plataforma en la nube, en lo que ya tenía experiencia, sin embargo se utilizó una plataforma, Heroku, que no había usado antes.

8.2 Trabajo futuro

Aunque se alcanzaron todos los objetivos que se definieron al comienzo del proyecto, durante el desarrollo del mismo aparecieron nuevas funcionalidades que se consideran de interés y que pueden ser abordadas como trabajo futuro. Son las siguientes:

- Mapa interactivo del recinto del evento en la página de compra de entradas, para que los usuarios puedan comprar sus entradas directamente desde él, pudiendo elegir entre las zonas del recinto que son de su interés.

- Mejora del filtrado de búsqueda de eventos, por ejemplo utilizar el API de Google Maps para mostrar un mapa en el que el usuario pueda filtrar la localización de los eventos que son de su interés, poder buscar los eventos más populares (basado en el número de entradas vendidas en las ediciones anteriores de dichos eventos), filtrar eventos por rangos de precios de sus entradas, etc.
- Adaptación de la interfaz de la aplicación para su uso con móviles.
- En la página de información de un evento mostrar la localización del mismo en un mapa, utilizando el API de Google Maps, en vez de mostrar solamente el nombre de la localización.
- Añadir nuevas estadísticas del usuario promotor, por ejemplo mostrando el número de usuarios que visitan la página de un evento.
- Poder compartir eventos a través de redes sociales.
- Proporcionar a los usuarios promotores y a los usuarios registrados, un calendario donde puedan ver sus eventos.

Apéndices

Apéndice A

Prototipos de pantallas

A continuación se muestran el resto de pantallas del diseño de la aplicación que no se han puesto en la Apartado 4.4.

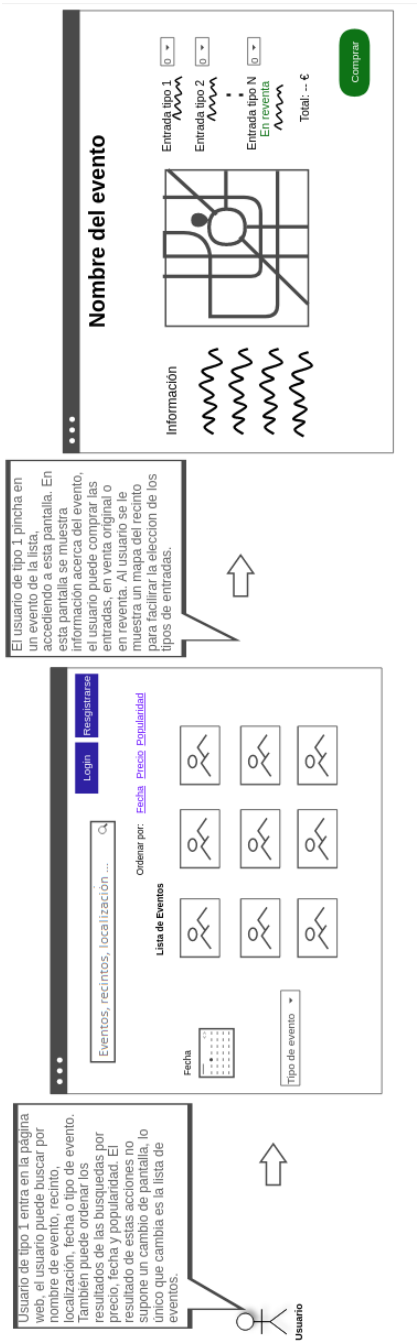

Figura A.1: Pantalla principal

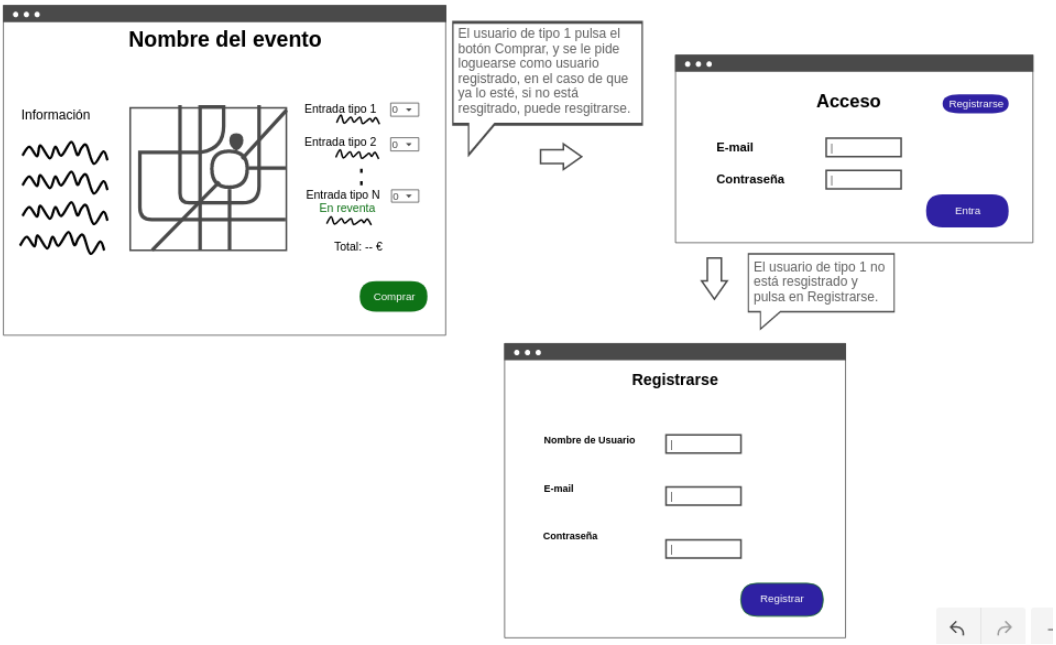

Figura A.2: Compra como usuario sin registrar

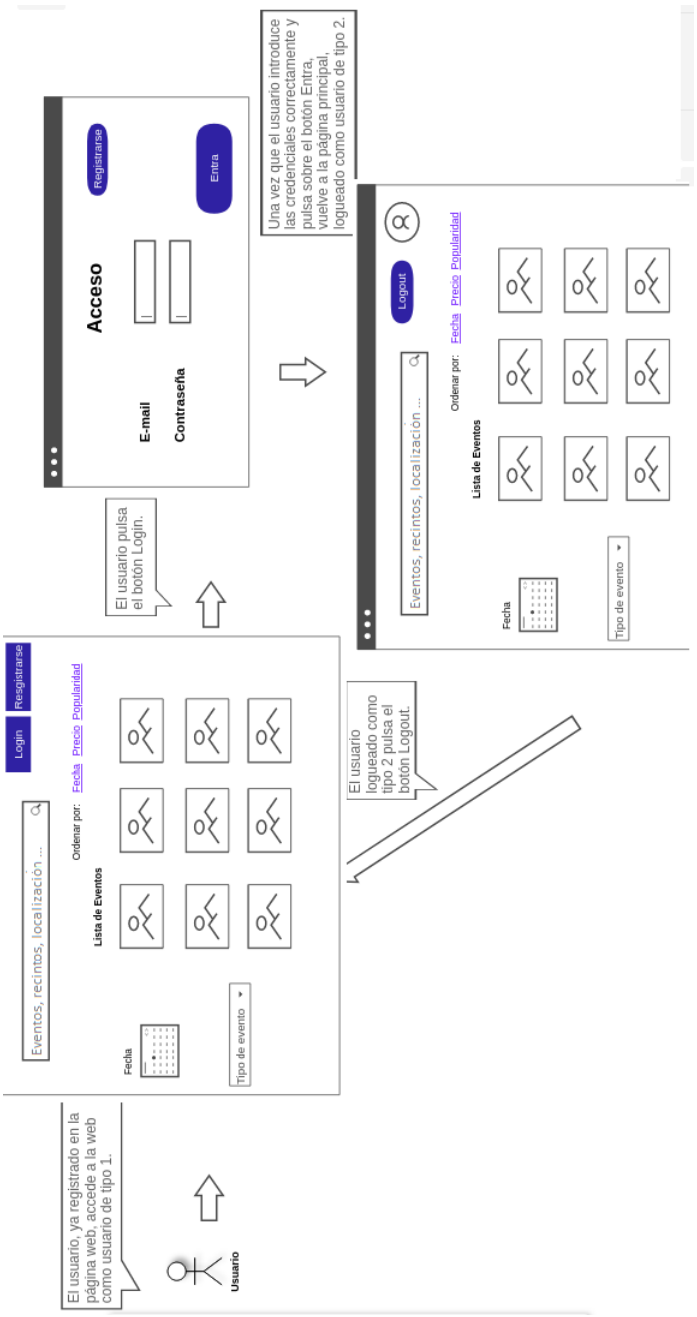

Figura A.3: Login en la aplicación

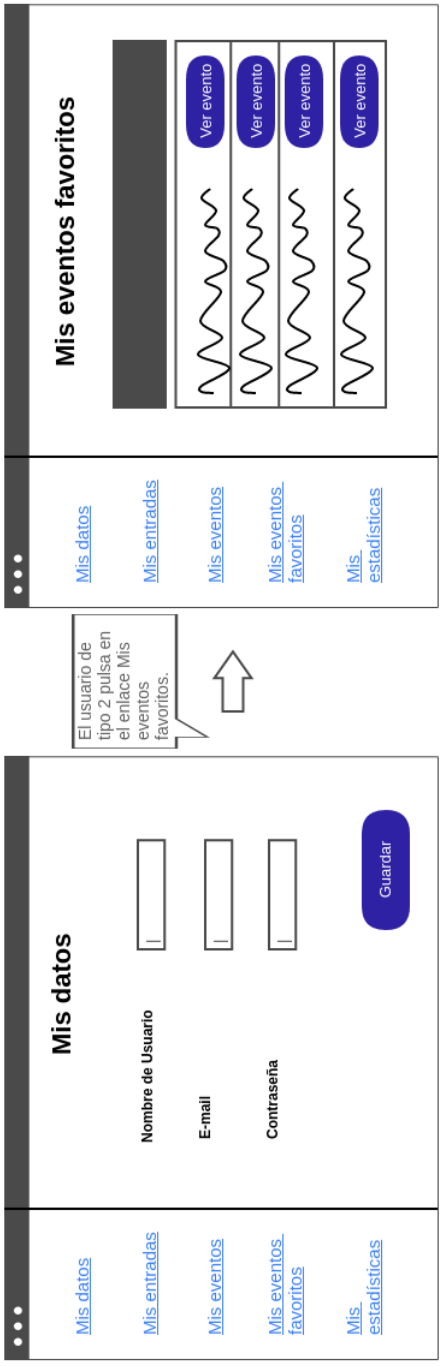

Figura A.4: Página de eventos favoritos

Figura A.5: Pantalla de estadísticas

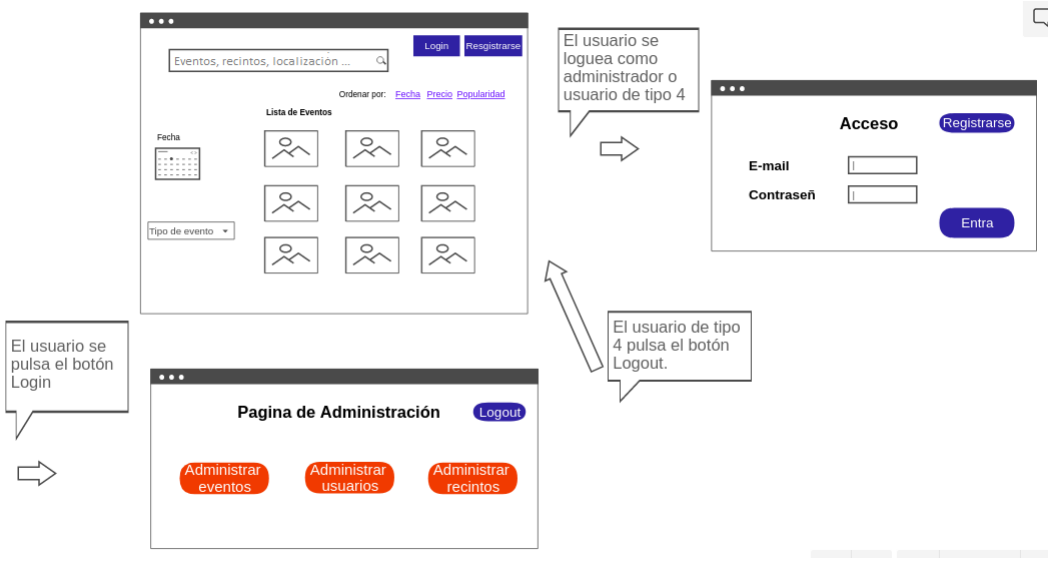

Figura A.6: Acceso a la página de administración

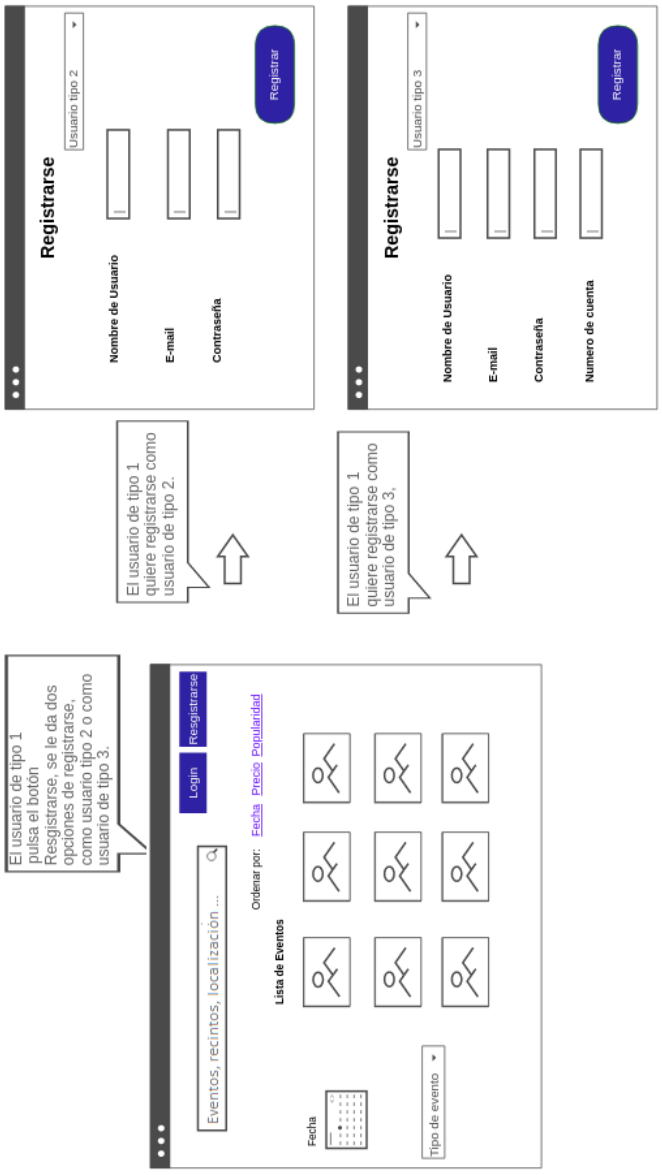

Figura A.7: Registro en la aplicación como Promotor o Usuario Registrado

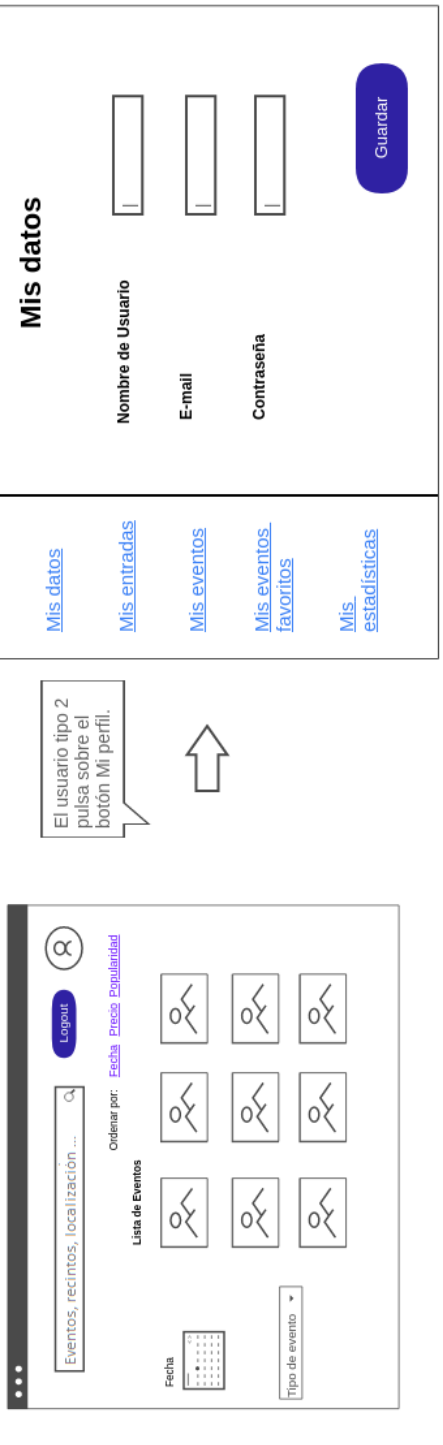

Figura A.8: Perfil del usuario registrado

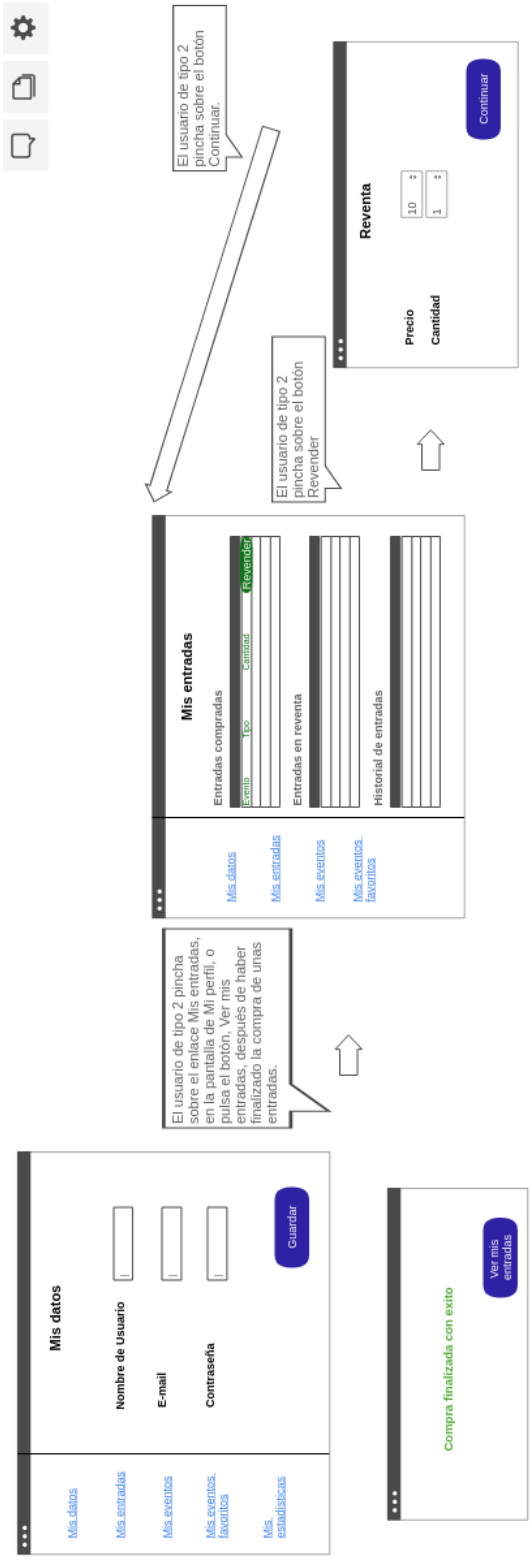

Figura A.9: Reventa de entradas

Apéndice B

Pantallas de la aplicación

A continuación se muestran el resto de pantallas de la aplicación que no se han puesto en la Capítulo 7.

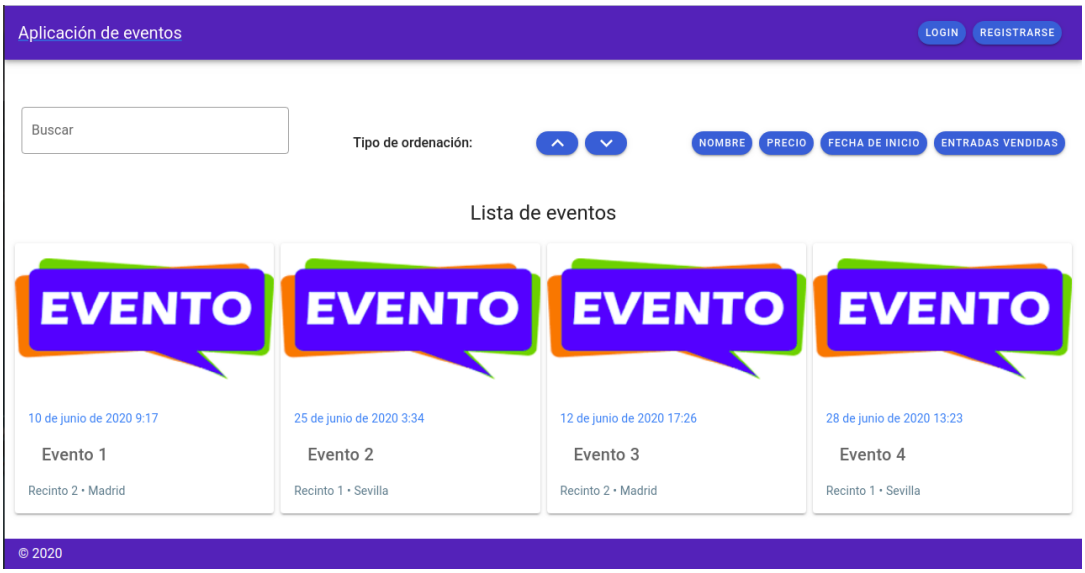

Figura B.1: Pantalla principal

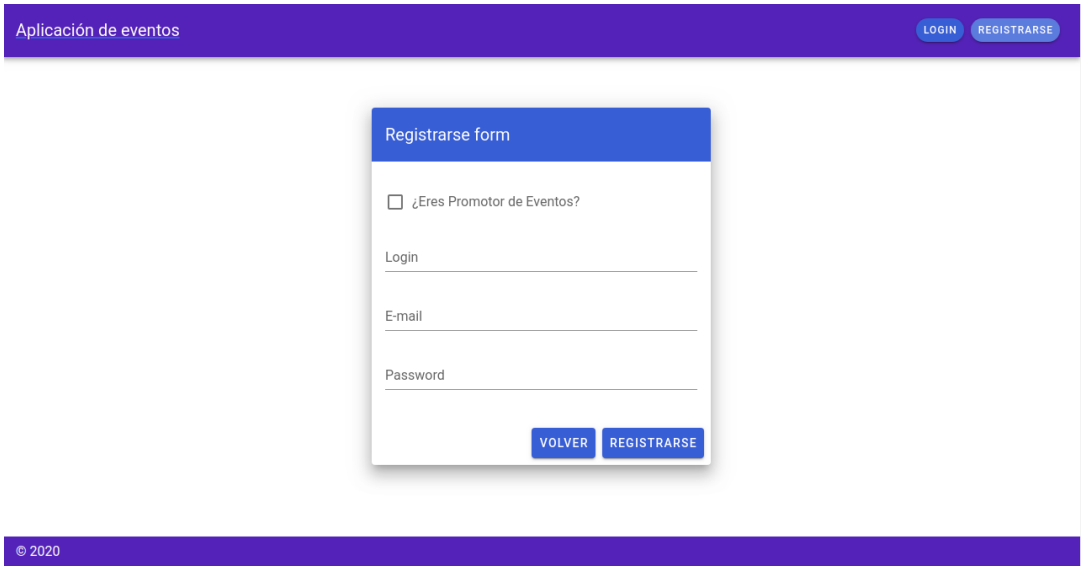

Figura B.2: Pantalla de registro

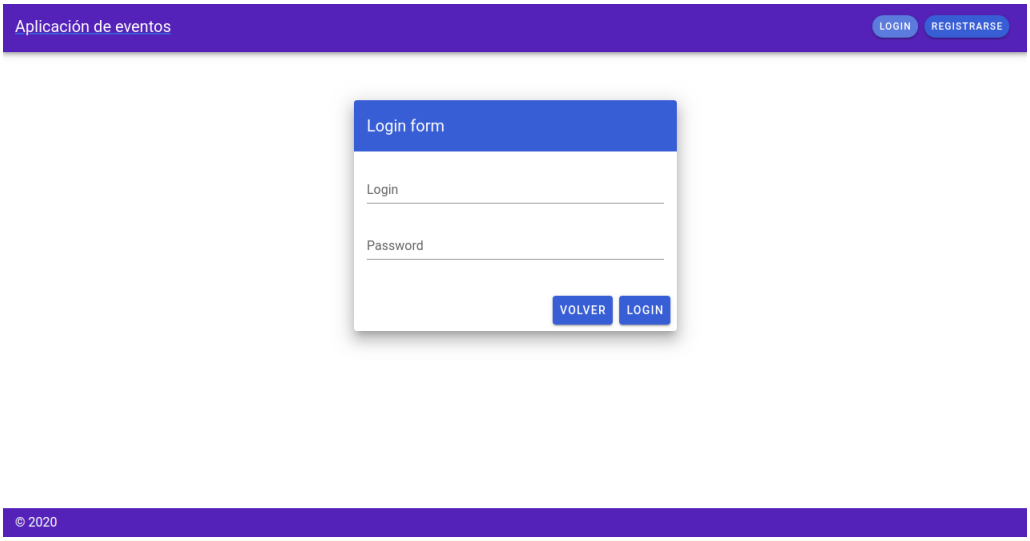

Figura B.3: Pantalla de login

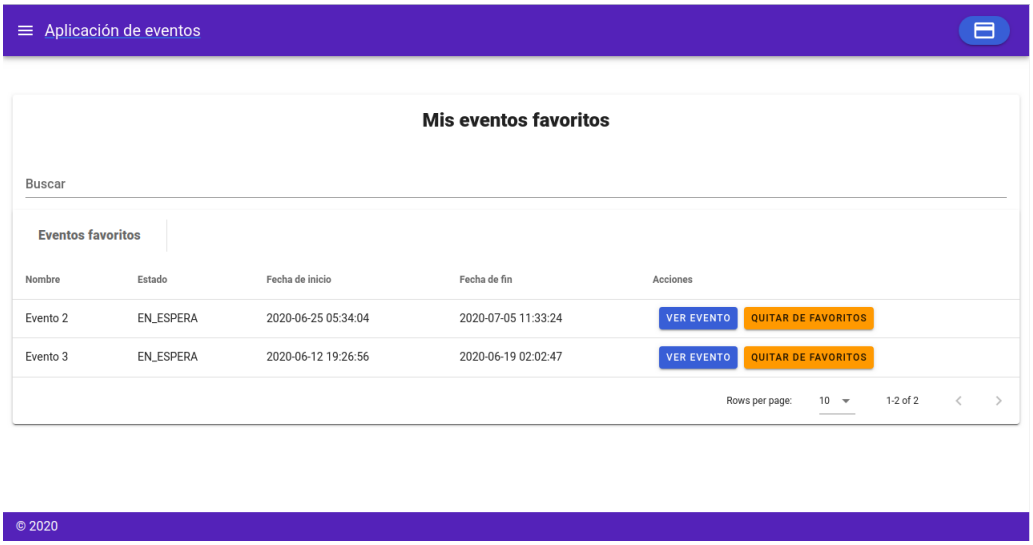

Figura B.4: Pantalla de eventos favoritos

Figura B.5: Pantalla de estadísticas de Usuario Registrado

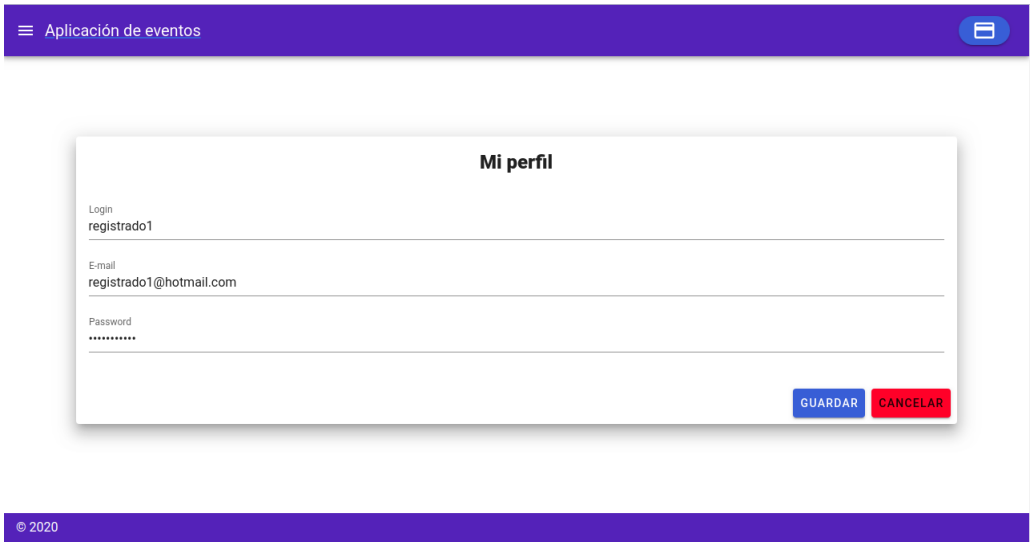

Figura B.6: Pantalla de perfil de usuario

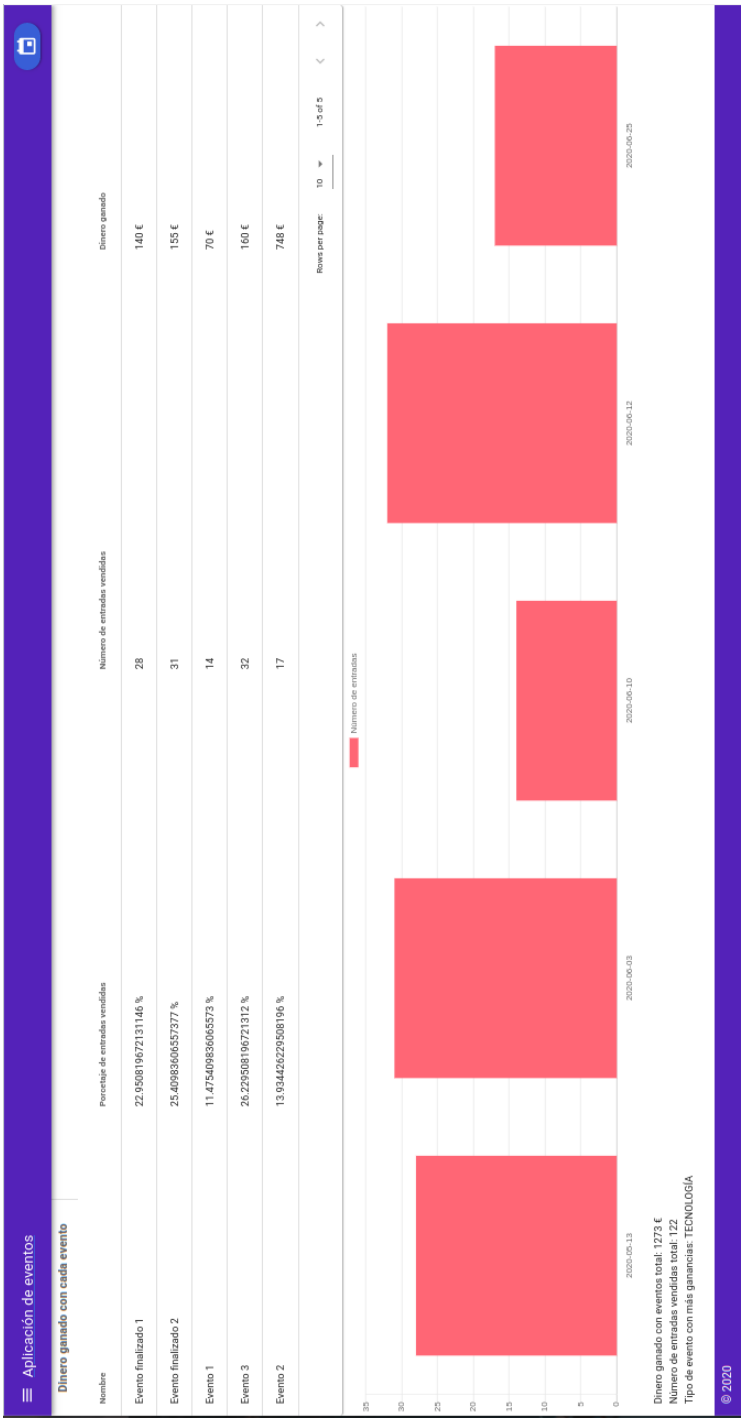

Figura B.7: Pantalla de estadísticas del promotor

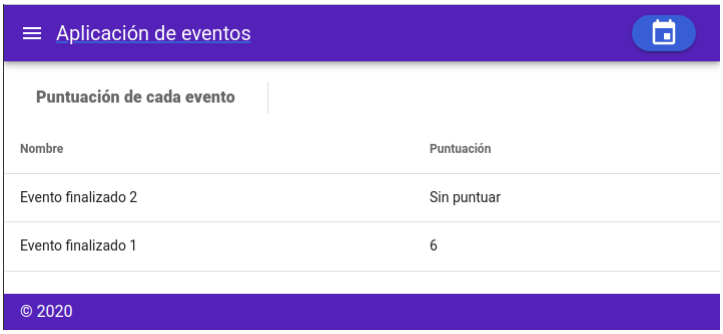

Figura B.8: Pantalla de puntuaciones

Figura B.9: Pantalla de información de evento

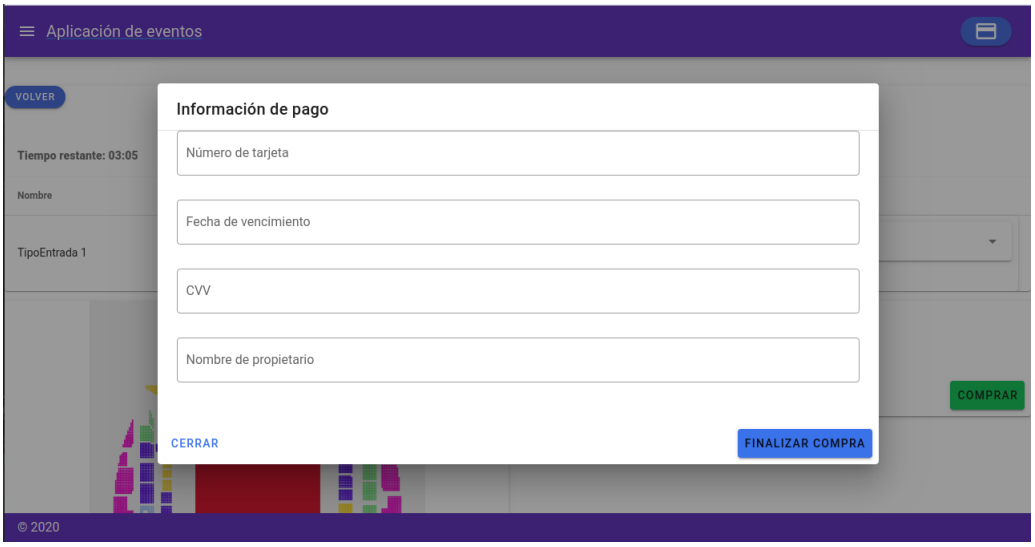

Figura B.10: Pantalla de información de compra

Apéndice C

Glosario de acrónimos

API *Application Programming Interface.*

CSS *Cascading Style Sheets.*

HTML *HyperText Markup Language.*

PaaS *Platform As A Service.*

PDF *Portable Document Format.*

REST *Representational State Transfer.*

URL *Uniform Resource Locator.*

Apéndice D

Glosario de términos

Diagrama de Gantt es una herramienta gráfica usada en la planificación de proyectos que permite determinar el tiempo previsto para la realización de una actividad. Permite realizar el seguimiento de dichas actividades conforme se avance en el desarrollo de las mismas.

Endpoint es el punto final de un canal de comunicación entre un API y otro sistema. Los puntos de contacto de la comunicación son considerados Endpoints.

Framework es una estructura base de módulos software predefinidos o herramientas como pueden ser programas o bibliotecas, que usados como punto de partida, simplifican la elaboración de tareas para la realización de un proyecto software.

Plataforma Como Servicio consiste en un servicio proporcionado por un proveedor que permite el acceso a un entorno cloud, en el cual los usuarios pueden crear y desplegar aplicaciones. Este proveedor es el encargado de facilitar la infraestructura subyacente a los usuarios.

Sprint dentro del desarrollo software, un Sprint es un período de tiempo durante el cual un trabajo determinado debe ser realizado y posteriormente revisado.

Bibliografía

[1] "Marco de trabajo de scrum," 2017. [En línea]. Disponible en: https://www.youtube.com/watch?v=P25JP0u6UKw/

[2] Z. C. de Mariño, "Roles de scrum." [En línea]. Disponible en: https://xn--zoraidaceballosdemario-4ec.info/scrum/scrum-roles-y-responsabilidades-del-scrum-team/

[3] "Página web del patrón de diseño fachada," 2018. [En línea]. Disponible en: https://elblogdelprogramador.home.blog/2018/11/30/patron-de-diseno-facade/

[4] "Página web del patrón de diseño mvvm," 2020. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Model-view-viewmodel

[5] "Página web de attendize," 2020. [En línea]. Disponible en: https://www.attendize.com/

[6] "Página web de ticketea(eventbrite)," 2020. [En línea]. Disponible en: https://www.eventbrite.es/

[7] "Página web de ticketbud," 2020. [En línea]. Disponible en: https://www.ticketbud.com/

[8] E. You, "Página web de vue.js," 2020. [En línea]. Disponible en: https://vuejs.org/

[9] "Guía tutorial de vue," 2020. [En línea]. Disponible en: https://vuejs.org/v2/guide/#

[10] I. o. i. a. VMware, "Página web de spring boot," 2020. [En línea]. Disponible en: https://spring.io/projects/spring-boot/

[11] A. Hughes, "Tutorial de spring boot con vue," 2018. [En línea]. Disponible en: https://developer.okta.com/blog/2018/11/20/build-crud-spring-and-vue/

[12] Pluralsight, "Página web de javascript," 2020. [En línea]. Disponible en: https://www.javascript.com/

[13] "Página web de hibernate," 2020. [En línea]. Disponible en: https://hibernate.org/

[14] T. P. G. D. Group, "Página web de postgresql," 2020. [En línea]. Disponible en: https://www.postgresql.org/

[15] I. o. i. a. VMware, "Página web de spring," 2020. [En línea]. Disponible en: https://spring.io/

[16] "Guía tutorial de spring," 2020. [En línea]. Disponible en: https://spring.io/guides/

[17] L. Vuetify, "Página web de vuetify," 2020. [En línea]. Disponible en: https://vuetifyjs.com/

[18] R. Data, "Página web de html," 2020. [En línea]. Disponible en: https://www.w3schools.com/html/

[19] ——, "Página web de css," 2020. [En línea]. Disponible en: https://www.w3schools.com/css/

[20] "Página web de heroku," 2020. [En línea]. Disponible en: https://heroku.com/

[21] "Página web de scrum," 2020. [En línea]. Disponible en: https://www.scrum.org/

[22] "Página web de gitlab," 2020. [En línea]. Disponible en: https://gitlab.lbd.org.es/

[23] "Página web de resttemlate," 2020. [En línea]. Disponible en: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html