GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

Aplicación web de reservas de actividades y restaurantes para hoteles

---

Autora:
Gema JUÁREZ ALMAZÁN

Supervisor:
Miguel SALAS ALEGRE
Tutora académica:
M. Ángeles LÓPEZ MALO

Fecha de lectura: 17 de septiembre de 2019

Curso académico: 2018/2019

Resumen

En este documento se explica el proceso de desarrollo de una aplicación que servirá para que los trabajadores de un hotel puedan llevar a cabo, de forma sencilla, la gestión de actividades y restaurantes ofrecidas por el hotel a sus clientes. Estos empleados podrán gestionar las reservas de estas actividades, reservadas previamente por los clientes. De la misma forma, esta aplicación cuenta también con una segunda parte que irá destinada a los clientes del hotel, donde estos podrán realizar las reservas de las actividades.

Las interfaces de ambas partes de la aplicación se implementarán con *Vue.js* y *JavaScript*, lo que dará un mejor comportamiento en la visualización de las interfaces web. Además, los clientes del hotel podrán realizar las reservas con unos dispositivos con tecnología *RFID*.

Palabras clave

Reservas, actividades, RFID, empresas hoteleras, Vue.js, JavaScript.

Keywords

Bookings, *activities*, *RFID*, *hotel companies*, *Vue.js*, *JavaScript*.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Este proyecto se desarrolló en una empresa llamada Easygoband. Esta nació en la compañía Paynopain y actualmente se encuentra ejerciendo de forma individual en la provincia de Castellón. Destaca en el sector terciario y se encarga de desarrollar aplicaciones que ofrezcan facilidades, seguridad y gestión en el control de accesos a eventos, así como también alternativas a la identificación de personas o al pago electrónico.

Easygoband no sólo se mueve por el territorio nacional, sino que se ha extendido por Europa y Latinoamérica. A pesar de ser una empresa pequeña, tiene un amplio número de trabajadores, tanto en su oficina en Castellón como en su segunda oficina en Málaga. Además, cuentan con algún comercial residente fuera de España facilitando la expansión de la empresa.

En la sede de Castellón, cuentan con aproximadamente veinticinco trabajadores. Los departamentos que forman la empresa son los siguientes: Márketing, administración, técnico y comercial. El desarrollo de este proyecto se ha llevado a cabo dentro del departamento técnico.

La empresa ha realizado grandes proyectos como Electrosplash y además ha participado con grandes empresas como Aquarama, Oasis Hotels & Resorts, Vodafone o la Universidad Jaume I.

Este proyecto surgió de la necesidad de facilitar la gestión de las reservas de los restaurantes y actividades que ofrecen algunas empresas hoteleras. Esto no se refiere sólo de cara a cómo administrarlas, sino también al difícil control que conlleva gestionar las reservas de estas actividades y restaurantes. Debido a este problema, Easygoband propone una aplicación para que los trabajadores de los hoteles puedan añadir, visualizar o editar cada actividad proporcionada. Además, estos trabajadores podrán administrar las futuras reservas de sus clientes. Con este proyecto mejora el rendimiento y la eficiencia en el negocio, dado que es mucho más sencillo llevar a cabo este trabajo por parte de la empresa hotelera y por parte sus clientes.

Paralelamente, la parte destinada al cliente se centra en realizar de forma sencilla la reserva de una actividad. Por ello, cuenta con una plataforma en la que el cliente podrá visualizar cada una de las actividades, seleccionar una y poder realizar una reserva. Esto último cuenta con un pequeño detalle y es que Easygoband trabaja con una serie de dispositivos que utilizan la tecnología *RFID*.

La tecnología *RFID* se basa en el almacenamiento y recuperación de datos de forma remota. Normalmente, esto se usa en etiquetas o tarjetas. Sin embargo, para un mejor uso y mayor simplicidad para el cliente, la empresa utiliza pulseras. Gracias a esto, el cliente podrá reservar los restaurantes y las actividades de las empresas hoteleras mediante el uso

de estos dispositivos. El cliente también podrá realizar estas reservas de manera tradicional con nombre y apellidos.

Esta sencillez permitirá a la empresa obtener beneficios debido al control de las actividades que proporcionan y también al posible incremento de clientes gracias a la simplicidad ofrecida con dicho servicio.

1.2. Objetivos del proyecto

Los objetivos del proyecto se centran en crear una aplicación que permita eliminar o reducir la dificultad de controlar la reserva de actividades en un hotel. Por ello, hay una serie de subobjetivos a mencionar dependiendo de si la aplicación a utilizar es de la parte del administrador o la del cliente.

Por una parte, en la parte destinada al administrador se pueden encontrar los siguientes objetivos:

- Realizar sin ningún tipo de inconveniente la incorporación de las nuevas actividades que puedan surgir en un momento dado con total sencillez, así como también su posterior posible modificación.
- Gestionar no sólo los datos relacionados con la información de una actividad (nombre, descripción, precio…), sino también la ocupación de cada una de ellas dada una fecha seleccionada, con sus correspondientes sesiones y grupos.
- Añadir un estilo que les haga sentir que la aplicación ha sido destinada únicamente para ellos y que parezca un complemento más de la empresa.
- Facilidad en la obtención de todos los datos que estén relacionados con las reservas que hayan sido realizadas por sus clientes.

Estos objetivos permitirán obtener una serie de beneficios:

- Facilidad en la gestión de las actividades.
- Mejoras que interese la imagen del hotel y esto le permitirá tener ventaja frente al resto de las empresas hoteleras.
- Mayor eficiencia al trabajador encargado en la administración de las actividades.
- Incremento de los beneficios de la empresa hotelera.

Por la otra parte, queremos conseguir los siguientes objetivos:

- Los clientes puedan reservar una actividad ofrecida incluyendo el uso de dispositivos con tecnología *RFID*.
- Los clientes puedan cancelar una reserva de forma sencilla.

Los beneficios conseguidos a raíz de estos objetivos son los siguientes:

- Facilidad a la hora de realizar una reserva, de forma segura y a la que puedan acceder en cualquier momento.
- Flexibilidad en el caso de querer cancelar una reserva. Esto también incluye que en el momento en el que se cancela la reserva por la parte del cliente, el administrador observe dicha cancelación al instante.

- Mayor comodidad por parte de los clientes que les empuje en seguir confiando en este tipo de estancias, así como también en la valoración de las mismas.

De forma general, contando con todos los objetivos mencionados se obtienen los siguientes beneficios:

- Sostenibilidad, dado que los cambios que se puedan producir en un futuro permitirá que sendas partes continúen actualizándose y no queden obsoletas.
- Facilidad en el rediseño de las aplicaciones. *Vue.js* permite modificarlas por componentes independientes sin que afecte al resto de la aplicación.
- Mayor eficiencia.
- Interfaz apropiada para el usuario ya que facilitará al máximo la comunicación y el uso del producto.

1.3. Estructura de la memoria

Para una mejor organización del contenido de este documento, voy a comentar de qué tratará cada capítulo.

En primer lugar, el capítulo 2 hace referencia a una descripción mucho más detallada del proyecto y de las tecnologías usadas.

En segundo lugar, el capítulo 3 habla acerca de la planificación del proyecto. Muestra la información relacionada con la metodología usada, la estimación de recursos y cómo ha sido el seguimiento del proyecto.

A continuación, en el capítulo 4 se detalla el análisis y el diseño de la aplicación, diferenciando la parte dirigida a los empleados como la dirigida a los clientes del hotel.

Después, el capítulo 5 muestra la implementación de dicha aplicación, así como el formato de los datos y las características del mismo.

Finalmente, el documento terminará con las conclusiones obtenidas tras el proceso de desarrollo, así como también una serie de anexos relacionados con él y la bibliografía que ha sido utilizada para llevar a cabo este proyecto.

Capítulo 2

Descripción del proyecto

2.1. Aplicación

Para empezar con la descripción del proyecto se debe tener en cuenta el tipo de usuario al que va destinado y cómo se estaba llevando anteriormente la gestión de reservas cuando este proyecto aún no existía.

Los trabajadores del hotel gestionaban las reservas manualmente. Esto significa que utilizaban herramientas como un documento de papel o un programa de ordenador de hojas de cálculo. Esto posibilitaba la opción de que se produjeran errores y un descenso de rendimiento a larga. Por ello nació esta idea de proyecto.

Como se ha comentado en el capítulo 1, la aplicación se divide en dos partes: la del administrador y la del cliente.

Por un lado, está la parte destinada al trabajador del hotel. Dicho trabajador podrá llevar a cabo tres acciones: Cambiar la configuración relacionada con el estilo e información del hotel, gestionar las actividades y restaurantes y visualizar el listado de reservas de todos sus clientes.

El administrador podrá cambiar la configuración del hotel, pudiendo actualizar la información relacionada con el nombre del mismo, el logotipo y el tipo de moneda. Asimismo, escogerá un primer y segundo color que definirá el estilo que tendrá la parte destinada al cliente. De la misma manera, existen dos opciones para escoger el tipo de contraste del texto sobre estos colores.

Paralelamente, el administrador podrá gestionar las actividades que proporciona el hotel. Esto es visualizar el listado de las mismas, poder desactivar o editar cada una de ellas y además la posibilidad de crear una actividad.

En cada actividad se ve la información detallada de las misma, incluyendo el calendario en el que se muestran las fechas en las que se encuentra disponible. En cada fecha se encuentran las sesiones y grupos que forman la actividad. Algunos ejemplos de actividades que se podrían llevar a cabo podría ser tener opción a acceder a un spa completo al que asistir con la familia, una cena romántica en pareja o una guardería en el que se pueda dejar a los niños si se necesita acudir a otra actividad.

En segundo lugar, los clientes podrán ver sus reservas y también todas las actividades disponibles. Algunas actividades pueden estar incluidas en la tarifa contratada con el hotel. Dichas actividades aparecerán sin coste alguno. Con respecto al resto de actividades, el cliente elegirá la actividad que más le interese y vería de forma detallada toda la información de la misma. Si el cliente decide continuar con la reserva, tendrá que escoger qué sesión y grupo desea y el nombre al que irá reservado. Al final de esto, visualizará un resumen de toda la información de la reserva y podrá llevarla a cabo

mediante la pulsera o mediante un nombre, dependiendo de si la actividad ofrece la posibilidad, o no, de reservarla mediante la tecnología *RFID*.

2.2. Tecnologías

Todo este proceso, incluyendo el frontend de ambas partes de la aplicación, se ha desarrollado mediante el uso de *JavaScript*, *Vue.js* y un conjunto de plugins y librerías de *Vue.js*: *Vuex, Vuetify y Vue-Router*.

Para empezar tenemos *JavaScript* [1], un lenguaje de programación basado en *EMACScript*[1] que permite desarrollar el comportamiento de una web. Proporciona mejoras en la interfaz debido a componentes dinámicos.

A continuación está *Vue.js* [2], un marco de *JavaScript* con el que se puede componer interfaces de usuario y aplicaciones.

Dentro de *Vue.js* se encuentran varias librerías que amplían esta funcionalidad y que la hacen mucho más sencilla.

Con *Vuex* [3] se puede obtener un almacenamiento donde se encuentran todos los componentes de la aplicación, garantizando el estado en el que se encuentra mediante un conjunto de reglas y proporcionando un conjunto de funcionalidades avanzadas.

Vuetify [4] es un plugin que mejora el diseño de la interfaz de usuario y que da un conjunto de componentes creados que facilita este sin que el desarrollador gaste parte de su tiempo en diseñarlos. Además, estos componentes son fácilmente personalizables lo que amplía aún más la utilidad de este plugin.

Por último está *Vue-Router* [5], una manera simple de añadir rutas a los componentes *vue* que se tienen creados y que facilita el renderizado del mismo.

Todas estas tecnologías se comentarán más ampliamente en la implementación del proyecto, en el capítulo 5.

Finalmente, se usa la herramienta de *Material Icons Design* [6] para la realización de una interfaz con iconos que mejoren la visualización de algunas funcionalidades.

2.3. Herramientas

De igual manera, lo que conlleva a la realización de un proyecto no sólo es la idea y cómo desarrollarla, que también, sino que se requieren un conjunto de herramientas para que esto pueda funcionar.

[1]Es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje *JavaScript*[7] propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262.

En primer lugar, se ha usado *Visual Studio Code* como editor de código. Gracias a esta herramienta ha sido muy útil el almacenamiento del código en remoto y la actualización a este cuando se han realizado cambios en el entorno local. Es aquí donde entra en juego la segunda herramienta: *Gitlab*.

Gitlab es un servicio de red que permite el control de versiones y desarrollo de un software determinado. La interacción con este mecanismo ha facilitado la revisión del código por parte de la empresa. En posteriores reuniones esta herramienta ha servido para visualizar cómo estaba desarrollado el proyecto y si había algo que mejorar.

Después se encuentra *Slack*, una herramienta que ha proporcionado una comunicación con la empresa para implementar el proyecto. Ha sido una forma sencilla de expresar dudas y ser resueltas, de acordar cierto tipo de ideas y quedar estas guardadas para luego tornar a ellas en caso de querer recordarlas. Además, ha permitido una mejor manera de planificar el trabajo.

Finalmente está *Teamwork*, una herramienta que ha permitido calcular las horas invertidas en cada parte del proyecto, empezando por la documentación y terminando por la implementación. Asimismo ha aportado una mejor estructura de lo que se tiene realizado y de lo que queda por realizar.

Capítulo 3

Planificación del proyecto

3.1. Metodología

Para realizar el proyecto se llevó a cabo una metodología similar a SCRUM. Esta metodología concreta consistió en:

1. Una reunión inicial acerca de cómo se llevaría a cabo el proyecto.
2. Cada una o dos semanas se realizaba un sprint, haciendo un total de 9 sprints. En cada sprint se revisaba el proyecto para confirmar si la estimación de la planificación se estaba llevando a cabo o se necesitaba alguna variación. En una de las reuniones fue cuando se confirmó que el proyecto cambiaría de trayectoria y por ello hubieron algunos cambios. Esto se comentará en el apartado de *Seguimiento del proyecto*.
3. Cada tres días se convocaban pequeñas reuniones donde se comentaban los avances del proyectos. Estas reuniones no tomaban más de 15 minutos y era una forma de no sólo confirmar que el proyecto seguía una continuación sino que también servían para presentar dudas y resolverlas, ya que muchas de ellas iban surgiendo conforme el proyecto fue desarrollándose. Aquí la empresa proporcionó mucha ayuda y un gran interés para que el alumno de prácticas avanzara de manera más rápida y eficiente.

También cabe mencionar los roles que conforman el proyecto:

- *Product owner:* Decide qué se debe desarrollar.
- *Project manager:* Coordina el proyecto.
- *Tech leader:* Diseña la arquitectura del sistema.

Los dos primeros roles los llevó a cabo el supervisor del proyecto mientras que el rol de *tech leader* fue un miembro de la empresa encargada en el desarrollo frontend.

3.2. Planificación

En primer lugar se llevó a cabo una planificación del proyecto inicial, la cual se ve en la tabla 1. Para ello, se realizó una tabla con el nombre de cada tarea a desarrollar junto con su duración, el inicio de la tarea, su fin y si dicha tarea tiene otra antecesora. La tabla comienza con el inicio del proyecto. Esto se subdivide en la definición del proyecto, el método del trabajo y el formato y estándar del mismo.

La siguiente fase es aquella que trata de la documentación y del inicio de la planificación del proyecto. Aquí entra la parte en la que se investigará acerca de qué entidades se manejarán o cuáles podrán ser más adecuadas, así como también las funcionalidades a desarrollar.

Tras esto, comenzaría la planificación del proyecto que tratará de especificar las entidades, detallar los casos de uso, realizar el diseño conceptual de datos y el prototipado de la aplicación. En este último punto fue la empresa la que se encargó de realizar el conjunto de prototipos que el cliente desearía. A pesar de que este trabajo lo realizó un miembro de la empresa anteriormente, se llevó a cabo un tiempo estimado de 15h en el que se revisaron cada uno de los prototipos, se comentaron cómo se podría desarrollar cada uno, los componentes a utilizar, etc.

Después se empezaría la fase de implementación del proyecto, la cual consta de varias tareas de investigación más las tareas de implementación. En el caso de investigación debemos tener en cuenta la formación en las distintas competencias que permitirán el desarrollo de la aplicación. Dicha formación se basaría en un conjunto de tutoriales [8] de *Vuetify, Vue.js* y *Firebase*. Una vez adquiridos estos conocimientos, se comenzó la implementación del proyecto, donde comienza el desarrollo usando estas tecnologías. Esta tarea es larga debido a que su duración se prolonga por la implementación de no sólo la parte destinada al administrador, sino también a la parte destinada al cliente.

Una vez la implementación estuviera realizada se pasaría a la creación de usuarios para poner en marcha el funcionamiento de la aplicación.

Finalmente, se llevarían a cabo una serie de mejoras en las funcionalidades de la aplicación, así como también agregación de otras nuevas que permitiera que esta quedara más completa.

	Nombre de la tarea		**Duración**	**Inicio**	**Fin**	**Predecesores**
1	**Inicio**		**4h**			
	1.1	Definir el proyecto con el tutor y supervisor	1h	31/01/19	31/01/19	
	1.2	Definir método de trabajo y documentación	2h	04/02/19	04/02/19	1.1
	1.3	Definir formato y estándares de trabajo	1h	04/02/19	04/02/19	1.2
2	**Documentar y planificar el proyecto**		**2h**			
	2.1	Investigación de las entidades del proyecto	1h	04/02/19	04/02/19	1
	2.2	Investigación de las funcionalidades a desarrollar	1h	05/02/19	05/02/19	2.1
3	**Planificación del proyecto**		**35h**			
	3.1	Especificación de las entidades del proyecto	4h	05/02/19	05/02/19	2
	3.2	Especificación de los casos de uso	4h	06/02/19	06/02/19	3.1
	3.3	Revisión de toda la documentación	8h	07/02/19	08/02/19	3.2
	3.4	Diseño conceptual de la base de datos	4h	11/02/19	11/02/19	3.3
	3.5	Prototipo de la aplicación	15h	12/02/19	15/02/19	3.4
4	**Implementación del proyecto**		**190h**			

	4.1	Estudio de JavaScript	20h	18/02/19	22/02/19	
	4.2	Estudio de Vue y Firebase	20h	25/02/19	01/03/19	
	4.3	Implementación de los prototipos	150h	04/03/19	25/03/19	3
5	**Creación y gestión de los usuarios**		**25h**			
	5.1	Comunicación de la aplicación con usuarios	25h	25/04/19	03/05/19	
6	**Mejora del proyecto**		**44h**			
	6.1	Diseño de nuevas funcionalidades	4h	06/05/19	06/05/19	
	6.2	Implementación de nuevas funcionalidades	40h	07/05/19	20/05/19	6.1

Tabla 1: Planificación inicial del proyecto.

3.3. Estimación de recursos y costes del proyecto

En cuanto a costes y recursos del proyecto nos encontramos con diferentes estimaciones. Por un lado está el coste del programador. Para calcularlo debemos tener en cuenta el número de horas trabajadas y el precio por hora. Las horas trabajadas son 300 y el precio medio por hora oscila entre 8/9€. Esto hace un total de 2400/2700€.

Si a este total le añadimos un consultor que revisará el proyecto cerca del 10% de las horas trabajadas, 30 horas, a un precio de 12€ a 15€, sumaría entre 360€ y 450€. Esto acumularía con el total del coste del programador un valor de, en el mejor de los casos, 2760€. Si por el contrario nos ponemos en el peor de los casos, este valor subiría a 3150€.

Por otro lado está las herramientas utilizadas para que este proyecto se lleve a cabo. Se ha utilizado el programa de Visual Studio Code el cual fue gratuito.

De la misma forma, al programador se le formó con un cursillo para mejorar su aportación al proyecto. Dicho cursillo costó cerca de 20€, lo que supondría un total de 2780€ o 3170€.

A continuación, en cuanto al hardware utilizado, entra en ello el monitor de la pantalla, teclado, ratón y torre, con un precio estimado en total de unos 400€. Este hardware tiene una duración estimada de 5 años (1825 días), lo que si contamos el uso llevado a cabo en las 300 horas (75 días), ronda un precio de 16,44€.

Si juntamos este valor con todos los anteriores, tendríamos un coste de 2796,44€ en el mejor de los casos y un coste de 3186,44€ en el peor. A esto hay que añadirle el 20% de gastos de luz, agua, etc. que pueden producirse en la empresa, dejando un total de 3355,73€ en el mejor de los casos y 3823,73€ en el peor.

	Mejor caso	Peor caso
Horas trabajadas por precio media	300 x 8€ = 2400€	300 x 9€ = 2700€
Revisión consultor	300 x 0,10 x 12 = 360€	300 x 0,10 x 15 = 450€
Visual Code Studio	Gratis	Gratis
Cursillo	20€	20€
Hardware	16,44€	16,44€
20% costes indirectos	2796,44 x 0,2 = 559,29€	3186,44 x 0,2 = 637,29€
Total del coste	**3355,73€**	**3823,73€**

Tabla 2: Tabla resumen del coste total del proyecto en el mejor o peor de los casos.

3.4. Seguimiento del proyecto

En un principio, la planificación inicial del proyecto se estuvo llevando al pie de la letra hasta que se llegó a la tarea de la implementación del proyecto (véase en la tabla 1). Aquí no sólo se necesitaron más horas en cuanto a la formación, sino que hubo un cambio en cómo se implementaría el proyecto. Por ello, la fase de implementación dejó de añadir la competencia de una funcionalidad completa para únicamente realizar las el frontend de la parte del panel de administración y del panel destinado a los clientes. Esto es debido a que la empresa cambió sus preferencias en cuanto al desarrollo del proyecto y prefirió que este se centrara más en el frontend y no realizar el backend.

De esta forma también cambió su punto de vista con respecto al almacenamiento de los datos en Firebase. Al principio la idea era no sólo conocer dicho almacenaje sino además todo lo necesario para que los datos permanecieran seguros y con varias técnicas de seguridad y privacidad. Dado que la prioridad en este momento cambió a únicamente el desarrollo de frontend, el almacenamiento de datos quedó en un segundo plano y los datos por mostrar quedaron almacenados en el sistema *localStorage* del navegador. A pesar de ello, cumplen la misma estructura que fue pensada para *Firebase*.

Debido a los diversos cambios producidos con respecto a la planificación inicial, en la tabla 3 se puede ver la planificación real y final que se utilizó.

	Nombre de la tarea		**Duración**	**Inicio**	**Fin**	**Predecesores**
1	**Inicio**		**4h**			
	1.1	Definir el proyecto con el tutor y supervisor	1h	31/01/19	31/01/19	
	1.2	Definir método de trabajo y documentación	2h	04/02/19	04/02/19	1.1
	1.3	Definir formato y estándares de trabajo	1h	04/02/19	04/02/19	1.2
2	**Documentar y planificar el proyecto**		**2h**			

	2.1	Investigación de las entidades del proyecto	1h	04/02/19	04/02/19	1
	2.2	Investigación de las funcionalidades a desarrollar	1h	05/02/19	05/02/19	2.1
3	**Planificación del proyecto**		**35h**			
	3.1	Especificación de las entidades del proyecto	4h	05/02/19	05/02/19	2
	3.2	Especificación de los casos de uso	4h	06/02/19	06/02/19	3.1
	3.3	Revisión de toda la documentación	8h	07/02/19	08/02/19	3.2
	3.4	Diseño conceptual de la base de datos	4h	11/02/19	11/02/19	3.3
	3.5	Prototipo de la aplicación	15h	12/02/19	15/02/19	3.4
4	**Implementación del proyecto (sprints 1-9)**		**251h**			
	4.1	Estudio de JavaScript	25h	18/02/19	22/02/19	
	4.2	Estudio de Vue y Firebase	30h	25/02/19	01/03/19	
	4.3	Implementación de la parte destinada a la administración	150h	04/03/19	25/03/19	3
	4.4	Implementación de la parte destinada al cliente	46h			
5	**Almacenamiento de los datos del proyecto**		**8h**			
	5.1	Almacenamiento del panel de administración	4h	06/05/19	06/05/19	
	5.2	Almacenamiento del panel del cliente	4h	07/05/19	20/05/19	6.1

Tabla 3: Planificación real del proyecto.

De la misma manera, las reuniones comentadas en el apartado de metodología se realizaron gracias a un conjunto de sprints.

3.4.1. Sprint 1

3.4.1.1. Especificación del Sprint 1

Este sprint fue la primera reunión que se llevó a cabo una vez se llegó a la fase de implementación y no duró más de quince minutos. En este caso, el *product owner* se centró en que la formación obtenida para realizar las aplicaciones fuera profunda. Por ello, esta reunión sirvió para conocer si se necesitaba más tiempo de estudio de *JavaScript*.

Dado que si era necesario un mayor conocimiento de *JavaScript*, el *tech leader* decidió alargar el tiempo de estudio.

3.4.1.2. Planificación del sprint siguiente

Al necesitar más tiempo para estudiar la tecnología de *JavaScript*, la siguiente reunión fue una semana después para así comprobar, de nuevo, si se requería más tiempo de estudio, en este caso de *Vue.js* y *Firebase*.

3.4.2. Sprint 2

3.4.2.1. Especificación del Sprint 2

Este sprint sirvió para ver si se necesitaba un mayor conocimiento de *Vue.js* y Firebase. Al tener que llevar a cabo un estudio uniendo ambas tecnologías, más la funcionalidad individual de cada una, el *product owner* insistió de nuevo en seguir profundizando en ello.

De nuevo, el *tech leader* decidió alargar el tiempo de estudio de *Vue.js* y *Firebase*.

3.4.2.2. Planificación del sprint siguiente

En la siguiente reunión se trataría el tema de la implementación inicial del proyecto, comenzando con la parte destinada a la empresa hotelera.

3.4.3. Sprint 3

3.4.3.1. Especificación del Sprint 3

Este sprint sirvió para ver comentar las necesidades que cubriría la parte del administrador del hotel y ver si existía algún matiz que impidiera el comienzo de la implementación de dicha parte. Por ello, el *product owner* comentó de nuevo las entidades con las que se trataría y la relación de las mismas, así como también qué interfaz sería la primera en desarrollarse.

En este caso, el *tech leader* comentó que la primera interfaz a desarrollar debería ser la personalización de la aplicación, ya que era una forma de comenzar a implementar formularios y conocer los métodos y herramientas necesitadas para la lectura y escritura de los campos de la interfaz.

3.4.3.2 Planificación del sprint siguiente

El tiempo de estimación de la siguiente reunión cambió de una semana a dos, para tener más tiempo de implementación y que los cambios fueran significativos. En la siguiente reunión se revisaría los tiempos de implementación, así como si el diseño era correcto.

3.4.4. Sprint 4

3.4.4.1. Especificación del Sprint 4

Este sprint comenzó primero con la cuestión de si el alumno había desarrollado la tarea comentada en el sprint anterior de forma sencilla. El *product owner* visualizó la primera interfaz realizada, así como también parte de una segunda que no se esperaba, lo que conllevó a que se cubriera la estimación inicial.

De la misma forma, el *tech leader* comentó algunas posibles mejoras en algunos componentes ya que, de esta forma, me resultaría más fácil desarrollar las siguientes interfaces.

3.4.4.2. Planificación del sprint siguiente

El tiempo de estimación de la siguiente reunión continuó dos semanas, donde se visualizaría el desarrollo de dos interfaces más.

3.4.5. Sprint 5

3.4.5.1. Especificación del Sprint 5

En este sprint se presentó el desarrollo de dos interfaces más, la segunda con unos pequeños errores. El *product owner* comentó que el proyecto estaba avanzando de forma significativa y que, a pesar de ello, podrían surgir algunos cambios en la funcionalidad, lo que llevaría a un pequeño cambio en la planificación.

De la misma forma, el *tech leader* comentó posibles soluciones a los pequeños errores mencionados.

3.4.5.2. Planificación del sprint siguiente

El tiempo de estimación de la siguiente reunión continuó a dos semanas, donde se visualizará el desarrollo de una interfaz que incluiría componentes más avanzados y donde se modificarían valores de lectura y escritura.

3.4.6. Sprint 6

3.4.6.1. Especificación del Sprint 6

En este sprint se presentó el desarrollo del componente avanzado, en el cual se comentaron algunos campos que quizá se deberían añadir para el almacenamiento de valores. Esto modificaría la forma y el código en el que se encontraban almacenados, afectando al archivo *Store*.

De la misma forma, el *tech leader* comentó posibles alternativas a los pequeños cambios mencionados.

3.4.6.2. Planificación del sprint siguiente

El tiempo de estimación de la siguiente reunión continuó a dos semanas, donde se visualizaría el desarrollo de una interfaz que incluiría componentes de rangos horarios y revisar si esto podría alargar la planificación ya que no era un componente fácil.

3.4.7. Sprint 7

3.4.7.1. Especificación del Sprint 7

En este sprint se revisaron las interfaces desarrolladas hasta el momento. A pesar de que la interfaz en la que se implementaron componentes de calendario presentaron problemas al alumno, el trabajo de desarrolló de forma correcta. De la misma forma, *product owner* comentó que el proyecto cambiaría de planificación a partir del momento y se decidió dejar de lado aquella funcionalidad que estuviera relacionada con Firebase, ya que la empresa tomó en ese momento como prioridad únicamente la implementación de las interfaces.

Por ello, *tech leader* comentó cuáles serían los siguientes pasos a continuar en la planificación, ultimando el panel de administración y, posteriormente, la creación de las interfaces destinadas al cliente.

3.4.7.2. Planificación del sprint siguiente

En este sprint, se planificó la siguiente interfaz a desarrollar y la próxima reunión que tendría lugar dos semanas después.

3.4.8. Sprint 8

3.4.8.1. Especificación del Sprint 8

En este sprint, se revisaron las interfaces relacionadas con las sesiones y el inicio de la parte destinada al cliente.

El *tech leader* decidió que en la próxima reunión se podría únicamente comprobar que la parte del cliente estuviera casi finalizada y el *product owner* comentó que la parte del administrador cumplía con los requisitos definidos.

3.4.8.2. Planificación del sprint siguiente

En el siguiente sprint se revisará la finalización de las interfaces de usuario de la parte destinada a los clientes que tendría lugar dos semanas después.

3.4.9. Sprint 9

3.4.9.1. Especificación del Sprint 9

Este sprint tuvo menor duración pues sólo sirvió para revisar que las interfaces de la parte destinada a los clientes estuvieran realizadas.

Capítulo 4

Análisis y diseño del sistema

4.1. Análisis del sistema

La mejor manera de realizar el análisis del sistema es mediante los diagramas de casos de uso, así como también la documentación de las entidades que formarán parte de la aplicación.

En primer lugar hay que tener en cuenta el diagrama de casos de uso, que se puede ver en la figura 1.

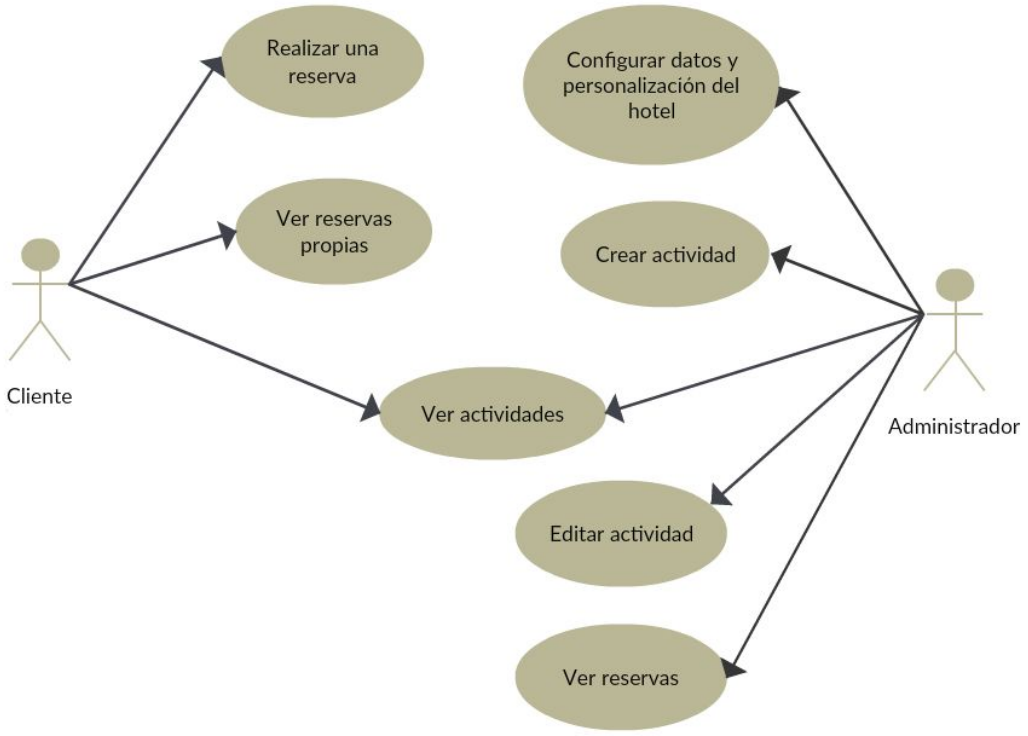

Figura 1: Diagrama de casos de uso: Reserva de actividades y restaurantes.

En segundo lugar, para que la aplicación de reservas se lleve a cabo debemos tener en cuenta las tareas a desarrollar y lo mismo con las subtareas.

Para ello, hay que tener en cuenta las entidades que formarán esta aplicación:

- Cliente: Contendrá la información sobre el cliente.
- Usuario: Todos aquellos usuarios que podrán acceder al panel de administración de actividades.
- Actividad: Todo aquel servicio que puede ser reservado durante un rango temporal.
- Rango horario: Rango de fechas en la que está disponible una actividad.

- Sesión: Sirve para indicar las diferentes horas a las que empieza una actividad.
- Grupo: Agrupa los diferentes datos de un grupo de una sesión, ya que una sesión pueden albergar varios grupos.
- Reserva: Almacena todos los datos relacionados con la reserva, inclusive las entidades Actividad, Sesión, Grupo y Cliente.

La relación entre estas entidades se puede visualizar en la figura 2.

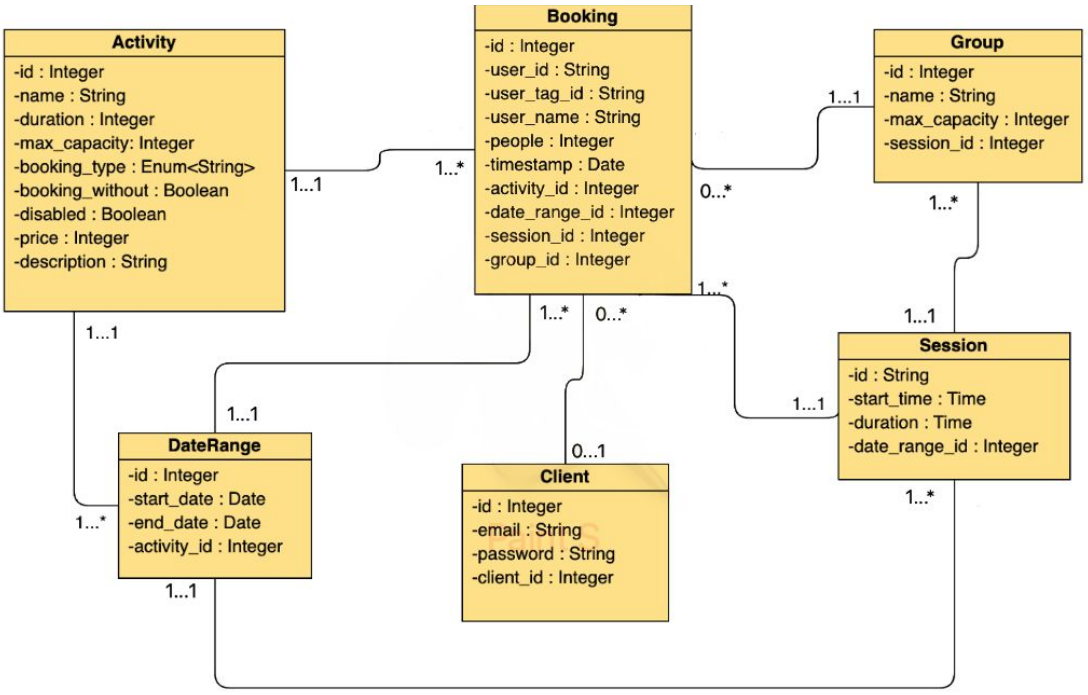

Figura 2: Diagrama de clases de las entidades.

Como se puede ver en la figura 2, en la entidad *Booking*, hay un campo llamado *user_tag_id*. Un tag hace referencia al campo donde se guardará la información de la conexión de la pulsera con la aplicación.

La descripción de los casos de uso se pueden ver desde la tabla 4 a la tabla 10.

Caso de uso: CU01 - Reserva de actividades en Panel del cliente			
Identificador	CU01	**Fecha creación**	04/02/2019
Nombre	Reserva de actividades	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: El proceso de reserva será un conjunto de interfaces que guiarán al usuario durante el proceso de forma que éste pueda introducir los datos necesarios para realizar una reserva de forma sencilla.			

Secuencia de pasos:

1. El sistema muestra las actividades al cliente.
2. El cliente selecciona la actividad que desea reservar.
3. El sistema muestra la información de la actividad.
4. El cliente indica la cantidad de asistentes así como la fecha en la que desea reservar la actividad.
5. El sistema muestra las sesiones de la actividad con la fecha indicada anteriormente.
6. El cliente selecciona una de las sesiones disponibles.
7. El sistema muestra los grupos asociados a la sesión indicando la capacidad de cada uno y la cantidad de aforo que ya hay ocupado.
8. El cliente selecciona uno de los grupos que estén disponibles.
9. El sistema muestra un resumen de la reserva que va a realizar el cliente y le solicita el nombre al que irá la reserva.
10. El cliente indica el nombre al que se asociará la reserva.
11. El sistema le pide al usuario que lea el tag para confirmar la reserva. Si la opción está activa, también permitirá confirmar la reserva sin leer el tag.
12. El cliente confirma la reserva.
13. El sistema indica que la reserva se ha realizado correctamente.
14. El sistema vuelve al listado de actividades.

Excepciones

1. Alguna entidad no existe o no pertenece a la actividad.
2. La fecha seleccionada no corresponde al calendario indicado.
3. La reserva no ha podido realizarse.
4. Alguna entidad no existe o no pertenece a la actividad.

Tabla 4: Caso de uso CU01 - Reserva de actividades, panel del cliente

Caso de uso: CU02 - Ver actividades parte Administrador			
Identificador	CU02	**Fecha creación**	04/02/2019
Nombre	Reserva de actividades	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: En el panel, se creará una interfaz donde el administrador localice cada actividad registrada. En ella podrá visualizar todos sus campos y los valores correspondientes.			
Secuencia de pasos: 1. El administrador accede al apartado de actividades. 2. El administrador solicita el listado de actividades. 3. El sistema muestra el listado de actividades.			
Excepciones: 1. El administrador no dispone de los permisos necesarios para acceder al			

apartado. 2. El actor vuelve al apartado de inicio de sesión.

Tabla 5: Caso de uso CU02 - Ver actividades, panel Administrador.

Caso de uso: CU02 - Ver actividades parte Cliente			
Identificador	CU02	**Fecha creación**	04/02/2019
Nombre	Reserva de actividades	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: La diferencia con respecto al listado que verá el administrador es la información mostrada. Esto se debe a que habrán campos que el usuario no podrá ver, como el código de la actividad.			
Secuencia de pasos: 1. El sistema solicita las actividades del cliente para el que está configurado. 2. El sistema muestra las actividades obtenidas.			
Excepciones: 1. El cliente no existe. 2. El cliente vuelve al apartado de inicio de sesión.			

Tabla 6: Caso de uso: CU02 - Ver actividades, panel Cliente.

Caso de uso: CU03 - Añadir actividad, panel de administración			
Identificador	CU03	**Fecha creación**	04/02/2019
Nombre	Reserva de actividades	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: Rellenará un formulario con la información de la nueva actividad. Si surge algún tipo de error a la hora de crear esta actividad aparecerá un mensaje indicando dicho error.			
Secuencia de pasos: 1. El sistema comprueba que el administrador tiene permiso para crear la actividad. 2. El sistema muestra el formulario de creación de actividad y sus subentidades. 3. El administrador introduce los datos de la actividad y sus subentidades.			

4. El sistema valida los datos introducidos. 5. El administrador indica que desea almacenar los cambios. 6. El sistema informa que los cambios se han almacenado correctamente.
Excepciones: 1. El administrador no tiene acceso al caso de uso. 2. El sistema encuentra un error en la validación de la entidad y sus subentidades. 3. El sistema no ha podido almacenar los datos.

Tabla 7: Caso de uso CU03 - Añadir actividad, panel de administrador.

Caso de uso: CU04 - Editar actividad, panel de administración			
Identificador	CU04	**Fecha creación**	04/02/2019
Nombre	Editar actividad	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: En el panel se creará una interfaz donde el administrador podrá editar una actividad. Podrá modificar la información en el formulario comentado antes.			
Secuencia de pasos: 1. El administrador selecciona la actividad a editar. 2. El sistema obtiene los datos de la actividad y sus subentidades. 3. El sistema muestra el formulario de edición de la actividad con los datos obtenidos. 4. El administrador modifica la actividad y sus subentidades. 5. El administrador solicita guardar los cambios. 6. El sistema valida los datos. 7. El sistema informa que los cambios que se han guardado correctamente.			
Excepciones: 1. La entidad solicitada no existe. 2. El sistema ha encontrado un error durante la validación. 3. El sistema no ha podido almacenar la entidad y sus subentidades.			

Tabla 8: Caso CU04 - Editar actividad, panel de administrador.

Caso de uso: CU05 - Mostrar reservas, panel de administración			
Identificador	CU05	**Fecha creación**	04/02/2019
Nombre	Mostrar reservas	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019

Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: En el panel se creará una interfaz donde el administrador podrá editar una actividad. Podrá modificar la información en el formulario comentado antes.			
Secuencia de pasos: 1. El administrador accede al apartado de reservas. 2. El sistema carga las reservas sin filtros. 3. El administrador puede introducir datos de filtro.			
Excepciones: 1. El administrador no tiene acceso al apartado.			

Tabla 9: Caso de uso: CU05 - Mostrar reservas, panel de administración.

Caso de uso: CU05 - Mostrar reservas, panel de cliente			
Identificador	CU05	**Fecha creación**	04/02/2019
Nombre	Mostrar reservas	**Fecha revisión**	10/02/2019
Autor	Gema Juárez	**Fecha aprobación**	10/02/2019
Origen	Petición por parte del usuario	**Versión**	2.0
Descripción: En el panel, el usuario podrá obtener el listado de sus reservas en la pantalla principal una vez logueado, una vez haber sido autenticado.			
Secuencia de pasos: 1. El cliente solicita acceder a sus reservas. 2. El sistema solicita al administrador que se identifique mediante el tag. 3. El sistema muestra las reservas asociadas al tag.			
Excepciones: 1. Error en la lectura del tag. 2. El cliente no tiene datos asociados de reserva. 3. El sistema ha tenido un error intentando obtener los datos.			

Tabla 10: Caso de uso: CU05 - Mostrar reservas, panel de cliente.

4.2. Diseño de la arquitectura del sistema

Una vez claro el análisis del sistema, se pasa al diseño de su arquitectura. Se trata de una aplicación web que está desarrollada con *Vue.js*. Este framework es progresivo debido a la modularización de las librerías que tiene instaladas. Asimismo, tiene una serie de aspectos a comentar.

En primer lugar, los componentes están diseñados bajo el patrón MVVM, lo cual permite que no haya preocupación en cuanto a la renderización del modelo en pantalla.

En segundo lugar, el sistema de componentes es reactivo lo que posibilita una mejor comunicación debido a los eventos asíncronos. Paralelamente, también contamos con *Vuex*, una librería de *Vue.js* encargada de los patrones de administración de estados. Dentro de esta librería se ha de mencionar que está formada por tres partes:

- Estado: El motivo que se ha de tener en cuenta en la aplicación.
- Vistas: Es el mapeo declarativo del estado.
- Acciones: Posibles formas en las que el estado podría cambiar en función de las entradas.

Para visualizar esto de mejor forma, véase la figura 3.

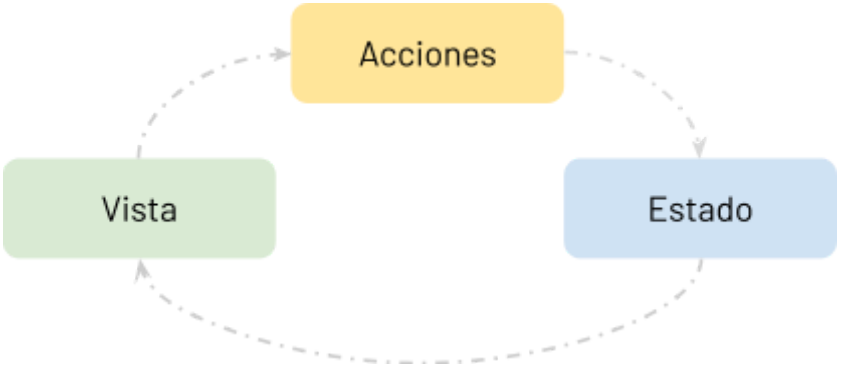

Figura 3: Partes de las que se forma *Vuex*.

Cuando se usan varios componentes es fácil romper la simplicidad. Dado este problema, *Vuex* define y separa los conceptos para administrar el estado y hace que las vistas mantengan una independencia entre ellas y los estados, dando una estructura y capacidad de mantenimiento. *Vuex* ha permitido mantener un esquema fácil en cuanto al valor de los campos de lectura y escritura, permitiendo así una mejora a la hora de controlar el valor de los datos.

Un sistema también a mencionar es *Vue-CLI* que no sólo permite que el proyecto sea interactivo sino que además proporciona una dependencia de tiempo de ejecución actualizable, configurable y extensible a través de plugins. De la misma forma te da la opción de obtener una colección de complementos oficiales y una interfaz gráfica de usuario para poder administrar todos los proyectos creados de *Vue.js*.

Para terminar, otra herramienta utilizada es *Vue-Router*, que como su nombre indica es el router de *Vue.js*. Permite mapear la ruta, configurar el enrutador modular basado en componentes y poder consultar comodines y enlaces con clases activas automáticas de *CSS*. Sin duda alguna, este ha sido el componente que más facilidad ha aportado ya que la gestión de rutas se realiza de forma muy sencilla e intuitiva, incluyendo los componentes que tomarán esa ruta.

4.3. Diseño de la interfaz

La interfaz del usuario realizada para el proyecto estuvo basada en el diseño de prototipos que habían sido desarrollados por un miembro de la empresa (véase Anexo I y Anexo II). Cuando las pantallas se acordaron finalmente, se dispuso a su implementación. Surgieron algunos cambios conforme fue pasando el tiempo ya que algunos componentes cambiaron de funcionalidad, como por ejemplo, el menú de actividades del panel destinado al cliente.

En la figura 4 se puede apreciar las páginas por las que puede navegar el administrador. Los niveles en los que se encuentran definidos cada una de las páginas sirven para identificar el orden de las mismas. Esto es que al nivel 2 sólo se puede acceder desde el nivel 1 y desde una página en concreto. Por ejemplo, sólo se puede ver una actividad si te encuentras en la página de actividades y siempre que hayas accedido a una actividad en concreto. Asimismo, no puedes acceder a esta si te encuentras en reservas o configuración.

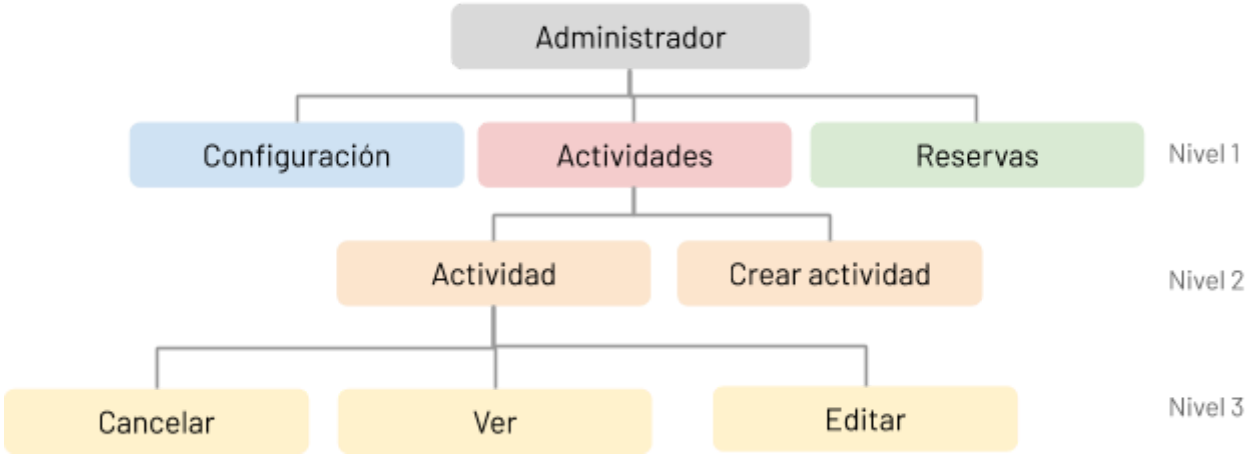

Figura 4: Mapa de navegación de la aplicación destinada al administrador.

En primer lugar nos encontramos con el inicio de la aplicación. Por comodidad visual y una mejor forma de seguir la aplicación, la explicación empieza por la primera opción del menú: Configuración. En esta opción visualizará una ventana donde podrá actualizar la información relacionada con el nombre del hotel, el tipo de moneda y el logotipo. Además, se podrá definir los dos colores a usar en la aplicación y el tono de claridad del texto que hará constraste con estos colores. Esta pestaña finaliza con la vista previa de toda esta información para facilitar la visualización de cómo quedaría la aplicación del cliente. Este componente se puede visualizar en las figuras 17, 18, 19 y 20 del Anexo I.

Si el administrador decide visualizar las actividades que proporcionan, aparecerá una ventana donde podrá ver este conjunto de actividades. Este listado se puede filtrar por una barra de búsqueda de tipo texto, para poder buscar una actividad en base a su nombre. En cada fila de la lista se visualiza el nombre de la actividad, una opción para poder desactivarla y otra opción en la que se podrá visualizarse más detalladamente.

Junto a la barra de búsqueda está la opción de crear una actividad, así como también escanear el código QR del administrador.

Para ver una actividad aparecerá una ventana con dos opciones. En la primera opción se podrá ver la información de la actividad como la descripción, el precio o la capacidad máxima de la misma. Si por el contrario se elige la segunda opción, se verá un calendario en el que se muestran las fechas de la actividad. Estas fechas están formadas por sesiones y cada sesión formada por un conjunto de grupos. Cada grupo tiene un nombre y una cantidad máxima de personas.

En la tercera opción del menú, el administrador podrá observar el listado de reservas realizadas por los clientes. Cada fila está formada por el nombre al que está hecha la reserva, el nombre de la actividad, la fecha y el grupo al que pertenece. Al igual que el listado de actividades, el administrador podrá filtrar la lista por nombre de actividad o por una barra de búsqueda.

La navegación de la parte destinada al cliente se puede ver en la figura 5. A diferencia del mapa de navegación del administrador, cuando una actividad es elegida, las pantallas que se muestren pertenecerán a la misma opción. Por ejemplo, el cliente observa el listado de actividades y elige la actividad que quiere reservar. Es a partir de aquí que se mostrará para esta acción en específico un conjunto de pantallas diferenciadas para no cargar al cliente con tanta información, pero con el mismo fin. Por ello, estas opciones quedan remarcadas con otro tipo de figura, para diferenciar que este nivel es el mismo --ya que en ningún momento va a realizar una opción diferente-- a pesar de que haya un desglose que aclare todo lo que conlleva realizar una reserva.

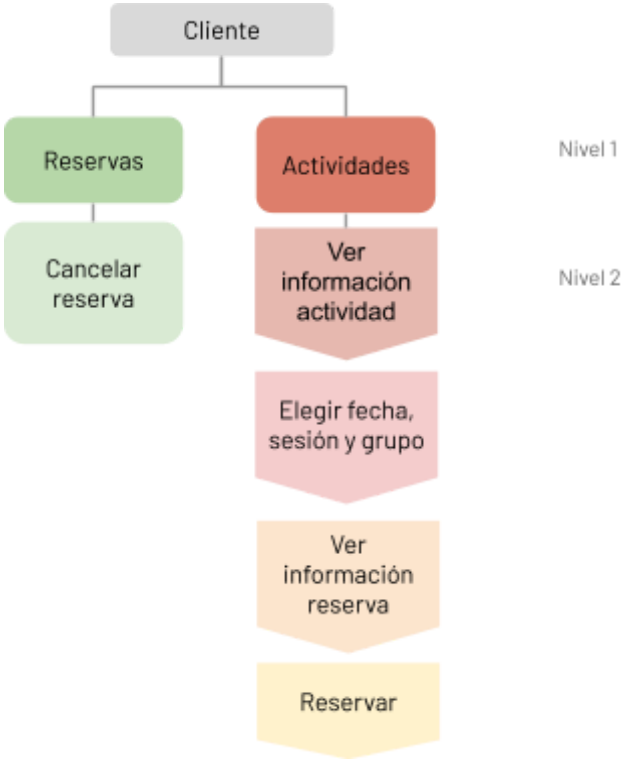

Figura 5: Mapa de navegación de la aplicación destinada al cliente.

El cliente iniciará sesión a través de un dispositivo con forma de pulsera que le permitirá reserva una actividad. Por lo tanto, una vez reconozca la aplicación al usuario, podrá apreciar una opción en la que podrá ver todas las reservas realizadas por él y también todas las actividades disponibles, incluyendo las que ya tiene contratadas con el hotel (debido a cuando contrataron el servicio del hotel, podrían estar incluidas algunas actividades de forma gratuita) Aquí el cliente elegiría la actividad que más le interesase y vería de forma detallada toda la información de la misma.

Si el cliente decide continuar con la reserva, se moverá por un conjunto de pantallas en las que tendrá que escoger qué sesión y grupo desea y el nombre al que irá reservado. Cuando llega a este punto visualizará un resumen de toda la información de la reserva y podrá llevarla a cabo mediante la pulsera o mediante un nombre, dependiendo de si la actividad ofrece la posibilidad o no de reservarla mediante la tecnología RFID. Tras esto, volverá a la pantalla principal.

Si por el contrario el cliente desea visualizar las reservas realizadas por él mismo, podrá hacerlo gracias a la primera opción, justo antes de que comience el listado de actividades. Si esta opción ha sido elegida, este verá un listado de sus reservas donde podrá cancelar aquella que finalmente no le interese tener reservada.

4.4. Diseño de datos

Como bien se ha comentado en los apartados anteriores, la empresa decidió a lo largo de la estancia de prácticas que era prioritario el diseño del frontend de la aplicación y dejar el backend como una mejora que se añadiría después. Sin embargo, es una aplicación que necesita manejar datos, bien sea para mostrar un listado de actividades o realizar una reserva si no, no se podría garantizar que el diseño se ha realizado correctamente.

Es por esto que se llevó a cabo el almacenamiento de los datos en una de las propiedades del navegador llamada *localStorage*. De esta forma se obtendría unos datos que servirían no sólo para comprobar que el diseño se ha realizado correctamente sino que también quedaría orientado para una funcionalidad posterior.

A pesar de todo esto se definieron para el proyecto un conjunto de entidades que servirían no sólo para el almacenamiento en el navegador sino además para cuando se añada la tecnología de Firebase, en una futura funcionalidad. En los siguientes apartados se detallarán cada una de las entidades.

4.4.1. Client

La entidad *client* (tabla 11) contendrá información sobre el cliente (en este caso la empresa hotelera) y servirá para agrupar las diferentes entidades por el mismo.

Nombre	**Tipo**	**Descripción**
id	String	Identificador de la entidad. Autogenerado
name	String	Nombre del cliente

logo	String	Opcional. URL donde está ubicado el logo del hotel
configuration	String	Formato JSON. Contendrá información sobre el color primario y secundario que deberán utilizarse en la parte del administrador

Tabla 11: Entidad Cliente.

4.4.2. User

La entidad *user* (tabla 12) hace referencia a todos aquellos usuarios que podrán acceder al panel de administración de actividades.

Nombre	**Tipo**	**Descripción**
id	Integer	Identificador de la entidad. Autogenerado
email	String	Email del usuario
password	String	Contraseña del usuario
client_id	Integer	Identificador externo. Referencia a client.

Tabla 12: Entidad User.

4.4.3. Activity

Esta entidad (tabla 13), representa a todo servicio que puede ser reservado durante un rango temporal. Está formada por los siguientes atributos:

Nombre	**Tipo**	**Descripción**
id	String	Identificador de la entidad. Autogenerado
name	String	Nombre de la actividad
duration	Integer	Duración por defecto en minutos
max_capacity	Integer	Cantidad por defecto de personas por grupo
booking_type	Enum<String>	Indicará en qué modo deben evaluarse las reservas ya realizadas a la hora de decidir si una reserva puede hacerse o no. Posibles valores: GLOBAL y GROUP

booking_without_tag_allowed	Boolean	Poder reservar una actividad sin utilizar el tag
price	Integer	Precio en céntimos
description	String	Opcional.
disabled	Boolean	La actividad se encuentra desactivada o no

Tabla 13: Entidad Actividad.

El campo *booking_type*, permitirá diferenciar la forma en la que deben tratarse las reservas ya realizadas. Si este campo contiene el valor "GLOBAL", tendrá en cuenta todas aquellas reservas previas que coincidan temporalmente con la reserva que está intentando realizarse. Por otro lado, si el valor que contiene es "GROUP", solo tendrá en cuenta aquellas reservas previas que coincidan temporalmente con la reserva y que, además, sean para el mismo grupo.

4.4.4. DateRange

DateRange (tabla 14) es una entidad que sirve para fijar un rango de fecha en la que está disponible una actividad. Esta entidad es necesaria ya que una misma actividad puede tener diferentes horarios dependiendo de la fecha. Los campos que forman dicha entidad se pueden ver a continuación:

Nombre	**Tipo**	**Descripción**
id	String	Identificador de la entidad
start_date	Date	Fecha de inicio de la actividad
end_date	Date	Fecha de fin de la actividad
activity_id	Integer	Identificador externo. Referencia a *activity*

Tabla 14: Entidad DateRange.

4.4.5. Session

Sessions (tabla 15) es una entidad utilizada para indicar las diferentes horas a las que empieza una actividad dependiendo del *dateRange* al que esté asociada la *session*. A continuación están los campos que componen dicha entidad:

Nombre	Tipo	Descripción
id	String	Identificador único de la entidad. Autogenerado
start_time	Time	Hora de inicio de la sesión
duration	Integer	Opcional. Duración en minutos de la sesión. Si no existe, se utilizará la duración indicada en la *activity*
date_range_id	Integer	Identificador externo. Referencia a *DateRange*

Tabla 15: Entidad Sesión.

4.4.6. Group

Esta entidad (tabla 16) se encarga de agrupar los datos sobre los diferentes *groups* de una *session*, puesto que en una misma *session* pueden llegar a haber varios *groups*. A continuación los campos que componen dicha entidad son:

Nombre	Tipo	Descripción
id	Integer	Identificador de la entidad. Autogenerado
name	String	Nombre del grupo
max_capacity	Integer	Opcional. Cantidad máxima de personas en el grupo
session_id	Integer	Identificador externo. Referencia a *session*

Tabla 16: Entidad Group.

4.4.7. Booking

Por otra parte, tenemos la entidad *booking* (tabla 17) la cual sirve para poder reunir todos los datos necesarios para poder identificar de forma única una reserva así como los datos de interés para la misma. Esta entidad está formada por los siguientes atributos:

Nombre	Tipo	Descripción
ID	String	Identificador de la entidad. Autogenerado
user_id	String	Opcional. Identificador del usuario que reserva la actividad

user_tag_id	String	Opcional. Tag con el que se ha realizado la reserva
user_name	String	Nombre del usuario que reserva la actividad
people	Integer	Cantidad de personas para las cuales se realiza la reserva
timestamp	Date	Fecha y hora del inicio de la reserva
activity_id	String	Identificador externo. Referencia a *Activity*
date_range_id	String	Identificador externo. Referencia a *DateRange*
session_id	String	Identificador externo. Referencia a *Session*
group_id	String	Identificador externo. Referencia a *Group*

Tabla 17: Entidad Booking.

Los campos de *user_id* y *user_tag_id* son opcionales debido a que la reserva puedes realizarla o bien con tu identificador de usuario o bien con la pulsera.

Capítulo 5

Implementación

5.1. Detalles de la implementación

Vue.js, como se ha comentado en el capítulo 4 y mencionado ligeramente en el capítulo 2, es un framework de *JavaScript* que va ganando terreno en el mundo del desarrollo frontend. Es progresivo dado que está modularizado en diferentes librerías con la peculiaridad de que esto facilita la incorporación de la funcionalidad conforme se va necesitando.

Vue.js proporciona una mejor organización ya que en un mismo componente puedes tener la estructura *HTML* que le corresponde, la funcionalidad de *JavaScript* para mejorar el comportamiento de la misma y el estilo que contendrá dicha estructura. Además, esto permitirá que la aplicación y la renderización se cargue más rápidamente.

Paralelamente se ha utilizado una serie de librerías que nos ha permitido mejorar la aplicación:

- *Vuex:* Para gestionar rutas, lo que simplifica la manera en la que se comunican los componentes.
- *Vuetify:* Para el diseño de ciertos componentes. El listado de actividades o el listado de reservas se ha implementado de forma más rápida gracias a un componente (*v-data-iterator*). De la misma forma, las barras de búsqueda, los selectores, los calendarios y cualquier elemento de la página se ha diseñado con este plugin.
- *Vue-Router:* Enrutado de los enlaces de los componentes. La ventaja frente a otras formas de enrutar, es que es mucho más fácil asignar parámetros de ruta o comodines. De la misma forma, cada ruta va configurada a un componente implementado y resulta más fácil de comprender qué ruta conforma cada componente.

5.2. Estructura de la aplicación

La aplicación ha sido desarrollada en *Vue.js,* lo que hace que siga una estructura determinada. Dicha estructura se puede visualizar en la figura 6 y es la misma en el proyecto dedicado al administrador y también la parte que va destinada al cliente. Lo que cambia entre estas partes, son la cantidad de componentes utilizados. Por ejemplo, el panel de administración está formado por un conjunto mayor de componentes con respecto al panel destinado al cliente.

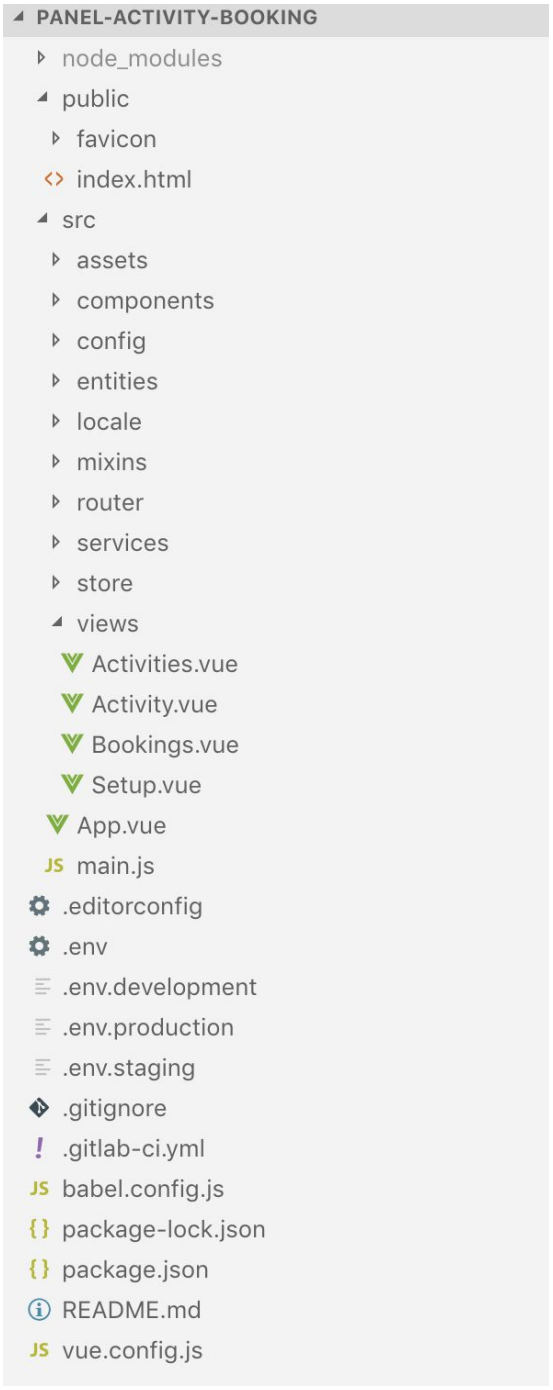

Figura 6: Estructura proyecto *Vue.js*.

Ahora bien, ¿qué aporta cada carpeta al proyecto?

En primer lugar, está el directorio *node_modules*. Esta carpeta guarda las herramientas de compilación. Además, es el lugar donde se almacenarán las librerías que se vayan a instalar.

En segundo lugar, está la carpeta *public* donde se encuentra el archivo que se utiliza para cargar el proyecto. Este archivo se llama *index.html* y es el archivo que carga la información que se encuentre en *App.vue*.

A continuación, en el directorio *src* es donde estará todo el código que conforma la aplicación. Las carpetas más destacadas de este directorio son las siguientes:

- **Components:** Se encuentran los componentes que forman la estructura de la aplicación y dan forma a los componentes principales. Esto es, por ejemplo, la estructura que tendrá cada fila que contenga una actividad en el listado de actividades. *ActivityRow.vue* dará forma a esa fila, pero es *Activities.vue* (almacenada en el directorio *views*) el archivo principal que mostrará la lista de dichas actividades.

- **Entities:** Hace referencia a la definición de cada una de las entidades que se llevarán a cabo, teniendo en cuenta cada uno de sus campos y cómo están definidos estos. Un ejemplo de desarrollo de una entidad se muestra en la tabla 18.

```
export class Setup {
 constructor (setup = {}) {
   this.name = setup.name ? setup.name : null
   this.currency = setup.currency ? setup.currency : null
   this.logo = setup.logo ? setup.logo : null
   this.primaryColor = setup.primary_color ? setup.primary_color : '#aa0000'
   this.secondaryColor = setup.secondary_color ? setup.secondary_color : '#00aa00'
   this.textOverPrimary = setup.text_over_primary ? setup.text_over_primary : 'dark'
   this.textOverSecondary = setup.text_over_secondary ? setup.text_over_secondary : 'dark'
 }
 serialize () {
   return {
     name: this.name ? this.name : null,
     currency: this.currency ? this.currency : null,
     primary_color: this.primaryColor ? this.primaryColor : null,
     secondary_color: this.secondaryColor ? this.secondaryColor : null,
     text_over_primary: this.textOverPrimary ? this.textOverPrimary : null,
     text_over_secondary: this.textOverSecondary ? this.textOverSecondary : null
   }
 }
}
```

Tabla 18: Ejemplo de entidad.

- **Locale:** En este directorio están los archivos relacionados con la traducción de aquella información que permanezca fija en la aplicación. Por ejemplo, en un formulario para introducir una actividad, estará el campo "Nombre de la actividad" donde el administrador rellenará el nombre de la misma. Este enunciado cambiará de idioma entre inglés y español dependiendo del idioma definido en el navegador.

- **Router**: Este directorio es importante ya que es donde se encuentra el archivo de configuración de la librería *Vue-Router*. Aquí es donde se podrá configurar la ruta de navegación de los componentes (véase tabla 19).

```
import Vue from 'vue'
import Router from 'vue-router'
import routeNames from './route-names'

Vue.use(Router)
const router = new Router({
 mode: 'history',
 base: process.env.BASE_URL,
 routes: [
   {
     path: '/setup',
     name: routeNames.setup,
     component: () => import('../views/Setup.vue')
   },
   {
     path: '/activities',
     name: routeNames.activities,
     component: () => import('../views/Activities.vue')
   },
   {
     path: '/bookings',
     name: routeNames.booking,
     component: () => import('../views/Bookings.vue')
   },
   {
     path: '/activities/:id',
     name: routeNames.activity,
     component: () => import('../views/Activity.vue')
   },
   {
     path: '*',
     redirect: { name: routeNames.activities }
   }
 ]
})
export default router
```

Tabla 19: Rutas de navegación de la parte del administrador.

Por ejemplo, en la ruta *"/setup"* cargará el componente importado llamado *Setup.vue*.

- **Store:** Este componente también es importante debido a que sin él no se podría almacenar la información. Es el encargado de mostrar los archivos y también de modificarlos y/o añadirlos.

  En la tabla 20, el método *getSetup()* carga la información en pantalla mientras que el método *saveSetup()* la almacena. Por consiguiente, en la tabla 21, cada uno de los métodos muestra o almacena la información relacionada con ese atributo de la entidad *Setup*.

```
const actions = {
 getSetup ({ commit }) {
   return new Promise((resolve, reject) => {
     requestService.getSetup().then(setup => {
       commit('setSetup', new Setup(setup))
       resolve(setup)
     }).catch(error => {
       reject(error)
     })
   })
 },
 saveSetup ({ state }) {
   return new Promise((resolve, reject) => {
     requestService.setSetup(state.setup).then(() => {
       resolve()
     }).catch(error => {
       reject(error)
     })
   })
 }
}
```

Tabla 20: Métodos de lectura y almacenamiento de una entidad.

```
const mutations = {
 setSetup (state, setup) {
   state.setup = setup
 },
 setName (state, name) {
   state.setup.name = name
 },
 setCurrency (state, currency) {
   state.setup.currency = currency
 },
```

```
 setPrimaryColor (state, color) {
   state.setup.primaryColor = color
 },
 setSecondaryColor (state, color) {
   state.setup.secondaryColor = color
 },
 setTextOverPrimary (state, text) {
   state.setup.textOverPrimary = text
 },
 setTextOverSecondary (state, text) {
   state.setup.textOverSecondary = text
 }
}
```

Tabla 21: Campos a leer o modificar de la entidad.

- **Views:** En este directorio se encuentran los componentes principales de la aplicación, es decir, aquellos que se mostrarán sí o sí por pantalla, formados por los archivos mencionados anteriormente en el directorio *components*.

Del resto de archivos que conforman la estructura de la figura 6, cabe mencionar *App.vue* (tabla 22) donde se cargan todos los componentes gracias a *Navbar.vue* y la etiqueta *<router-view/>*, que es donde se mostrarán esos componentes en función de la opción elegida en el *Navbar.vue*, y el archivo de configuración *main.js*, donde se importan todos los plugins utilizados en la aplicación.

```
<template>
 <div id="app">
   <loading-screen v-if="!state.firstLoad"/>
   <v-app v-else>
     <v-progress-linear v-if="state.loading" indeterminate class="ma-0"
height="4" style="position: absolute; z-index: 10;" />
     <navbar />
     <v-content>
       <v-fade-transition mode="out-in">
         <router-view/>
       </v-fade-transition>
     </v-content>
     <v-snackbar
       v-model="state.message.show"
       bottom
       :color="state.message.color"
     >
       {{ state.message.text }}
```

```
      </v-snackbar>
    </v-app>
  </div>
</template>

<script>
import { mapState } from 'vuex'

export default {
  data () {
    return {
      title: process.env.VUE_APP_TITLE,
      openConfirmDialog: false
    }
  },
  components: {
    LoadingScreen: () => import('@/components/LoadingScreen.vue'),
    Navbar: () => import('@/components/app-components/Navbar.vue')
  },
  computed: {
    ...mapState(['state'])
  }
}
</script>
```

Tabla 22: Archivo App.vue

Para que esta estructura comentada quede más clara, se encuentra la figura 7. La aplicación comienza cargando el archivo *App.vue* el cual llama a *Navbar.vue*. La etiqueta *<router-view/>* mencionada anteriormente, cargará la información de *Setup.vue*, *Activities.vue* y *Bookings.vue* dependiendo de la opción elegida en el navbar.

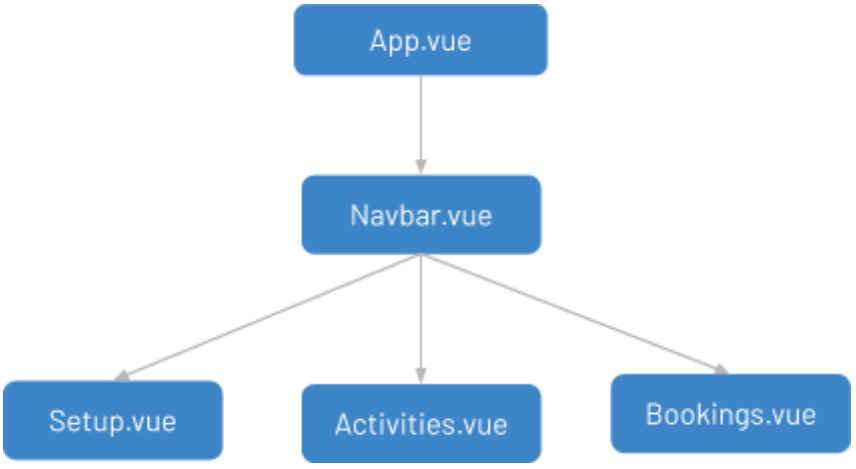

Figura 7: Navegación desde el navbar.

Si en la opción elegida es el archivo de personalización *Setup.vue* (que se encuentra en el directorio *views*) llamará a *SetupRow.vue* (que se encuentra en el directorio *components*), y le dará forma a cada una de las filas del primer panel (véase figura 8).

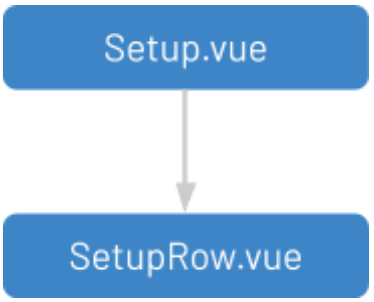

Figura 8: Estructura de la página de personalización.

Si por el contrario la opción elegida es el listado de actividades *Activities.vue* llamará a *ActivityRow.vue* para dar forma a cada una de las filas de la lista. Aquí cambia con respecto a *Setup.vue* la interacción con el componente *Activity.vue* que tiene relevancia debido a que al elegir una actividad determinada del listado, se cargará este último archivo. De nuevo, empieza la estructura correspondiente de *Activity.vue,* el cual está formado por *ActivityItem.vue* o *CalendarItem.vue* dependiendo de la opción escogida. Lo mismo ocurre cuando se llama a *SessionItem.vue* y *GroupItem.vue*. Esta estructura de puede visualizar en la figura 9.

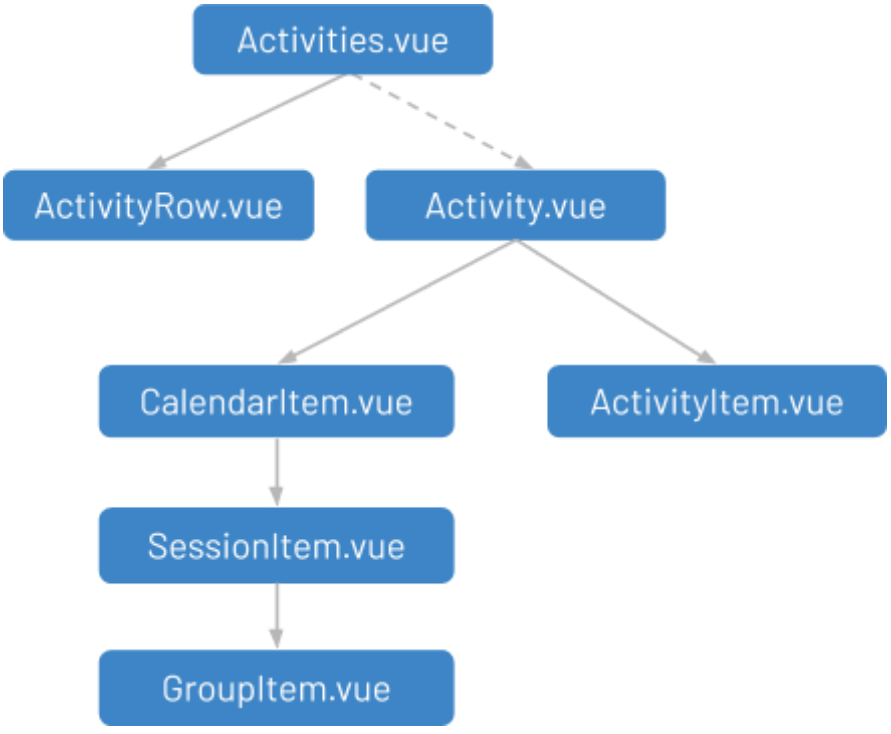

Figura 9: Estructura de una actividad**.**

Si la opción elegida es *Booking.vue*, cargará su estructura debido a *BookingRow.vue* (véase figura 10).

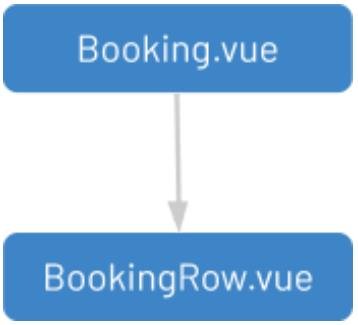

Figura 10: Estructura de la parte del administrador.

La estructura de la parte cliente es similar, ya que también tiene los mismos archivos de configuración, componentes y entidades. La diferencia está en cómo se muestra dichos componentes. Esto se puede ver en la figura 11.

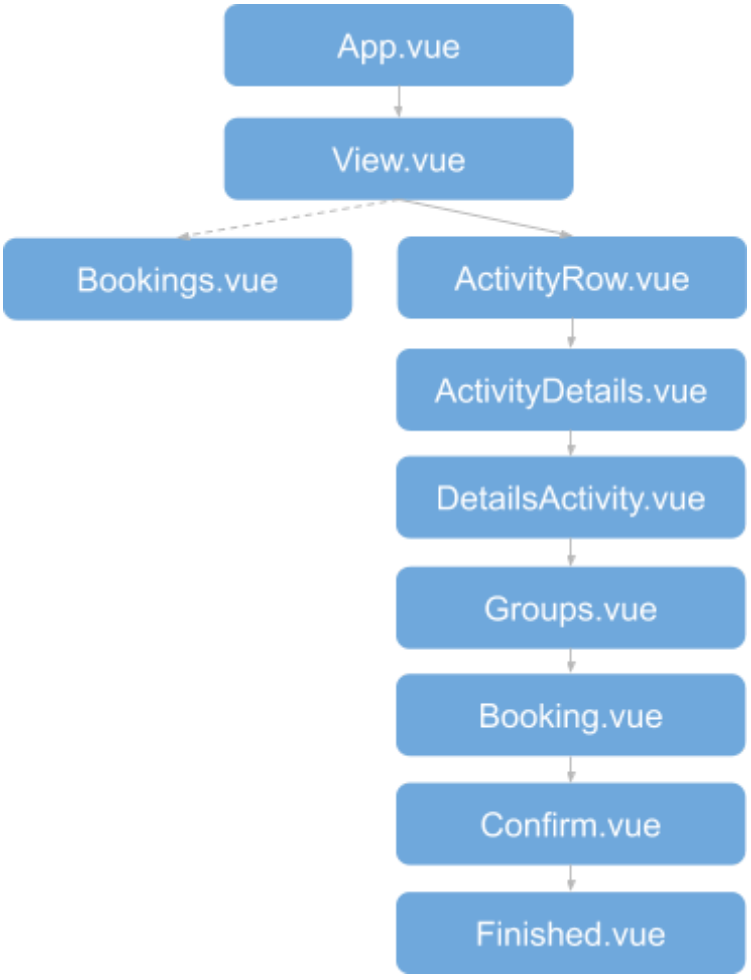

Figura 11: Estructura de la parte del cliente.

5.3. Resultados de la aplicación

5.3.1. Panel del administrador

5.3.1.1. Personalización

La figura 12 muestra la ventana donde el administrador podrá personalizar la parte que verá el cliente, eligiendo cómo ver la información.

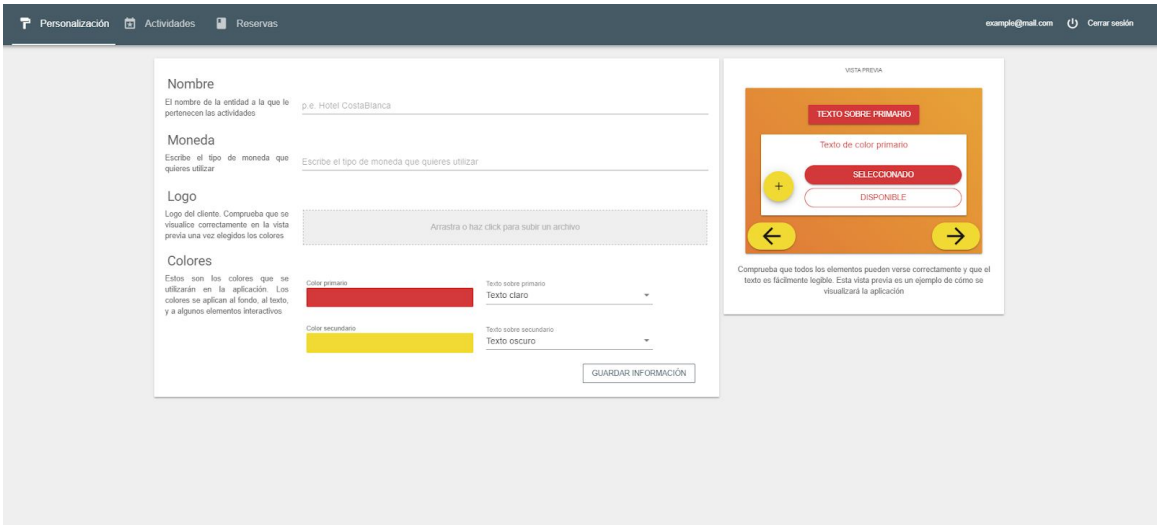

Figura 12: Pantalla de personalización del panel del administrador.

Tal y como se puede ver en la figura 13, el administrador rellenará una serie de campos, como en nombre del hotel, el tipo de moneda que utilizarán, el logotipo de la empresa hotelera, los colores y el contraste del texto sobre estos, que tendrá la parte cliente.

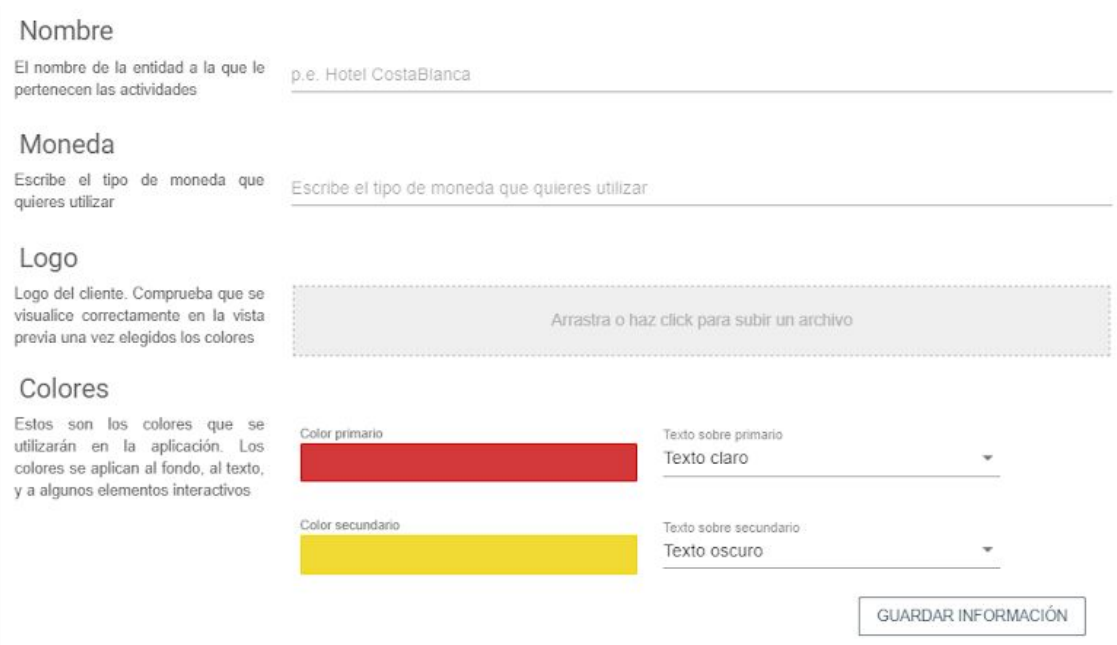

Figura 13: Información a rellenar por el administrador para personalizar la parte del cliente.

Además, tendrá un pequeño panel donde podrá apreciar la vista previa de la configuración que ha llevado a cabo, tal y como se puede visualizar en la figura 14.

Figura 14: Vista previa de la parte cliente con las configuraciones por defecto.

En la figura 15 se puede encontrar una posible configuración real realizada por el trabajador encargado de esta aplicación.

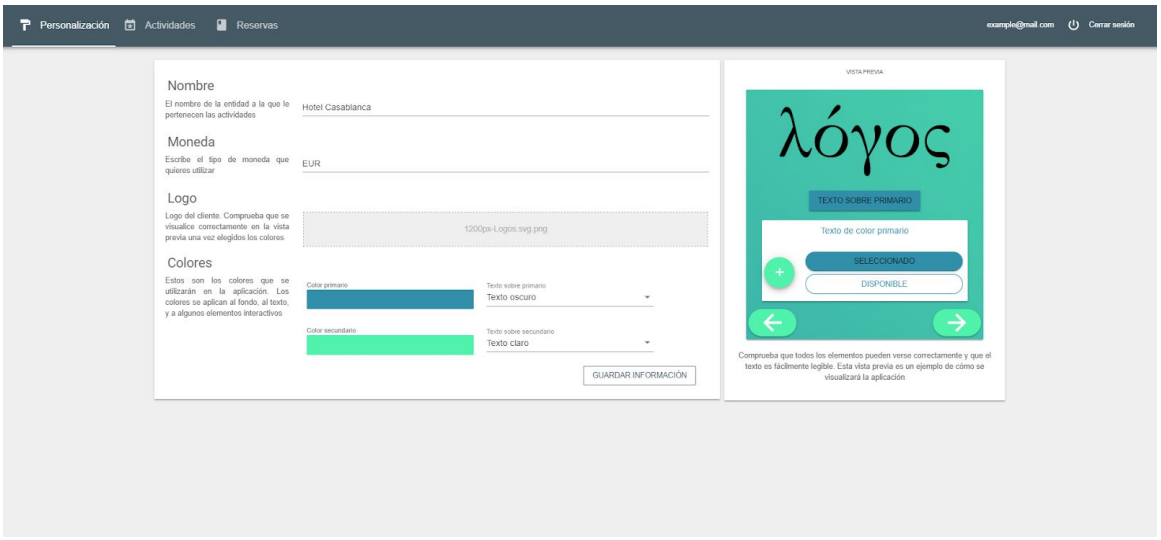

Figura 15: Personalización de la parte cliente.

5.3.1.2. Gestión de actividades

La segunda opción es *Actividades* y como su nombre indica, nos llevará a la ventana del conjunto de actividades que habrá añadido la empresa hotelera. Esto se puede ver en la figura 16.

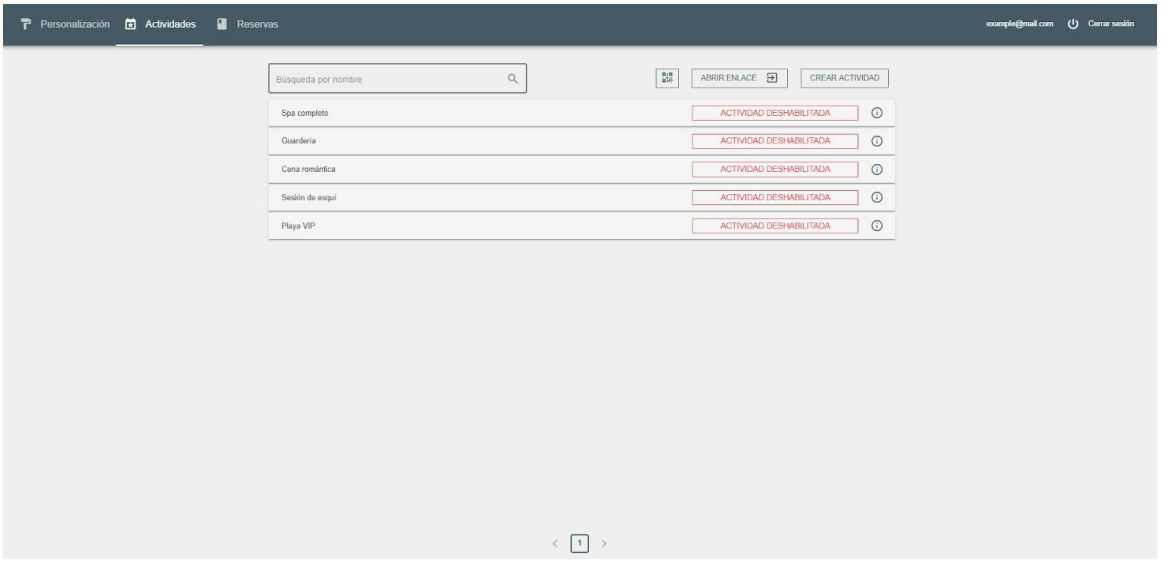

Figura 16: Listado de actividades del hotel.

Tal y como se puede ver en la figura 17, aparece cada una de las actividades añadidas por el hotel. Sobre este listado hay un conjunto de herramientas, como la barra de búsqueda. En esta herramienta se podrá buscar la actividad por su nombre. En la figura 18 se vería la funcionalidad en caso de la búsqueda encuentre algún elemento que coincida con los valores introducidos. En caso contrario, se mostraría un mensaje diciendo que no existe ningún elemento con esos valores (véase figura 19).

Figura 17: Listado de actividades.

Las opciones restantes de la figura 17 se dividen en el lector del QR para redirigir a la parte del cliente, en el enlace para redirigirse a dicha parte y la opción de crear una actividad.

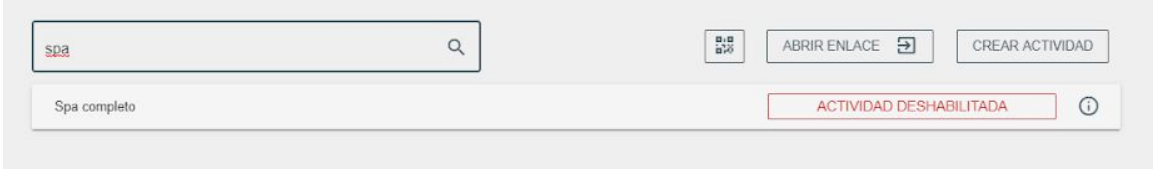

Figura 18: Búsqueda de un elemento.

Figura 19: Búsqueda de un elemento que no ha sido encontrado.

De la misma forma, el administrador podrá deshabilitar una actividad o ver la información de esta. Para ello, tiene las opciones que aparecen tras el nombre de la actividad. Una vez elegida la opción de ver una actividad, podrá visualizarla y modificar cada uno de los campos que contiene. Esto puede verse en la figura 20.

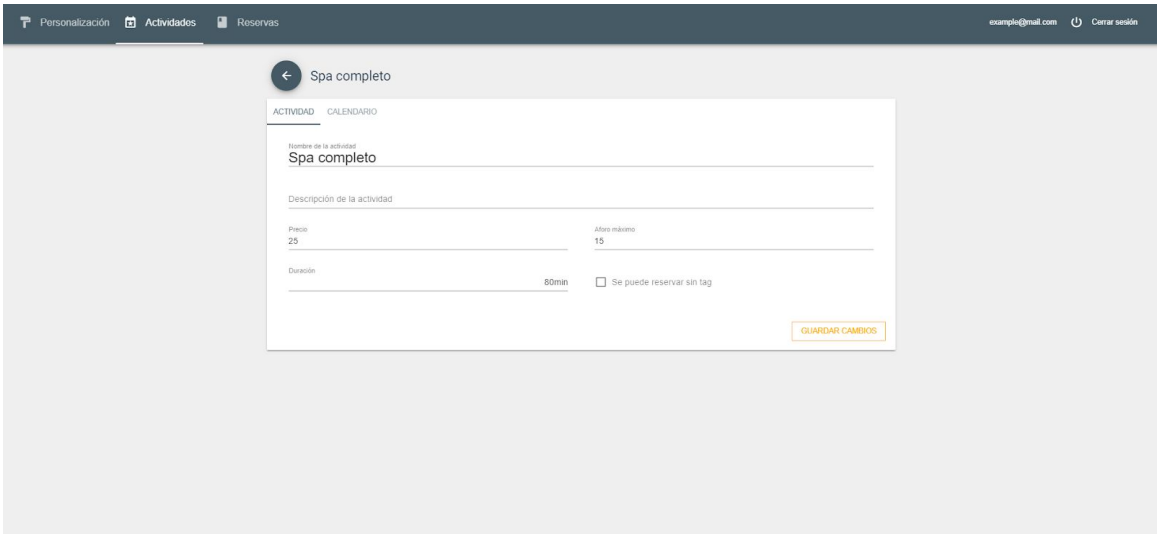

Figura 20: Ver una actividad.

Si nos centramos en la figura 21, la tarjeta que muestra la actividad, se puede ver que existen dos opciones. La primera de ellas hace referencia a la información de la actividad y a lo que se mencionó anteriormente acerca de que el administrador podrá modificar cada uno de los campos. Estos campos si han sido modificados sólo se podrán almacenar si se le da a la opción de “Guardar cambios”.

Si por el contrario el administrador prefiere visualizar o actualizar algún elemento que corresponda a la fecha de la actividad o las sesiones y grupos disponibles para la misma, escogerá la opción “Calendario”. Esta pestaña mostrará si hay un rango de fechas disponibles para la actividad y la posibilidad de crear un nuevo calendario. Cada calendario añadido tendrá la opción de ser editado o de visualizar las sesiones disponibles, que llevaría al administrador hasta la ventana que puede verse en la figura 22. Esta información se almacenará con la opción “Guardar cambios”, que tiene la misma funcionalidad que la opción de guardar de la pestaña “Actividad”.

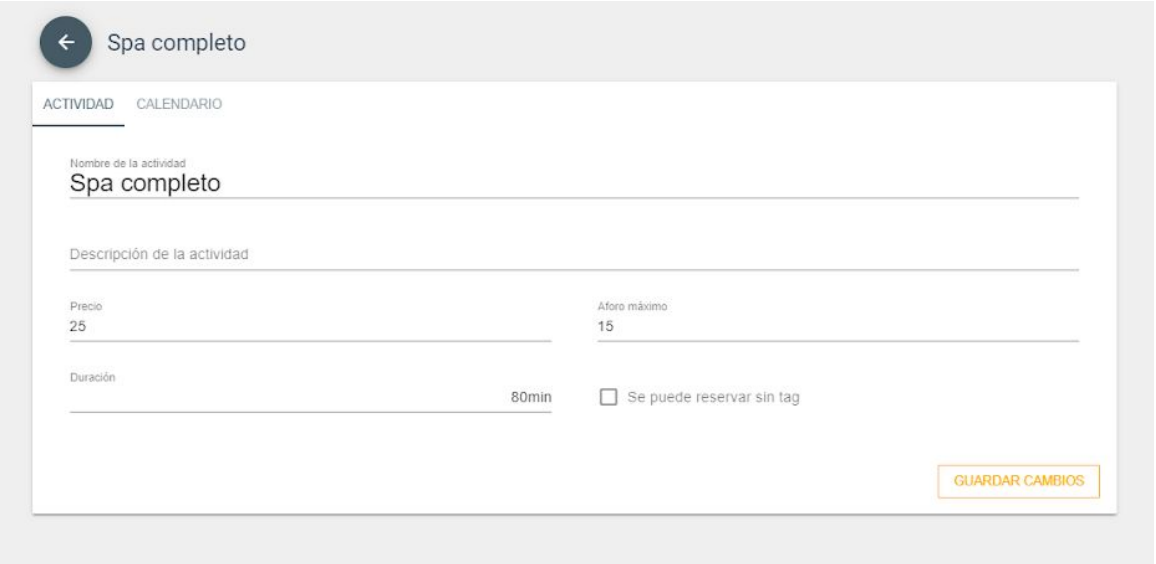

Figura 21: Información de una actividad.

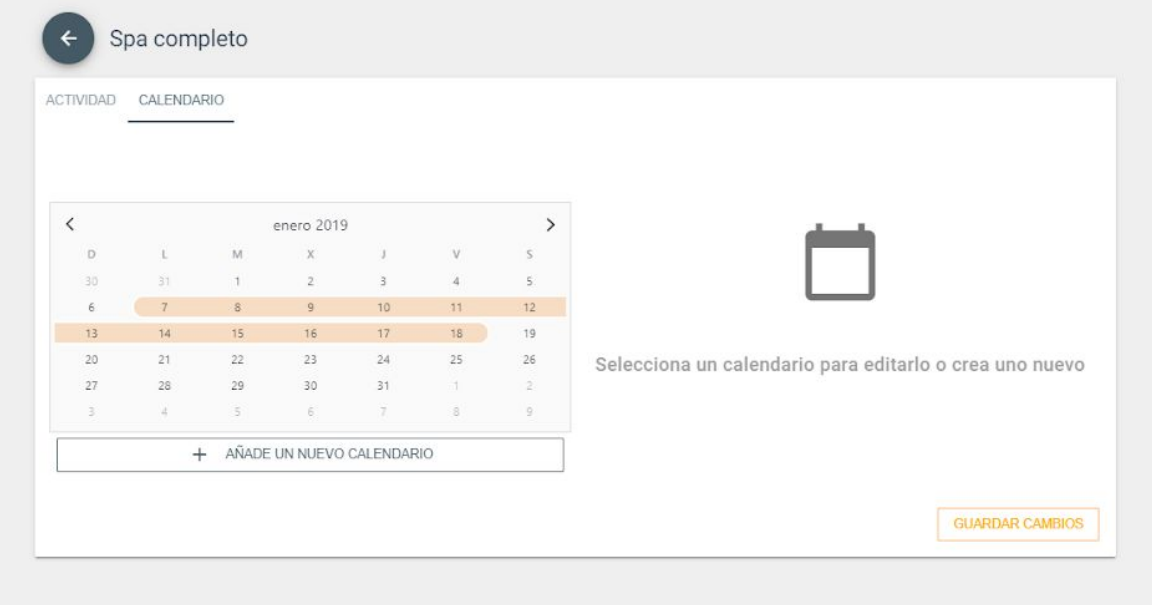

Figura 22: Información de la fecha en la que se encuentra una actividad disponible.

Si el administrador desea visualizar las sesiones pertenecientes a una actividad, se encontrará con la ventana que se muestra en la figura 23.

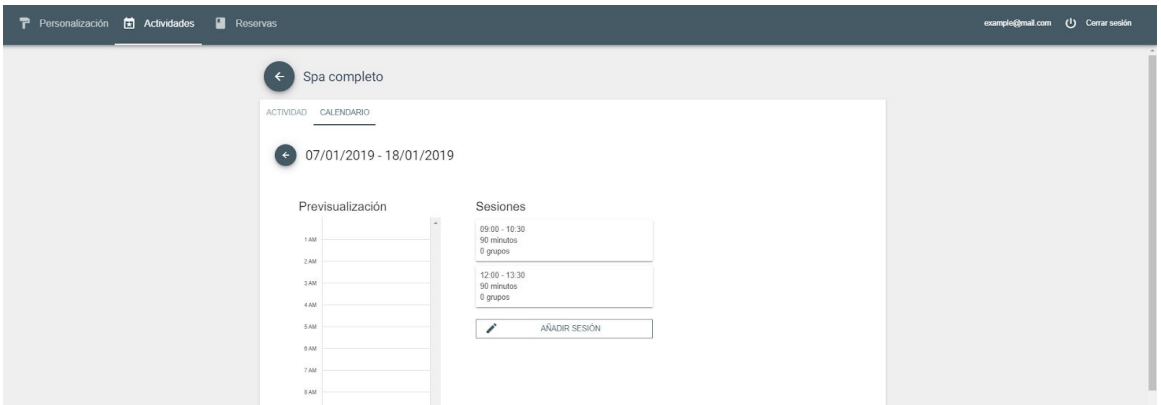

Figura 23: Visualizar sesiones de una fecha determinada.

Dado que es una ventana que se expande a lo largo, para poder visualizar mejor sus funciones, se aprecia la figura 24. En esta ventana, por una parte, está el panel de previsualización cuya función es mostrar las sesiones que se encuentran disponibles a una hora determinada, lo que facilita la visualización del administrador en ver las actividades ordenadas por hora.

Por otra parte, están las sesiones, ordenadas por orden de inserción, con el horario de cada actividad, la duración y los grupos pertenecientes en cada sesión.

Si lo que el administrador desea es editar una de las sesiones que se encuentran disponibles, escogerá la opción y aparecerá un panel a la derecha con la información a modificar. Esto se puede ver en la figura 25.

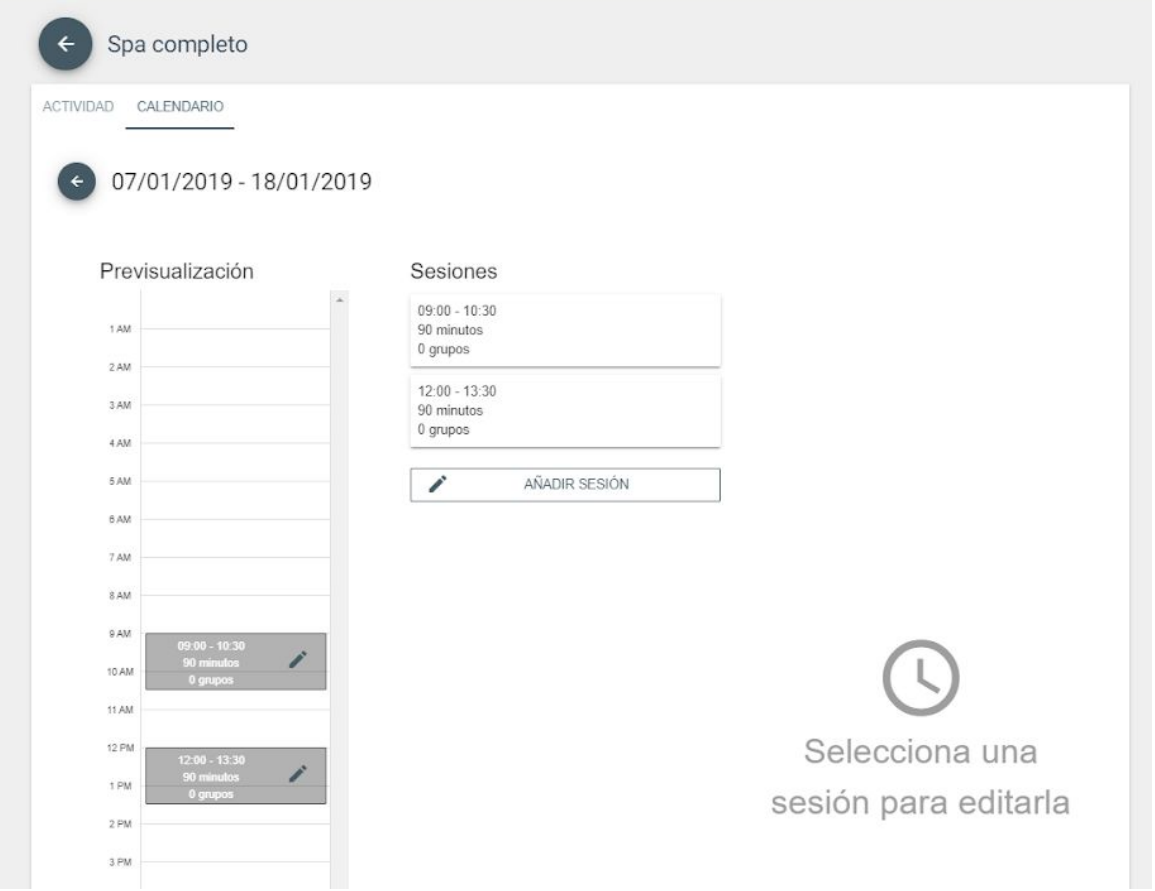

Figura 24: Sesiones disponibles para una fecha determinada.

Cuando el administrador llegue a este punto podrá modificar la hora a la que empieza una actividad. Para ello aparecerá un panel emergente con un reloj en el que podrá poner si es horario de mañana o tarde, la hora y los minutos. En la figura 25, el administrador cambia la hora de la actividad a las 15:10. En la figura 26, se ve el cambio realizado.

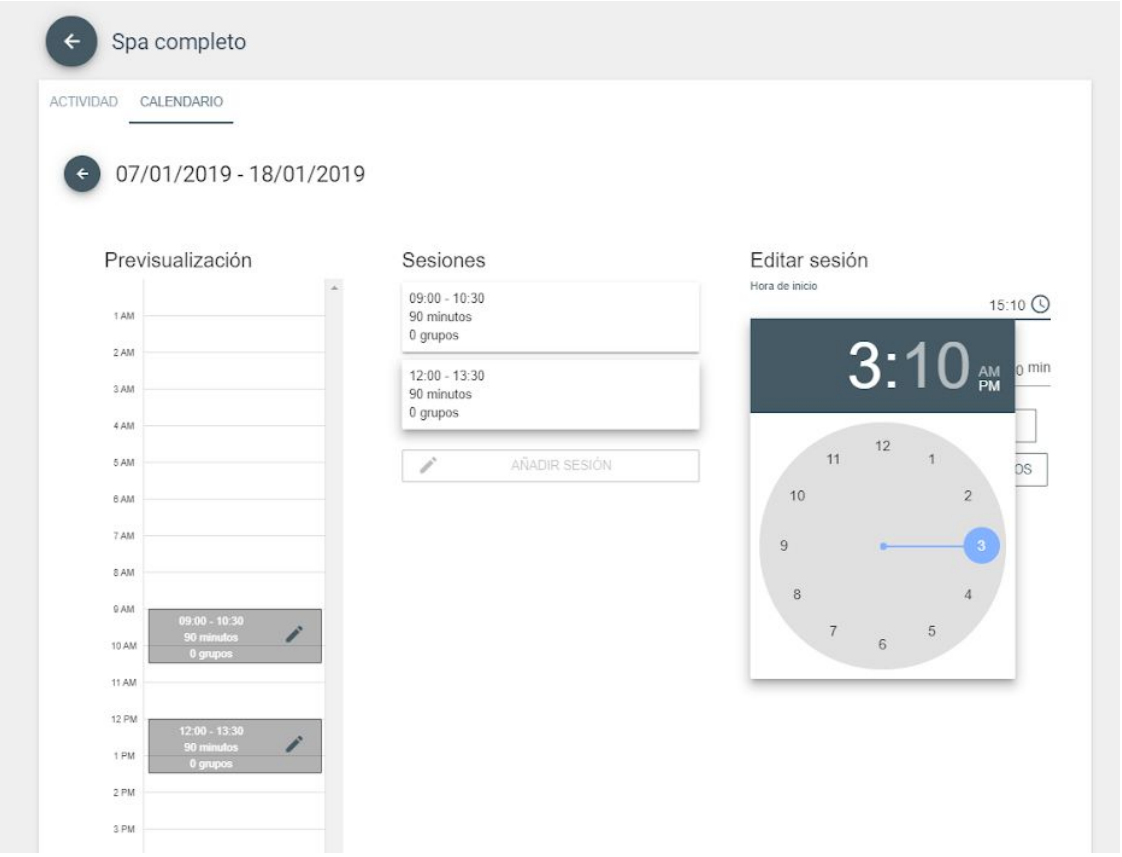

Figura 25: Modificación de la hora de inicio de la sesión.

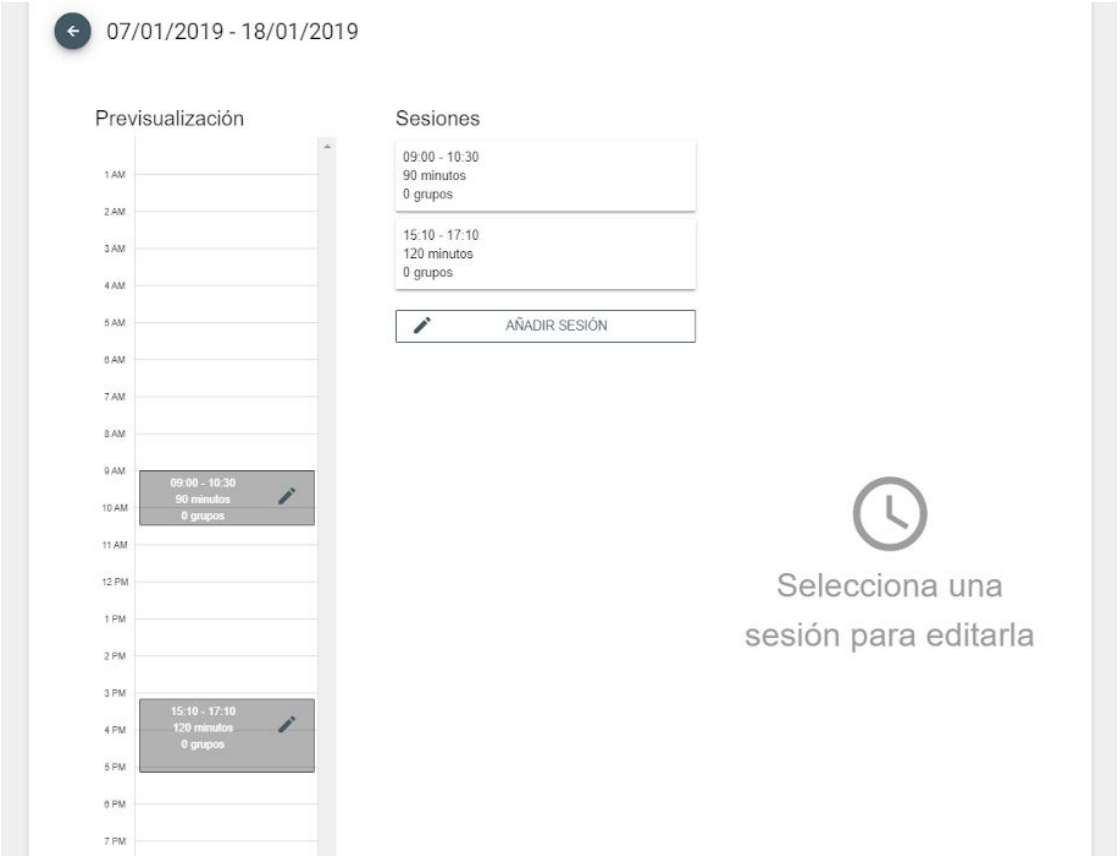

Figura 26: Sesión modificada.

Si el administrador desea crear un grupo, aparecerá un panel en el que añadirá el nombre del grupo y el aforo máximo que acepta (véase figura 27).

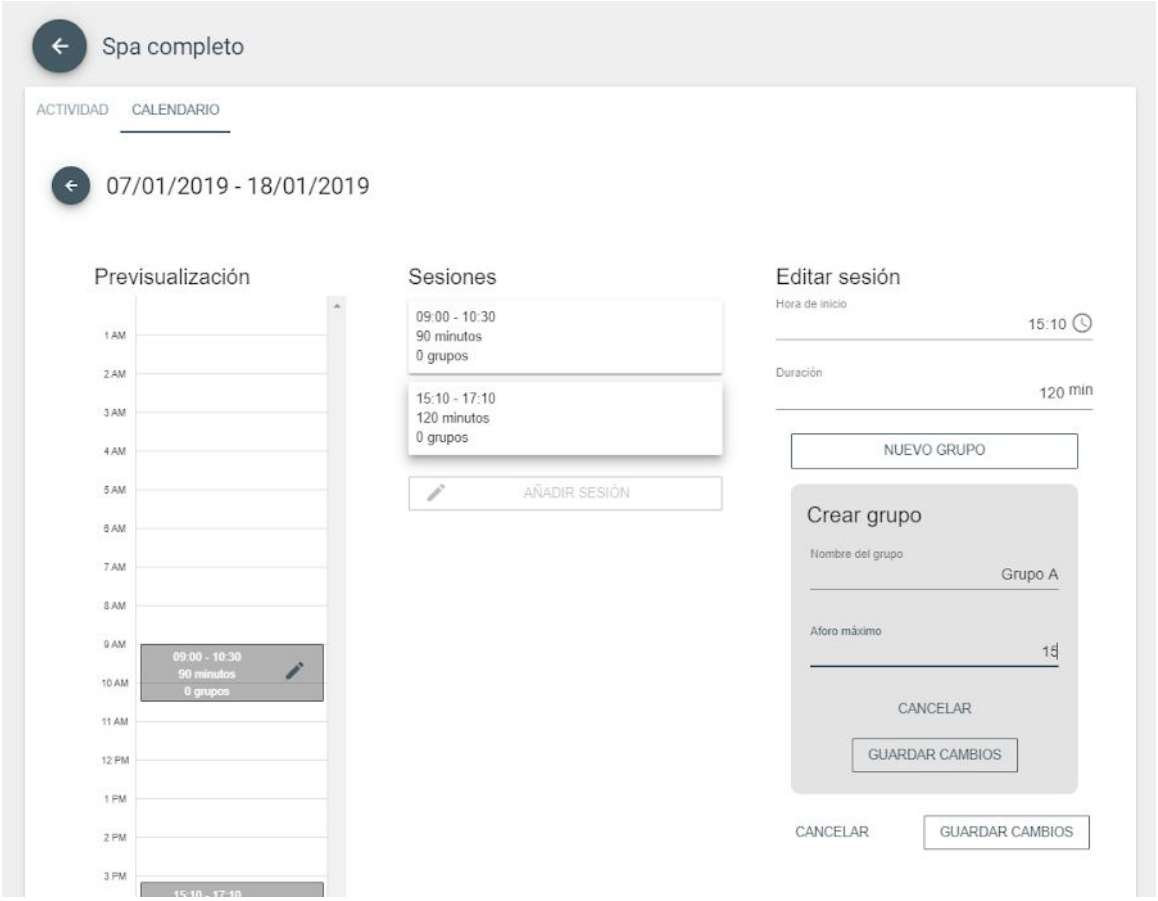

Figura 27: Crear un grupo en una sesión.

Finalmente, cuando un grupo se cree en una sesión, la ventana quedará como en la figura 28.

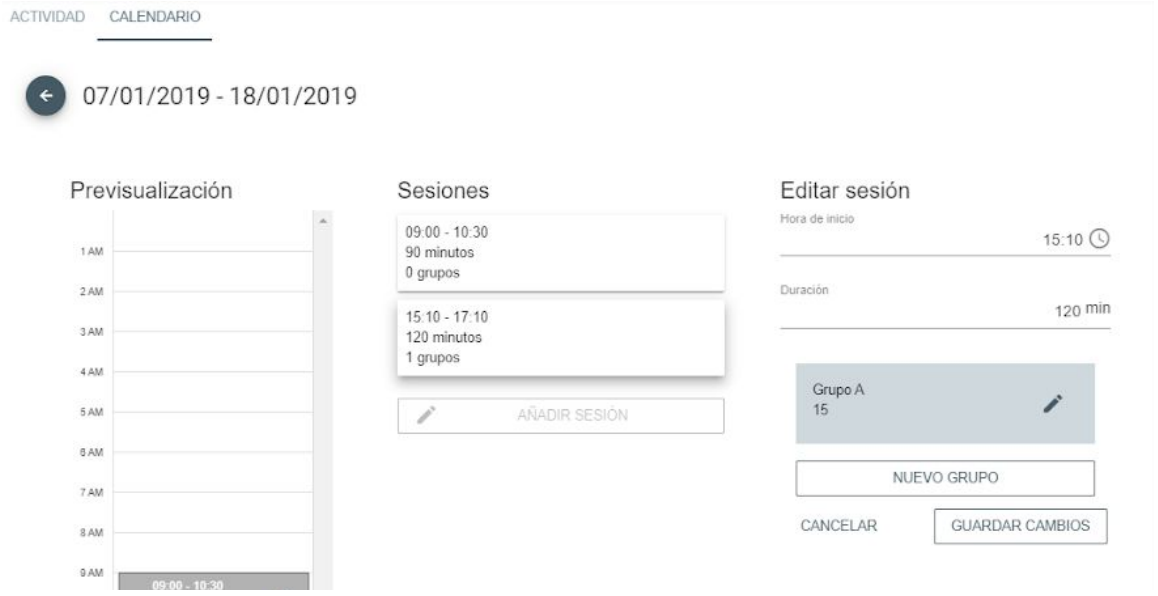

Figura 28: Grupo creado para una sesión.

5.3.1.3. Gestión de reservas

La tercera y última opción del panel de administrador es la opción de "Reservas". Aquí el administrador del hotel podrá visualizar el listado de clientes que han realizado una reserva (véase figura 29).

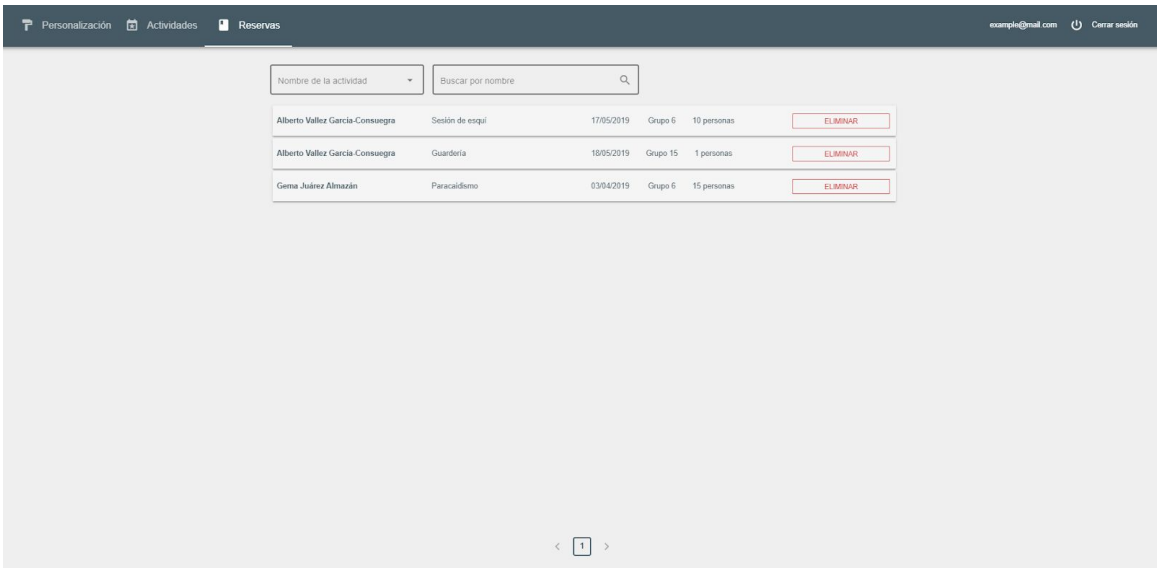

Figura 29: Gestión de reservas.

En cada reserva se muestra el nombre del cliente, la actividad, la fecha, el grupo y la cantidad de personas que formarán la reserva. La reserva se podrá cancelar con la opción "eliminar" (véase figura 30).

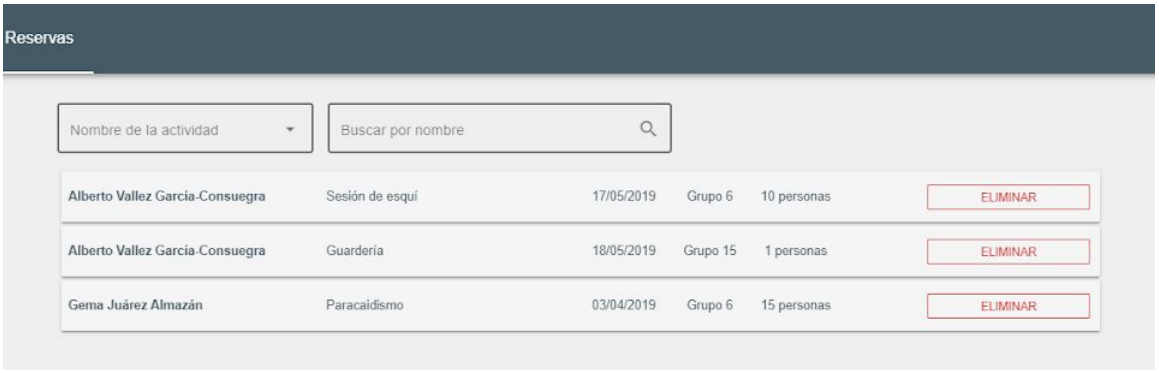

Figura 30: Listado de reservas.

Existe un selector de actividades y una barra de búsqueda para filtrar las reservas. Si existe una actividad que contenga parte del elemento escrito en la barra de búsqueda, esta reserva quedará filtrada (véase figura 31). En caso de no encontrarse el elemento, aparecerá un mensaje informativo de que no hay resultados (véase figura 32).

Figura 31: Búsqueda de reservas por barra de búsqueda.

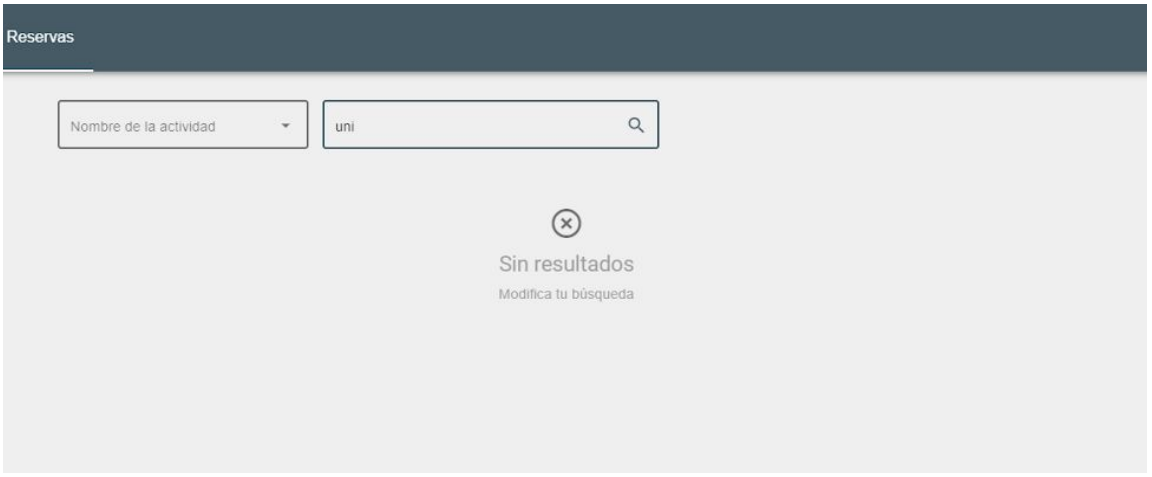

Figura 32: Búsqueda de una reserva no encontrada.

También se pueden filtrar reservas a través de un selector de actividades. Por ejemplo, en el caso de la figura 33, se busca filtrar una actividad por la cual no tiene reservas.

Figura 33: Búsqueda de actividades por actividad.

5.3.2. Panel del cliente

Por otra parte, si es el cliente que se conecta a la parte que ha sido destinada para él, la primera pantalla que visualizará será la de la figura 34. Aquí podrá elegir el idioma con el que querrá usar la aplicación. Además, aparecerá la opción de ver sus reservas y el conjunto de actividades ofrecidas por la empresa hotelera. Si la actividad entra dentro del plan contratado por el cliente, esta actividad será gratuita para el mismo y se identificará por la etiqueta “Incluido”.

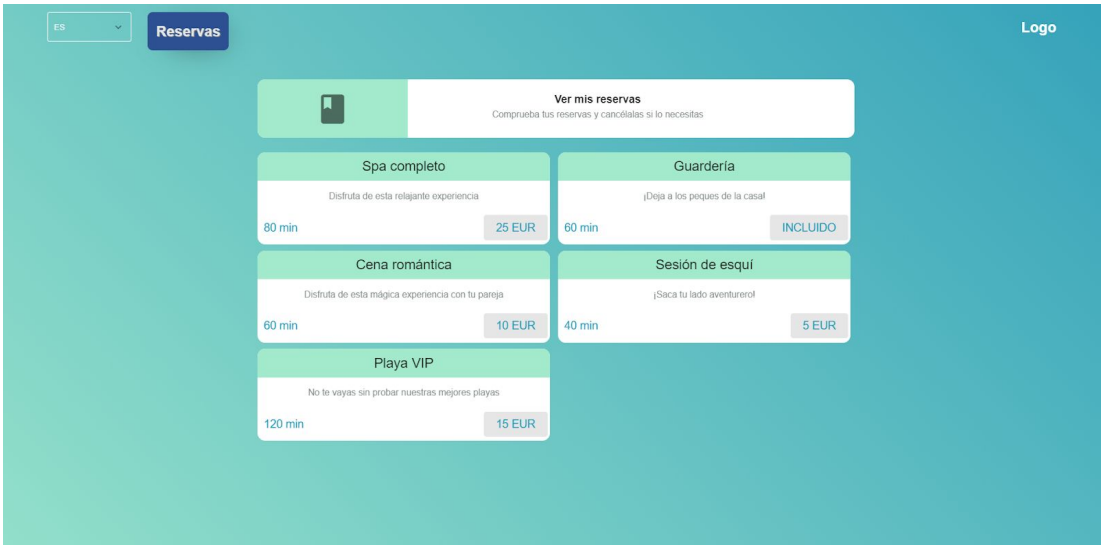

Figura 34: Página principal de la parte destinada para el cliente.

Si el cliente escoge reservar la primera actividad, verá la primera pantalla de la reserva. Esta contiene la información relacionada con dicha actividad y además con la opción de elegir el número de personas que participarán y el día que el cliente desea hacerla. Esto puede verse en la figura 35.

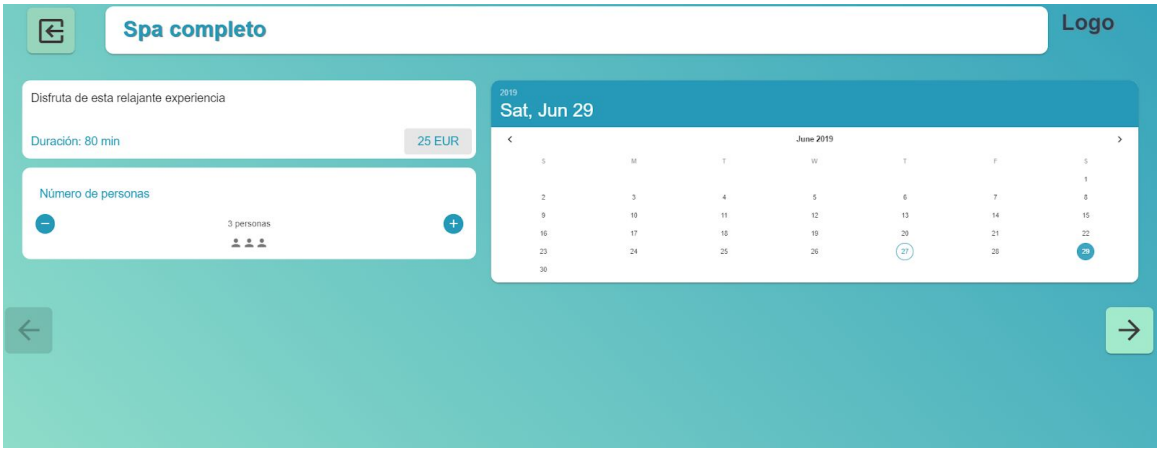

Figura 35: Información de la actividad y primeras opciones de reserva.

Cuando se han escogido las opciones de fecha y número de personas, aparece otra pantalla con el horario disponible (véase figura 36). Hasta que el cliente no escoja una hora predeterminada, no aparecerá el listado de grupos disponibles. Esto se puede ver en la figura 37.

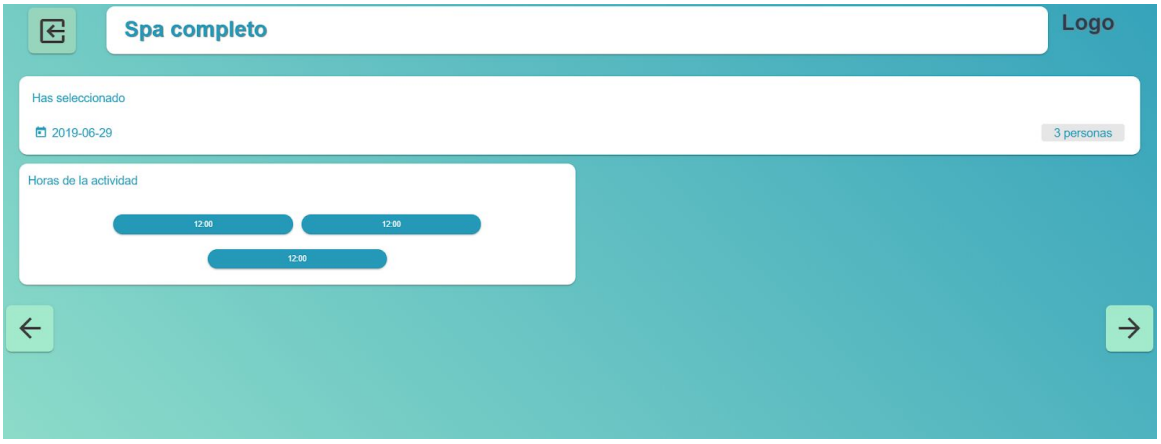

Figura 36: Selección de horario.

Figura 37: Selección de grupos.

Una vez seleccionado el grupo, el cliente deberá dejar registrado el nombre al que irá la reserva. Esta configuración de puede ver en la figura 38.

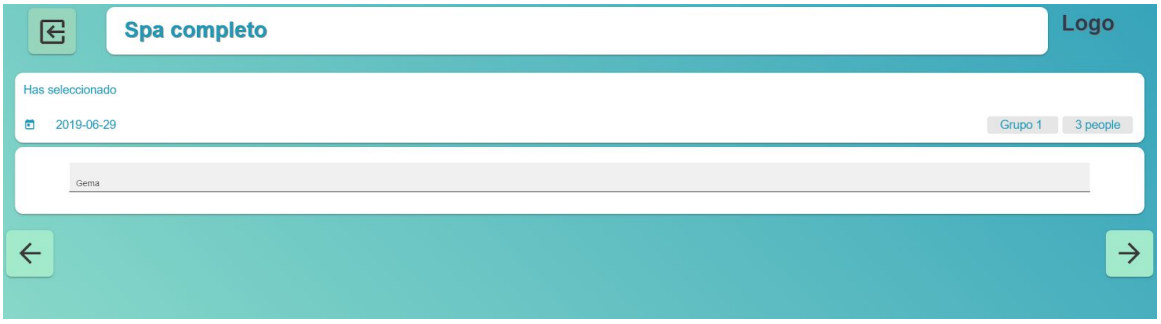

Figura 38: Nombre que tendrá la reserva.

A continuación, aparecerá una pantalla con la información de la reserva que ha sido confirmada. Además, aparecerá una cuenta atrás para volver a la página principal (figura 39).

Figura 39: Confirmación de reserva.

Finalmente, si en la página principal (figura 34) el cliente escoge ver sus reservas, aparecerán aquellas que haya reservado, tal y como puede verse en la figura 40. Por el contrario, aparecerá un mensaje informativo de que aún no tiene reservas (véase figura 41).

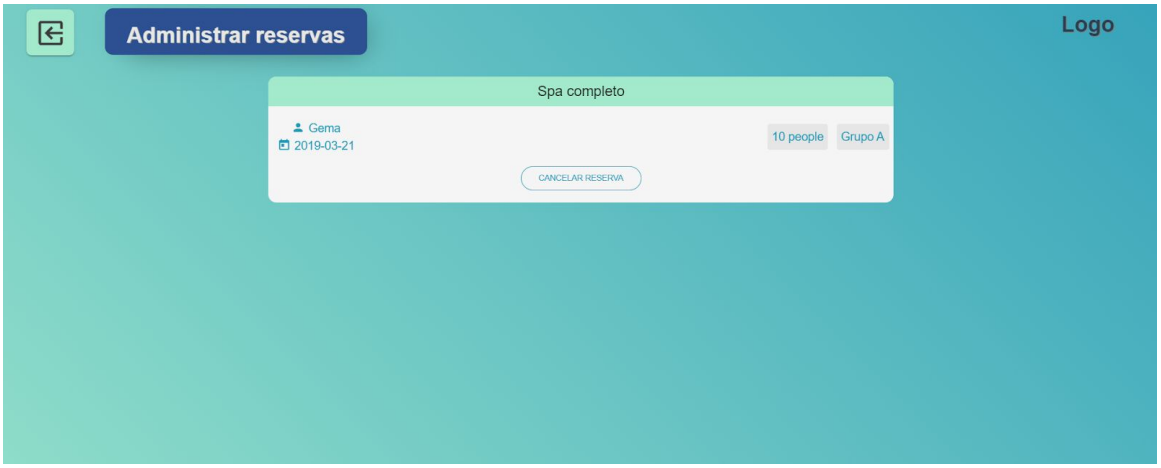

Figura 40: Listado de reservas del cliente.

Figura 41: Mensaje informativo cuando no hay reservas.

5.4. Pruebas

Una forma de comprobar que la aplicación funciona correctamente es mediante *localStorage*. Esto es debido a que los datos se almacenan en el navegador, ya que como se ha comentado en apartados anteriores, el proyecto se ha volcado solamente en el

desarrollo del frontend. Por ello, la forma de comprobar su funcionamiento es comprobando que en *localStorage* aparece almacenada los datos que hemos configurado en la aplicación. De la misma forma, por consola, también se puede imprimir la promesa que devolverá los datos por pantalla. Este ejemplo puede verse en la figura 42.

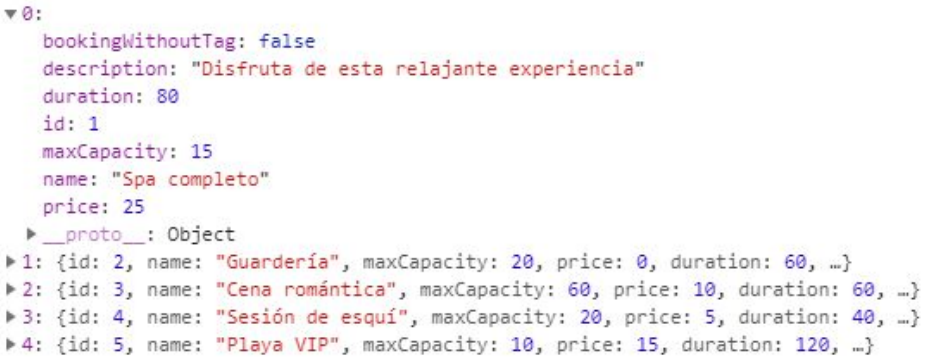

Figura 42: Impresión por consola de *this.$store.dispatch('activity/getActivities')*.

Otro tipo de prueba utilizada para ver un correcto funcionamiento de la aplicación es mediante la consola, donde se ejecutaban eventos por consola en el caso de que las operaciones (ej. edición del nombre de una actividad) se había realizado correctamente. Otra forma de comprobar que los datos se modifican, se crea una actividad o sesión o grupo de la sesión, es que los datos se visualicen en los componentes correspondientes.

Todas estas pruebas se han realizado sobre cada componente.

Capítulo 6

Conclusiones

Sin duda alguna haber realizado un proyecto como este no sólo me ha servido para aprender un nuevo tipo de tecnología, sino también la oportunidad de saber cómo se crea un proyecto y qué cosas se han de tener en cuenta para que se lleve a cabo de la forma más correcta y eficiente.

Por una lado, conocer *Vue.js* ha sido lo que más me ha ilusionado de todo el proyecto, dado que es una tecnología nueva y que tiene un futuro bastante garantizado. Además, utilizarlo para llevar a cabo un proyecto de gestión de reservas me ha hecho entender que cualquier proyecto puede desarrollarse con esta tecnología.

Por otro lado, desarrollar este proyecto me ha otorgado profundizar en *JavaScript*, dado que los conocimientos que tenía sobre este lenguaje de programación eran básicos.

De la misma forma, conocer cómo funciona una empresa y cómo se organiza, planifica y se lleva a cabo un proyecto es lo que más curiosidad me ha dado, ya que, a pesar de haber tenido asignaturas relacionadas con ello, no lo visualizas hasta que tratas con ello.

En conclusión, ha sido una experiencia que me ha permitido decidir acerca de qué rama en el mundo de la informática, querría desarrollar y especializar.

Bibliografía

[1] Main page JavaScript. https://www.javascript.com, realizado por PluralSight [Consulta: 9 de junio de 2019]

[2] Main page Vue.js. https://vuejs.org/, realizado por Evan You y el equipo de desarrolladores de Vue.js [Consulta: 6 de junio de 2019]

[3] Main page Vuex. https://vuex.vuejs.org/, realizado por Evan You y el equipo de desarrolladores de Vue.js [Consulta: 6 de junio de 2019]

[4] Main page Vuetify.js. https://vuetifyjs.com/, realizado por John Leider y el equipo de desarrolladores de Vuetify.js [Consulta: 8 de junio de 2019]

[5] Main page Vue-Router. https://router.vuejs.org/, realizado por Evan You y el equipo de desarrolladores de Vue.js [Consulta: 6 de junio de 2019]

[6] Material Design Icons. https://material.io/tools/icons/?style=baseline, realizado por Andrew C. Dvorak [Consulta: 10 de junio de 2019]

[7] JavaScript. https://es.wikipedia.org/wiki/JavaScript, realizado por Wikipedia [Consulta: 9 de junio de 2019]

[8] Tutorial Vuetify.

https://www.youtube.com/playlist?list=PL4cUxeGkcC9g0MQZfHwKcuB0Yswgb3gA5, realizado por The Net Ninja [Consulta: 15 de marzo de 2019]

Anexo I

Prototipos de la parte destinada al administrador

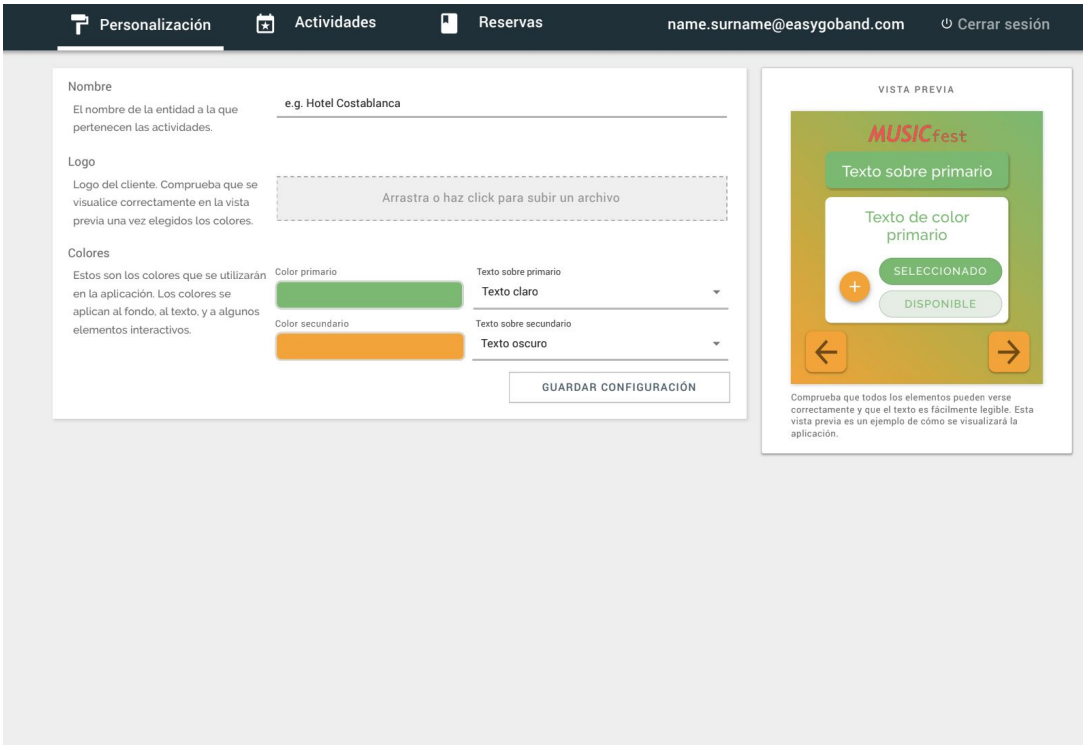

Figura 43: Prototipo de personalización (panel de administración)

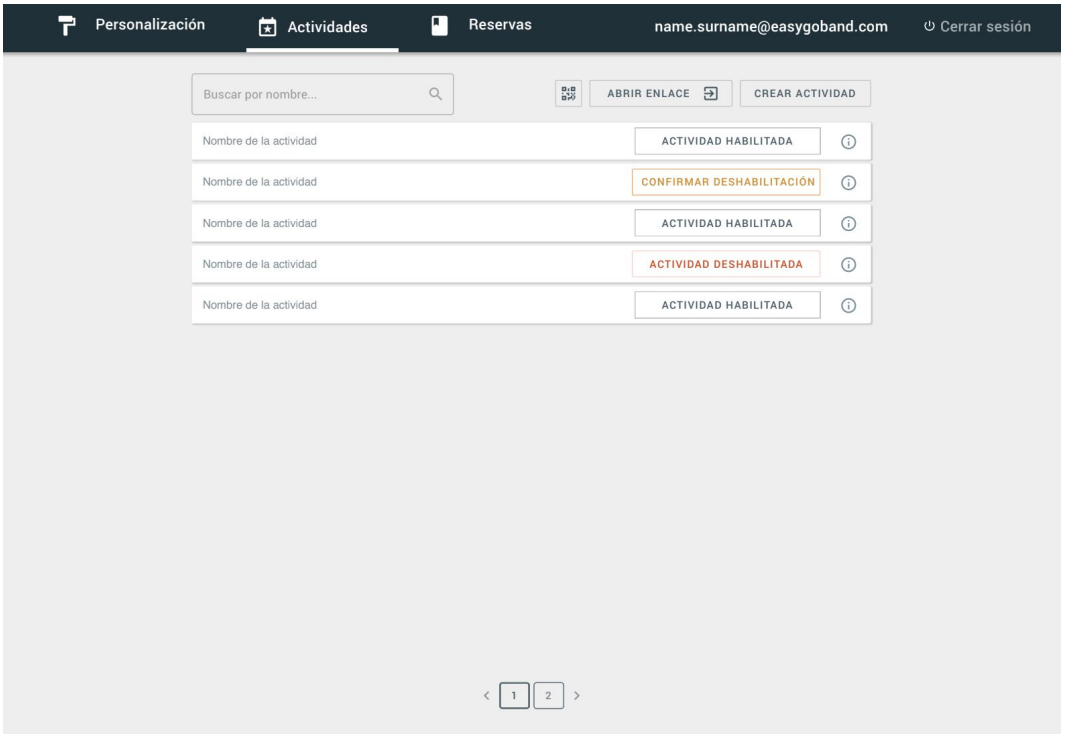

Figura 44: Prototipo de listado de actividades (panel de administración)

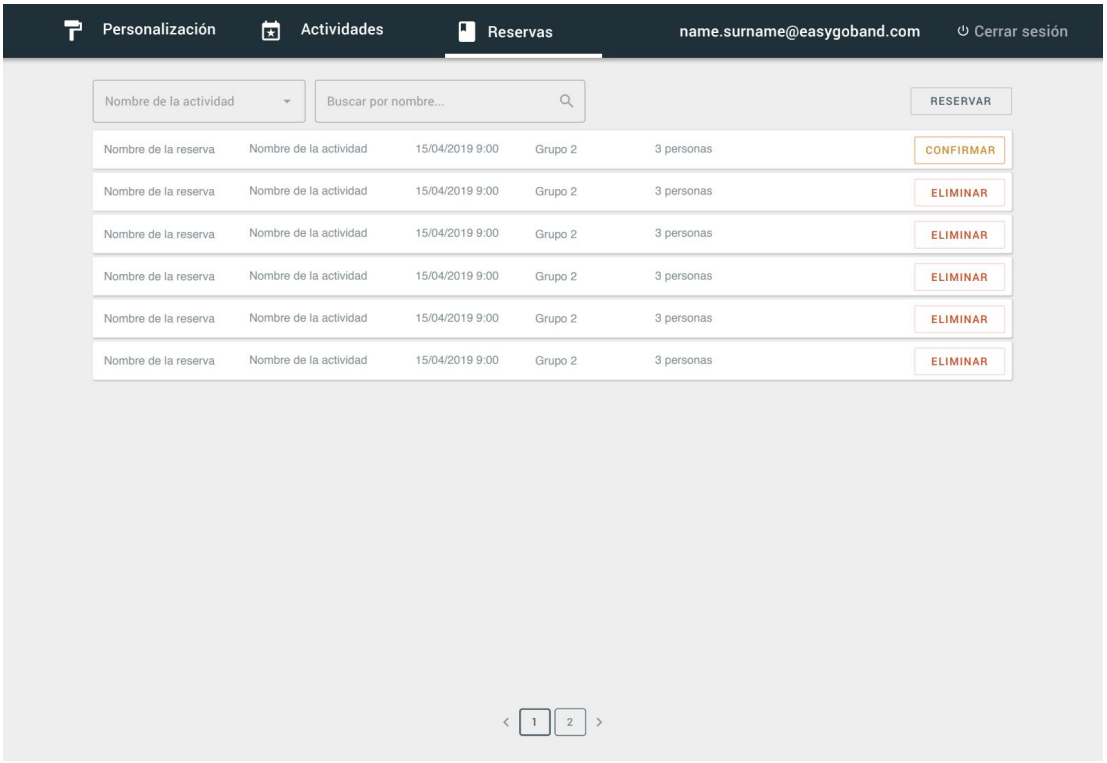

Figura 45: Listado de reservas (panel del administrador).

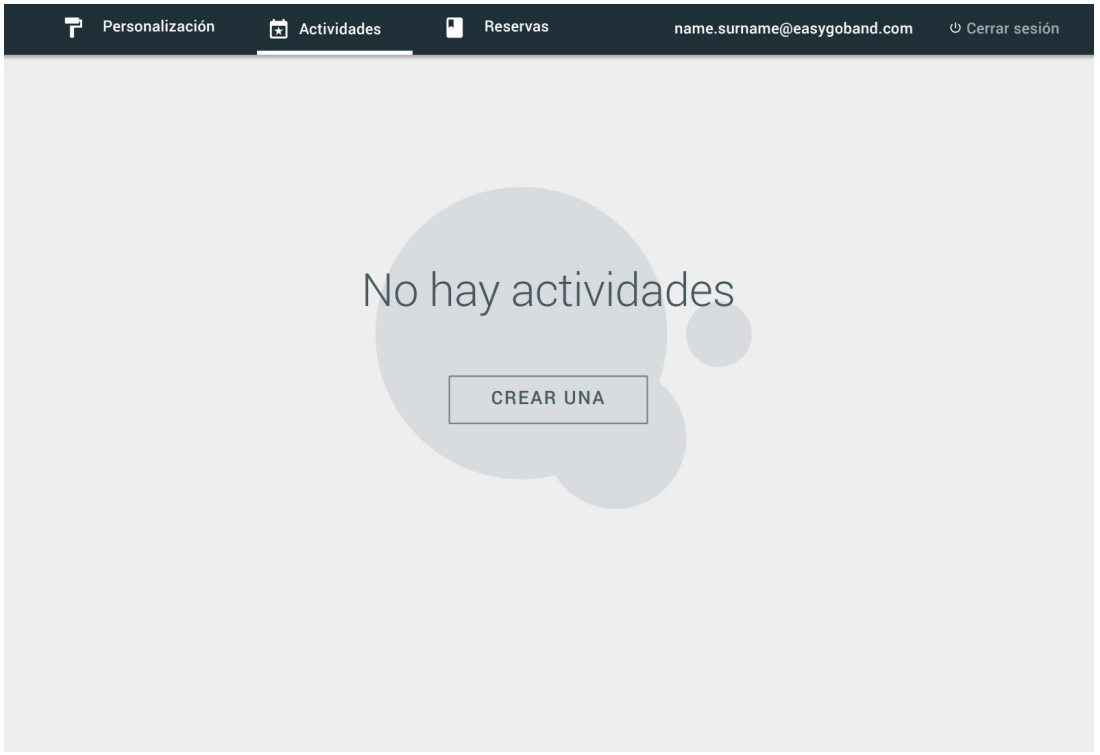

Figura 46: Mensaje informativo de cuando no hay actividades (panel de administración).

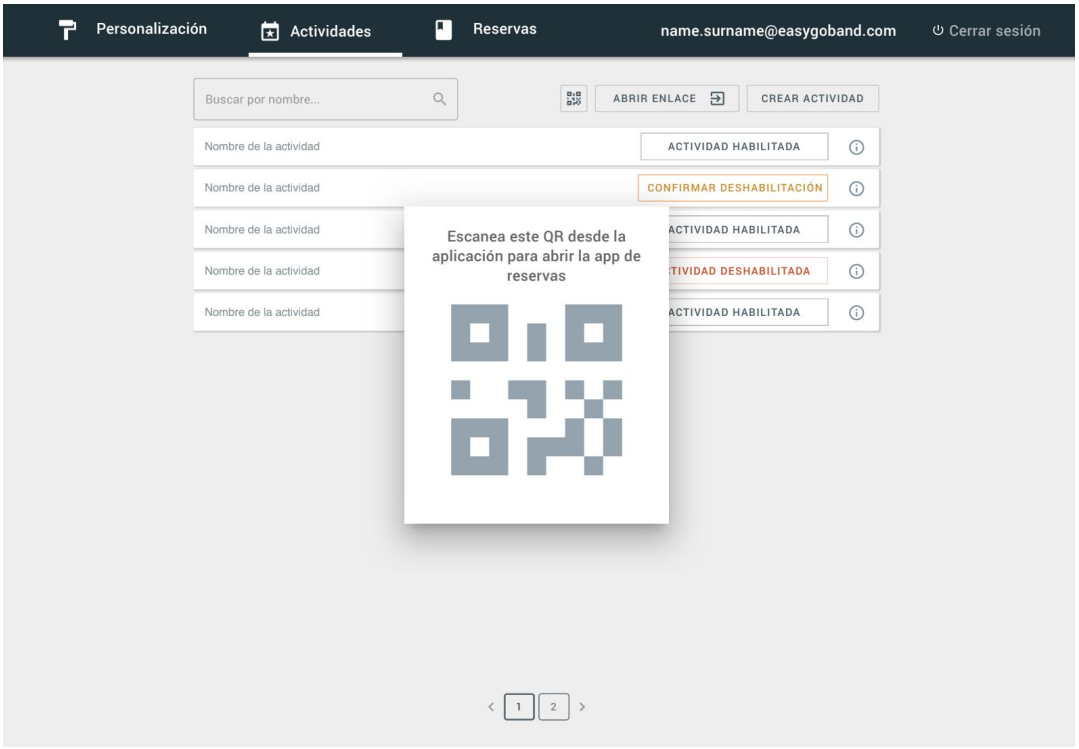

Figura 47: Lector de QR (panel de administración).

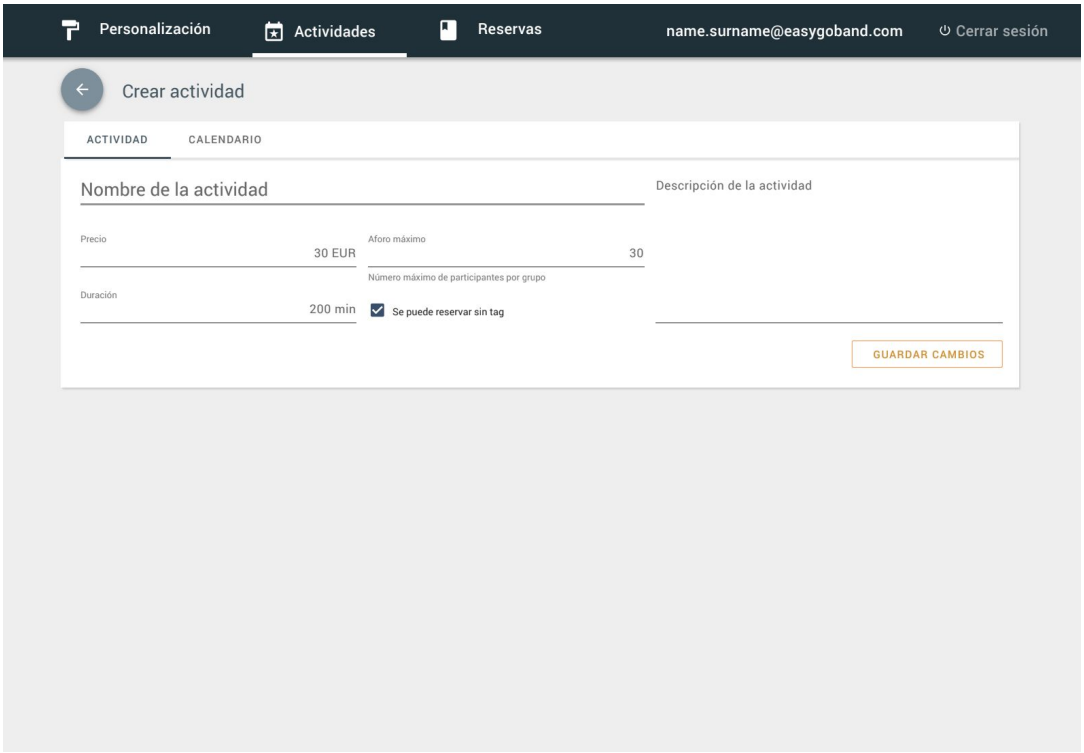

Figura 48: Crear actividad (panel de administración).

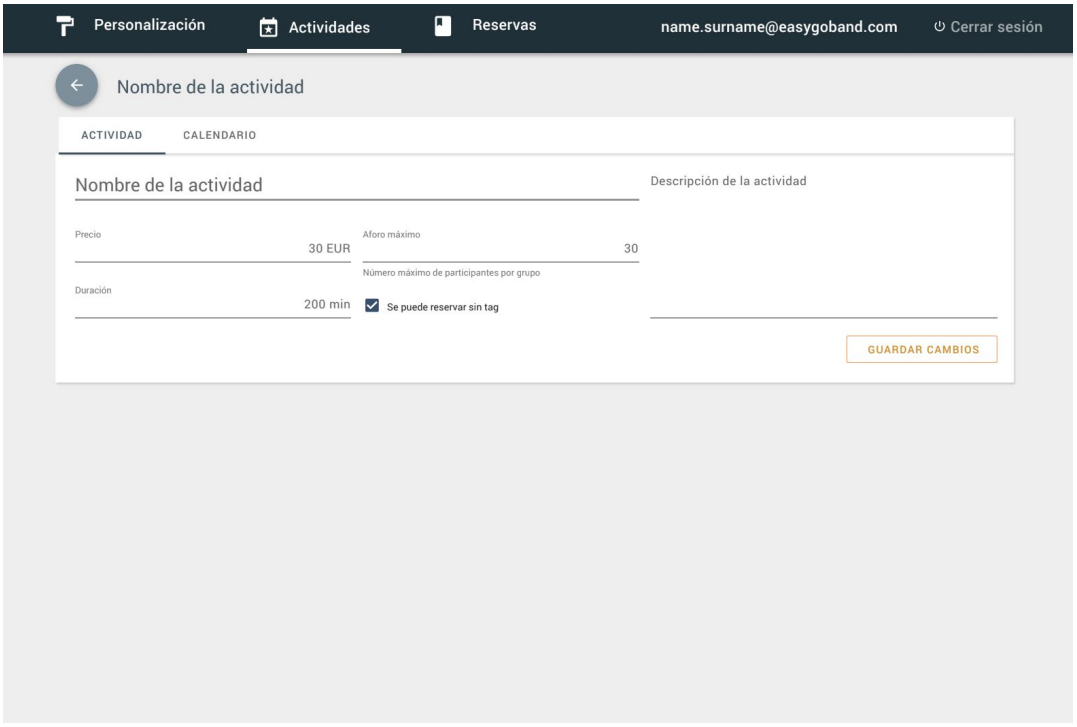

Figura 49: Editar actividad (panel de administración).

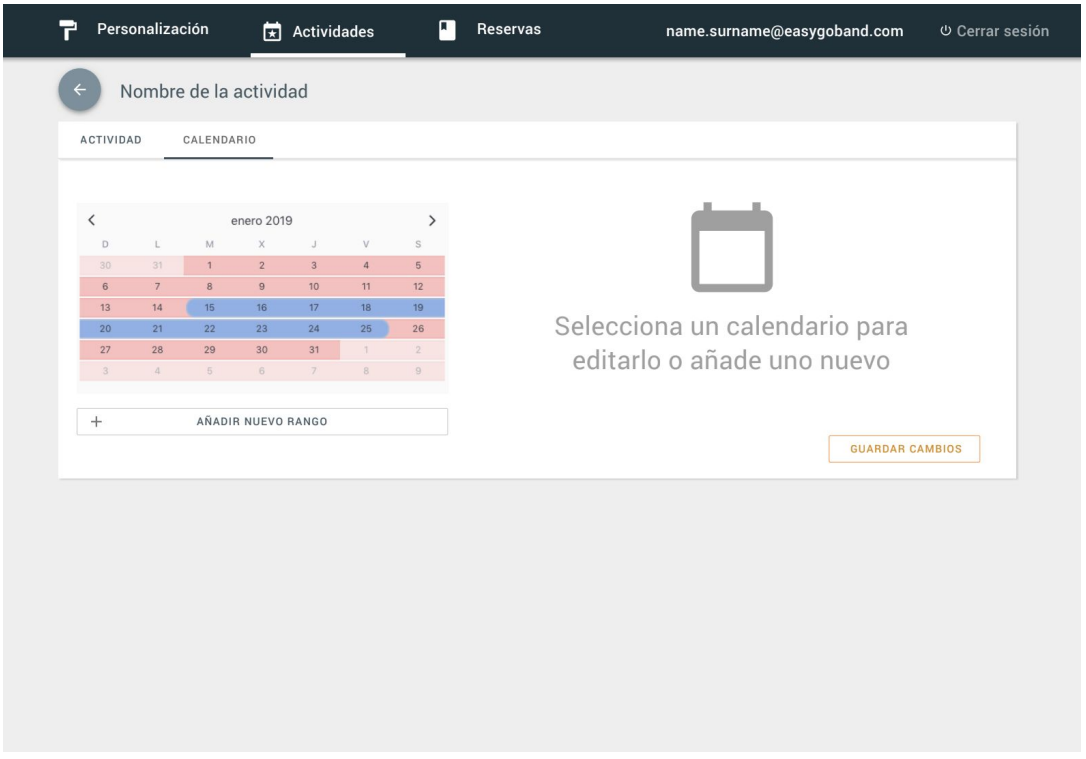

Figura 50: Calendario de la actividad (panel de administración).

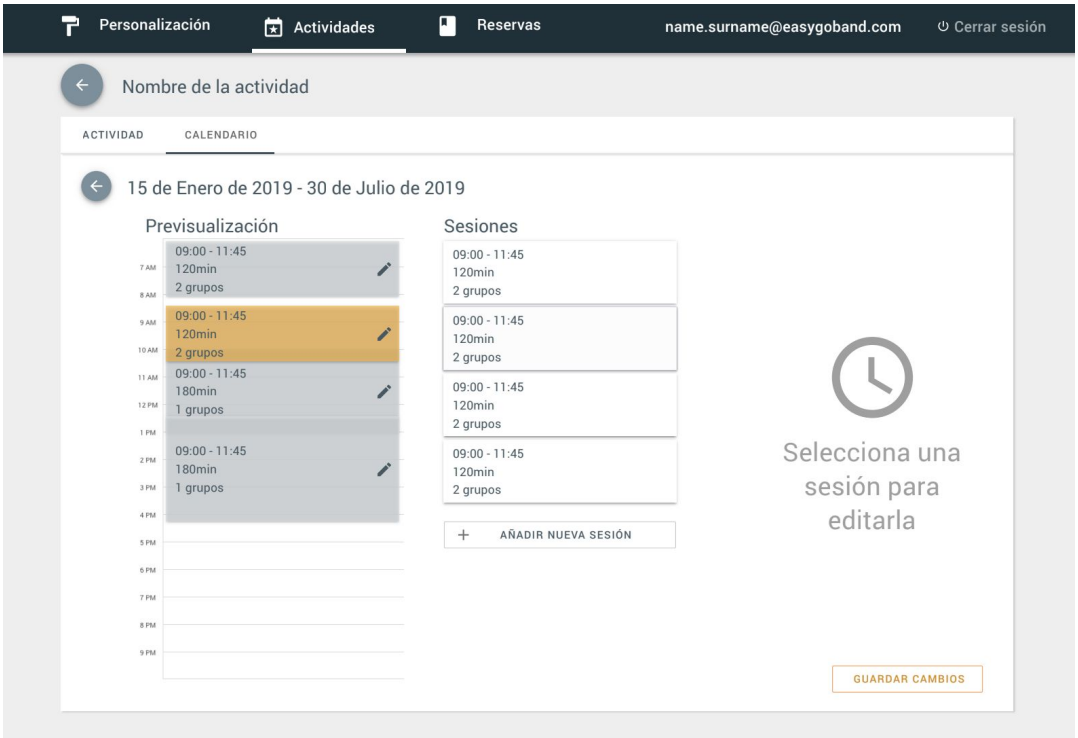

Figura 51: Visualización de sesiones (panel de administración).

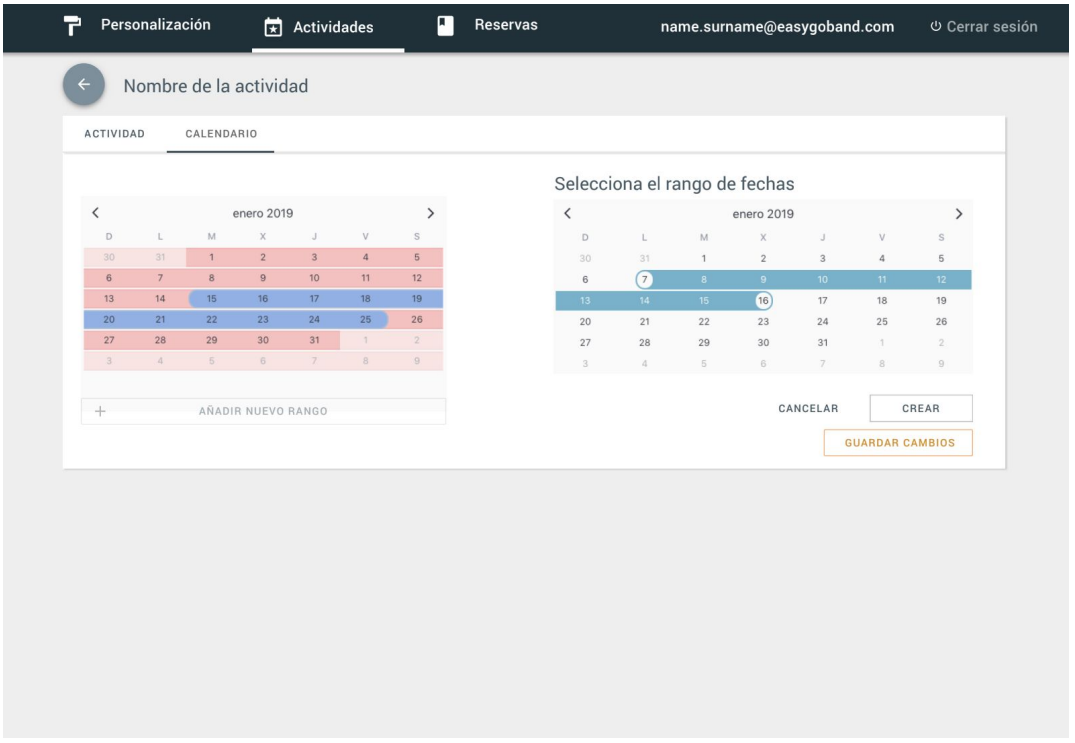

Figura 52: Crear rango de fechas (panel de administración).

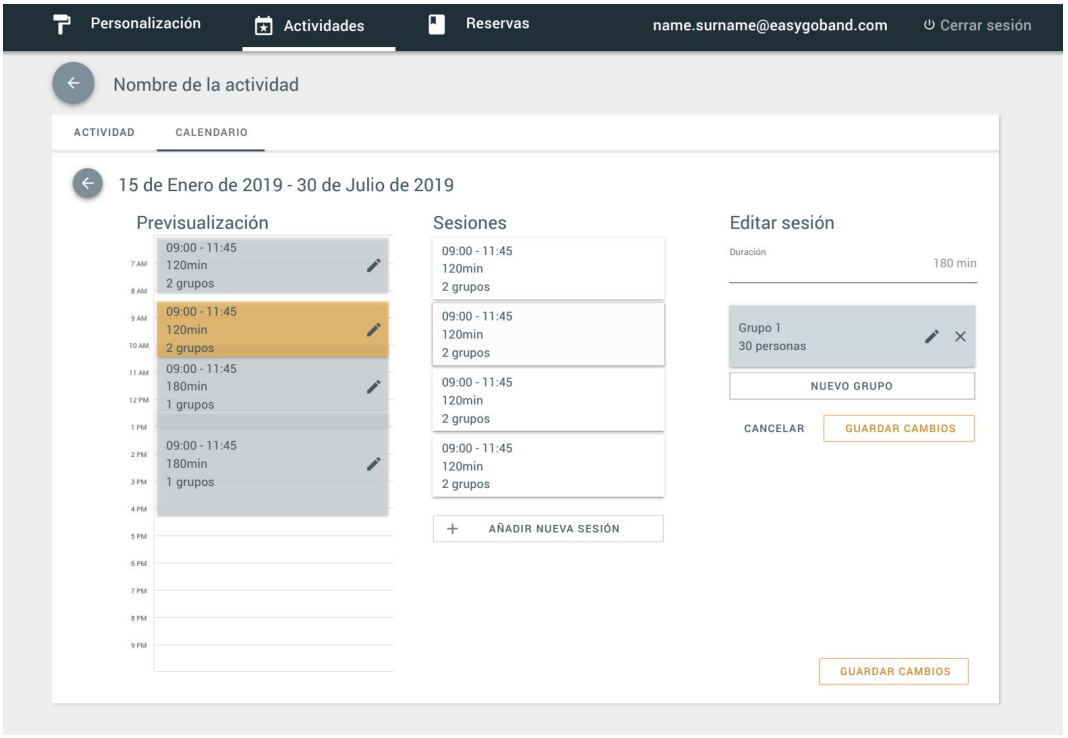

Figura 53: Editar sesión (panel de administrador).

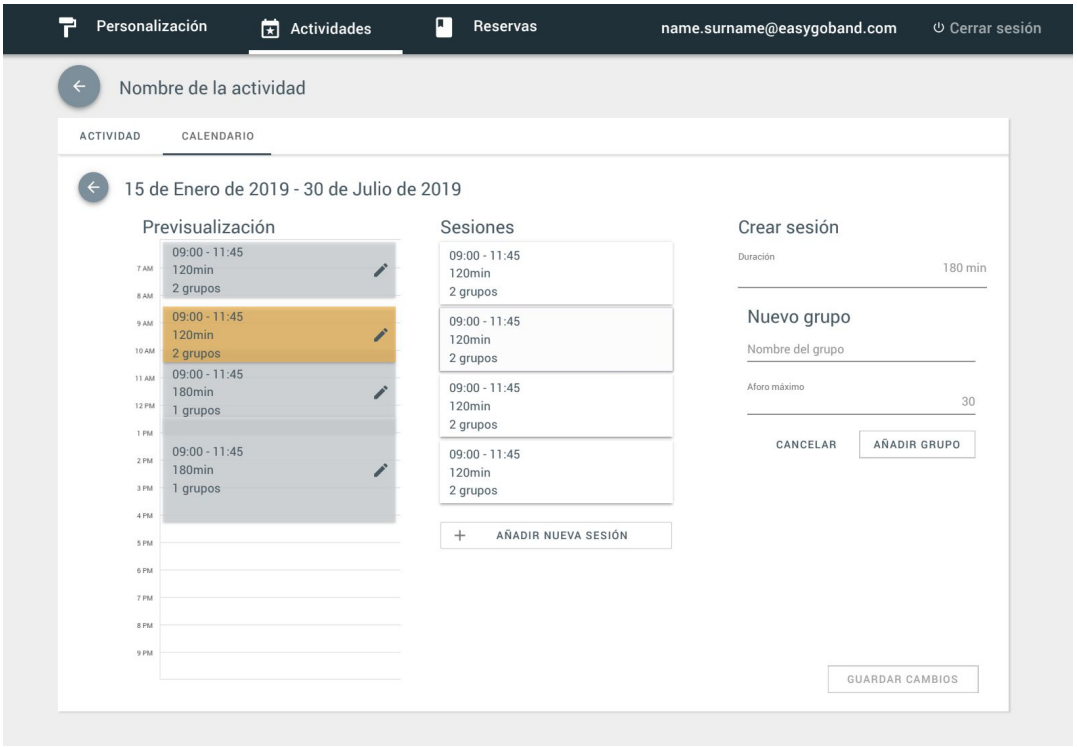

Figura 54: Añadir sesión (panel de administrador).

Anexo II

Prototipos de la parte destinada al cliente

Figura 55: Listado de actividades (panel del cliente).

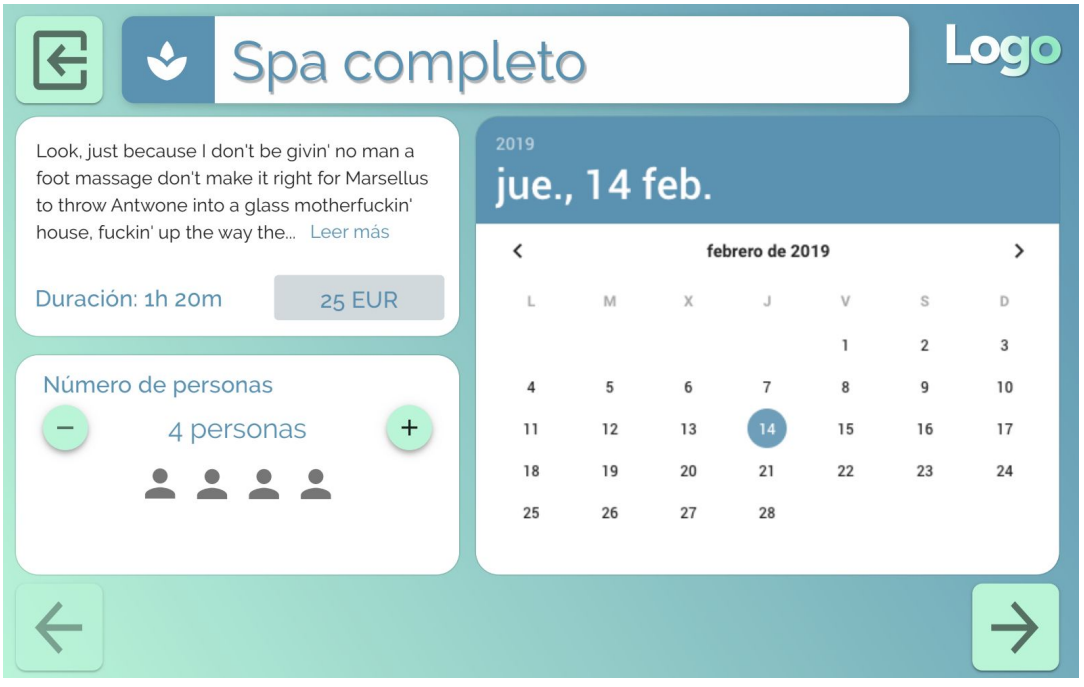

Figura 56: Selección de una actividad (panel del cliente).

Figura 57: Selección de actividad, horario y grupo (panel del cliente).

Figura 58: Nombre de la reserva (panel del cliente).

Figura 59: Reservar con o sin pulsera (panel del cliente).

Figura 60: Confirmación de reserva (panel del cliente).

Figura 61: Listado vacío de reservas (panel del cliente).