Grado en Ingeniería Informática

Trabajo de Final de Grado

Herramienta para la generación de reservas de hotel personalizables

Autor:
Unai Benajes Esbrí

Supervisor:
Miguel Salas Alegre
Tutor académico:
Jorge Sales Gil

Fecha de lectura: 13 de julio de 2022
Curso académico 2021/2022

Resumen

El presente documento expone el Trabajo de Fin de Grado realizado durante la estancia en prácticas en la empresa Easygoband. Pertenece al itinerario de Sistemas de Información del grado en Ingeniería Informática de la Universitat Jaume I.

La empresa Easygoband cuenta con una plataforma dirigida a los clientes para gestionar y administrar complejos hoteleros de forma cómoda y sencilla.

A lo largo de este documento se detalla el desarrollo de una herramienta para la generación de reservas de hotel personalizables, con el objetivo de realizar pruebas y demostraciones en la plataforma. Está dirigida al equipo técnico, comercial y operacional de la empresa.

La herramienta ofrece una personalización total para las reservas, permitiendo especificar habitaciones, añadir huéspedes y crear distintas cuentas del hotel.

El proyecto se ha realizado empleando una metodología predictiva tradicional con un ciclo de vida en cascada, donde se han definido y llevado a cabo una serie de tareas con un tiempo estimado.

Algunas de las herramientas y tecnologías utilizadas han sido: Vue.js como lenguaje de programación, Visual Studio Code como entorno de programación, GitLab para la gestión del código, Figma para el diseño de la interfaz y Jira para el control y seguimiento del tiempo.

Palabras clave

Reservas, hotel, huéspedes, formulario, one-page, aplicación web, Goguest, Vue.js, front-end.

Keywords

Bookings, hotel, guests, form, one-page, web application, Goguest, Vue.js, front-end.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

La empresa Easygoband World SL (Easygoband) [8] se dedica a la creación de aplicaciones *software* y soluciones *hardware* orientadas al sector turístico para la gestión integrada de hoteles, parques y eventos. En la Figura 1.1 se muestra su imagotipo.

Figura 1.1: Imagotipo de Easygoband World SL [8]

Principalmente, la empresa ofrece herramientas de análisis de datos para los clientes y soluciones para mejorar la estancia de las personas. Entre las soluciones más destacables, se encuentran unos dispositivos electrónicos (pulseras, tarjetas, etc.) capaces de agilizar los pagos y trámites dentro de las instalaciones. Además, son idóneos para identificar a las personas, y así mejorar la seguridad y controlar el acceso a las instalaciones.

Algunos clientes para los que la empresa ha realizado importantes proyectos son: Rototom, Aquarama y Nickelodeon Hotels & Resorts, entre muchos otros. También colabora con organizaciones destacadas, como por ejemplo, la Universitat Jaume I.

La empresa cuenta con dos plataformas bien diferenciadas: una plataforma para la gestión de eventos, denominada Gofun, y una plataforma para la gestión de hoteles, denominada Goguest. En la Figura 1.2 se muestran sus respectivos imagotipos.

El proyecto descrito en este documento se centra en la plataforma Goguest. Se trata de una herramienta capaz de generar reservas configurables en los hoteles registrados en la plataforma. Está completamente diseñada para el equipo interno, más en detalle, para el equipo técnico, el comercial y el operacional. La motivación principal para su desarrollo es poder proporcionar-

Figura 1.2: Imagotipos de Gofun y Goguest [8]

le al equipo técnico una herramienta para probar distintas configuraciones de reservas, y así facilitarles el testeo de la plataforma. Asimismo, poder proporcionarle al equipo comercial y operacional una plantilla de reservas a modo de demostración para mostrarles a los clientes el funcionamiento de Goguest.

De esta forma, el equipo interno puede tener a su disposición un formulario donde generar reservas totalmente personalizables y de forma rápida, en vez de crearlas manualmente cada vez que las requieran. Con el uso de esta herramienta, se ahorra en tiempo y esfuerzo, además de incrementar la autonomía entre departamentos.

1.2. Descripción del proyecto

Como ya he comentado, la herramienta será utilizada por el equipo interno de la empresa. Se podrán generar reservas de hotel personalizadas para facilitar las pruebas en la plataforma y servirá a modo de demostración para exponer el funcionamiento de las distintas aplicaciones.

Goguest cuenta con un panel de gestión para los clientes y para el equipo interno de la empresa, y una aplicación web para los huéspedes de las instalaciones. La herramienta a desarrollar se desplegará como extensión en el panel de gestión y se tratará de una aplicación web conteniendo un formulario con un diseño *one-page*[1].

En el formulario, se podrá elegir cualquier instalación[2] disponible en el servidor donde se haya desplegado la extensión. Se podrá especificar la duración de la estancia, los huéspedes que asistirán, las habitaciones reservadas y las cuentas con su respectivo saldo. Las cuentas sirven para controlar todos los gastos de los huéspedes, pueden soportar distintos tipos de divisa y se pueden asociar al huésped, a la habitación o a la reserva. Además, para cada huésped se podrá especificar su perfil completo y su periodo de la estancia si es que difiere de la especificada en la reserva.

El propósito de la herramienta es dar soporte a las pruebas y demostraciones, por lo que se buscará que sea visual y rápida de utilizar, y así lograr la mayor eficiencia posible y evitar la pérdida de tiempo. Se buscará hacer uso de herramientas de generación de datos y de campos rellenados por defecto para agilizar el proceso de reserva. Aún así, la herramienta deberá tener la posibilidad de ofrecer personalización detallada, para que el equipo interno pueda utilizarla en profundidad.

[1] De una sola página
[2] Hotel o complejo de hotel

1.3. Objetivos del proyecto

El objetivo principal de este proyecto es desarrollar una herramienta que permita generar reservas en las distintas instalaciones registradas en Goguest de forma rápida e intuitiva. Para facilitar la comprensión del objetivo principal, se desglosa en los siguientes subobjetivos (empresariales y tecnológicos):

A continuación, los objetivos empresariales del proyecto:

- Facilitar las pruebas al equipo técnico de Goguest.
- Desarrollar una herramienta para el equipo comercial y operacional que les permita mostrar a los clientes una demostración del funcionamiento de Goguest.
- Incrementar la autonomía del equipo comercial y operacional en la plataforma.

A continuación, los objetivos tecnológicos del proyecto:

- Permitir al equipo interno generar reservas en instalaciones de Goguest. Se debe permitir reservar un número determinado de habitaciones para un número determinado de huéspedes.
- Permitir al equipo interno asociar una cuenta a la reserva y agregar distintos tipos de divisa. También debe permitir asociar nuevas cuentas opcionales vinculadas directamente al huésped o a la habitación.
- Implementar una interfaz sencilla que permita al equipo interno generar de manera rápida e intuitiva reservas en las instalaciones.
- Implementar una librería para autogenerar datos de huéspedes para que sea más rápido su registro.
- Añadir datos autocompletados y campos por defecto para agilizar y facilitar la creación de reservas. Debe ser prioritario minimizar la interacción con la herramienta. En caso de buscar una personalización más detallada, permitir la modificación de los campos.

1.4. Alcance del proyecto

El alcance del proyecto abarca desarrollar una herramienta que genere reservas, permitiendo el registro de huéspedes, la asignación de habitaciones y la creación de cuentas. A continuación, se detalla el alcance, desglosado en: funcional, organizativo e informático.

1.4.1. Alcance funcional

La herramienta a desarrollar debe cumplir con las siguientes funcionalidades para el equipo interno (el equipo técnico, comercial y operacional):

- La herramienta debe permitir generar reservas en las instalaciones. Se deben tener en cuenta las fechas y horas de *check-in* y *check-out*. Además, debe permitir especificar la instalación donde generar la reserva y su estado (reservado, cancelado, bloqueado, etc.).
- La herramienta debe permitir crear un perfil de contacto para la reserva, pudiendo especificar su información general (nombre, apellido, correo electrónico, teléfono, etc.), el tipo de perfil (empresa, agencia, huésped, etc.) o la categoría de perfil (regular, celebridad, etc.).
- La herramienta debe permitir crear perfiles de huéspedes para la reserva. Para cada perfil de huésped, debe permitir determinar unas fechas y horas de *check-in* y *check-out* específicas. Además, se debe poder especificar la habitación del hotel donde se hospeda el huésped, el tipo y categoría de perfil, el tipo de huésped (niño, adulto, etc.) y la categoría de huésped (general, vip, familia numerosa, etc.).
- La herramienta debe permitir la creación de cuentas opcionales vinculables a la reserva, al huésped o a la habitación. Cada cuenta debe poder soportar distintos tipos de divisa (euro, dolar, la moneda del hotel, etc.), y para cada una de ellas, la herramienta debe permitir la posibilidad de configurarlas como pospago. Alternativamente, si se desean configurar como prepago, la herramienta debe permitir añadirles un saldo inicial.

1.4.2. Alcance organizativo

La herramienta a desarrollar tendrá un diseño *one-page*, que contará con los siguientes módulos y secciones:

- Módulo de la reserva: Engloba todos los aspectos sobre la reserva. Se divide en las siguientes secciones:
  - Sección de información general de la reserva: En esta sección se especificarán los datos genéricos correspondientes a la reserva.
  - Sección del perfil de la reserva: Esta sección incluirá la personalización del perfil de contacto asociado a la reserva.
  - Sección de la cuenta de la reserva: Esta sección incluirá la personalización de la cuenta asociada a la reserva.
- Módulo de los huéspedes: Engloba todos los aspectos sobre los huéspedes y la reserva de habitaciones. Se divide en las siguientes secciones:
  - Sección de la habitación: En esta sección se especificará la habitación deseada y se incluirá la personalización de la cuenta asociada a dicha habitación. Se podrán añadir tantas habitaciones como disponibles haya en la instalación seleccionada.

- Sección del huésped: Esta sección incluirá la personalización de los perfiles de los huéspedes y de las cuentas asociadas a dichos huéspedes. Se podrán añadir los huéspedes que se deseen, y estarán vinculados a una habitación en concreto.

1.4.3. Alcance informático

La plataforma Goguest cuenta con un servidor por cada instalación, con una base de datos independiente para cada uno. El acceso a la base de datos está implementado por medio de una API (Interfaz de Programación de Aplicaciones, del inglés *Application Programming Interface*) [10], que se conecta con la aplicación web y con el panel de gestión.

La herramienta a desarrollar será una aplicación web desplegada como extensión en el panel de gestión de la plataforma, donde únicamente se implementará la parte *front-end* [43] haciendo uso de la API. Será necesario determinar qué *endpoints* de la API [21] son los necesarios para el funcionamiento de la herramienta.

1.5. Herramientas y tecnologías

Durante el transcurso del proyecto, se han utilizado numerosas herramientas y tecnologías para el desarrollo e implementación del código, el diseño de interfaces, la gestión del tiempo, el seguimiento de tareas e incidencias y el control de versiones. En la Figura 1.3 se muestran las más importantes.

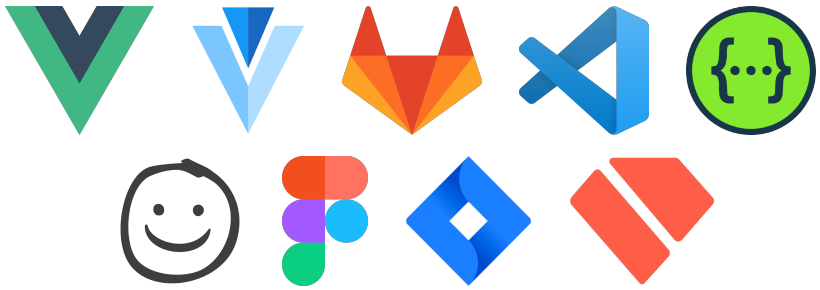

Figura 1.3: Herramientas y tecnologías utilizadas

Como lenguaje de programación se ha utilizado Vue.js [34], un *framework* de JavaScript. Es idóneo para el proyecto, ya que se trata de una tecnología diseñada para crear interfaces *one-page* de forma rápida y modular. Además, es el *framework* que utiliza la empresa para proyectos *front-end*, por lo que su uso facilita la integración de la herramienta a desarrollar en su plataforma.

Acompañado de Vue.js, se ha utilizado Vuetify [38], una de sus librerías para la implementación de componentes de diseño y usabilidad. También se ha hecho uso de Vuex [39], otra de sus librerías que implanta un patrón de diseño, permitiendo disponer de un almacén centralizado para todos los componentes de la aplicación y así asegurar un control en el acceso a los datos.

Para la gestión del código y el control de versiones se ha utilizado GitLab [13], desarrollado y basado en Git [12]. El repositorio del proyecto se ha vinculado a través del entorno de desarrollo Visual Studio Code [29], utilizado para la realización del código de la herramienta. También se ha hecho uso de Vetur [52], una extensión de Visual Studio Code para facilitar la programación con Vue.js. Además, para visualizar la documentación de la API requerida para el desarrollo de la herramienta, se ha utilizado Swagger UI [28].

Para el diseño de interfaces, se ha hecho uso de Balsamiq [5] y Figma [11]. Primero, para presentar un primer boceto sobre como podría ser la herramienta, se ha utilizado Balsamiq. Una vez corregido y revisado el boceto, se ha diseñado la interfaz en Figma, la herramienta de diseño colaborativa que utiliza la empresa.

Respecto al seguimiento de tareas e incidencias, se ha utilizado la herramienta en línea Jira [3]. De esta forma, se ha tenido un control sobre el desarrollo del proyecto y las incidencias que han podido llegar a surgir.

Para el control y seguimiento del tiempo, también se ha utilizado Jira, ya que permite especificar la estimación de cada tarea y su respectiva duración. Aunque adicionalmente, se ha hecho uso de la aplicación Holded [18], donde se pueden registrar las horas de trabajo realizadas y tener un seguimiento del horario laboral.

Respecto al *hardware* utilizado, se ha hecho uso de un ordenador portátil personal, aunque también estaban disponibles los ordenadores sobremesa de la empresa.

1.6. Estructura de la memoria

El primer capítulo, el actual, contiene una breve introducción a la empresa y al proyecto, el alcance y los objetivos del mismo, y las herramientas y tecnologías utilizadas.

El segundo capítulo cuenta con la planificación del proyecto, donde se detalla la metodología utilizada, la planificación de las tareas, la gestión de riesgos, la estimación de recursos y costes y el seguimiento del proyecto.

El tercer capítulo contiene el análisis del sistema, donde se detalla el diagrama de casos de uso y los requisitos funcionales de la aplicación.

El cuarto capítulo cuenta con el diseño del sistema, donde se detalla la arquitectura del sistema, la estructura de la API y el diseño de la interfaz.

El quinto capítulo contiene la implementación completa de la aplicación, donde además se especifican los patrones de diseño utilizados.

El sexto capítulo detalla las pruebas realizadas en la aplicación y su implantación.

El séptimo capítulo contiene las conclusiones personales, además de posibles extensiones para el proyecto.

Capítulo 2

Planificación del proyecto

2.1. Metodología

Para la elaboración del proyecto, se ha empleado una metodología predictiva tradicional basada en PMBOK (del inglés *Project Management Body of Knowledge*, en español Guía de los Fundamentos para la Dirección de Proyectos) [44]. Consideré que esta metodología era la que más se adaptaba para un proyecto de este estilo, ya que no es ni muy extenso ni complejo. Además, cuenta con tareas claramente definidas y con una estructura secuencial marcada, idóneo para este tipo de metodología.

El ciclo de vida utilizado en el proyecto es el modelo en cascada. Se trata de un modelo lineal, donde el final de una fase es el inicio de la siguiente. Aún así, en caso de haber fallos en fases anteriores, el modelo permite retroceder y corregirlos [17]. En la Figura 2.1 se muestra cada una de las fases del modelo en cascada.

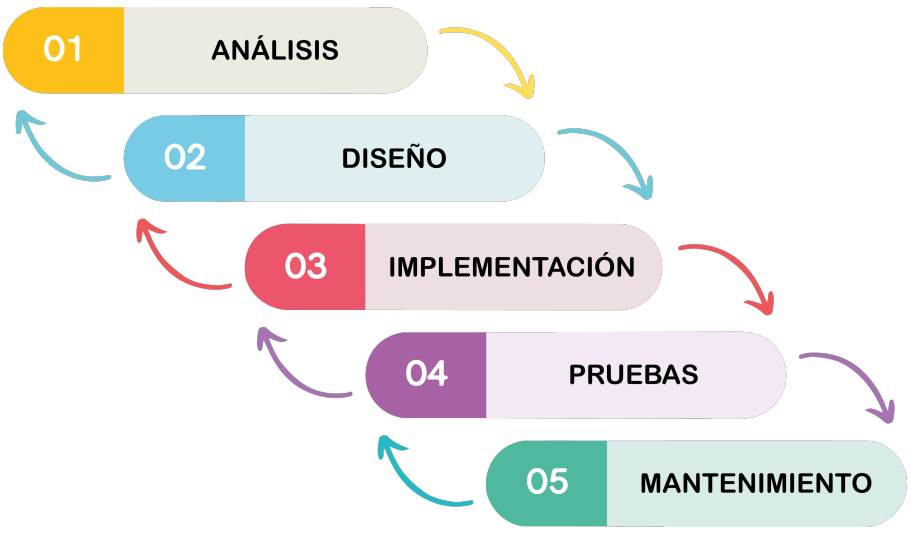

Figura 2.1: Esquema del modelo en cascada

Para adaptar correctamente esta metodología al proyecto, se ha desglosado cada una de las fases en tareas y subtareas más específicas. De esta forma, se puede apreciar mejor cada uno de los cometidos del proyecto y tener más definidos los pasos a seguir. En la Sección 2.2 se explica en detalle.

Una pequeña modificación a esta metodología tradicional, por parte de la empresa, consistió en realizar breves reuniones diarias al principio de la jornada para comentar los cambios y novedades del proyecto a los supervisores. De esta forma, se consiguió establecer cierto control sobre el desarrollo del proyecto. Personalmente, estas reuniones me sirvieron de ayuda para aclarar dudas y resolver los problemas que me iban surgiendo durante la semana.

2.2. Planificación

Como se ha detallado en la Sección 2.1, se ha utilizado una metodología predictiva con un ciclo de vida en cascada para el desarrollo del proyecto. En los siguientes apartados, se detallan las fases y tareas que se han planificado, con su correspondiente estimación temporal. También se incluyen la EDT (Estructura de Desglose del Trabajo, del inglés *Work Breakdown Structure* o WBS por sus siglas) [42] y el diagrama de Gantt [4] del proyecto.

2.2.1. Fases y tareas

A raíz del modelo en cascada de la Figura 2.1, se han definido las siguientes fases y tareas para el proyecto:

1. Inicio: Para dar comienzo al proyecto, se definen el alcance y los objetivos, y con ello, se redacta la propuesta técnica del proyecto. También es esencial en esta fase entender y adaptarse al funcionamiento básico de la empresa.
2. Planificación: En esta fase se incluye la gestión de recursos, de riesgos, de tiempo y de costes. Es esencial para este proyecto planificar qué recursos se van a utilizar, a qué coste y durante cuánto tiempo. La gestión de riesgos es fundamental para identificar y analizar los potenciales imprevistos que puedan surgir durante el proyecto.
3. Análisis: Para el análisis, es necesario definir los requisitos de la herramienta. Al tratarse de una aplicación web donde únicamente se desarrolla la parte *front-end*, no es necesario el análisis de la parte *back-end*. Se incluyen el diagrama de casos de uso y la especificación de requisitos.
4. Diseño: En esta fase se plantea y se diseña la interfaz *one-page* de la herramienta. Como ya he comentado, únicamente es necesario diseñar la parte *front-end*. Aún así, es fundamental en esta fase identificar los *endpoints* de la API requeridos para la aplicación.
5. Implementación: Se trata de la fase más importante del proyecto, donde se incluye la formación en las nuevas tecnologías y la implementación de la interfaz *one-page*. Esta última tarea se desglosa en dos grandes módulos, que abarcan la mayor parte del tiempo del proyecto.

6. Pruebas: Para el testeo de la aplicación, se incluyen las pruebas de validación, unitarias y de integración. De esta forma, se comprueba que la herramienta cumple con las especificaciones y funciona correctamente.

7. Mejora del proyecto: Esta fase comprende el diseño e implementación de nuevas funcionalidades. Se trata de una fase opcional, que puede llevarse a cabo en caso tener tiempo adicional una vez completadas las fases de implementación y pruebas de la aplicación.

8. Implantación: Esta fase incluye el despliegue de la aplicación en el panel de gestión de la empresa, y su mantenimiento en caso que pueda haber algún fallo durante su uso.

9. Cierre: Para dar por finalizado el proyecto, se documentan los aspectos importantes que debe saber la empresa sobre la herramienta.

2.2.2. Estructura de Desglose del Trabajo (EDT)

Para visualizar mejor las tareas dentro de cada fase del proyecto, en la Figura 2.2 se muestra la EDT o Estructura de Desglose del Trabajo.

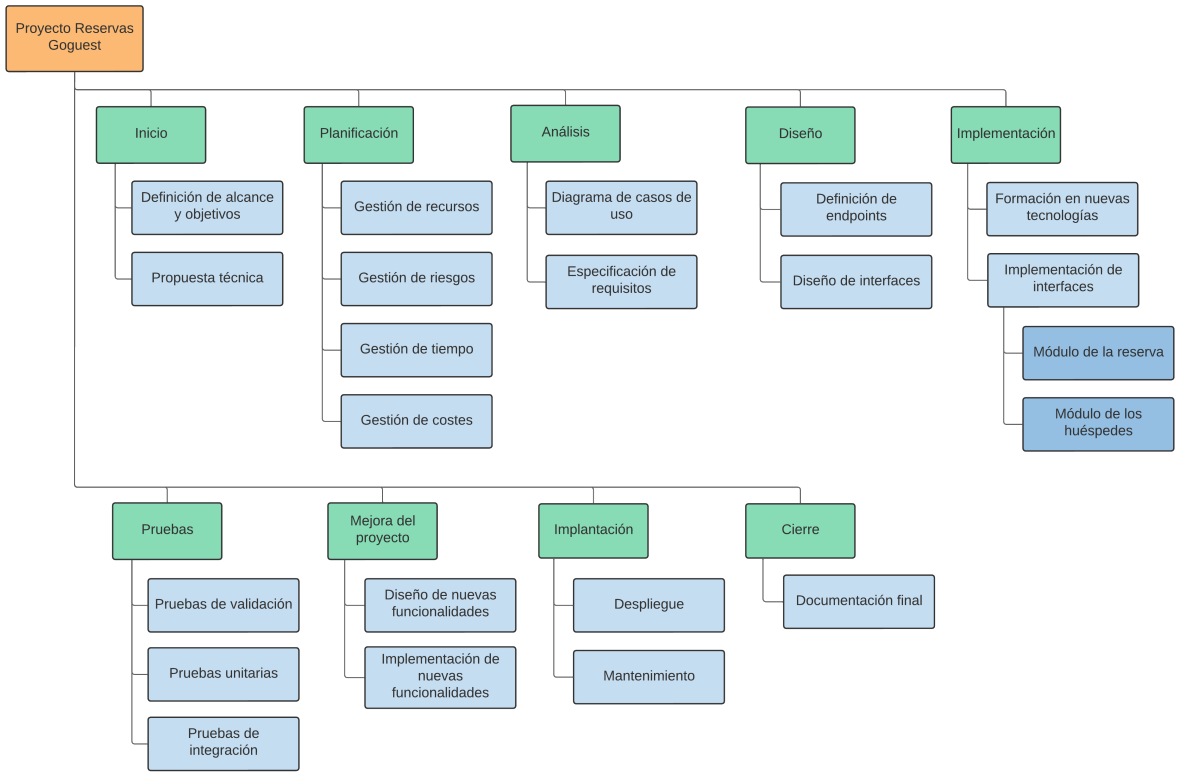

Figura 2.2: Estructura de Desglose del Trabajo (EDT)

2.2.3. Diagrama de Gantt

El proyecto tiene una duración total de 300 horas, divididas en 25 horas semanales. Se han planificado temporalmente las fases y tareas del proyecto para una duración total de 60 días laborales, teniendo en cuenta los días festivos. Con comienzo el martes 22 de febrero, se ha estimado su finalización para el jueves 19 de mayo.

La duración de cada tarea se ha regido por mi experiencia universitaria y por mi investigación en proyectos similares. Aún así, cualquier modificación o rectificación de la planificación realizada durante la estancia en prácticas, se puede revisar en detalle en la Sección 2.5.

En la Figura 2.3 se muestra cada una de las fases y tareas del proyecto con su correspondiente estimación temporal, y en la Figura 2.4 se muestra el diagrama de Gantt completo del proyecto.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Proyecto Reservas Goguest**	**60 días**	**mar 22/02/22**	**jue 19/05/22**	
2		**Inicio**	**3 días**	**mar 22/02/22**	**jue 24/02/22**	
3		Definición de alcance y objetivos	2 días	mar 22/02/22	mié 23/02/22	
4		Propuesta técnica	1 día	jue 24/02/22	jue 24/02/22	3
5		**Planificación**	**4 días**	**vie 25/02/22**	**mié 02/03/22**	
6		Gestión de recursos	1 día	vie 25/02/22	vie 25/02/22	4
7		Gestión de riesgos	1 día	lun 28/02/22	lun 28/02/22	6
8		Gestión de tiempo	1 día	mar 01/03/22	mar 01/03/22	7
9		Gestión de costes	1 día	mié 02/03/22	mié 02/03/22	8
10		**Análisis**	**4 días**	**jue 03/03/22**	**mar 08/03/22**	
11		Diagrama de casos de uso	1 día	jue 03/03/22	jue 03/03/22	9
12		Especificación de requisitos	3 días	vie 04/03/22	mar 08/03/22	11
13		**Diseño**	**4 días**	**mié 09/03/22**	**lun 14/03/22**	
14		Definición de endpoints	1 día	mié 09/03/22	mié 09/03/22	12
15		Diseño de interfaces	3 días	jue 10/03/22	lun 14/03/22	14
16		**Implementación**	**30 días**	**mar 15/03/22**	**jue 28/04/22**	
17		Formación en nuevas tecnologías	3 días	mar 15/03/22	jue 17/03/22	15
18		**Implementación de interfaces**	**27 días**	**vie 18/03/22**	**jue 28/04/22**	
19		Módulo de la reserva	12 días	vie 18/03/22	lun 04/04/22	17
20		Módulo de los huéspedes	15 días	mar 05/04/22	jue 28/04/22	19
21		**Pruebas**	**4 días**	**vie 29/04/22**	**mié 04/05/22**	
22		Pruebas de validación	1 día	vie 29/04/22	vie 29/04/22	20
23		Pruebas unitarias	2 días	lun 02/05/22	mar 03/05/22	22
24		Pruebas de integración	1 día	mié 04/05/22	mié 04/05/22	23
25		**Mejora del proyecto**	**6 días**	**jue 05/05/22**	**jue 12/05/22**	
26		Diseño de nuevas funcionalidades	2 días	jue 05/05/22	vie 06/05/22	24
27		Implementación de nuevas funcionalidades	4 días	lun 09/05/22	jue 12/05/22	26
28		**Implantación**	**3 días**	**vie 13/05/22**	**mar 17/05/22**	
29		Despliegue	1 día	vie 13/05/22	vie 13/05/22	27
30		Mantenimiento	2 días	lun 16/05/22	mar 17/05/22	29
31		**Seguimiento y control**	58 días	mar 22/02/22	mar 17/05/22	
32		**Cierre**	**2 días**	**mié 18/05/22**	**jue 19/05/22**	
33		Documentación final	2 días	mié 18/05/22	jue 19/05/22	30

Figura 2.3: Planificación temporal de las fases y tareas

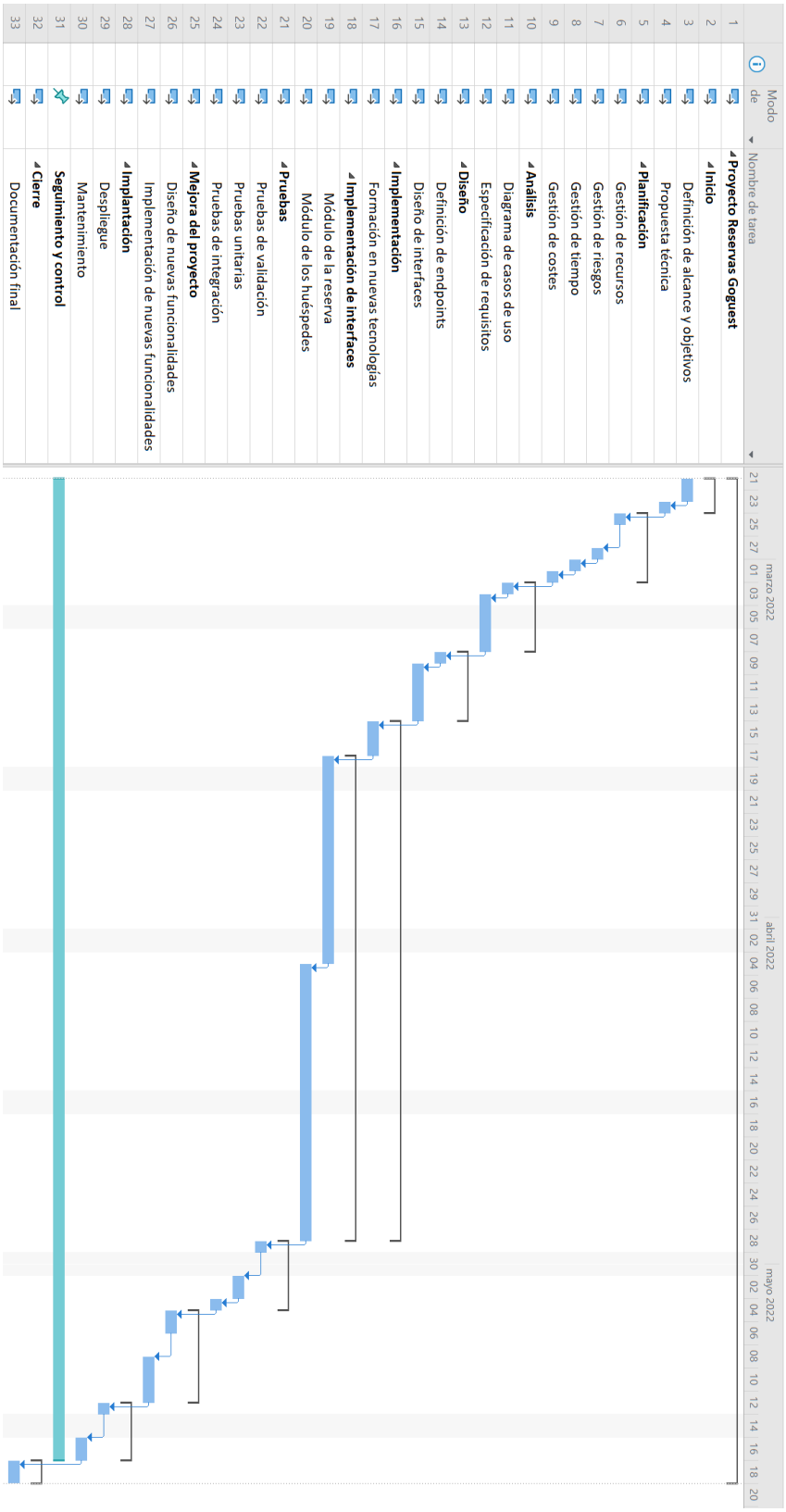

Figura 2.4: Diagrama de Gantt

2.3. Gestión de riesgos

El proyecto se ha realizado durante una estancia en prácticas, por lo que la gestión de riesgos ha sido fundamental para minimizar el impacto de los imprevistos ocasionados. En la Tabla 2.1, se muestra cada uno de los riesgos que se han tenido en cuenta para la elaboración del proyecto.

Riesgo	Descripción	Tipo
R1	Requisitos incorrectos, ambiguos o poco específicos	Riesgo del producto
R2	Falta de comunicación entre compañeros	Riesgo del proyecto
R3	Poca experiencia en el ámbito empresarial	Riesgo del proyecto
R4	Estimación temporal incorrecta	Riesgo del proyecto
R5	Pocos conocimientos sobre las herramientas y tecnologías	Riesgo del producto
R6	Alcance del proyecto incompleto o demasiado ambicioso	Riesgo del proyecto

Tabla 2.1: Listado de riesgos

Los riesgos que consideré más importantes son: *R1 - Requisitos incorrectos, ambiguos o poco específicos* y *R4 - Estimación temporal incorrecta*. Dado a la poca experiencia en proyectos de este estilo, era posible que algunos requisitos no estuvieran correctamente definidos, y por tanto, tener complicaciones a la hora de desarrollar el producto. También consideré la estimación temporal como un riesgo potencialmente alto por el mismo motivo, además que se trataba de un factor clave planificar el proyecto con el tiempo estipulado para la estancia en prácticas.

Otros dos riesgos que consideré que podían llegar a ocurrir son: *R2 - Falta de comunicación entre compañeros* y *R3 - Poca experiencia en el ámbito empresarial*. Antes de la realización de este proyecto, no había trabajado en ninguna empresa, por lo que dichos riesgos podían llevar a problemas durante el desarrollo. Uno de estos problemas, podía ser la falta de comunicación entre compañeros, y por culpa de ello, alargar innecesariamente el tiempo del proyecto debido a la acumulación de dudas o falta de ayuda.

Y por último, dos riesgos menos probables, pero no por ello menos importantes, son: *R5 - Pocos conocimientos sobre herramientas y tecnologías* y *R6 - Alcance del proyecto incompleto o demasiado ambicioso*. Consideré normal que un estudiante en prácticas como yo no dispusiera de los conocimientos suficientes para la realización del proyecto. Es por ello que, una correcta definición del alcance, es de suma importancia. Establecer un alcance incorrecto, puede llevar a un proyecto incompleto o a una sobrecarga del trabajo. Por esta razón, consideré ambos riesgos de forma complementaria, ya que para una correcta gestión del alcance, hay que tener en cuenta los conocimientos sobre las herramientas y tecnologías que se van a utilizar.

En la Tabla 2.2 se muestra, para cada riesgo, las medidas que se decidieron para su prevención y contingencia. Los planes de prevención se llevaron a cabo para evitar la aparición de los riesgos, y los planes de contingencia para minimizar las consecuencias en caso de manifestarse alguno.

Riesgo	Plan de prevención	Plan de contingencia
R1	Revisar los requisitos en detalle, consultarlos con los supervisores y dedicar más tiempo a esta tarea	Programar una reunión con los supervisores, redefinir los requisitos y corregir los errores
R2	Planificar quedadas y reuniones informales para conocer y llevarse mejor con los compañeros de trabajo	Establecer reuniones conjuntas con los compañeros de trabajo (a parte de las reuniones diarias) para comentar progresos y dudas genéricas
R3	Establecer reuniones iniciales con los supervisores para comentar el funcionamiento de la empresa	Preguntar en las reuniones diarias cualquier duda sobre el funcionamiento de la empresa
R4	Dejar un margen temporal para cada tarea, por si surgen imprevistos	Corregir la estimación temporal de las tareas y, en caso de necesitar más tiempo, descartar la fase *Mejora del proyecto*
R5	Realizar proyectos pequeños a modo de aprendizaje durante la formación	Ampliar el tiempo planificado para la formación
R6	Consultar el alcance con los supervisores	Programar una reunión con los supervisores, redefinir el alcance y adaptarse a los cambios

Tabla 2.2: Planes de prevención y contingencia para los riesgos

2.4. Estimación de recursos y costes del proyecto

Para estimar el coste total del proyecto, se distinguen tres tipos de recursos: los recursos humanos, los recursos tecnológicos (*hardware* y *software*) y los recursos indirectos. En los siguientes apartados, se muestra el cálculo del coste para cada tipo de recurso.

2.4.1. Costes humanos

Los costes humanos se calculan en base a todos los trabajadores implicados directamente en el desarrollo del proyecto, donde me incluyo yo como programador principal, junto a mis dos supervisores de la empresa.

Yo he trabajado un total de 300 horas, que es la duración estipulada para la estancia en prácticas. Mis dos supervisores se han encargado de ayudarme y guiarme a lo largo del desarrollo del proyecto, pero no han dedicado las mismas horas. Uno de ellos, al que nombraré como supervisor encargado, simplemente se ha ocupado de la burocracia y de asistir a las reuniones diarias para estar al corriente de las novedades. El otro supervisor, al que nombraré como supervisor práctico, se ha encargado de atender mis dudas, instruirme en los distintos aspectos de la empresa y corregir todas las versiones de mi proyecto. El supervisor encargado ha dedicado 15 minutos diarios a mi proyecto durante 60 días, lo que suma un total de 15 horas. En cambio,

el supervisor práctico, ha dedicado 1 hora diaria a mi proyecto durante 60 días, lo que suma un total de 60 horas.

A continuación, como yo no he cobrado un sueldo estándar durante las prácticas, se utilizará el sueldo medio de un programador junior, al igual que para los supervisores el sueldo medio de un programador estándar. Según Glassdoor [14][15], el sueldo medio de un programador junior en España es de 18.936€/año (1.578€/mes) y el de un programador estándar es de 25.680€/año (2.140€/mes).

Teniendo en cuenta la información anterior, en la Tabla 2.3 se muestra el cálculo del coste total de los recursos humanos. La cifra resultante, estimando un 20 % del coste total como gastos de contratación (impuestos de seguridad social, prestación social, etc.), es de 4.382,10€.

Recurso	Coste/mes	Coste/hora[1]	Horas	Coste total
Programador principal	1.578 €/mes	9,09 €/hora	300 horas	2.727 €
Supervisor encargado	2.140 €/mes	12,33 €/hora	15 horas	184,95 €
Supervisor práctico	2.140 €/mes	12,33 €/hora	60 horas	739,80 €
	5.858 €/mes	**33,75 €/hora**	**375 horas**	**3.651,75 €**
Contratación (20 %)				730,35 €
				4.382,10 €

Tabla 2.3: Costes humanos

2.4.2. Costes tecnológicos

Los costes tecnológicos se calculan en base a los recursos *software* y los recursos *hardware* utilizados para el desarrollo del proyecto.

En cuanto a recursos *hardware*, únicamente se ha hecho uso de un ordenador portátil personal. La empresa contaba con ordenadores de sobremesa propios, pero no se han incluido en los costes *hardware* ya que no se ha hecho uso de ellos. Como el recurso es de uso personal y no forma parte del producto, es necesario prorratear su coste para los 3 meses que ha durado la estancia en prácticas y el desarrollo del proyecto. En la Tabla 2.4 se muestra el coste prorrateado de este recurso *hardware*. La cifra resultante es de 67,50€.

Recurso	Coste	Vida útil	Meses	Coste prorrateado
Ordenador portátil	1.350 €	60 meses	3 meses	67,50 €
	1.350 €	**60 meses**	**3 meses**	**67,50 €**

Tabla 2.4: Costes tecnológicos *hardware*

[1]Considerando 8 horas al día, 5 días por semana, y por tanto, 40 horas semanales

Los recursos *software* utilizados, se describen en detalle en la Sección 1.5. También se han incluido como recursos *software* el dominio web de la empresa y el servidor donde se almacena la base de datos. La empresa contrata para estos dos servicios, entre otros, Amazon Web Services (AWS por sus siglas) [1]. Se trata de una gran colección de servicios de computación en la nube ofrecidos a través de Internet [40]. En detalle, se utiliza el servicio Amazon EC2 (Amazon Elastic Compute Cloud) [2] para desplegar un servidor por cada instalación de Goguest y crear todos los dominios y subdominios de la empresa. Por ello, el servidor implementado en Amazon EC2 se ha considerado un recurso *software*, ya que la empresa no dispone de servidores físicos.

En la Tabla 2.5 se muestra el cálculo del coste total de todos estos recursos *software*. La cifra resultante es de 178,50€.

Recurso	Coste/mes	Meses	Coste total
Vue.js	Gratuito	3 meses	0 €
GitLab	Gratuito (versión sin coste)	3 meses	0 €
Visual Studio Code	Gratuito	3 meses	0 €
Swagger UI	Gratuito	3 meses	0 €
Balsamiq	Gratuito (versión de prueba)	3 meses	0 €
Figma	12 €/mes	3 meses	36 €
Jira	7,50 €/mes	3 meses	22,50 €
Holded	15 €/mes	3 meses	45 €
Amazon EC2 (Servidor)	10 €/mes	3 meses	30 €
Amazon EC2 (Dominio)	15 €/mes	3 meses	45 €
	59,50 €/mes	**3 meses**	**178,50 €**

Tabla 2.5: Costes tecnológicos *software*

2.4.3. Costes indirectos

Los costes indirectos son aquellos costes que la empresa debe costear para el desarrollo de sus funciones, pero no están directamente relacionados con la producción [6]. Por ejemplo, el alquiler de oficinas, la electricidad, el agua, el internet, etc.

Para este proyecto, los costes indirectos supondrán un incremento del 20 % del coste total resultante. En el siguiente apartado, se muestra el coste total del proyecto incluyendo el incremento de los costes indirectos.

2.4.4. Coste total del proyecto

En la tabla 2.6, se muestra el coste total del proyecto, sumando los costes humanos, los costes tecnológicos (*hardware* y *software*) y los costes indirectos. La cifra resultante es de 5.553,72€.

Costes	Coste total
Costes humanos	4.382,10 €
Costes tecnológicos *hardware*	67,50 €
Costes tecnológicos *software*	178,50 €
	4.628,10 €
Costes indirectos (20 %)	925,62 €
	5.553,72 €

Tabla 2.6: Coste total del proyecto

2.5. Seguimiento del proyecto

En esta sección se detalla el seguimiento del proyecto que se llevó a cabo durante su realización en la estancia en prácticas. Como se ha detallado en la Sección 2.1 y en la Sección 2.2, el proyecto se ha planificado siguiendo una metodología predictiva tradicional con un ciclo de vida en cascada, y con una duración de 300 horas. A continuación, se muestra el seguimiento del proyecto dividido en los siguientes apartados: comienzo del proyecto, seguimiento del proyecto al 50 %, seguimiento del proyecto al 75 % y finalización del proyecto.

2.5.1. Comienzo del proyecto

Durante los primeros días, participé en una serie de reuniones para aprender sobre el funcionamiento básico de la empresa y conocer a mis compañeros y supervisores. El objetivo fue realizar una primera toma de contacto con la empresa y comenzar a tomar notas sobre los requisitos del proyecto, y así poder empezar la redacción de la propuesta técnica.

Los siguientes días comencé a redactar el alcance y los objetivos del proyecto, además de una pequeña descripción sobre la empresa. También elaboré una primera versión sobre la gestión temporal del proyecto y realicé la propuesta técnica del proyecto. Después de la redacción de la propuesta, comencé con la fase de planificación, específicamente con la gestión de recursos, riesgos, tiempo y costes.

La empresa contaba con documentos de *onboarding* (presentaciones, guías, videotutoriales, explicaciones, etc.), por lo que estos días también los aproveché para consultar dichos documentos y mejorar mis conocimientos sobre las distintas tecnologías y metodologías empleadas en la empresa.

Por lo general, durante esta etapa conseguí adaptarme al tiempo planificado y completé las fases de inicio y planificación exactamente como las había estimado. No surgió ningún imprevisto notable o algún problema que pudiera retrasar o dificultar esta primera etapa del proyecto.

2.5.2. Seguimiento del proyecto al 50 %

Después de las fases de inicio y planificación, comencé la fase de análisis. Realicé el diagrama de casos de uso y especifiqué los requisitos de la aplicación. Posteriormente, seguí con la fase de diseño donde especifiqué los *endpoints* necesarios para el proyecto, y comencé con el diseño de la interfaz *one-page*.

En lo referente a estas dos fases, no tuve ningún imprevisto, al igual que las fases iniciales. Se realizaron según estaba planificado. En cambio, sí surgieron modificaciones e imprevistos en la siguiente etapa.

Durante la fase de implementación, comencé con la formación en las nuevas tecnologías. Como hacía mucho tiempo que no realizaba proyectos con JavaScript y no practicaba lenguajes como HTML o CSS, la formación se alargó más de lo esperado. Esto generó un problema temporal que afectó a la siguiente tarea, la implementación de las interfaces, específicamente a la tarea de implementación del módulo de la reserva. Al ser una de las tareas más importantes del proyecto, tuve que solucionar el problema sin modificar su tiempo estimado.

Para solucionar el problema, decidí guiarme por los planes de contingencia definidos en la Sección 2.3, la gestión de riesgos. En concreto, el plan de contingencia definido para el riesgo *R4 - Estimación temporal incorrecta*. El plan expone "corregir la estimación temporal de las tareas y, en caso de necesitar más tiempo, descartar la fase *Mejora del proyecto*". Al no tener margen para ajustar el tiempo (debido a la importancia de las tareas posteriores), opté por eliminar completamente la fase mencionada. Además, eliminándola, conseguí añadir un poco más de tiempo a la tarea de implementación del módulo de la reserva, e ir un poco más holgado hacia esta fase. Fue una lástima no poder implementar nuevas funcionalidades para el proyecto, pero mejor eso que quedarse sin tiempo para terminar la parte fundamental de la aplicación.

Para apreciar mejor los cambios temporales de esta fase del proyecto, en la Figura 2.5 se muestra el diagrama de Gantt de seguimiento al 50 %. Las barras horizontales negras representan el trayecto planificado inicialmente, las barras horizontales azules el trayecto modificado completado y las barras horizontales rojas el trayecto modificado por completar.

Además, para dejar en claro los cambios temporales que se han realizado y qué tareas han sido las afectadas, se ha incluido la Tabla 2.7 que lo muestra en detalle.

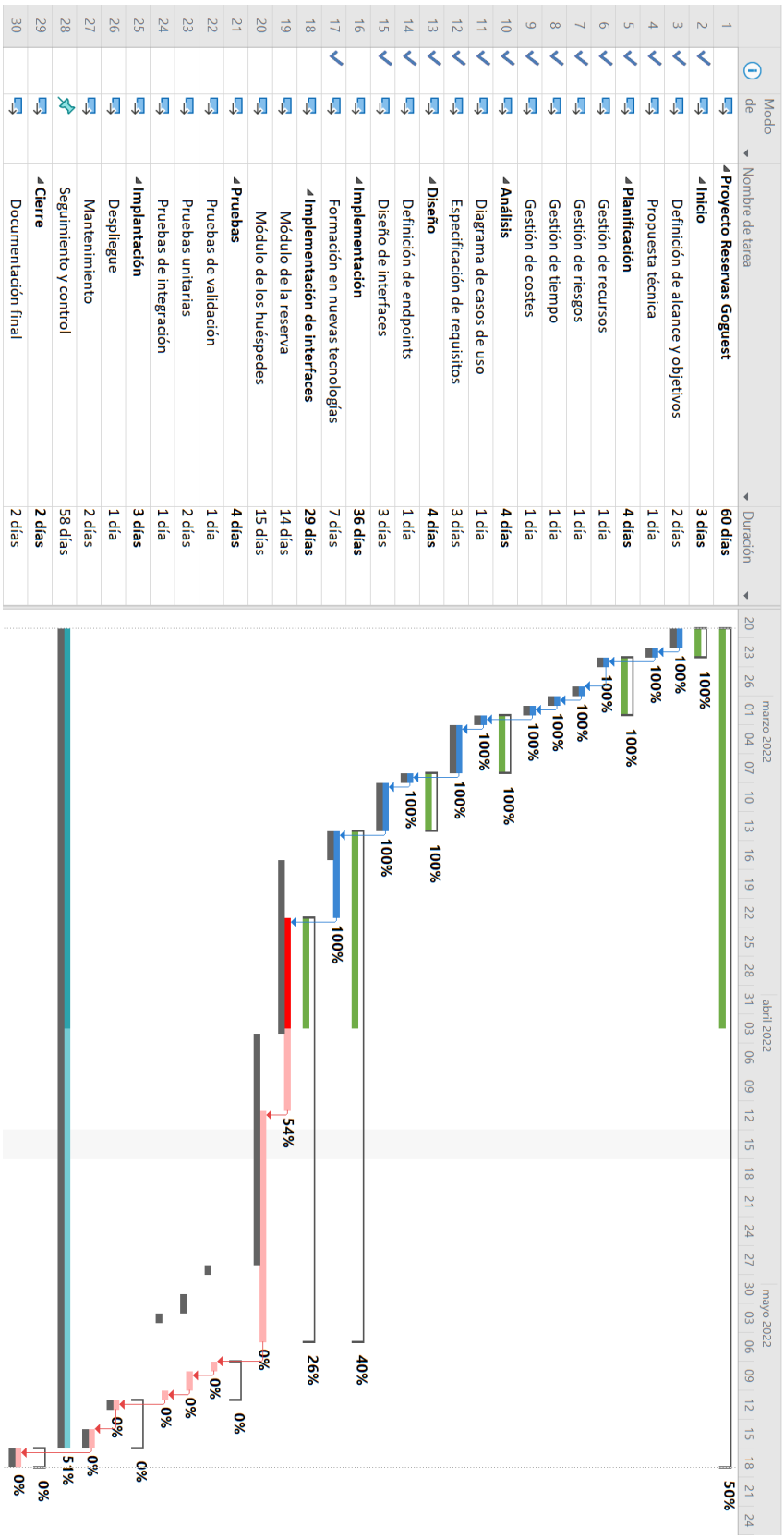

Figura 2.5: Diagrama de Gantt de seguimiento al 50 %

Identificador	Fase/Tarea	Duración inicial	Nueva duración
19	Formación en nuevas tecnologías	3 días	7 días
21	Módulo de la reserva	12 días	14 días
27[2]	Mejora del proyecto	6 días	0 días

Tabla 2.7: Cambios temporales en las fases y tareas para subsanar los imprevistos manifestados en el seguimiento al 50 %

2.5.3. Seguimiento del proyecto al 75 %

Continuando con la fase de implementación, específicamente con la tarea de implementación del módulo de la reserva, me di cuenta que acabarla no llevaba tanto tiempo como había planeado. Al terminarla, aún me sobraban cuatro días enteros (casi una semana) planificados para esa tarea.

Por tanto, al igual que en el apartado anterior, decidí seguir el plan de contingencia definido para la misma tarea. La diferencia ahora era que, al tener tiempo de margen, podía adaptar el tiempo de las tareas cómodamente. Entonces, lo que hice fue añadir los cuatro días de margen al tiempo de la siguiente tarea, la implementación del módulo de los huéspedes, que también se trataba de una de las tareas más importantes.

Continuando con el proyecto, me alivié habiendo tomado esta decisión, ya que me surgieron algunas complicaciones durante la implementación del módulo de huéspedes. Resultó que ciertos requisitos referentes a este módulo eran muy ambiguos y no se especificaron correctamente. Por tanto, al igual que había afrontado con el resto de imprevistos, decidí seguir el plan de contingencia pero esta vez para la tarea *R1 - Requisitos incorrectos, ambiguos o poco específicos*. En este caso, el plan expone "programar una reunión con los supervisores, redefinir los requisitos y corregir los errores". De modo que programé una reunión con ellos, y conseguimos corregir las ambigüedades. Aún así, los imprevistos ocasionados me hicieron perder tiempo, ya que tuve que modificar y rehacer ciertas partes del código.

Para apreciar mejor los cambios temporales de esta fase del proyecto, en la Figura 2.6 se muestra el diagrama de Gantt de seguimiento al 75 %. Al igual que el anterior diagrama de seguimiento, las barras horizontales negras representan el trayecto planificado inicialmente, las barras horizontales azules el trayecto modificado completado y las barras horizontales rojas el trayecto modificado por completar.

Del mismo modo que en el apartado anterior, para dejar en claro los cambios temporales que se han realizado y qué tareas han sido las afectadas, se ha incluido la Tabla 2.8 que lo muestra en detalle.

[2]Como se ha eliminado la fase por completo, el identificador corresponde al diagrama de Gantt de la Figura 2.4

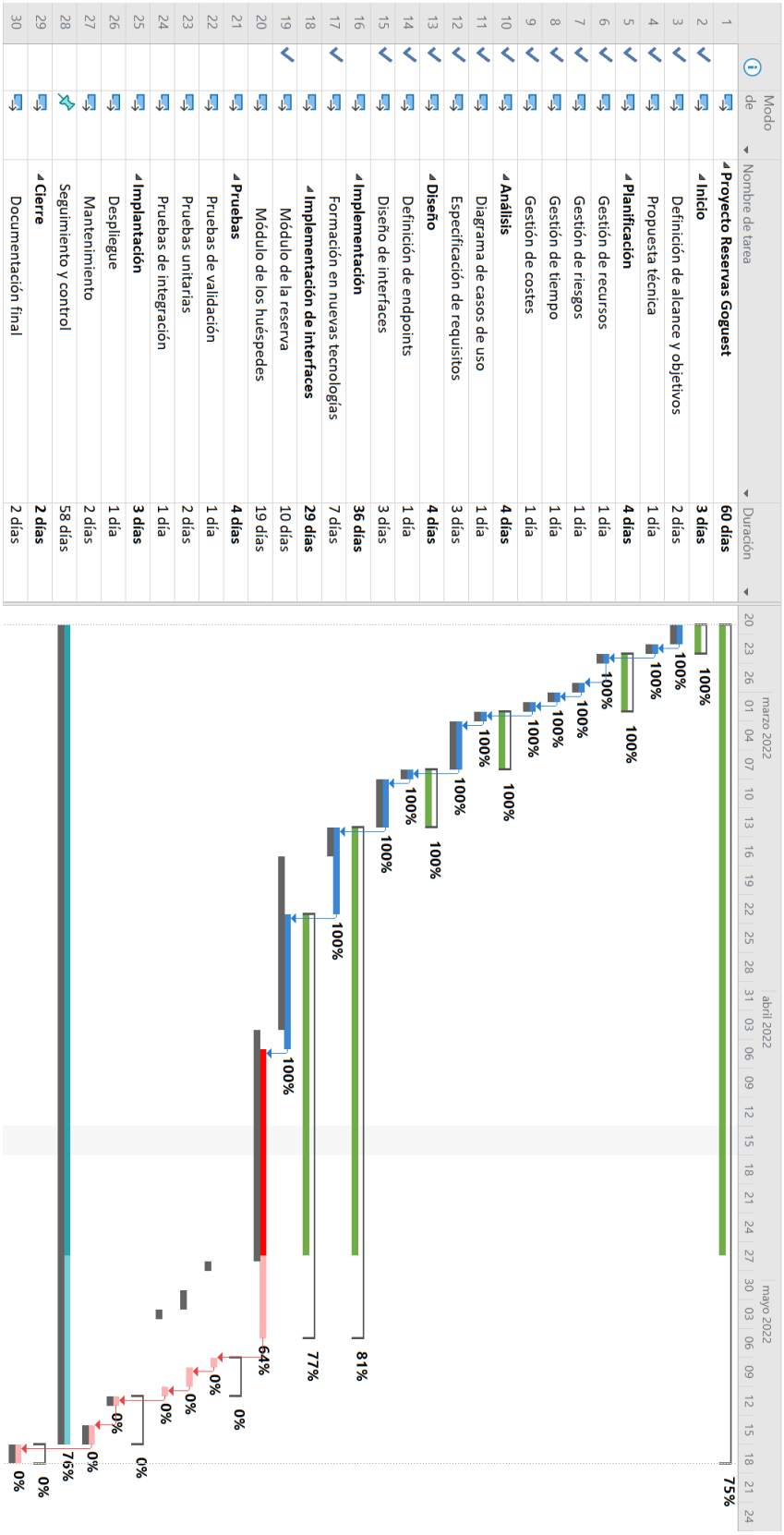

Figura 2.6: Diagrama de Gantt de seguimiento al 75 %

Identificador	Fase/Tarea	Duración inicial	Nueva duración
21	Módulo de la reserva	14 días	10 días
22	Módulo de los huéspedes	15 días	19 días

Tabla 2.8: Cambios temporales en las fases y tareas para subsanar los imprevistos manifestados en el seguimiento al 75 %

2.5.4. Finalización del proyecto

Continuando con el proyecto, finalicé justo a tiempo la fase de implementación. Fue gracias a que añadí más tiempo a la tarea de implementación del módulo de huéspedes que conseguí terminarla.

Siguiendo con las fases planificadas, dediqué cuatro días a las pruebas de la aplicación. Después desplegué la aplicación en el entorno de la empresa y me encargué del mantenimiento durante un par de días. A lo largo de estas dos fases no ocurrió ningún imprevisto, por lo que fue según lo planeado.

Finalmente, y para terminar el proyecto por completo, documenté algunos aspectos que consideré importantes para que la empresa pudiera consultarlos en caso de requerirlos. La mayoría, destinados al equipo interno.

Capítulo 3

Análisis del sistema

3.1. Diagrama de casos de uso

El diagrama de casos de uso es un diagrama de comportamiento en Lenguaje de Modelado Unificado (*Unified Modelling Language*, UML por sus siglas), con la finalidad de representar las relaciones de todos los objetos (participantes, casos de uso, etc.) involucrados en un determinado sistema [19]. En la Figura 3.1 se muestra el diagrama de casos de uso del proyecto.

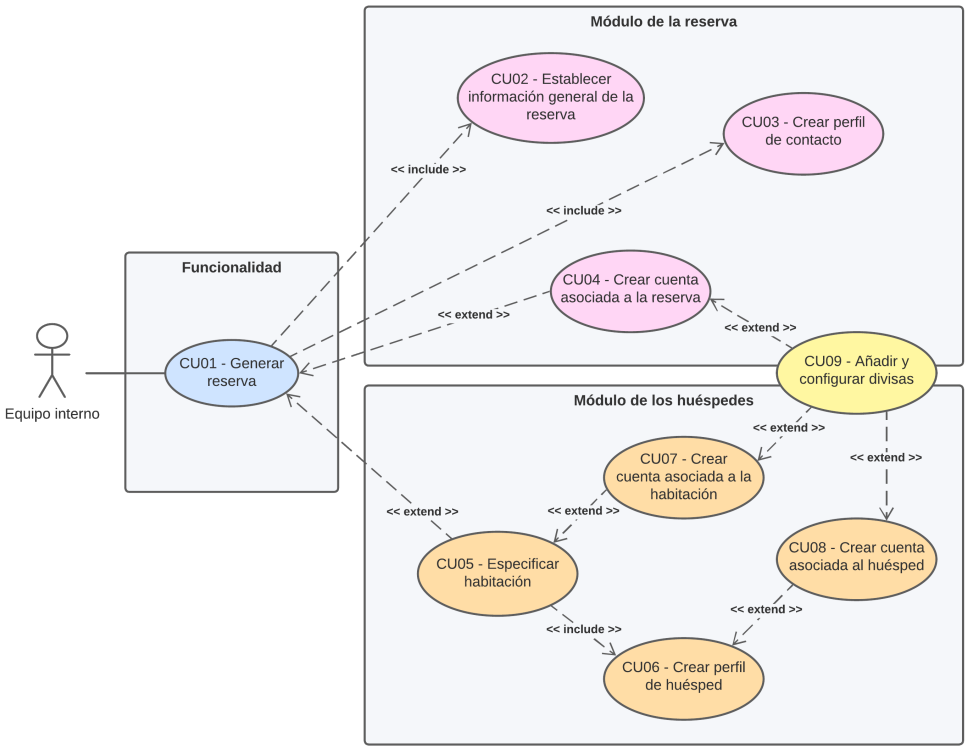

Figura 3.1: Diagrama de casos de uso

Como se observa en el diagrama, el equipo interno es el único actor presente. Se debe a que la herramienta solo está destinada a las pruebas del sistema y a la demostración del funcionamiento de las distintas aplicaciones de Goguest. A modo de recordatorio, el equipo interno está formado por el equipo técnico, el comercial y el operacional.

En el paquete de la funcionalidad, se observa el caso de uso principal *CU01 - Generar reserva*. A partir de este, se incluyen el resto de casos de uso divididos en dos paquetes distintos, simbolizando los dos módulos de implementación del proyecto. También comentar que el caso de uso *CU09 - Añadir divisas* se utiliza en los dos módulos, por lo que se ha posicionado en el borde de los dos paquetes.

Por último, destacar que algunos casos de uso son opcionales y están representados con una relación *"extend"*, mientras que otros son obligatorios y están representados con una relación *"include"*. Para este diagrama de casos de uso en concreto, he decidido utilizar en abundancia este tipo de relaciones por un motivo: al tratarse de una herramienta que contiene una única interfaz *one-page* incluyendo un formulario, he creído fundamental destacar qué casos de uso son obligatorios y cuáles son opcionales respecto a su caso de uso base. De esta forma, se puede apreciar qué secciones del formulario incluyen a otras, y también cuáles son opcionales y cuáles son obligatorias. No suelo hacer un uso excesivo de este tipo de relaciones en los diagramas de casos de uso, pero lo he creído conveniente dada la estructura peculiar del proyecto. Para más información sobre este tipo de relaciones, consultar (Kashif, s.f.) [22].

3.2. Requisitos funcionales

A partir de los casos de uso especificados en el apartado anterior, se han definido los requisitos funcionales de la aplicación, donde para cada caso de uso se ha detallado un requisito funcional. Con esto, se facilita la etapa de implementación, ya que se tienen documentados todos los comportamientos que debe cumplir la aplicación al detalle. Las Tablas 3.1 a la 3.9 muestran cada uno de los requisitos funcionales definidos.

Requisito funcional RF01	
Caso de uso	CU01 - Generar reserva
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir generar reservas en la plataforma.
Actor principal	Equipo interno
Precondición	n/a
Secuencia normal	1. El usuario ejecuta la aplicación desde el panel de gestión 2. El sistema abre una nueva pestaña con el formulario 3. El usuario rellena el formulario y genera la reserva 4. El sistema cierra la pestaña con el formulario y redirige al usuario al panel de gestión
Excepción	Errores de validación en el formulario.
Importancia	Necesario
Prioridad	Alta
Comentarios	Para generar múltiples reservas, es necesario volver a ejecutar la aplicación desde el panel de gestión.

Tabla 3.1: Requisito funcional RF01

Requisito funcional RF02	
Caso de uso	CU02 - Establecer información general de la reserva
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir introducir los datos básicos relativos a una reserva.
Actor principal	Equipo interno
Precondición	El usuario ha ejecutado la aplicación desde el panel de gestión.
Secuencia normal	1. El usuario selecciona la instalación donde generar la reserva 2. El usuario selecciona el estado de la reserva 3. El usuario establece las fechas y horas de *check-in* y *check-out*
Excepción	n/a
Importancia	Necesario
Prioridad	Alta
Comentarios	Los campos para la instalación y la fecha y hora de *check-in* son obligatorios.

Tabla 3.2: Requisito funcional RF02

Requisito funcional RF03	
Caso de uso	CU03 - Crear perfil de contacto
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir crear un perfil de contacto asociado a la reserva.
Actor principal	Equipo interno
Precondición	El usuario ha ejecutado la aplicación desde el panel de gestión.
Secuencia normal	1. El usuario introduce los datos personales del perfil 2. El usuario selecciona el tipo de perfil 3. El usuario selecciona la categoría de perfil
Excepción	n/a
Importancia	Necesario
Prioridad	Alta
Comentarios	De los campos de datos personales, es obligatorio introducir el nombre.

Tabla 3.3: Requisito funcional RF03

Requisito funcional RF04	
Caso de uso	CU04 - Crear cuenta asociada a la reserva
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir crear una cuenta vinculada a la reserva.
Actor principal	Equipo interno
Precondición	El usuario ha ejecutado la aplicación desde el panel de gestión.
Secuencia normal	Si el usuario quiere crear una cuenta vinculada a la reserva: 1. El usuario especifica en el selector crear una nueva cuenta Si el usuario no quiere crear una cuenta vinculada a la reserva: 1. El usuario especifica en el selector no crear ninguna cuenta
Excepción	n/a
Importancia	No vital
Prioridad	Media
Comentarios	n/a

Tabla 3.4: Requisito funcional RF04

Requisito funcional RF05	
Caso de uso	CU05 - Especificar habitación
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir especificar las habitaciones para la reserva.
Actor principal	Equipo interno
Precondición	El usuario ha ejecutado la aplicación desde el panel de gestión.
Secuencia normal	1. El usuario especifica en el selector la habitación
Excepción	La habitación ya se encuentra seleccionada en otro selector.
Importancia	Deseada
Prioridad	Alta
Comentarios	Este requisito se repite según las habitaciones que el usuario desee añadir. Se puede no especificar ninguna habitación, por si el usuario quiere añadir huéspedes a la reserva pero a ninguna habitación en concreto.

Tabla 3.5: Requisito funcional RF05

Requisito funcional RF06	
Caso de uso	CU06 - Crear perfil de huésped
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir crear perfiles de huésped en las habitaciones especificadas.
Actor principal	Equipo interno
Precondición	El usuario ha especificado una habitación.
Secuencia normal	Para cada perfil de huésped que el usuario quiera crear: 1. El usuario introduce los datos personales del perfil 2. El usuario establece unas fechas y horas de *check-in* y *check-out* específicas para el huésped 3. El usuario selecciona el tipo y categoría de perfil 4. El usuario selecciona el tipo y categoría de huésped
Excepción	n/a
Importancia	Deseada
Prioridad	Alta
Comentarios	Este requisito se repite por cada habitación especificada. Es obligatorio añadir un perfil de huésped al especificar una habitación.

Tabla 3.6: Requisito funcional RF06

Requisito funcional RF07	
Caso de uso	CU07 - Crear cuenta asociada a la habitación
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir crear cuentas vinculadas a las habitaciones especificadas.
Actor principal	Equipo interno
Precondición	El usuario ha especificado una habitación.
Secuencia normal	Si el usuario quiere crear una cuenta vinculada a la habitación: 1. El usuario especifica en el selector crear una nueva cuenta Si el usuario no quiere crear una cuenta vinculada a la habitación: 1. El usuario especifica en el selector no crear ninguna cuenta
Excepción	No hay especificada ninguna habitación.
Importancia	No vital
Prioridad	Baja
Comentarios	Este requisito se repite por cada habitación especificada.

Tabla 3.7: Requisito funcional RF07

Requisito funcional RF08	
Caso de uso	CU08 - Crear cuenta asociada al huésped
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir crear cuentas vinculadas a los huéspedes creados.
Actor principal	Equipo interno
Precondición	El usuario ha creado un perfil de huésped.
Secuencia normal	Si el usuario quiere crear una cuenta vinculada al huésped: 1. El usuario especifica en el selector crear una nueva cuenta Si el usuario no quiere crear una cuenta vinculada al huésped: 1. El usuario especifica en el selector no crear ninguna cuenta
Excepción	n/a
Importancia	No vital
Prioridad	Baja
Comentarios	Este requisito se repite por cada huésped creado.

Tabla 3.8: Requisito funcional RF08

Requisito funcional RF09	
Caso de uso	CU09 - Añadir y configurar divisas
Autor	Unai Benajes Esbrí
Descripción	La aplicación debe permitir añadir divisas a las cuentas creadas y la posibilidad de configurarlas como pospago. Alternativamente, si se desean configurar como prepago, la aplicación debe permitir añadirles un saldo inicial.
Actor principal	Equipo interno
Precondición	El usuario ha creado una cuenta.
Secuencia normal	Para cada divisa que el usuario quiera añadir: 1. El usuario selecciona la divisa Si el usuario quiere configurar la divisa como pospago: 2. El usuario marca la opción de pospago Si el usuario quiere configurar la divisa como prepago: 2. El usuario desmarca la opción de pospago 3. El usuario introduce un saldo inicial
Excepción	La divisa ya se ha encuentra seleccionada en otro selector.
Importancia	No vital
Prioridad	Media
Comentarios	Este requisito se repite por cada cuenta creada y asociada a la reserva, a la habitación o al huésped. Es obligatorio especificar la divisa.

Tabla 3.9: Requisito funcional RF09

Capítulo 4

Diseño del sistema

4.1. Diseño de la arquitectura del sistema

El diseño de arquitectura que se ha utilizado en la aplicación se basa en Flux [26], un patrón de arquitectura propuesto por Meta Platforms [47] (anteriormente conocido como Facebook) para el desarrollo de aplicaciones *one-page* [27]. En la Figura 4.1 se muestra el esquema del funcionamiento de Flux.

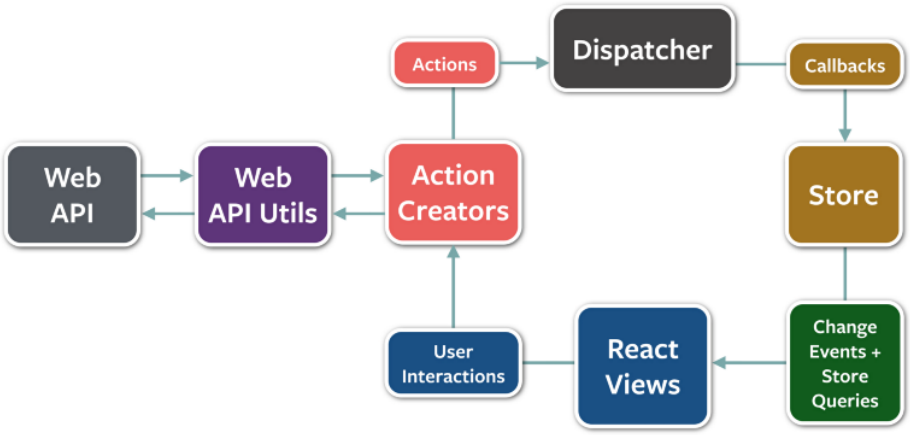

Figura 4.1: Diagrama del patrón de arquitectura Flux [27]

La arquitectura se divide en cinco secciones bien diferenciadas, cada una con una función diferente [27]:

- *Action creators* (creador de acciones): Se encarga de encapsular acciones en funciones. Las acciones son objetos que contienen toda la información necesaria para realizar dicha acción.
- *Dispatcher* (despachador): Se encarga de transmitir las acciones recibidas por el creador de acciones al almacén.

- *Store* (almacén): Se trata de un objeto que gestiona el estado de los datos a través de métodos. El almacén escucha todas las acciones entrantes y decide sobre cuál de ellas actuar. Una vez el almacén ha realizado los cambios de estado, actúa como emisor de eventos, donde emite un evento de cambio a las vistas correspondientes.
- *Views* (vistas): Son los componentes encargados de renderizar las interfaces y de manejar la interacción con el usuario. Pueden conectarse con el almacén y escuchar los eventos emitidos para actualizar su propio contenido. Además, las vistas también pueden transmitir acciones, en respuesta a la interacción del usuario.
- *Web API* (API web): Las llamadas a la API utilizan acciones para actualizar los datos del almacén. Dichas acciones se ejecutan recurriendo a las funciones de la librería Web API Utils.

Además de esta arquitectura, la empresa añade una capa de servicios para la gestión de peticiones a la API. Se muestra el diagrama en la Figura 4.2.

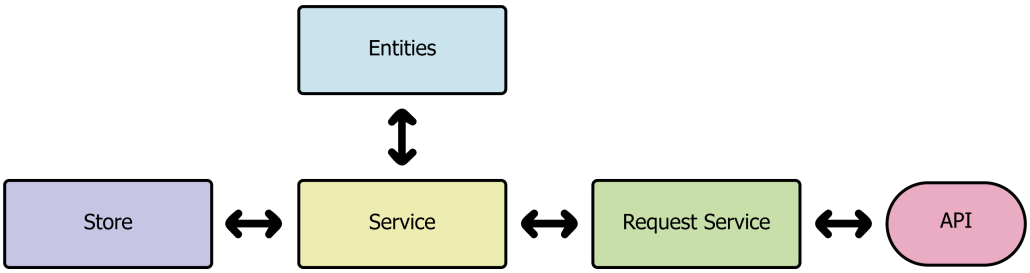

Figura 4.2: Diagrama de la capa de servicios

Para explicar superficialmente el funcionamiento de la capa de servicios, se distinguen tres secciones:

- *Store* (almacén): Desde el almacén de datos se transmiten las acciones para iniciar las llamadas a la API. Dichas acciones ejecutan funciones dentro del paquete de servicio.
- *Service* (servicio): Paquete con los ficheros de servicio, que contienen las funciones correspondientes a los métodos de petición HTTP (Protocolo de Transferencia de Hipertexto, del inglés *Hypertext Transfer Protocol*) [20] utilizables para la aplicación ("GET", "PUT" y "POST"). Existe un fichero por entidad, donde una entidad es una clase de JavaScript que representa un objeto de la base de datos con sus respectivos atributos. Las respuestas en formato JSON (Notación de Objeto de JavaScript, del inglés *JavaScript Object Notation*) [45] de la base de datos se transforman en clases para su cómoda manipulación en las vistas.
- *Request Service* (petición de servicio): Se trata de un fichero genérico que se encarga de construir la petición HTTP y realizar la llamada a la API.

Para implementar esta arquitectura en Vue.js, se ha utilizado la librería Vuex. La arquitectura, y también dicha librería, son las que utiliza la empresa para todos sus proyectos *front-end* desarrollados en este lenguaje, incluyendo la capa de servicios mencionada.

4.2. Definición de *endpoints*

Un *endpoint* de una API es una ubicación digital en la que una API recibe solicitudes sobre un recurso específico en su servidor. Se utilizan métodos de petición HTTP para indicar las acciones sobre este recurso [21][25].

Al no desarrollar la parte *back-end* de la aplicación, no se ha realizado el diseño de la base de datos. Aún así, ha sido necesario seleccionar los *endpoints* requeridos para el proyecto. En la Tabla 4.1 se muestran los *endpoints* definidos para el desarrollo de la aplicación.

Método HTTP	***Endpoint***	**Descripción**
POST	/accounts	Crea la cuenta especificada. Se determina en este *endpoint* si la cuenta se vincula a la reserva.
PUT	/accounts/{accountId}/- -/currencies/{currencyId}	Establece una determinada configuración de divisa para la cuenta proporcionada (prepago o pospago).
POST	/bookings	Crea la reserva especificada.
PUT	/bookings/{bookingId}/- -/rooms/{roomId}/- -/accounts/{accountId}	Vincula la cuenta proporcionada a una habitación reservada.
GET	/booking-types	Devuelve los estados de una reserva.
GET	/currencies	Devuelve todas las divisas.
GET	/facilities	Devuelve todas las instalaciones.
POST	/guests	Crea el huésped especificado. Se determina en este *endpoint* si el huésped dispone de una cuenta vinculada y si se hospeda en alguna habitación.
GET	/guest-categories	Devuelve las categorías de huésped.
GET	/guest-types	Devuelve los tipos de huésped.
POST	/profiles	Crea el perfil especificado.
GET	/profile-categories	Devuelve las categorías de perfil.
GET	/profile-types	Devuelve los tipos de perfil.
GET	/rooms	Devuelve todas las habitaciones.
POST	/transactions	Crea la transacción especificada (para añadir el saldo inicial a las divisas).

Tabla 4.1: *Endpoints* de la API requeridos

4.3. Diseño de la interfaz de usuario

La interfaz de usuario desarrollada para la aplicación cuenta con un diseño *one-page*, estilizada según los estándares de diseño de la empresa. A continuación se muestra la guía de estilo utilizada y su diseño en las herramientas Balsamiq y Figma.

4.3.1. Guía de estilo

La guía de estilo descrita a continuación se ha elaborado en base al estándar de diseño que utiliza la empresa para todos sus proyectos. Comentar que se ha detallado únicamente aquello que se utilizado durante el desarrollo de la aplicación, y no el estándar completo de la empresa.

Colores

Los colores forman una paleta sencilla y minimalista, ideal para un diseño de interfaz *one-page*. Como color primario se ha utilizado un azul oscuro, uno de los tonos principales que representan a Goguest (se puede apreciar en el correspondiente imagotipo de la Figura 1.2). Además del color primario, se ha utilizado una escala de azules grisáceos, empleados para el diseño de las secciones. Y por último, se ha utilizado un color rosa vivo para acentuar los componentes de diseño más importantes. En la Figura 4.3 se muestra la paleta de colores.

Figura 4.3: Paleta de colores

Registro

La forma verbal utilizada principalmente en la aplicación ha sido el infinitivo. Se ha utilizado en su mayoría debido a que la aplicación es un formulario. Algunos ejemplos son: “añadir divisa”, “eliminar huésped” o “crear”.

Cuando no se ha hecho uso del infinitivo, se ha utilizado un registro formal. Principalmente, se ha utilizado al comunicarse con el usuario en la validación del formulario. Algunos ejemplos son: “corrija los errores” o “complete los campos”.

Tipografía

La fuente que se ha utilizado para la tipografía de la aplicación es Roboto, de tipo *sans-serif*, desarrollada por Google [50][16]. Se trata de la fuente por defecto que aplica Vuetify a los encabezados HTML. Además, Vuetify también permite modificar su estilo (grosor, tamaño, espaciado, etc.) con ciertas etiquetas predefinidas. En la Figura 4.4 se muestran los diferentes estilos elegidos para los diferentes componentes de texto empleados en la interfaz.

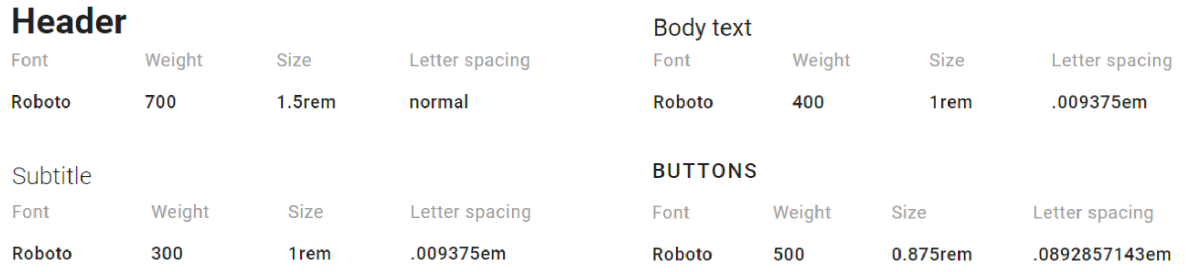

Figura 4.4: Tipografía

Iconografía

La iconografía utilizada en la aplicación proviene de Material Design Icons (MDI por sus siglas), una página web creada y actualizada por la comunidad [24]. Vuetify implementa MDI para aumentar la comodidad a la hora de importar iconos para los componentes, y es por eso que se ha utilizado. En la Figura 4.5 se muestran los iconos utilizados en la aplicación.

Figura 4.5: Iconografía

Componentes de diseño

Vuetify ofrece una gran variedad de componentes de diseño funcionales, con una amplia posibilidad de personalización. En (Vuetify, s.f.) [36], se pueden observar todos los componentes que ofrece Vuetify. Para cada componente, se incluyen una serie de ejemplos para entender su funcionamiento y todas las posibles opciones de personalización que dispone.

En la Figura 4.6 se muestran los principales componentes utilizados en el formulario. Se trata de dos selectores y un campo de entrada de texto. Los dos selectores surgen del componente selector base que ofrece Vuetify, pero como se puede observar en la figura, el segundo ha sido personalizado por completo.

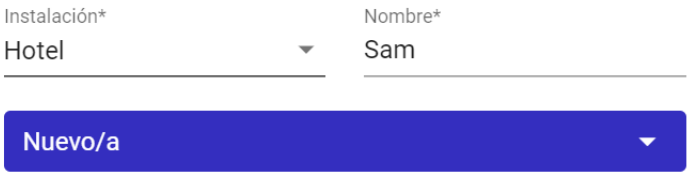

Figura 4.6: Componentes del formulario

En la Figura 4.7 se muestran los selectores de fecha y hora utilizados en el formulario. Son muy parecidos a los que ofrece Vuetify por defecto, aunque han sido ligeramente modificados.

Figura 4.7: Selectores de fecha y hora del formulario

En la Figura 4.8 se muestran los principales botones utilizados en el formulario. Ambos tipos de botones, los redondos y los estándar, los ofrece Vuetify sin ningún tipo de personalización requerida.

Figura 4.8: Botones del formulario

Y por último, en la Figura 4.9 se muestra un ejemplo de desplegable del formulario. Vuetify ofrece una gran comodidad y sencillez para implementar este tipo de componentes desplegables. En el formulario, se ha utilizado este componente para algunas de las secciones, y también para añadir algunos campos secundarios dentro de las mismas.

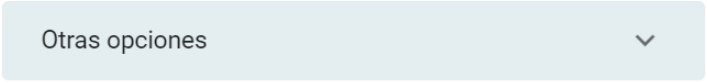

Figura 4.9: Componente desplegable

4.3.2. Diseño de la interfaz

Para el diseño de la interfaz, primero se realizó un primer boceto, y luego, en base a las correcciones, se realizó el diseño final. Se exponen a continuación.

Boceto inicial

En la Figura 4.10 se muestra la parte del boceto correspondiente al módulo de la reserva y en la Figura 4.11 la parte correspondiente al módulo de los huéspedes.

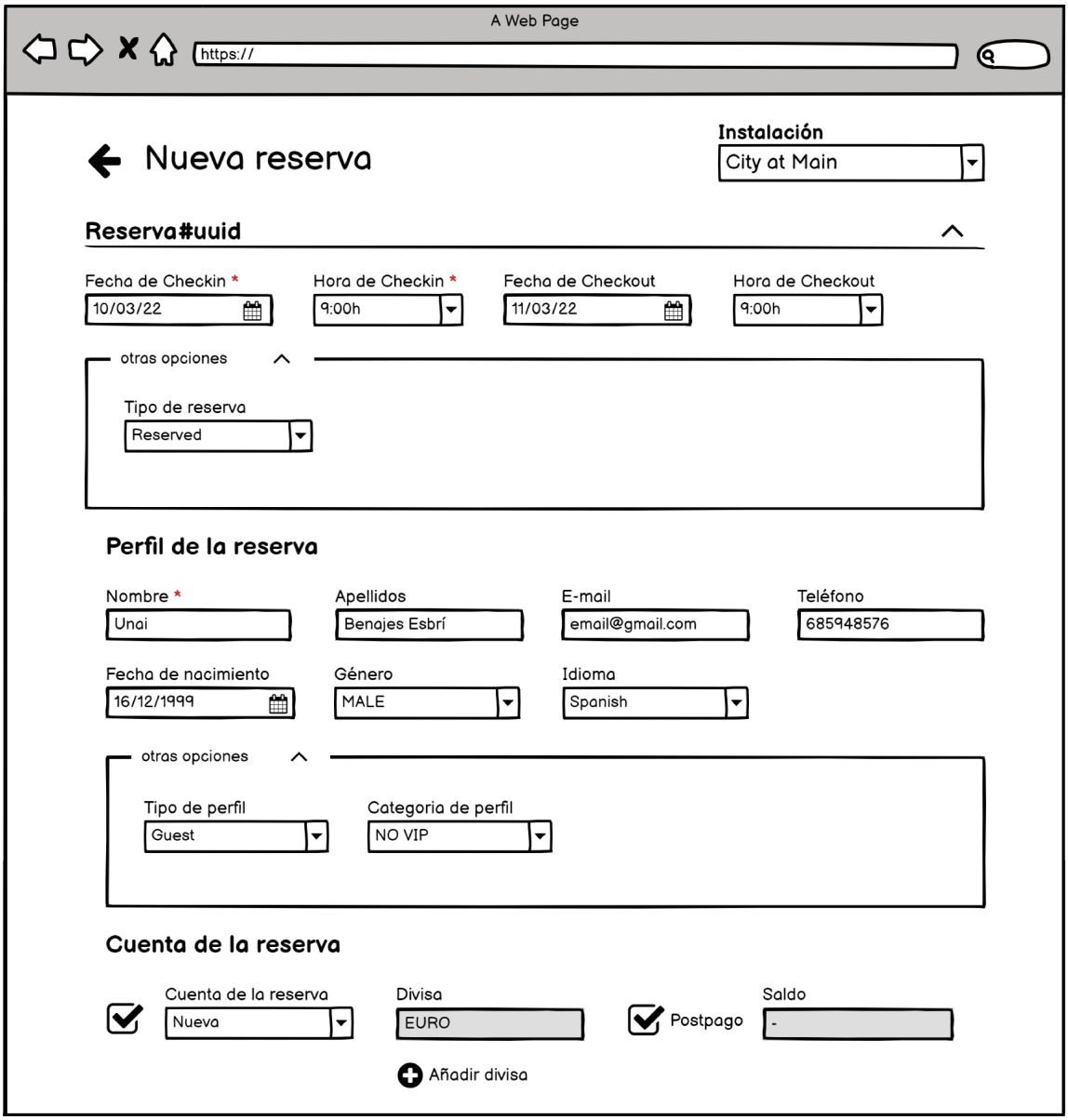

Figura 4.10: Boceto del módulo de la reserva [5]

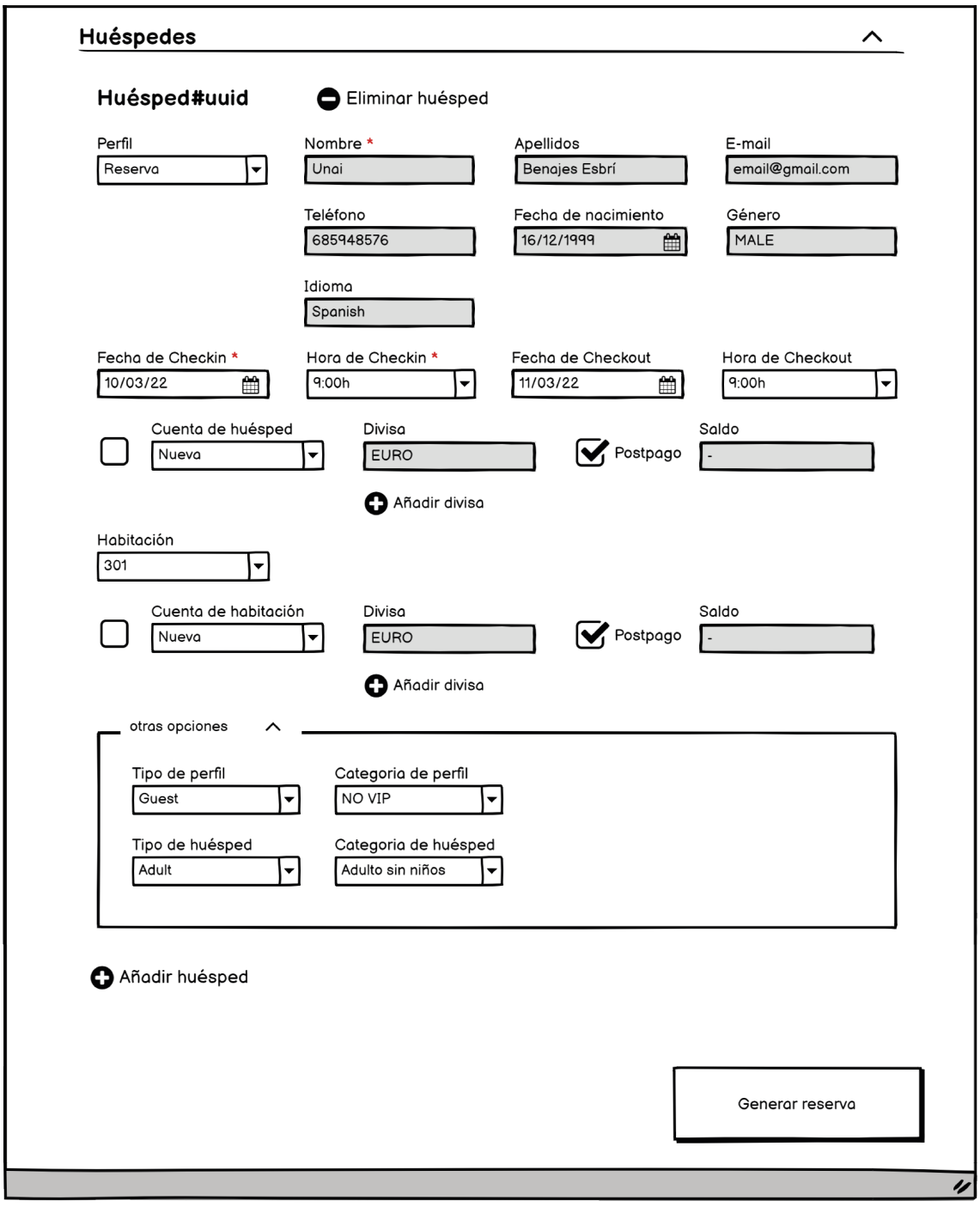

Figura 4.11: Boceto del módulo de los huéspedes [5]

Los bocetos iniciales los realicé en base al alcance y los requisitos establecidos en las fases de inicio y análisis. Principalmente, buscaba diseñar el funcionamiento del formulario, lo esencial a corregir. La parte estética de la página aún no era importante, ya que se regía por los estándares de diseño de la empresa y no necesitaba una corrección por parte de los supervisores. La herramienta que utilicé fue Balsamiq, ideal para priorizar la funcionalidad frente al diseño.

Diseño final

En la Figura 4.12 se muestra el diseño final de la cabecera, en la Figura 4.13 la parte correspondiente al módulo de la reserva y en la Figura 4.14 la parte correspondiente al módulo de los huéspedes.

Figura 4.12: Diseño final de la cabecera [11]

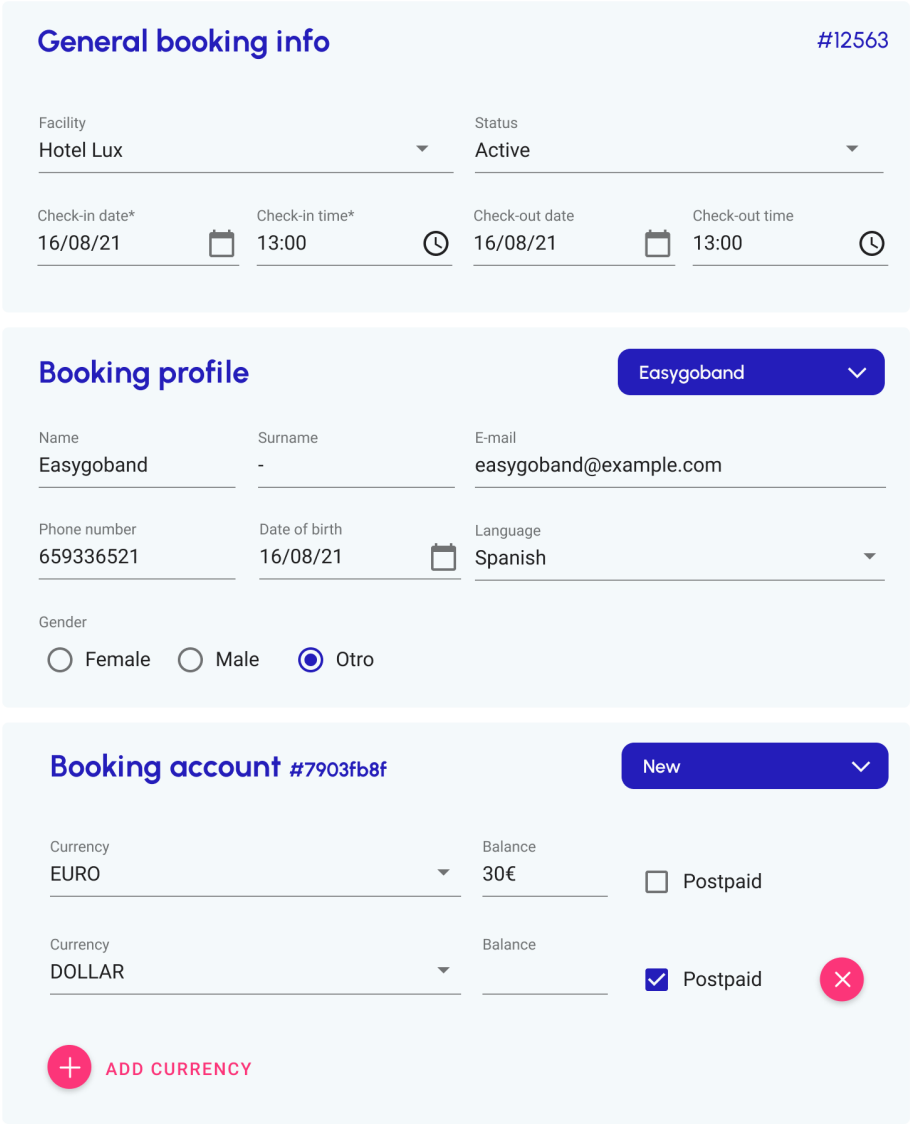

Figura 4.13: Diseño final del módulo de la reserva [11]

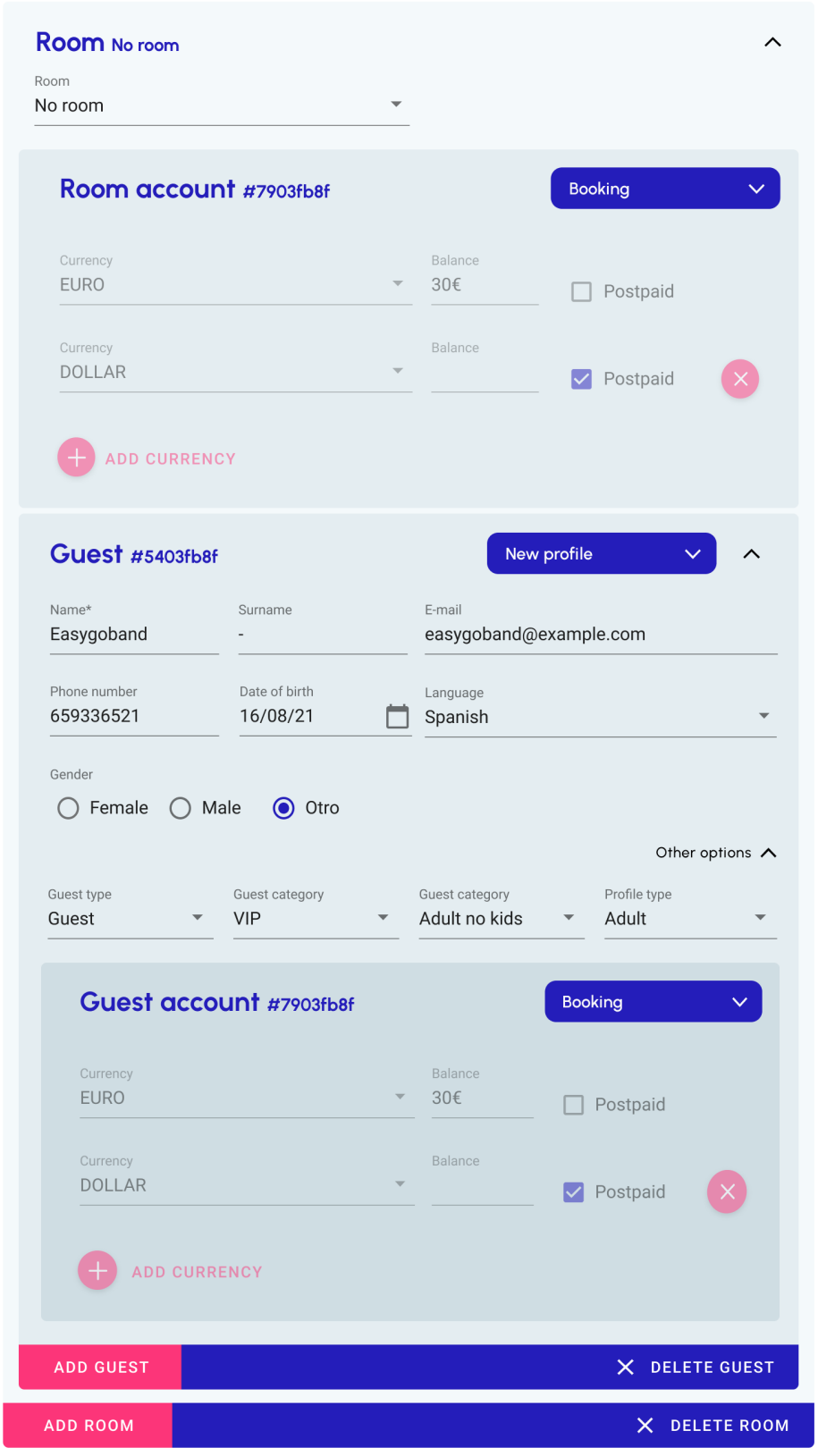

Figura 4.14: Diseño final del módulo de los huéspedes [11]

Para el diseño final, esta vez utilicé Figma, la herramienta que usa la empresa para el diseño de interfaces. Tuve en cuenta las correcciones de los supervisores y los estándares de diseño de la empresa. Comentar que, aunque este diseño tiene una elevada similitud a la interfaz implementada, no es idéntica. Durante el desarrollo de la interfaz, modifiqué algunos aspectos sobre el diseño, sobre todo en el módulo de huéspedes, debido a ciertas complicaciones con la definición de requisitos. Dichas complicaciones están detalladas en el Apartado 2.5.3 de la sección del seguimiento del proyecto.

Capítulo 5

Implementación

5.1. Detalles de implementación

La implementación de este proyecto ha conllevado la escritura de 4.750 líneas de código y la modificación de 70 ficheros distintos. El lenguaje de programación empleado ha sido Vue.js, un *framework* de JavaScript utilizado por la empresa para proyectos *front-end*, acompañado de Vuetify para la implementación de componentes de diseño y usabilidad. El editor de código empleado ha sido Visual Studio Code, debido a la simplicidad y comodidad que ofrece.

El proyecto se ha desarrollado a partir de una plantilla creada por la empresa, conteniendo la configuración básica necesaria para la conexión con la API y el panel de gestión. La estructura de la plantilla está elaborada con el CLI de Vue.js [30], un paquete que permite crear y organizar un proyecto estándar preconfigurado, para no tener que comenzar uno desde cero.

Además, el CLI permite instalar distintos paquetes y configuraciones. Para este proyecto se ha utilizado ESLint [41], una herramienta dedicada al análisis del código capaz de detectar patrones problemáticos o estructuras incorrectas en el lenguaje JavaScript.

A continuación, en la Figura 5.1 se muestra la estructura final del proyecto. Todos los ficheros se han ido creando y modificando a medida que se ha ido desarrollando cada uno de los módulos y secciones del proyecto. Como se puede observar en la figura, el proyecto sigue la arquitectura descrita en la Sección 4.1. Destacan los siguientes paquetes:

- Paquete *"components"*: Contiene los componentes creados y utilizados en la aplicación.
- Paquetes *"config"* y *"locale"*: Contienen la configuración del entorno, las direcciones de la API y los diccionarios de términos (en inglés y en español).
- Paquetes *"entities"* y *"services"*: Forman la capa de servicios. Se encuentran las entidades de la base de datos y su conexión con la API mediante los ficheros de servicio.
- Paquete *"store"*: Corresponde al almacén de datos, seccionado por módulos.
- Paquete *"views"*: Contiene la vista correspondiente a la interfaz *one-page* de la aplicación.

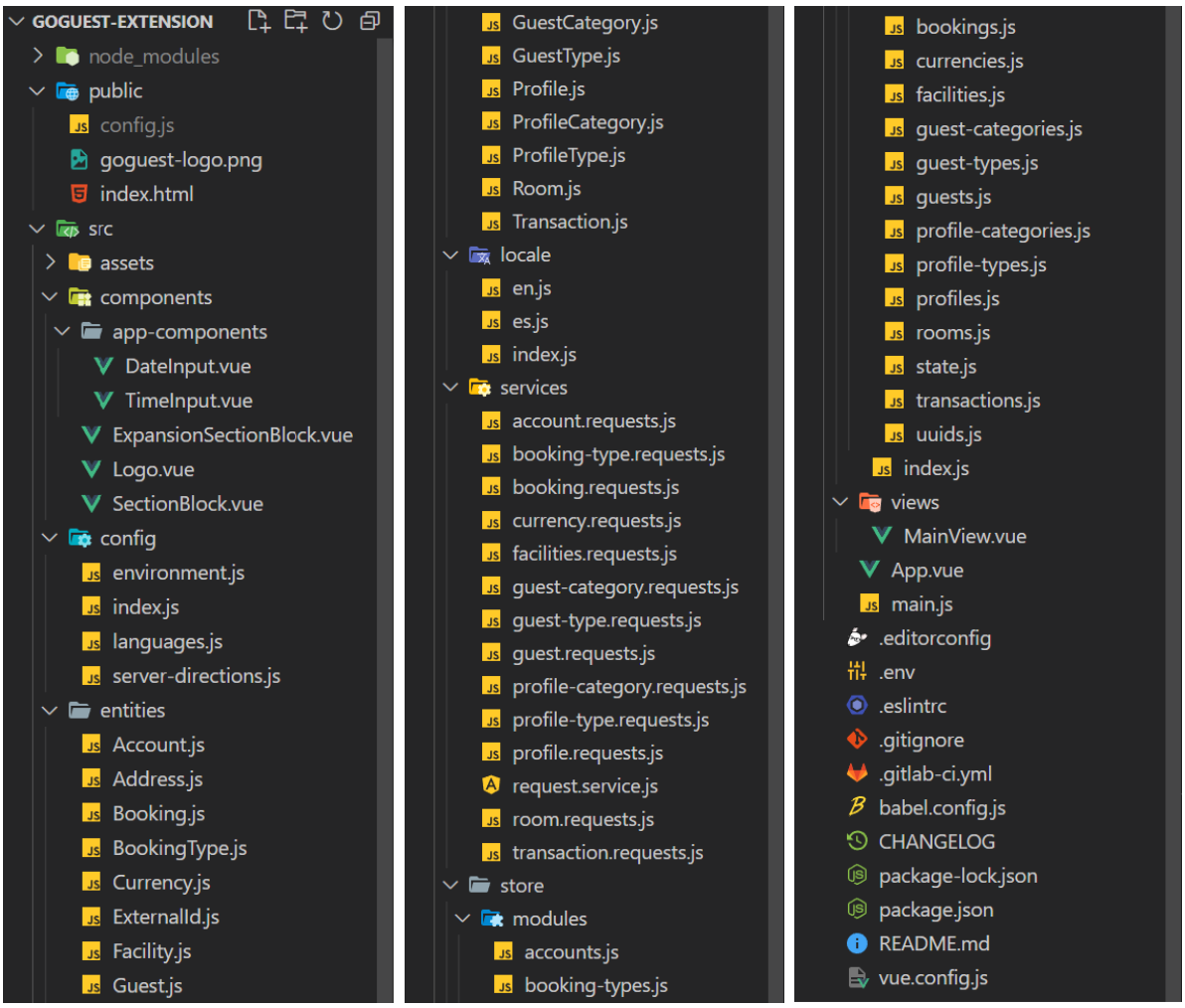

Figura 5.1: Estructura final del proyecto

El fichero *"index.html"* es el archivo raíz de la aplicación, donde está insertado el componente padre *"App.vue"*. Este componente, junto con el fichero JavaScript *"main.js"*, es el encargado de arrancar y montar la aplicación.

Dentro del componente padre *"App.vue"*, está insertada la vista principal *"MainView.vue"*. Esta vista es la encargada de renderizar y manejar todos los cambios de la interfaz, y de transmitir las acciones al almacén según la interacción con el usuario. Por ello, se trata del fichero más importante de toda la aplicación, además de ser el más laborioso y voluminoso de todos.

5.2. Patrones de diseño

A la hora de implementar la aplicación, se ha hecho uso de algunos patrones útiles que proporciona Vue.js. A continuación, se explica cada uno de ellos y el por qué de su uso.

5.2.1. Patrón observador

El patrón observador (*Observer pattern* en inglés) define una dependencia entre objetos, de forma que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes [48]. Vue.js proporciona dos características muy útiles que utilizan este patrón de diseño, y se han hecho uso durante el desarrollo de la aplicación.

La primera característica que ofrece son los ganchos del ciclo de vida de un componente (*component lifecycle hooks* en inglés) [31], utilizados para reaccionar al estado de los componentes. De esta forma, se puede ejecutar cualquier trozo de código deseado dependiendo del estado del componente. Por ejemplo, en el momento que se crea o destruye, o una vez se monta o desmonta. En la Figura 5.2 se muestran todos los ganchos del ciclo de vida de un componente.

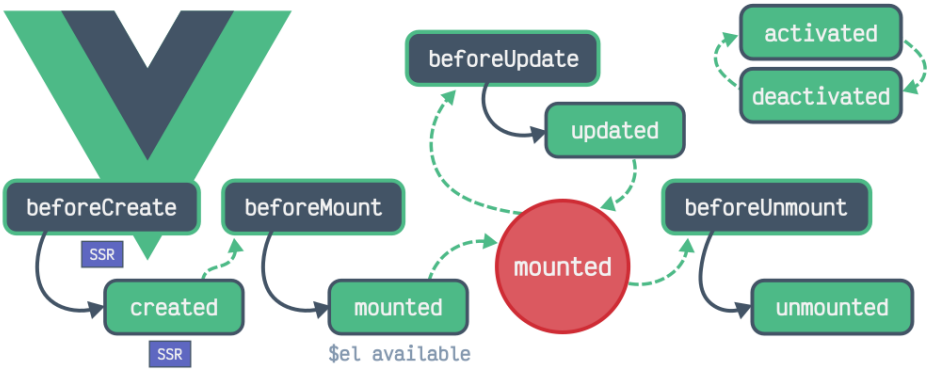

Figura 5.2: Ganchos del ciclo de vida de un componente [23]

La segunda característica que ofrece son los observadores (*watchers* en inglés) [35], utilizados para reaccionar, esta vez, a los componentes de diseño y usabilidad de Vuetify. De esta forma, se puede actuar frente a los cambios en los selectores o en los campos de entrada de texto, y contar con un formulario de datos dinámico.

5.2.2. Patrón de estados

El patrón de diseño de estados (*State pattern* en inglés) se utiliza cuando el comportamiento de un objeto cambia dependiendo del estado del mismo [51]. Vue.js integra este patrón dentro de sus componentes, de forma que cada uno de ellos maneja su propio estado.

Los componentes de Vue.js acceden al almacén cuando requieren de datos compartidos por todos los componentes de la aplicación. Cuando acceden a datos propios, no utilizan el almacén, sino que gestionan su propio estado. Hacen uso de un flujo unidireccional de datos, donde las acciones se transmiten al propio componente, se actualiza su estado y, consecuentemente, la vista que muestra los datos al usuario. En la Figura 5.3 se muestra este flujo de datos unidireccional.

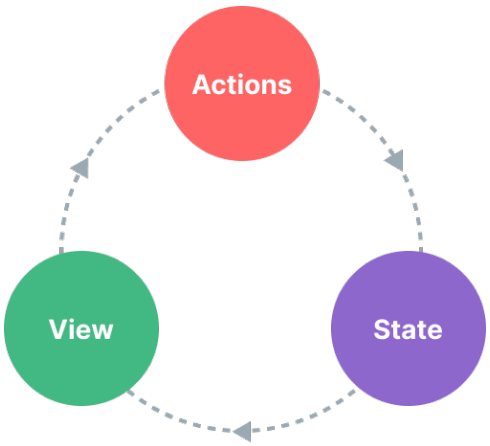

Figura 5.3: Flujo de datos unidireccional dentro de un componente [33]

Este patrón de diseño es la esencia de Vue.js. Permite modificar de manera cómoda y sencilla la vista que proporciona y recibe información del usuario mediante código JavaScript, localizado en el mismo fichero. De esta forma, se facilita mucho la programación que reacciona a los datos introducidos por el usuario.

Durante el desarrollo del proyecto, tuve muy en cuenta este patrón para una buena gestión y organización del código. Creí muy importante distinguir correctamente qué parte del código accede a datos compartidos y qué parte del código gestiona su propio estado.

5.2.3. Patrón de plantilla

El patrón de diseño de plantilla define el esqueleto de un programa, de forma que permite redefinir ciertos aspectos del mismo sin cambiar su estructura [49]. Los componentes de Vue.js son los encargados de suplir esta funcionalidad.

Uno de los propósitos que tienen los componentes de Vue.js, es hacer uso de plantillas genéricas y permitir la reutilización de código. Cada vez que se utiliza un componente se crea una instancia del mismo, por lo que todas las instancias tienen la misma estructura, pero cada una gestiona su propio estado.

Dentro de los componentes, también se puede hacer uso de los *slots* [32] que proporciona Vue.js. Ofrecen a los componentes padre la posibilidad de inyectar una estructura personalizada a la plantilla genérica proporcionada por el componente hijo. De esta forma se permite una personalización total sobre las plantillas utilizadas.

El uso de plantillas en Vue.js me facilitó el desarrollo del proyecto, ya que me permitió crear un componente con una estructura genérica para todas las secciones del formulario. Además, me permitió tener el código mejor organizado, teniendo separada la funcionalidad del componente y la de la aplicación.

5.3. Desarrollo y resultados

A continuación, se muestra el desarrollo completo de la aplicación y los resultados obtenidos. Los apartados se encuentran seccionados en base al alcance organizativo del Apartado 1.4.2, incluyendo otras funcionalidades importantes de la implementación.

5.3.1. Módulo de la reserva

Antes de comenzar con el desarrollo de las funcionalidades principales del módulo de la reserva, tuve que decidir como estructurar las diferentes secciones del formulario. Vuetify ofrece un sistema de *grids* [37] muy cómodo de utilizar, que es el que empleé para organizar tanto las diferentes secciones, como el contenido dentro de cada una de ellas. Además, debido a la gran similitud entre las diversas secciones del formulario, decidí crear un componente que englobara la estructura de todas ellas.

El componente genérico para las secciones corresponde al fichero *"SectionBlock.vue"*. Cuenta con una cabecera, donde se puede añadir un título y un subtítulo en la parte superior izquierda y un componente opcional en la parte superior derecha. Seguidamente, se encuentra el cuerpo de la sección, que se debe implementar utilizando el sistema de *grids* comentado anteriormente. Y por último, el componente también cuenta con un apartado desplegable opcional para otras opciones.

Se trata de un componente muy sencillo, que engloba todas las características del módulo de la reserva. Utilizando este componente genérico, se han implementado las secciones comentadas a continuación.

Sección de información general de la reserva

La implementación final de esta sección se contempla en la Figura 5.4. Como se puede observar, la sección contiene su correspondiente título y un identificador, que en este caso, representa unívocamente a la reserva en su completo. Decidí añadir estos identificadores en el diseño para proporcionarle al equipo interno una forma de localizar las entidades creadas en la base de datos, en caso de requerirlo.

Primeramente, en el cuerpo de la reserva, implementé los selectores de la instalación y del estado de la reserva. Los elementos contenidos en ambos selectores los obtuve realizando las correspondientes llamadas a la API por medio de sus respectivos *endpoints*.

Seguidamente, implementé los selectores de las fechas y horas de *check-in* y *check-out*. Como estos selectores se repiten a lo largo del formulario, decidí crear dos componentes, uno para las fechas y otro para las horas. Además, ambos diseños incluyen iconos en el lateral y cuentan con un menú desplegable personalizado, por lo que decidí que era mejor implementarlos como componentes, y así excluir de la vista principal su implementación. Estos componentes corresponden a los ficheros *"DateInput.vue"* y *"TimeInput.vue"*.

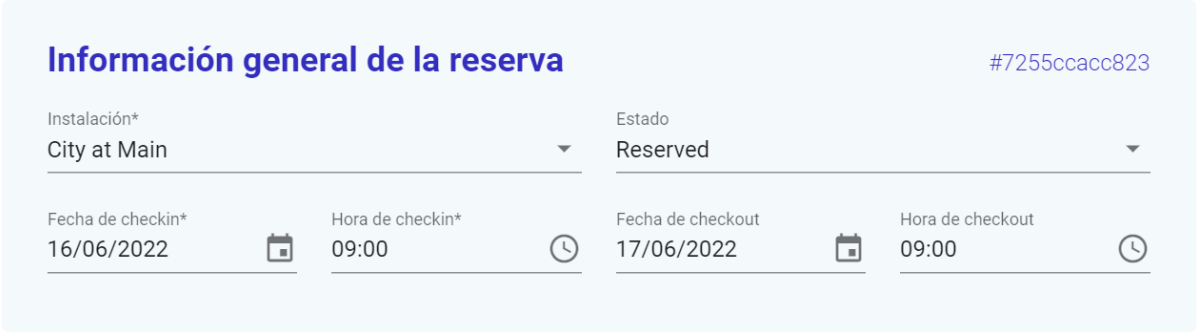

Figura 5.4: Implementación de la sección de información general de la reserva

La única restricción impuesta en esta sección es la obligación de introducir una instalación donde realizar la reserva. No hay ninguna restricción al introducir las fechas y horas de *check-in* y *check-out*, ya que la aplicación es exclusivamente para uso interno, y es posible que se quieran probar algunas configuraciones ficticias o imposibles.

Sección del perfil de la reserva

La implementación final de esta sección se contempla en la Figura 5.5. Como se puede observar, la sección contiene su correspondiente título, un subtítulo con su correspondiente identificador y un desplegable en el lateral. En este caso, el identificador representa unívocamente al perfil de contacto especificado para la reserva.

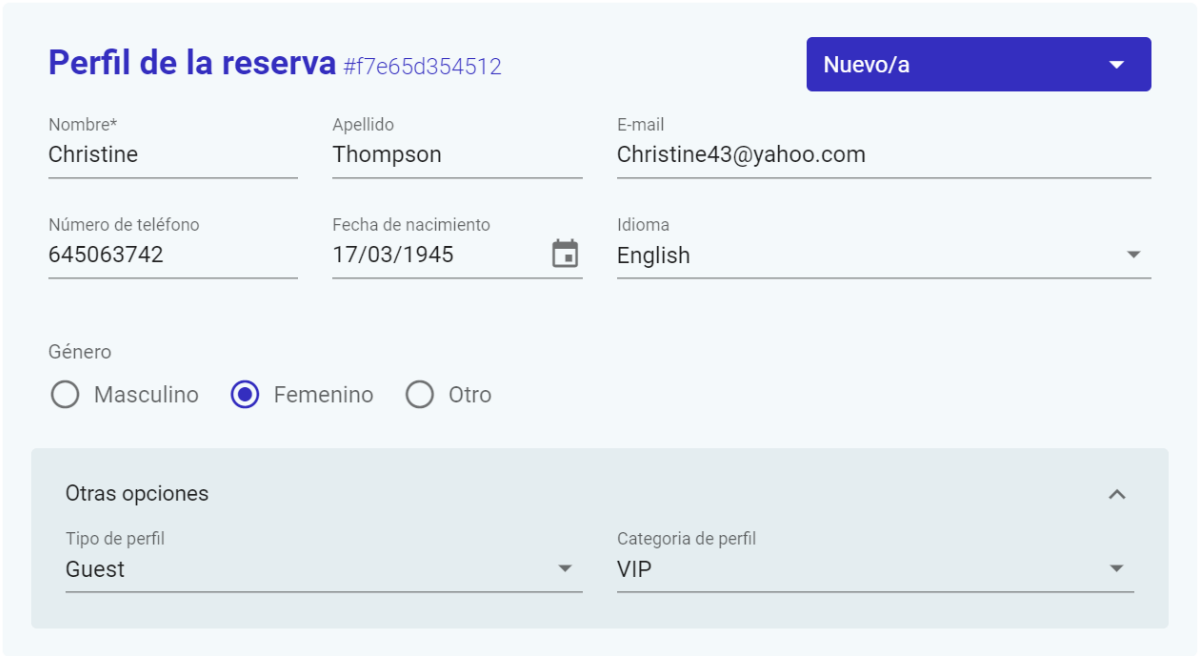

Figura 5.5: Implementación de la sección del perfil de la reserva

En primer lugar, comencé con la implementación de los seis selectores y el botón de opción correspondientes a los datos de un perfil (nombre, apellido, correo electrónico, número de teléfono, fecha de nacimiento, idioma y género). La fecha de nacimiento utiliza el componente para la fecha creado anteriormente, mientras que el resto son simples componentes básicos de Vuetify. Comentar también que los elementos utilizados para el selector del idioma los obtuve del fichero *"languajes.js"*.

En segundo lugar, implementé el componente desplegable de otras opciones incluyendo dos selectores, uno para el tipo de perfil y otro para la categoría de perfil. Esta vez, los elementos contenidos en los selectores los obtuve realizando las correspondientes llamadas a la API por medio de sus respectivos *endpoints*.

Por último, destacar que el selector lateral de la cabecera de la sección solo admite crear un nuevo perfil. Esto se debe a que, para crear una reserva, es obligatorio la creación de un perfil de contacto para la misma. El motivo por el que se ha implementado este selector, es por si en algún momento la empresa decide expandir la funcionalidad de la aplicación, de forma que se puedan especificar perfiles personalizados para la reserva. Este aspecto se explica con detalle en la Sección 7.1.

La única restricción impuesta en esta sección es la obligación de introducir un nombre para el perfil de la reserva. No hay ninguna otra restricción, ya que el resto de campos son todos opcionales.

Sección de la cuenta de la reserva

La implementación final de esta sección se contempla en la Figura 5.6. Como se puede observar, al igual que la sección del perfil de la reserva, esta sección cuenta con su correspondiente título, un subtítulo con su correspondiente identificador y un desplegable en el lateral. En este caso, el identificador representa unívocamente a la cuenta asociada a la reserva.

Figura 5.6: Implementación de la sección de la cuenta de la reserva

Primero, comencé con la implementación del selector para la divisa. Los elementos contenidos en este selector, los obtuve realizando una llamada a la API por medio su respectivo *endpoint*.

Luego, implementé el campo de entrada numérico para el saldo inicial y la *checkbox* para marcar la cuenta como pospago. Hay que tener en cuenta que, si una divisa está marcada como pospago, no tiene sentido añadir un saldo inicial. Alternativamente, si una divisa no está marcada como pospago, significa que está marcada como prepago, y por tanto se debe permitir añadir un saldo inicial. Para lograr este propósito, hice que al marcar la divisa como pospago, se deshabilitara el campo de entrada numérico para el saldo inicial. Esta característica está contemplada también internamente en el código, por lo que en el envío del formulario, no se realiza ninguna transacción del saldo cuando la *checkbox* está marcada. Aún así, es necesario destacar esta característica visualmente para el usuario.

Después, implementé la funcionalidad de añadir y eliminar divisas en la cuenta. Para ello, encapsulé todos los datos respectivos a la divisa en un objeto de JavaScript, para así poder manejar varios de ellos dentro de una colección de objetos. De esta forma, los botones de añadir y eliminar simplemente modifican esta colección, añadiendo o eliminando objetos por índice.

Y por último, implementé el selector lateral de la cabecera de la sección. Al contrario que en la sección del perfil de la reserva, este selector si que cuenta con dos posibles opciones: una para crear la cuenta y otra para no crearla. La opción para no crear la cuenta se incluye debido a que todas las cuentas son opcionales, y por ello se debe permitir su omisión. Esta funcionalidad se contempla internamente, aunque también se aprecia visualmente para el usuario. En la Figura 5.7 se muestra la sección de la cuenta de la reserva deshabilitada.

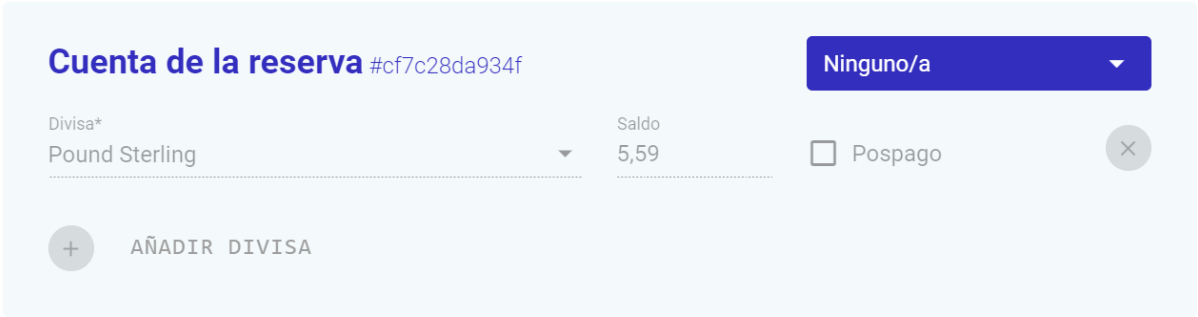

Figura 5.7: Sección de la cuenta de la reserva desactivada

La única restricción impuesta en esta sección es la obligación de introducir una divisa, aunque en este caso en concreto implica tener en cuenta una peculiar característica: evitar que el usuario pueda introducir la misma divisa más de una vez en el listado. Es por ello que implementé un filtro para los elementos del selector. Si algún elemento ya se encuentra en el listado de divisas, simplemente hay que deshabilitarlo, y así evitar que el usuario pueda seleccionarlo en el selector. De esta forma, me aseguré de que el listado tuviera divisas distintas y no provocara ningún fallo en el envío del formulario.

En la Figura 5.8 se muestra un ejemplo de esta característica implementada. En la parte izquierda se muestra el listado de divisas y en la parte derecha el selector para la divisa. Como se puede observar, las divisas añadidas al listado se encuentran deshabilitadas en el selector.

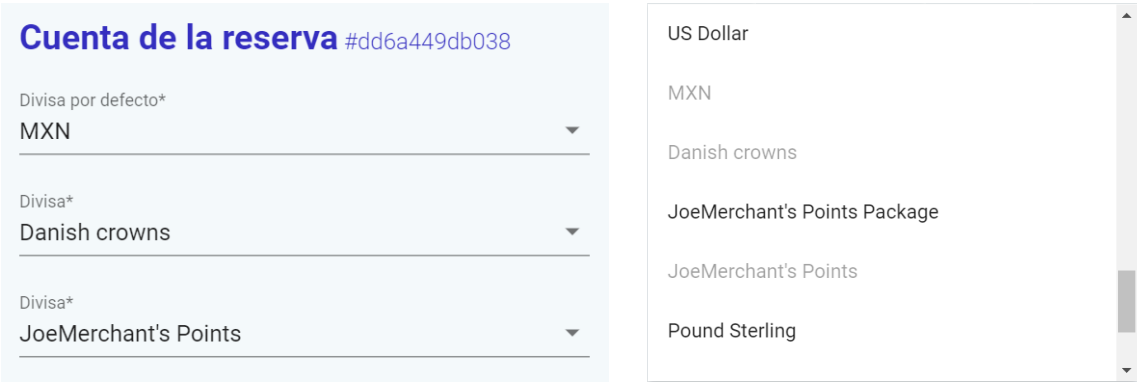

Figura 5.8: Selector de la sección de la cuenta de la reserva con divisas deshabilitadas

5.3.2. Módulo de los huéspedes

Antes de comenzar con el desarrollo de las funcionalidades principales del módulo de los huéspedes, tuve que modificar el componente base para las secciones *"SectionBlock.vue"*. Fue necesario debido a que este módulo cuenta con secciones dentro de otras, por lo que adapté el componente para que funcionara como un subcomponente.

Además, también fue necesaria la creación de un nuevo componente, que corresponde al fichero *"ExpansionSectionBlock.vue"*. Se trata de un componente idéntico al utilizado para el módulo de la reserva, pero en vez de ser fijo, es desplegable. De esta forma, se facilita la visualización de todos los elementos contenidos en un componente, ya que este módulo puede llegar a tener mucha información.

Este componente se ha utilizado para implementar las distintas secciones detalladas a continuación, pero comentar que no solo se ha utilizado este nuevo componente desplegable para este módulo, si no que también se ha utilizado el original.

Sección de la habitación

La implementación final de esta sección se contempla en las Figuras 5.9 y 5.10. Como se puede observar, esta sección cuenta con su correspondiente título y un subtítulo mostrando la habitación seleccionada. Decidí mostrar la habitación como subtítulo debido a que la sección completa es plegable, y de esta forma se puede visualizar cómodamente la habitación que se encuentra seleccionada. En la Figura 5.11 se puede apreciar correctamente.

Para comenzar, implementé el selector de la habitación, el único en toda la sección. Los elementos contenidos en este selector los obtuve realizando una llamada a la API por medio su respectivo *endpoint*. Hay que tener en cuenta que puede haber reservas con huéspedes sin habitación, por lo que decidí tomar la decisión de añadir en el selector una opción simbolizando la ausencia de habitación. De esta forma, una sección que tenga esta opción seleccionada, sirve para añadir huéspedes sin habitación a la reserva. En la Figura 5.11 también se puede apreciar esta característica.

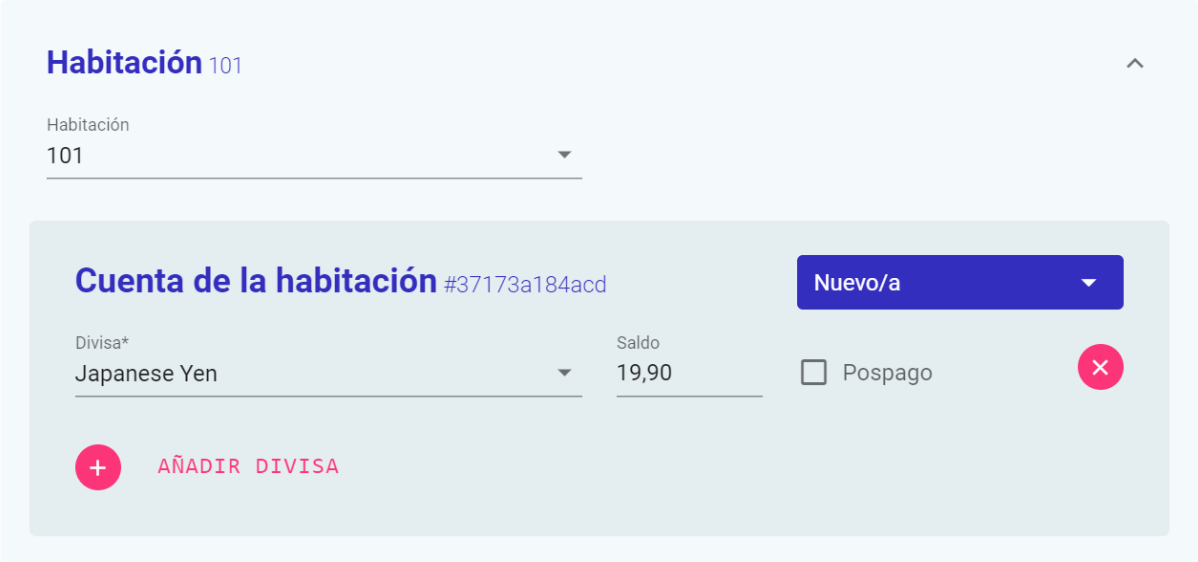

Figura 5.9: Implementación de la sección de la habitación

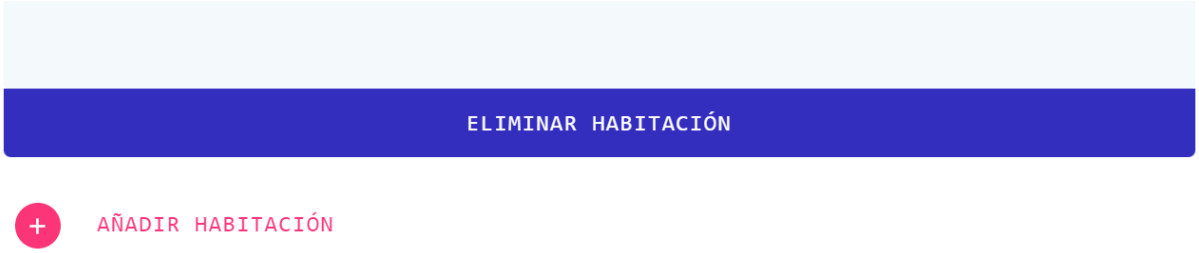

Figura 5.10: Funcionalidad para añadir y eliminar habitaciones

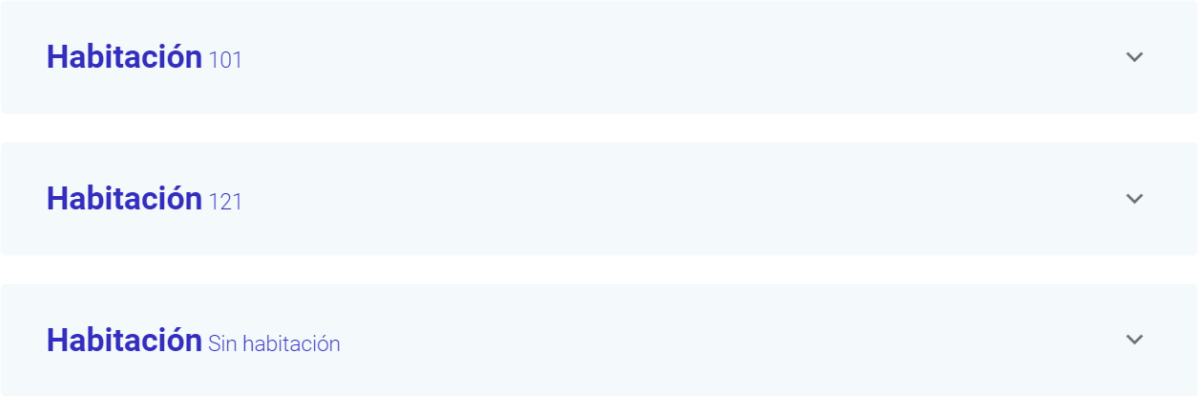

Figura 5.11: Secciones de habitación contraídas

Después, seguí con la implementación de la cuenta de la habitación. Su desarrollo fue idéntico al de la cuenta de la reserva, solo que esta vez la cuenta está asociada a la habitación. Algunos otros cambios superficiales son: su color de fondo y su tamaño, ya que se trata de una subsección.

A continuación, implementé la funcionalidad de añadir y eliminar habitaciones. Al igual que con las divisas de una cuenta, encapsulé todos los datos respectivos a la habitación en un objeto

de JavaScript, para así poder manejar varios de ellos en una colección de objetos. En este caso, en vez de añadir divisas a un listado, se crea una sección por habitación. Sin embargo, al añadir esta funcionalidad fue necesario añadir un filtro para los elementos del selector de la habitación. Al igual que las divisas de una cuenta, no puede haber una habitación seleccionada varias veces, por lo que tuve que implementar esta funcionalidad. De esta forma, cada habitación se trata de una sección diferente con una habitación única.

Y finalmente, implementé una funcionalidad muy importante de esta sección. Para empezar, el *endpoint* utilizado para el selector de las habitaciones devuelve todas las habitaciones del servidor, es decir, todas las habitaciones de todas las instalaciones registradas. Esto ocasiona el siguiente problema: seleccionar una habitación de otra instalación. Es por ello que tuve que implementar otro filtro para seleccionar solo aquellas habitaciones pertenecientes a la instalación introducida en el módulo de la reserva. Además, este filtro debía reaccionar a los cambios realizados en el selector de la instalación, por lo que utilicé un observador, explicado en detalle en el Apartado 5.2.1 de la sección de patrones de diseño. De esta forma, al cambiar de instalación, se actualiza cada uno de los selectores de habitaciones.

Al igual que con la cuenta de la reserva, la única restricción impuesta en esta sección es la obligación de introducir una divisa en los selectores de la cuenta de la habitación.

Sección del huésped

La implementación final de esta sección se contempla en la Figura 5.12. Como se puede observar, esta sección cuenta con su correspondiente título y un subtítulo con su correspondiente identificador. Al igual que la sección de la habitación, esta sección está implementada con el componente desplegable, ya que puede llegar a haber una gran cantidad de huéspedes.

Concretamente, esta sección se trata de una subsección, ya que está contenida dentro de la sección de la habitación, al igual que la cuenta de la habitación. Esta subsección, a su vez, cuenta con más secciones dentro, relacionadas con el huésped correspondiente.

En primer lugar, comencé con la implementación de los seis selectores y el botón de opción correspondientes a los datos del perfil del huésped. Su realización fue idéntica a la del perfil de la reserva, excepto por varias características adicionales. Una de ellas es la opción de añadir unas fechas y horas de *check-in* y *check-out* específicas para el huésped en concreto. De esta forma se puede concretar su estancia si no se adapta a la especificada en la reserva. Y la otra característica es la opción de introducir el tipo y categoría de huésped, también para el huésped en concreto. Al igual que el tipo y categoría de perfil, los añadí en el desplegable de otras opciones. Sus elementos los obtuve realizando las correspondientes llamadas a la API por medio de sus respectivos *endpoints*.

En segundo lugar, continué con la implementación de la subsección para la cuenta del huésped. Se encuentra dentro de la subsección del huésped y su implementación fue idéntica a la de la cuenta de la habitación.

Y para terminar, implementé la funcionalidad de añadir y eliminar huéspedes. Al igual que con la sección de las habitaciones, encapsulé todos los datos respectivos al huésped en

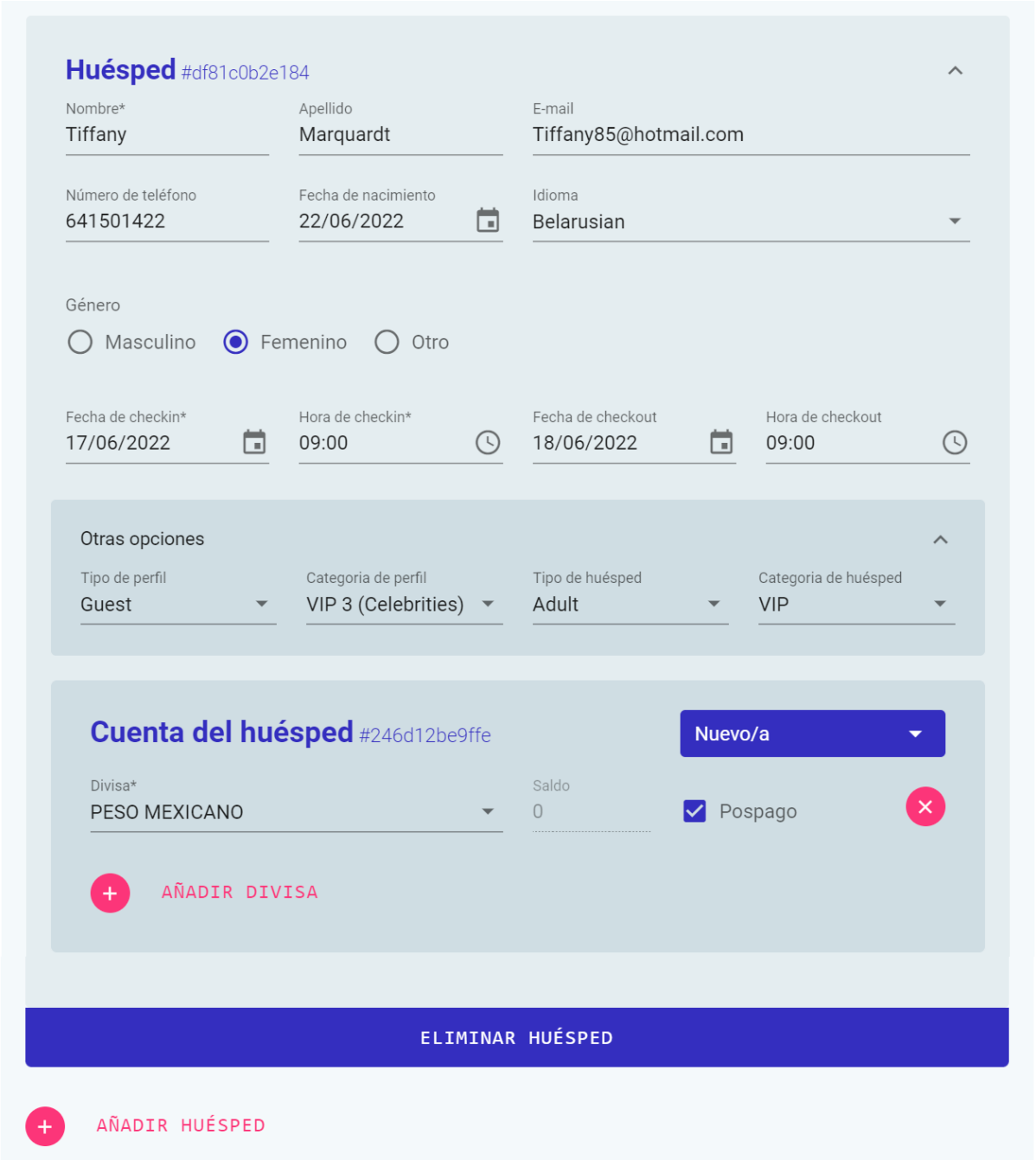

Figura 5.12: Implementación de la sección del huésped

un objeto de JavaScript, para así poder manejar varios de ellos en una colección de objetos. Su funcionalidad es idéntica, solo que con una característica adicional: no poder eliminar un huésped si es el único en la habitación. Lo implementé debido a que el hecho de añadir una habitación solo tiene sentido si hay al menos un huésped en ella. Es por ello que el botón de eliminar huésped se deshabilita si solo hay uno en la sección de la habitación correspondiente.

Para esta sección, que dispone de un perfil y una cuenta para el huésped, la única restricción impuesta es la obligación de introducir el nombre para el perfil y las divisas de los selectores

de la cuenta. También cuenta con la restricción de añadir como mínimo un huésped en una habitación, pero como se ha comentado anteriormente, ya viene impuesta por el diseño del formulario.

5.3.3. Otras funcionalidades

A continuación, se detalla el proceso del envío del formulario y la implementación de datos autocompletados y campos por defecto para el proyecto.

Envío del formulario

Una vez rellenadas todas las secciones del formulario, y el usuario presiona el botón para crear la reserva, se crean las entidades necesarias para enviar las peticiones. A cada entidad creada, se le añaden los datos correspondientes introducidos por el usuario. De esta forma, se tienen todos los datos del formulario organizados en clases de JavaScript y listos para ser enviados por medio de la API. A continuación, se detalla el orden que sigue la aplicación para enviar las peticiones:

1. Se crea el perfil de la reserva.
2. Se crea la cuenta de la reserva y se añaden las divisas correspondientes.
3. Se crea la reserva, con su respectiva información general.
4. Se crea el perfil del huésped.
5. Se crea la cuenta del huésped y se añaden las divisas correspondientes.
6. Se crea el huésped y se asocia a la habitación correspondiente.
7. Se crea la cuenta de la habitación y se añaden las divisas correspondientes.

Una vez enviadas las peticiones, se muestra al usuario un mensaje de éxito indicando que todo ha ido correctamente. Al cabo de cinco segundos, el mensaje junto con la página del formulario se cierran, y se refresca el panel de gestión para que el usuario pueda disponer de la información actualizada. En la Figura 5.13 se muestra el mensaje de éxito mostrado al usuario.

Figura 5.13: Mensaje de éxito en el envío del formulario

En cambio, si el formulario cuenta con algún fallo, se muestra al usuario un mensaje de error indicando que se corrijan los errores. Al igual que el mensaje de éxito, el mensaje de error se cierra al cabo de cinco segundos. Comentar que en vez de cerrarse la página, esta vez se

detallan los errores cometidos en el formulario, para que el usuario pueda localizarlos fácilmente y corregirlos con éxito. En la Figura 5.14 se muestra el mensaje de error mostrado al usuario junto con un ejemplo de campo incorrecto.

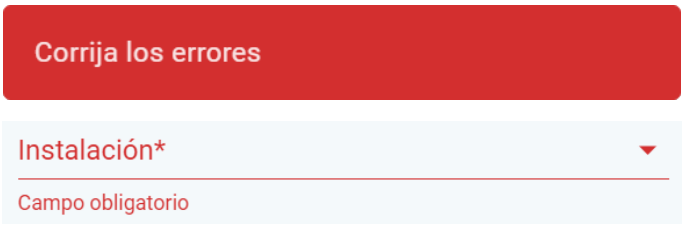

Figura 5.14: Mensaje de error junto a un campo incorrecto en el envío del formulario

Datos autocompletados y campos por defecto

Un objetivo tecnológico clave, definido en la Sección 1.3, es minimizar la interacción con la herramienta. Para ello, añadí datos autocompletados y campos por defecto para agilizar y facilitar la creación de las reservas.

Entre mis supervisores y yo decidimos una configuración estándar para las reservas. Determinamos que al iniciar la aplicación, debía estar configurada ya una reserva con las siguientes características:

- Un día de duración de la estancia.
- Un perfil de contacto.
- Una cuenta asociada a la reserva, con la divisa base de la instalación configurada como pospago.
- Dos huéspedes con su respectivo perfil en una habitación.

Para comenzar, decidí implementar los datos autocompletados de los perfiles, tanto para el de la reserva como para el de los huéspedes. Para ello, utilicé una librería llamada Faker.js [9], que genera datos ficticios para el testeo y el desarrollo de aplicaciones. De esta forma, conseguí que al iniciar la aplicación, o al añadir un nuevo huésped, se completaran los datos de forma automática. Así, el usuario no tiene que perder tiempo en rellenar cada uno de los perfiles, pero puede modificarlos en caso de querer más personalización. Comentar que los únicos datos autocompletados son los personales, y no los selectores de tipo y categoría de perfil o de huésped. Dichos selectores no se autocompletan, ya que no son campos ni importantes ni obligatorios.

Después, implementé los campos por defecto de la sección de la información general de la reserva. Comencé por el selector de la instalación, donde decidí que al iniciar la aplicación, se seleccionara la primera instalación del listado. El estado de la reserva no se trata un campo importante, por lo que no le establecí un valor por defecto. Y por último, implementé el valor por defecto de la duración de la estancia, que como se ha detallado anteriormente, es de un día.

Establecí como fecha y hora de *check-in* el día actual a las nueve de la mañana, y como fecha y hora de *check-out* el día siguiente también a las nueve de la mañana.

Luego, con respecto a las cuentas, establecí que todas estuvieran desactivadas por defecto, excepto la cuenta asociada a la reserva. Esta cuenta, según la configuración estándar comentada anteriormente, debía tener la divisa base de la instalación configurada como pospago. Dicha información la obtuve de la entidad de la instalación, que contiene un atributo con la divisa por defecto que utiliza. Implementé esta característica reaccionando a los cambios del selector de la instalación, donde si cambia la instalación, también cambia la divisa por defecto. Utilicé un observador para su implementación, comentado en detalle en el Apartado 5.2.1 de la sección de patrones de diseño. Además, si el usuario decide modificar los campos de esta divisa, deja de ser la divisa por defecto, y se convierte en un campo normal, que deja de reaccionar a los cambios del selector de la instalación. De esta forma, si el usuario ha cambiado los valores, no se pierden al cambiar de instalación. En la Figura 5.15 se puede observar el campo de la divisa por defecto.

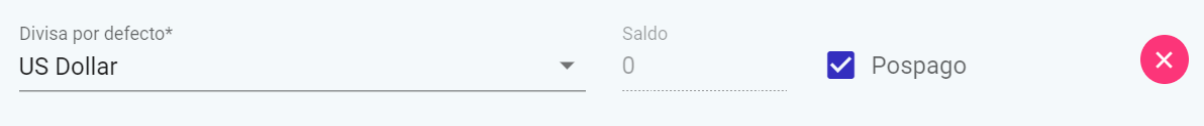

Figura 5.15: Campo de la divisa por defecto en la cuenta de la reserva

Y por último, implementé los campos por defecto de la sección de los huéspedes. Primero, hice que al iniciar la aplicación, se añadiera al formulario la primera habitación disponible de la instalación, y a su vez, dos huéspedes con sus respectivos datos por defecto. Si por algún motivo no hay instalación seleccionada, o la instalación no tiene habitaciones disponibles, la opción por defecto es sin habitación. Además, también establecí que al añadir una habitación nueva, por defecto se escogiera la primera del listado. De esta forma, si al usuario no le importan las habitaciones que escoger, no tiene que perder el tiempo en seleccionarlas.

Capítulo 6

Pruebas e implantación

6.1. Pruebas del sistema

Una vez terminada la parte de la implementación de la aplicación, fueron necesarias una serie de pruebas para comprobar que todo funcionaba correctamente. Se efectuaron, como estaba planificado, pruebas de validación, unitarias y de integración.

Las pruebas de validación que llevé a cabo fueron necesarias para comprobar que la aplicación cumplía con los objetivos y requisitos establecidos para el proyecto. Las realicé junto con mis supervisores, para así poder validar la aplicación correctamente. Por otro lado, las pruebas unitarias y de integración las realicé a mano, ya que se trataba de un proyecto pequeño y manejable. Es por eso que no vi necesaria la implementación de tests para detectar fallos en la aplicación.

Con respecto a las pruebas unitarias, comprobé el funcionamiento completo de cada sección individualmente. De esta forma, me aseguré que las peticiones correspondientes a esa sección en concreto funcionaran correctamente. Además, también comprobé que cada uno de los campos del formulario estuvieran bien implementados, y no admitieran valores erróneos o sin sentido.

Con respecto a las pruebas de integración, comprobé el funcionamiento de la aplicación en su conjunto. Generé reservas probando todas las posibles combinaciones y configuraciones, y me aseguré que estuvieran correctamente creadas.

6.2. Despliegue y mantenimiento

Una vez realizadas las pruebas del sistema, se desplegó la aplicación en el panel de gestión de Goguest y se llevó a cabo el mantenimiento de la herramienta.

Para el despliegue se utilizó GitLab, donde se ejecutó un *script* para subir la herramienta al panel de gestión. Más en detalle, se utilizó Docker [7], una plataforma que utiliza la empresa

para el despliegue de aplicaciones. El equipo interno es el encargado de realizar estas tareas, por lo que simplemente ayudé a que se completara con éxito.

Una vez desplegada la aplicación, me encargué de mantener la aplicación. El equipo interno probó la herramienta generando reservas personalizadas durante un par de días, y dediqué ese tiempo a explicarles el funcionamiento de la herramienta y resolver sus dudas. Comentar que yo mismo también realicé algunas reservas a modo de prueba, para comprobar la herramienta ya desplegada funcionaba correctamente.

Capítulo 7

Conclusiones

7.1. Posibles extensiones del proyecto

El proyecto se ha implementado de forma que se puedan añadir nuevas funcionalidades en un futuro si la empresa lo requiere. Algunas de ellas las había pensado implementar en la fase de mejora del proyecto, pero debido a los inconvenientes ocasionados durante el proyecto (comentados en el Apartado 2.5.2 de la sección del seguimiento del proyecto), no pude llegar a desarrollarlas.

Una mejora del proyecto consiste en poder seleccionar en el perfil de la reserva un perfil de huésped ya creado. De esta forma, un huésped se podría asociar como contacto para la reserva. Ahora mismo, no está implementada esta funcionalidad, y se tendría que realizar manualmente si se requiriera.

Y otra posible mejora consiste en ampliar el formulario. Ahora mismo, están añadidos los campos más importantes para una reserva, pero hay más información que se podría introducir en alguna de las secciones. Por ejemplo, para la sección de los huéspedes, la dirección o el patrón dietético del huésped.

7.2. Conclusiones personales

Como conclusión, comentar que el proyecto ha sido todo un éxito. Se han cumplido todos los objetivos y requisitos establecidos para el proyecto, además en el tiempo planificado. Tanto la empresa como yo estamos muy contentos con los resultados obtenidos.

Respecto al ámbito formativo, he aprendido a utilizar una gran cantidad de tecnologías y herramientas muy útiles que seguro utilizaré en un futuro. La más importante y de la que más he aprendido ha sido Vue.js, un lenguaje que me ha parecido de lo más cómodo e interesante. Con él, he podido conocer más en detalle la parte *front-end* de un proyecto, ya que hasta ahora solo había realizado proyectos mixtos donde únicamente utilizaba HTML y CSS por encima.

Además, se trata de un *framework* muy popular actualmente, y al aprenderlo, he podido apreciar como es realmente la programación de páginas web. También, como añadido, para el desarrollo de este documento he aprendido a utilizar LaTeX[46], que seguro lo usaré en un futuro para el desarrollo de documentos técnicos.

Respecto al ámbito profesional, he conseguido integrarme debidamente en el mundo laboral. Hasta ahora, solo había trabajado con compañeros de clase o con algunos amigos en varios proyectos personales, pero nunca había estado en una empresa de verdad. Por lo general, me ha ayudado a ver cómo es realmente una empresa por dentro, entender su funcionamiento y relacionarme profesionalmente con compañeros de trabajo. Además, me ha servido para saber cómo organizar y planificar un proyecto correctamente, y cumplir con el horario establecido para su desarrollo. Definitivamente, ha sido mi primera experiencia en el mundo laboral, y he tratado de aprovecharla al máximo.

Y respecto al ámbito personal, puedo comentar que me ha resultado una experiencia de lo más divertida. Tenía muchas ganas de poder trabajar en un ambiente laboral y poder conocer este mundo que para mi, hasta ahora, era desconocido. Sobretodo, ha superado mis expectativas, y no ha sido una experiencia para nada aburrida. En todo momento, me he sentido parte de la empresa y he podido participar en actividades y quedadas extralaborales. Además, he conocido gente increíble y talentosa, que me han hecho esforzarme más en mi trabajo y me han dado ganas para seguir aprendiendo y mejorar más aún mis aptitudes.

Mi estancia en la empresa ha sido una experiencia de lo más provechosa, y me alegro de haber podido compartir esta aventura con Easygoband.

Bibliografía

[1] Amazon. *AWS — Cloud Computing - Servicios de informática en la nube*. https://aws.amazon.com/es/. [Consulta: 07 de Junio de 2022].

[2] Amazon. *AWS — Elastic compute cloud (EC2) de capacidad modificable en la nube*. https://aws.amazon.com/es/ec2/. [Consulta: 07 de Junio de 2022].

[3] Atlassian. *Jira — Software de seguimiento de proyectos e incidencias*. https://www.atlassian.com/es/software/jira/. [Consulta: 03 de Junio de 2022].

[4] Atlassian. *¿Qué es un diagrama de Gantt? — Atlassian*. https://www.atlassian.com/es/agile/project-management/gantt-chart. [Consulta: 04 de Junio de 2022].

[5] Balsamiq. *Rapid, Effective and Fun Wireframing Software — Balsamiq*. https://balsamiq.com/. [Consulta: 04 de Junio de 2022].

[6] Debitoor. *Costes indirectos - Qué son los costes indirectos — Debitoor*. https://debitoor.es/glosario/costes-indirectos. [Consulta: 07 de Junio de 2022].

[7] Docker. *Home - Docker*. https://www.docker.com/. [Consulta: 25 de Junio de 2022].

[8] Easygoband World SL. *Easygoband - Plataforma integradora para negocios eficientes*. https://www.easygoband.com/. [Consulta: 03 de Junio de 2022].

[9] Faker.js. *Faker*. https://fakerjs.dev/. [Consulta: 24 de Junio de 2022].

[10] Fernández, Yúbal. *API: qué es y para qué sirve*. https://www.xataka.com/basics/api-que-sirve. [Consulta: 11 de Junio de 2022].

[11] Figma. *Figma: the collaborative interface design tool*. https://www.figma.com/. [Consulta: 04 de Junio de 2022].

[12] Git. *Git*. https://git-scm.com/. [Consulta: 03 de Junio de 2022].

[13] GitLab. *The One DevOps Platform — GitLab*. https://gitlab.com/. [Consulta: 03 de Junio de 2022].

[14] Glassdoor. *Sueldo: Programador (Junio, 2022) — Glassdoor*. https://www.glassdoor.es/Sueldos/programador-sueldo-SRCH_KO0,11.htm. [Consulta: 06 de Junio de 2022].

[15] Glassdoor. *Sueldo: Programador Júnior (Junio, 2022) — Glassdoor*. https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm. [Consulta: 06 de Junio de 2022].

[16] Google Fonts. *Roboto - Google Fonts*. https://fonts.google.com/specimen/Roboto. [Consulta: 12 de Junio de 2022].

[17] Guevara, Jymmy. *Ciclo de Vida Clásico - Análisis y diseño detallado de aplicaciones informáticas de gestión*. https://sites.google.com/site/adai6jfm/ciclo-de-vida-clsico. [Consulta: 04 de Junio de 2022].

[18] Holded. *Holded the Smart Management Software for SMEs*. https://www.holded.com/. [Consulta: 04 de Junio de 2022].

[19] IONOS. *Diagrama de casos de uso: estructura y función - IONOS*. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/. [Consulta: 08 de Junio de 2022].

[20] IONOS. *HTTP Request: guía fácil con ejemplos de sintaxis - IONOS*. https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/http-request/. [Consulta: 11 de Junio de 2022].

[21] Juviler, Jamie. *What Is an API Endpoint? (And Why Are They So Important?)*. https://blog.hubspot.com/website/api-endpoint. [Consulta: 11 de Junio de 2022].

[22] Kashif, Eman. *What are include and extend relationships in a use case diagram?* https://www.educative.io/edpresso/what-are-include-and-extend-relationships-in-a-use-case-diagram. [Consulta: 08 de Junio de 2022].

[23] Manz.dev. *Ciclo de vida (Lifecycle hooks) - Javascript en español - Lenguaje JS*. https://lenguajejs.com/vuejs/componentes/ciclo-vida/. [Consulta: 16 de Junio de 2022].

[24] Material Design Icons. *Material Design Icons*. https://materialdesignicons.com/. [Consulta: 12 de Junio de 2022].

[25] MDN. *Métodos de petición HTTP*. https://developer.mozilla.org/es/docs/Web/HTTP/Methods. [Consulta: 12 de Junio de 2022].

[26] Meta Platforms. *Flux — Flux*. https://facebook.github.io/flux/. [Consulta: 11 de Junio de 2022].

[27] Salcescu, Cristian. *An introduction to the Flux architectural pattern*. https://www.freecodecamp.org/news/an-introduction-to-the-flux-architectural-pattern-674ea74775c9/. [Consulta: 11 de Junio de 2022].

[28] Swagger. *REST API Documentation Tool — Swagger UI*. https://swagger.io/tools/swagger-ui/. [Consulta: 04 de Junio de 2022].

[29] Visual Studio Code. *Visual Studio Code - Code Editing Redefined*. https://code.visualstudio.com/. [Consulta: 03 de Junio de 2022].

[30] Vue.js. *Home — Vue CLI*. https://cli.vuejs.org/. [Consulta: 15 de Junio de 2022].

[31] Vue.js. *Lifecycle Hooks — Vue.js*. https://vuejs.org/guide/essentials/lifecycle.html. [Consulta: 16 de Junio de 2022].

[32] Vue.js. *Slots — Vue.js*. https://es.vuejs.org/v2/guide/components-slots.html. [Consulta: 16 de Junio de 2022].

[33] Vue.js. *State Management — Vue.js*. https://vuejs.org/guide/scaling-up/state-management.html. [Consulta: 16 de Junio de 2022].

[34] Vue.js. *Vue.js - The Progressive JavaScript Framework — Vue.js*. https://vuejs.org/. [Consulta: 03 de Junio de 2022].

[35] Vue.js. *Watchers — Vue.js*. https://vuejs.org/guide/essentials/watchers.html. [Consulta: 16 de Junio de 2022].

[36] Vuetify. *Application service — Vuetify*. https://vuetifyjs.com/en/components/application/. [Consulta: 13 de Junio de 2022].

[37] Vuetify. *Grid system — Vuetify*. https://vuetifyjs.com/en/components/grids/. [Consulta: 17 de Junio de 2022].

[38] Vuetify. *Vuetify — A Material Design Framework for Vue.js*. https://vuetifyjs.com/en/. [Consulta: 03 de Junio de 2022].

[39] Vuex. *What is Vuex? — Vuex*. https://vuex.vuejs.org/. [Consulta: 03 de Junio de 2022].

[40] Wikipedia. *Amazon Web Services - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Amazon_Web_Services. [Consulta: 07 de Junio de 2022].

[41] Wikipedia. *ESLint - Wikipedia*. https://en.wikipedia.org/wiki/ESLint. [Consulta: 15 de Junio de 2022].

[42] Wikipedia. *Estructura de descomposición del trabajo - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Estructura_de_descomposici%C3%B3n_del_trabajo. [Consulta: 04 de Junio de 2022].

[43] Wikipedia. *Front end y back end - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Front_end_y_back_end. [Consulta: 12 de Junio de 2022].

[44] Wikipedia. *Guía de los fundamentos para la dirección de proyectos - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Gu%C3%ADa_de_los_fundamentos_para_la_direcci%C3%B3n_de_proyectos. [Consulta: 04 de Junio de 2022].

[45] Wikipedia. *JSON - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/JSON. [Consulta: 11 de Junio de 2022].

[46] Wikipedia. *LaTeX - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/LaTeX. [Consulta: 25 de Junio de 2022].

[47] Wikipedia. *Meta Platforms - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Meta_Platforms. [Consulta: 11 de Junio de 2022].

[48] Wikipedia. *Observer (patrón de diseño) - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o). [Consulta: 16 de Junio de 2022].

[49] Wikipedia. *Patrón de método de la plantilla - Wikipedia, la enciclopedia libre.* `https://es.wikipedia.org/wiki/Patr%C3%B3n_de_m%C3%A9todo_de_la_plantilla`. [Consulta: 16 de Junio de 2022].

[50] Wikipedia. *Roboto - Wikipedia, la enciclopedia libre.* `https://es.wikipedia.org/wiki/Roboto`. [Consulta: 12 de Junio de 2022].

[51] Wikipedia. *State (patrón de diseño) - Wikipedia, la enciclopedia libre.* `https://es.wikipedia.org/wiki/State_(patr%C3%B3n_de_dise%C3%B1o)`. [Consulta: 16 de Junio de 2022].

[52] Wu, Pine. *Vetur - Visual Studio Marketplace.* `https://marketplace.visualstudio.com/items?itemName=octref.vetur`. [Consulta: 03 de Junio de 2022].