

Space

**Aplicación web para la gestión de espacios y de las solicitudes
para el uso de estos**

Autor: Kepa Sarasola Bengoetxea

Tutor: Jordi Ustrell Garrigos.

Profesor: Joan Soler Adillon

Grado de Multimedia

Ingeniería web

08/06/2020

El presente cuadro de texto tiene solamente finalidades informativas y no tiene que ser incluido en la memoria del estudiante. Asimismo, esta página tampoco tiene que ser incluida.

SOBRE LOS CONTENIDOS DE ESTE DOCUMENTO

Este documento incluye estilos predeterminados de texto, ejemplos de citas bibliográficas, notas a pie de página e inserción de figuras (imágenes y gráficos) y tablas, así como sección de bibliografía e índices automatizados listos para usar.

SOBRE LOS CAPÍTULO DE ESTE DOCUMENTO

Aquellos apartados (i.e. capítulos, apartados, subapartados, etc.) con el título en color negro son obligatorios para todos los TFP, mientras que aquellos en color gris son opcionales, es decir, susceptibles de ser incluidos en la memoria según el tipo de TFP realizado. Es recomendable adaptar el orden de los capítulos a la naturaleza del TFP a realizar, e incluso combinar dos o más capítulos en uno si se considera oportuno.

Tened en cuenta que el número máximo de páginas que puede tener la memoria es 90, incluyendo anexos y bibliografía.

Créditos/Copyright

Una página con la especificación de créditos/copyright para el proyecto (ya sea aplicación por un lado y documentación por el otro, o unificadamente), así como la del uso de marcas, productos o servicios de terceros (incluidos códigos fuente). Si una persona diferente al autor colaboró en el proyecto, tiene que quedar explicitada su identidad y qué hizo.

A continuación se ejemplifica el caso más habitual y una lista de posibles alternativas:



Esta obra está sujeta a una licencia de Reconocimiento- NoComercial-SinObraDerivada

[3.0 España de Creative Commons.](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Aplicación web para la gestión de espacios</i>
Nombre del autor:	<i>Kepa Sarasola Bengoetxea</i>
Nombre del colaborador/a docente :	<i>Jordi Ustrell Garrigos</i>
Nombre del PRA:	<i>Joan Soler Adillon</i>
Fecha de entrega (mm/aaaa):	<i>06/2020</i>
Titulación o programa:	<i>Grado de Multimedia</i>
Área del Trabajo Final:	<i>Ingeniería web</i>
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>Reserva, espacios, digitalización</i>

Resumen del Trabajo:

Este trabajo surge de la necesidad de agilizar las gestiones cotidianas de personas y equipos de personas respecto de la utilización y reserva de locales y espacios públicos o privados pero que pueden ser susceptibles de ser usados por personas y grupos diferentes en el tiempo.

El trabajo se basa en dos principios fundamentales, digitalización del proceso y transparencia en cuanto a la información sobre la disponibilidad de dichos espacios y sobre las solicitudes realizadas por parte de los usuarios.

Se desarrollará un backend en PHP, para lo que se usará el framework Laravel, este backend se encargará de gestionar todas las peticiones recibidas e interactuará con la base de datos. Uno de los objetivos de la aplicación es que su base de información pueda ser usada por otras aplicaciones, por lo que el backend se desarrollará siguiendo el protocolo API Rest.

Se trabajará con metodologías ágiles con el objetivo de implicar al cliente, en este caso el consultor, en la toma de decisiones en todo el proceso de desarrollo del proyecto.

El producto final cumple con los objetivos marcados, de manera que un usuario interesado en usar un espacio determinado puede consultar su disponibilidad, realizar su solicitud y tener el seguimiento de esta solicitud por medios electrónicos.

Abstract:

This work arises from the need to expedite the day-to-day managements of people and teams of people regarding the use and reservation of public or private spaces and spaces but that may be susceptible to be used by different people and groups over time.

The work is based on two fundamental principles, digitalization of the process and transparency regarding to the information on the availability of these spaces and on the status of the requests made by the users.

A PHP backend will be developed, for which the Laravel framework will be used. This backend will be in charge of managing all the requests received and will interact with the database. One of the objectives of the application is that its information base can be used by other applications, so the backend will be developed following the API Rest protocol.

We will work with agile methodologies with the objective of involving the client, in this case the consultant, in making decisions throughout the project development process.

The final product must fulfil the objectives set, so that a user interested in using a specific space could check its availability, make their request and obtain the follow-up of this request electronically.

Dedicatoria/Cita

Dedicado a mi familia.

Mila esker zareten bezalakoak izateagatik!!

Lier, Lore, Nahia, Nerea, Aita eta ama, maite zaituztet!!

Agradecimientos

Quiero mostrar mi agradecimiento a todos aquellos que en este camino me han ayudado, en especial a mi familia y a mis compañeros de Goiener S.Coop.

Notaciones y Convenciones

- Título Principal de sección:
 - Tipo: Arial
 - Tamaño: 20
 - Ejemplo: **Título1**
- Título de subsección:
 - Tipo: Arial
 - Tamaño: 13
 - Ejemplo: **Título subsección**
- Título secundario de subsección:
 - Tipo: Arial
 - Tamaño: 13
 - Ejemplo: **Título secundario**
- Contenido:
 - Tipo: Arial
 - Tamaño 10
 - Ejemplo: Texto del contenido
- Código JSON:
 - Tipo: Courier New
 - Tamaño: 9 pt
 - Ejemplo:

```
{
  "employees": [
    {"firstName": "John", "lastName": "Doe"},
    {"firstName": "Anna", "lastName": "Smith"},
    {"firstName": "Peter", "lastName": "Jones"}
  ]
}
```


Índice

1. Introducción	13
1.1. Introducción/Prefacio.....	13
1.2. Descripción/Definición	15
1.3. Objetivos generales	17
1.3.1. Objetivos principales	17
1.3.2. Objetivos secundarios.....	18
1.4. Metodología y proceso de trabajo.....	19
1.5. Planificación.....	21
1.5.1. Diagramas de Gantr y PERT	22
1.6. Presupuesto	24
1.7. Estructura del resto del documento	25
2. Análisis de mercado	26
2.1. Público objetivo	26
2.2. Competencia/Antecedentes	26
2.2.1. Herramientas no específicas utilizadas para este mismo fin	27
2.2.2. Aplicaciones específicas disponibles en el mercado	29
2.3. Conclusiones del estudio de aplicaciones similares.....	30
2.3.1. Puntos a tener en cuenta	30
2.3.2. Conclusiones.....	30
2.4. Análisis DAFO.....	31
2.4.1. Características internas del producto	31
2.4.2. Características externas	31
3. Propuesta	32
3.1. Definición de especificaciones del producto.....	32
3.1.1. Especificaciones del usuario cliente	32
3.1.2. Especificaciones del usuario gestor.....	32
3.1.3. Especificaciones del usuario ciudadano.....	33
3.2. Modelo de negocio	33
3.2.1. Instalación de instancias de software	33

3.2.2.	Como servicio (SaaS)	34
3.2.3.	Viabilidad económica del proyecto.....	35
4.	Diseño	36
4.1.	Arquitectura general de la aplicación/sistema/servicio	36
4.1.1.	Almacenamiento de la información	36
4.1.2.	Backend.....	38
4.1.3.	Frontend.....	38
4.1.4.	Diagrama y relaciones Frontend-Backend.....	40
4.1.5.	Especificación y lista de peticiones de la API.....	40
4.1.6.	Diagrama general	48
4.1.7.	Casos de uso más comunes.....	49
4.2.	Arquitectura de la información y diagramas de navegación	53
4.2.1.	Arquitectura de la información	53
4.2.2.	Estructura de la información: el caso administrador	53
4.2.3.	Diagrama de navegación de usuario no logueado (ciudadano).....	54
4.2.4.	Diagrama de navegación usuario logueado (administrador)	55
4.3.	Diseño gráfico e interfaces	56
4.3.1.	Estilos.....	56
4.3.2.	Nombre comercial, logotipo y símbolo.....	56
4.3.3.	Paleta de colores	57
4.3.4.	Fuentes	59
4.3.5.	Usabilidad /UX	59
4.4.	Lenguajes de programación y APIs utilizados	60
4.4.1.	Capa de datos: MySQL.....	61
4.4.2.	Capa de lógica de negocio: PHP y Laravel.....	62
4.4.3.	Capa de presentación: Angular	65
5.	Implementación	67
5.1.	Requisitos de instalación	67
5.2.	Instrucciones de instalación.....	68
6.	Demostración.....	69
6.1.	Instrucciones de uso.....	69
6.2.	Prototipos.....	69

6.2.1.	Mock-ups pantalla PC	70
6.2.2.	Mock-up Smartphone.....	76
6.3.	Ejemplos de uso del producto (o guía de usuario)	78
6.3.1.	Ejemplo de realizar una solicitud de una reserva.....	78
6.3.2.	Ejemplo de evaluación una solicitud realizada por un ciudadano	79
7.	Conclusiones y líneas de futuro	80
7.1.	Conclusiones	80
7.2.	Líneas de futuro.....	82
	Bibliografía	84
	Anexos	85

Figuras y tablas

Lista de imágenes, tablas, gráficos, diagramas, etc., numeradas, con títulos y las páginas en las cuales aparecen. Para actualizar cada uno de los índices, hay que hacer botón derecho con el ratón y escoger la opción “Actualizar campos”.

Índice de figuras

Figura 1: Sala de reuniones, una de los posibles espacios susceptibles de ser gestionada por la aplicación.....	13
Figura 2: Imagen del Diagrama de Gantt	22
Figura 3: Imagen del Diagrama PERT	23
Figura 4: Diagrama de clases de la base de datos	37
Figura 5: Diagrama y relaciones Backend-FrontEnd	40
Figura 6: Diagrama general, flujo de la aplicación y responsabilidades	48
Figura 7: Diagrama de navegación para un usuario no logueado, del tipo ciudadano.....	54
Figura 8: Diagrama de navegación, caso de usuario logueado, tipo administrador.....	55
Figura 9: Logotipo y símbolo del proyecto	56
Figura 10: Paleta de colores.....	58
Figura 11: Evolución del uso de diferentes lenguajes (The estate of the Octoverse 2019 - GitHub)	62
Figura 12: Mock-up Home, web publica página de inicio.....	70
Figura 13: Mock-up Listado, web publica página con listado de salas	71
Figura 14: Mock-up Home, web detalle de un espacio	72
Figura 15: Mock-up Admin. Login para los administradores	73
Figura 16: Mock-up Admin. Listado de salas, con opciones de editar y eliminar.....	74
Figura 17: Mock-up Admin. Detalle de una solicitud.....	75
Figura 18: Mock-up Smartphone. Pantallas home y detalle de una sala.....	76

Índice de tablas

Tabla 1: Planificación del proyecto	21
Tabla 2: Presupuesto económico del proyecto	24
Tabla 3: Roles de administración y permisos.....	53
Tabla 4: Uso de las principales librerías JavaScript (GitHub)	65

1.Introducción

1.1. Introducción/Prefacio

Actualmente muchos de los locales y espacios públicos como salas de reuniones, salas de exposiciones o infraestructuras deportivas son gestionados de manera independiente por la entidad más cercana y encargada de su gestión, por ejemplo: casas de cultura, centros cívicos o polideportivos. En muchas ocasiones la gestión de las solicitudes es manual y exige realizarlas de manera presencial.

En la mayoría de los casos no existe la opción de ventanilla única, es decir, la posibilidad de realizar una solicitud para la utilización de un espacio/local mediante una única interface o puerta de entrada para al ciudadano.



Figura 1: Sala de reuniones, una de los posibles espacios susceptibles de ser gestionada por la aplicación

Este proyecto ofrece tanto al ciudadano como a la entidad gestora de estos espacios, la oportunidad de gestionar las solicitudes y las reservas de una manera digital, poniendo en valor las ventajas de los procesos online en cuanto a la accesibilidad de los procesos y de la información.

El finalidad de este proyecto es el de realizar un proceso de digitalización de la necesidad descrita, con el objetivo de proponer un sistema de tratamiento de la información transparente y en el que se asegura la integridad y veracidad de los datos recogidos y ofrecidos por esta.

El producto final ofrece la posibilidad de que un usuario interesado en usar un espacio determinado pueda consultar su disponibilidad, realizar su solicitud y obtener el seguimiento de esta solicitud por medios electrónicos.

Por otra parte, en cuanto a las entidades gestoras de estos espacios, permite una gestión descentralizada de las solicitudes recibidas, pudiendo nombrar uno o más usuarios-administradores que se encargarán de valorar y en su caso admitir o denegar dichas solicitudes

1.2. Descripción/Definición

El uso de las nuevas tecnologías para facilitar las gestiones cotidianas es cada vez mayor en la actualidad, cada vez más operaciones como las operaciones comerciales, bancarias, etc. se realizan mediante aplicaciones web o aplicaciones móviles. Estos avances tecnológicos posibilitan la realización de dichas gestiones de una manera mucho más fácil y eficaz, tanto desde el punto de vista del usuario como desde el punto de vista de quien tiene que administrar estas.

Este proyecto se centra en la gestión de los espacios públicos, tales como salas de reuniones, salas de exposiciones, infraestructuras deportivas, etc. en las que es necesario gestionar diferentes actividades o usos en el tiempo de dichas instalaciones. Lo más habitual en estos casos es gestionar estas infraestructuras mediante solicitudes en las que se comunica además de la razón de la solicitud, las fechas y horarios concretos en los que se desea usarlas, para lo cual es de máxima importancia conocer la disponibilidad de los recursos que se quieren utilizar.

Actualmente en la mayoría de los casos, estas solicitudes han de realizarse de manera presencial o por email, según las opciones que ofrezca el centro o entidad que gestione el espacio en cuestión. Estos sistemas presentan varios problemas, los más evidentes son observables en el caso de que la solicitud tenga que realizarse de manera presencial, pero en todos los casos uno de los problemas con mayor dificultad de solución es la de tener que realizar la solicitud 'a ciegas', es decir, sin tener ninguna constancia de que el espacio del que se quiere disponer esté libre para las fechas y horarios en los que el solicitante necesite de estos.

El sistema que se presenta en esta propuesta aborda el problema desde el punto de vista de la gestión transparente y on-line de la información, ofreciendo al solicitante una plataforma en la que poder consultar la disponibilidad de diferentes espacios que gestione la entidad encargada. Una vez realizada la consulta, el usuario dispondrá de un sistema de solicitud de estos espacios de manera que quede constancia de la solicitud realizada y de la respuesta remitida por la entidad encargada de la gestión del espacio.

En definitiva lo que pretende la propuesta es ofrecer información acerca de los diferentes espacios y su disponibilidad, esta información se ofrecerá calendarizada, para facilitar la comprensión y la transmisión de la información, cumpliendo así con uno de los objetivos del proyecto, la gestión transparente. Además, la gestión de las solicitudes permitirá un seguimiento de estas, en principio mediante mensajes de correo electrónico (aunque la propuesta queda abierta en ese aspecto al uso de mensajes SMS, Telegram¹, etc..., para el envío de dichas notificaciones).

¹ Telegram (<https://es.wikipedia.org/wiki/Telegram>) Telegram es una plataforma de mensajería y VOIP desarrollada por los hermanos Nikolái y Pável Dúrov. La aplicación está enfocada en la mensajería instantánea, el envío de varios archivos y la comunicación en masa.

El resultado es una aplicación web, con un diseño 'responsive'² y por lo tanto accesible desde diferentes dispositivos, mediante la cual el ciudadano tiene la opción de:

- Realizar solicitudes de uso de espacios o locales, mediante internet, sin las limitaciones espacio/tiempo que puede suponer el método presencial.
- Recibir información de manera telemática sobre el estado de su solicitud.
- Consultar información concreta sobre cada espacio (aforo, m2, ubicación, mobiliario disponible, fotografías, etc...)
- Consultar la disponibilidad de estos espacios.
- Disponer de toda la información centralizada, pudiendo consultar un calendario de reservas (ya confirmadas) de los espacios, obteniendo información sobre la disponibilidad de estos en tiempo real, ya que esta se actualizará según se vayan gestionando las solicitudes recibidas.

En cuanto a los gestores que utilizarán esta herramienta para la gestión del uso de los mismos, cabe destacar que la aplicación permitirá la gestión descentralizada de estos espacios, pudiendo asignar la gestión de las solicitudes a diferentes administradores. Esta asignación vendrá dada por el espacio en cuestión.

Como base técnica para la consecución de estos objetivos y funcionalidades, se utilizará una arquitectura de tres capas:

1. Una capa de datos para la persistencia de estos para la cual se utilizará el sistema de gestión de base de datos MySQL,
2. Una capa de lógica de negocio o backend, que se desarrollará en PHP y para la que se usará el framework Laravel. Este backend se encargará de gestionar todas las peticiones recibidas e interactuará con la base de datos.
3. Una capa de presentación o frontend, que se desarrollará en JavaScript mediante el framework Angular.

Cabe destacar que el desarrollo de este proyecto tendrá como resultado además de un producto web una API mediante la cual se podrán realizar las diferentes acciones previstas en el proyecto, con lo que la información contenida en el proyecto podrá ser compartida por otras aplicaciones web o móviles, mediante un protocolo API-Rest³

² Diseño responsive (https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable) es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas.

³ Rest(https://es.wikipedia.org/wiki/Transferencia_de_estado_representacional) es un estilo de arquitectura software para sistemas hipermedia distribuidos como la WWW. Se originó en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

1.3. Objetivos generales

Los objetivos generales que persigue este proyecto van de la mano de la gestión colaborativa y transparente, para lo cual utiliza tecnologías accesibles para todo tipo de público. Este es el listado de los objetivos por orden de relevancia:

- Poner a disposición del ciudadano información acerca de la disponibilidad de diferentes espacios públicos, con información real y actualizada.
- Facilitar al ciudadano la comunicación con los gestores de estos espacios para la solicitud de estos.
- Asegurar al ciudadano la información acerca de la solicitud realizada.
- Facilitar la gestión compartida de las solicitudes recibidas a los gestores de los espacios.
- Asegurar que tanto la información que ofrece la aplicación como las formas de interactuar con ella cumplen los requisitos de usabilidad⁴.
- Asegurar que la información que maneja la aplicación pueda ser compartida por otras aplicaciones que puedan ser interesantes para las entidades que gestionan dichos espacios

1.3.1. Objetivos principales

Objetivos de la aplicación:

- Poner a disposición del ciudadano información acerca de la disponibilidad de diferentes espacios públicos, con información real y actualizada.
- Asegurar que tanto la información que ofrece la aplicación como las formas de interactuar con ella cumplen los requisitos de usabilidad.
- Asegurar que la información que maneja la aplicación pueda ser compartida por otras aplicaciones que puedan ser interesantes para las entidades que gestionan dichos espacios

Objetivos para el cliente usuario:

- Facilitar al ciudadano la comunicación con los gestores de estos espacios para poder realizar la solicitud de estos y su posterior seguimiento
- Asegurar al ciudadano la información acerca de la solicitud realizada.
- Facilitar la gestión compartida de las solicitudes recibidas a los gestores de los espacios.

⁴ Requisitos de usabilidad (<https://es.wikipedia.org/wiki/Usabilidad>) se refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. La usabilidad también puede referirse al estudio de los principios que hay tras la eficacia percibida de un objeto.

Objetivos personales del autor del TF:

- Realizar una aplicación que facilite tanto el acceso como la gestión de la información de una manera transparente.
- Que esta aplicación sea capaz de ser accesible, usable y que respete los derechos del usuario en cuanto a la protección de datos.

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Facilitar gestiones que normalmente suelen ser farragosas, ya que pueden suponer, desplazamientos por parte del usuario, el uso del papel en la mayoría de los casos, dificultades para conocer el estado de la solicitud
- El tratamiento transparente de la información tanto en cuanto la información que tratará la aplicación es pública y no ofrecerá datos personales a otras aplicaciones que puedan utilizar datos de esta aplicación usando su API
- El proyecto trabaja con el objetivo de que la aplicación sea transparente tanto para el ciudadano como para el gestor de la aplicación.

1.4. Metodología y proceso de trabajo

La metodología utilizada para la realización del TFG está totalmente relacionada con la caracterización del propio proyecto. Este proyecto está básicamente orientado al usuario y cubre una necesidad concreta que es la de acceder a información previamente gestionada por los administradores de los diferentes espacios para poder evaluar si cabe o no realizar la solicitud para el uso de un espacio concreto. Una vez realizada la solicitud la información clave para el usuario es la del estado de esta solicitud, es decir poder hacer el seguimiento de la solicitud realizada mediante medios electrónicos.

Se ha realizado un análisis previo a la presentación de la idea en el que se ha evaluado la posibilidad de mejorar o adaptar un producto ya creado anteriormente pero no se ha encontrado software libre que permita estudiar el código y modificarlo para adecuarlo al proyecto que nos ocupa, por lo que la opción elegida ha sido la de desarrollar un nuevo proyecto.

Para el desarrollo de este proyecto la opción técnicamente más interesante es la de trabajar mediante en dos planos, backend y frontend:

Backend:

La necesidad de tener una fuente de información que deba de ser compartida justifica la necesidad de trabajar con una API, que en su formato más estándar debería de cumplir la especificación APIRest. Esta necesidad se sustenta en uno de los objetivos principales de la aplicación. "Asegurar que la información que maneja la aplicación pueda ser compartida por otras aplicaciones que puedan ser interesantes para las entidades que gestionan dichos espacios".

Este objetivo marca claramente el camino a seguir, teniendo pues que construir una aplicación que nos permita compartir la información que gestiona la aplicación con otras aplicaciones. De esta manera el proyecto plantea un backend que gestionará la información recibida por parte del usuario y que devolverá información a la propia aplicación web o a otras aplicaciones que necesiten de esta información y que estén autorizadas para ello.

Frontend:

Una vez tomada la decisión de la creación de una API, la parte que usará el usuario se plantea como una aplicación de frontend que tratará los inputs necesarios para pedir información o realizar operaciones mediante la aplicación de backend y devolverá al usuario la información gestionada por esta. Para ello se desarrollará un interface de comunicación con el usuario que cumplirá el siguiente objetivo "Que esta aplicación sea capaz de ser accesible, usable y que respete los derechos del usuario en cuanto a la protección de datos."

Metodología:

En cuanto a la metodología de desarrollo, se seguirá una metodología ágil, tal y como está definida en Wikipedia: “El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.”⁵

Esta metodología se considera la más apropiada para el desarrollo del proyecto, teniendo en cuenta las siguientes necesidades:

1. Se dispone de una primera idea para el desarrollo del proyecto, desde la cual se obtiene una base de producto que mediante la interacción con el consultor encargado de la asignatura podrá y deberá de ser ampliada hasta que se pueda obtener una aplicación acorde con la exigencia de la asignatura.
2. La entrega de partes funcionales del proyecto permitirá la evaluación de estos, la corrección de errores y la aportación de nuevas propuestas por parte del consultor.
3. Permite a partir de un producto básico ir añadiendo nuevas funcionalidades a este, asegurando el correcto funcionamiento de las soluciones entregadas anteriormente y añadiendo así valor al producto.

⁵ Desarrollo ágil de software según Wikipedia
(https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software)

1.5. Planificación

La planificación del proyecto contempla la ejecución del mismo desde el 24 de febrero hasta el 8 de junio de 2019. Estas son las fechas clave en el desarrollo del proyecto.

Nombre	Duración	Inicio	Final
PEC1. Propuesta del proyecto	4 días	19/02/2019	23/02/2019
PEC2. Definición formal del proyecto. Cap 1	14 días	24/02/2019	08/03/2019
Primer Informe del trabajo	7 días	24/02/2019	02/03/2019
Contraste con el consultor y mejoras al informe	7 días	02/03/2019	08/03/2019
Estudio de las tecnologías a usar, backend	7 días	02/03/2019	08/03/2019
PEC3. Definición	28 días	09/03/2019	05/04/2019
Primer Informe de trabajo, Capítulos 2,3,4 y 6.2	18 días	09/03/2019	26/03/2019
Contraste con el consultor y mejoras al informe	10 días	27/03/2019	05/04/2019
Análisis de requisitos	7 días	09/03/2019	15/03/2019
Creación de la base de datos	2 días	15/03/2019	17/03/2019
Diseño del backend	5 días	18/03/2019	22/03/2019
Desarrollo del primer backend	15 días	22/03/2019	05/04/2019
Realización de pruebas	3 días	02/04/2019	05/04/2019
PEC4. Desarrollo del frontend y Beta	35 días	06/04/2019	10/05/2019
Completar Informe de trabajo	35 días	06/04/2019	10/05/2019
Diseño de interfaces	7 días	06/04/2019	12/04/2019
Creación del entorno Frontend	7 días	13/04/2019	19/04/2019
Desarrollo del FronEnd	21 días	20/04/2019	10/04/2019
Pruebas FrontEnd	3 días	07/04/2019	10/04/2019
PEC5. Entrega final	21 días	11/05/2019	08/06/2019
Completar Informe de trabajo	21 días	11/05/2019	08/06/2019
Finalización del trabajo	21 días	11/05/2019	08/06/2019
Ajustes y pruebas	21 días	11/05/2019	08/06/2019
Defensa virtual	5 días	15/06/2019	19/06/2019

Tabla 1: Planificación del proyecto

1.5.1. Diagramas de Gantr y PERT

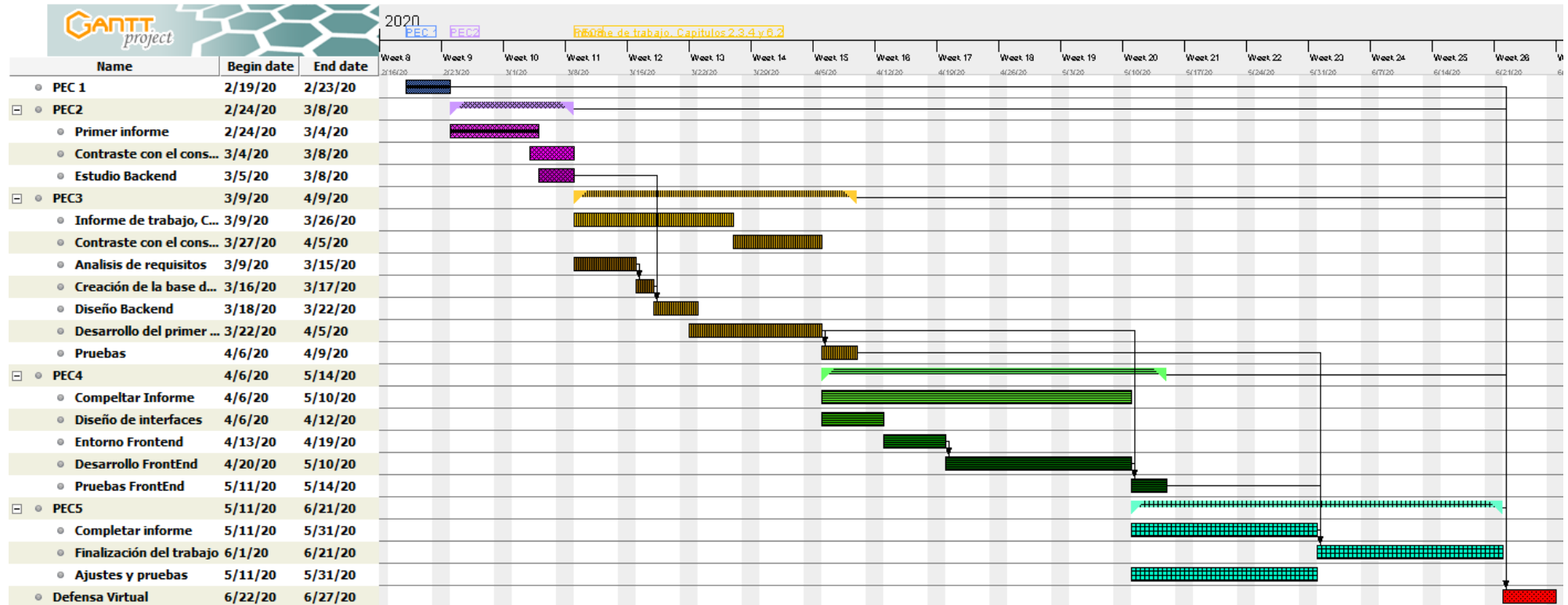


Figura 2: Imagen del Diagrama de Gantr

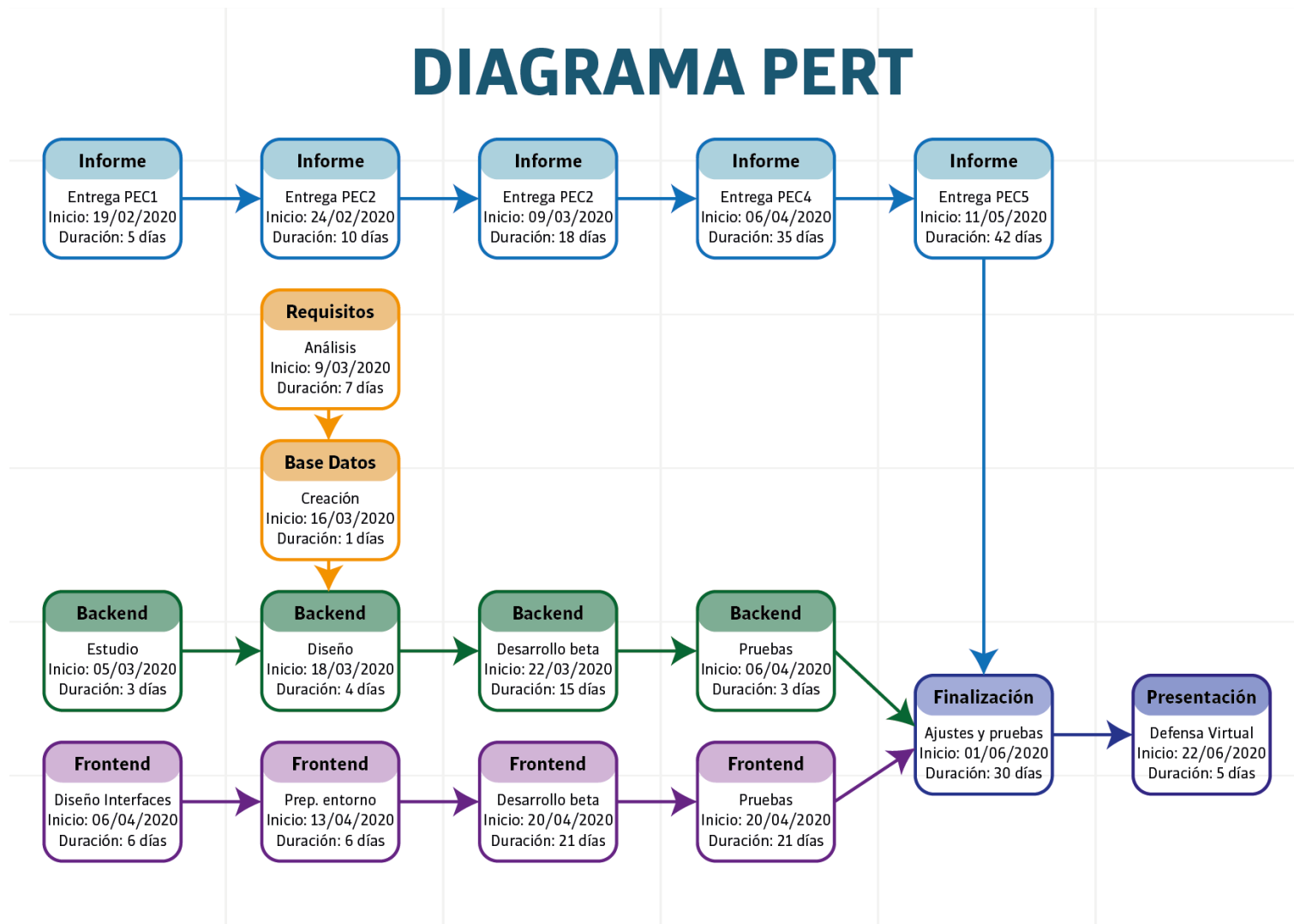


Figura 3: Imagen del Diagrama PERT

1.6. Presupuesto

Concepto	Horas	€/hora	Total
Fase de Análisis			
Definición del proyecto y requisitos	65	37 €	2405 €
Fase de Diseño			
Backend	35	37 €	1.295 €
Frontend	20	37 €	740 €
Fase de Desarrollo			
Configuración del entorno	12	37 €	444 €
Bases de datos	18	37 €	666 €
Backend	90	37 €	3.330 €
Frontend	108	37 €	3.996 €
TOTAL			12.876 €

Tabla 2: Presupuesto económico del proyecto

1.7. Estructura del resto del documento

Los siguientes apartados de los que se compone esta memoria son:

2. Análisis

En el que se analizarán tanto el estado actual de aplicaciones de características similares a la que se plantea en el proyecto, como las diferentes tecnologías que se usarán en él.

Además de este análisis se realizará un análisis de mercado y un estudio de los aspectos legales implicados en el proyecto, mayormente referidos a la protección de datos personales

3. Propuesta

En este apartado se desarrollarán los objetivos que se quieren lograr con la aplicación y sus características técnicas

4. Diseño

Se trabajará la arquitectura general de la aplicación, teniendo en cuenta el diagrama de la base de datos, el flujo de la información y el flujo de navegación.

Una vez definido los flujos de información se trabajará sobre el diseño gráfico, apuntando cómo se han aplicado las técnicas de usabilidad.

En este apartado también se encontrará la información que describirán los procesos de programación y APIs que se han utilizado

5. Implementación

Se ofrecerá información sobre los requisitos e instrucciones de instalación del producto.

6. Demostración

Instrucciones de uso e información sobre diferentes versiones desarrolladas

7. Conclusiones

Conclusiones del trabajo realizado y líneas que se deberán atacar en el futuro para mejorar el producto.

2. Análisis de mercado

2.1. Público objetivo

En cuanto al público objetivo, hay que tener en cuenta que esta aplicación tiene como objetivo cumplir las necesidades de dos tipos de usuario o roles:

- Los administradores o gestores de los diferentes: normalmente este tipo de usuario será un usuario que usa la aplicación en un entorno laboral, es decir, la gestión de estos espacios será parte de sus responsabilidades laborales. Así el usuario tipo del rol de administración de espacios se define como una persona de mediana edad, habituada al uso de aplicaciones informáticas.
- Los usuarios que demandan el uso de dichas instalaciones: En este caso el perfil de usuario es bastante más heterogéneo, con un rango de edad muy amplio, siempre a partir de los 18 años y con unas capacidades de uso de herramientas informáticas muy diferentes, desde habilidades muy básicas hasta capacidades más profundas. Se observa que un rango tan amplio de usuarios hace necesario un trabajo importante en cuanto a la accesibilidad y usabilidad de la aplicación.

En cualquier caso se observa que, por un lado, dado el uso frecuente que ha de tener la aplicación por parte de los administradores de espacios, como por otro lado la variedad de usuarios que pueden usar dicha aplicación para consultar la disponibilidad y realizar solicitudes para el uso de estos espacios, hace necesaria una interface clara y sencilla, y que esta interface será una claves para el éxito del proyecto.

2.2. Competencia/Antecedentes

Actualmente, en la mayoría de los casos, la gestión y consulta del estado de los espacios disponibles se realiza de manera presencial, normalmente es necesario realizar la solicitud por escrito y en muchos casos de una manera presencial ante la entidad encargada de la gestión recursos. Esta solicitud siempre se realiza a ciegas, es decir sin saber la disponibilidad real del espacio que se quiere utilizar.

En los casos en los que las entidades ofrecen información acerca de la disponibilidad de estos espacios, la gestión de esta información suele ser manual, se suelen utilizar herramientas o aplicaciones informáticas para ello, pero no son consecuencia de una administración automatizada. La actualización de la información exige la intervención continuada de los administradores para el seguimiento de la disponibilidad de estos espacios. Este seguimiento se realiza mayormente utilizando

aplicaciones ofimáticas o directamente documentos en formato editable o en formatos cerrados como el PDF⁶.

El problema principal en estos casos es mantener estos documentos actualizados y con una buena calidad de información. Por otro lado dificultan extraordinariamente la gestión compartida por varios administradores de un mismo recurso o grupos de recursos. En estos casos en los que la administración de los recursos se realiza de una manera manual es muy probable la existencia de errores y la generación problemas y confusiones de las solicitudes y asignaciones realizadas.

2.2.1. Herramientas no específicas utilizadas para este mismo fin

En este apartado se estudiarán aplicaciones que pueden ser usados para cumplir los mismos objetivos que cumple la aplicación desarrollada en este proyecto. La mayoría de las aplicaciones son de propósito general, es decir, tienen como objetivo la organización personal o profesional de espacios y tiempo.

Normalmente estas aplicaciones permiten la planificación de eventos en un calendario e implementan ciertas funcionalidades interesantes, pero en ninguno de los casos se tratan de aplicaciones específicas que persiguen cumplir el principal objetivo de este proyecto, aunque sean usadas en muchos casos con este objetivo.

Las aplicaciones que se han testeado son aplicaciones que permiten su uso sin el pago de ninguna licencia, esta es una de las razones por las que son usadas frecuentemente.

2.2.1.1 GoToMeeting



Es una aplicación que entre otras cosas ofrece la posibilidad de realizar reuniones virtuales, más allá de esta característica, permite la gestión de las invitaciones y asistencia a reuniones virtuales o presenciales. Ofrece además diversas opciones como encuestas en tiempo real, herramientas de dibujo y un directorio compartido para los asistentes de dichas reuniones.

⁶ PDF (<https://es.wikipedia.org/wiki/PDF>) es un formato de almacenamiento para documentos digitales independiente de plataformas de software o hardware. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto). Fue inicialmente desarrollado por la empresa Adobe Systems, oficialmente lanzado como un estándar abierto el 1 de julio de 2008

2.2.1.2 Doodle



Es una aplicación de propósito general se puede usar para la gestión de reuniones, aunque siendo una aplicación para la realización de encuestas, en lo que concierne a este es usada de manera habitual para la fijación de fechas y lugares para la realización de reuniones. Una de las funciones más interesantes en cuanto a este proyecto es la posibilidad integrar la aplicación con un calendario permitiendo la sincronización con otros eventos que el usuario tenga asignados en este.

2.2.1.3 Google calendar



Quizás estamos ante la herramienta de propósito general más usada para la gestión de reuniones y eventos. Es una aplicación gratuita integrada en el entorno más popular y usado, por lo que la curva de aprendizaje de esta herramienta es casi nula.

Google Calendar es una aplicación gratuita integrada en el entorno GSuite⁷ de Google. Con base en un calendario es una aplicación tiene muchas funcionalidades, entre ellas cabe destacar las siguientes:

- Crear editar y eliminar eventos.
- Invitar a otras personas a los eventos creados.
- Gestión de salas y recursos.
- Notificar a los interesados la creación y modificación de eventos.
- Confirmar o rechazar la asistencia a estos eventos.
- Comprobación de asistencia de los invitados.
- Compartir calendarios de otros usuarios.

Aunque no es una herramienta específica, es muy usada para cumplir los objetivos de este TFG.

⁷ GSuite (https://es.wikipedia.org/wiki/G_Suite) es un servicio de Google que proporciona varios productos de Google con un nombre de dominio personalizado por el cliente. Cuenta con varias aplicaciones web con funciones similares a las suites ofimáticas tradicionales, incluyendo Gmail, Hangouts, Calendar, Drive, Docs, Sheets, Slides, Groups, News, Play, Sites y Vault.

2.2.2. Aplicaciones específicas disponibles en el mercado

Además de las herramientas estudiadas anteriormente existen aplicaciones en el mercado que cumplen los mismos propósitos que se plantean en este proyecto. La mayoría de ellas son muy específicas y ofrecen diversas funcionalidades que para nada son triviales, una de sus características principales es que todas son de pago, el cual es gestionado de diferentes maneras mediante la compra por la licencia de uso o pagos por suscripción.

2.2.2.1 Meetingroomapp (<https://www.meetingroomapp.com/>)



Es un sistema de gestión de espacios pensado para la gestión de espacios de un mismo edificio. Combina hardware y software, añadiendo tabletas para realizar la solicitud de reserva u ocupar espacios dentro del edificio. Aunque puede tener funciones online, el planteamiento principal de la aplicación es la de gestionar espacios de reunión dentro de la misma empresa o edificio. Aunque es una opción interesante no cumple las necesidades de llegar a toda la ciudadanía mediante una gestión totalmente online.

Funciona con un sistema de pago con suscripción.

2.2.2.2 Planyo (<https://www.planyo.es/>)



De las aplicaciones existentes en el mercado que pueden cumplir con los objetivos específicos que se plantean en este proyecto, Planyo es la herramienta que más se acerca a estos objetivos. Siendo una herramienta muy específica para la gestión de espacios está planteada para su uso de manera general para cualquier tipo de usuario entorno y necesidad. Es una aplicación de pago, por lo que solamente se han podido testear las demos que ofrecen en su web y aunque es una herramienta muy flexible parece que no es fácil de configurar.

2.2.2.3 Simplybook.me (<https://simplybook.me/>)



En este caso el planteamiento que recoge la necesidad y ofrece una solución que no es visible a menos que se pague por ella. En su página web se hace ver que esta aplicación puede cumplir las expectativas que se plantean en el proyecto, pero no hay ninguna manera de acceder a una demo de la aplicación o a código para poder evaluar está.

2.3. Conclusiones del estudio de aplicaciones similares

2.3.1. Puntos a tener en cuenta

Entre las aplicaciones que se han estudiado solamente han podido aportar puntos de vista y soluciones a este proyecto las aplicaciones estudiadas en el apartado '2.2.2. Aplicaciones específicas disponibles en el mercado' y más concretamente las aplicaciones Meetingroomapp y Planyo.

Se han encontrado características muy interesantes en ambas aplicaciones:

- Buena visualización de los espacios que se ofrecen.
- Buen diseño y aspecto cuidado de la aplicación.
- Se tienen en cuenta estándares de accesibilidad y usabilidad.
- Facilidad de uso, la estructura de la aplicación se maneja de una manera sencilla e intuitiva.

En cuanto a las debilidades o puntos en los que la aplicación que en este proyecto se está desarrollando supera a ambas aplicaciones destacan las siguientes:

- Opciones de visualización del uso u ocupación de salas más amplias mediante un calendario.
- Opciones más abiertas en cuanto a compartir información por otras aplicaciones del propio cliente mediante la API.
- Sistema de administración más sencillo.
- Varias opciones de comercialización

2.3.2. Conclusiones

Una vez realizado el estudio de las diferentes aplicaciones propuestas existentes en el mercado que pueden dar solución a las necesidades que se plantean en el proyecto, se observa que por una parte la mayoría de las soluciones usadas en el entorno en el que se ve la necesidad de implementar un software con el objetivo de gestionar espacios, se usan aplicaciones generalistas, sobre todo porque permiten un uso gratuito y son aplicaciones muy versátiles y con muchas opciones. El uso de estas aplicaciones cuyo fin principal no es el de la gestión de espacios (aunque posibilita una solución parcial) no cumple de manera objetiva con los requerimientos planteados en este TFG.

Entre las aplicaciones comerciales y específicas que se han estudiado la solución de Planyo, es la más apropiada para cumplir con los objetivos planteados. Entre sus puntos débiles cabe destacar que al ser una aplicación pensada para su integración en otros entornos, sobre todo páginas web corporativas, la instalación y configuración de la aplicación plantea dificultades que un usuario medio o no técnico no puede salvar, siendo necesaria la intervención de un profesional para su instalación y configuración.

Otro de los puntos importantes a destacar es que la mayoría de las soluciones que se centran en el problema planteado son soluciones de pago y que no ofrecen el código de manera abierta por lo que no se podría modificar ni mejorar ninguno de los aspectos relacionados con la aplicación.

2.4. Análisis DAFO

Seguidamente se analizan de las características internas del proyecto, los beneficios que este puede aportar al usuario de espacios públicos y sus administradores por una parte y las desventajas que puede tener con respecto a la competencia analizada.

2.4.1. Características internas del producto

Fortalezas:

En cuanto a las fortalezas del producto se pueden tener en cuenta los siguientes puntos:

- Es un producto abierto, orientado sobre todo al sector público.
- Puede usarse mediante fórmulas compartidas por ayuntamientos, mancomunidades, etc. Con una única instalación
- Ofrece un buen servicio al ciudadano
- Facilita la gestión al trabajador encargado de la gestión de dichos recursos.

Debilidades:

En cuanto a las debilidades del producto se observan los siguientes puntos:

- Al estar enfocado sobre todo al sector público, su comercialización requiere de un esfuerzo comercial importante.
- El proyecto no plantea como afrontar la comercialización del producto.

2.4.2. Características externas

Oportunidades:

- Cada vez es mayor la necesidad de digitalizar procesos y no sólo en la administración pública
- En la sociedad actual existe una tendencia creciente a realizar todo tipo de gestiones mediante aplicaciones web o aplicaciones para móviles
- En general se observa un aumento de la sensibilidad sobre los servicios digitales por parte de las entidades públicas
- La situación económica actual es favorable y puede propiciar la inversión en proyectos de este tipo.

Amenazas:

- Existe mucha competencia en el sector Online.
- La mayoría de las empresas de la competencia disponen de una fuerza comercial importante

3.Propuesta

3.1. Definición de especificaciones del producto

Esta aplicación permite agilizar y simplificar la realización de solicitudes para el uso de espacios y dota de transparencia y constancia a la información sobre la disponibilidad de estos.

La aplicación está enfocada a tres grandes grupos de usuarios, cuyos objetivos aunque convergen, son diferentes. En cuanto a las especificaciones de producto se han tenido en cuenta estos tres roles de usuario.

3.1.1. Especificaciones del usuario cliente

El usuario definido como cliente es el que se asigna cuando se llega a un acuerdo de uso de la aplicación entre una entidad y los promotores de la aplicación. Su función principal será la de crear los espacios y nombrar a sus “usuarios gestores” para que puedan ocuparse de la gestión de estos espacios. La aplicación les permite:

- Dar de alta a edificios que contendrán espacios o salas. Una vez dados de alta estos, podrán modificar estos datos.
- Dar de alta a espacios que se incluirán dentro de los edificios, estos espacios serán los que se ofrezcan para su uso.
- Dar de alta a los “usuarios gestor” necesarios para la gestión de dichos espacios y asignar los usuarios a los espacios que les corresponden.

3.1.2. Especificaciones del usuario gestor

El usuario definido como gestor, se encargará del mantenimiento de la información sobre los espacios disponibles que contendrá la aplicación, además de recibir y en su caso aceptar o denegar las solicitudes que se tramitar a través de esta. El usuario con este rol dispone de herramientas para:

- Visualizar y modificar información general sobre los espacios, como su dirección, capacidad o aforo máximo, fotografías, etc.
- Aceptar o denegar las solicitudes registradas mediante la aplicación para el uso de espacios de los que tenga responsabilidad.
- Personalización de los mensajes de respuesta automática para los diferentes estados de las solicitudes gestionadas mediante la aplicación

3.1.3. Especificaciones del usuario ciudadano

El interés del usuario con rol de “ciudadano”, es el de poder consultar la disponibilidad de los espacios y el de realizar la solicitud para el uso de estos de una manera rápida y sencilla, para lo cual se dispone de un sistema que transmite de una manera rápida y veraz la información sobre la disponibilidad de estos. Además se incorpora un sistema de mensajería automática (por email) para que el usuario que realice la solicitud sepa en qué situación se encuentra esta.

Para poder cumplir estos objetivos el usuario ciudadano dispone de herramientas que le posibilitan:

- Consultar los espacios disponibles en la aplicación.
- Consultar la disponibilidad de estos espacios para una fecha concreta.
- Realizar la solicitud de uso del espacio que le interese para una fecha dada, deberá de especificar sus datos personales, así como la fecha y hora de entrada, y la de salida.
- Recibir emails automatizados sobre el estado de su solicitud, una vez recibida esta y en el momento en el que cambie el estado de esta.
- Permitir realizar estas gestiones protegiendo sus datos tal y como marca la Ley de Protección de datos⁸.

3.2. Modelo de negocio

Teniendo en cuenta el público objetivo a los que va dirigido el proyecto, sobre todo entidades públicas y empresas de gran tamaño, este es un producto que va a tratar de financiarse mediante dos modelos de negocio diferentes:

- Instalaciones en los servidores⁹ de los clientes
- Como servicio (SASS)

3.2.1. Instalación de instancias de software

En este caso el software se instala en los servidores del cliente, con el objetivo de mantener los datos generados por la aplicación en un servidor que controlará la propia empresa o entidad usuaria del servicio.

⁸ LOPD (<https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>) Ley de Protección de Datos

⁹ Servidor (<https://es.wikipedia.org/wiki/Servidor>) es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora. En la mayoría de los casos una misma computadora puede proveer múltiples servicios y tener varios servidores en funcionamiento. La ventaja de montar un servidor en computadoras dedicadas es la seguridad. Por esta razón la mayoría de los servidores son procesos diseñados de forma que puedan funcionar en computadoras de propósito específico.

En cuanto al modelo de negocio, se ofrecerán dos tipos de contrato:

1. Un contrato por la instalación y el uso del software de por vida con un coste de 1750€, que incluirá una primera instalación del software en los servidores indicados por el cliente y la realización de todas las comprobaciones y pruebas necesarias para asegurar el correcto funcionamiento del software.
2. Se ofrece la posibilidad de contratar un servicio de mantenimiento anual por un valor de 450€ anuales en el que se incluirán las mejoras en cuanto al uso del software por una parte y por otra las refactorizaciones del software generadas por los cambios de versiones de los lenguajes y herramientas de base de datos utilizadas con el objetivo de mantener la seguridad de la aplicación.

3.2.2. Como servicio (SaaS)

Un sistema SaaS o Software as a Service, es un modelo de distribución de software en el que tanto el software como los datos manejados son centralizados y alojados en un único servidor externo a la empresa que contrata el servicio, la implantación de un sistema SaaS para el software empresarial puede ofrecer muchas ventajas a la organización, entre ellas:

- Rapidez en la instalación, ya que el software ya se encuentra instalado y configurado.
- Reducción de los costes, este modelo es más económico que el que se presenta en el punto anterior, ya que la ubicación compartida del mismo reduce los costes de mantenimiento del software. Esta modalidad puede dar acceso al software a PYMEs¹⁰ y pequeñas instituciones que no podrían permitirse asumir los costes generados de la contratación y mantenimiento de un servidor propio.
- Escalabilidad, al operar en un contexto cloud, se puede escalar de una manera más sencilla, permitiendo incluso la integración con otras soluciones SaaS.
- Actualizaciones, es el propietario del software el que se encarga de realizar todas las actualizaciones y operaciones de mantenimiento necesarias para que el software siga funcionando con normalidad.

¹⁰ PYME(https://es.wikipedia.org/wiki/Peque%C3%B1a_y_mediana_empresa) La pequeña y mediana empresa es una empresa que cuenta con ciertos límites ocupacionales y financieros prefijados por los Estados o regiones. Las pymes son agentes con lógicas, culturas, intereses y un espíritu emprendedor específicos.

En cuanto a este modelo de negocio se ofrecerá una única opción de contrato que tendrá un precio dependiente del número de salas con las que se puede operar, según el número de salas los rangos de precios serán los siguientes:

- Menos de 5 salas: 90 € al mes o una cuota de 900€ en un único pago anual
- Entre 5 y 20 salas: 180 € al mes o una cuota de 1800€ en un único pago anual
- Más de 20 salas: 300€ al mes o una cuota de 3000 € en un único pago anual

3.2.3. Viabilidad económica del proyecto

Para calcular la viabilidad del producto, ha de tenerse en cuenta por una parte los gastos ocasionados como consecuencia de la puesta en marcha del producto y confrontar estos con los posibles ingresos.

Con la intención de que el cálculo sea bastante fidedigno, se ha calculado un tiempo de amortización del producto de 3 años, dividiendo entonces al coste de la puesta en marcha del producto en 3 años, se obtendría que el cálculo del coste anual por el desarrollo del producto es de 4300€ al año. A partir de unas ventas mayores a esta cifra, se podría considerar el producto viable.

Haciendo diferentes combinaciones, puede observarse que la viabilidad del proyecto es bastante probable, por ejemplo: la venta de una instalación local del software más 2 contratos de entre 5 y 20 salas anuales darían como resultado la viabilidad del proyecto.

4. Diseño

4.1. Arquitectura general de la aplicación/sistema/servicio

La arquitectura general de la aplicación se soporta sobre 3 servicios principales, el almacenamiento de la información, el backend que se encargará de gestionar las peticiones que le llegarán por parte del usuario y finalmente el frontend o la aplicación que se comunicará directamente con el usuario y que funcionará en el cliente web o navegador de este.

4.1.1. Almacenamiento de la información

El almacenamiento de la información tendrá como soporte un sistema de gestión de bases de datos MySQL. El sistema de bases de datos MySQL utiliza el lenguaje SQL¹¹ (Structured Query Language), que es el lenguaje estándar para tratar con bases de datos relacionales. El lenguaje SQL se usa principalmente para insertar, buscar, actualizar, eliminar registros de la base de datos.

La información necesaria para el funcionamiento de la aplicación ha sido dividida en las siguientes 5 entidades que guardan de manera persistente la siguiente información:

- **Customers:** gestiona la información sobre los clientes de la aplicación, en el caso de que la aplicación sea instalada en formato 'cloud', existirán más de un cliente, mientras que el formato de instalación en servidor local, solamente existirá una única instancia de cliente. Cada cliente tiene asignados varios edificios y usuarios que pueden gestionar estos.
- **Buildings:** información sobre los edificios. En realidad, esta es una unidad de colección de salas, mediante la cual se trasladan la propiedad de estos a las salas. Un edificio es gestionado por un único cliente.
- **Rooms:** las salas forman parte de un único edificio, que es gestionado por un único cliente, pero puede ser gestionado por varios usuarios
- **Users:** lista de usuarios que pueden acceder a la aplicación. Pertenecen a un único cliente y pueden gestionar varias salas siempre pertenecientes a edificios del cliente al que pertenecen
- **Room-request:** solicitudes realizadas por los ciudadanos, que los usuarios asignados para la gestión de salas pueden aceptar o denegar.

¹¹ SQL (<https://es.wikipedia.org/wiki/SQL>) es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

Diagrama de clases de la base de datos

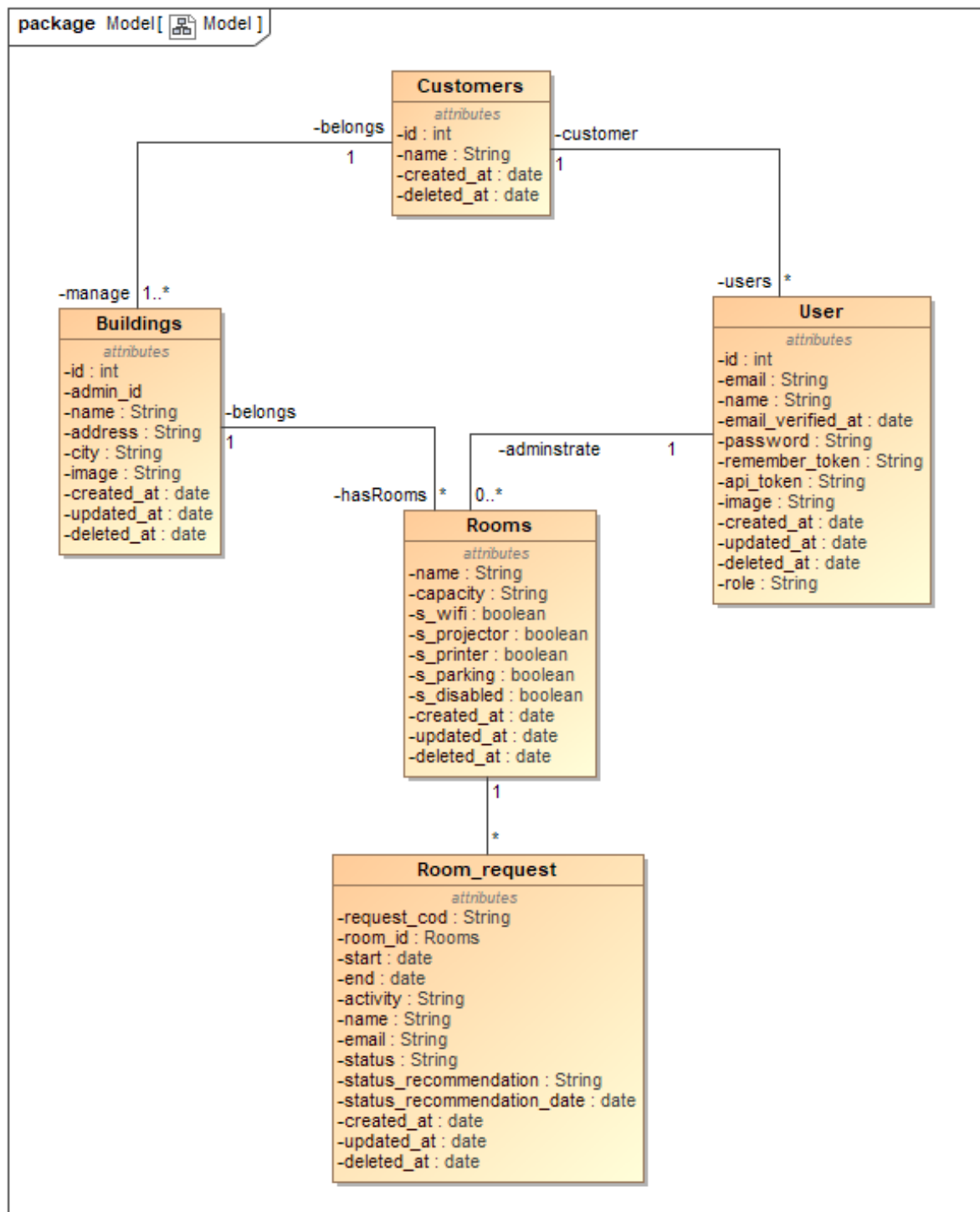


Figura 4: Diagrama de clases de la base de datos

4.1.2. Backend

El backend es la parte de la aplicación que se ocupa de interactuar con la base de datos y de aplicar la lógica de negocio a los datos y a las peticiones introducidas por el usuario. Esta función la realiza una aplicación programada en PHP¹², mediante el framework 'Laravel'¹³, que usa como sistema de comunicación un protocolo API-Rest para el intercambio de información entre el cliente del usuario y la aplicación.

Esta parte de la aplicación se encarga de recoger los inputs que el usuario ha introducido en su navegador, procesar los datos si ese el caso y devolver una respuesta que la aplicación Frontend pueda entender y actuar en consecuencia. El trasvase de información entre las dos partes implicadas es la que se realiza mediante la especificación API-Rest, que no es más que un protocolo de comunicación, que en este caso usa el formato JSON¹⁴ para el intercambio de datos.

4.1.3. Frontend

La aplicación Frontend es la que interactúa directamente con el usuario de la misma, su labor consiste en facilitar al usuario un interface (digamos que un modo de interactuar con los datos), tanto para la entrada de datos por parte de este como para la consulta de estos.

En este proyecto la aplicación de Frontend será programada en JavaScript, utilizando para el desarrollo el framework Angular¹⁵. Angular es un framework muy potente y con una implementación muy amplia en el sector empresarial.

Dado que en este proyecto interactúan con el sistema por lo menos dos tipos de usuarios, los ciudadanos y los administradores nombrados por las entidades que gestionan los espacios, se han desarrollado dos aplicaciones, específicas para cada uno de los tipos de usuario definido. Estos dos

¹² PHP (<https://es.wikipedia.org/wiki/PHP>, <https://www.php.net/>), acrónimo recursivo en inglés de PHP: Hypertext Preprocessor (preprocesador de hipertexto), es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el preprocesado de texto plano en UTF-8.

¹³ Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET. (Fuente: Wikipedia)

¹⁴ JSON (acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje. (Fuente: Wikipedia)

¹⁵ Angular (comúnmente llamado Angular.js o Angular), es un framework de JavaScript de código abierto, mantenido por google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (Fuente: Wikipedia)

tipos de usuario usarán sus respectivas aplicaciones con criterio y objetivos diferenciados, siendo pues aplicaciones con objetivos diferentes, niveles de seguridad diferenciados etc...

Aplicación Frontend para gestores de salas

El objetivo de esta aplicación es la de permitir tanto a los clientes de la misma como a los usuarios gestores de las salas, realizar las labores propias de gestión en la aplicación. Para ello se tienen en cuenta los permisos con los que cuenta cada usuario, en cuanto a la visualización y manejo de información que han sido definidos previamente mediante roles.

Esta aplicación realiza toda la parte de la gestión de la información sobre usuarios, edificios, salas, usuarios y solicitudes, por lo que podemos decir que interactúa de manera total con el backend previo a la capa de datos.

Aunque es evidente, el uso de esta aplicación está sujeto a un previo registro realizado por parte del cliente. El objetivo principal de esta aplicación es que la gestión de todos los datos que se manejan pueda realizarse de una manera sencilla y transparente para el usuario.

Aplicación Frontend para el ciudadano

La aplicación que usará el ciudadano es básicamente una aplicación de consulta tanto de los datos que previamente haya introducido algún usuario registrado, tanto en cuanto a los espacios de los que se dispone como sobre la ocupación de estos.

El objetivo en este caso es la de transmitir toda la información que se dispone sobre los espacios en general y sobre las solicitudes recibidas y aprobadas en firme en particular, es decir esta aplicación cubre uno de los principales objetivos de este proyecto, el de usar y compartir información contrastada, veraz y persistente sobre la situación concreta de las salas disponibles.

Cabe destacar que esta última aplicación, la que respecta al ciudadano, puede verse ampliada por otras aplicaciones en formato web, como esta, o integradas en otras webs, aplicaciones de escritorio, etc... gracias al diseño realizado, basado en una comunicación backend-frontend, que permite manejar los datos desde diferentes situaciones, que pueden ser aplicaciones propiamente dichas o partes de otras.

4.1.4. Diagrama y relaciones Frontend-Backend

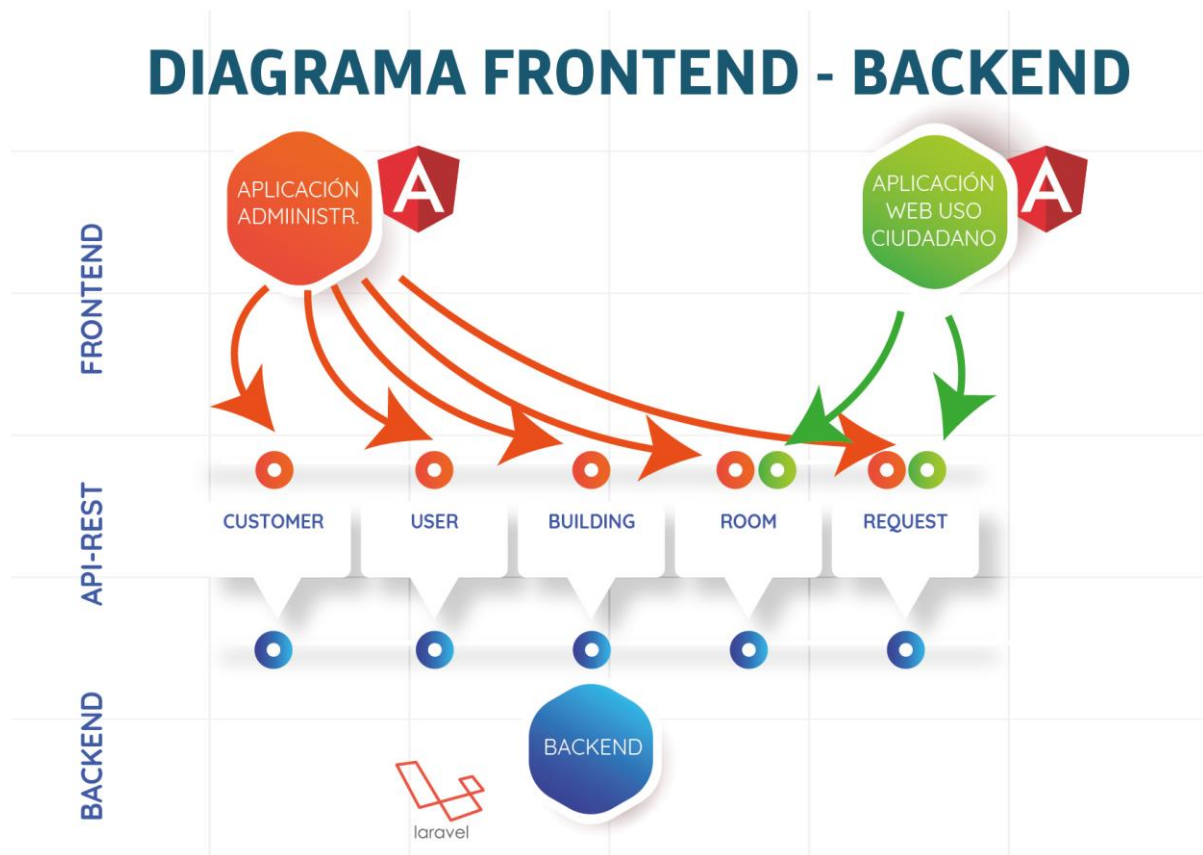


Figura 5: Diagrama y relaciones Backend-FrontEnd

4.1.5. Especificación y lista de peticiones de la API

En este apartado se detalla y documenta la lista de peticiones API utilizada para generar consultas en el backend desde cualquier aplicación de frontend. Se puede observar que dependiendo del método HTTP¹⁶ utilizado para realizar la petición, el sentido de esta y los resultados obtenidos varían.

¹⁶ HTTP (https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto) el Protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, siendo el más importante de ellos el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

Usando la especificación API-Rest los métodos¹⁷ utilizados para las consultas en la API del backend son los siguientes:

- **GET:** Es utilizado únicamente para consultar información al servidor, muy parecidos a realizar un SELECT a la base de datos.
- **POST:** Es utilizado para solicitar la creación de un nuevo registro, es decir, algo que no existía previamente, es decir, es equivalente a realizar un INSERT en la base de datos.
- **PUT:** Se utiliza para actualizar por completo un registro existente, es decir, es parecido a realizar un UPDATE a la base de datos.
- **DELETE:** Este método se utiliza para eliminar un registro existente, es similar a DELETE a la base de datos.

Lista de peticiones de elementos generales

Nombre:	Room Request. Gestión de los solicitudes de uso de una sala
URL¹⁸:	api/room-request
Método:	GET: se obtiene una lista de las solicitudes disponibles POST: crea una nueva instancia del objeto RoomRequest
Respuesta	La solicitud ha tenido éxito
status 200:	GET: Lista paginada en formato JSON de todas las instancias de RoomRequest disponibles. POST: representación del objeto RoomRequest creado, en formato JSON
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: error en el servidor backend

¹⁷ Métodos (https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto) HTTP define una serie predefinida de métodos de petición (algunas veces referido como "verbos") que pueden utilizarse. El protocolo tiene flexibilidad para ir añadiendo nuevos métodos y para así añadir nuevas funcionalidades.

¹⁸ URL (https://es.wikipedia.org/wiki/Localizador_de_recursos_uniforme) Un localizador de recursos uniforme (más conocido por las siglas URL, del inglés Uniform Resource Locator)¹ es un identificador de recursos uniforme (Uniform Resource Identifier, URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. Están formados por una secuencia de caracteres de acuerdo con un formato modélico y estándar que designa recursos en una red como, por ejemplo, Internet.

Nombre:	Room Request. Gestión de las solicitudes
URL:	api/room-reuquest/{id}
Método:	GET: se obtiene el objeto RoomRequest que coincide con el id de la solicitud PUT: actualiza en base de datos la instancia de la entidad RoomRequest que coincide con el id de la solicitud DELETE: borra la instancia de RoomRequest que coincide con el id de la solicitud
Respuesta:	GET: representación de la instancia del objeto RoomRequest, en formato JSON PUT: representación de la instancia del objeto RoomRequest, en formato JSON DELETE: estado de la operación.
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: erro en el servidor backend

Nombre:	User. Gestión de los usuarios
URL:	api/user
Método:	GET: se obtiene una lista de los usuarios disponibles POST: crea una nueva instancia del objeto User
Respuesta:	GET: Lista paginada en formato JSON de todos las instancias de User disponibles. POST: representación del objeto Room creado, en formato JSON
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: erro en el servidor backend

Nombre:	User. Gestión de los usuarios
URL:	api/user/{id}
Método:	GET: se obtiene el objeto User que coincide con el id de la solicitud PUT: actualiza en base de datos la instancia de la entidad User que coincide con el id de la solicitud DELETE: borra la instancia de User que coincide con el id de la solicitud
Respuesta:	GET: representación de la instancia del objeto User, en formato JSON PUT: representación de la instancia del objeto User, en formato JSON DELETE: estado de la operación.

Nombre:	Room. Gestión de las salas disponibles
URL:	api/room
Método:	GET: se obtiene una lista de las salas disponibles POST: crea una nueva instancia del objeto Room
Respuesta:	GET: Lista paginada en formato JSON de todas las instancias de Room disponibles. POST: representación del objeto Room creado, en formato JSON
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: error en el servidor backend

Nombre:	Room. Gestión de los edificios
URL:	api/room/{id}
Método:	GET: se obtiene el objeto Room que coincide con el id de la solicitud PUT: actualiza en base de datos la instancia de la entidad Room que coincide con el id de la solicitud DELETE: borra la instancia de Room que coincide con el id de la solicitud
Respuesta:	GET: representación de la instancia del objeto Room, en formato JSON PUT: representación de la instancia del objeto Room, en formato JSON DELETE: estado de la operación.
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: error en el servidor backend

Nombre:	Building. Gestión de los edificios
URL:	api/building
Método:	GET: se obtiene una lista de los edificios disponibles POST: crea una nueva instancia del objeto Building
Respuesta:	GET: Lista paginada en formato JSON de todas las instancias de Building disponibles. POST: representación del objeto Building creado, en formato JSON

Nombre:	Building. Gestión de los edificios
URL:	api/building/{id}
Método:	GET: se obtiene el objeto Building que coincide con el id de la solicitud PUT: actualiza en base de datos la instancia de la entidad Building que coincide con el id de la solicitud DELETE: borra la instancia de Building que coincide con el id de la solicitud
Respuesta:	GET: representación de la instancia del objeto Building, en formato JSON PUT: representación de la instancia del objeto Building, en formato JSON DELETE: estado de la operación.
Lista de status	200 OK: la solicitud ha tenido éxito 401 Unauthorized: no tiene autorización para realizar la solicitud 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: erro en el servidor backend

Lista de peticiones de imágenes

Nombre:	File upload. Upload de imágenes de cualquier tipo de elemento
URL:	api/file-upload/{type}/{id}
Método:	POST: realiza un upload de la imagen en cuestión al servidor y actualiza el campo 'image' del elemento en cuestión
Respuesta:	POST: estado de la operación
Lista de status	200 OK: la solicitud ha tenido éxito 404 Not Found: no se encuentra el recurso sobre el que realiza la solicitud 500 Internal Server Error: erro en el servidor backend

Lista de peticiones de login/logout de usuarios

Nombre:	Login. Permite el acceso a usuarios registrados en la aplicación de administración
URL:	api/login
Método:	POST: realiza una petición de login en el sistema de usuarios
Respuesta:	POST: crea un token basado en JWT

Nombre:	Logout. Deja de permitir el acceso al usuario en la aplicación de administración
URL:	api/logout
Método:	GET: realiza una petición al sistema para destruir el token generado
Respuesta:	GET: destruye el token JWT generado dejando sin autorización la sesión creada

Ejemplo respuesta json (http://localhost:8000/api/room)¹⁹:

```
{
  "data": [
    {
      "id": 1,
      "admin_id": 4,
      "building_id": 1,
      "building": {
        "id": 1,
        "admin_id": 1,
        "customer_id": 0,
        "name": "BEC Baracaldo",
        "address": "Sit neque et velit eligendi saepe.",
        "city": "Baracaldo",
        "image": "1588955894.jpeg",
        "created_at": "2020-04-20T15:27:11.000000Z",
        "updated_at": "2020-05-08T16:38:14.000000Z"
      },
      "name": "Sala azull",
      "capacity": 150,
      "image": "1588955866.jpeg",
      "s_wifi": 1,
      "s_projector": 1,
      "s_printer": 0,
      "s_parking": 0,
      "s_disabled": 1
    },
    {
      "id": 2,
      "admin_id": 0,
      "building_id": 2,
      "building": {
        "id": 2,
        "admin_id": 1,
        "customer_id": 0,
        "name": "Palacio de Congresos Kursall",
        "address": null,
        "city": "San Sebastian",
        "image": "1589018893.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:08:13.000000Z"
      },
      "name": "Kubo txikia",
      "capacity": 250,
      "image": "1589019034.jpeg",
      "s_wifi": 1,
      "s_projector": 1,
      "s_printer": 0,
      "s_parking": 1,
      "s_disabled": 1
    },
    {
      "id": 3,
      "admin_id": 0,
      "building_id": 5,
      "building": {
        "id": 5,
        "admin_id": 2,
        "customer_id": 0,
        "name": "Baluarte",
        "address": null,

```

¹⁹ Más ejemplos de respuestas en formato JSON en el ANEXO D

```

        "city": "Iruñea",
        "image": "1589018942.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:02.000000Z"
    },
    "name": "Auditorio Baluarte",
    "capacity": 750,
    "image": "1589019147.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 1,
    "s_parking": 1,
    "s_disabled": 1
},
{
    "id": 4,
    "admin_id": 0,
    "building_id": 4,
    "building": {
        "id": 4,
        "admin_id": 1,
        "customer_id": 0,
        "name": "Palacio Europa",
        "address": null,
        "city": "Vitoria-Gasteiz",
        "image": "1589018929.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:08:49.000000Z"
    },
    "name": "Sala auditorio",
    "capacity": 450,
    "image": "1589021396.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 1,
    "s_parking": 1,
    "s_disabled": 1
},
{
    "id": 5,
    "admin_id": 0,
    "building_id": 3,
    "building": {
        "id": 3,
        "admin_id": 1,
        "customer_id": 0,
        "name": "Palacio Euskalduna",
        "address": null,
        "city": "Bilbo",
        "image": "1589018995.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:55.000000Z"
    },
    "name": "Sala Euskalduna Tres",
    "capacity": 125,
    "image": "1589022461.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 1,
    "s_parking": 1,
    "s_disabled": 1
}
],
"status": 200,
"links": {

```

```
    "first": "http://localhost:8000/api/room?page=1",
    "last": "http://localhost:8000/api/room?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "path": "http://localhost:8000/api/room",
    "per_page": 15,
    "to": 5,
    "total": 5
  }
}
```

4.1.6. Diagrama general



Figura 6: Diagrama general, flujo de la aplicación y responsabilidades

4.1.7. Casos de uso más comunes

Usuario Ciudadano. Consultar disponibilidad de un espacio

Caso de uso²⁰: Consultar disponibilidad de un espacio

Actor principal: Usuario ciudadano

Nivel de objetivo: usuario

Stakeholders e intereses:

El usuario quiere consultar la disponibilidad de un espacio para una fecha dada

El usuario administrador quiere verificar que la información contenida sobre la disponibilidad es correcta

Precondición: el usuario ha tenido que elegir previamente a la consulta una sala

Garantías en caso de éxito: el sistema mostrará al usuario las solicitudes aprobadas para un espacio concreto mediante un calendario.

Escenario principal de éxito:

1. El usuario mediante el índice de la home de la aplicación web pública elige una sala de entre las que están disponibles
2. El sistema muestra la información general y concreta sobre la sala, mediante texto, imágenes e iconos.
3. El usuario ve como opción predeterminada el calendario con los datos de las solicitudes aprobadas en un mes.
4. Clicando sobre los días del calendario puede ver la ocupación concreta para una fecha, un punto de color le avisará si existe alguna solicitud aprobada.

Usuario Ciudadano. Realizar solicitud de un espacio para una fecha determinada

Caso de uso: Realizar solicitud de un espacio para una fecha determinada

Actor principal: Usuario ciudadano

Nivel de objetivo: usuario

Stakeholders e intereses:

El usuario quiere realizar la solicitud de un espacio para una fecha dada

Precondición: el usuario ha tenido que elegir previamente a la solicitud una sala para poder realizarla.

²⁰ Caso de uso (https://es.wikipedia.org/wiki/Caso_de_uso) es la descripción de una acción o actividad. Un diagrama de caso de uso es una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso. Los personajes o entidades que participarán en un diagrama de caso de uso se denominan actores. En el contexto de ingeniería del software, un diagrama de caso de uso representa a un sistema o subsistema como un conjunto de interacciones que se desarrollarán entre casos de uso y entre estos y sus actores en respuesta a un evento que inicia un actor principal.

Garantías en caso de éxito:

1. El sistema mostrará al usuario un código de solicitud en la pantalla, mediante el cual podrá realizar la consulta sobre la situación de esta.
2. El usuario recibirá un email con el código de solicitud.

Escenario principal de éxito:

1. El usuario mediante el índice de la home de la aplicación web pública elige una sala de entre las que están disponibles.
2. El sistema muestra la información general y concreta sobre la sala, mediante texto, imágenes e iconos.
3. El usuario podrá ver en el calendario los datos de las solicitudes aprobadas en el mes.
4. Si lo estima oportuno, puede Consultar disponibilidad de un espacio.
5. Pulsando sobre la pestaña RESERVAR puede rellenar el formulario y enviarlo.
6. El usuario recibirá en pantalla un código para poder así consultar la situación sobre su solicitud.
7. El usuario recibirá un mail con el código para poder así consultar la situación sobre su solicitud.

Usuario Ciudadano. Acceder al área privada para hacer seguimiento de su solicitud

Caso de uso: Acceder al área privada para hacer seguimiento de su solicitud

Actor principal: Usuario ciudadano

Nivel de objetivo: usuario

Stakeholders e intereses:

El usuario quiere saber la situación de una solicitud concreta que ha realizado

Precondición: el usuario ha tenido que realizar previamente la solicitud.

Garantías en caso de éxito:

1. El sistema mostrará al usuario un código de solicitud en la pantalla, al realizar la solicitud.
2. El usuario habrá recibido un email con el código de solicitud.
3. El usuario podrá consultar la situación de su solicitud mediante el código recibido.

Escenario principal de éxito:

1. El usuario selecciona la opción de entrar al área privada.
2. Introduce el código de la solicitud que ha recibido por pantalla y por email, junto con el email usado para realizar la solicitud
3. El usuario puede observar la situación de su solicitud

Escenario alternativo:

1. El usuario no tiene el código de la solicitud.
2. No podrá ver la situación de su solicitud

Usuario Administrador. Crear una nueva sala

Caso de uso: Crear una nueva sala

Actor principal: Usuario administrador

Nivel de objetivo: usuario

Stakeholders e intereses:

El usuario administrador quiere dar de alta una nueva sala.

Precondición:

El usuario administrador se debe de haber identificado en el sistema mediante el login.

El usuario administrador debe de tener un rol que le permita realizar la acción de crear una sala.

Garantías en caso de éxito:

3. El sistema mostrará un mensaje de que la nueva instancia se ha creado con éxito.

Escenario principal de éxito:

1. El usuario debe de autenticarse con éxito en la aplicación de administración.
2. El usuario debe de elegir en la opción de salas el botón “Nueva Sala”.
3. El usuario debe de rellenar todos los datos necesarios para la creación de una sala.
4. El usuario administrador puede incluir fotografías y datos de segundo nivel sobre la sala, como los servicios que ofrece.

Escenario alternativo:

1. El usuario no tiene permiso para crear una nueva sala.
2. El usuario no ve el botón “crear sala”.

Usuario Administrador. Aprobar una solicitud de uso de una sala

Caso de uso: Aprobar una solicitud de uso de una sala

Actor principal: Usuario administrador

Nivel de objetivo: usuario

Stakeholders e intereses:

El usuario administrador quiere aprobar la solicitud de uso de una sala.

El usuario ciudadano que ha realizado la solicitud quiere obtener una respuesta a su solicitud.

Precondición:

El usuario administrador se debe de haber identificado en el sistema mediante el login.

El usuario administrador debe de tener un rol que le permita realizar la acción de aprobar una solicitud.

Garantías en caso de éxito:

1. El sistema mostrará un mensaje de que la solicitud se ha aprobado con éxito.
2. El sistema enviará una notificación por mail al usuario ciudadano, notificándole que su solicitud ha cambiado de estado.

Escenario principal de éxito:

1. El usuario debe de autenticarse con éxito en la aplicación de administración.
2. El usuario debe de elegir la opción de consultar las últimas solicitudes.
3. El usuario puede ver si existe alguna recomendación para la solicitud concreta.
4. El usuario aprueba la solicitud realizada por el usuario.

Escenario alternativo:

1. El usuario no tiene permiso para gestionar esa solicitud.
2. El sistema responde con un error en el caso de que un administrador intente realizar operaciones sobre una solicitud para la cual no tiene permisos de gestión.

4.2. Arquitectura de la información y diagramas de navegación

4.2.1. Arquitectura de la información

En cuanto a la arquitectura de la información, es la propia estructura de la aplicación la que va a servir de base para estructurar la información. Por jerarquía se observa la siguiente:

1. Edificios
2. Espacios o salas
3. Formulario para la solicitud de uso de los espacios

Esta estructura que se plantea es abierta, es decir estructurada en estas 3 categorías o niveles la estructura de la información que se expone en cada una de ellas será la misma en todos los casos, la información cambia según la disponibilidad de espacios que dé cada uno de los clientes.

4.2.2. Estructura de la información: el caso administrador

La estructura de la información en este caso también seguirá una estructura jerárquica, marcada por la estructura y funcionalidades de la aplicación, aunque lo cierto es que esta estructura se verá limitada por los permisos de acceso de cada uno de los tipos o roles de administración definidos.

En principio la estructura jerárquica de la información será la siguiente:

1. Cliente
2. Administradores
3. Edificios
4. Salas
5. Solicitudes, donde se evalúan las solicitudes recibidas.

Al inicio de este apartado se ha explicado que existen roles diferentes de administración que mediante la gestión de permisos tendrán disponible una estructura de información que variará según estos. Los roles de administración que se prevén en la aplicación y la definición de sus permisos o privilegios son:

	CRUD ²¹ administradores	CRUD edificios	CRUD salas	Asignar admin/edif	Asignar admin/sala	Evalúan Solicitud
Cliente	SI	SI	SI	SI	SI	SI
Admin.Edif.	NO	NO	SI	NO	SI	SI
Admin.Sala	NO	NO	NO	NO	NO	SI

Tabla 3: Roles de administración y permisos

²¹ CRUD (<https://es.wikipedia.org/wiki/CRUD>) En informática, CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

4.2.3. Diagrama de navegación de usuario no logueado (ciudadano)



Figura 7: Diagrama de navegación para un usuario no logueado, del tipo ciudadano

4.2.4. Diagrama de navegación usuario logueado (administrador)



Figura 8: Diagrama de navegación, caso de usuario logueado, tipo administrador

4.3. Diseño gráfico e interfaces

4.3.1. Estilos

Este proyecto está enfocado a la realización o construcción de una herramienta específica de consulta, realización de solicitudes y gestión de éstas, por lo que claramente se trata de una herramienta para la mejora mediante la digitalización de procesos que actualmente tienen un coste de tiempo (y de movilidad en algunos casos) bastante alto y aunque no es un proceso complejo, dada la cantidad más o menos grande de solicitudes que hay que gestionar si puede llegar a ser un proceso farragoso.

El proyecto pues, tiene en cuenta las dificultades actuales a la hora de realizar y gestionar las solicitudes y plantea una línea de trabajo clara y sencilla en cuanto al estilo gráfico, la navegación y la usabilidad.

Se trabajará pues sobre una interface blanca, es decir sin apenas adornos y en el que prime la facilidad de uso y de transmisión de la información.

4.3.2. Nombre comercial, logotipo y símbolo

El nombre comercial que se usará para la comercialización y las acciones de márketing en torno al producto será “space”, en clara referencia al objeto principal del proyecto.

Se ha diseñado un logotipo acorde con las líneas o estilo gráfico definido en el punto anterior, con un estilo claro, fácilmente identificable y con una paleta de colores fríos pero naturales, que aunque identificarán de manera clara el proyecto no interferirán en la navegación del usuario.



Figura 9: Logotipo y símbolo del proyecto

4.3.3. Paleta de colores

El color no ha sido un elemento destacable en el diseño de la aplicación, su uso está limitado al logotipo y como elemento para destacar cierta información mediante iconos. Así pues, en cuanto a la paleta de colores, se ha seguido la línea marcada por el logotipo, que está formado por un degradado de 5 colores fríos.

De entre estos 5 colores fríos, se ha elegido un color principal que en su notación hexadecimal se define como #87BD24 para destacar los elementos de información principales usados en la aplicación como los iconos. El resto de los 5 colores se han usado a modo complementario siempre con el objetivo de facilitar la identificación de la información que contiene la aplicación.

En cuanto a los fondos en la pantalla, el color predominante es el blanco, junto con las combinaciones de gris para las cabeceras y fondo de elementos o agrupaciones de elementos. Para este uso también se ha definido como guía una paleta formada por 5 variaciones de gris.

La que sigue es la definición de la paleta de colores utilizada en la aplicación:



Figura 10: Paleta de colores

4.3.4. Fuentes

La fuente usada en toda la aplicación es la fuente 'Roboto' que es una de las fuentes más utilizada en las aplicaciones web, dada su buena legibilidad. Roboto es una familia de fuentes tipográficas del tipo sans-serif creada por Google para utilizarla como fuente predeterminada de su sistema operativo Android 4.0 ICS. Esta familia de fuentes moderna y accesible incluye diversos grosores y trazos para que cada usuario o desarrollador pueda adaptarlo a sus necesidades.

4.3.5. Usabilidad /UX

En cuanto a la usabilidad del proyecto, el trabajo realizado se ha centrado básicamente en la capa de frontend, ya que es esta la capa que trabaja directamente con el usuario. Podemos definir la usabilidad de una página web como la facilidad con la que el usuario puede navegar y conseguir aquella información que busca. Este planeamiento es el que se ha seguido en el desarrollo del proyecto en cuanto a la navegabilidad y usabilidad de este, la facilidad para encontrar aquella información que se está buscando.

A la hora de enfocar el trabajo sobre la usabilidad se han utilizado como guía los 10 principios básicos definidos por Jakob Nielsen²² en 1995:

- Visibilidad del estado del sistema. En este caso se ha aplicado sobre las solicitudes realizadas por los ciudadanos.
Las solicitudes tienen sus propios estados de sistema, es decir, se reciben, se evalúan, se aceptan o se deniegan. La información sobre estos estados es la que se actualiza y se comunica al usuario, mediante la propia aplicación cuando sea posible y cuando esto no pueda ser así la comunicación se realizará a través de emails personalizados.
- Adecuación entre el sistema y el mundo real. Se ha cuidado el uso del lenguaje de manera que este sea fácilmente comprensible por el usuario.
- Libertad y control por el usuario. Los usuarios pueden volver fácilmente a un estado anterior. Una vez realizada una solicitud por parte del ciudadano la aplicación permite anularla en cualquier momento.
- Consistencia y estándares. Se mantiene un óptimo equilibrio entre sus elementos internos (botones, navegación, texto, enlaces, etc.). Y se respetan los estándares preestablecidos.

²² Jakob Nielsen (https://es.wikipedia.org/wiki/Jakob_Nielsen) (nacido el 5 de octubre de 1957, en Copenhague, Dinamarca) es una de las personas más respetadas en el ámbito mundial sobre usabilidad en la web. Este ingeniero de interfaces obtuvo su doctorado en diseño de interfaces de usuario y ciencias de la computación en la Universidad Técnica de Dinamarca. Su andadura profesional le ha hecho pasar por empresas como Bellcore, IBM y Sun Microsystems. Actualmente figura como cofundador de Nielsen Norman Group con Donald Norman, otro experto en usabilidad.

- Prevención de errores. Este apartado se ha trabajado de manera muy profunda, mediante diferentes comprobaciones y un diseño en el que han primado las validaciones en la capa de frontend, para la prevención de errores.
- Reconocer mejor que recordar. El uso de iconos para las opciones más comunes sobre todo en la información referente a los espacios/salas, mejora la localización de la información.
- Flexibilidad y eficiencia de uso. Este aspecto se ha desarrollado sobre todo en el panel de administración, trabajando la posibilidad de usar mensajes y atajos para notificar la necesidad de iniciar los procesos de evaluación de nuevas solicitudes.
- Estética y diseño minimalista. Se ha tendido a la simplificación en cuanto a elementos gráficos y de diseño, manteniendo solamente aquellos que son relevantes para la identificación de la información
- Ayudar a los usuarios a reconocer, diagnosticar y solucionar los errores. Los mensajes de error utilizados en la aplicación expresan claramente cuál ha sido el problema que ha generado dicho error.
- Ayuda y documentación. En este caso, también se ha trabajado especialmente el panel de administración, ya que en la parte de uso del ciudadano no se ha detectado la necesidad de incluir documentación.

4.4. Lenguajes de programación y APIs utilizados

Existen multitud de lenguajes de programación que pueden ser usados para el desarrollo de este proyecto. En este caso se han elegido 3 tecnologías muy populares y muy usadas en el mundo del desarrollo web, MySQL, PHP y JavaScript. El conjunto de estas tres tecnologías es conocido como LAMP y es un paquete muy usado para el desarrollo de aplicaciones similares a esta. Estas tres tecnologías responden cada una de ellas a las tres capas de desarrollo mediante las que se ha planteado la arquitectura del proyecto:

- MySQL, para la capa de datos.
- PHP, para la capa de lógica de negocio.
- JavaScript, para la capa de presentación

En cuanto al uso de APIs externas, el proyecto no usa ninguna API externa para su desarrollo.

Se observa que los 3 lenguajes de programación elegidos para el desarrollo del proyecto son lenguajes que además de ser muy populares y usados para el desarrollo web, han sido utilizados en los estudios cursados en el grado.

4.4.1. Capa de datos: MySQL

La elección en este caso ha sido bastante clara, aunque también se han evaluado otros sistemas de gestión de bases de datos, como PostgreSQL o MariaDB. Definitivamente se ha elegido MySQL debido a que es un sistema de base de datos extremadamente rápido para cargas de trabajo con mucha lectura, aunque a veces sea a costa de la concurrencia cuando se combina con operaciones de escritura, es decir, es un sistema que asegura un buen rendimiento para operaciones de consulta, que son las que más se van a realizar en nuestro caso.

Además del buen rendimiento, en el caso de MySQL confluyen que existe muchísima documentación en la web sobre su instalación, capacidades y desarrollo, así como una variada oferta de hosting en el que se ofrece esta base de datos instalada por defecto.

También ha sido importante a la hora de tomar la decisión la experiencia adquirida en los estudios del grado con la base de datos MySQL, ya que ha sido la base de datos de referencia en las asignaturas del grado sobre base de datos.

Estudio de la alternativa PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales orientados a objetos y de código abierto. Impulsando este sistema de base de datos existe una comunidad de desarrolladores que trabajan de forma desinteresada, por lo que el desarrollo de PostgreSQL no depende de ninguna empresa. En cuanto a sus características más importantes destacar las siguientes:

- Está disponible para diferentes sistemas operativos, es un sistema multiplataforma.
- Es de gran escalabilidad, permite configurar el sistema según el hardware del que se disponga, tanto en cantidad de CPUs como en cantidad de memoria asignada.
- Un sistema muy estable, ya que tiene más de 20 años de desarrollo activo. Dispone de un entorno de Alta disponibilidad ante recuperación de desastres o espera por sobrecargas.
- Implementa casi todas las funcionalidades del estándar SQL.
- Garantiza que las transacciones no interfieran unas con otras, ya que cumple con la característica ACID Compliant. Con ello se garantiza la información de las Bases de Datos y que los datos perduren en el sistema.
- Extensibilidad: tenemos a nuestra disponibilidad una gran variedad de extensiones distribuidas por el grupo de desarrolladores de PostgreSQL y por terceros.

Estudio de la alternativa MariaDB

Los sistemas de gestión de bases de datos MySQL y MariaDB están basados en el mismo núcleo de software. MariaDB es una bifurcación (“fork” en inglés) de MySQL 5.1, aunque con el paso del tiempo se ha convertido en un sistema de gestión de bases de datos autónomo.

El campo de aplicación de MySQL no se diferencia del de MariaDB. Ambos buscan una compatibilidad al 100 % con el lenguaje de base de datos SQL. Tanto MySQL como MariaDB son adecuados para escenarios con bases de datos distribuidas. Para poder ofrecer a los usuarios unos sistemas con alta disponibilidad y escalabilidad lineal, ambos proyectos de software ofrecen soluciones de clúster.

El equipo de desarrolladores de MariaDB, formado en torno a Michael Widenius, ha adoptado lo mejor del software principal MySQL y lo ha ampliado con numerosas características.

4.4.2. Capa de lógica de negocio: PHP y Laravel

Para el desarrollo de la capa de lógica de negocio de este proyecto se ha elegido el lenguaje PHP. PHP es un lenguaje de código abierto muy popular, adecuado para realizar desarrollos web. Es un lenguaje muy popular gracias a su uso en el desarrollo de páginas web y en diferentes CMS (sistema de gestión de contenido), entre los que cabe destacar WordPress que representa el 23,4% de las web en todo el mundo según sus propios datos.

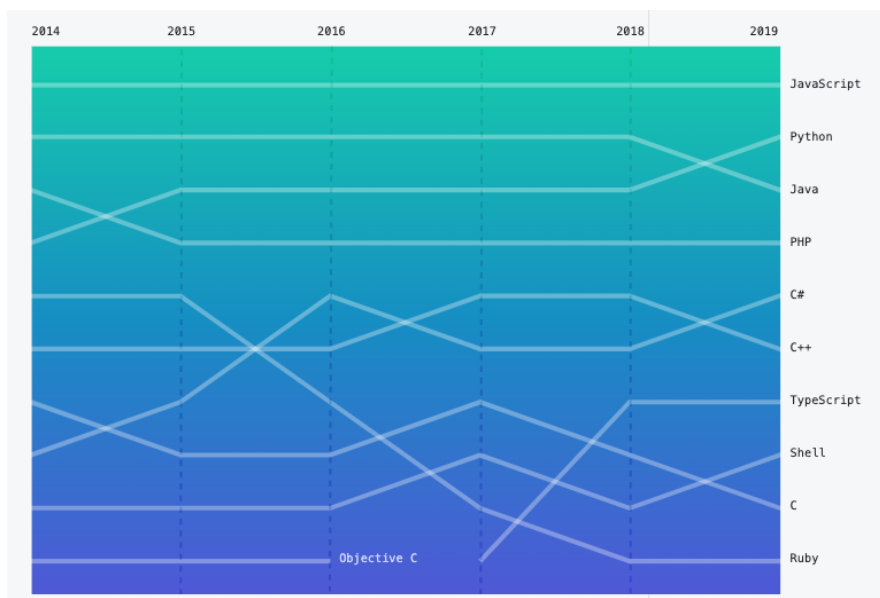


Figura 11: Evolución del uso de diferentes lenguajes (The estate of the Octoverse 2019 - GitHub)

PHP es conocido como un lenguaje basado en servidores. Esto es porque el PHP no se ejecuta en el ordenador del cliente, sino en el propio servidor donde está alojado el código, mostrándose el resultado de ese código en el navegador del cliente.

Entre las características de PHP más destacables se encuentran:

- Es un lenguaje de código abierto
- Es multiplataforma, por lo que puede funcionar en varios sistemas operativos como Linux, Unix y Windows entre otros.
- Debido a su popularidad, existen muchas referencias, tutoriales y guías de desarrollo en la web.
- Tiene una gran comunidad de desarrolladores
- Una completa documentación que explica las diferentes opciones que ofrece el lenguaje.

El framework Laravel

Laravel es un framework PHP cuya principal característica es que es de código abierto y ofrece una curva de aprendizaje relativamente suave. Fue creado por Taylor Otwell en 2011 y se basa en Symphony. Su filosofía es desarrollar código PHP de forma elegante y simple basado en un modelo MVC (Modelo-Vista-Controlador) y su lema es que Laravel es el “framework PHP para artesanos”.

Los Frameworks son un conjunto de utilidades o módulos ya programados que toman como base una programación nativa en una tecnología y “evolucionan” el lenguaje base para hacerlo más operativo y modular.

Un gran punto a favor y que hay destacar de este framework es la gran comunidad y documentación que existe:

- Una comunidad de desarrolladores muy activa que aporta conocimiento y funcionalidades a través de librerías que fomentan el código libre y amplían las funcionalidades del propio framework.
- En cuanto a la documentación cabe destacar que es muy completa y de mucha calidad pensada para los propios desarrolladores y en la que estos también están implicados. La documentación oficial se puede encontrar en la página web oficial del framework (<https://laravel.com/>)

Las características técnicas más destacables que aporta el framework son:

- Eloquent: Eloquent es el ORM que incluye Laravel para manejar de una forma fácil y sencilla los procesos correspondientes al manejo de bases de datos en nuestro proyecto. Transforma las consultas SQL a un sistema MVC lo que no permite procesar consultas SQL directamente y así protegernos de la inyección SQL.
- Artisan: Artisan es una herramienta de la consola de Laravel, que nos permite realizar diferentes tareas, ejecutar comandos, pruebas, crear objetos, incluso crear sus propios comandos. La automatización es clave para el desarrollo de un proyecto.
- Routing: Laravel proporciona un sistema de organización y gestión de rutas que nos permite controlar de manera exhaustiva las rutas de nuestro sistema.
- Middlewares: Son una especie de controladores que se ejecutan antes y después de una petición al servidor, lo que nos permite insertar múltiples controles, validaciones o procesos en estos puntos del flujo de la aplicación.
- Inyección de dependencias en rutas y controladores
- Algo más que MVC²³: El modelo vista controlador, es uno de los modelos de programación más seguro, sin embargo, Laravel va más lejos, ya que no solo utiliza vistas y controladores, sino que tiene rutas, middleware y otros sistemas de seguridad que blindan aún más el proyecto que el MVC.

Como conclusión cabe destacar que Laravel está diseñado para un rápido desarrollo de aplicaciones. El framework tiene un motor de plantillas bien construido que permite una amplia gama de tareas comunes, como la autenticación, el almacenamiento en caché, la administración de usuarios y el enrutamiento RESTfull, que facilita las tareas a los desarrolladores de software.

Otras alternativas

Como ya se ha descrito anteriormente existen multitud de lenguajes que pueden posibilitar un desarrollo como el descrito en este proyecto de una manera clara y eficiente, por ejemplo Java, C# o incluso JavaScript mediante node.

²³ Modelo-vista-controlador (MVC)

(<https://es.wikipedia.org/wiki/Modelo%20%93vista%20%93controlador>) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario

4.4.3. Capa de presentación: Angular

La capa de presentación, por aspectos que ya se han tratado en este documento como la usabilidad, navegabilidad y la experiencia de usuario cobra tanta o más relevancia que la capa de lógica en este proyecto. La capa de presentación es la capa con la que interactúa directamente el usuario, y siguiendo criterios definidos anteriormente, se ha trabajado para que sea clara, con información fácilmente identificable y que no genere errores o por lo menos que los prevenga mediante las validaciones.

Para esta capa de presentación todas las alternativas que se han estudiado son frameworks que trabajan con JavaScript, en concreto se han estudiado de una manera profunda los frameworks Angular, Vue²⁴ y React²⁵.

	stars 🌟	forks 🍴	issues ⚠️
react	95112	17903	454
angular.js	58401	28928	583
vue	93210	13694	179
angular	35933	8700	2339

Tabla 4: Uso de las principales librerías JavaScript (GitHub)

Los tres frameworks ofrecen posibilidades de cubrir las necesidades que se plantean en el proyecto de una manera eficiente, por lo que la decisión de usar Angular para la programación de la capa de presentación ha estado basada en intereses personales del estudiante, ya que es un framework desconocido para él, que no se ha estudiado en ninguna asignatura del Grado y que tiene una fuerte demanda de profesionales en el mercado.

El framework Angular

Angular es un framework desarrollado por Google en 2010 enfocado a la creación y programación de aplicaciones web de una sola página (SPA). Está basado en el patrón MVC y al ser modular y escalable, permite crear nuevas etiquetas HTML personalizadas que pueden reutilizarse, además al

²⁴ Vue (<https://vuejs.org/>)

²⁵ React(<https://reactjs.org/>)

ser una framework creado e impulsado por una gran compañía como Google, tiene una gran cantidad de documentación y un muy buena comunidad de desarrolladores a su alrededor.

Las principales ventajas del uso de este framework son:

1. Es un sistema consistente. Permite construir aplicaciones Web robustas y es muy útil para la construcción de un sistema basado en SPA²⁶ como el de este proyecto.
2. Permite la creación de componentes, por lo que destaca la posibilidad de reutilización del código de los elementos creados.
3. Velocidad y rendimiento. Gracias al diseño basado en componentes reutilizables, no hay que estar continuamente recargando la página, solo los elementos que se necesitan para cada una de las vistas que componen la aplicación, esto hace que la aplicación se cargue rápidamente.
4. Productividad. Es un framework con uso a largo plazo, los editores IDE²⁷ como Visual Studio Code²⁸ obtienen sugerencias de código inteligente, detección de errores, etc. Mediante las herramientas de líneas de comando CLI²⁹, permite la generación de estructuras de código bastante avanzadas.
5. TypeScript como lenguaje de programación. TypeScript³⁰ es un superset de JavaScript que permite usar JavaScript de una forma más sencilla que utilizando JavaScript puro y que además compila directamente en JavaScript.
6. La demanda de Angular en el mercado es muy grande, hay una gran demanda de expertos en el mercado laboral.
7. Permite implementar REDUX³¹ para datos en tiempo real

²⁶ SPA, (<https://es.wikipedia.org/wiki/SPA>) single-page application, o aplicación de página única, es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan una sola vez o los recursos necesarios se cargan dinámicamente cuando lo requiera la página.

²⁷ IDE (https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) Un entorno de desarrollo integrado¹² o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

²⁸ Visual Studio (https://es.wikipedia.org/wiki/Visual_Studio_Code) Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows , Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias

²⁹ CLI (https://es.wikipedia.org/wiki/Interfaz_de_l%C3%ADnea_de_comandos) En español, interfaz de línea de comandos, es un método que permite a los usuarios dar instrucciones a algún programa informático por medio de una línea de texto simple. Debe notarse que los conceptos de CLI, shell y emulador de terminal no son lo mismo. Sin embargo los tres suelen utilizarse como sinónimos.

³⁰ TypeScript (<https://es.wikipedia.org/wiki/TypeScript>) TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases.

³¹ Redux ([https://es.wikipedia.org/wiki/Redux_\(JavaScript\)](https://es.wikipedia.org/wiki/Redux_(JavaScript))) es una librería JavaScript de código abierto para el manejo del estado de una aplicación. Es comúnmente usada con otras librerías como React o Angular para la construcción de Interfaces de Usuario. Dan Abramov y Andrew Clark se inspiraron en otra librería de Facebook, Flux 1 para crear Redux.

5. Implementación

5.1. Requisitos de instalación

En cuanto a los requisitos de instalación del software desarrollado, hay que tener en cuenta que se trata de un software basado en el stack de aplicaciones LAMP (Linux, Apache, MySQL, PHP). El stack usado es básico en cuanto a los requisitos de instalación del software ya que será necesario tener en cuenta los siguientes puntos:

- Sistema Operativo: el stack LAMP funciona correctamente y está optimizado para trabajar sobre un sistema operativo Linux, en este caso el recomendado es el Ubuntu en su versión 18.04 que a día de hoy es un sistema operativo muy popular, del que se dispone de mucha documentación y con un soporte a largo plazo de 5 años, dada la característica LTS (Long Term Support) de esta versión.
- Servidor Web: el servidor web usado es Apache, que cuenta con una larga y probada trayectoria como servidor web.
- Sistema de gestión de bases de datos MySQL en su versión 5.7
- Lenguaje de programación PHP en su versión 7.2.5 o superior.

Una vez realizada la instalación de la infraestructura básica necesaria, es necesario instalar una serie de paquetes que nos permitirán poner en marcha y actualizar fácilmente la aplicación.

- Composer³², que es un manejador de paquetes para PHP que proporciona un estándar para administrar, descargar e instalar dependencias y librerías.
- Npm, que es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript, a través del cual podemos obtener cualquier librería, está especializado en librerías JavaScript.
- Sistema de control de versiones GIT, mediante esta aplicación se tiene acceso a la última versión del software desarrollado.

Los requisitos de instalación presentados en cuanto a software son bastante comunes en los servicios de hosting más populares, son pues unos requerimientos que se pueden cumplir a un coste muy bajo. Además, cabe destacar que existe abundante documentación sobre la instalación y configuración de los sistemas necesarios para el funcionamiento de la aplicación.

³² Composer (<https://es.wikipedia.org/wiki/Composer>, <https://getcomposer.org/>) Composer es un sistema de gestión de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías de PHP. Fue desarrollado por Nils Adermann y Jordi Boggiano quienes continúan dirigiendo el proyecto. Ambos comenzaron el desarrollo en abril de 2011 y en marzo de 2012 presentaron la primera versión. Composer está inspirado en Node.js, npm y en Bundler Ruby.

5.2. Instrucciones de instalación

Para la instalación de la aplicación es necesario haber instalado previamente en el servidor la infraestructura necesaria y explicada en el punto anterior. A partir de esta instalación y para poder instalar y desplegar la aplicación, hay que seguir los siguientes pasos:

1. Copiar del repositorio el todo el código necesario.
2. Mediante el gestor de librerías de PHP 'composer', realizar la instalación del framework Laravel y de las librerías necesarias definidas como dependencias en el proyecto.
3. Una vez en marcha la aplicación Laravel que se ocupa de activar el backend de la aplicación, es necesario insertar los valores mínimos en la base de datos. Mediante comandos de artisan propios del framework Laravel es necesario:
 - Activar las migraciones, con el objetivo de crear la estructura de datos necesaria para la aplicación, mediante los siguientes comandos:
 - `php artisan migrate:install`
 - `php artisan migrate`
 - Poner en marcha los seed para tener los datos mínimos en la aplicación para poder poner en marcha esta, mediante el siguiente comando:
 - `php artisan db:seed`
4. Mediante el gestor de librerías de JavaScript 'npm', realizar la instalación del framework Angular y de las librerías definidas como dependencias en el proyecto.
5. Ejecutar la aplicación Angular, mediante el siguiente comando:
 - `ng serve`

6. Demostración

6.1. Instrucciones de uso

No son necesarias instrucciones de uso, si lo fueran, uno de los objetivos principales del proyecto, la facilidad de uso y una buena experiencia de usuario no estarían cubiertas por lo que habría que replantearse el diseño del proyecto.

6.2. Prototipos

Para trabajar el diseño y la maquetación de los elementos en la aplicación se han elaborado unas plantillas en forma de mock-ups como base para la maquetación de la aplicación. Los mock-up presentados, conceptualizan la estructura de la capa de presentación de la aplicación y sirven de guía para la distribución de los diferentes elementos en la página.

Al tratarse de una aplicación 'responsive', en su capa de presentación se tiene en cuenta desde que dispositivo se está utilizando la aplicación, por lo que se han desarrollado los mock-ups para los tamaños estándar de visualización en formato PC y formato Smartphone.

Se ha buscado mantener una estructura sencilla, priorizando la información visual frente a la textual, siendo significativo el uso de iconos para la representación de elementos de información.

La retícula principal consta de 3 columnas para la visualización en pantallas grandes y de una única columna para la visualización en Smartphone.

6.2.1. Mock-ups pantalla PC

Web pública - Home

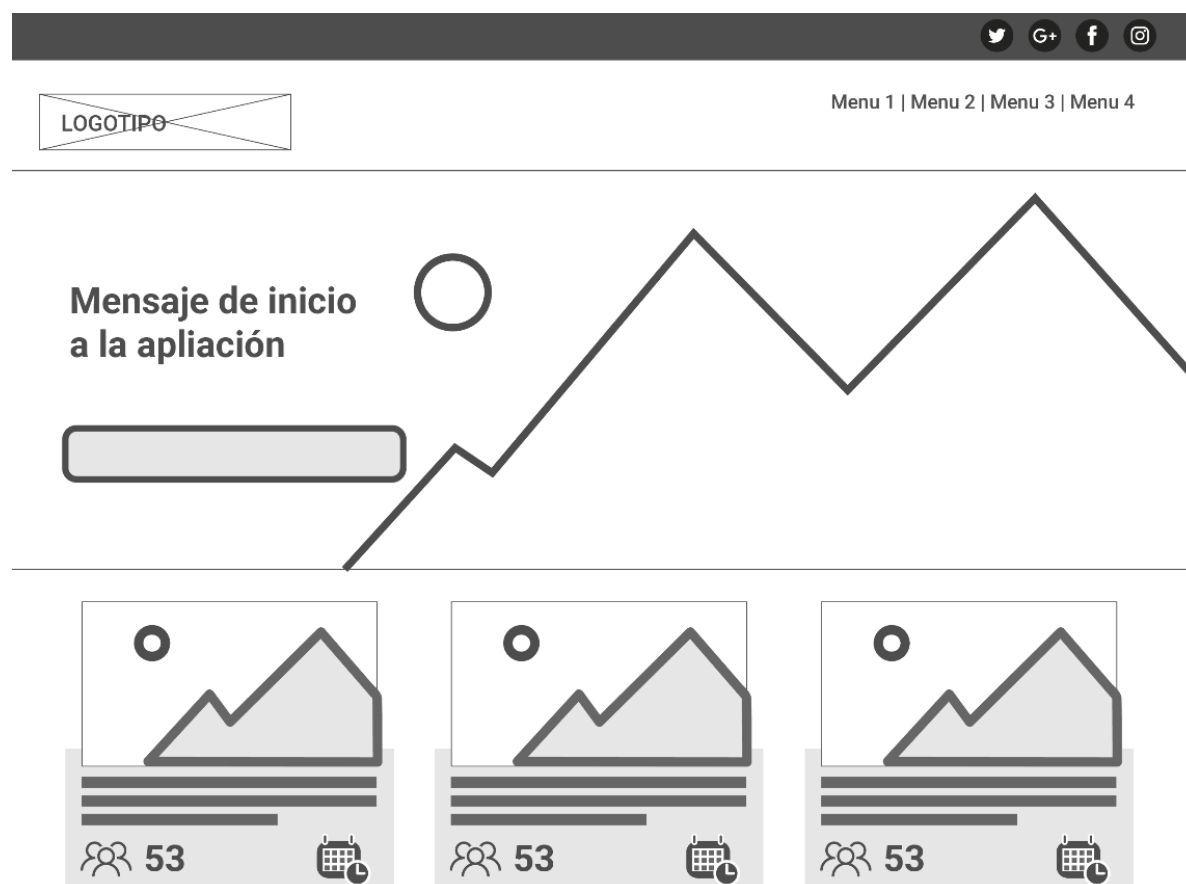


Figura 12: Mock-up Home, web publica página de inicio

La web pública tiene una doble función, por un lado contiene información general sobre el servicio, datos legales, contacto, etc... y por otro lado de informa al ciudadano de la oferta de salas disponibles, así como de alguna de sus características técnicas como el aforo.

Se observan 2 cabeceras una con los datos para el contacto en redes sociales y la segunda con un logotipo que acompaña a un menú con muy pocos elementos. Son éstos los que se usarán para dirigir al usuario hacia la información general del proyecto.

El cuerpo de la web lo componen, una imagen a modo de presentación visual del proyecto y un listado con algunos de los espacios disponibles.

Web pública – Listado de espacios

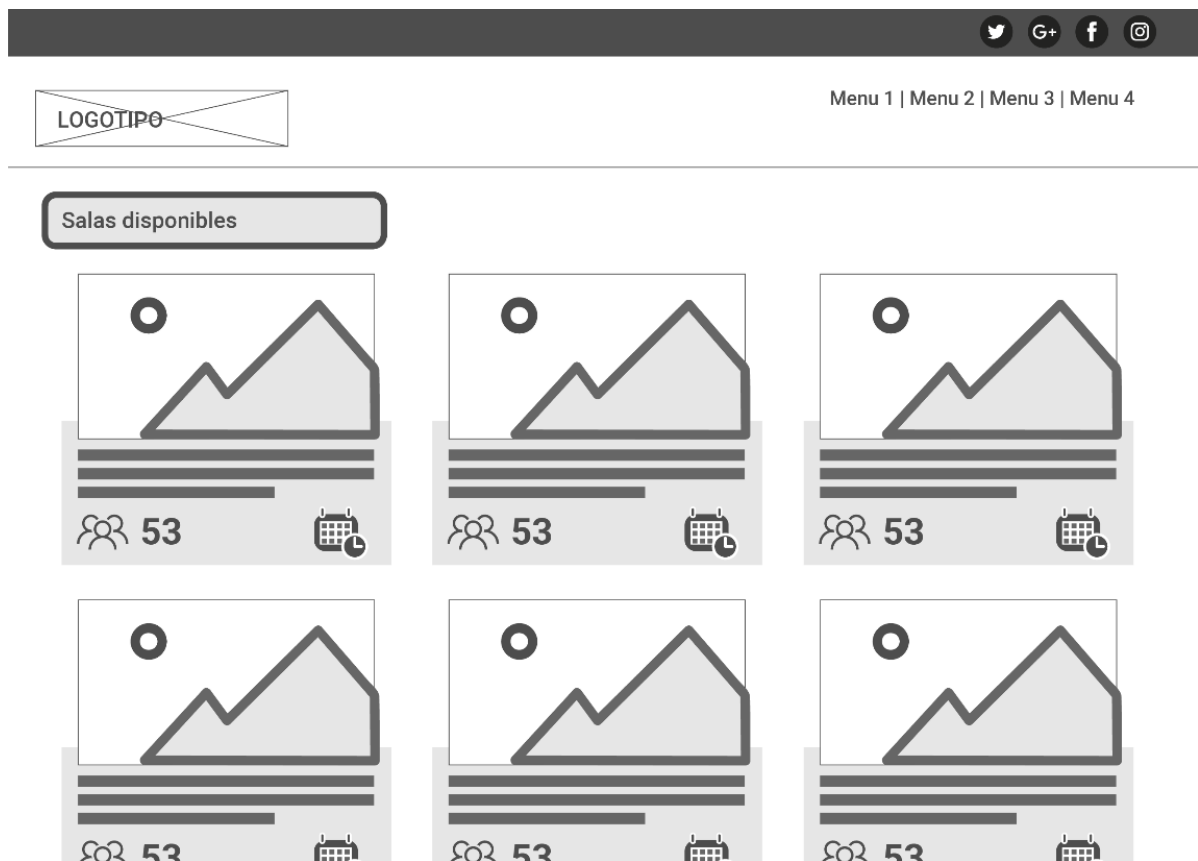


Figura 13: Mock-up Listado, web publica página con listado de salas

El listado de las salas es una continuación de la anterior página, la home, en este caso con el objetivo de ganar espacio visual se elimina la imagen superior que ocupa todo el ancho de la pantalla y se aumenta el número de bloques con información sobre las salas y acceso directo a estas.

En esta pantalla se resume la información básica de cada sala o espacio, mediante una imagen representativa de la misma, una breve descripción y la información (mediante el icono) del aforo máximo de la sala. El icono del calendario remite al link para ver la información de la sala concreta.

Web pública – Detalle

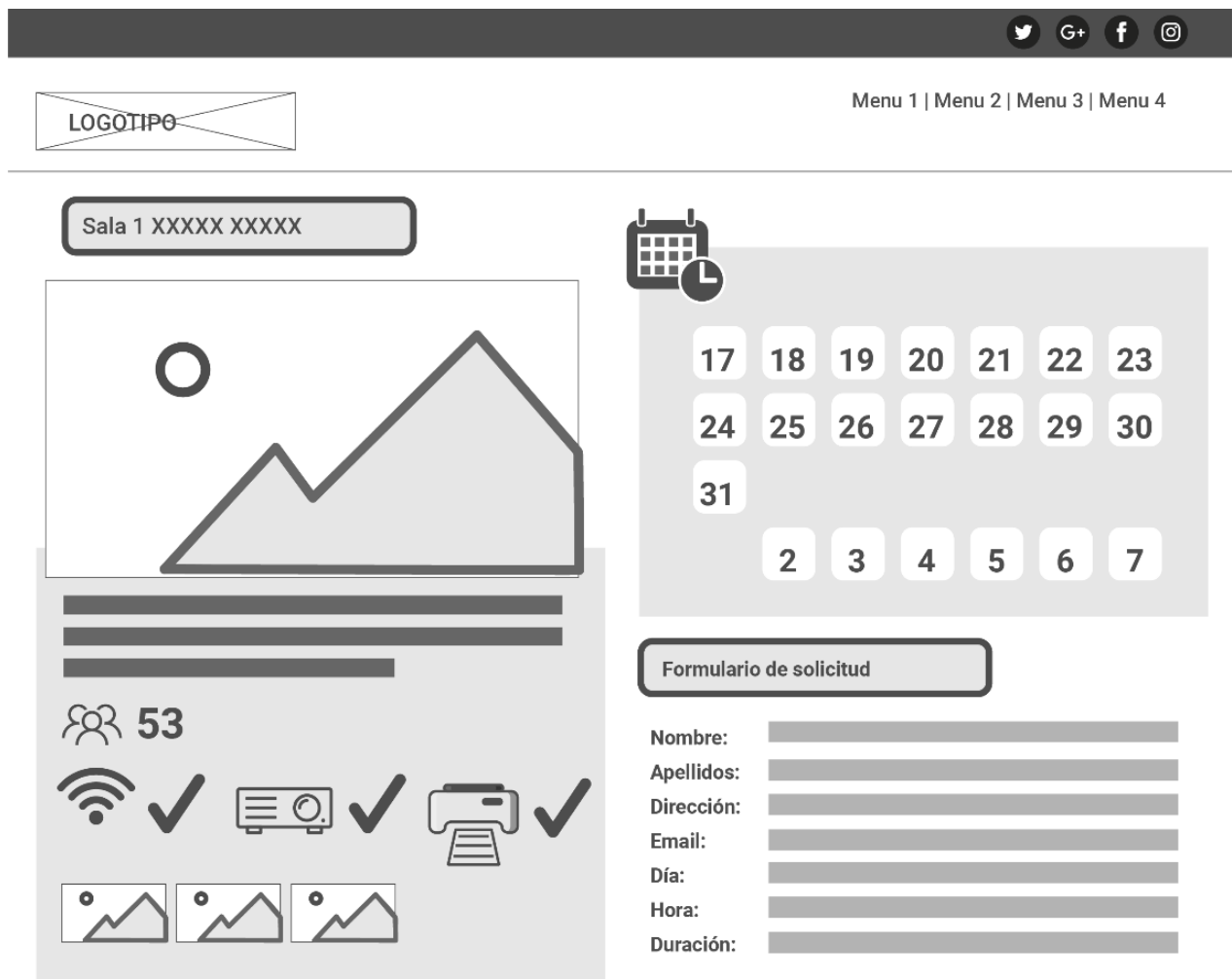






Figura 14: Mock-up Home, web detalle de un espacio

Esta página la componen los tres elementos básicos de información en el elemento sala o espacio disponible:


- Información sobre la sala, mediante imágenes e iconos que dan información sobre los servicios disponibles
- Información calendarizada sobre la ocupación de la sala
- Formulario para realizar la solicitud de la sala

La distribución de estos elementos en este mockup es confusa, solamente muestra los elementos a introducir en la pantalla, no es una distribución definitiva.

Zona administración – Login



LOGOTIPO



Login de usuario

LOGIN

Figura 15: Mock-up Admin. Login para los administradores

Las pantallas para los usuarios gestores están protegidas por contraseñas, para salvaguardar su uso de usuarios o accesos no autorizados por los clientes propietarios o usuarios del sistema de reservas.

Cada uno de los clientes puede dar de alta a tantos usuarios como necesite para gestionar los edificios, salas y solicitudes que reciba.

Zona administración – Listado

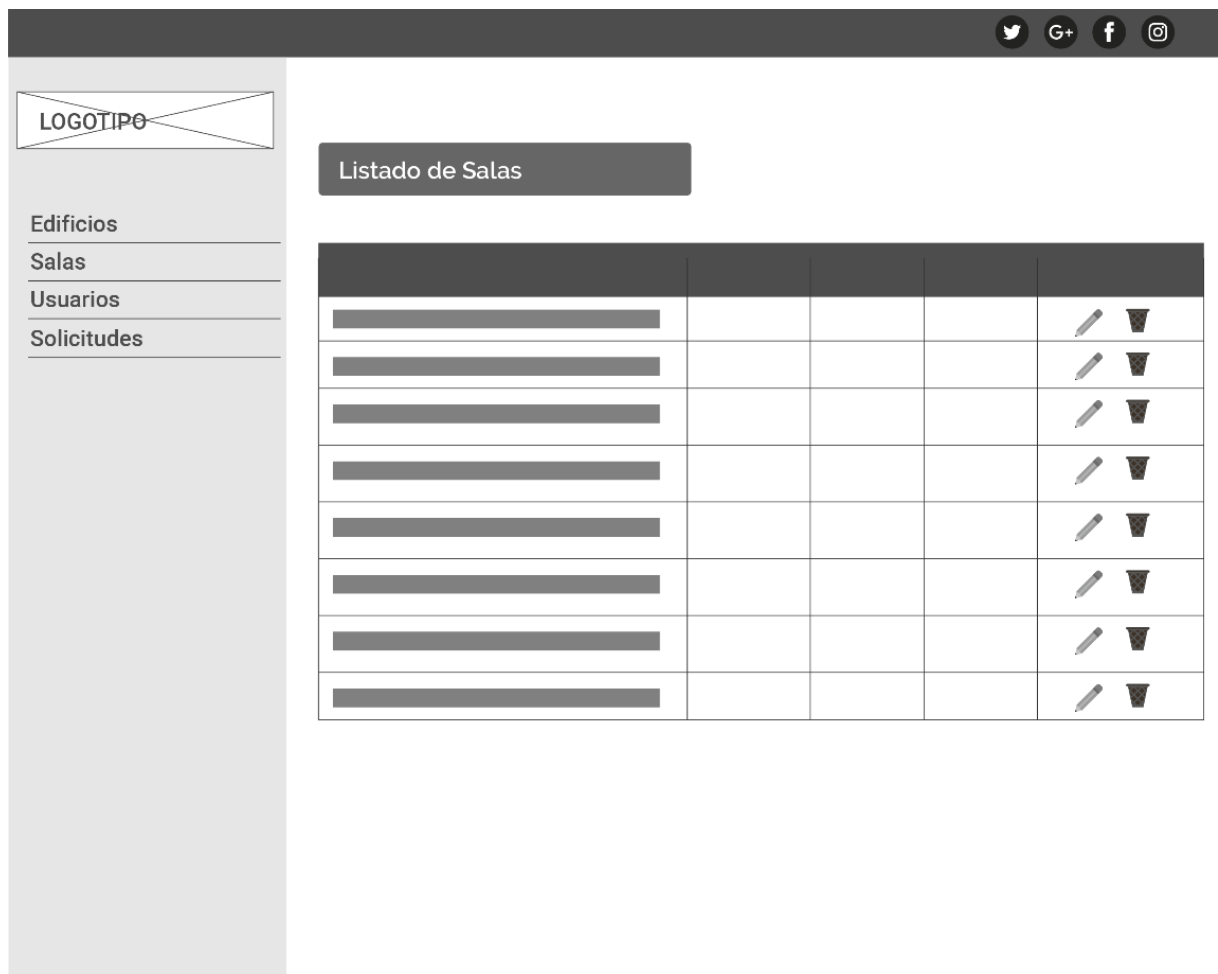


Figura 16: Mock-up Admin. Listado de salas, con opciones de editar y eliminar

Los listados son el formato de gestión básico en la parte desarrollada para los gestores de los espacios. Es un formato con un uso sencillo y fácil de comprender y utilizar.





Las opciones más comunes en todos los listados son los:

- Editar y realizar cambios sobre un elemento.
- Borrar un elemento.

En algunos casos como el de los usuarios, será posible realizar operaciones concretas como las siguientes desde el mismo listado de usuarios:

- Cambiar el avatar del usuario
- Cambiar el rol asignado al usuario

Zona administración – Detalle de solicitud



LOGOTIPO

Edificios

Salas

Usuarios

Solicitudes

Solicitud: sala XX. Cod.XXXXXX

Nombre:

Apellidos:

Dirección:

Email:

Día:

Hora:

Duración:





Figura 17: Mock-up Admin. Detalle de una solicitud

Las pantallas de detalle, muestran toda la información disponible sobre el elemento en cuestión.

Permiten realizar las siguientes operaciones:

- Modificar los datos recogidos sobre el elemento
- Añadir nuevos datos al elemento como imágenes
- Cambiar su estado, por ejemplo en el caso de las solicitudes

6.2.2. Mock-up Smartphone



Figura 18: Mock-up Smartphone. Pantallas home y detalle de una sala

La base de diseño utilizada para la realización de las aplicaciones para los tipos de usuarios administradores y usuario ciudadano, es un diseño basada en el diseño adaptable o responsivo, que permite la visualización de toda la información disponible en la web en todos los soportes.

En este caso el diseño para Smartphone tiene su origen en el diseño pensado para pantallas grandes. El uso del diseño adaptable ha permitido que teniendo en cuenta ciertas limitaciones, se pueda mostrar toda la información disponible para cada pantalla en varios soportes de diferentes tamaños, sin tener que programar diseños específicos para cada dispositivo.

En los mock-ups que se muestran se ha eliminado la información superficial como la imagen que ocupa todo el ancho de la pantalla Home en pantallas grandes.

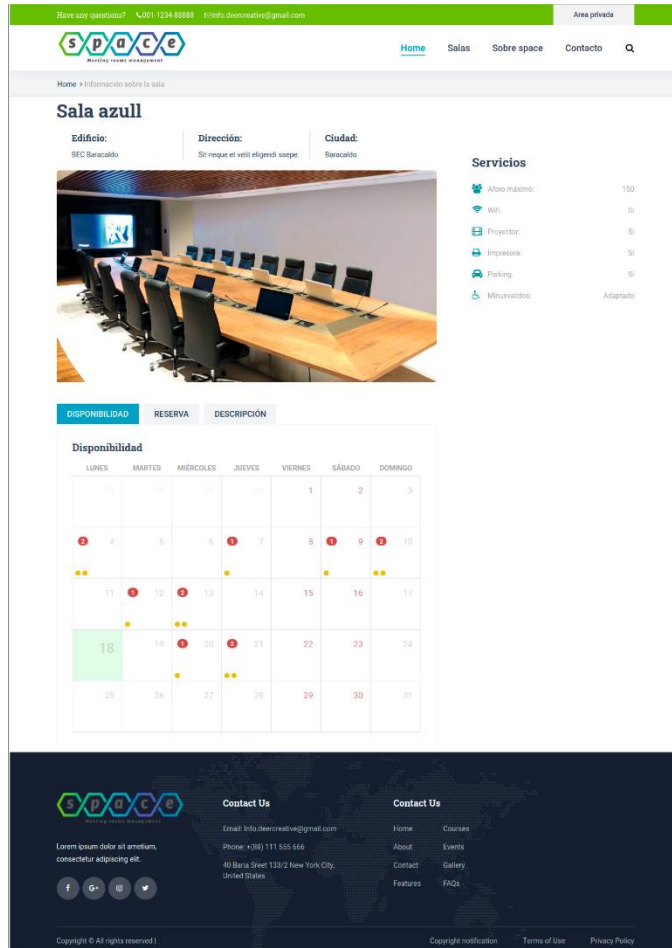
Por otro lado realizar el diseño con una base responsiva, permite la visualización de toda la información disponible de la sala en un formato vertical.

En cuanto a los menús, se mantiene la opción de navegar mediante los menús sobre la información general del proyecto.

6.3. Ejemplos de uso del producto

6.3.1. Ejemplo de realizar una solicitud de una reserva

Paso 1



Un usuario ciudadano ingresa en la web, y tras estudiar las características de las salas disponibles, puede elegir una sala, para ver la disponibilidad de esta:

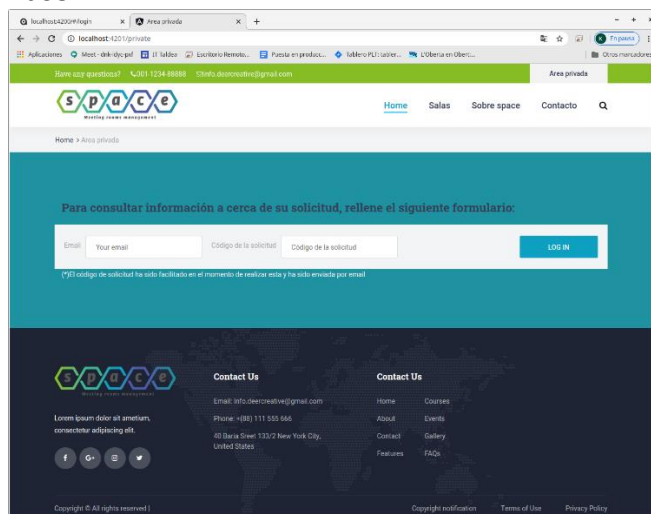
1. En la zona superior puede observar las características de la sala en cuanto a su localización.

2. En la zona derecha puede ver los servicios de los que dispone una sala

3. En la zona inferior, debajo de la fotografía puede ver en un calendario las solicitudes que ya han sido aceptadas por los gestores de la sala, es decir, puede evaluar la disponibilidad de esta.

4. Pulsando en la pestaña “Reserva” y rellenando el formulario podrá realizar una solicitud de reserva para esta sala, entre una fecha y hora de inicio y una fecha y hora de fin.

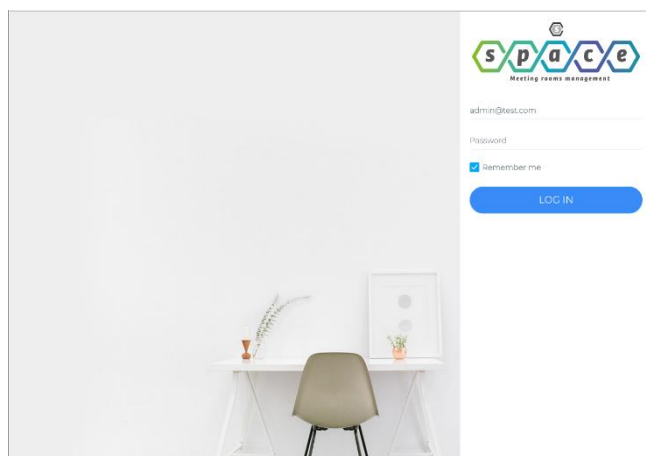
Paso 2



Una vez realizada la reserva, el usuario recibirá un código de seguimiento de la reserva, mediante la cual y la dirección de email que hay usado para realizar la solicitud podrá ingresar en el área privada y ver el estado de esta.

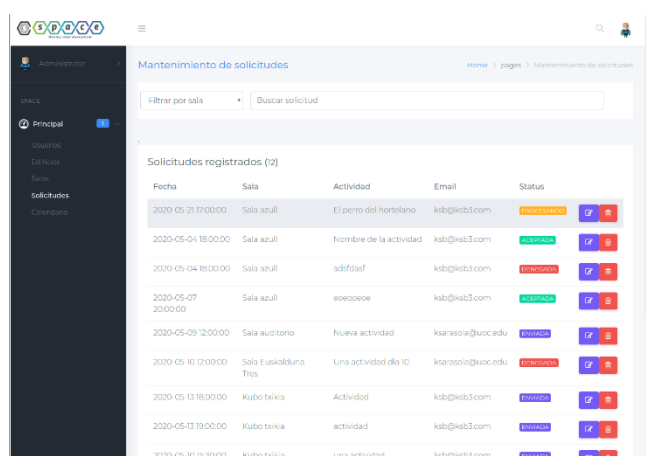
6.3.2. Ejemplo de evaluación una solicitud realizada por un ciudadano

Paso 1



El usuario gestor debe de autenticarse para ingresar en la aplicación del panel de administración.

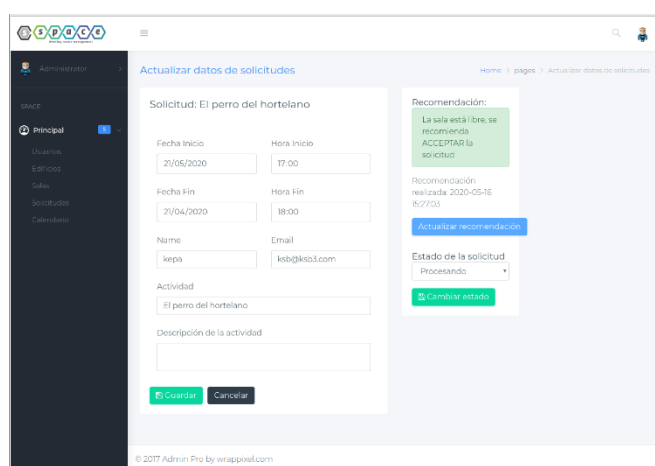
Paso 2



En la pantalla de solicitudes, puede ver las solicitudes asociadas a las salas que le correspondan según su rol de usuario.

Puede filtrar las solicitudes por salas y realizar búsquedas textuales en los campos de nombre, email o actividad.

Paso 3



En la pantalla de la solicitud existe la opción de recibir recomendaciones sobre la situación de la solicitud, pulsando un botón: Estas recomendaciones se basan en la situación actual de las solicitudes, es decir, evalúa la existencia de otras solicitudes que coincidan en fechas de inicio y fin con esta. Es decir localiza si existen conflictos entre las solicitudes

7. Conclusiones y líneas de futuro

7.1. Conclusiones

Conclusiones del trabajo:

El desarrollo de este proyecto me ha permitido integrar muchas de los conocimientos adquiridos en el grado, sobre todo aquellos relacionados con la programación, diseño de bases de datos, diseño de interacción, desarrollo de proyectos etc. Siendo esto así, he de decir, que además de los conocimientos ya trabajados en el grado he tenido que realizar una labor importante de formación en lo que se refiere a la programación Front y Back que apenas se ven de una forma bastante básica en el grado.

La realización de una aplicación de estas características y con una vocación clara de convertirse en una aplicación comercial, es decir, cuyo nicho de mercado hemos localizado y que su uso podría ser bastante generalizado, genera una cantidad tal de detalle que creo necesario la implicación de un equipo de desarrollo para poder llevar la aplicación hasta un punto en el que pueda considerarse una aplicación completa.

Esta es bajo mi punto de vista la lección más importante, partiendo como se ha partido en mi caso de una idea que se asocia con una necesidad real, no es para nada sencillo realizar una aplicación que resuelva el alcance planteado en el inicio, de una manera profesional, en el plazo de tiempo establecido para este proyecto y por una sola persona.

Reflexión crítica sobre el logro de los objetivos planteados inicialmente

Respecto de los objetivos planteados inicialmente, creo que tanto los objetivos generales, como los principales objetivos de la aplicación se han cumplido en el desarrollo del proyecto.

Se han desarrollado dos aplicaciones dirigidos a dos tipos de usuarios muy concretos, con el objetivo de dotar a cada aplicación de las funcionalidades necesarias para el cumplimiento de los objetivos por parte de los usuarios gestores y los ciudadanos. Las dos aplicaciones desarrolladas cumplen los objetivos planteados en el inicio que de manera resumida se concretan en:

- Poner a disposición del ciudadano información acerca de la disponibilidad de diferentes espacios públicos, con información real y actualizada.
- Facilitar la gestión compartida de las solicitudes recibidas a los gestores de los espacios.
- Asegurar que la información que maneja la aplicación pueda ser compartida por otras aplicaciones que puedan ser interesantes para las entidades que gestionan dichos espacios.

Con los objetivos principales cumplidos, creo que existen partes en el proyecto que merecen ser trabajadas especialmente con el objetivo de mejorar la usabilidad de este. Pienso que se ha logrado una buena base, pero que es posible trabajar para mejorar la usabilidad de la aplicación, sobre todo en cuanto a la aplicación enfocada a los gestores de las salas

Análisis crítico del seguimiento de la planificación y metodología a lo largo del proyecto

Se ha seguido la planificación realizada al comienzo del proyecto, completando al final de cada fase principal definida (asociada a la entrega de cada PEC) los objetivos planteados para ella.

La metodología de desarrollo, basada en un enfoque 'agile' ha permitido tener partes del proyecto con funcionalidades concretas completadas y puestas en marcha, que han sido ampliadas posteriormente por otras funcionalidades, pero manteniendo siempre el proyecto en marcha.

Cambios en el proyecto

La metodología utilizada ha permitido entre otras cosas identificar la necesidad de desarrollar dos aplicaciones Frontend, una enfocada para los gestores de la aplicación y otra enfocada al ciudadano, ya que durante el propio desarrollo del proyecto se ha ido definiendo la necesidad de diferenciar las aplicaciones, buscando cumplir objetivos diferentes para usuarios diferentes.

Este ha sido el cambio más grande realizado durante el proyecto con respecto a la primera idea de este y a la planificación realizada. Aunque las necesidades de ambos tipos de usuario se podrían haber solucionado mediante una única aplicación de Frontend, finalmente he decidido desdoblar esta en dos basándome en los siguientes criterios:

- Claramente existen dos tipos de usuario diferentes, los usuarios gestores y los ciudadanos.
- Los dos tipos de usuario tienen objetivos diferentes.
- Los usuarios de un tipo no interactúan con las funcionalidades que están asociadas al otro tipo de usuario.
- Existe la necesidad de crear espacios privados (asociados a un proceso de autenticación) en los dos casos, pero las necesidades son diferentes y en consecuencia los espacios también han de serlo.
- Un usuario gestor puede interactuar con la aplicación en un momento dado como lo haría un usuario ciudadano, es decir, según el contexto podría ser los dos tipos de usuario en el mismo tiempo, por lo que es importante diferenciar el espacio.

Bajo mi punto de vista el diferenciar estos dos espacios en dos aplicaciones diferentes ha sido un acierto que permitirá en un futuro realizar un desarrollo más específico y diferenciado para cada tipo de usuario.

7.2. Líneas de futuro

Desarrollo de la lógica para comercializar la aplicación en modo SaaS

En cuanto a las ampliaciones o líneas de futuro del proyecto, destacaría que la aplicación aunque está enfocada a poder ser comercializada con un enfoque SaaS, en verdad falta desarrollar ciertos aspectos muy importantes para que esta opción fuera real, entre ellas, el pago de la suscripción, el control de la caducidad de esta, diferentes opciones para la suscripción, etc.

Mejora de las opciones de comunicación gestor-ciudadano

Respecto de las funcionalidades desarrolladas hasta ahora, me parece importante profundizar más en la comunicación entre el usuario gestor y el usuario ciudadano, desarrollando opciones que permitan avanzar en el 'área privada' de la aplicación web y que permitan al ciudadano cambiar los datos de la solicitud, realizar preguntas a los gestores y en general desarrollar vías de comunicación mediante la plataforma web entre los dos tipos de usuarios.

En cuanto a la comunicación, ha quedado sin desarrollar el envío del email confirmando la recepción de la solicitud, y como línea de futuro, creo que una funcionalidad que permitiese otros tipos de comunicación como SMS, Telegram, etc. aportaría a la aplicación una versatilidad importante.

Mejoras en las recomendaciones

En el desarrollo del proyecto se ha trabajado en un sistema de recomendaciones, que ayuda al gestor a evaluar la situación concreta de una solicitud, señalándole, si existen otras solicitudes que pueden entrar en conflicto con la que se está tratando.

Esta opción es muy interesante y merecería la pena trabajarla más, para que en el caso de no poder aprobar una solicitud porque el espacio ya estuviera reservado para una fecha concreta, el sistema pudiera dar otras opciones de fechas o salas alternativas a la solicitud original y realizar propuestas que el usuario gestor podría enviar al ciudadano que ha realizado la solicitud.

Nuevas opciones de reserva de sala: reserva repetitiva

Una opción interesante a este respecto es la de realizar un tipo de reserva repetitiva en el tiempo. Esta opción es muy interesante en el caso de querer reservar un espacio para ser usado en cursos, entrenamientos, etc... que pueden repetirse por ejemplo semanalmente, dos veces por semana, etc.

Esta opción aunque complica bastante la aplicación del gestor sería muy interesante para el tipo de usuario ciudadano, ya que le posibilitaría realizar la solicitud para un curso académico completo de mediante una única solicitud.

Iteración con otros sistemas: llave digital

Esta es una oportunidad para ampliar el proyecto permitiendo la iteración con otros sistemas que pueden ser complementarios con este. En concreto y teniendo en cuenta que este proyecto gira en torno a la gestión de salas, es bastante lógico pensar que una ampliación podría darse mediante el uso de llaves digitales que podrían activarse y desactivarse mediante la iteración de esta aplicación con otros sistemas.

Gestión de contenidos de la web

Una parte de la web pública que no se ha desarrollado, ha sido la correspondiente a la gestión de los contenidos multimedia y contenidos textuales de la web, por ejemplo, los textos explicativos del producto, la parte de contacto o la posibilidad de cambiar las imágenes del slider de la pantalla de inicio.

Bibliografía

Aguilar Luis, Medium website: <https://medium.com/@insomniocode/angular-autenticaci%C3%B3n-usando-interceptors-a26c167270f4> consultado 04/05/2020

Angular documentación website: <https://angular.io/docs> consultado 9/03/2020

Castelo André, website: <https://www.toptal.com/laravel/restful-laravel-api-tutorial> consultado el 7/04/2020

Dhiraj, website: <https://www.devglan.com/angular/angular-8-crud-example> consultado 23/02/2020

Dhiraj, website (github): <https://github.com/only2dhir/angular8-demo> consultado 23/02/2020

Freepick website: <https://www.freepik.com/> consultado 22/04/2020

Harihara Subramanian, Pethuru Raj (Packt Publishing, enero 2019) *Hands-On RESTful API Design Patterns and Best Practices*

Herrera Fernando Youtube: <https://www.youtube.com/channel/UCuaPTYj15JSkETGnEseaFFq> consultado 6/04/2020

Hristozov Krasimir, website: <https://dev.to/oktadev/tutorial-build-a-basic-crud-app-with-laravel-and-angular-1hb3> consultado 08/03/2020

Laravel website: <https://laravel.com/> consultado 18/05/2020

MySQL Reference Manual website: <https://dev.mysql.com/doc/refman/5.7/en/introduction.html>, consultado 12/05/2020

PHP Manual website: <https://www.php.net/manual/en/index.php> consultado 8/04/2020

Ramirez Londoño, Youtube: <https://www.youtube.com/watch?v=IUbpifsnLm8> consultado 13/04/2020

Riedweg Martin, website: <https://medium.com/@martin.riedweg/laravel-5-7-api-authentication-with-laravel-passport-92b909e12528> consultado 9/04/2020

Stauffer, Matt (O'Reilly Media, abril 2019) *Laravel: Up & Running, 2nd Edition*

Vallejo Claudio, website: <https://medium.com/@cvallejo/sistema-de-autenticaci%C3%B3n-api-rest-con-laravel-5-6-572a16e3929b> consultado 20/04/2020

Zama Khan Mohammed (Packt Publishing, enero 2019) *Angular Projects*

Anexos

Anexo A: Glosario

Angular: es un framework de JavaScript de código abierto, mantenido por google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

(<https://angular.io/>)

API-Rest: Es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo (Fuente: Wikipedia)

Desarrollo ágil de software: El desarrollo de software ágil es un concepto usado en el desarrollo de software para describir las metodologías de desarrollo incrementales (Cohen, Lindvall & Costa, 2003). Es una alternativa en la gestión tradicional de proyectos TI, donde se hace hincapié en el empoderamiento de las personas para colaborar y tomar decisiones en equipo, además potencia la planificación continua, pruebas permanentes y la integración conjunta del código y los despliegues.

Diseño responsive: es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas.

(https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable)

HTTP: el Protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, siendo el más importante de ellos el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

(https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto)

IDE: Un entorno de desarrollo integrado¹² o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software

(https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado)

Laravel: es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET.

(laravel.com)

MVC: Modelo-vista-controlador es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

(<https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>)

PHP: PHP es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales web están creadas con PHP. Código abierto significa que es de uso libre y gratuito para todos los programadores que quieran usarlo. Incrustado en HTML significa que en un mismo archivo vamos a poder combinar código PHP con código HTML, siguiendo unas reglas.

(<https://www.php.net/manual/es/intro-what-is.php>)

Requisitos de usabilidad: se refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. La usabilidad también puede referirse al estudio de los principios que hay tras la eficacia percibida de un objeto.

(<https://es.wikipedia.org/wiki/Usabilidad>)

SPA: single-page application, o aplicación de página única, es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan una sola vez o los recursos necesarios se cargan dinámicamente cuando lo requiera la página, normalmente como respuesta a las acciones del usuario.

(<https://es.wikipedia.org/wiki/SPA>)

SQL: SQL es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

(<https://es.wikipedia.org/wiki/SQL>)

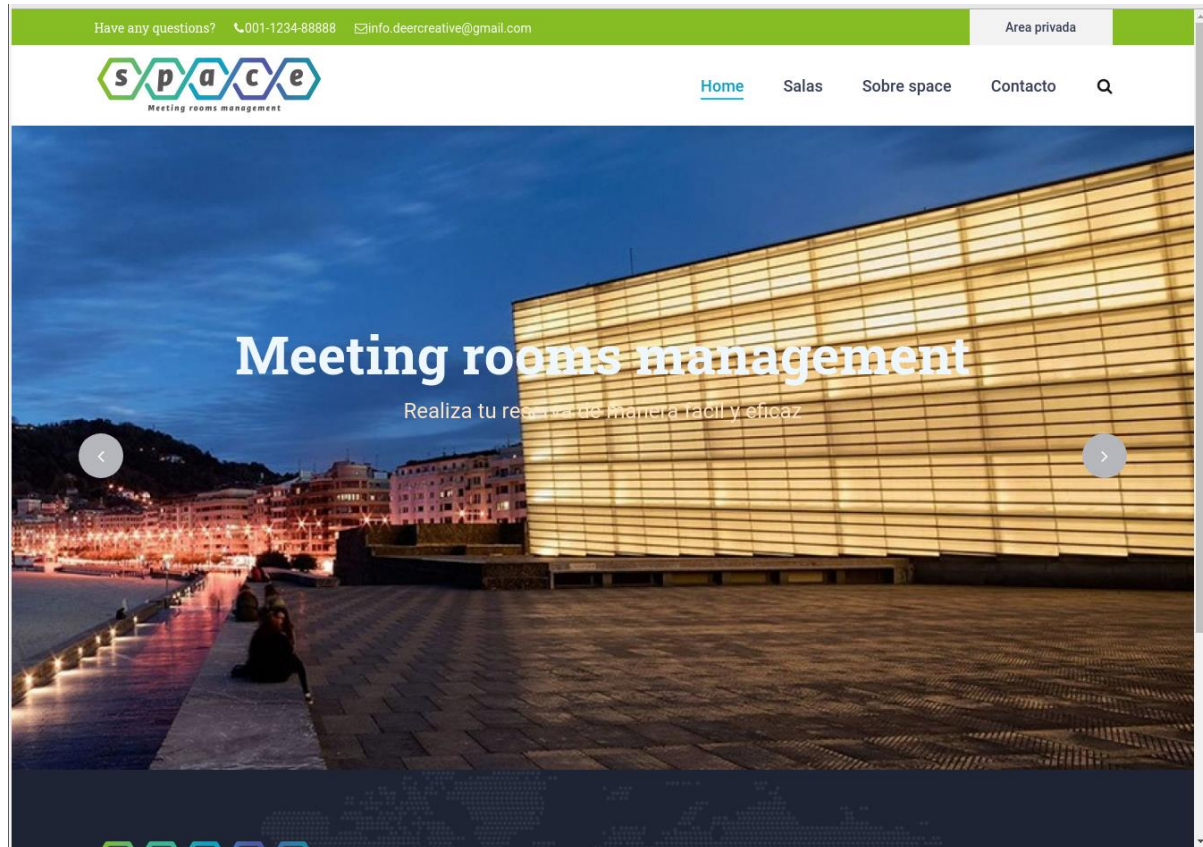
Anexo B: Entregables del proyecto

- **Carpeta “tfg-backend”:** corresponde a la aplicación codificada mediante el framework Laravel y que realiza las funciones de backend, desplegando una aplicación que tiene como resultado un interface basado en la especificación API-Rest.
- **Carpeta “tfg-frontend”:** corresponde a la aplicación Angular creada como panel de administración, con el objetivo de que los usuarios gestores de las salas puedan interactuar con el sistema y los datos que aloja.
- **Carpeta “tfg-web”:** contiene la aplicación de Angular encargada de desplegar la web pública del proyecto desde la que el usuario puede consultar la disponibilidad de las salas, así como realizar las solicitudes asociadas a estas.
- **Fichero “backend.sql”:** contiene un backup de la base de datos con datos ficticios que se ha usado para pruebas, con el objetivo de que se pueda probar la aplicación.

Anexo C: Capturas de pantalla

Capturas pantalla web


Homepage: el objetivo es visualizar de una manera moderna y agradable el proyecto. Incluye un slide de imágenes para ilustrar los diferentes edificios y salas que contiene la aplicación




Salas: índice de las salas disponibles en la aplicación con información resumida.

Have any questions? 001-1234-8888 info.deercreative@gmail.com

Area privada




[Home](#)[Salas](#)[Sobre space](#)[Contacto](#)

Home > Salas disponibles


All Categories ▾


SEARCH NOW



Sala azull


Lorem ipsum dolor sit amet, consectetur
adipi elit sed do eiusmod tempor


 150



Kubo txikia


Lorem ipsum dolor sit amet, consectetur
adipi elit sed do eiusmod tempor

 250



Auditorio Baluarte

Lorem ipsum dolor sit amet, consectetur
adipi elit sed do eiusmod tempor

 750

89

Salas: información completa de la sala, acceso al calendario para consultar disponibilidad y la opción de realizar una reserva.

Have any questions? 001-1234-8888 info.deercreative@gmail.com

Area privada

s p a c e

Meeting rooms management

[Home](#) [Salas](#) [Sobre space](#) [Contacto](#) [Q](#)


Home > Información sobre la sala

Sala azull

Edificio:
BEC Baracaldo

Dirección:
Sit neque et velit eligendi saepe.

Ciudad:
Baracaldo



Servicios

Aforo máximo:	150
Wifi:	Si
Proyector:	Si
Impresora:	Si
Parking:	Si
Minusvalidos:	Adaptado

DISPONIBILIDAD

RESERVA

DESCRIPCIÓN

Disponibilidad

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁBADO	DOMINGO
27	28	29	30	1	2	3
2 4	5	6	1 7	8	1 9	2 10
11	1 12	2 13	14	15	16	17
18	19	1 20	2 21	22	23	24
25	26	27	28	29	30	31

s p a c e

Meeting rooms management

Lorem ipsum dolor sit ametium,
consectetur adipiscing elit.

f

G+

ig

tw

Contact Us

Email: info.deercreative@gmail.com

Phone: +(88) 111 555 666

40 Baria Sreet 133/2 New York City,
United States

Contact Us

Home

Courses

About

Events

Contact

Gallery

Features

FAQs

Copyright © All rights reserved |

Copyright notification

Terms of Use

Privacy Policy

Area privada:

The screenshot shows a web browser window with the URL `localhost:4200/#/login` and a tab titled 'Area privada'. The page has a green header with contact information: 'Have any questions? 001-1234-88888 info.deercreative@gmail.com'. Below the header is a navigation bar with the 'space' logo and links for 'Home', 'Salas', 'Sobre space', and 'Contacto'. A breadcrumb trail shows 'Home > Area privada'. The main content area has a teal background with the text 'Para consultar información a cerca de su solicitud, rellene el siguiente formulario:'. Below this is a login form with fields for 'Email' (containing 'Your email'), 'Código de la solicitud' (containing 'Código de la solicitud'), and a 'LOG IN' button. A note below the form states: '(*)El código de solicitud ha sido facilitado en el momento de realizar esta y ha sido enviada por email'. The footer is dark blue with the 'space' logo, placeholder text 'Lorem ipsum dolor sit amet...', social media icons, and contact details: 'Email: info.deercreative@gmail.com', 'Phone: +(88) 111 555 666', and '40 Baria Sreet 133/2 New York City, United States'. A sidebar menu on the right lists 'Home', 'Courses', 'About', 'Events', 'Contact', 'Gallery', 'Features', and 'FAQs'. The bottom of the page contains copyright and policy links: 'Copyright © All rights reserved | Copyright notification Terms of Use Privacy Policy'.

localhost:4200/#/login x Area privada x +

localhost:4201/private

Aplicaciones Meet - dnk-idyc-pxf IT Taldea Escritorio Remoto... Puesta en producc... Tablero PLT: tabler... L'Oberta en Obert... Otros marcadores

Have any questions? 001-1234-88888 info.deercreative@gmail.com Area privada

space Meeting rooms management Home Salas Sobre space Contacto Q

Home > Area privada

Para consultar información a cerca de su solicitud, rellene el siguiente formulario:

Email Your email Código de la solicitud Código de la solicitud LOG IN

(*)El código de solicitud ha sido facilitado en el momento de realizar esta y ha sido enviada por email

space Meeting rooms management

Lorem ipsum dolor sit ametum, consectetur adipiscing elit.

f G+ @

Contact Us

Email: info.deercreative@gmail.com

Phone: +(88) 111 555 666

40 Baria Sreet 133/2 New York City, United States

Contact Us

Home Courses

About Events

Contact Gallery

Features FAQs

Copyright © All rights reserved | Copyright notification Terms of Use Privacy Policy

Anexo D: Ejemplos respuesta JSON

GET: User (<http://localhost:8000/api/user>)

```
{
  "data": [
    {
      "id": 1,
      "customer_id": 1,
      "name": "Administrator",
      "email": "admin@test.com",
      "image": "1588955764.png",
      "role": "CUSTOMER"
    },
    {
      "id": 2,
      "customer_id": 1,
      "name": "Brandt Cormier",
      "email": "dale83@gmail.com",
      "image": "1588955806.png",
      "role": "BUILDING_ADMIN"
    },
    {
      "id": 3,
      "customer_id": 1,
      "name": "Cole Connelly1",
      "email": "scot20@yahoo.com",
      "image": "1590425508.png",
      "role": "ROOM_ADMIN"
    },
    {
      "id": 4,
      "customer_id": 1,
      "name": "Duncan Greenholt",
      "email": "gwilderma@jast.biz",
      "image": "1590425541.png",
      "role": "ROOM_ADMIN"
    },
    {
      "id": 5,
      "customer_id": 1,
      "name": "Hermann Treutel DVM",
      "email": "lenna.murazik@yahoo.com",
      "image": "1590426850.png",
      "role": "ROOM_ADMIN"
    }
  ],
  "status": 200,
  "links": {
    "first": "http://localhost:8000/api/user?page=1",
    "last": "http://localhost:8000/api/user?page=3",
    "prev": null,
    "next": "http://localhost:8000/api/user?page=2"
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 3,
    "path": "http://localhost:8000/api/user",
    "per_page": 5,
    "to": 5,
    "total": 13
  }
}
```

GET: Building (http://localhost:8000/api/building)

```

{
  "data": [
    {
      "id": 1,
      "admin_id": 1,
      "customer_id": 0,
      "rooms": [
        {
          "id": 1,
          "admin_id": 4,
          "building_id": 1,
          "name": "Sala azull",
          "capacity": 150,
          "image": "1588955866.jpeg",
          "s_wifi": 1,
          "s_projector": 1,
          "s_printer": 0,
          "s_parking": 0,
          "s_disabled": 1
        }
      ],
      "name": "BEC Baracaldo",
      "address": "Sit neque et velit eligendi saepe.",
      "city": "Baracaldo",
      "image": "1588955894.jpeg",
      "created_at": "2020-04-20T15:27:11.000000Z",
      "updated_at": "2020-05-08T16:38:14.000000Z"
    },
    {
      "id": 2,
      "admin_id": 1,
      "customer_id": 0,
      "rooms": [
        {
          "id": 2,
          "admin_id": 0,
          "building_id": 2,
          "name": "Kubo txikia",
          "capacity": 250,
          "image": "1589019034.jpeg",
          "s_wifi": 1,
          "s_projector": 1,
          "s_printer": 0,
          "s_parking": 1,
          "s_disabled": 1
        }
      ],
      "name": "Palacio de Congresos Kursall",
      "address": null,
      "city": "San Sebastian",
      "image": "1589018893.jpeg",
      "created_at": "2020-05-09T12:05:51.000000Z",
      "updated_at": "2020-05-09T10:08:13.000000Z"
    },
    {
      "id": 3,
      "admin_id": 1,
      "customer_id": 0,
      "rooms": [
        {
          "id": 5,
          "admin_id": 0,
          "building_id": 3,

```

```

        "name": "Sala Euskalduna Tres",
        "capacity": 125,
        "image": "1589022461.jpeg",
        "s_wifi": 1,
        "s_projector": 1,
        "s_printer": 1,
        "s_parking": 1,
        "s_disabled": 1
    },
    {
        "name": "Palacio Euskalduna",
        "address": null,
        "city": "Bilbo",
        "image": "1589018995.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:55.000000Z"
    },
    {
        "id": 4,
        "admin_id": 1,
        "customer_id": 0,
        "rooms": [
            {
                "id": 4,
                "admin_id": 0,
                "building_id": 4,
                "name": "Sala auditorio",
                "capacity": 450,
                "image": "1589021396.jpeg",
                "s_wifi": 1,
                "s_projector": 1,
                "s_printer": 1,
                "s_parking": 1,
                "s_disabled": 1
            }
        ],
        "name": "Palacio Europa",
        "address": null,
        "city": "Vitoria-Gasteiz",
        "image": "1589018929.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:08:49.000000Z"
    },
    {
        "id": 5,
        "admin_id": 2,
        "customer_id": 0,
        "rooms": [
            {
                "id": 3,
                "admin_id": 0,
                "building_id": 5,
                "name": "Auditorio Baluarte",
                "capacity": 750,
                "image": "1589019147.jpeg",
                "s_wifi": 1,
                "s_projector": 1,
                "s_printer": 1,
                "s_parking": 1,
                "s_disabled": 1
            }
        ],
        "name": "Baluarte",
        "address": null,
        "city": "Iruñea",
    }

```

```

        "image": "1589018942.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:02.000000Z"
    }
},
"status": 200,
"links": {
    "first": "http://localhost:8000/api/building?page=1",
    "last": "http://localhost:8000/api/building?page=1",
    "prev": null,
    "next": null
},
"meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "path": "http://localhost:8000/api/building",
    "per_page": 15,
    "to": 5,
    "total": 5
}
}

```

GET: Room (<http://localhost:8000/api/room>)

```

{
    "data": [
        {
            "id": 1,
            "admin_id": 4,
            "building_id": 1,
            "building": {
                "id": 1,
                "admin_id": 1,
                "customer_id": 0,
                "name": "BEC Baracaldo",
                "address": "Sit neque et velit eligendi saepe.",
                "city": "Baracaldo",
                "image": "1588955894.jpeg",
                "created_at": "2020-04-20T15:27:11.000000Z",
                "updated_at": "2020-05-08T16:38:14.000000Z"
            },
            "name": "Sala azul1",
            "capacity": 150,
            "image": "1588955866.jpeg",
            "s_wifi": 1,
            "s_projector": 1,
            "s_printer": 0,
            "s_parking": 0,
            "s_disabled": 1
        },
        {
            "id": 2,
            "admin_id": 0,
            "building_id": 2,
            "building": {
                "id": 2,
                "admin_id": 1,
                "customer_id": 0,
                "name": "Palacio de Congresos Kursall",
                "address": null,
                "city": "San Sebastian",
                "image": "1589018893.jpeg",
            }
        }
    ]
}

```

```

        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:08:13.000000Z"
    },
    "name": "Kubo txikia",
    "capacity": 250,
    "image": "1589019034.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 0,
    "s_parking": 1,
    "s_disabled": 1
},
{
    "id": 3,
    "admin_id": 0,
    "building_id": 5,
    "building": {
        "id": 5,
        "admin_id": 2,
        "customer_id": 0,
        "name": "Baluarte",
        "address": null,
        "city": "Iruñea",
        "image": "1589018942.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:02.000000Z"
    },
    "name": "Auditorio Baluarte",
    "capacity": 750,
    "image": "1589019147.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 1,
    "s_parking": 1,
    "s_disabled": 1
},
{
    "id": 4,
    "admin_id": 0,
    "building_id": 4,
    "building": {
        "id": 4,
        "admin_id": 1,
        "customer_id": 0,
        "name": "Palacio Europa",
        "address": null,
        "city": "Vitoria-Gasteiz",
        "image": "1589018929.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:08:49.000000Z"
    },
    "name": "Sala auditorio",
    "capacity": 450,
    "image": "1589021396.jpeg",
    "s_wifi": 1,
    "s_projector": 1,
    "s_printer": 1,
    "s_parking": 1,
    "s_disabled": 1
},
{
    "id": 5,
    "admin_id": 0,
    "building_id": 3,
    "building": {

```



```

        "id": 3,
        "admin_id": 1,
        "customer_id": 0,
        "name": "Palacio Euskalduna",
        "address": null,
        "city": "Bilbo",
        "image": "1589018995.jpeg",
        "created_at": "2020-05-09T12:05:51.000000Z",
        "updated_at": "2020-05-09T10:09:55.000000Z"
    },
    {
        "name": "Sala Euskalduna Tres",
        "capacity": 125,
        "image": "1589022461.jpeg",
        "s_wifi": 1,
        "s_projector": 1,
        "s_printer": 1,
        "s_parking": 1,
        "s_disabled": 1
    }
],
"status": 200,
"links": {
    "first": "http://localhost:8000/api/room?page=1",
    "last": "http://localhost:8000/api/room?page=1",
    "prev": null,
    "next": null
},
"meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "path": "http://localhost:8000/api/room",
    "per_page": 15,
    "to": 5,
    "total": 5
}
}

```

GET: User (<http://localhost:8000/api/user>)

```

{
    "data": [
        {
            "id": 1,
            "customer_id": 1,
            "name": "Administrator",
            "email": "admin@test.com",
            "image": "1588955764.png",
            "role": "CUSTOMER"
        },
        {
            "id": 2,
            "customer_id": 1,
            "name": "Brandt Cormier",
            "email": "dale83@gmail.com",
            "image": "1588955806.png",
            "role": "BUILDING_ADMIN"
        },
        {
            "id": 3,
            "customer_id": 1,
            "name": "Cole Connelly1",
            "email": "scot20@yahoo.com",

```

```
        "image": "1590425508.png",
        "role": "ROOM_ADMIN"
    },
    {
        "id": 4,
        "customer_id": 1,
        "name": "Duncan Greenholt",
        "email": "gwilderma@jast.biz",
        "image": "1590425541.png",
        "role": "ROOM_ADMIN"
    },
    {
        "id": 5,
        "customer_id": 1,
        "name": "Hermann Treutel DVM",
        "email": "lenna.murazik@yahoo.com",
        "image": "1590426850.png",
        "role": "ROOM_ADMIN"
    }
],
"status": 200,
"links": {
    "first": "http://localhost:8000/api/user?page=1",
    "last": "http://localhost:8000/api/user?page=3",
    "prev": null,
    "next": "http://localhost:8000/api/user?page=2"
},
"meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 3,
    "path": "http://localhost:8000/api/user",
    "per_page": 5,
    "to": 5,
    "total": 13
}
}
```

Anexo E: Currículum Vitae

Nacido en 1972 y trabajando desde los 21 años, hoy en día soy programador en una cooperativa comercializadora de energía renovable. Tengo más de 20 años de experiencia como desarrollador de aplicaciones web, compaginando las labores de programación backend y frontend, por lo que me considero Full Stack Developer.

He trabajado en internet como medio de desarrollo casi desde los inicios en nuestro entorno, más o menos desde finales de los 90, trabajando con HTML, PERL primero y PHP después. Actualmente desarrollo módulos del ERP de la empresa en la que trabajo usando este stack tecnológico:

- PostgreSQL
- .NET
- Git

Comencé mis estudios en el grado de multimedia cuando este no era oficial (año 2003), aunque por opciones personales como la paternidad tuve que dejarlo, inicié de nuevo mi andadura en la UOC en 2016.