Aplicación web progresiva dedicada a la gestión de incidencias para la Residencia Universitaria de Tafira

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Stefany Martín Socas

TUTORIZADO POR:
Alexis Quesada Arencibia
CO-TUTORIZADO POR:
Jonathan Alemán Alemán

Noviembre 2021

Agradecimientos

A mi tutor, Dr. Alexis Quesada Arencibia, y a mi co-tutor, Jonathan Alemán Alemán, por toda la paciencia, apoyo y ayuda recibida durante el desarrollo de este proyecto y su disponibilidad para responder a todas mis dudas.

A mi familia y amigos por apoyarme y darme ánimos en los días en los que más falta me hacía.

A mi pareja, cuyo apoyo durante toda la finalización de la carrera ha sido inestimable.

Resumen

El objetivo de este TFT consiste en renovar la aplicación actual que usa la Residencia Universitaria de Tafira para la gestión de incidencias y otros trámites. Para ello se convertirá en una aplicación web progresiva que, además de replicar las funcionalidades existentes, agregue funcionalidades añadidas. Las aplicaciones web progresivas están a medio camino entre las aplicaciones web y las aplicaciones nativas: son básicamente páginas web, pero mediante el uso de Service Workers y otras tecnologías se comportan más como aplicaciones normales que como páginas web. Además, con esta transformación se pretende simplificar y agilizar el mantenimiento y desarrollo de nuevas características sin que exista una alta dependencia respecto a las actualizaciones de los sistemas operativos iOS y Android.

Abstract

The aim of this TFT is to renew the current application used by the Residencia Universitaria de Tafira for the management of incidences and other procedures. To achieve this, it will be converted into a progressive web application that, in addition to replicating the existing functionalities, will add additional functionalities. Progressive web applications are halfway between web applications and native applications: they are basically web pages, but through the use of Service Workers and other technologies they behave more like normal applications than web applications. In addition, this transformation is intended to simplify and speed up the maintenance and development of new features without a high dependency on iOS and Android operating system updates.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

Actualmente, la Residencia Universitaria de la Universidad de Las Palmas de Gran Canaria hace uso de una aplicación móvil para gestionar las incidencias reportadas por los residentes, las ausencias previstas o la comunicación de avisos por parte de la Residencia. La aparición de una aplicación de este estilo posibilitó no solo el registro de este tipo de gestiones, sino además una mayor facilidad para realizar un seguimiento de las mismas. De esta forma, se mejora considerablemente la fluidez en la comunicación entre ambas partes implicadas. Se trata de una aplicación multiplataforma que surgió como un proyecto del Instituto Universitario de Ciencias y Tecnologías Cibernéticas en 2017, a la que finalmente denominaron como "Alloro ULPGC".

Dentro de las funcionalidades implementadas, la aplicación permite a sus usuarios registrar cualquier tipo de incidencia que tenga lugar dentro de las instalaciones de la residencia o de los apartamentos universitarios, además de realizar un seguimiento hasta su posterior resolución. La aplicación además permite notificar a los residentes de cualquier evento o anuncio importante que considere oportuno el personal de la residencia. Asimismo, otra de las funcionalidades consideradas en la aplicación actual es la de poder registrar las ausencias que los residentes tengan previstas durante el transcurso del curso.

Durante el desarrollo de este trabajo se realizará una versión revisada de esta aplicación, con el objetivo principal de realizar un cambio de tecnologías. Se partirá de la aplicación multiplataforma para obtener como resultado final una aplicación web progresiva. Esta nueva aplicación deberá contar con las funcionalidades básicas ya existentes, además de incluir nuevas funcionalidades que puedan proporcionar valor al usuario final o a la residencia. También se mejorará la usabilidad de la misma teniendo en cuenta principios básicos de diseño de interfaces de usuario y se obtendrá una aplicación intuitiva y fácil de usar para todos los usuarios sin importar el sistema operativo o dispositivo que se use para acceder a ella.

2 Estructura del documento

Este documento está separado en distintos apartados que se describirán en las siguientes líneas:

- **Capítulo 3 – Estado del arte y objetivos iniciales**: en este apartado se establece cuál ha sido el punto de partida para este TFT y cuáles serán los objetivos inicialmente previstos para el desarrollo del mismo.
- **Capítulo 4 – Competencias**: en este apartado se enumeran y se justifican todas las competencias cubiertas mediante el desarrollo de este TFT.
- **Capítulo 5 – Aportaciones**: en este capítulo se justifica qué puede aportar este TFT tanto al entorno socio-económico como al entorno personal.
- **Capítulo 6 – Tecnologías y herramientas utilizadas**: en este apartado se hace una pequeña introducción a todas las tecnologías y herramientas utilizadas para el desarrollo del proyecto.
- **Capítulo 7 – Normativa y legislación**: en este apartado se incluyen las licencias utilizadas para las herramientas o tecnologías de las que se ha hecho uso durante el progreso de este TFT.
- **Capítulo 8 – Metodología de trabajo y planificación del proyecto**: se describe la metodología aplicada durante el desarrollo del trabajo y sus distintas fases.
- **Capítulo 9 – Análisis**: en este capítulo se detalla el proceso de análisis realizado en el proyecto, indicando además los requisitos funcionales de la aplicación.
- **Capítulo 10 – Diseño**: en este apartado se describe el diseño de arquitectura que se ha decidido seguir para el desarrollo de la aplicación y los principios de diseño considerados para la interfaz de usuario.
- **Capítulo 11 – Desarrollo**: en este capítulo se describe el proceso que se ha seguido durante el desarrollo de este TFT, además de ciertas características del mismo que se han considerado especialmente relevantes.
- **Capítulo 12 – Resultados y conclusiones**: en este apartado se detallan los resultados del proyecto y los objetivos cumplidos. Además, se sugieren futuros trabajos a implementar partiendo de este TFT.

3 Estado del arte y objetivos iniciales

3.1. Análisis de las funcionalidades

Actualmente, la aplicación de “Alloro” permite a la Residencia Universitaria cubrir las necesidades básicas para los trámites más comunes e importantes del día a día en la misma: gestión de incidencias, gestión de ausencias y comunicaciones generales desde el personal de la residencia hacia los residentes. Esta aplicación se utiliza tanto en la Residencia Universitaria de Las Palmas y en la Residencia de Tafira, como también en los Apartamentos Universitarios de Tafira y en los Bungalows.

Ilustración 3.1: Pantallas principales de la aplicación

En la Ilustración 3.1 se puede ver la pantalla inicial de carga que verá el usuario al entrar en la aplicación, así como el menú de inicio desde donde se podrá acceder

a todas estas funcionalidades. A continuación, se hará una descripción detallada de dichas funcionalidades, así como del estado actual de la aplicación. Se dividirán las funcionalidades en dos grupos distintos: las funcionalidades en las que no es necesario tener una sesión iniciada y las que están reservadas a usuarios que han iniciado sesión previamente.

3.1.1. Funcionalidades para todos los usuarios

3.1.1.1. Avisos

Uno de los apartados disponibles desde el menú inicial es el de "Avisos". En esta sección, el usuario podrá recibir cualquier tipo de notificación o comunicación que envíe el personal de la residencia para todos los usuarios de la *app*, como se puede observar en la Ilustración 3.2.

Esta funcionalidad está pensada para todas las comunicaciones que sean de carácter general, como pueden ser avisos de corte de agua, de internet o cualquier otro tipo de notificaciones que afecte a todos los residentes. De esta forma, en este apartado no se podrá comunicar individualmente incidencias ni mensajes, ya que es un apartado de carácter general.

Ilustración 3.2: Pantalla de avisos

3.1.1.2. Servicios

Otro de los apartados públicos al que se puede acceder desde el menú inicial de la aplicación es el de "Servicios". En esta sección se mostrará una serie de enlaces y datos relevantes a la residencia, como el número de teléfono y correo de la residencia, además de un enlace a su página web. También se muestran los datos de la tienda de la ULPGC y de la web del *International Mobility Point*. En la Ilustración 3.3 se puede observar cada una de las pantallas de los servicios mencionados anteriormente.

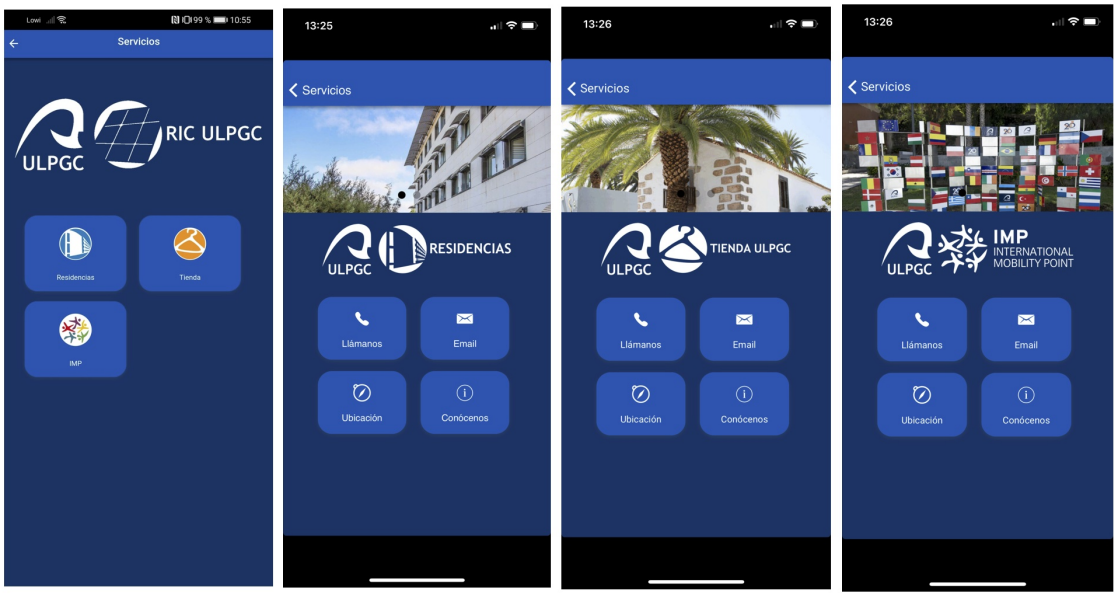

Ilustración 3.3: Pantallas de servicios

3.1.1.3. Mi perfil

Para poder hacer uso de algunas de las funcionalidades esenciales de la aplicación de Alloro, será necesario haber iniciado sesión previamente. Para ello existe un apartado llamado "Mi perfil" en la parte superior derecha del menú inicial de la aplicación.

Si el usuario accede a "Mi perfil" sin haber iniciado sesión previamente, se mostrará un formulario para realizar el inicio de sesión, como se puede observar en la primera imagen de la Ilustración 3.4. Este formulario cuenta además con una opción para recuperar la contraseña. Si por el contrario el usuario ha iniciado sesión previamente en la aplicación, aparecerán los datos del perfil de inicio de sesión: usuario, número de teléfono y correo, como se puede observar en la segunda imagen de la Ilustración 3.4.

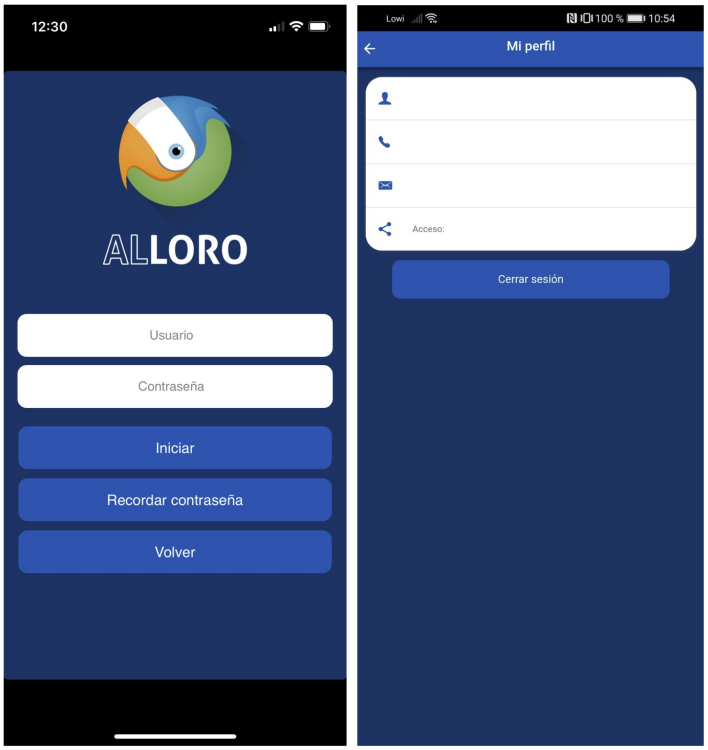

Ilustración 3.4: Pantallas de inicio de sesión y datos de perfil

3.1.2. Funcionalidades reservadas a usuarios con sesión iniciada

3.1.2.1. Mensajes

La aplicación cuenta con un apartado de "Mensajes". En este apartado, se podrán ver en forma de historial de mensajes todas las incidencias que se hayan comunicado en el pasado, como se puede observar en la Ilustración 3.5. Además, al seleccionar un mensaje en específico se podrá ver con detalle todo el proceso de resolución de la incidencia, ya que el personal de la residencia podrá ir actualizando el estado de la misma a medida que se vaya necesitando: cuando la incidencia se traslada al área indicada para su resolución, cuando el personal indicado está encargado de la resolución de la incidencia, y finalmente, cuando la incidencia ya esté dada por finalizada. Además, se puede buscar una incidencia en el historial de mensajes buscando por el título.

Si el usuario no tiene la sesión iniciada en el momento de entrar al apartado de "Mensajes", aparecerá únicamente un formulario para el inicio de sesión, visto anteriormente en la Ilustración 3.4.

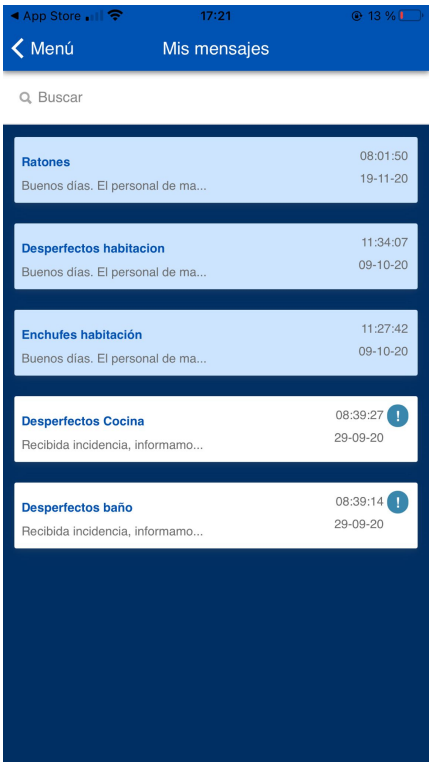

Ilustración 3.5: Pantalla de mensajes

3.1.2.2. Gestiones (Ausencias)

En el menú inicial existe un apartado denominado "Gestiones". En este apartado se engloba tanto la gestión de incidencias como la gestión de ausencias. En primer lugar, se describirá las características de la gestión de ausencias.

Las ausencias están pensadas para que los residentes puedan comunicar al personal de la residencia un lapso de tiempo en específico en el que se encontrarán fuera de la misma.

A la hora de añadir una nueva ausencia, el usuario tendrá una serie de opciones disponibles. En primer lugar, se deberá seleccionar en qué residencia reside dentro de las opciones disponibles: Residencia Universitaria de Tafira, Residencia de Las Palmas, Apartamentos Universitarios o Bungalows. En el siguiente paso, el usuario deberá seleccionar las fechas en las que estará fuera de la residencia. Para ello, seleccionará tanto la fecha de salida como la fecha de vuelta. Por último, si el usuario lo considera conveniente, podrá añadir cualquier observación que considere necesaria. Todos estos pasos se pueden observar en la Ilustración 3.6.

Al igual que con el apartado mencionado anteriormente, si el usuario no tiene la sesión iniciada en el momento de entrar al apartado de "Gestiones", aparecerá únicamente

un formulario para el inicio de sesión.

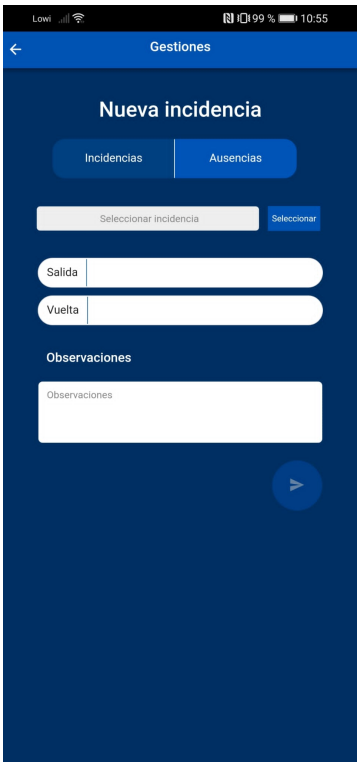

Ilustración 3.6: Pantalla de ausencias

3.1.2.3. Gestiones (Incidencias)

Dentro del apartado de "Gestiones" del menú inicial, existe un apartado para crear una nueva incidencia. La gestión de incidencias está pensada para que un residente pueda comunicar de una manera sencilla y detallada cualquier desperfecto que encuentre en las instalaciones de la residencia, o cualquier otro tipo de queja que requiera solución.

Para añadir una nueva incidencia mediante la aplicación, será necesario que el usuario seleccione la ubicación desde donde desea realizar la incidencia. Como se ha mencionado previamente, el usuario podrá seleccionar la Residencia de Tafira, la Residencia de Las Palmas, los Apartamentos de Tafira o los Bungalows.

Además, dentro de la residencia que seleccione el usuario, la aplicación permite seleccionar el área de servicio de la residencia a la que irá dirigida, que serán: lavandería, comedor, recepción, mantenimiento, limpieza, comunicaciones, equipamiento/infraestructuras, administración y ruidos.

Por último, antes de enviar la nueva incidencia, es necesario que el usuario indique un título explicativo y una pequeña descripción de la incidencia de 150 caracteres o

menos. Como añadido a la descripción de las incidencias, es posible añadir imágenes o audio como archivos adjuntos.

Todo este flujo de trabajo descrito previamente para reportar una nueva incidencia se puede observar en la Ilustración 3.7

Una vez enviada la incidencia, se podrá seguir su resolución en el apartado de "Mensajes", descrito anteriormente.

Si el usuario intenta entrar en este apartado sin tener una sesión iniciada, aparecerá la misma pantalla de formulario de *login* que en los apartados previamente descritos, ya que esta funcionalidad está únicamente disponible para usuarios con sesión iniciada.

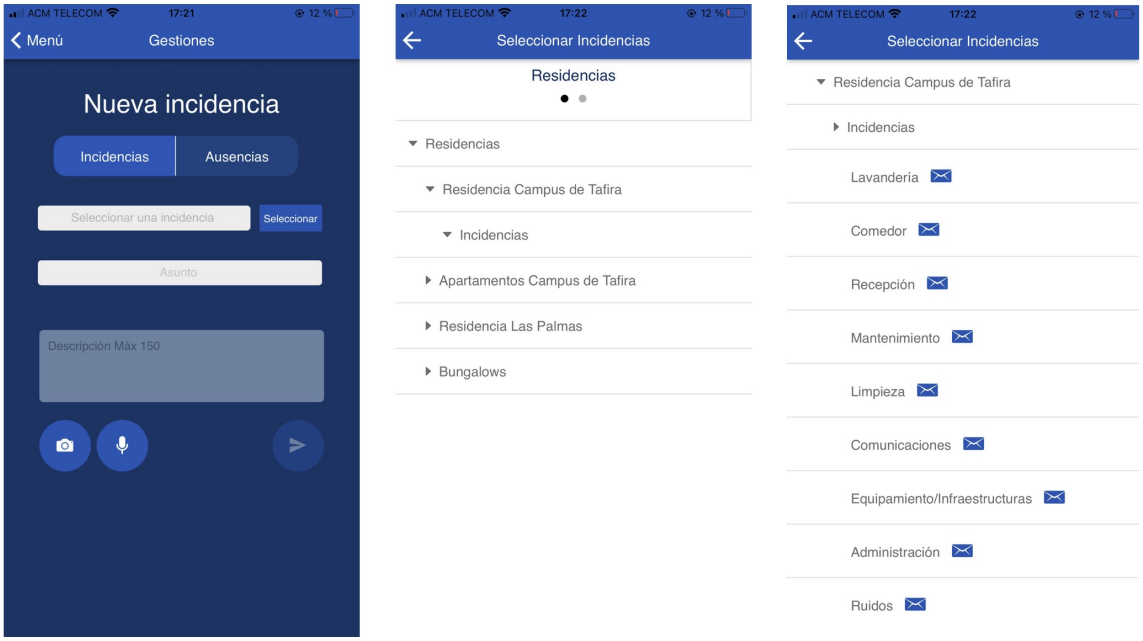

Ilustración 3.7: Pantallas para la creación de una incidencia

3.1.3. Carencias de la aplicación actual

Previamente al desarrollo de la aplicación de este TFT, se ha realizado un análisis preliminar para detectar las posibles carencias de la aplicación actual.

En cuanto al diseño de la aplicación actual, cuenta con un diseño muy simple y anticuado. Además, no se trata de un diseño completamente responsivo, como podemos observar en varias de las ilustraciones incluidas en el apartado de "Análisis de las funcionalidades". En determinados modelos se verá correctamente la aplicación, mientras que, en modelos más modernos de *smartphones*, la aplicación no se adapta al tamaño de la

pantalla. Esto es necesario que sea corregido, ya que puede ocasionar que, dependiendo del dispositivo con el que se use la aplicación, pueda llegar a verse poco profesional. Debido al diseño obsoleto con el que cuenta, puede llegar a resultar en ocasiones poco intuitivo dependiendo del trámite que se desee realizar.

Por último, actualmente los trámites que se pueden realizar usando esta aplicación aún son muy limitados. Como objetivo de este TFT, se añadirán nuevas funcionalidades a la aplicación que puedan aportar valor tanto al personal de la residencia, al facilitarle ciertos trámites, como al usuario.

3.2. Objetivos iniciales

Teniendo en cuenta el contexto en el que se desarrolló esta primera versión de la aplicación, el objetivo de este Trabajo de Fin de Título consiste principalmente en transformar la aplicación existente en una aplicación web progresiva. Para conseguirlo, se desarrollará una versión más actualizada de la aplicación vigente, haciendo uso de nuevas tecnologías que nos permitirán incluir todas las funcionalidades de una manera más eficiente y moderna. De esta forma, los objetivos de este TFT se pueden describir como:

- Convertir la aplicación en una aplicación web progresiva mediante el uso de *service workers*.
- Replicar y mejorar las funcionalidades existentes actualmente para conseguir como resultado una versión actualizada y optimizada de Alloro.
- Ofrecer una interfaz más profesional e intuitiva para los usuarios.
- Incluir nuevas funcionalidades que pueden resultar útiles para el centro y sus residentes, que sustituirán a los procedimientos realizados de manera tradicional.

Las aplicaciones web progresivas [1] (o PWA por sus siglas en ingles, *Progressive Web Apps*) están a medio camino entre las aplicaciones web y las aplicaciones nativas: son básicamente páginas web, pero mediante el uso de *Service Workers* y otras tecnologías se comportan más como aplicaciones nativas que como aplicaciones web. Por lo tanto, se podría decir que las aplicaciones web progresivas son una evolución natural de las aplicaciones web, donde se difumina la barrera entre las páginas web y las aplicaciones, pudiendo realizar tareas que generalmente solo las aplicaciones nativas podían llevar a cabo.

Algunos ejemplos son el funcionamiento sin conexión a Internet o la posibilidad de probar una versión más ligera antes de proceder con un desarrollo nativo.

Además, con esta transformación se pretende simplificar y agilizar el mantenimiento y desarrollo de nuevas características sin que exista una alta dependencia respecto a las

actualizaciones de los sistemas operativos iOS y Android.

4 Competencias

4.1. Comunes a Ingeniería Informática

4.1.1. CII01

"Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente."

Durante el desarrollo de la aplicación de Alloro se ha desarrollado tanto la parte de *frontend* como la del *backend*, controlando los datos a los que tiene acceso el usuario.

4.1.2. CII08

"Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados."

Con el objetivo de desarrollar una versión mejorada de la aplicación existente, se ha realizado un análisis exhaustivo de los diferentes lenguajes y tecnologías disponibles y se han elegido las opciones teniendo siempre en cuenta cuáles eran las más indicadas para esta aplicación.

4.1.3. CII13

"Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web."

Debido a que en este TFT se ha desarrollado una aplicación web progresiva, el desarrollo de código realizado ha sido similar al que se haría para el de una página web. Además, en este proyecto se ha realizado también el desarrollo de la API a la que se accede para la obtención de los datos.

4.1.4. CII016

"Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software."

Para el desarrollo de este proyecto se ha utilizado un ciclo de vida incremental o iterativo, en el que cada iteración finaliza con la obtención de un producto con más valor añadido que en la iteración anterior.

4.1.5. CII017

"Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas."

Durante el desarrollo de la aplicación web progresiva de Alloro se ha tenido en cuenta los principios de diseño recomendados para conseguir una interfaz de usuario intuitiva y fácil de usar por el usuario final.

4.2. Trabajo de Fin de Grado

4.2.1. TFG01

"Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas."

En este proyecto se ha tenido que pasar por fases de desarrollo similares a las que se harían en un proyecto real. Además, para concluir un desarrollo satisfactorio, ha sido necesario aplicar los conocimientos adquiridos durante el grado de Ingeniería Informática.

4.3. Ingeniería de Software

4.3.1. IS01

"Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente,

sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software."

Para el desarrollo de esta aplicación, se ha seguido una metodología de trabajo en la que ha sido necesario tener siempre presentes los requisitos del proyecto, consiguiendo de esta forma una aplicación que se considere de calidad y que aporte un valor real a los usuarios objetivos de la misma.

4.3.2. IS03

"Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles."

Durante el desarrollo de este TFT se ha tenido que hacer uso de diferentes tecnologías y herramientas. Además, se ha realizado el desarrollo de la parte de interfaz de usuario o *frontend*, a la vez que se ha realizado una versión simulada del apartado del *backend*, por lo que ha sido necesario realizar una integración de ambas partes.

4.3.3. IS04

"Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales. "

Como en la mayoría de proyectos de desarrollo, durante el proceso de este TFT se han encontrado ciertas dificultades. Para poder sobrellevarlas, ha sido necesario una ardua tarea de investigación para llegar a las mejores soluciones disponibles que mejor se adecúen a este proyecto y sus necesidades concretas.

5 Aportaciones

5.1. Entorno socio-económico

Este proyecto pondrá a disposición tanto de la Residencia Universitaria como de los residentes un método de comunicación directo mucho más eficiente y rápido de lo que podría ser el método tradicional usando formularios en papel. Esto trae consigo las siguientes consecuencias positivas:

- La aplicación estará a disposición de todos los residentes: debido a la naturaleza de las aplicaciones web progresivas, todos los residentes podrán tener acceso a ella, sin importar el sistema operativo o dispositivo del que se disponga para poder acceder a sus funcionalidades.
- Se aumenta la eficacia con la que se realizan y se gestionan todos los trámites realizados a través de la *app*: se eliminan casos como podrían ser pérdidas o confusiones entre los documentos.
- Se disminuye la necesidad de impresiones en papel, que a su vez implica tanto consecuencias positivas para el medio ambiente como para la propia residencia, ya que se disminuyen a su vez los gastos económicos que esto genera.
- Se reduce la necesidad de la comunicación interpersonal con la residencia. Esto supone una disminución de colas y de cantidad de gente a la espera de procesar un trámite indicado en la recepción, lo que beneficia tanto al residente como a la residencia.
- Mejora la comunicación entre la residencia y el residente, ya que este último podrá recibir comentarios sobre los trámites directamente del personal de gerencia.
- Reduce la necesidad de ir en persona a la recepción de la residencia. En algunos casos, como puede ser para personas con movilidad reducida, implicaría una mejora notoria en la calidad de vida durante su residencia. Igualmente afectaría a todos aquellos residentes donde tendrían que movilizarse para ir a la recepción, como puede ser en los apartamentos de Tafira.

5.2. Personal

A nivel personal, el desarrollo de este proyecto ha sido una experiencia muy enriquecedora. Este proyecto ha brindado la oportunidad de poner a prueba todos los conocimientos aprendidos a lo largo del grado de Ingeniería Informática. El hecho de poder aplicar todos estos conocimientos en un trabajo que engloba todas estas competencias ha sido una experiencia enriquecedora que ayuda a hacerse una idea sobre cómo es el funcionamiento real de un proceso de desarrollo para una aplicación de este estilo y todas las fases que abarca. Además, ha brindado la oportunidad de aprender nuevas tecnologías tanto para el desarrollo del *backend* como del *frontend*.

Por una parte, durante el desarrollo de este proyecto, se ha tenido que investigar sobre el funcionamiento y ventajas de las *Progressive Web Apps*, y sobre cómo implementar este tipo de tecnología. Esto ha resultado muy interesante ya que es un concepto relativamente novedoso en el ámbito de las aplicaciones móviles.

Por otra parte, para el desarrollo del *frontend* se ha utilizado el *framework* de *Vue*, por lo que ha sido necesario una fase de adquisición de conocimientos desde cero. En cuanto al *backend*, se ha hecho uso de *Laravel* con PHP, que por un lado ha proporcionado la oportunidad de aprender una nueva tecnología como es *Laravel* y, por otro, profundizar en los conocimientos adquiridos en la carrera sobre el lenguaje de PHP.

Como conclusión, este proyecto ha sido una gran aportación para el desarrollo profesional, tanto por todos los conocimientos afianzados como por los aprendidos.

6 Tecnologías y herramientas utilizadas

6.1. Tecnologías

Las tecnologías utilizadas en este TFT son las siguientes:

- **PHP 7**: se trata de un lenguaje de programación del lado del servidor ampliamente utilizado para desarrollar páginas web dinámicas e interactivas. Fue uno de los primeros lenguajes que se podía incrustar en los ficheros HTML, lo que implica que se hace más sencillo añadir funcionalidad a las páginas web sin necesidad de llamadas a ficheros externos. Actualmente tiene soporte en la mayoría de servidores y plataformas sin ningún coste, al tratarse de un lenguaje *open source* [2].
- **Vue**: se trata de un *framework* progresivo de Javascript que se utiliza para el *frontend*, es decir, para el desarrollo de interfaces de usuario. Al estar diseñado para utilizarse incrementalmente, se trata de un *framework* sencillo y fácil de usar y de integrar con otras tecnologías o librerías externas. Además, promueve el desarrollo de *Single Page Applications* junto con el uso de tecnologías modernas [3].
- **Javascript**: es un lenguaje de programación que puede utilizarse tanto del lado del cliente como del lado del servidor y permite el desarrollo de páginas web interactivas. Usando Javascript se proporciona al usuario una serie de elementos interactivos para atraer al usuario, como pueden ser barras de búsqueda o refrescar el contenido de una página. Incorporando Javascript se obtiene una página más atractiva para el usuario [4].
- **Laravel**: es un *framework* de PHP de código abierto diseñado para facilitar y agilizar el desarrollo *backend* de aplicaciones web proporcionando una estructura y un punto de partida para la creación de una aplicación gracias a sus características integradas, como la inyección de dependencias, una capa de abstracción de la base de datos, pruebas unitarias y de integración, entre otros [5].
- **CSS**: se trata de un lenguaje basado en hojas de estilo para describir la presentación de un documento estructurado en HTML o XML. Mediante el uso de hojas de estilo CSS se describe cómo se representarán gráficamente los diferentes elementos de una aplicación o página web. Se trata de uno de los lenguajes básicos

para el desarrollo de páginas y aplicaciones web y está estandarizado en todos los navegadores [6].

- **HTML**: Es un lenguaje de marcado de texto que define la estructura básica y el contenido de una página o aplicación web mediante etiquetas y texto HTML. Se considera el lenguaje más importante a la hora de desarrollar páginas web, ya que su invención fue una parte crucial para la expansión de la *World Wide Web* (WWW). Actualmente, es el estándar que todos los navegadores han adoptado para la visualización de las páginas webs [7].
- **Git**: es actualmente la herramienta de control de versiones más utilizada alrededor del mundo. Se trata de una herramienta de código abierto para mantener un control sobre los cambios que se realizan en el código fuente de un proyecto. Además, esta herramienta facilita mantener una copia de seguridad en la nube estructurada de todos estos cambios y del código fuente actualizado [8].
- **Node.js y NPM**: NPM es una herramienta para la gestión de paquetes o módulos de Node.js. Los paquetes de Node.js contienen todos los archivos necesarios para el uso de un módulo. A su vez, un módulo es una librería de Javascript que se puede añadir en el proyecto para añadir funcionalidades. El programa NPM se instala automáticamente al instalar Node.js. [9]

6.2. Herramientas

Se han utilizado las siguientes herramientas durante el transcurso del proyecto:

- **PhPStorm**: se trata de un entorno de desarrollo para utilizar con herramientas que utilicen como PHP como lenguaje de programación, como pueden ser Laravel, Symfony, etc. Este *IDE* (del inglés *Integrated Development Environment*) pertenece a la plataforma de JetBrains IntelliJ IDEA [10].
- **WebStorm**: se trata de un entorno de desarrollo para utilizar principalmente con Javascript y tecnologías o *frameworks* relacionados, como puede ser Vue, Angular, React, entre otros. Este *IDE* pertenece a la plataforma de JetBrains IntelliJ IDEA.
- **Visual Paradigm**: es una herramienta para realizar esquemas o diagramas en formato UML [11].
- **Google Chrome**: se trata de un navegador web que cuenta con herramientas que ayudan al desarrollo de aplicaciones web.
- **Safari**: se trata de otro navegador web, utilizado principalmente por dispositivos que usen iOS o MacOS.
- **GitHub**: es una plataforma para alojar las copias de seguridad del código fuente de un proyecto mediante el uso del sistema de control de versiones de Git.

- **PhpMyAdmin**: se trata de una herramienta de PHP con la que se puede administrar bases de datos MySQL utilizando un navegador web. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos [12].

6.3. Servidores

- **MAMP**: se trata de un entorno de servidor local que se instala en un ordenador y se utiliza principalmente para trabajar y realizar pruebas en un entorno de desarrollo local. Utiliza MySQL para las bases de datos y se puede utilizar con lenguajes como PHP, Python, Perl o Ruby [13].

7 Normativa y legislación

7.1. Licencia de Software

Una licencia [14] se considera un contrato entre el autor de los derechos de explotación y distribución de una determinada herramienta o tecnología informática y el licenciatario o usuario consumidor, profesional o empresa. Este contrato determina los términos y condiciones de uso establecidas y los permisos otorgados al usuario sobre el producto en concreto. Además, las licencias suelen tener un plazo de duración determinado.

Las licencias de software [14] establecen los derechos que se ceden al usuario final sobre una copia del programa informático, el plazo de duración de la cesión de los derechos y el ámbito geográfico en los que se valida el contrato. Además puede incluir otro tipo de condiciones de uso que vendrán determinadas por el autor del software.

A continuación se nombrarán las diferentes licencias utilizadas en el desarrollo de este TFT:

7.1.1. Licencia educacional de JetBrains

Para el uso de los entornos de desarrollo de WebStorm, para el desarrollo del frontend con Vue, y de PhpStorm, para el desarrollo del backend con Laravel, se ha utilizado una licencia proporcionada para estudiantes perteneciente a JetBrains.

Esta licencia [15] no puede ser vendida, y en sus condiciones de uso se especifica que no se pueden utilizar sus productos con propósitos económicos. JetBrains mantiene todos los derechos sobre los productos, incluyendo la propiedad intelectual.

7.1.2. Licencia pública general

La licencia pública genera de GNU [16] es una de las licencias que ceden derechos de autor más utilizada en el mundo del software libre o del código abierto, ya que concede a los usuarios finales la libertad de poder utilizar el software a su gusto, e incluso modificarlo.

El objetivo de las licencias públicas generales es declarar que el software se considera un software libre, y además prevenir de intentos de apropiación del software por parte de terceros que deseen la restricción de uso del mismo, mediante una práctica conocida como *copyleft* [16].

La mayoría de los programas de licencia GNU son *copyleft*, aunque no al completo. Sin embargo, todo el software GNU debe ser software libre [16].

Esta licencia es usada por: Git, MAMP, PhpMyAdmin.

7.1.3. Software gratuito

Bajo esta licencia se ofrece el software con su funcionalidad al completo de manera gratuita, estando disponible para el uso del usuario pero manteniendo las restricciones en su *copyright*, por lo que no se puede modificar o vender como ocurriría con el software libre [17].

Esta licencia es usada por: Google Chrome, Safari, Visual Paradigm.

7.1.4. Licencia PHP

La licencia PHP [18] es la licencia bajo la que se publica el lenguaje de programación de PHP. De acuerdo a la *Free Software Foundation* es una licencia de software libre que no es *copyleft* y de código abierto, según la *Open Source Initiative*. Debido a la restricción en el uso del término "PHP", no es compatible con la licencia pública general de GNU.

7.1.5. Licencia MIT

La licencia MIT [19] es una licencia de software libre permisiva, por lo que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente compatibilidad de licencia. Esta licencia es compatible con muchas licencias *copyleft*, como la licencia pública general de GNU (el software con licencia MIT puede integrarse en software con licencia pública general, pero no al contrario). Además, esta licencia no cuenta con *copyright*, lo que permite su modificación.

Esta licencia es usada por: Vue, Laravel, Node.js.

8 Metodología de trabajo y planificación del proyecto

Las metodologías de trabajo [20] forman actualmente una base muy importante a la hora de organizar y planificar eficazmente el desarrollo del software. El objetivo de trabajar haciendo uso de estas metodologías de trabajo consiste principalmente en conseguir una organización eficaz y adaptada a las necesidades del proyecto en desarrollo. Así, utilizando una buena metodología, se conseguirá reducir los errores o retrasos y se agilizará y mejorará en general el resultado final del desarrollo.

A la hora de escoger una metodología a seguir para un proyecto, es muy importante considerar factores como costes, planificación temporal, la dificultad de las tareas, número de personas en el equipo de desarrollo, entre otros.

8.1. Metodología de trabajo

Teniendo en cuenta todo lo mencionado anteriormente, para este proyecto se ha utilizado una metodología incremental o iterativa.

Un desarrollo incremental o iterativo es una implementación de un ciclo de vida del software que se enfoca en una implementación inicial y simplificada del producto que irá ganando progresivamente más complejidad y funcionalidades en diferentes etapas incrementales hasta que se concluya el desarrollo [21]. De esta forma, al final de cada proceso iterativo, se contará con un producto con más funcionalidades que con el que partimos.

Una de las ventajas más relevantes del modelo incremental es la posibilidad de comprobar el funcionamiento de ciertas partes del proyecto antes de su finalización, lo que permite una comunicación más fluida con el cliente a la hora de recibir e implementar el *feedback*, y así se conseguirá un proceso de desarrollo con un mínimo porcentaje de errores.

Para que esto se pueda llevar a cabo, es muy importante la correcta evaluación en las diferentes fases iterativas [22]. Los responsables del proyecto, en este caso los tutores de este TFT, deberán valorar si los resultados son los esperados o si es necesario implementar alguna solución alternativa en ese punto del desarrollo.

8.2. Planificación del proyecto

Previo al comienzo del desarrollo de la aplicación de Alloro, se realizó una planificación inicial para estimar la duración de todas las tareas necesarias para completar este TFT, como podemos ver en la Tabla 8.1:

Fases	Duración estimada (horas)	Tareas
Estudio previo / Análisis	40	Tarea 1.1: Estudio del problema y limitaciones actuales.
		Tarea 1.2: Aprendizaje de herramientas y conocimientos necesarios.
		Tarea 1.3: Especificación de requisitos.
Diseño / Desarrollo	170	Tarea 2.1: Definición de tareas a desarrollar y planificación.
		Tarea 2.2: Diseño y mockups de las principales funcionalidades.
		Tarea 2.3: Desarrollo del código en una aplicación web progresiva.
		Tarea 2.4: Ampliación de funcionalidades identificadas en la fase de análisis.
Evaluación / Validación	35	Tarea 3.1: Desarrollo de pruebas de funcionamiento.
		Tarea 3.2: Validación del software desarrollado con la Residencia Universitaria de Tafira.
Documentación	35	Tarea 4.1: Realización de la memoria.
		Tarea 4.2: Preparación de la presentación del trabajo.

Tabla 8.1: Tabla de planificación inicial

Esta planificación inicial sufrió algunos contratiempos a medida que se avanzaba con el proyecto, ya que se hicieron presente ciertas dificultades con algunas de las tecnologías que se iban a utilizar en primer momento, por lo que finalmente la etapa de desarrollo llevó más tiempo del que se había estimado inicialmente.

Durante la primera fase, en la tarea de aprendizaje de herramientas y conocimientos necesarios, se dedicó una cierta cantidad de horas al aprendizaje del *framework* de Angular, ya que en un principio era la tecnología que se usaría para el desarrollo del *frontend* de este proyecto. Sin embargo, una vez pasado a la fase de desarrollo, se hicieron presente ciertas dificultades a la hora de la utilización de esta tecnología, por

lo que se tuvo que realizar un cambio a Vue. Esto ocasionó una inversión de más horas de las previstas en la tarea de aprendizaje de herramientas y conocimientos ya que fue necesario volver a esta fase con el objetivo de familiarizarse con el funcionamiento de Vue. Además, se hizo necesario empezar de nuevo con la fase de desarrollo ya que hubo que cambiar todo lo que ya estaba hecho en Angular a Vue.

Debido a todo esto, la duración estimada en horas de la planificación inicial de cada tarea se vio afectada tanto en la fase de estudio previo/análisis como en la fase de diseño/desarrollo, en las que se tuvieron que invertir más horas de las estimadas inicialmente.

8.3. Pre-análisis

Antes de comenzar con la fase de desarrollo, se realizó un análisis previo para confirmar el estado actual de la aplicación de Alloro y cuáles eran las limitaciones que se solucionarían en este TFT.

En primer lugar, se identificaron las tareas básicas y necesarias con las que necesitaría contar la nueva versión revisada de Alloro, que consistirán en una réplica de las funcionalidades ya existentes en la aplicación actual con ciertos aspectos mejorados para proporcionar una mejor experiencia al usuario final.

El primer requisito acordado en esta primera fase fue la migración de tecnologías, que en este caso se tratará de una *Progressive Web App*, que proporciona una serie de ventajas de las que disfrutan las aplicaciones nativas con la flexibilidad añadida que ofrece una página web.

La siguiente fase consistió en el aprendizaje de las nuevas tecnologías a utilizar para el desarrollo de este TFT. Para esta fase se invirtieron una serie de horas con el objetivo de familiarizarse con las nuevas tecnologías.

Una vez adquiridos los conocimientos necesarios, se comenzó con el desarrollo de este proyecto. Como se mencionó anteriormente, siguiendo la metodología incremental en la que se basa este proyecto, se realizaron la siguiente serie de iteraciones:

1. **Iteración 1**: Instalación de las herramientas necesarias para el futuro desarrollo. Primera versión del aspecto de la aplicación, realizando un prototipo haciendo uso de una plantilla proporcionada por el tutor.
2. **Iteración 2**: Réplica de las funcionalidades existentes en la aplicación actual en el proyecto como una aplicación web progresiva.
3. **Iteración 3**: Modificación de aspecto y funcionamiento de las funcionalidades existentes según *feedback* recibido y añadido de nuevas funcionalidades.

4. **Iteración 4**: Despliegue de la aplicación para poder llevar a cabo pruebas de funcionamiento.

9 Análisis

En esta sección se hablará de los resultados de la etapa de análisis de la aplicación de Alloro y sus principales funcionalidades y requisitos.

9.1. Actores

A continuación se explicará la función de cada uno de los actores de la aplicación identificados en la fase de análisis, indicando qué papel tienen dentro de la aplicación Alloro.

9.1.1. Gestor de la residencia

Las principales funciones del gestor de la residencia son las siguientes:

- **Gestión de avisos**: el gestor de la residencia podrá mandar avisos a los residentes o usuarios de la aplicación para comunicar a todos los residentes una información que se considere importante. Estos avisos serán públicos e iguales para todos los usuarios.
- **Gestión de incidencias**: el gestor se encargará de gestionar adecuadamente todas aquellas incidencias reportadas por cada uno de los usuarios de la aplicación. Esto significa que deberá mantener un seguimiento sobre el proceso de la incidencia y comunicarle al residente en todo momento cómo se está solucionando la incidencia reportada. A su vez, será el responsable de cerrar una incidencia una vez se considera resuelta.
- **Registro de usuarios**: la gestión de la residencia será el área encargada de registrar a los usuarios para que puedan iniciar sesión en la aplicación y hacer uso de las funcionalidades reservadas a usuarios con sesión iniciada.

9.1.2. Residente

El actor más importante de la aplicación de Alloro será el del residente, ya que se trata del usuario objetivo de este proyecto. Las principales funcionalidades del residente

en la aplicación serán las siguientes:

- **Inicio de sesión**: para algunas de las funcionalidades básicas de la aplicación, será necesario que el usuario inicie sesión para tener acceso a ellas.
- **Creación de nuevas incidencias**: los residentes podrán en cualquier momento reportar una nueva incidencia. Para hacerlo, tendrán que rellenar un formulario en el que se especificará la residencia en la que se alojan, el área al que pertenece la incidencia a reportar, y por último, será necesario añadir un título y descripción para explicar detalladamente la incidencia. Además, se podrá añadir imágenes o audio como ficheros adjuntos.
- **Seguimiento de incidencias**: una vez una incidencia ha sido creada, el residente podrá realizar un seguimiento de la misma. La gestión de la residencia mantendrá informado al residente sobre el proceso, y a su vez, este podrá responder a la residencia si lo necesita. De la misma forma, se podrá añadir imágenes o audio como ficheros adjuntos.
- **Reporte de ausencias**: una de las principales funcionalidades de la aplicación es la posibilidad de reportar una ausencia programada por parte del residente. La residencia necesita saber todos aquellos lapsos de tiempo en los que un residente no se alojará en su habitación. El usuario podrá enviar el rango de fechas en las que se encontrará fuera para que la residencia lo registre adecuadamente.
- **Visualización de avisos**: otra de las principales funcionalidades de la aplicación es que los usuarios reciban diariamente avisos importantes por parte de la residencia.

Hasta este punto, las funcionalidades identificadas son las ya existentes en la aplicación actual. A continuación se mencionarán las nuevas funcionalidades añadidas:

- **Filtrar y ordenar avisos e incidencias**: para la comodidad del usuario, se podrá buscar una incidencia o aviso en concreto haciendo uso de una barra de búsqueda, así como ordenar por título o por fecha.
- **Reserva de estancias comunes**: el residente podrá hacer uso de la aplicación para reservar en una determinada fecha y hora una estancia de la residencia. Las estancias disponibles serán: comedor, cancha de baloncesto, canchas de pádel y sala común.
- **Ver reservas**: los usuarios podrán ver las reservas futuras que hayan realizado para cada una de las estancias disponibles. Para cada reserva, se podrá ver la fecha, la hora y la duración de la misma.
- **Filtrar reservas por fecha**: el usuario podrá filtrar las reservas por una fecha específica para reducir la cantidad de reservas que visualiza en la página.
- **Cancelar reservas**: a la hora de visualizar las reservas futuras que tiene un

usuario, este podrá cancelar una reserva si lo considera necesario.

- **Rellenar formulario de estado inicial de la habitación**: al comienzo de la estancia del nuevo año escolar, todos los residentes tendrán la opción de rellenar un formulario de estado inicial de la habitación, mediante el cual se podrá informar de los imperfectos identificados en la habitación al inicio de la estancia.
- **Ver estado inicial de habitación**: una vez enviado el formulario de estado inicial de la habitación, el usuario podrá consultarlo en cualquier momento. No será posible editar esta información, únicamente consultarla.
- **Enviar mensajes a la residencia**: dentro del apartado de mensajes, el usuario contará con una conversación fija con la residencia, en la que podrá mandar mensajes a la gestión de la residencia siempre que considere necesario. Se podrá adjuntar imágenes a los mensajes.

9.2. Requisitos

Para la representación de los requisitos funcionales de la aplicación de Alloro, se ha hecho uso del lenguaje de modelado UML empleando la herramienta *Visual Paradigm*.

El lenguaje unificado de modelado (o UML, por sus siglas en inglés, *Unified Modeling Language*) [23] se trata del lenguaje de modelado de aplicaciones software más utilizado actualmente. Se utiliza para representar y especificar aspectos conceptuales como funcionalidades o procesos de un proyecto de software. Es decir, se trata del lenguaje más utilizado para describir el modelo de un proyecto.

La elección de utilizar *Visual Paradigm* como herramienta para la realización de estos esquemas se ha basado en dos razones principales: por un lado, se trata de un software *online* gratuito, el cual ofrece todas las funcionalidades para uso personal. Por otro lado, es una herramienta muy completa y fácil de utilizar, lo que significa menos tiempo de aprendizaje dedicado a esta herramienta.

Como los objetivos de este Trabajo de Fin de Título se basan exclusivamente en las funcionalidades de la aplicación del lado de los residentes, se analizarán únicamente los casos de uso pertenecientes a este actor.

9.2.1. Casos de uso

A continuación se mostrarán los diferentes diagramas de casos de uso para ilustrar todos los requisitos funcionales identificados en el análisis. Se dividirán los diferentes casos de uso en distintas imágenes para facilitar su visualización. Además, se distinguirá

entre requisitos ya existentes en la aplicación actual, representados con el color azul, y las nuevas funcionalidades añadidas en este TFT, representadas en color amarillo.

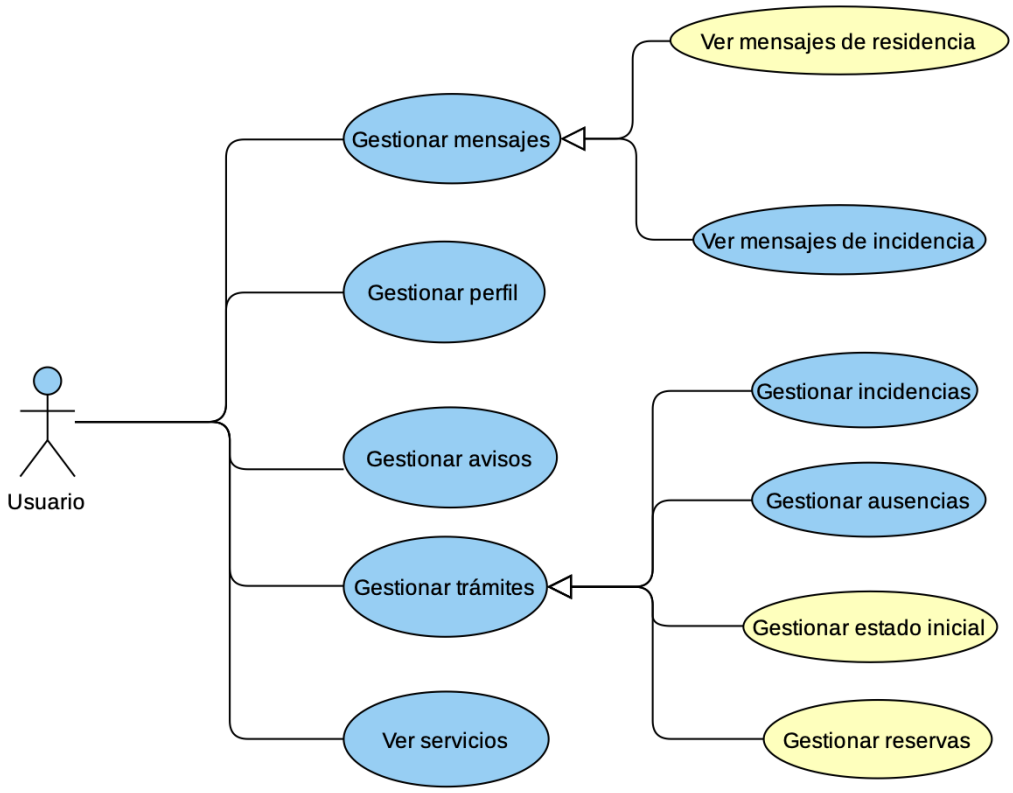

Ilustración 9.1: Diagrama de casos de uso (Resumen)

En la Ilustración 9.1 se muestran todos los casos de uso generales que se encontrarán en la aplicación, más detallados en los diagramas que se mostrarán a continuación.

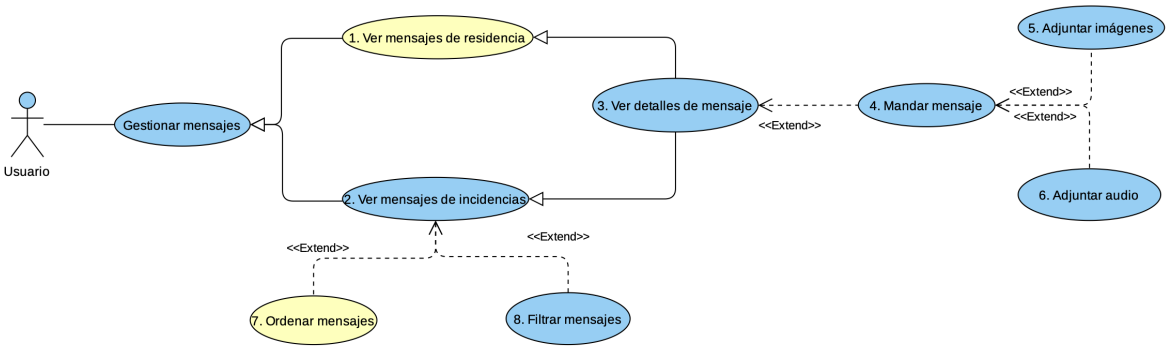

Ilustración 9.2: Diagrama de casos de uso (Gestionar mensajes)

En la Ilustración 9.2 se muestra el diagrama que describe el caso de uso de "Gestionar mensajes". En esta sección se encuentra una nueva funcionalidad añadida en este TFT: "Ver mensajes de la residencia", que consistirá en un chat reservado únicamente a la comunicación directa del personal de la residencia con el usuario.

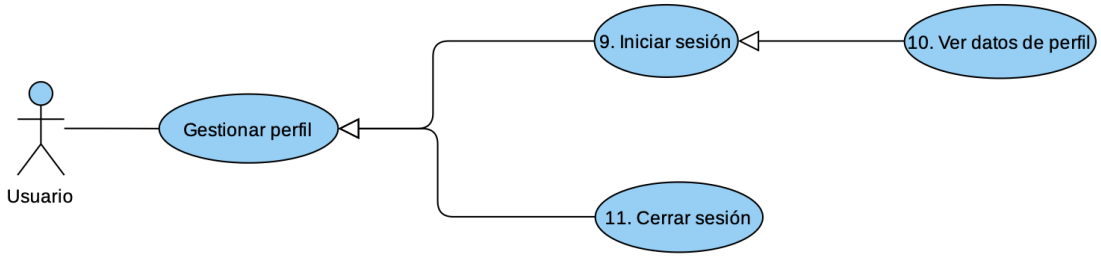

Ilustración 9.3: Diagrama de casos de uso (Gestionar perfil)

En la Ilustración 9.3 se muestra el diagrama que describe el caso de uso de "Gestionar perfil". En esta sección se encuentran las funcionalidades básicas de inicio y cierre de sesión de usuario.

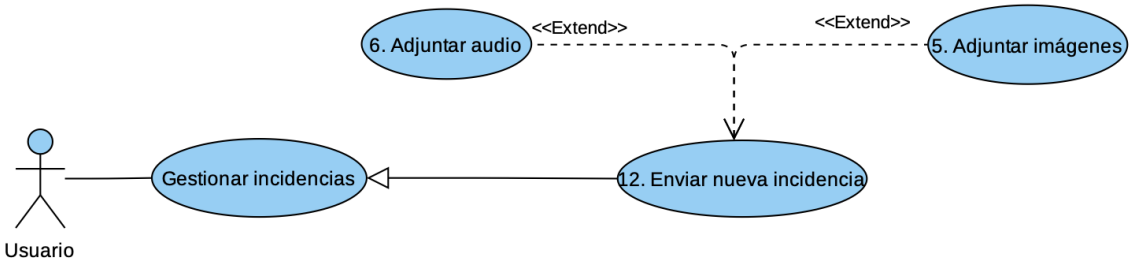

Ilustración 9.4: Diagrama de casos de uso (Gestionar incidencias)

En la Ilustración 9.4 se muestra el diagrama que describe el caso de uso de "Gestionar incidencias". Se trata de una funcionalidad replicada de la aplicación actual de Alloro, ya que es uno de los casos de uso más importantes.

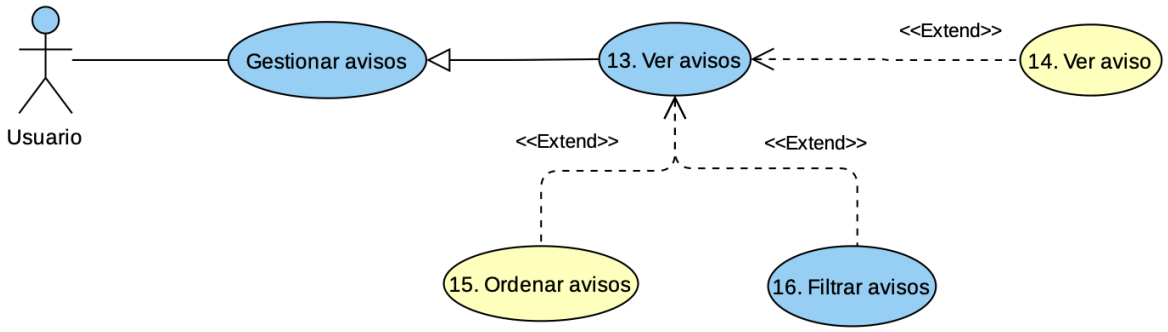

Ilustración 9.5: Diagrama de casos de uso (Gestionar avisos)

En la Ilustración 9.5 se muestra el diagrama que describe el caso de uso de"Gestionar avisos". Este caso de uso es una versión revisada de la funcionalidad de avisos de la aplicación actual.

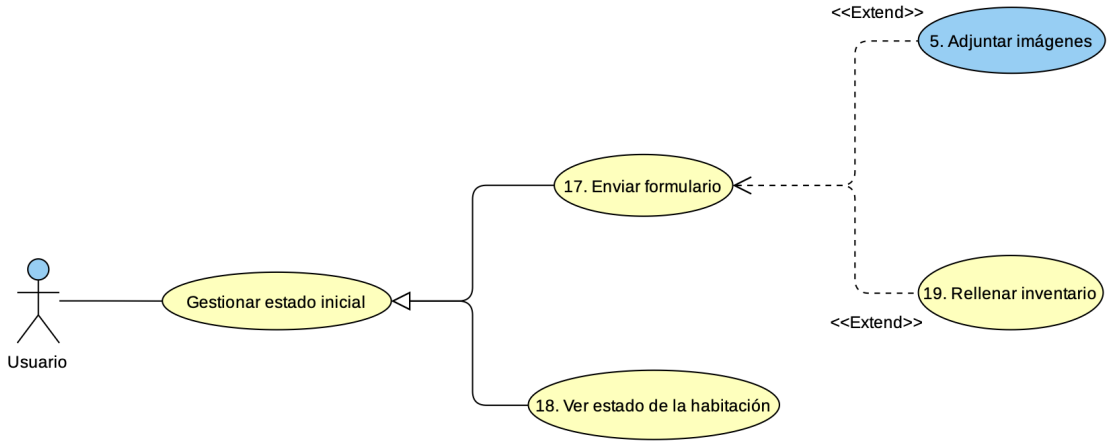

Ilustración 9.6: Diagrama de casos de uso (Gestionar estado inicial)

En la Ilustración 9.6 se muestra el diagrama que describe el caso de uso de "Gestionar estado inicial". Se trata de uno de los nuevos casos de uso que se han añadido en este TFT, y consiste en una funcionalidad para que el usuario pueda reportar el estado inicial de su habitación haciendo uso de la aplicación.

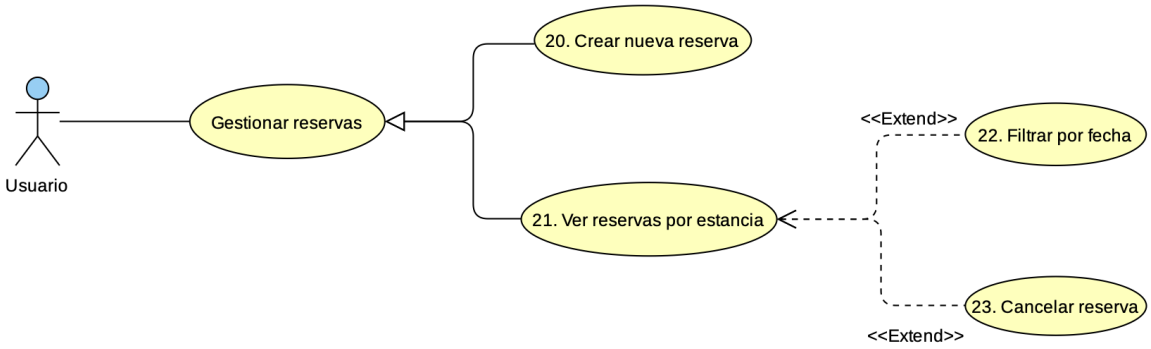

Ilustración 9.7: Diagrama de casos de uso (Gestionar reservas)

En la Ilustración 9.7 se muestra el diagrama que describe el caso de uso de "Gestionar reservas". Se trata de otro de los nuevos casos de uso añadidos, y consiste en una nueva funcionalidad para poder realizar reservas de las estancias comunes de la residencia.

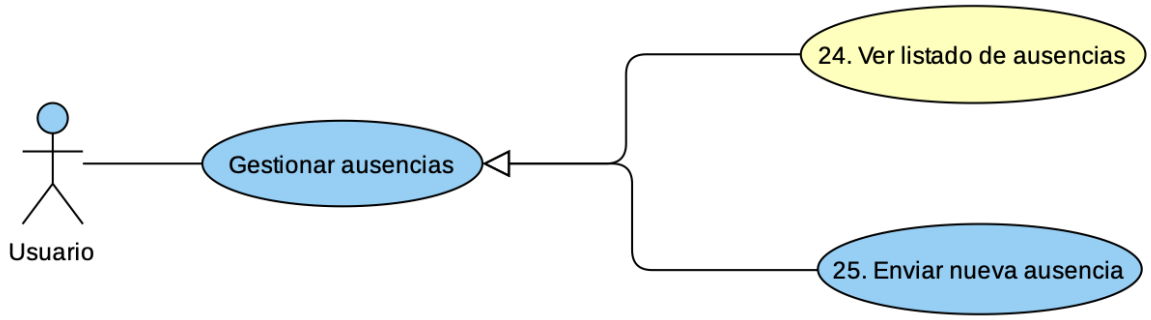

Ilustración 9.8: Diagrama de casos de uso (Gestionar ausencias)

En la Ilustración 9.8 se muestra el diagrama que describe el caso de uso de "Gestionar ausencias". Este caso de uso consiste en una versión revisada de la funcionalidad básica existente en la aplicación actual.

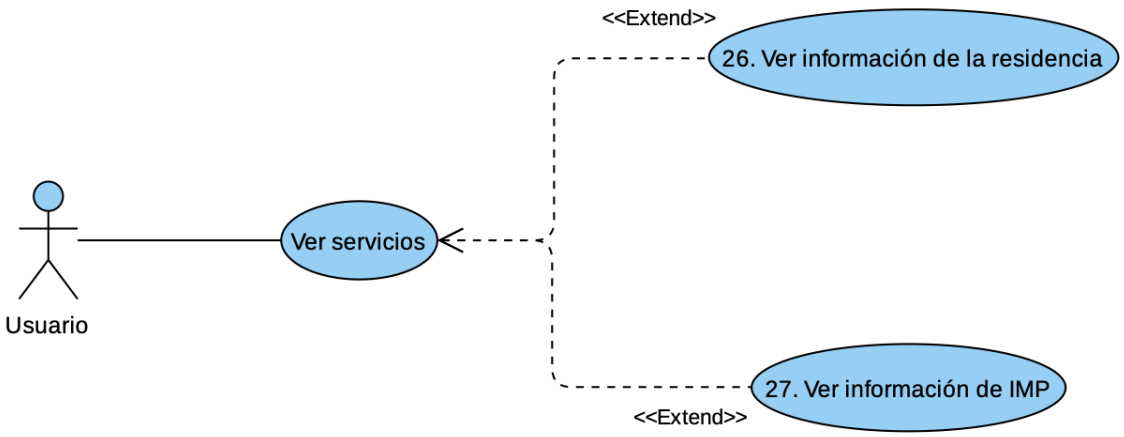

Ilustración 9.9: Diagrama de casos de uso (Ver servicios)

En la Ilustración 9.9 se muestra el diagrama que describe el caso de uso de "Ver servicios". Este caso de uso es una réplica de la funcionalidad de servicios de la aplicación en uso.

A continuación, en la Tabla 9.1, se muestra en un listado condensado los casos de uso vinculados al residente:

ID	Actor	Descripción
1	Usuario	Ver mensajes de residencia
2	Usuario	Ver mensajes de incidencias
3	Usuario	Ver detalles de mensaje
4	Usuario	Mandar mensaje
5	Usuario	Adjuntar imágenes
6	Usuario	Adjuntar audio
7	Usuario	Ordenar mensajes
8	Usuario	Filtrar mensajes
9	Usuario	Iniciar sesión
10	Usuario	Ver datos de perfil
11	Usuario	Cerrar sesión
12	Usuario	Enviar nueva incidencia
13	Usuario	Ver avisos
14	Usuario	Ver aviso
15	Usuario	Ordenar avisos
16	Usuario	Filtrar avisos
17	Usuario	Enviar formulario estado inicial

18	Usuario	Ver estado de la habitación
19	Usuario	Rellenar inventario
20	Usuario	Crear nueva reserva
21	Usuario	Ver reservas por estancia
22	Usuario	Filtrar por fecha
23	Usuario	Cancelar reserva
24	Usuario	Ver listado de ausencias
25	Usuario	Enviar nueva ausencia
26	Usuario	Ver información de la residencia
27	Usuario	Ver información de IMP

Tabla 9.1: Resumen de casos de uso para Usuario

9.3. Especificación de casos de uso

La especificación de casos de uso sirve para proporcionar una descripción del flujo y estructura que represente los requisitos funcionales de un proyecto software [24].

En el anexo A se mostrarán las especificaciones de los casos de uso que se consideran más importantes a la hora del desarrollo de la aplicación de Alloro. Como se ha mencionado anteriormente, los objetivos de este TFT abarcan únicamente las funcionalidades relacionadas con el actor de usuario o residente, por lo que será el único actor para el que se analizará sus casos de uso.

En la ilustración 9.10 se puede observar un ejemplo de especificación de caso de uso con la misma estructura que los que estarán incluidos en el anexo.

CASO DE USO	3	Ver detalles de mensaje	
Descripción	El usuario desea ver el historial de mensajes de una conversación iniciada		
Actores	Usuario		
Precondiciones	El usuario tiene que haber iniciado sesión previamente		
Flujo normal	**Paso**	**Acción**	
	1	Se accede al apartado de mensajes	
	2	Se selecciona la conversación de la que se desea ver los mensajes	
Postcondiciones	Se muestra el historial de mensajes de la conversación		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1a	El usuario quiere ordenar los mensajes	7. Ordenar mensajes
	1b	El usuario quiere buscar un mensaje en concreto	8. Filtrar mensajes
Excepciones			

Ilustración 9.10: Especificación caso de uso ver mensajes

10 Diseño

10.1. Diseño de la arquitectura del proyecto

La programación por capas consiste en una estructura de desarrollo de software que se basa en el desacoplamiento de las diferentes capas que conforman un proyecto de software. Las diferentes capas en las que se puede dividir el desarrollo de un proyecto software son las siguientes: capa de presentación o *frontend*, capa de negocio o *backend* y capa de datos [25].

En la ilustración 10.1 se puede observar el esquema general de una arquitectura por capas: la capa de presentación se comunica con la capa de negocio para realizar las peticiones del usuario. La capa de negocio se comunica con la capa de datos para poder proporcionarle una respuesta a la petición del usuario. Por último, la capa de negocio comunica esta respuesta de nuevo a la capa de presentación.

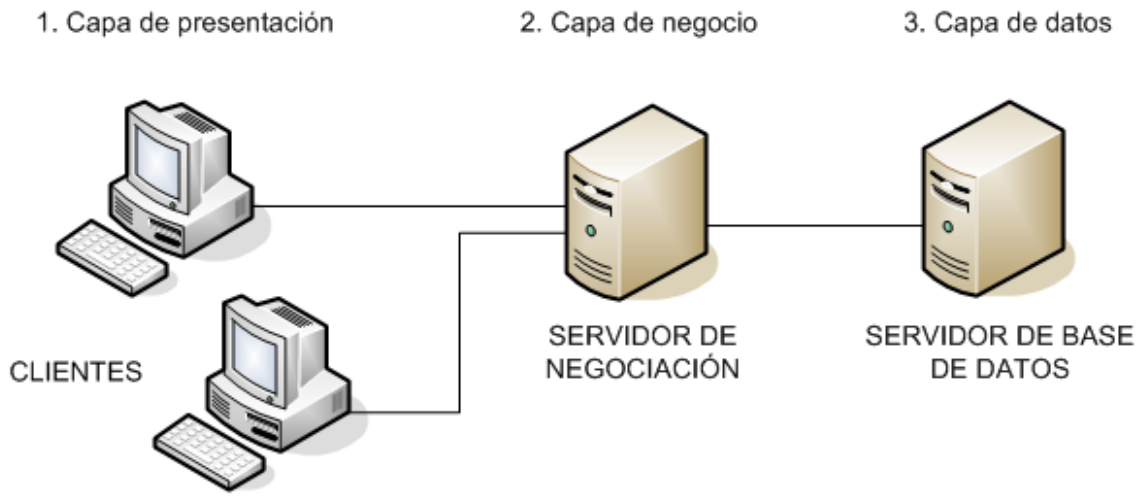

Ilustración 10.1: Esquema de arquitectura por capas. Recuperada de https://es.wikipedia.org/wiki/Programación_por_capas

Los beneficio de trabajar en el desarrollo de un proyecto software haciendo uso de este tipo de arquitectura son los siguientes:

- **Aislamiento**: uno de las ventajas de utilizar una arquitectura por capas consiste en el aislamiento que se obtiene a la hora de realizar cambios dentro de un proyecto. Cuando surja la necesidad de efectuar un cambio, este no afectará a la totalidad del proyecto, sino a un único módulo, lo que reduce el impacto de posibles cambios en el código y favorece el desarrollo de un código limpio y modular [25].
- **Fácilmente escalable**: al tener un gran nivel de desacoplamiento entre capas, se aumenta la escalabilidad del proyecto.
- **Fácil mantenimiento**: el hecho de tener las capas de un proyecto tan claramente diferenciadas tiene como consecuencia un mantenimiento mucho más sencillo de realizar.
- **Distribución de trabajo**: trabajar con este tipo de arquitecturas permite una distribución de trabajo mucho más desacoplado. Mientras un equipo de desarrollo trabaja en uno de los niveles, el resto puede trabajar en otro módulo completamente abstraído de los cambios que se están realizando [25].

Actualmente, el diseño de proyectos de software con una arquitectura por capas es el más utilizado debido a todas las ventajas mencionadas anteriormente [25].

10.1.1. Capa de presentación

La capa de presentación [25] la constituye todo aquello que ve el usuario al usar la aplicación. También se conoce como interfaz de usuario o *frontend*. En este TFT, la capa de presentación consiste en la aplicación web progresiva "Alloro", a la que tendrán acceso los residentes para la realización de trámites con la Residencia.

Para el desarrollo de esta interfaz de usuario, se ha utilizado el *framework* de desarrollo Vue, que cuenta con soporte para aplicaciones web progresivas. El diseño de la aplicación *frontend* se basó en los requisitos funcionales del usuario de la aplicación, es decir, de los residentes.

La aplicación desarrollada, al tratarse de una aplicación web progresiva, podrá utilizarse en cualquier dispositivo o sistema operativo, tanto móvil como de escritorio, en su versión web y en su versión instalable.

10.1.2. Capa de negocio

La capa de negocio [25] consiste en el programa que se encarga de recibir todas las peticiones que realizan los usuarios de la aplicación y aplicar la lógica adecuada para enviar las respuestas. Se denomina capa de negocio dado que es en esta capa donde se

establecen todas aquellas reglas que deberán cumplirse en el uso de la aplicación. Esta capa también se conoce como *backend*.

El *backend* requiere además comunicación tanto con la capa de presentación como con la capa de datos, ya que recibirá las peticiones desde el usuario y necesitará acceder a los datos para poder enviar la respuesta.

Para este desarrollo, se ha hecho uso del *framework* Laravel. Laravel es un *framework* de PHP de código abierto, diseñado para agilizar el desarrollo *backend* de aplicaciones web [5]. Mediante el uso de esta tecnología se realizó el desarrollo de una API completamente funcional que pudiese simular al completo las características necesarias para el cumplimiento de los requisitos funcionales.

10.1.3. Capa de datos

La capa de datos consiste en el lugar donde están almacenados todos los datos de la aplicación y es la capa encargada de acceder a ellos.

La base de datos utilizada para este proyecto es una base de datos MySQL. En este caso, el desarrollo de la capa de datos se ha realizado integrado con la capa de negocio, ya que Laravel cuenta con una herramienta para la interacción con la base de datos llamada Eloquent. Al utilizar Eloquent, cada tabla de a base de datos tiene asociado un "Modeloçorrespondiente, que será el que se utilizará para acceder a los datos de la tabla [26].

Las instancias de los modelos de las tablas de base de datos contendrán toda la información de las columnas que contendrá cada tabla en la base de datos, así como las relaciones que puedan existir entre distintas tablas.

Además, Eloquent proporciona otras herramientas para la administración de las bases de datos:

- **Migraciones**: las migraciones se usan para crear automáticamente las tablas que se necesiten dentro de la base de datos. Una vez creados los modelos necesarios y las migraciones asociadas correspondientes, para actualizar la base de datos sólo es necesario ejecutar el comando *"php artisan migrate:fresh"*, que se encargará de limpiar la base de datos y ejecutar todas las migraciones existentes.
- ***Seeders***: los *seeders* se utilizan para alimentar a una tabla con unos datos predefinidos. Para alimentar las tablas de la base de datos, una vez actualizada la base de datos con las migraciones existentes, se ejecuta el comando *"php artisan db:seed"*.

De esta forma, se puede utilizar las herramientas que ofrece Eloquent para la administración de la base de datos, ya sea añadir una nueva tabla a la base de datos o actualizar los datos de una tabla.

10.2. Principios de diseño

Uno de los aspectos más importantes al trabajar en el desarrollo de un proyecto de software *frontend* es considerar el diseño de interfaz de usuario que tendrá la aplicación. Para esto se tendrá en cuenta todas aquellas características que puedan favorecer a que el usuario la considere intuitiva, que le resulte sencillo relacionarse con ella y realizar las acciones que desee.

En este TFT se han seguido los principios de diseño definidos por *Ben Shneiderman*, conocidos como las "ocho reglas de oro del diseño de interfaces"[27].

1. **Consistencia**: el primero de los principios de *Shneiderman* es el de la consistencia. Para conseguir esta consistencia, es necesario buscar el desarrollo de una interfaz de usuario cohesiva. Esto se consigue con unos patrones de diseño consistentes, mediante el uso de tipografía consistente o de los colores utilizados. Debido a que en este TFT se ha hecho uso de una plantilla, ha facilitado conseguir una consistencia muy alta en el apartado gráfico [27].

2. **Informar al usuario**: otro de los apartados de diseño más importantes es el de mantener siempre al usuario informado sobre la consecuencia de las acciones que realiza sobre la aplicación [27]. Para conseguir este tipo de retroalimentación hacia el usuario, se hace uso de ventanas emergentes que confirman las acciones realizadas por el usuario, o que bien buscan una doble confirmación para realizar la acción deseada.

3. **Atajos**: en las aplicaciones actuales un aspecto muy común que se suele implementar son los atajos, ya que, para tareas que se realizan de manera recurrente, son de gran utilidad [27]. En esta aplicación se aplican todos los atajos más comunes aplicables a la mayoría de aplicaciones web: como el de copiar, pegar, entre otros.

4. **Diálogos**: el uso de los diálogos constituye una característica muy importante de un buen desarrollo de interfaces de usuario. Los diálogos se utilizan para avisar al usuario de las consecuencias producidas por las acciones realizadas [27]. En la aplicación de Alloro, se hace uso de los diálogos para avisar al usuario de las consecuencias que se producirán por una determinada acción o para realizar una doble confirmación para acciones delicadas, como puede ser cancelar una reserva.

5. **Manejo de errores**: un aspecto esencial para el desarrollo de una buena interfaz de usuario es el manejo de errores. Los sistemas deben diseñarse para estar, dentro de lo posible, a prueba de errores. Sin embargo, de vez en cuando se producirán errores inevitables. En estos casos, es imprescindible que los usuarios reciban instrucciones sencillas para resolver el problema de la forma más rápida posible [27]. En la aplicación de Alloro, se controlan todo este tipo de errores mandando siempre un mensaje al usuario explicando por qué ha sucedido el error y cómo

solucionarlo.

6. **Permitir el retroceso de acciones**: para un buen diseño de interfaz de usuario, es esencial que el usuario se sienta cómodo con sus acciones y pueda retroceder en cualquier momento. En la aplicación de Alloro, el retroceso de ciertas acciones viene determinado por los requisitos funcionales que permite la residencia. Sin embargo, siempre existirán ciertas acciones que el usuario puede revertir. A la hora de seleccionar imágenes o audio en mensajes, en el caso de equivocarse, siempre podrá seleccionar nuevas imágenes limpiando la selección anterior. De la misma forma, si se realiza una reserva, siempre tendrá la opción de cancelarla.

7. **Control interno**: otro de los principios más importantes de diseño de interfaces es conseguir que el usuario sienta que siempre tiene el control sobre lo que sucede en la aplicación; es decir, que sea siempre el usuario el que, mediante sus acciones, inicie los sucesos [27]. En la aplicación de Alloro, el usuario tiene el control completo sobre los eventos que se producen, ya que, si el usuario no realiza ningún cambio en las opciones del menú, no se lanzará ningún evento visible en la aplicación.

8. **Reducir la carga de memoria**: por último, es muy importante en un buen diseño de usuario no recurrir a la memoria de largo plazo del usuario. Esto quiere decir que hay que evitar en la medida de lo posible que el usuario tenga que recordar características específicas de la aplicación para que pueda utilizarla. Para conseguir esto, se suelen usar elementos que son familiares para el usuario. Con esto se consigue aprovechar el conocimiento previo que puede poseer el usuario sobre ciertos elementos, como puede ser la barra de menús, utilizada en una amplia gama de aplicaciones de todo tipo de contenidos, el uso de *pop ups* de confirmación, etc.

11 Desarrollo

Este TFT se compone de dos partes diferenciadas: el *frontend* y el *backend*. Para ambas partes se ha utilizado un repositorio en GitHub para alojar el código. El repositorio utilizado para el *frontend* es el siguiente: https://github.com/sseunie/alloro_pwa. Para el código de la API se ha usado el siguiente repositorio: https://github.com/sseunie/alloro-api.

11.1. Plantilla

Para el desarrollo de este TFT se decidió inicialmente hacer uso de una plantilla que redujera significativamente el tiempo de desarrollo de hojas de estilo en CSS y HTML para el diseño de interfaz de usuario.

La elección de la plantilla se centró principalmente en encontrar alguna que tuviese soporte para aplicaciones web progresivas, ya que es el aspecto más importante del desarrollo de la aplicación revisada de Alloro.

Finalmente, tras un estudio de las plantillas disponibles en el portal de *themeforest*, se decidió por la plantilla que se muestra en la Ilustración 11.1, llamada "*Azures Mobile Template & PWA*".

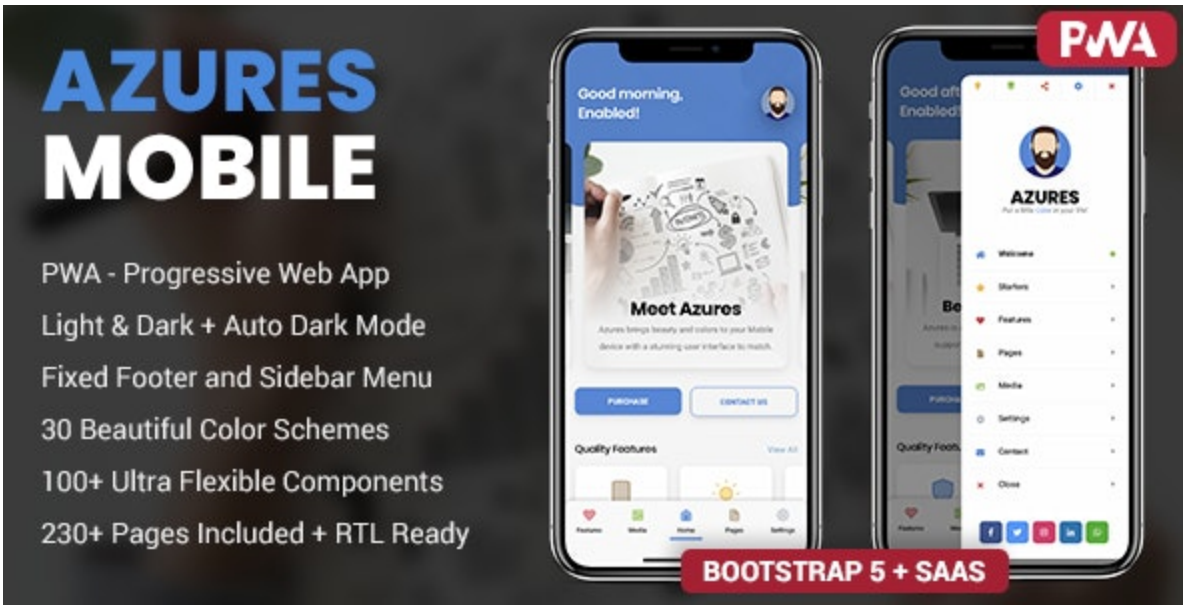

Ilustración 11.1: Plantilla seleccionada. Recuperada de https://themeforest.net/

Para facilitar el uso de la plantilla, esta cuenta con un manual de uso donde se podrá leer todas las indicaciones pertinentes además de información adicional como puede ser el uso de los colores, la estructura del HTML requerida o dónde encontrar los diferentes componentes disponibles.

Como primer paso en el desarrollo de este proyecto, se procedió a realizar un prototipo utilizando los componentes que proporcionaba la plantilla, con el objetivo de comprobar si se podría llegar a un buen resultado utilizando estos componentes. A la hora de usar todos estos componentes, se pueden encontrar en la página principal de la misma. Una vez esté decidido cuál es el que se desea utilizar, solo es necesario copiar y pegar la estructura HTML de dicho componente o página en el proyecto, teniendo siempre en consideración la estructura HTML concreta que precisa la plantilla para su buen funcionamiento. En esta parte del desarrollo se realizó una primera toma de contacto con la plantilla, sin añadir ningún tipo de funcionalidad al proyecto.

11.2. Vue 3

En un primer lugar, para el desarrollo de este TFT, se había propuesto el uso del *framework* de Angular, ya que se trata de uno de los *frameworks* más populares y que proporciona soporte a proyectos de desarrollo de aplicaciones web progresivas. Sin embargo, a continuación se argumentarán las razones por las que se decidió el cambio de tecnología de Angular a Vue.

En un primer momento, se comenzó el desarrollo del proyecto de este TFT aprendiendo las nociones básicas de Angular y poniéndolas en práctica. Sin embargo, como se ha mencionado en el apartado anterior, para el desarrollo de esta aplicación se decidió hacer uso de una plantilla. Una vez se comenzó el desarrollo en Angular del proyecto y se intentó integrar con la plantilla adquirida, surgieron una serie de problemas de uso. Debido a que la plantilla no estaba preparada para ser utilizada con *frameworks* como Angular, fue necesario realizar una serie de ajustes para intentar integrar ambas partes.

Sin embargo, después de una larga serie de pruebas, se concluyó que la mejor solución consistía en realizar un cambio de tecnologías, y se procedió a buscar una alternativa al *framework* de Angular.

La primera alternativa que seguía a Angular por popularidad fue Vue, ya que se trata de un *framework* relativamente novedoso que utiliza Javascript y guarda cierto parecido con Angular. Además, Vue cuenta también con soporte para aplicaciones web progresivas, soportando además la instalación de *plugins* para automatizar el desarrollo de las mismas.

Vue [3] es un *framework* progresivo dedicado al desarrollo de interfaces de usuario. Se trata de un *framework* muy sencillo y fácil de usar que además, se puede integrar con otras librerías de terceros o proyectos existentes muy fácilmente. El uso de Vue ofrece las siguientes ventajas:

- ***Framework MVVM***: Vue sigue un patrón de arquitectura software llamado modelo-vista-modelo de vista, (MVVM por sus siglas en inglés *model-view-viewmodel*), que se caracteriza por desacoplar la lógica de un proyecto de la interfaz de usuario [28]. Con *frameworks* de tipo MVVM se pueden conseguir proyectos con un código muy bien estructurado [29].
- **Vinculación de datos bidireccional**: Vue emplea una vinculación de datos bidireccional que se sincroniza con el esquema DOM del proyecto [29].
- ***Templates* declarativos**: uno de los aspectos más característicos de Vue es el código HTML fácilmente modificable [29].
- **DOM Virtual**: el uso de un DOM virtual proporciona un mejor rendimiento. De echo, se considera a Vue como uno de los *framworks* líderes en rendimiento de *renderizado* [29].
- **Tamaño de Vue**: se trata de un *framework* muy ligero, lo que hace que se reduzcan los tiempos de carga y aumente la velocidad de las aplicaciones web. Además, se trata de un tamaño flexible que puede aumentarse si se necesita el uso de aplicaciones o librerías de terceros [29].
- **Facilidad de uso**: Vue es considerado un *framework* más sencillo y rápido de aprender que otros como Angular o React, ya que además usa *vanilla* Javascript. Además, cuenta con una documentación muy extensa que ayuda a los desarro-

lladores a adquirir rápidamente las nociones básicas para comenzar el desarrollo [29].

Una vez pasado el proyecto a Vue, se detectó que el problema surgía por el javascript que proporcionaba la plantilla, y que no tenía que ver con el *framework* que se estaba usando. Sin embargo, por todas estas ventajas mencionadas anteriormente, además de preferencia personal, se decidió continuar el proyecto haciendo uso del *framework* de Vue.

11.2.1. Estructura del proyecto

Como la mayoría de *frameworks* de desarrollo, Vue tiene una estructura de proyectos específica que es necesario conocer para poder desarrollar un buen código. En la ilustración 11.2 se puede observar la estructura general que se obtiene después de crear un nuevo proyecto en Vue.

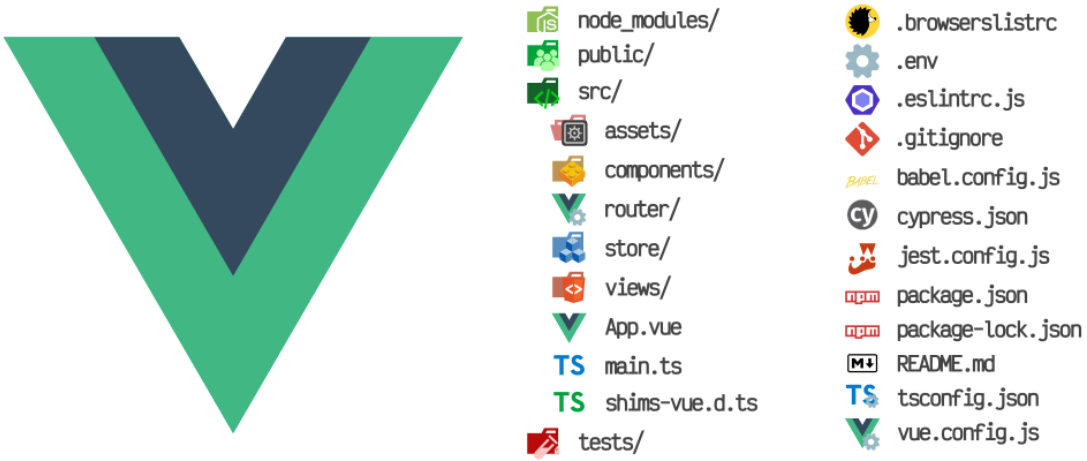

Ilustración 11.2: Estructura general de proyectos en Vue. Recuperada de: https://lenguajejs.com/vuejs/introduccion/estructura-carpetas/

Comenzando por el primer eslabón en la estructura del proyecto, el directorio raíz, se explicarán los archivos y carpetas existentes en el proyecto de este TFT más relevantes:

- **.env**: contiene todas las variables de entorno del proyecto. En este fichero se guarda como variable de entorno la URL de las llamadas a la API, para acceder a ella desde cualquier parte del proyecto.
- **.gitignore**: indica los ficheros o carpetas que deben ser ignorados por Git.
- **package.json**: se trata de un fichero JSON que contiene la información de todas las dependencias de Node.js utilizadas en el proyecto.

- **node_modules**: en esta carpeta es donde se encuentran todas las dependencias que se generan con NPM (el gestor de paquetes).
- **public**: en esta carpeta se almacenarán los ficheros estáticos que no necesitan ser procesados por Vue, como son el icono de la aplicación o el fichero index.html.
- **src**: se trata de la carpeta más importante del proyecto ya que es donde se almacenan todos los archivos que configuran el código fuente del proyecto.
    - **App.vue**: el fichero App.vue es un componente especial en los proyectos de Vue, ya que se trata del componente sobre el que se cargarán todos los demás componentes que conformen la aplicación.
    - **main.js**: es el fichero encargado de cargar Vue y todos los *plugins* necesarios.
    - **registerServiceWorker.js**: este fichero aparece en el directorio si se trata de aplicaciones PWA. Se encarga de registrar el *service worker* necesario para la funcionalidad de aplicación progresiva.
    - **assets**: se utiliza para almacenar ficheros estáticos como imágenes, a los que no se podrá acceder fuera del código del proyecto.
    - **scripts**: esta carpeta fue creada con el objetivo de almacenar todos los *scripts* adicionales que fuesen necesarios durante el desarrollo, como pueden ser los *scripts* de las llamadas a la API o de la *store* de Vuex.
    - **components**: en esta carpeta se almacenan todos los diferentes componentes en Vue que se han ido creando a lo largo del desarrollo del proyecto.

11.3. Laravel

Para el desarrollo del *backend* de la aplicación, se decidió inicialmente realizar una API completamente simulada, es decir, ejecutar un servidor que simule la lógica real del *backend* al recibir las llamadas de la aplicación en Vue.

En un principio, se intentó implementar este servidor haciendo uso de la librería *json-server* para realizar esta tarea. Sin embargo, la dificultad que surgía a la hora de simular el funcionamiento real era que estas librerías no permitían la carga de imágenes o audio, una característica esencial en los requisitos funcionales de este proyecto.

Como consecuencia, se decidió finalmente utilizar una herramienta en la que se podría desarrollar una API completamente realista y funcional aunque con alguna de las funcionalidades limitadas.

Para el desarrollo de esta API se ha utilizado la herramienta de Laravel, debido a que ofrece las siguientes ventajas:

- **Seguridad**: con Laravel resulta muy sencillo implementar medidas de seguridad [30].
- **Mejor rendimiento**: a diferencia de otros *frameworks* en PHP, Laravel soporta el almacenamiento en caché, lo que mejora considerablemente el rendimiento de una web [30].
- **Facilidad de integración con librerías de terceros**: Laravel hace que la integración con librerías de terceros se realice sin dificultades [30].
- **Escalabilidad**: gracias a la naturaleza escalable de PHP y el soporte incorporado por Laravel, la escalabilidad de Laravel constituye uno de sus fuertes [30].
- **Eloquent**: Laravel incluye una herramienta llamada *Eloquent* para la interacción con la base de datos. Cada base de datos tiene un "Modelo"que se usará para interactuar con su tabla. Además, los modelos de *eloquent* permiten insertar, actualizar y eliminar fácilmente registros de la tabla [5].
- **Simple y eficaz**: por último, se trata de uno de los *frameworks* más usados de PHP, por lo que cuenta con una documentación muy extensa que convierte a Laravel en un *framework* muy fácil de aprender [30].

11.3.1. Estructura del proyecto

La estructura de proyectos utilizada para este TFT se basa en la que proporciona Laravel por defecto al crear un nuevo proyecto. A continuación se describirán los archivos y carpetas más relevantes, partiendo desde la raíz del proyecto [31]:

- **app**: en esta carpeta se almacenará el código principal del proyecto. Dentro de este directorio se pueden encontrar las siguientes carpetas:
  - ***Events***: esta carpeta no viene creada por defecto. Para crearla se ejecutan los comandos *php artisan event:generate* o *php artisan make:event*. En esta carpeta se almacenan todos los eventos que alertan a otros componentes de la aplicación de que ha sucedido un evento en concreto.
  - **Http**: este directorio contiene todos los controladores de las clases del proyecto.
  - ***Listeners***: esta carpeta no existe por defecto. Para crearla, se ejecutan los comandos *php artisan event:generate* o *php artisan make:listener*. En ella se almacenan todas las clases que administran los eventos, es decir, escuchan a los eventos y realizan la lógica necesaria de respuesta al mismo.
  - ***Models***: en esta carpeta se encuentran todos los modelos de *Eloquent*. Cada tabla de la base de datos tiene su "Modelo.ᵉᑫuivalente que es el que se utilizará para interactuar con esa tabla.

- **bootstrap**: dentro de esta carpeta se encuentra el fichero *app.php* que arranca el *framework*.
- **config**: en esta carpeta se almacenan los ficheros de configuración de la aplicación.
- ***database***: dentro de esta carpeta se encuentran todas las migraciones y los *seeders* para la administración de la base de datos.
  - ***migrations***: dentro de esta carpeta se almacenan las migraciones de la base de datos.
  - ***seeders***: dentro de esta carpeta se encuentran los *seeders* con los que se alimentará la base de datos.
- ***public***: en esta carpeta se almacenan los ficheros estáticos, como el fichero index.php.
- ***storage***: en esta carpeta se almacenan los ficheros generados por el usuario, como pueden ser imágenes subidas por los usuarios de la aplicación.
- ***routes***: en esta carpeta se almacenan todas las definiciones de las rutas de la aplicación. Por defecto, Laravel crea algunos ficheros, de los cuales los más importantes en este proyecto son los siguientes:
  - ***api.php***: contiene todas las rutas disponibles en la API de la aplicación.
  - ***channels.php***: en este fichero se registran todos aquellos canales de transmisión de eventos que se necesiten en la aplicación.
- ***vendor***: contiene los archivos de las dependencias instaladas a través de Composer.

11.3.2. Enrutamiento

Para obtener los datos de la base de datos desde el *frontend*, se ha creado una serie de rutas de peticiones de datos HTTP.

Los métodos necesarios para manejar los datos de la base de datos se basan en las cuatro acciones CRUD (del inglés *Create, Read, Update, Delete*):

- **POST**: para crear un nuevo recurso (*Create*)
- **GET**: para obtener un recurso (*Read*)
- **PATCH**: para actualizar parcialmente un recurso en concreto (*Update*)
- **DELETE**: para eliminar un recurso (*Delete*)

La elección del uso del método PATCH en vez PUT es debido a que se trata de un método más adecuado para las peticiones en las que el objetivo no es actualizar el recurso al completo sino una parte del mismo.

```
Route::post( uri: '/login', [AuthController::class, 'login']);

Route::get( uri: '/notifications', [NotificationsController::class, 'getNotifications']);

Route::get( uri: '/residences', [IncidencesController::class, 'getResidences']);
Route::get( uri: '/incidenceAreas', [IncidencesController::class, 'getIncidenceAreas']);

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/users/{id}', [UsersController::class, 'getUser']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/users/{id}/room', [UsersController::class, 'setRoomInitialState']);
Route::middleware( middleware: 'auth:sanctum')->get( uri: '/roomState-date/{id}', [UsersController::class, 'getRoomInitialStateFinishDate']);
Route::middleware( middleware: 'auth:sanctum')->get( uri: '/users/{id}/inventory', [UsersController::class, 'userInventory']);

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/incidences', [IncidencesController::class, 'getIncidences']);
Route::middleware( middleware: 'auth:sanctum')->get( uri: '/incidences/{id}', [IncidencesController::class, 'getIncidence']);
Route::middleware( middleware: 'auth:sanctum')->patch( uri: '/incidences/{id}', [IncidencesController::class, 'updateReadStatus']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/incidences', [IncidencesController::class, 'createIncidence']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/incidences/{id}/messages', [IncidencesController::class, 'createMessage']);

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/absences', [AbsencesController::class, 'getAbsences']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/absences', [AbsencesController::class, 'createAbsence']);

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/reservationRoomTypes', [ReservationsController::class, 'roomTypes']);
Route::middleware( middleware: 'auth:sanctum')->get( uri: '/reservations', [ReservationsController::class, 'reservations']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/reservations', [ReservationsController::class, 'newReservation']);
Route::middleware( middleware: 'auth:sanctum')->delete( uri: '/reservations/{id}', [ReservationsController::class, 'cancelReservation']);

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/chats/{id}', [ChatsController::class, 'getChat']);
Route::middleware( middleware: 'auth:sanctum')->post( uri: '/chats/{id}', [ChatsController::class, 'createMessage']);
Route::middleware( middleware: 'auth:sanctum')->patch( uri: '/chats/{id}', [ChatsController::class, 'updateReadStatus']);
```

Ilustración 11.3: Rutas creadas para la API

En la creación de las rutas se ha intentado seguir una estructura coherente y similar para todas las peticiones, como se puede observar en la ilustración 11.3.

Al declarar las rutas en Laravel, es necesario hacer uso de la clase *Route*. A continuación, dependerá si la ruta a declarar precisa de un mecanismo de autorización o no. Si la ruta es de acceso público, se indicará el método HTTP que se utilizará (*get, post, patch o delete*). Adicionalmente, si la URL es de acceso privado, se hará uso del *middleware* Sanctum, una herramienta que utiliza Laravel por defecto para llevar a cabo la autenticación y autorización. Por último, será necesario indicar la URL de la petición, el controlador que gestionará la petición y el método que contendrá la lógica de respuesta a dicha petición.

11.3.3. Seguridad

La seguridad de la aplicación se implementó siguiendo dos conceptos: autenticación y autorización. Ambos procesos se implementaron haciendo uso de la herramienta Laravel Sanctum, que se trata de un paquete de Laravel que se utiliza para emitir *tokens*

de autenticación a los usuarios. Laravel Sanctum [32] ofrece esta característica almacenando los *tokens* de la API del usuario en una única tabla de la base de datos y autorizando las peticiones HTTP entrantes a través de la cabecera *Authorization* que debe contener un *token* válido de la API.

Por una parte, la autenticación se lleva a cabo cuando el usuario inicia sesión en la aplicación (*Login*). Este método, si el resultado es un inicio de sesión correcto, devuelve un JWT necesario para el proceso de autorización. JWT (por sus siglas en inglés, *JSON Web Token*) se trata de un estándar que define una manera de transmitir información de forma segura que podrá ser verificable, ya que son *tokens* firmados digitalmente [33]. Con esto se consigue que cada usuario obtenga su propio *token* para posteriormente utilizarlo en el proceso de autorización.

Finalmente, cuando el usuario realiza un petición a la API en una ruta protegida por Sanctum, deberá indicar el *token* JWT obtenido en el proceso de autenticación. Este se deberá indicar en el encabezado *Authorization* de la petición como un *Bearer token*. Si Laravel comprueba que este token es válido, autorizará al usuario y le permitirá realizar la consulta.

11.4. Despliegue en Heroku

Debido a que era necesario realizar pruebas y comprobar el correcto funcionamiento de la aplicación progresiva y la instalación en diferentes dispositivos, se decidió realizar el despliegue del proyecto usando la plataforma de Heroku, ya que permite subir los proyectos de forma gratuita.

Para realizar el despliegue de la aplicación a la plataforma [34], el primer paso a realizar es crear una cuenta e instalar el CLI de Heroku. A continuación se creará una nueva *app* dentro de Heroku, en la que se subirá el proyecto. Debido a que este TFT se compone de dos partes diferenciadas, el *backend* y el *frontend*, se han creado dos aplicaciones diferentes en Heroku para poder proceder con el despliegue de ambas a la plataforma.

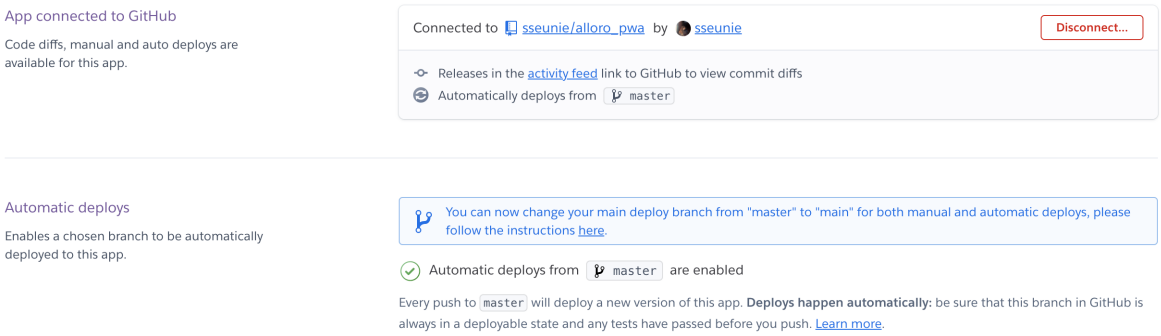

Ilustración 11.4: Despliegue automático en Heroku

Una vez creadas las aplicaciones en Heroku, existe la posibilidad de enlazar el repositorio en GitHub a la aplicación. La ventaja de realizar el despliegue de la aplicación de esta forma reside en que se puede configurar de forma que siempre que se realice un *push* a una rama determinada, la aplicación en Heroku se actualiza de manera automática. Como consecuencia, este proceso elimina la necesidad de realizar un despliegue manual. En la ilustración 11.4 se puede observar la configuración seleccionada para el despliegue de la aplicación de Vue. La rama seleccionada para que se actualice ha sido la rama *master*, ya que en el desarrollo de este TFT se ha trabajado siguiendo la rama *develop* para la mayoría de los cambios en el desarrollo y la rama *master* para los *releases* con ciertos niveles de funcionalidad.

Finalmente, Heroku proporciona una URL predeterminada para la aplicación creada. Al acceder de esta manera a la aplicación, se pudo comprobar si podía instalarse en los dispositivos y que cumpliese con los requisitos necesarios para el buen desarrollo de una PWA.

11.5. Comprobación de requisitos PWA

Como último paso del desarrollo, se comprobó que la aplicación desarrollada cumpliera con todos los requisitos necesarios para considerarse una aplicación web progresiva.

Para ello se usó una herramienta [35] desarrollada por Microsoft con el objetivo de promover el desarrollo de aplicaciones web progresivas llamada "*PWA Builder*". Esta herramienta se usa como apoyo para el desarrollo de ciertos aspectos de una PWA como los *service workers* o el *web manifest*. Además, cuenta con un apartado en el que permite enviar la URL de una aplicación web progresiva y recibir una puntuación de acuerdo al funcionamiento de la misma. Usando esta herramienta se envió la URL de la aplicación progresiva de Alloro desplegada en Heroku para comprobar que todos los

requisitos de PWA estuvieran cubiertos y se recibió la puntuación que se puede observar en la ilustración 11.5.

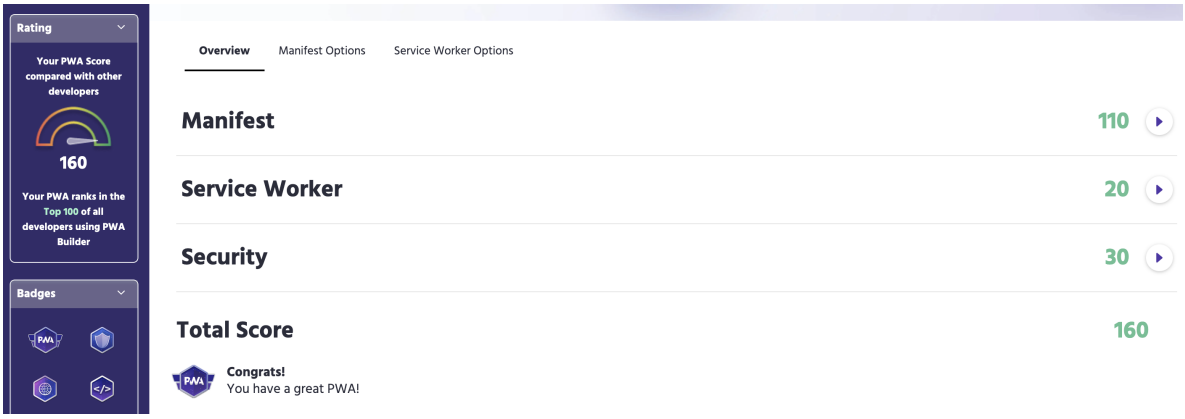

Ilustración 11.5: Puntuación obtenida en PWA Builder

En las siguientes ilustraciones 11.6, 11.7 y 11.8 se puede observar los requisitos obligatorios y recomendados del manifiesto JSON, de *service workers* y de seguridad para las aplicaciones PWA y la puntuación recibida en cada uno de ellos.

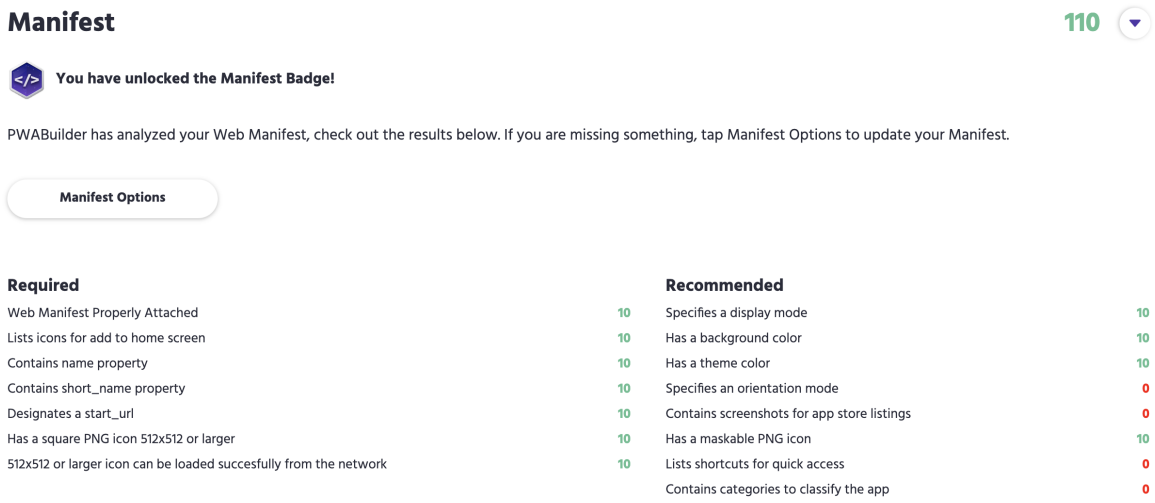

Ilustración 11.6: Requisitos de manifiesto web para PWA

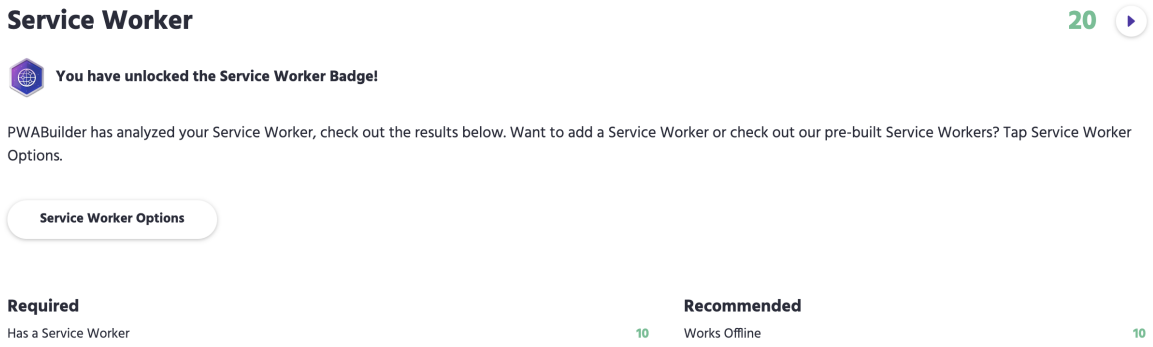

Ilustración 11.7: Requisitos de service worker para PWA

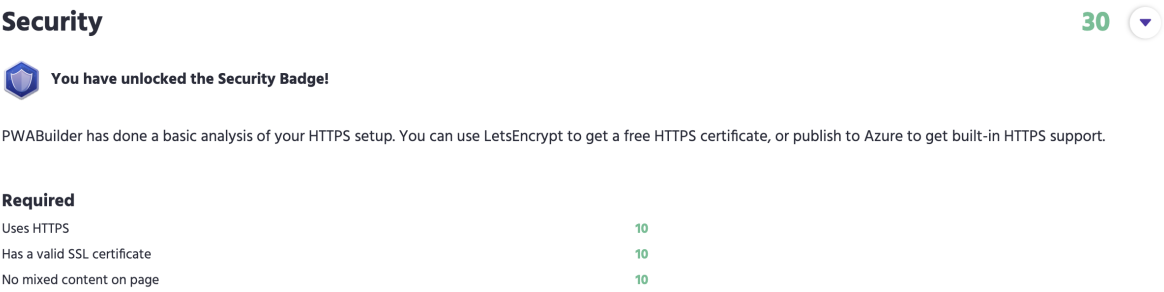

Ilustración 11.8: Requisitos de seguridad para PWA

De esta forma, al recibir una buena puntuación en cada uno de estos apartados, se concluyó el desarrollo de la aplicación web progresiva.

12 Resultados y conclusiones

12.1. Resultados y conclusiones

Los objetivos de este TFT se pueden dar por completados, ya que, gracias al desarrollo de este proyecto, se ha conseguido obtener una aplicación web progresiva que cumple con las características y requisitos planteados al comienzo del desarrollo.

Para este TFT se ha partido de una aplicación multiplataforma de la que actualmente hacen uso los residentes de la Residencia Universitaria de Tafira, la Residencia Universitaria de Las Palmas, los Apartamentos Universitarios de Tafira y los Bungalows. Esto ha supuesto un punto de partida para este proyecto, ya que el principal objetivo consistía en una migración de tecnologías: partiendo de una aplicación multiplataforma para obtener una aplicación web progresiva. Al realizar esta migración, se consigue una nueva versión revisada que resultará mas sencilla de mantener a lo largo del tiempo, ya que no es necesario tener diferentes flujos de desarrollo para la misma aplicación para los distintos sistemas operativos, sino que una misma versión funcionará para todos los dispositivos. Además, con esta transformación se pretende simplificar y agilizar el mantenimiento y desarrollo de nuevas características sin que exista una alta dependencia respecto a las actualizaciones de los sistemas operativos iOS y Android.

Como resultado de este trabajo se ha obtenido una aplicación web progresiva con un diseño consistente y responsivo, fácil e intuitiva de usar y que además, al contar con funcionalidades adicionales, aporta un valor real al usuario final. Adicionalmente, también se ha obtenido una API desarrollada según las necesidades del *frontend*. Si bien esta es una versión simulada y limitada respecto a la API real, cumple con todas las funcionalidades necesarias para el buen funcionamiento de la aplicación.

Al tratarse de un proyecto tan amplio, en comparación con el resto de trabajos realizados durante el transcurso del grado de Ingeniería Informática, se han superado muchas de las fases reales de desarrollo de un proyecto. Esto supone una visión más completa de todos estos conocimientos aprendidos previamente, ya que, aunque se han ido realizando trabajos prácticos en el transcurso del grado, la puesta en común de todos ellos en un proyecto más amplio proporciona una imagen real de cómo se relacionan todos estos conocimientos de las diferentes asignaturas entre sí.

Si bien durante el transcurso del grado universitario no se explican las tecnologías concretas utilizadas en este TFT, se enseña una serie de conocimientos básicos los

cuales conforman los cimientos de todas aquellas tecnologías que se deseen aprender a utilizar. En última instancia, esto es más importante que aprender estas tecnologías tan concretas, dado que el mundo de la informática está en constante desarrollo y siempre surgirán nuevas tecnologías, con lo cual el hecho de contar con unos buenos conocimientos básicos es esencial.

Dicho esto, este TFT ha brindado la oportunidad de sumergirse en el mundo de los frameworks de desarrollo y seleccionar el que mejor se adapte o más ventajas proporcione. De esta forma, durante el progreso del proyecto se han podido aprender algunas tecnologías relativamente modernas como pueden ser Vue, Laravel o PWA. Todo este aprendizaje ha aportado un gran valor a nivel profesional.

12.2. Trabajo futuro

Aunque el proyecto realizado cumple con los objetivos básicos, todavía quedan ciertos aspectos en los que se podría seguir trabajando.

Debido a la reciente pandemia, se ha hecho evidente el beneficio que proporciona la posibilidad de realizar ciertos trámites de manera *online* o no presencial. En este TFT se añaden nuevas funcionalidades que permiten la realización *online* de ciertas gestiones que antes consistían en tramitaciones en papel en la propia recepción de la residencia. Sin embargo, aun quedan varias funcionalidades que se podrían añadir como parte de la aplicación, con el objetivo de reducir cada vez más el número de trámites que necesiten hacerse presenciales.

Por otro lado, se podría añadir notificaciones *push* a la aplicación de Alloro. Debido a que la nueva aplicación consiste en una aplicación web progresiva, es posible implementar este tipo de notificaciones. Sin embargo, puede tener sus dificultades, ya que la manera de realizar esta implementación dependerá en gran medida del sistema operativo en el que se use la aplicación, por lo que puede ser un trabajo futuro que aportaría gran valor al usuario.

Como última sugerencia de trabajo futuro a realizar, podría llevarse a cabo la integración del *frontend* realizado en este proyecto con la lógica del *backend* real del que hace uso la residencia. Si bien en este proyecto se han añadido nuevas funcionalidades, el *backend* realizado sirve como una estructura base desde la que partir, ya que, aunque se trata de una API parcialmente simulada, se ha realizado siguiendo buenas prácticas y requerimientos necesarios por parte del *frontend*.

Anexos

A Especificación de casos de uso

ANEXO I: Especificación de casos de uso

CASO DE USO	3	Ver detalles de mensaje	
Descripción	El usuario desea ver el historial de mensajes de una conversación iniciada		
Actores	Usuario		
Precondiciones	El usuario tiene que haber iniciado sesión previamente		
Flujo normal	**Paso**	**Acción**	
	1	Se accede al apartado de mensajes	
	2	Se selecciona la conversación de la que se desea ver los mensajes	
Postcondiciones	Se muestra el historial de mensajes de la conversación		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1a	El usuario quiere ordenar los mensajes	7. Ordenar mensajes
	1b	El usuario quiere buscar un mensaje en concreto	8. Filtrar mensajes
Excepciones			

Ilustración 1: Especificación de caso de uso ver mensajes

CASO DE USO	4	Mandar un mensaje	
Descripción		El usuario desea mandar un nuevo mensaje	
Actores		Usuario	
Precondiciones		El usuario tiene que haber iniciado sesión previamente	
Flujo normal	**Paso**	**Acción**	
	1	Se accede al apartado de mensajes	
	2	Se selecciona la conversación a la que se desea responder	
	3	Se escribe en el cuadro de texto un mensaje	
	4	Se envía el mensaje	
Postcondiciones		El mensaje se enviará y se mostrará en el historial de la conversación	
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	3a	El usuario desea adjuntar una imagen	5. Adjuntar imágenes
	3b	El usuario desea adjuntar un audio	6. Adjuntar audio
Excepciones			

Ilustración 2: Especificación de caso de uso mandar mensaje

CASO DE USO	12	Enviar nueva incidencia	
Descripción		El usuario desea reportar una nueva incidencia	
Actores		Usuario	
Precondiciones		El usuario tiene que haber iniciado sesión previamente	
Flujo normal		**Paso**	**Acción**
		1	Se accede al apartado de incidencias
		2	Se selecciona una residencia
		3	Se selecciona el área al que pertenece la incidencia
		4	Se escribe un título
		5	Se añade una descripción a la incidencia
		6	Se envía la nueva incidencia
Postcondiciones		Se envía la incidencia y se muestra en el listado de conversaciones	
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**
		5a	El usuario desea adjuntar una imagen	5. Adjuntar imágenes
		5b	El usuario desea adjuntar un audio	6. Adjuntar audio
Excepciones			

Ilustración 3: Especificación de caso de uso enviar nueva incidencia

CASO DE USO	14	Ver aviso		
Descripción		El usuario desea ver un aviso que ha enviado la residencia		
Actores		Usuario		
Precondiciones				
Flujo normal		**Paso**	**Acción**	
		1	Se accede al apartado de avisos	
		2	Se selecciona el aviso que se desea ver en detalle	
Postcondiciones		Se mostrarán los detalles del aviso		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
		1a	El usuario quiere ordenar los avisos	15. Ordenar avisos
		1b	El usuario quiere buscar un aviso concreto	16. Filtrar avisos
Excepciones				

Ilustración 4: Especificación de caso de uso ver aviso

CASO DE USO	17	Enviar formulario estado inicial	
Descripción		El usuario desea rellenar el formulario de estado inicial de su habitación	
Actores		Usuario	
Precondiciones		El usuario tiene que haber iniciado sesión previamente, estar dentro del plazo indicado y no haber enviado previamente el formulario.	
Flujo normal		**Paso**	**Acción**
		1	Se accede al apartado de habitación
		2	Se rellena el cuadro de texto describiendo el estado de la habitación
		3	Se rellena el inventario
		4	Se envía el formulario
Postcondiciones		Se mostrará el estado de la habitación que ha sido enviado	
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**
		3a	El usuario quiere adjuntar imágenes	5. Adjuntar imágenes
Excepciones			

Ilustración 5: Especificación de caso de uso enviar formulario estado inicial

CASO DE USO	20	Crear nueva reserva	
Descripción	El usuario desea realizar una nueva reserva para una estancia de la Residencia		
Actores	Usuario		
Precondiciones	El usuario tiene que haber iniciado sesión previamente		
Flujo normal	**Paso**	**Acción**	
	1	Se accede al apartado de reservas	
	2	Se selecciona nueva reserva	
	3	Se selecciona una estancia	
	4	Se selecciona la fecha de la reserva	
	5	Se selecciona una hora del listado de horas disponibles	
	6	Se confirma la reserva	
Postcondiciones			
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			

Ilustración 6: Especificación de caso de uso crear nueva reserva

CASO DE USO	21	Ver reservas por estancia		
Descripción		El usuario desea ver las reservas futuras para una estancia determinada		
Actores		Usuario		
Precondiciones		El usuario tiene que haber iniciado sesión previamente		
Flujo normal		**Paso**	**Acción**	
		1	Se accede al apartado de reservas	
		2	Se selecciona la estancia de la que se desea ver las reservas en el apartado “Ver mis reservas”	
Postcondiciones		Se muestran todas las reservas futuras realizadas para esa estancia		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
		2a	El usuario quiere filtrar las reservas por fecha	22. Filtrar por fecha
Excepciones				

Ilustración 7: Especificación de caso de uso ver reservas por estancia

CASO DE USO	23	Cancelar una reserva	
Descripción		El usuario desea cancelar una reserva que ha realizado anteriormente	
Actores		Usuario	
Precondiciones		El usuario tiene que haber iniciado sesión y haber realizado una reserva previamente	
Flujo normal		**Paso**	**Acción**
		1	Se accede al apartado de reservas
		2	Se selecciona la estancia de la que se desea ver las reservas en el apartado “Ver mis reservas”
		3	Se selecciona cancelar reserva
		4	Se confirma la cancelación de la reserva
Postcondiciones		Se cancela la reserva y se elimina de la lista de reservas pendientes	
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**
		2a	El usuario quiere filtrar las reservas por fecha	22. Filtrar por fecha
Excepciones			

Ilustración 8: Especificación de caso de uso cancelar una reserva

CASO DE USO	25	Enviar nueva ausencia	
Descripción		El usuario desea enviar una nueva ausencia	
Actores		Usuario	
Precondiciones		El usuario tiene que haber iniciado sesión previamente	
Flujo normal		**Paso**	**Acción**
		1	Se accede al apartado de ausencias
		2	Se selecciona “Nueva ausencia”
		3	Se indica una fecha de salida
		4	Se indica una fecha de vuelta
		5	Se escriben las observaciones
		6	Se envía la nueva ausencia
Postcondiciones		Se registra la nueva ausencia y se muestra en el listado	
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones			

Ilustración 9: Especificación de caso de uso enviar nueva ausencia

B Manual de usuario

1 Anexo II: Manual de usuario

1.1. Introducción

El manual de usuario sirve como apoyo para entender de una manera clara y sencilla el funcionamiento de la app desarrollada y todas las funcionalidades permitidas en ella. A lo largo de este manual se irán añadiendo capturas ilustrativas a la vez que una pequeña explicación para lograr un buen entendimiento sobre el funcionamiento de la app.

La página web desde la que se puede acceder a la aplicación desde cualquier dispositivo con acceso a internet es la siguiente: https://alloro-pwa.herokuapp.com/.

1.2. Ventana de instalación

Ilustración 1.1: Mensaje de confirmación de instalación

Al entrar a la aplicación web, si no está instalada en el dispositivo, aparecerá una ventana de confirmación con la opción de instalar la aplicación en el dispositivo que se esté usando, como se puede observar en la Ilustración 1.1.

Al seleccionar “Instalar”, aparecerá otra ventana de confirmación, en el caso de estar usando un navegador, como se puede observar en la Ilustración 1.2.

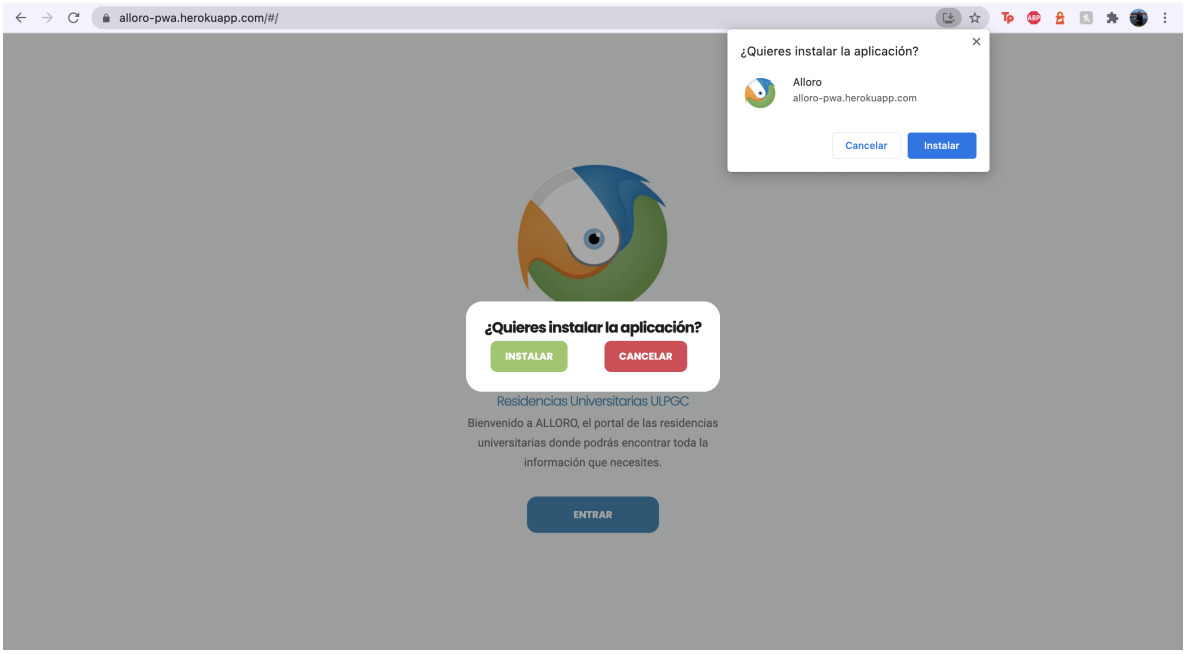

Ilustración 1.2: Ventana de instalación en el navegador

1.3. Inicio

El menú de inicio será el que se verá al entrar a la aplicación. Desde este menú, se podrá acceder a algunas de las principales funcionalidades de la aplicación, como se muestra en la Ilustración 1.3.

Ilustración 1.3: Menú de inicio

Para volver a esta pantalla de inicio desde cualquiera de las distintas pantallas de la aplicación, solo sería necesario seleccionar el tercer icono, contando de izquierda a derecha, de la barra de menú inferior.

1.4. Barra de menú

Como en la mayoría de aplicaciones móviles, Alloro cuenta con una barra de menú para acceder rápidamente a ciertas funcionalidades. En la Ilustración 1.4 se pueden observar los cuatro iconos disponibles con los que cuenta la barra de menú. Estos accesos directos son los siguientes, de izquierda a derecha:

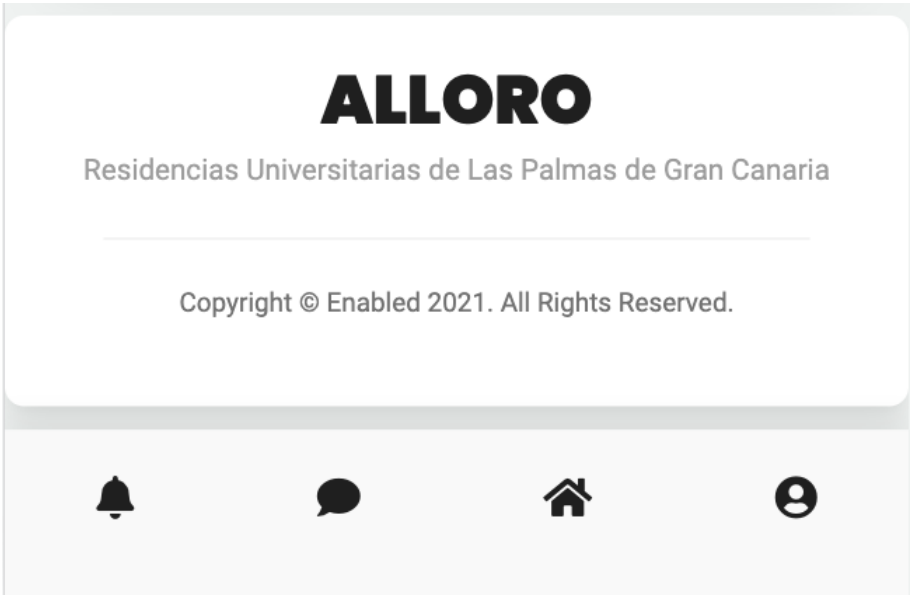

Ilustración 1.4: Barra de menú

- Avisos: se accede al listado de avisos.
- Mensajes: se accede al listado de mensajes.
- Inicio: se accede a la pantalla inicial.
- Mi perfil: se accede a los datos de perfil, o al formulario de inicio de sesión si no hay una sesión iniciada previamente.

1.5. Inicio de sesión y perfil

Para acceder a ciertas funcionalidades de la aplicación será necesario haber iniciado sesión previamente. Para iniciar sesión, se podrá acceder directamente al apartado de "Mi perfil", en la barra de menú inferior, y aparecerá directamente el formulario para iniciar sesión si no hay una sesión iniciada previamente. Si por otro lado intentamos acceder a alguna de las funcionalidades reservadas a usuarios con sesión iniciada, se cargará la misma pantalla de formulario que se puede observar en la Ilustración 1.5. Para iniciar sesión se especificará la dirección de correo electrónico del usuario y su contraseña correspondiente.

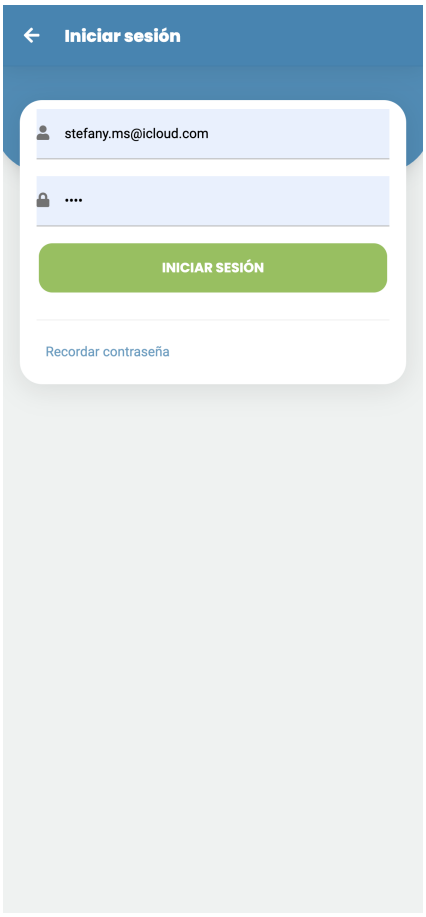

Ilustración 1.5: Formulario de inicio de sesión

Una vez iniciada la sesión, se redirigirá a la página de “Mi perfil”, como se puede observar en la Ilustración 1.6, donde se verán todos los datos del usuario que ha iniciado sesión, además de la opción de cerrar la sesión.

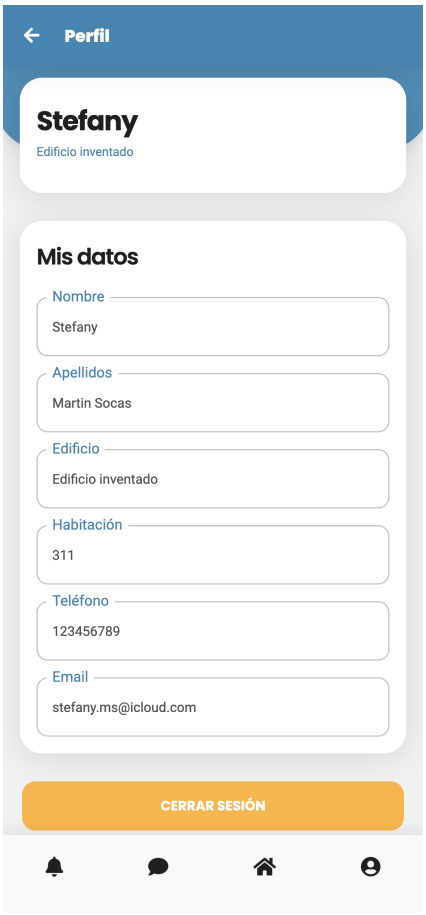

Ilustración 1.6: Mi perfil

1.6. Funcionalidades permitidas sin inicio de sesión

En la aplicación de Alloro existen ciertas funcionalidades que están permitidas para todos los usuarios, independientemente de si han iniciado sesión previamente o no.

1.6.1. Avisos

Se accederá a la pantalla de avisos a través de la barra de navegación inferior de la aplicación, haciendo *click* en el icono ubicado más a la izquierda. Una vez en la ventana de avisos, se puede ver un listado de diferentes avisos que haya mandado la residencia previamente, como se puede ver en la Ilustración 1.7.

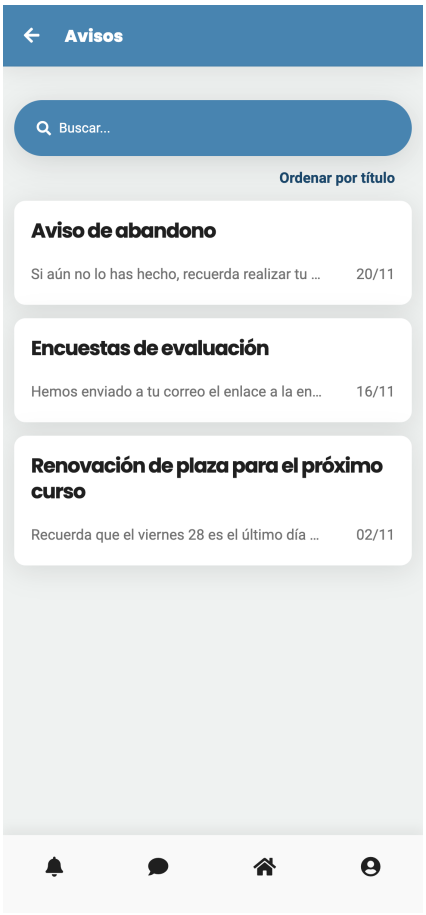

Ilustración 1.7: Listado de avisos

En este listado aparecerá una lista de diferentes avisos en los que se podrá ver un pequeño fragmento del mensaje que contiene. Al seleccionar uno de los avisos se podrá ver el texto al completo además de acceder a los enlaces si los hubiera, como se muestra en la Ilustración 1.8.

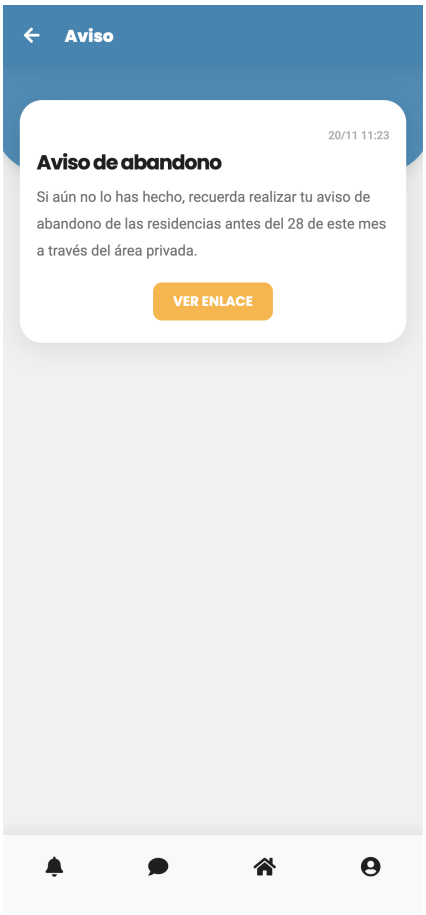

Ilustración 1.8: Detalles de un aviso

Por último, se pueden filtrar los avisos por palabras clave usando la barra de búsqueda, como se muestra en la Ilustración 1.9.

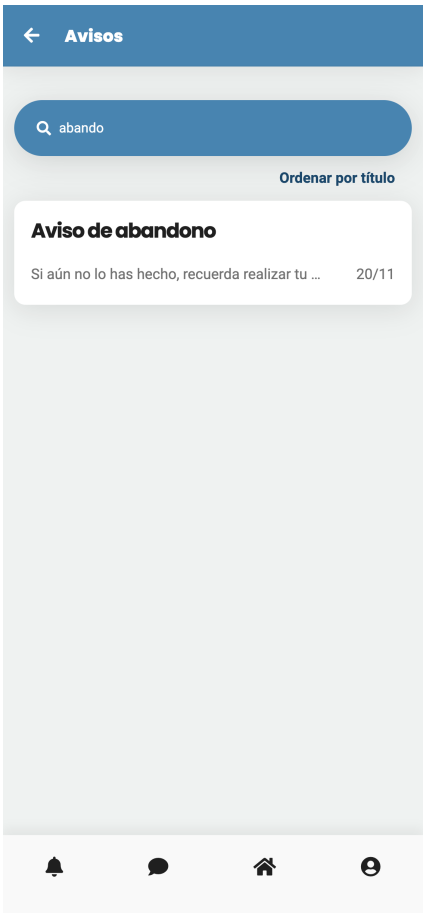

Ilustración 1.9: Avisos filtrado por palabra clave

1.6.2. Servicios

Otra de las funcionalidades accesibles para los usuarios sin sesión iniciada es la de servicios. Esta funcionalidad se accede a través de la página de inicio, mostrada previamente en la Ilustración 1.3.

Ilustración 1.10: Servicios

En esta sección se podrán ver en detalle los servicios de IMP y de la propia residencia, como se muestra en la Ilustración 1.10. Si seleccionamos uno de estos servicios, se mostrarán los datos de teléfono, correo, ubicación y página web, como se observa en la Ilustración 1.11.

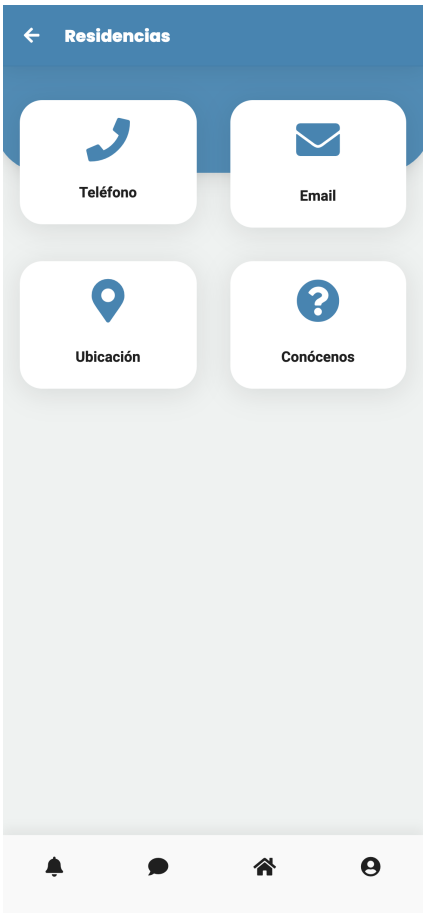

Ilustración 1.11: Detalles de servicios

1.6.3. Información

Se podrá acceder a la pantalla de información desde el menú de inicio visto anteriormente. Esta es la última de las funcionalidades accesibles para usuarios que no han iniciado sesión.

Ilustración 1.12: Información

En esta sección se podrá ver información sobre Alloro y condiciones de uso, como se muestra en la Ilustración 1.12.

1.7. Incidencias

Desde el menú inicial, y con la sesión iniciada, se puede acceder al apartado de "Incidencias". En este apartado, el usuario podrá reportar nuevas incidencias a la residencia rellenando el formulario que se muestra en la Ilustración 1.13.

Ilustración 1.13: Formulario para añadir nueva incidencia

Para reportar una nueva incidencia, será necesario rellenar por completo el formulario. Primero, se seleccionará de un desplegable la residencia sobre la que se desea reportar la incidencia, como se muestra en la Ilustración 1.14.

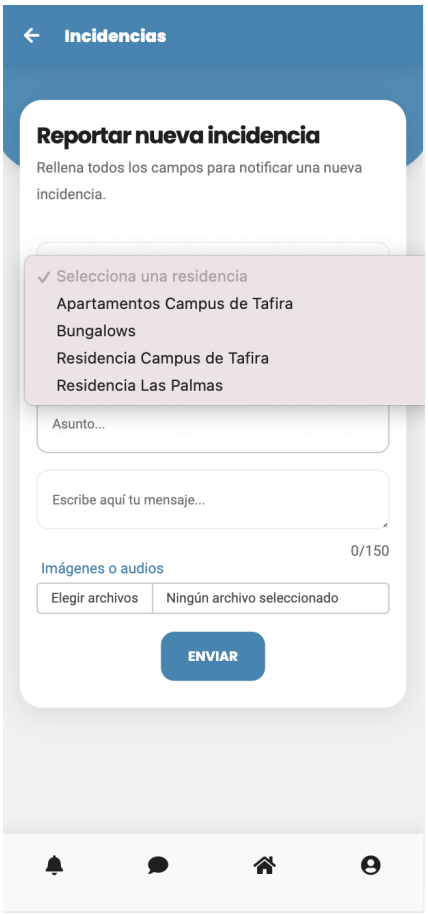

Ilustración 1.14: Desplegable para seleccionar residencia

De la misma forma que se selecciona la residencia, se deberá seleccionar el área indicada de la incidencia, como puede ser cocina, administración o mantenimiento, entre otras. Después de seleccionar la residencia y el área indicada, se deberá rellenar una pequeña descripción de la incidencia a reportar.

En primer lugar, se deberá añadir un título descriptivo a la incidencia. Además, se deberá rellenar una breve descripción de la misma. Este mensaje no podrá superar los 150 caracteres.

Por último, el formulario permite la opción de añadir archivos adjuntos si se desea. Los tipos de archivos que se aceptan serán archivos de imágenes o de audio.

Una vez rellenado el formulario, al hacer *click* en "Enviar", se mostrará una ventana emergente confirmando que la nueva incidencia se ha creado correctamente, como se observa en la Ilustración 1.15.

Ilustración 1.15: Nueva incidencia creada correctamente

1.8. Mensajes

En el apartado de mensajes se mostrarán todas las incidencias creadas previamente y el proceso de seguimiento de las mismas además de un chat fijo con el que se facilitará la comunicación directa del residente con la gestión de la residencia.

Después de crear una nueva incidencia, al aceptar la ventana emergente que aparece, se redirigirá por defecto al apartado de mensajes. Para acceder a ella desde cualquier otra parte de la aplicación, se seleccionará el segundo icono, contando de izquierda a derecha, de la barra de menú. Por defecto, el listado de mensajes de las incidencias estará ordenado por fecha, como se muestra en la Ilustración 1.16.

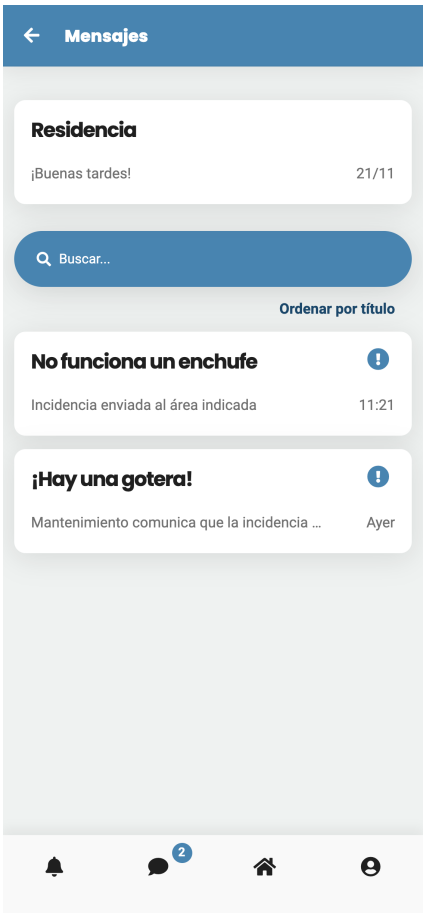

Ilustración 1.16: Listado de mensajes

En el apartado de mensajes, en primer lugar se mostrará el chat con la residencia. Este chat siempre estará disponible para enviar algún mensaje a la gestión de la residencia.

Seguidamente, después de una barra de búsqueda similar a la que se pudo ver anteriormente en el apartado de avisos, se listarán todas las incidencias reportadas previamente. Además, es posible cambiar el orden por el que se muestra este listado. Para seleccionar el método por que el que se quiere ordenar el listado, será necesario seleccionar entre las opciones de "Ordenar por fechaz "Ordenar por título", lo que hará que se cambie el orden del listado por fecha o por título respectivamente. En la Ilustración 1.17 se puede observar el listado de mensajes ordenado por título.

Además, en todas aquellas incidencias en las que existan mensajes no leídos se mostrará un icono de exclamación para indicar al usuario que aún no ha leído los nuevos mensajes de esa incidencia.

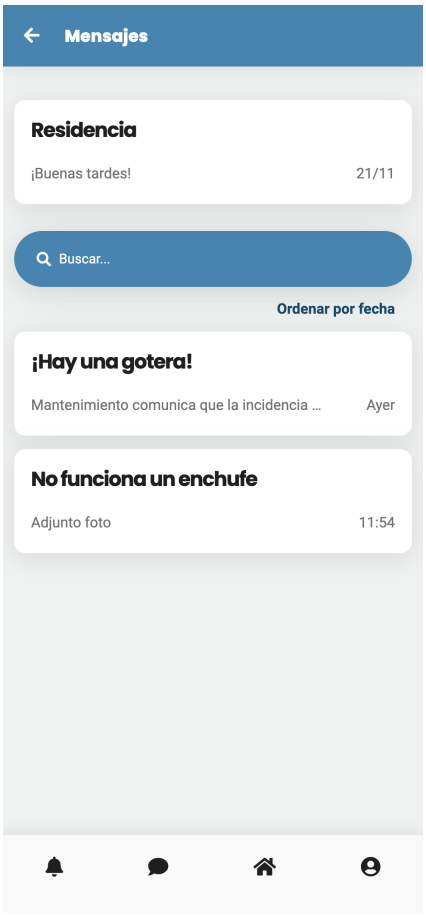

Ilustración 1.17: Listado de mensajes ordenado por título

Para acceder a los detalles de mensajes de una incidencia bastará con hacer *click* sobre la misma. Esto redirigirá a la pantalla que se muestra en la Ilustración 1.18.

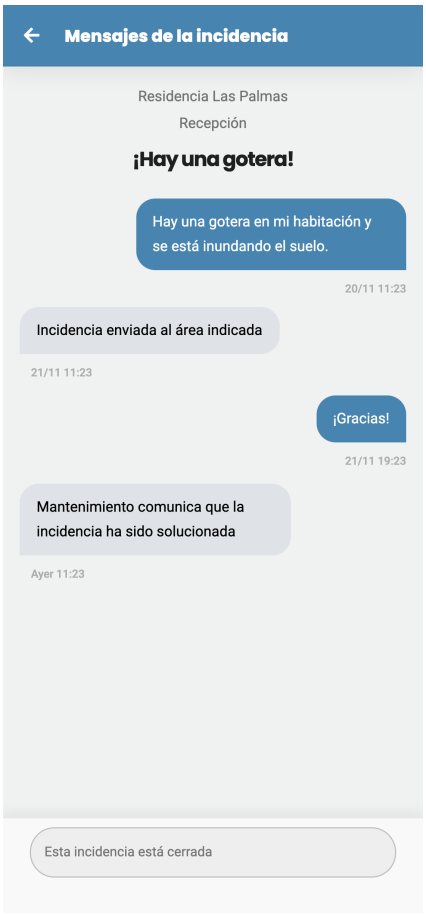

Ilustración 1.18: Detalles de una incidencia

En la Ilustración 1.18 se puede observar una incidencia que se encuentra cerrada. Esto quiere decir que no será posible responder más a esta incidencia, ya que ha sido considerada resuelta por el personal de la residencia.

Si accedemos a los detalles de una incidencia que se considera resuelta no será posible enviar un nuevo mensaje. Sin embargo, si accedemos a una incidencia que esté todavía abierta, se permitirá al usuario enviar un nuevo mensaje, e incluso adjuntar ficheros de imágenes o audio al mensaje.

Haciendo *click* en el botón izquierdo de la barra inferior en la conversación se podrá adjuntar hasta 3 ficheros adjuntos. En la Ilustración 1.19 se puede observar cómo se indica la cantidad de ficheros adjuntos seleccionados en el mensaje y la vista preliminar del texto del mensaje que se enviará.

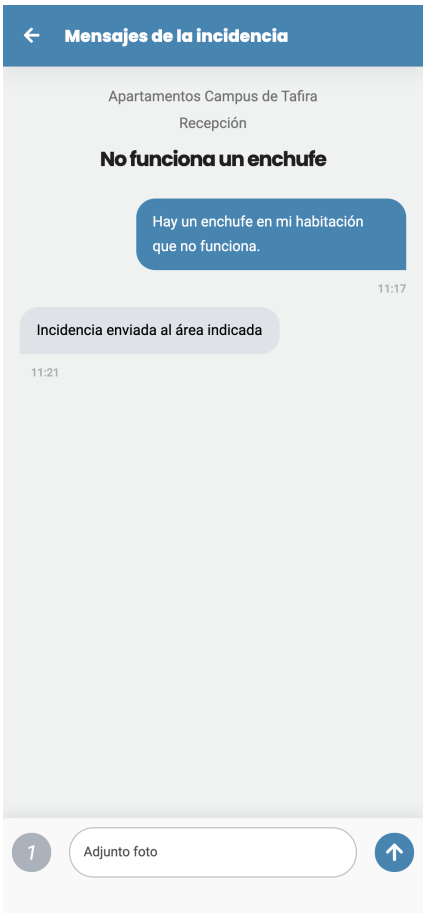

Ilustración 1.19: Nuevo mensaje

Al enviar el mensaje, este se mostrará inmediatamente en la conversación. Las imágenes o audio adjuntados se mostrarán junto al texto del mensaje en la conversación, como se muestra en la Ilustración 1.20.

Ilustración 1.20: Nuevo mensaje enviado con foto adjunta

1.9. Ausencias

Desde el menú inicial se puede acceder al apartado de ausencias. En este apartado se podrá ver un listado de ausencias programadas, además de poder añadir nuevas ausencias. Este apartado puede observarse en la Ilustración 1.21.

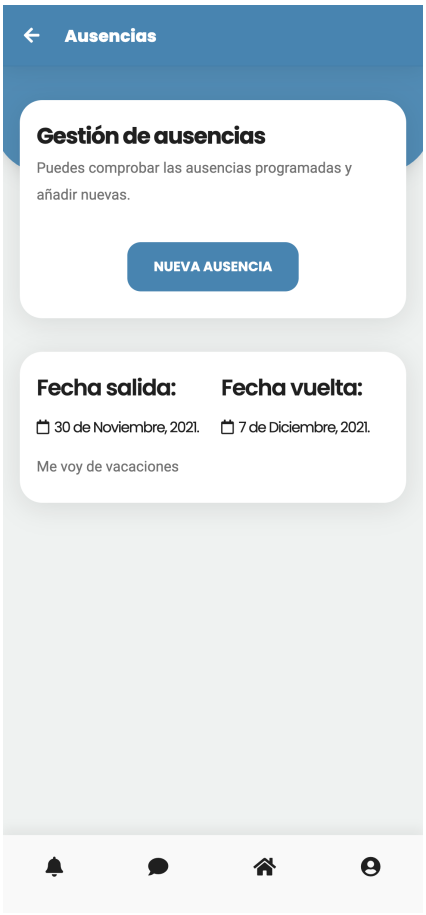

Ilustración 1.21: Apartado de ausencias

Para poder añadir una nueva ausencia, se hará *click* en el botón de "Nueva ausencia". Seguidamente se abrirá un formulario que habrá que rellenar para cumplimentar los datos de la nueva ausencia como se muestra en la Ilustración 1.22.

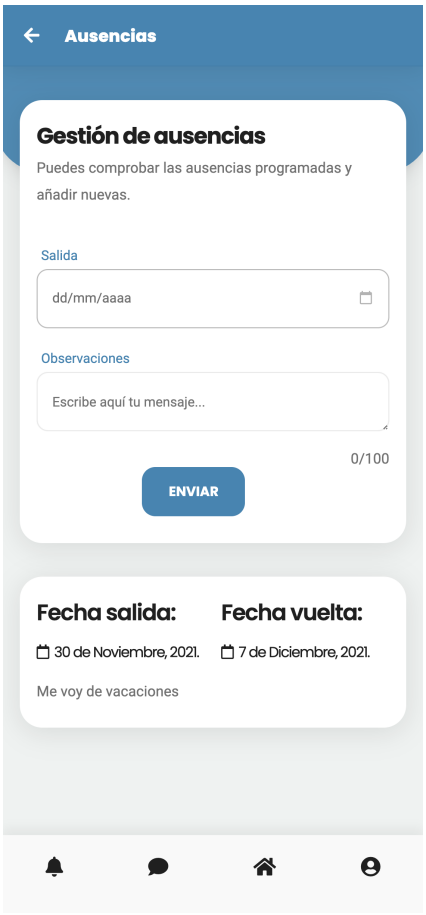

Ilustración 1.22: Formulario para nueva ausencia

En primer lugar será necesario seleccionar la fecha de comienzo de la ausencia. Una vez seleccionada esta fecha, aparecerá otro apartado para seleccionar la fecha de regreso. Solo se permitirá seleccionar una fecha mayor a la que se seleccionó anteriormente, como se puede apreciar en la Ilustración 1.23. Además, se puede rellenar un pequeño apartado de descripción de la ausencia si se desea.

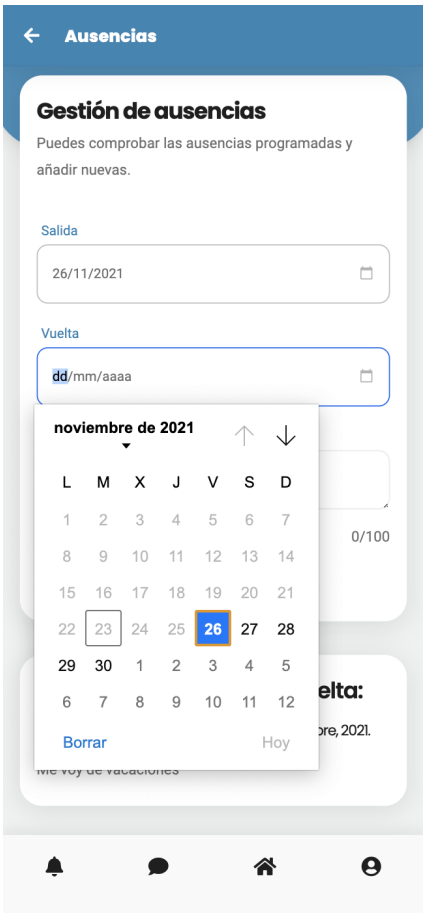

Ilustración 1.23: Selección de fecha de regreso en ausencia

Una vez enviada la ausencia se mostrará automáticamente en el listado.

1.10. Reservas

En el apartado de reservas se pueden añadir nuevas reservas para las diferentes estancias de la residencia, además de ver las reservas futuras. La pantalla principal del apartado de reservas se divide en dos secciones: la gestión de reservas, donde se puede crear una nueva reserva; y la sección en la que se podrá ver las reservas pendientes para las distintas estancias. En la Ilustración 1.24 se puede observar la pantalla principal de la sección de reservas.

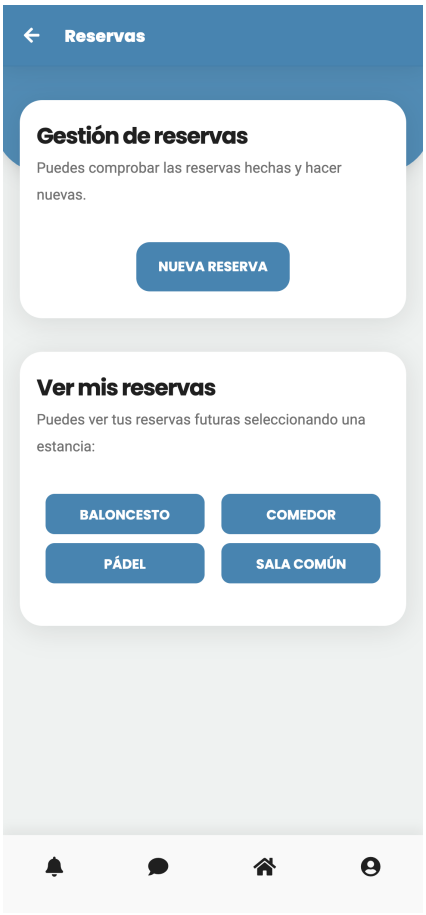

Ilustración 1.24: Pantalla principal de reservas

Al hacer *click* en el botón de "Nueva reserva", se mostrará un formulario para realizar una nueva reserva, como se muestra en la Ilustración 1.25. Para realizar una nueva reserva, en primer lugar se seleccionará la estancia en la que se desea reservar y la fecha indicada. Al seleccionar la estancia y la fecha, aparecerá un desplegable para seleccionar la hora de la reserva, como se muestra en la Ilustración 1.26. Dependiendo de la estancia y el día que se seleccione, aparecerán diferentes horas disponibles, debido a que cada estancia tendrá un aforo determinado.

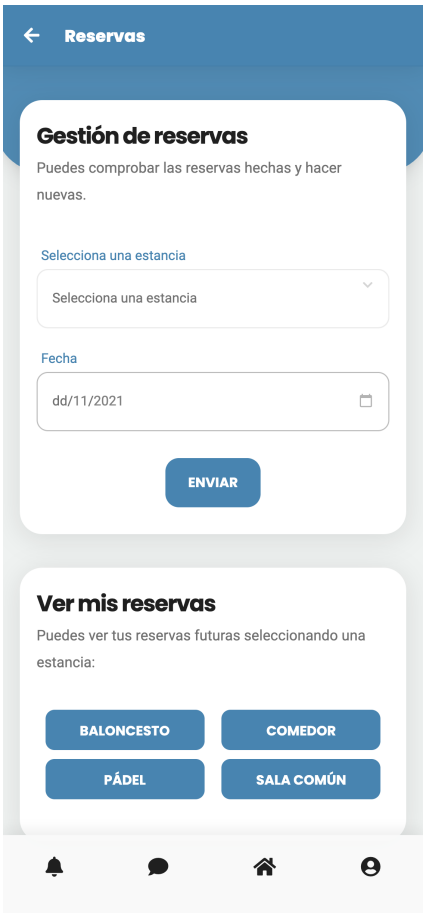

Ilustración 1.25: Formulario de nueva reserva

Ilustración 1.26: Desplegable para seleccionar la hora de la reserva

Al realizar una nueva reserva, el formulario se cerrará automáticamente. Para poder ver la reserva realizada, será necesario acceder a la sección de "Ver mis reservas", seleccionando la estancia indicada. Una vez seleccionada, se mostrarán todas las reservas futuras para esa estancia y los detalles de la misma, como se muestra en la Ilustración 1.27. Además, a la hora de ver todas las reservas para una estancia, es posible aplicar un filtro de fecha para ver únicamente las reservas de un día específico. Una vez aplicado el filtro, aparecerá una nueva opción situada antes del listado de las reservas para eliminar el filtro de fecha si se desea, como se puede observar en la Ilustración 1.28.

Ilustración 1.27: Reservas futuras para baloncesto

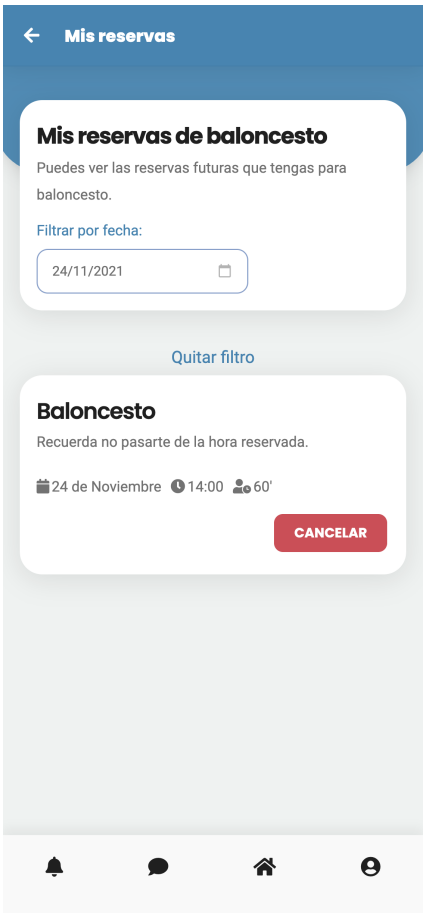

Ilustración 1.28: Filtro de fecha para las reservas de baloncesto

1.11. Estado inicial

La última funcionalidad disponible para los usuarios con la sesión iniciada es la de rellenar el estado inicial de la habitación al comienzo del curso escolar.

Este formulario lo habilita la gestión de la residencia, por lo que tiene una fecha límite en la que se puede rellenar. Debido a esto, se mostrarán pantallas diferentes dependiendo de si la fecha actual está dentro del plazo o no.

Si el plazo para rellenar el formulario ha concluido y no se envió previamente el estado inicial de la habitación, aparecerá un mensaje de aviso como se muestra en la Ilustración 1.29 y no se permitirá realizar ninguna acción más.

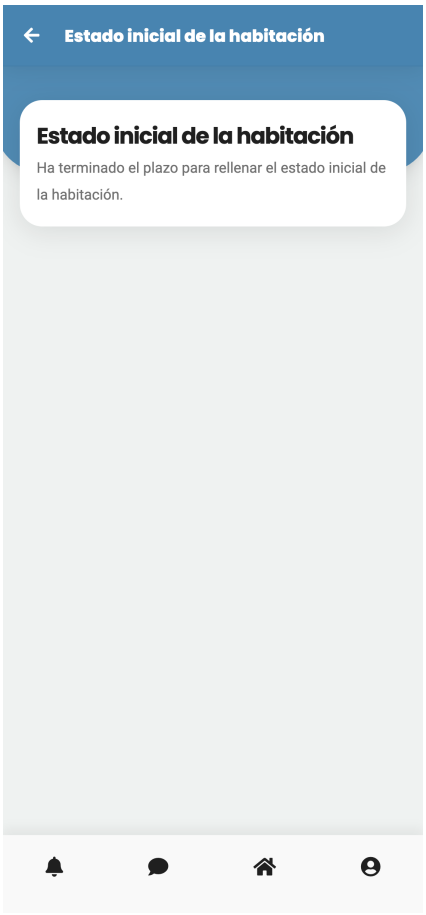

Ilustración 1.29: Plazo terminado para el estado inicial

Sin embargo, si la fecha actual se encuentra dentro del plazo indicado y no hemos enviado el estado inicial, se mostrará un formulario y la fecha límite para rellenar el mismo, como se puede observar en la Ilustración 1.30. Para rellenar este formulario será necesario escribir una descripción del estado de la habitación con todos aquellos desperfectos que puedan encontrarse, además de adjuntar fotos como apoyo al texto. Además, se deberá seleccionar en una lista de *checkbox* si la habitación cuenta con todas las utilidades que debería.

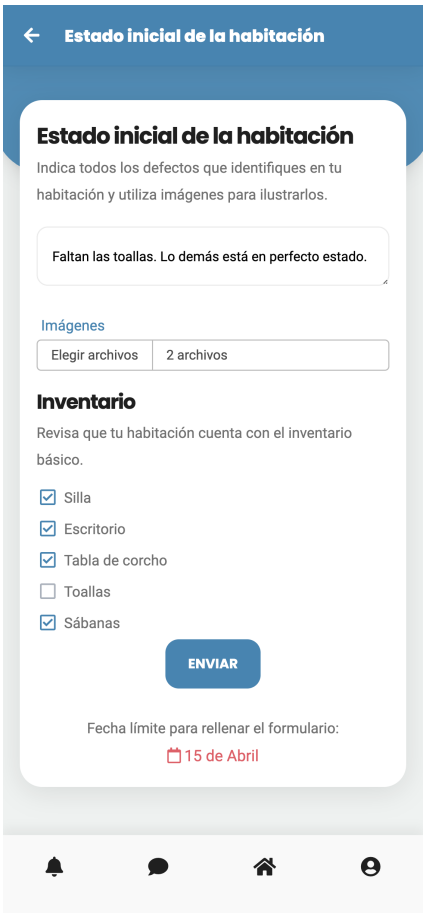

Ilustración 1.30: Formulario estado inicial

Si se intenta enviar el formulario sin seleccionar ningún *checkbox* de la lista, aparecerá una ventana emergente que permitirá cancelar el envío del formulario o continuar con el mismo, como se muestra en la Ilustración 1.31. Una vez enviado el formulario, se mostrará en detalle el mensaje enviado y todas las opciones seleccionadas en la lista, como se observa en la Ilustración 1.32, además de un botón para poder ver las imágenes enviadas como adjunto en una nueva pantalla, como se muestra en la Ilustración 1.33.

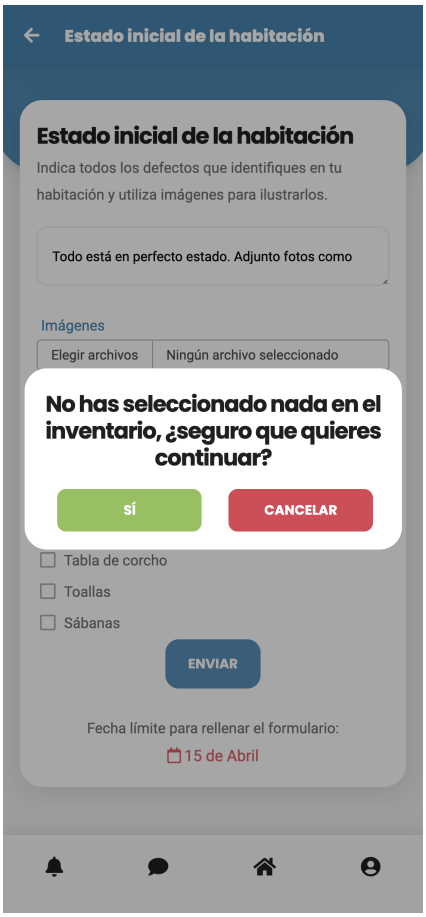

Ilustración 1.31: Plazo terminado para el estado inicial

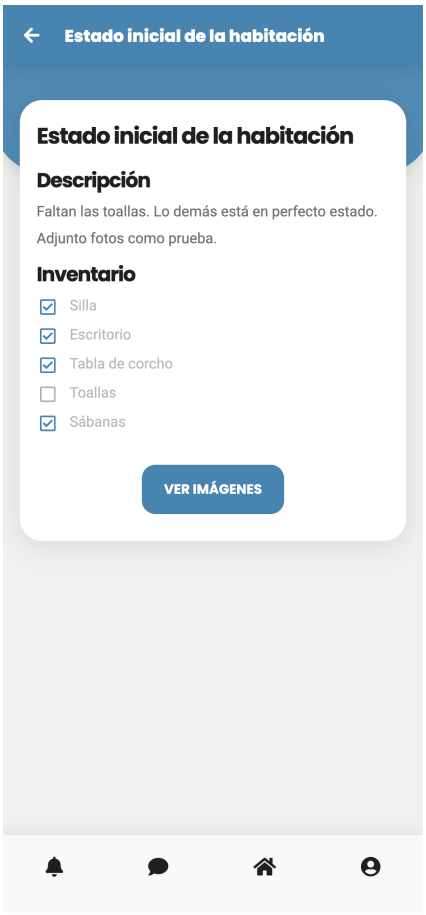

Ilustración 1.32: Estado inicial enviado

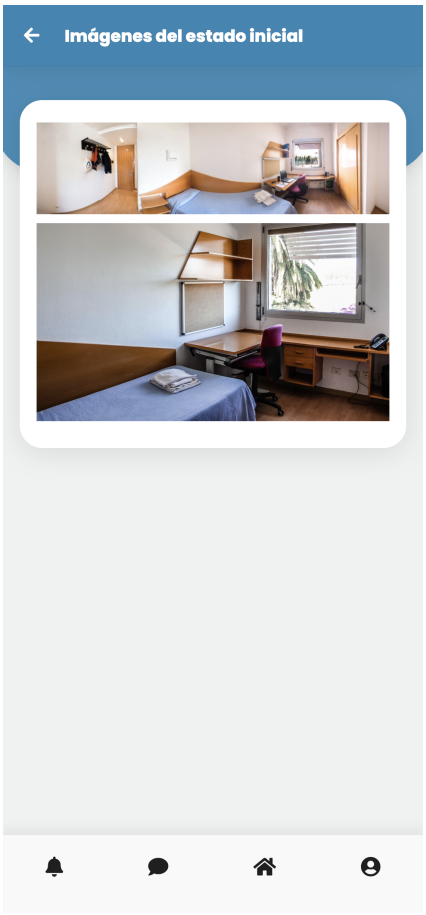

Ilustración 1.33: Imágenes enviadas en el estado inicial

C Manual de instalación

ANEXO III: MANUAL DE INSTALACIÓN

Para la aplicación PWA de Alloro es necesario tener instalados los siguientes componentes en el sistema:

- NPM (Node.js)
- Vue 3
- Vue CLI

En caso de no tener instalados estos componentes en el sistema, se instalará en primer lugar Node.js desde el siguiente enlace: https://nodejs.org/es/.

A continuación, se instalarán Vue 3 y Vue CLI usando el gestor de paquetes NPM instalado previamente a través de Node.js. Para ello, mediante el uso de una terminal, se usarán los siguientes comandos:

Para la instalación de Vue 3:

```
npm install vue@next
```

Para la instalación de Vue CLI:

```
npm install –g @vue/cli
```

Una vez instalados todos los componentes necesarios, se podrá ejecutar la aplicación haciendo uso del siguiente comando, ejecutándolo desde el directorio del proyecto:

```
npm run serve
```

Si todo ha funcionado correctamente, se estará ejecutando la aplicación PWA de Alloro en el explorador en *localhost*.

Para ejecutar el *backend* de la aplicación de Alloro, es necesario tener instalados los siguientes componentes en el sistema:

- Intérprete PHP 7
- Composer
- Laravel
- MAMP

Para ejecutar la API en un servidor local se ha hecho uso de la herramienta MAMP. Para instalar esta herramienta se accede al siguiente enlace: https://www.mamp.info/en/downloads/.

Una vez instalado MAMP, se instalará Composer, que gestionará las dependencias de paquetes en los proyectos que usen PHP. Para instalar esta herramienta se accederá al siguiente enlace: https://getcomposer.org/.

Una vez instalado composer, se instalará Laravel haciendo uso del siguiente comando:

```
composer global require laravel/installer
```

Para comprobar las distintas opciones para instalar Laravel se puede acceder al siguiente enlace: https://laravel.com/docs/8.x/installation#installation-via-composer.

Una vez instalados todos los componentes necesarios, se puede ejecutar la API de Alloro haciendo uso del siguiente comando desde el directorio del proyecto:

```
php artisan serve
```

Si todo ha funcionado correctamente, se habrá ejecutado la API de Alloro en localhost.

Bibliografía

[1] Iván Ramírez. ¿qué es una aplicación web progresiva o pwa? https://www.xataka.com/basics/que-es-una-aplicacion-web-progresiva-o-pwa, Jul 2018.

[2] 8 reasons why php is still so important for web development. https://www.jobsity.com/blog/8-reasons-why-php-is-still-so-important-for-web-development.

[3] Introducción - vue.js. https://v3.vuejs.org/guide/introduction.html.

[4] Hack Reactor. What is javascript used for? https://www.hackreactor.com/blog/what-is-javascript-used-for, Aug 2021.

[5] Installation. https://laravel.com/docs/8.x.

[6] Css: Cascading style sheets. https://developer.mozilla.org/en-US/docs/Web/CSS.

[7] Html. https://es.wikipedia.org/wiki/HTML, Oct 2021.

[8] Atlassian. Qué es git: Conviértete en todo un experto en git con esta guía. https://www.atlassian.com/es/git/tutorials/what-is-git.

[9] https://www.w3schools.com/nodejs/nodejs_npm.asp].

[10] Phpstorm: El ide rápido e inteligente para programación en php de jetbrains. https://www.jetbrains.com/es-es/phpstorm/.

[11] Webstorm: El ide más inteligente para javascript, creado por jetbrains. https://www.jetbrains.com/es-es/webstorm/.

[12] Phpmyadmin. https://es.wikipedia.org/wiki/PhpMyAdmin, Aug 2021.

[13] MAMP GmbH. Mamp - your local web development solution. https://www.mamp.info/en/mamp/windows/.

[14] Licencia de software. https://es.wikipedia.org/wiki/Licencia_de_software, Oct 2021.

[15] Toolbox subscription agreement for students and teachers. https://www.jetbrains.com/legal/docs/toolbox/license_educational/.

[16] Gnu general public license. https://es.wikipedia.org/wiki/GNU_General_Public_License, journal=Wikipedia, Oct 2021.

[17] Software gratis. https://es.wikipedia.org/wiki/Software_gratis, Sep 2021.

[18] Licencia php. https://es.wikipedia.org/wiki/Licencia_PHP, Sep 2019.

[19] Licencia mit. https://es.wikipedia.org/wiki/Licencia_MIT, Oct 2021.

[20] Metodologías de desarrollo de software: ¿qué son? https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html.

[21] Iterative model: What is it and when should you use it? https://airbrake.io/blog/sdlc/iterative-model, Jan 2021.

[22] Ana Pérez. Características y fases del modelo incremental. https://www.obsbusiness.school/blog/caracteristicas-y-fases-del-modelo-incremental.

[23] Lenguaje unificado de modelado. https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado, Oct 2021.

[24] Definición de casos de uso. https://www.ibm.com/docs/es/elm/6.0.3?topic=requirements-defining-use-cases.

[25] Programación por capas. https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas, Jun 2021.

[26] Eloquent: Getting started. https://laravel.com/docs/8.x/eloquent.

[27] Euphemia Wong. Shneiderman's eight golden rules will help you design better interfaces. https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces.

[28] Modelo–vista–modelo de vista. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93modelo_de_vista, Jul 2021.

[29] Adrián Alonso. ¿por qué vue.js está ganando tanta popularidad? https://adrianalonsodev.medium.com/por-qu%C3%A9-vue-js-est%C3%A1-ganando-tanta-popularidad-e44590cdd0fd, Jan 2018.

[30] Why you should use laravel: 8 benefits of laravel. https://www.fastfwd.com/why-choose-laravel-for-your-next-web-project/, Jul 2021.

[31] Directory structure. https://laravel.com/docs/8.x/structure#the-app-directory.

[32] Laravel sanctum. https://laravel.com/docs/8.x/sanctum.

[33] auth0.com. Json web tokens introduction. https://jwt.io/introduction.

[34] How heroku works: Heroku dev center. https://devcenter.heroku.com/articles/how-heroku-works.

[35] https://www.pwabuilder.com/.