Proyecto Web: SolucionesVecinales

Francisco Javier Sueza Rodríguez

3 de marzo de 2025

Centro: IES Aguadulce
Ciclo Formativo: Desarrollo Aplicaciones Web (Distancia)
Asignatura: Proyecto DAW

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Presentación

En este documento se detalla el proyecto final desarrollado para el **CFGS Desarrollo de Aplicaciones Web** en el **IES Aguadulce**. El proyecto elegido ha sido una aplicación web para la **gestión y administración de comunidades de vecinos**, llamada **SolucionesVecinales**.

El objetivo de esta aplicación es facilitar la **gestión de comunidades de vecinos** por parte tanto de los administradores de la propiedad como de los inquilinos o propietarios, ofreciéndose de **forma gratuita** y poniendo especial énfasis en las **accesibilidad** y **facilidad de uso**, con una interfaz web amigable y accesible desde cualquier dispositivo que disponga una navegador web.

La aplicación trabaja con **3 tipos diferentes de usuarios**, incluyendo administradores, inquilinos y visitantes, siendo el público objetivo los administradores de propiedad y presidentes de comunidad así como todos los inquilinos y propietarios de las comunidades de vecinos. Los diferentes usuarios tendrán diferentes permisos y responsabilidades.

A lo largo de este documento, desgranaremos todas los requisitos de la aplicación, el diseño elegido y todos los detalles del proceso de desarrollo y testeo.

1.2. Contexto

Esta aplicación es un **proyecto personal** que se ha decidido realizar tras el análisis de diferentes aplicaciones similares y ver determinadas carencias que pueden dificultar la inclusión de todos los potenciales usuarios. Hay que tener en cuenta, que los usuarios objetivos son un grupo muy heterogéneo donde podemos encontrar personas con diferente formación, conocimientos sobre tecnologías o con problemas de accesibilidad, así como gente con dificultades económicas.

Por este motivo, se ha decidido realizar una **aplicación gratuita**, centrándonos en la **accesibilidad** y **facilidad de uso**, que ayude a cualquier potencial usuario a usar la aplicación e implicarse en la gestión de su comunidad de vecinos.

1.3. Planteamiento del Problema

La gestión de comunidades de vecinos es una tarea compleja que requiere realizar un conjunto de gestiones administrativas asegurando la participación de los vecinos y de una forma transparente. En este aspecto, nuestra aplicación debería permitir que se realicen las siguientes tareas:

- **Gestión de documentos**: gestión de los documentos generados en los diferentes procesos, como resolución de incidencias, documentos sobre juntas vecinales, etc. Permitiendo que estos puedan ser accedidos por todos los vecinos de la comunidad.
- **Reserva de espacios comunes**: automatizar la reserva y gestión de espacios comunes permitiendo a los vecinos reservar o cancelar reservas de dichos espacios, así como acceder a una lista con los espacios y el estado actual de reserva.
- **Gestión de Incidencias**: se debe permitir la creación de incidencias por parte de los vecinos así como su administración por parte del presidente de la comunidad o el administrador de fincas.
- **Gestión de Usuarios y Comunidades**: la aplicación debe permitir la creación, baja, actualización y consulta de usuarios con diferentes roles así como de comunidades de vecinos.

- **Gestión de Finanzas**: la aplicación debe realizar, de forma automática, cálculos para administrar económicamente la comunidad, calculando el balance anual a partir de las cuotas que se están pagando y las diferentes facturas que se han pagado.

Además de estas tareas, hay que garantizar la seguridad de la información almacenada, ya que es información sensible, así como que el proceso para realizar estas tareas sea sencillo e intuitivo, además de accesible.

1.4. Visión General

Este documento se compone de 6 secciones principales que tratan diferentes aspecto del proyecto. Para facilitar al interesado la navegación por éste, vamos a explicar brevemente en que consiste cada sección del documento:

- **Análisis de Requisitos**: en esta sección se especifican los requisitos, tanto funcionales como no funcionales de la aplicación, especificando que es lo que se quiere conseguir con este proyecto y cuales son las restricciones que vienen impuestas. bien por el problema a tratar o por los propios requisitos.
- **Hardware y Software Necesario.**: en esta sección se especifica el hardware necesarios para realizar el desarrollo y poner en funcionamiento la aplicación así como el software que se va a necesitar para este mismo propósito. También se especifica el presupuesto y los costes de hosting para la aplicación.
- **Casos de Uso**: en esta sección se realiza un análisis de los principales casos de uso, empleando diferentes diagramas, así como una descripción más detalladas de los casos de uso generales.
- **Diseño de la Interfaz**: en esta sección se especifica el diseño de la interfaz de usuario, mostrando todas las páginas de la aplicación así como la relación entre estas y sus diferentes elementos.
- **Diseño de la Base de Datos**: en este apartado se especifica el diseño de la base de datos, empleando un esquema Entidad-Relación así como el paso a tablas de ésta, explicando los puntos que sean oportunos sobre las decisiones de diseño tomadas. Además, se incluyen dos script, uno para la creación de la base de datos y otro para poblarla con datos.
- **Diagrama de Componentes**: en esta sección se muestra el diagrama de componentes de la aplicación, poniendo de relieve la arquitectura de la aplicación así como las interacción entre los diferentes elementos y sistemas.

2. Análisis de Requisitos

2.1. Introducción

En esta sección se va definir la **especificación de requerimientos** que establece los requisitos funcionales y no funcionales de la aplicación SolucionesVecinales, realizando una análisis del funcionamiento esperado de aplicación y estableciendo unos límites para el diseño de ésta.

2.1.1. Propósito

El propósito de esta sección es **establecer los requisitos**, tanto funcionales como no funcionales, de la aplicación que se va a desarrollar, realizando una **análisis exhaustivo de su funcionamiento** y estableciendo las restricciones necesarias para que la aplicación cumpla su funcionalidad de forma correcta,

segura y eficiente. Estos requisitos nos ayudarán a planificar el proceso de desarrollo así como a tomar las decisiones oportunas durante el diseño de la aplicación.

Esta sección va dirigida principalmente al **equipo de desarrollo**, que serán los encargados de diseñar la aplicación a partir de los requerimientos y restricciones establecidos en este documento, aunque teniendo en cuenta que también esta incluido en el documento del proyecto en general cabe la posibilidad de que tengan acceso a él otros interesados, como por ejemplo, posibles inversores.

2.1.2. Alcance

Nuestro producto se llama **SolucionesVecinales** y es una aplicación web para la **gestión de comunidades de vecinos**. El fin de nuestra aplicación es ayudar a todos los integrantes en una comunidad de vecinos a realizar diferentes gestiones relacionadas con las administración de su comunidad, de forma simple y sencilla.

Es una aplicación que se **ofrece de forma gratuita**, cuya meta principal es llegar al mayor número de usuarios, implementando buenas **políticas de accesibilidad** y ofreciendo una **interfaz sencilla** e **intuitiva** que pueda usar cualquiera persona, incluso aquellos que no están familiarizados con las nuevas tecnologías.

Partiendo de esta base, la aplicación deberá **ayudar a realizar las tareas** más comunes en las que deben participar los vecinos en una comunidad, como el acceso de documentos, creación de incidencias, consulta de las cuentas de la comunidad, reserva de espacios comunes, etc.

2.1.3. Definiciones, Siglas y Abreviaturas

Las definiciones, siglas y abreviaturas se incluyen al final del documento, dentro de un Glosario, donde se incluyen todos los términos tanto de esta sección como del resto de secciones.

2.1.4. Referencias

La referencias, al igual que el glosario, tanto de esta sección como del resto de secciones se incluyen al final de documento, para no duplicar secciones. Además, en cada texto que haga referencia a alguna página o documento, se incluirá la cita adecuada apunta a dicha referencia, para facilitar su acceso.

2.1.5. Visión General

Esta sección se divide a su vez en 4 secciones, incluyendo la actual. Cada sección describe un aspecto diferentes de los requisitos del software y son las siguientes:

- **Introducción** (Sección 2.1): en esta primera sección se realiza una introducción tanto al software como a este documento, explicando su propósito, alcance, etc...
- **Descripción General** (Sección 2.2): en esta sección se realiza una descripción del software que se va a desarrollar, incluyendo su funcionalidad, restricciones, características de los usuarios potenciales, interfaces, etc. Nos sirve como base para especificar los requisitos en el siguiente apartado.
- **Requerimientos Específicos** (Sección 2.3): en la tercera sección vamos a establecer los requisitos específicos del software, extraídos de la descripción general y separados en requisitos funcionales y no funcionales.

- **Requerimientos de Documentación** (Sección 2.4): en esta sección de la parte de requisitos se especifican los requisitos de documentación que va a tener el software, incluyendo manuales, ayuda en línea, instalación, etc.

2.2. Descripción General

En esta sección se van a describir todos los factores y restricciones que afectan al producto, estableciendo una base para la definición de los requisitos específicos en la siguiente sección, incluyendo una perspectiva general del producto, así como las restricciones que se deben aplicar en su diseño.

2.2.1. Perspectiva del Producto

En primer lugar vamos a realizar una descripción del producto, especialmente de las diferentes interfaces que se van a emplear y su interacción con los diferentes elementos y sistemas con los que deberá interactuar nuestro software.

2.2.1.1. Interfaz de Usuario

Debido a que las especificaciones sobre al interfaz de usuario son más extensas, se han incluido como un anexo. Estas especificaciones se pueden consultar, en concreto, en el Anexo A.

2.2.1.2. Interfaces con Hardware

Nuestro software **no interactúa directamente con el hardware** de los dispositivos, ya que en la parte del cliente la interacción se realiza con el navegador web, mientras que en la parte del servidor la interacción se realiza con el servidor web. Estos, a su vez, realizarán las interacciones necesarias con el SO.

Sí cabe destacar en esta sección, ya que no veo otra donde pueda incluir esta información, que **nuestro software** se deberá usar desde **cualquier dispositivo** que cuente con un **navegador web**, por lo que deberá realizarse el diseño teniendo esto en cuenta. Realmente esto solo afecta a la estética de la aplicación en cada dispositivo, y no en su funcionamiento, y tampoco requiere del empleo de ninguna interfaz específica, por lo que solo afectará a la **interfaz de usuario**.

2.2.1.3. Interfaces con Software

Nuestra aplicación **no usa ninguna interfaz externa** para comunicarse con ningún software externo.

La **única interacción** que cabe mencionar es la que realiza la parte de cliente de nuestra aplicación con el **navegador web** del usuario, aunque en este aspecto no se emplea ninguna interfaz específica para llevar a cabo esta interacción, si no que se realiza a través de los lenguajes empleados para desarrollar el propio cliente, como HTML, CSS, Javascript, etc.

En el **lado del servidor**, nuestra aplicación se ejecutará en un servidor web, aunque como hemos comentado, no se va a emplear ninguna interfaz para comunicarse con el. Para ejecutar nuestra aplicación vale cualquier servidor web, aunque si hay que tener en cuenta un detalle, y es que cuando la aplicación se este **ejecutando en producción**, el servidor deberá tener soporte para SSL, ya que nuestra aplicación va a emplear el protocolo **HTTPS** para comunicarse.

2.2.1.4. Interfaces de Comunicación

Para que nuestra aplicación funciones correctamente **necesita comunicarse mediante una red**, ya sea local o internet, realizando al comunicación entre nuestro cliente, con el navegador del usuario, y nuestro servidor. Para llevar a cabo esta comunicación, se va a emplear el protocolo HTTP.

El **protocolo HTTP** es el estándar para la comunicación web desde hace muchos años, inventado por Tim Berners-Lee en 1989, es un protocolo de la capa de aplicación que es la base de la comunicación en la World Wide Web. [1]

En este documento **no vamos a discutir** ni exponer las **especificaciones** de todo el **protocolo HTTP**, ya que el uso de este protocolo se va a realizar mediante funciones predefinidas por los lenguajes sin tener que preocuparnos de como se implementa dicho protocolo, aunque si vamos a comentar un par de puntos que si conviene tener en cuenta.

En primer lugar, hay que hablar de los **métodos HTTP**, que son un conjunto de "verbos" que se usan para solicitar, desde el cliente, diferentes tipos de acciones que debe realizar el servidor, como puede ser mandar información, modificarla, eliminarla, etc... Los verbos HTTP que podemos emplear son GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE y PATCH. [2]

Cada método tiene una funcionalidad diferente, con diferentes ventajas e inconvenientes, pero nosotros no vamos a describir aquí todos, si no que nos vamos a centrar en los únicos que nos interesan y que vamos a usar en nuestra aplicación, que son **GET**, **POST**, **PUT** y **DELETE**. Las principales características de estos métodos son las siguientes:

- **GET**: este método se emplea para realizar **peticiones al servidor** sobre el **estado representacional de algún recurso**. Las peticiones con este método se pueden almacenar en una caché y la información de la petición se pueden ver en la URL que se le pasa al servidor. Además es un método seguro, ya que no altera información del servidor e idempotente. [3]
- **POST**: este método se emplea para **enviar información al servidor**, la cual se envía en el **cuerpo de la petición**. El resultado puede ser alguna acción realizada por el servidor como comprobar que los datos son correctos, crear nuevos datos o modificarlos. Es un método no seguro, ya que modifica información en el servidor, y tampoco es cacheable ni idempotente. [4]
- **DELETE**: con éste método se le indica al servidor que elimine un recurso específico. Este método no es seguro ni cacheable, pero si es idempotente. [5]
- **PUT**: este método crea o modifica un recurso en el servidor añadiendo el contenido en el cuerpo de la solicitud. La principal diferencia entre PUT y POST es que PUT es idempotente. [6]

Estos métodos tienen que tenerse claros en el proceso de desarrollo, ya que la API de nuestra aplicación deberá manejar correctamente las diferentes peticiones que se hagan con estos métodos, y nuestro cliente deberá realizar las peticiones con el método adecuado.

Cabe destacar que nuestra aplicación usara **HTTPS**, aunque esto no va a afectar en como van a tratarse los métodos ni a procesarse, sino en como se realizará la conexión, que tiene más que ver con la configuración del servidor.

Por otro lado, debemos hablar de los **códigos HTTP de respuesta**. Estos códigos son los que deberá responder el servidor a las diferentes peticiones que realice el cliente. No vamos a explicar cuales son todos los códigos, ya que sería muy extenso, simplemente comentar que en todo momento se deberá enviar el código adecuado al resultado de la operación solicitada por el cliente y con mensajes descriptivos que expliquen bien el motivo de ese error.

Para consultar con más detalle los diferentes códigos que podemos encontrar podemos visitar la web de MDN sobre dichos códigos, donde hay una lista extensiva con todos y cada uno de los códigos empleados.

2.2.1.5. Restricciones de Memoria

La **restricciones de memoria** en nuestra aplicación son muy dispares, ya que las que se aplican al servidor no son las mismas que las que se aplican al cliente, por lo que vamos a tratarlas las dos de forma individual.

En nuestro **servidor** las restricciones tanto en memoria primaria como secundaria son mínimas. La cantidad que se use de ambas va a estar estrechamente **relacionada con el número de usuarios** que tenga nuestra aplicación, ya que el principal uso de memoria principal va ser ocupado por las peticiones de los usuarios y las acciones que tenga que llevar, en consecuencia, el servidor sobre el sistema. Por otro lado, nuestra aplicación debe almacenar una gran cantidad de datos por cada comunidad, incluyendo archivos multimedia como es el caso de imágenes, por lo que la cantidad almacenamiento secundario que necesitaremos también va a depender en gran medido del número de usuarios que tengamos.

Por otro lado, nuestro **cliente** si va a tener **fuertes restricciones**, especialmente en el uso de **memoria principal**, ya que no se espera uso de memoria secundaria más de lo que puedan ocupar las cookies y otros elementos de este tipo. Pero en la memoria principal si tendremos restricciones, ya que hay que tener en cuenta que nuestra aplicación debe ejecutarse en muchos dispositivos que tienen recursos muy limitados, como pueden ser los dispositivos móviles, especialmente los más antiguos. En este aspecto **deberá minimizarse** el uso de **memoria principal**, evitando efectos indeseados como excesivos renderizados o utilizando imágenes comprimidas.

2.2.1.6. Requerimientos de Adecuación al Entorno

La aplicación **no tiene requerimientos especiales** para su ejecución en diferentes entornos, ya que esta se va a ejecutar en cualquier servidor web que tenga soporte para SSL. Respecto a la **visualización en diferentes tipos de dispositivos**, en la **sección de restricciones de diseño** se añadirán detalles sobre como afrontar esta situación.

2.2.2. Funciones del Producto

A lo largo de este documento ya hemos comentado que nuestro software, **SolucionesVecinales**, es una aplicación de **gestión de comunidades de vecinos**, pero no hemos especificado cual es su funcionamiento, así que en esta sección vamos a realizar una descripción general del funcionamiento de la aplicación.

- La aplicación deberá **gestionar los diferentes usuarios** permitiendo que éstos se registren en el sistema como inquilinos o administradores y que accedan a las diferentes herramientas para gestionar su comunidad, así como a la modificación de su perfil o su eliminación.
- Los **administradores** podrán añadir nuevas comunidades, donde los usuarios podrán inscribirse. El **sistema generará automáticamente** una **solicitud de inscripción**, cuando se registre un inquilino, a la comunidad con la misma dirección del inquilino. El **administrador deberá aprobar** la solicitud para que el inquilino quede inscrito.

- Una vez que los **inquilinos o administradores** estén inscritos en una comunidad, tendrán acceso al sistema de gestión de comunidades, donde podrán realizar diferentes acciones en su comunidad, además de tener la opción de editar su perfil o eliminar su usuario desde su página de perfil.
- Dentro de las **herramientas** que provee nuestro software para **gestionar la comunidad**, tenemos las siguientes:
    - **Inicio**: en la pantalla de inicio de gestión de comunidad se mostrará información de la comunidad, así como información relativa a los elementos creados por el inquilino, como pueden ser incidencias, documentos descargados, reservas de espacios comunes, etc.
    - **Tablón de Anuncios**: en el tablón de anuncios el administrador podrá **crear mensajes** que van dirigidos a los miembros de la comunidad. Los mensajes se mostrarán como una **lista** con la fecha de creación debajo y todos los usuarios podrán consultarlos desde su tablón de anuncios. Solo el **administrador** podrá **crear** o **eliminar** mensajes.
    - **Documentos**: en esta sección el **administrador** subirá los diferentes **documentos** relativos a la **gestión de su comunidad**, como pueden ser resoluciones judiciales, legislación, actas de reuniones. El resto de usuarios podrán visualizar y descargar dichos archivos. Los archivos se **mostrarán como un conjunto de iconos** con el nombre debajo y el tamaño. Solo el **administrador** puede subir o eliminar archivos de esta sección.
    - **Incidencias**: esta sección permitirá a todos los inquilinos la **creación de incidencias**, para ponerlas en conocimiento de los administradores y que tomen las medidas oportunas para su resolución. Cuando un inquilino ingrese en esta página podrá **ver** todas las **incidencias creadas por el mismo** o por otros miembros de la comunidad que hayan **marcado la incidencia como pública**.

      Las incidencias pueden estar en tres estados: “Creada”, “En Proceso” o “Cerrada”, según el momento de procesamiento en el que se encuentre ésta. Las incidencias se mostrarán en una tabla, con su nombre, descripción, fecha de creación y estado. El **administrador** podrá **cambiar el estado** de las incidencias, además de eliminarlas.
    - **Zonas Comunes**: en esta sección los inquilinos podrán **reservar los espacios comunes** que sean objeto de reserva en la comunidad. Los espacios comunes serán **añadidos por el administrador**, que será también el que establezca el horario en el que se pueden reservas.

      Los **inquilinos**, podrán consultar el estado de reserva de un espacio común y realizar una reserva en el horario y día que este disponible. Cuando un inquilino tenga reservas realizadas, aparecerán debajo de los espacios comunes, permitiéndole borrar o cambiarlas.
    - **Finanzas**: la sección de finanzas **mostrará una tabla con todos los ingresos y gastos** previstos para este año, aunque también se podrán consultar los del ejercicio anterior. El administrador podrá añadir o eliminar registros, lo que provocará que la aplicación haga el cálculo de nuevo, mostrando el nuevo resultado.
- Además de estos servicios, que serán ofrecidos a los usuarios, la aplicación tiene un **backoffice**, donde el administrador web (**WebAdmin**) podrá conectarse y gestionar los diferentes servicios que tiene la aplicación, así como consultar estadísticas con diferentes parámetros.

Estás son las principales funcionalidades que tiene nuestro software. **No es algo inamovible**, y en un futuro podría (y debería) plantearse la adición de nuevas funcionalidades. Para ello será interesante crear una canal de feedback con los usuarios y escuchar que tienen que decir, especialmente, que echan en falta.

2.2.3. Características de los Usuarios

Los usuarios objetivos de nuestra aplicación son un **grupo muy heterogéneo**, que abarca a **cualquier persona** que viva en una comunidad de vecinos, es decir, la gran mayoría de la población. Por ello vamos a encontrarnos usuarios con todos los niveles de experiencia, nivel educacional, edad, etc.

Teniendo esto en cuenta, la aplicación debe ser lo suficientemente simple e intuitiva en su uso para que los usuarios con menos experiencia, nivel educativo o los que tiene problemas de accesibilidad puedan usarla correctamente.

2.2.4. Restricciones de Diseño

En este apartado vamos a realizar una lista con todas las restricciones de diseño que deben aplicarse a la hora de desarrollar la aplicación, desde lenguajes de programación y herramientas de desarrollo, pasando por políticas de testeo, seguridad de datos, etc...

2.2.4.1. Lenguajes de Programación

El lenguaje de programación principal empleado, tanto en el frontend (cliente) como en el backend, será **Typecript**, ya que proporciona toda la funcionalidad de Javascript con elementos interesantes como la definición de tipos o interfaces. En el **backend** se usará, además, SQL para trabajar con la base de datos.

2.2.4.2. Software para el Desarrollo

Para el desarrollo de la aplicación se usarán diferentes librerías, **entornos de ejecución**, etc. En nuestro caso, se va a especificar el software que es de obligatorio uso para el desarrollo de nuestra aplicación.

- **FrontEnd**: en esta parte se va a emplear **React.js** para desarrollar el cliente empleando componentes. Además se debe emplear el preprocesador de CSS **SASS**, para crear archivos CSS modulares y reutilizables.

  Después de una análisis más profundo de la aplicación **se ha descartado el uso de Redux**, ya que con **Context** de React se podrá realizar la misma funcionalidad disminuyendo la complejidad.

- **BackEnd**: en la parte de backend se va a emplear **Node**, para poder ejecutar Typescript en el lado del servidor junto con **Express**, que nos facilitara la creación de la API. Además, se empleará para la base de datos **Postgresql**, junto con **Prisma**, un ORM que facilitará el trabajo con la base de datos.

- **Herramientas de Desarrollo**: cada programador podrá usar el editor o IDE que sea de su preferencia, pero si se deberá usar **Prettier**, para establecer una estándar en el estilo código y **Eslint**, para detectar errores y adherirse a buenas prácticas.

- **Gestión del Proyecto**: para la gestión del proyecto se van a emplear **Github**, como repositorio de software y **Trello**, que se empleará para la gestión de las tareas que hay que realizar a modo de tablero Kanban.

No se va a especificar el software para el testing ni el control de calidad, ya que este se especificará en su respectiva restricción.

2.2.4.3. Control de Calidad y Testeo

Se tiene que implementar una **política de control de calidad** que incluya el testeo continuo de la aplicación a varios niveles, para segurar su calidad y localizar errores de forma más rápida. Esta política consiste en los siguientes puntos:

- El desarrollo de cada módulo de software se realizará empleando la metodología **TDD**, **implementando primeramente los test** con la funcionalidad básica y desarrollando posteriormente el código que cumpla con dicha funcionalidad. Posteriormente se **refactorizará** teniendo en cuenta otros aspectos como la performance, la eficiencia, seguridad, etc..
- Para realizar los test unitarios se empleará **react-testing-library** en la parte el cliente y **Jest + Superset** para la parte del servidor.
- Se realizarán un proceso de **integración continua** empleando para ello **Github Actions**, siendo prioridad que un trozo de software que se ha desarrollado pase todos los test y haya una **cobertura del 100 %** del código con los test creados.
- Se integrará **Cypress** en este proceso integración continua para la realización de los e2e.
- El código de la aplicación estará incluida en **Code Climate**, donde se deberá revisar periódicamente y realizar los cambios indicados por este software para tener un código de calidad. Esto será complementario, como se ha comentado en otro punto, con el uso de **linters** como **Eslint**.
- Se seguirán las **buenas prácticas** tanto en el desarrollo del cliente como del servidor, siguiendo los estándares y recomendaciones de la **W3C**.

2.2.4.4. Seguridad de la Aplicación

A la hora de tratar con los datos de los usuarios y las comunidades hay que ser extra precavido, y tener en cuenta unas restricciones para garantizar la seguridad de dichos datos. Estas restricciones son:

- Los **datos que se obtenga de formularios** deben ser **doblemente validados**. Por un lado, se procesarán en el cliente, comprobando que tienen el formato correcto y eliminando aquellos que tengan ciertas palabras que pueden considerarse maliciosas, especialmente para ataques de **inyección SQL**. En una segunda parte se **volverán a validar en el servidor** antes de incluirlos en la base de datos.
- **Nunca** se **almacenarán las contraseñas** en **texto plano** en la base de datos. Para su almacenamiento, se empleará alguna función de hasing´ como **bcrypt**, almacenando el resultado en vez de la contraseña.
- Las **contraseña de los usuarios** deberán tener un **longitud de 12 caracteres o más**, e incluir algún **número** y **signo de puntuación**.
- Nuestra base de datos, que estará físicamente en el mismo servidor que el servidor web, **solo debe aceptar consultar locales**, para evitar determinados tipos de ataques.
- Las **peticiones por segundo** al servidor **estarán limitadas**, para evitar ataques de tipo DOS.
- El **software** que se emplee en el desarrollo y necesario para la ejecución de la aplicación deben estar siempre **actualizado a la última última versión**.
- Solo se permitirán conexiones **HTTPS** con la aplicación.

- Los usuarios deberán **ingresar al sistem**a empleando su **correo y contraseña**, salvo el **administrador web**, que podrá usar el **usuario WebAdmin** y su contraseña.

2.2.4.5. Ley de Protección de Datos

A la hora de implementar nuestra aplicación hay que tener en cuenta la **leyes de protección de datos** aplicables. En nuestro caso, son la **LOPD** (Ley Organica de Protección de Datos) y la **RGPD** (Reglamente General de Protección de Datos) de la Unión Europea. En este aspecto, y teniendo en cuenta los datos que trata nuestra web, debemos incluir en esta [7]:

- **Política de Privacidad**: texto legal para informar a los interesados toda la información sobre el tratamiento de sus datos.
- **Política de Cookies**: texto legal donde se informa de la política de cookies a los usuarios, donde además se pedirá su consentimiento para el almacenamiento de cookies en el sistema.
- **Cesión de Datos a Terceros**: si se decide la utilización de software de terceros (alto que aún no esta decidido) para obtener métricas del acceso a la web, como puede ser Google Analytics, se deberá firmar un acuerdo en que se establecerán las obligaciones para el encargado.

En principio estos son los puntos más importantes que se deben cumplir de las **leyes de protección de datos**, aunque deberemos estar pendientes si se añaden nuevas funcionalidades, o anuncios, para aplicar nuevas medidas.

2.2.4.6. Accesibilidad

Uno de los puntos más importantes de nuestra aplicación es la **accesibilidad** y **facilidad de uso**, por lo que en este aspecto se deben aplicar ciertas restricciones para garantizar que todo el mundo puede emplear nuestra aplicación con eficiencia.

En este punto, vamos a listar alguno de los **puntos más importantes**, pero no vamos a hacer una descripción exhaustiva de todos los puntos implicados ya que el documento sería muy extenso. Los principales elementos a tener en cuenta son [8]:

- Todo **elemento** que **no sea de texto** deberá tener una **alternativa textual**. Por ejemplo, empleando el atributo **alt** en imágenes.
- Deberá emplearse un **contraste** mínimo en texto/fondo de **4.5:1**, para textos pequeños y de **3:1** para textos grandes, aunque es muy deseable que estos valores sean de **7:1** y **4.5:1** respectivamente.
- Se deberá proporcionar un conjunto de enlaces de tipo **breadcrumbs** para que la web sea más navegable.
- La aplicación debe ser totalmente **navegable** usando el teclado.
- Se deberán usar **etiquetas descriptivas** en todos los elementos de la interfaz.
- Se deberá emplear de forma eficiente los **elementos HTML semánticos** para estructurar correctamente la web y el contenido.
- El **tamaño de las fuentes** debe de poder cambiarse de forma rápida y que la web siga manteniendo su estructura visual.
- Se deberán **usar colores** que **no generen problemas** para usuarios **daltónicos**.

- Los **mensajes de error** generados en cualquier parte de la aplicación deben ser descriptivos

Estos son algunas de las restricciones que deberán aplicarse, pero los desarrolladores deberán aplicar más, según vaya surgiendo la necesidad usando la guía para la accesibilidad de contenido WCAG 2.2, proporcionada por la W3C.

2.2.5. Supuestos y Dependencias

Aunque nuestro proyecto no tiene muchas dependencias o supuestos, si hay que tener en cuenta los cambios que pueda haber en alguna legislación o buenas prácticas que podrían cambiar ligeramente nuestras restricciones de diseño. En aspecto, deberemos estar atentos a:

- **Cambios** en la **legislación sobre protección de datos**, ya que algunos cambios en esta legislación supondría la necesidad de añadir (o eliminar) algunas restricciones.
- **Cambios** en las **guías de accesibilidad** de la W3C. Ahora mismo esta en desarrollo la **versión 3 de la WCAG**, que podría añadir algunos nuevas prácticas para mejorar la accesibilidad en sitios web.

2.3. Requerimientos Específicos

En esta sección vamos, por fin, a definir los **requisitos funcionales** y **no funcionales** de nuestra aplicación. Esto requisitos se han extraído de la sección anterior, teniendo en cuenta la descripción de las interfaces, el funcionamiento de nuestra aplicación, etc..

Estos requisitos **especifican las características principales** de nuestro software, pero a pesar de ello, puede que se nos hayan pasado cosas por alto en el análisis, o que **durante el desarrollo** nos encontremos con problemas inesperados. Por ello, es de esperar que durante el proceso de desarrollo puedan **surgir nuevos requisitos** o se deban **modificar algunos** de los que ya hemos definido.

Adicionalmente a esta sección, se ha incluido una tabla con todos los requisitos funcionales y no funcionales en el Anexo B, para un acceso más rápido a los datos.

2.3.1. Requisitos Funcionales

En primer lugar, vamos a establecer los **requisitos funcionales**. Se va a realizar en formato de lista especificando el código **RFX**, donde X es el número del requisito, su **nombre** y una descripción del requisito en forma de lista.

- **RF1 - Consultar Web Principal**: Cualquier usuario podrá acceder a la página web principal para consultar la información sobre nuestra aplicación.
- **RF2 - Contactar Equipo**: Cualquier usuario podrá contactar con el equipo de desarrollo usando el formulario de la página principal, dejando un mensaje y su correo.
- **RF3 - Registrarse en la Aplicación**: Los invitados podrán registrarse en nuestra aplicación empleando el formulario de la página de registro e introduciendo su correo, nombre, apellidos, nombre de usuario y dirección.
- **RF4 - Generar Solicitud**: El sistema deberá generar una solicitud de inscripción en la comunidad que coincida con la dirección que ha especificado el usuario que se ha registrado.
- **RF5 - Aprobar Solicitud**: El administrador de la comunidad deberá poder aceptar o denegar la solicitud de inscripción en la comunidad de un inquilino.

- **RF6 - Iniciar Sesión**: Un inquilino o administrador podrá ingresar al sistema empleando su correo y contraseña como credenciales.
- **RF7 - Añadir Comunidad**: El administrador podrá añadir una nueva comunidad de vecinos, independientemente de que este inscrito ya en alguna otra, especificando un nombre, descripción, imagen y dirección de la comunidad.
- **RF8 - Modificar Perfil**: Un inquilino o administrador podrá acceder a su perfil y modificar sus datos en el sistema.
- **RF9 - Eliminar Usuario**: Un inquilino podrá eliminar su usuario desde su página de perfil, pudiendo ser también eliminado por el administrador.
- **RF10 - Eliminar Administrador**: El administrador podrá eliminar su información desde su página de perfil, siempre y cuando no este adscrito a ninguna comunidad.
- **RF11 - Eliminar Comunidad**: El administrador podrá eliminar una comunidad de vecinos, que haya creado, desde la sección para administrar la comunidad.
- **RF12 - Cargar Comunidad**: Cuando un inquilino o administrador inicie sesión, se cargará la página principal de la comunidad mostrando los datos de ésta, así como las notificaciones del inquilino.
- **RF13 - Notificar Inquilino**: El sistema creará notificaciones con los documentos descargados en los últimos 10 días, las reservas de espacios comunes y las incidencias de un inquilino, mostrándolos en la página principal de la comunidad.
- **RF14 - Consultar Tablón**: Un inquilino o administrador podrá consultar los mensajes creados en el tablón de anuncios.
- **RF15 - Gestionar Tablón**: El administrador podrá crear o eliminar mensajes del tablón de anuncios de la comunidad.
- **RF16 - Consultar Documentos**: Un inquilino podrá consular los documentos relativos a la comunidad y descargarlos a su dispositivo.
- **RF17 - Gestionar Documentos**: El administrador podrá añadir o eliminar documentos de una comunidad.
- **RF18 - Consultar Incidencias**: Un inquilino podrá consultar las incidencias publicas o creadas por el mismo, mientras que una administrador podrá consultar todas las incidencias relativas a una comunidad.
- **RF19 - Gestionar Incidencias**: Un inquilino podrá crear incidencias, así como borrar o modificar las incidencias creadas por el mismo, mientras que el administrador podrá eliminar las incidencias de cualquier inquilino.
- **RF20 - Modificar Estado de Incidencia**: El administrador podrá cambiar el estado de las incidencias entre: “Creada”, “En Proceso” y “Resuelta”.
- **RF21 - Añadir Espacio Común**: El administrador podrá añadir espacios comunes especificando su nombre, descripción y horario de reserva.
- **RF22 - Reservar Espacio Común**: Un inquilino podrá reservar un espacio común seleccionando en un calendario el día y hora que quiere realizar la reserva, siempre y cuando este libre.
- **RF23 - Cancelar Reserva**: Un inquilino podrá cancelar una reserva de un espacio común desde la sección de espacios comunes o desde la página principal de la comunidad.

- **RF24 - Gestionar Espacios Comunes**: El administrador podrá eliminar o modificar la información relativa a un espacio común.
- **RF25 - Consultar Finanzas**: Un inquilino podrá consultar las finanzas de la comunidad en la sección finanzas que se mostrarán en una tabla con los datos estimados para el curso actual, pudiendo también consultar los datos del curso anterior.
- **RF26 - Gestionar Finanzas**: El administrador podrá añadir, eliminar o modificar los registros de la tabla de finanzas.
- **RF27 - Calcular Finanzas**: El sistema realiza el calculo de las finanzas cada vez que se añada o elimine un pago o un ingreso.
- **RF28 - Cerrar Sesión**: Un inquilino o administrador podrán cerrar la sesión en el sistema desde el icono de usuario.
- **RF29 - Administrar Servicios**: el administrador web podrá iniciar sesión empleando el usuario WebAdmin y acceder al panel de control para administrar los diferentes servicios web así como para obtener datos de uso de la aplicación.

2.3.2. Requisitos No Funcionales

En esta sección se incluyen todos los requisitos no funcionales, referentes a la interfaz, seguridad, accesibilidad, etc. Al igual que con la sección anterior, se incluye una tabla con estos requisitos en el Anexo B.

- **RNF1 - Arquitectura**: La aplicación se implementará empleando una arquitectura cliente-servidor.
- **RNF2 - Protocolo Comunicación**: La aplicación empleará el protocolo HTTPS entre el cliente y el servidor, no permitiendo conexión con otro protocolo que no sea este.
- **RNF3 - Peticiones Limitadas**: Las peticiones por segundo al servidor estarán limitadas a 10, para evitar ataques de tipo DOS.
- **RNF4 - Almacenado de Contraseñas**: El sistema se encargará de almacenar las contraseñas en la base de datos de forma segura, nunca en texto plano, empleando un algoritmo hash y almacenando el resultado encriptado en la base de datos.
- **RNF5 - Robustez Contraseñas**: El sistema solo aceptará contraseña con un mínimo de 12 caracteres y con al menos un número y un signo de puntuación.
- **RNF6 - Validación de Datos**: Los datos introducidos por los usuarios serán validados tanto en el cliente como en el servidor.
- **RNF7 - Tiempo de Carga**: La aplicación deberá tener un tiempo de carga en el navegador del usuario inferior a 3 segundos.
- **RNF8 - Compresión Archivo Multimedia**: Los archivos multimedia empleados en la web deberán estar comprimidos lo máximo posible sin que estos pierdan calidad significativamente.
- **RNF9 - Uso de Memoria**: El cliente de nuestra aplicación deberá usar el mínimo de memoria posible para cumplir su función de forma eficiente.
- **RNF10 - Interfaz Adaptable**: El cliente de nuestra aplicación deberá tener una interfaz que se adapte a diferentes dispositivos, incluyendo dispositivos móviles y tablets.

- **RNF11 - Alternativa Textual**: Cualquier elemento de la interfaz que no sea texto deberá tener una alternativa textual.
- **RNF12 - Contraste**: La interfaz tiene que tener un contraste mínimo entre el texto y el fondo de 4.5:1 para textos pequeños y 3:1 para textos grandes.
- **RNF13 - Navegabilidad**: La interfaz deberá ser totalmente navegable con el teclado.
- **RNF14 - Mensajes de Error Descriptivos**: Cualquier mensaje de error que produzca la aplicación, ya sea para los usuarios o desarrolladores, debe explicar correctamente por qué se ha producido el error.
- **RNF15 - Test Unitarios**: Todo componente, función o clase deberá tener implementando su correspondiente test unitario.
- **RNF16 - Cobertura de Test**: Los test deberán cubrir el 100 % del código desarrollado.
- **RNF17 - CD/CI**: Los test unitarios, así como los e2e y los de integración deberán ejecutarse con un sistema de integración/despliegue continuo.
- **RNF18 - Calidad Global del Código**: La calidad global del código debe garantizarse, empleando para ellos herramientas como Code Climate, eslint y todas aquellas que sean necesarias para que el código se adapte a las buenas prácticas y estándares.
- **RNF19 - Formato del Código**: El código se formateará empleando prettier, con una configuración predefinida para que todo el código, de cualquier desarrollador, tenga el mismo formato.
- **RNF20 - LOPD y RGPD**: En cumplimiento de las leyes LODP y RGPD, deberán mostrarse a los usuarios los documentos legales siguientes: Política de Privacidad y Política de Cookies.

2.4. Requerimientos de Documentación

En esta sección se va a describir la documentación que va a emplear nuestro software, en que formato debe presentarse y el contenido que debe tener.

2.4.1. Manual de Usuario

En nuestro caso, no se necesita una manual de usuario para nuestra aplicación, ya que los usuarios van a acceder a ella de forma online y se va a emplear ayuda online.

2.4.2. Ayuda en Línea

Nuestra aplicación ofrecerá una **extensa ayuda en línea** que se encargará de explicar como realizar los diferentes procedimientos que pueden llevar a cabo los usuarios en nuestra aplicación.

En cada página, aparecerá un **icono con el signo de interrogación**, cerca de las opciones de menú y los elementos interactivos, como formularios y espacios donde se visualiza información sobre la comunidad. Al pulsar este botón, se **desplegará una ventana popup** con la ayuda en línea.

En primer lugar, la ventana deberá contener una **captura de pantalla de la interfaz** donde se resalte y expliquen los elementos que la componen y su uso. Además, en el caso de que se tengan que realizar más acciones, como por ejemplo, en el momento de crear una incidencia o realizar una reserva, **debajo de la explicación de la interfaz** deberá explicarse el procedimiento, paso a paso y acompañado con imágenes de todas las pantallas por las que vamos a pasar y los diferentes resultados que podemos obtener.

El posicionamiento del icono de ayuda deberá realizarse de **forma visible** e **intuitiva**, añadiendo además textos alternativo al elemento así como un mensaje que se mostrará al pasar el botón por encima indicando que es al ayuda de la página.

2.4.3. Guía de Instalación, Configuración y Archivo Léeme

Se incluirá una **guía de instalación en un entorno local** en el que se especificará como se instala el software en un servidor Apache 2.0 y la configuración que se debe realizar, si es necesario. La guía lo explicará paso a paso

Respecto al archivo Léeme, se incluirá un archivo **README** en formado Markdown en el **repositorio de Github** de nuestra aplicación, en el que se explicará un poco el propósito de nuestro software, como ejecutarlo de forma local, tecnologías empleadas, forma de contribuir al proyecto, etc. Este archivo, probablemente, sea más interesante para otros desarrolladores que para la mayoría de usuarios de nuestra aplicación.

2.4.4. Licencia de la Aplicación

Esta última sección se ha sustituido por la de la licencia, ya que no hay elementos de empaquetado o etiquetado en nuestra aplicación al margen de al licencia.

En este aspecto, nuestra aplicación usara una **Licencia MIT**, que permite la obtención de copias y su uso sin restricción y sin límites en el derecho de uso. [9]

En el repositorio se podrá acceder a una **copia integra de esta licencia**, así como desde el píe de página de nuestra aplicación. En este enlace se puede acceder al texto completo en inglés de la licencia.

2.5. Planificación del Proyecto

En esta sección vamos a realizar una pequeña planificación del proyecto, explicando de forma escueta su alcance y coste, e incluyendo una diagrama de Gantt con la temporalización del proyecto.

2.5.1. Alcance

El proyecto que vamos a desarrollar se llama **SolucionesVecinales**, y consiste en una aplicación web para la gestión de comunidades de vecinos gratuita y fácil de usar.

Nuestro **objetivo**, es crear un **software** que proporcione las herramientas para **gestionar los principales aspectos** de una **comunidad de vecinos**. El software ofrecerá herramientas para la gestión de incidencias, reserva de espacios comunes, gestión de documentos, tablón de anuncios y consulta de la finanzas de la comunidad. Todo eso, poniendo especial énfasis en la facilidad de uso y la accesibilidad.

El **plazo de entrega** de entrega del proyecto es, como máximo, del **5 de Mayo de 2025**, por lo que en el momento de escribir estas líneas hay un plazo de 2 meses para el desarrollo de la aplicación. Por lo que vamos a establecer un **conjunto de entregas** que se deberán respetar, no solo para llevar el proyecto a buen termino en el tiempo especificado, sino para dejar claro que esta, también, fuera del alcance de este proyecto.

Las **entregas** que se han planificado son:

- **7 de Marzo**: Página principal de la aplicación con la información y el formulario de contacto, concretando los elementos de diseño finales. Creación de la Base de Datos.

- **21 de Marzo**: Sistema de autenticación de usuario, añadir comunidades y página principal de comunidad.
- **7 de Marzo**: Sistema de incidencias y Tablón de Anuncios. Página principal de administración de la comunidad.
- **21 de Abril**: Sistema de gestión de espacios Comunes.
- **28 de Abril**: Sistema de Finanzas y Backoffice

Hay que tener en cuenta que las **entregas** se han establecido por **funcionalidades completas**, por lo que se deberá esta trabajando en el **backend** y el **frontend** de forma simultánea. Además, todo el proceso irá dirigido por lo test, empleando la **metodología TDD**.

2.5.2. Coste

En el coste de nuestro proyecto hay que incluir varios elementos. Algunos de ellos ya los tenemos adquiridos con antelación, pero aún así, se incluirá en el coste del proyecto, para tenerlo en cuenta.

Para el desarrollo de nuestra aplicación, se va a emplear un **ordenador portátil** ya adquirido, en concreto un **HP eq0003ns**, al que se han añadido **32 GB de RAM**. Además, hay que tener en cuenta el sueldo del desarrollador, ya que solo va a haber uno, durante los 2 meses de desarrollo, a lo que habrá que sumarle el Hosting aproximado, ya que aún se están valorando diferentes opciones. Eso, supone el siguiente costo:

- HP eq0003ns con 32 GB: **600€**
- Sueldo Programador 2 meses: **2500€**
- Hosting (Anual): **200€**
- Total: **3300€**

Este sería el coste inicial del proyecto, pero siempre pueden surgir algún contratiempo que aumente el coste o lo abarate (poco probable).

2.5.3. Diagrama de Gantt

Para finalizar esta pequeña planificación del proyecto, se expone en la siguiente figura un diagrama de Gantt con la cronología de lo que se ha realizado hasta el momento y de lo que aún queda por realizar.

Figura 2.1: Diagrama de Gantt del Proyecto

3. Requerimientos de Hardware y Software. Financiación.

En esta sección de van a determinar los **requerimientos tanto de hardware** como de **software** para nuestro proyecto, así como el personal, la infraestructura y el modo de financiación que se ha elegido.

3.1. Requisitos Hardware

Para nuestra aplicación **no vamos a necesitar** una **infraestructura hardware muy compleja**, no solo por el tipo de aplicación que vamos a desarrollar, sino porque la aplicación se alojará en un **servicio de hosting externo**, cuando este en producción. Además, solo hay un desarrollador, por lo no se va a tener que instalar la misma infraestructura que si hubiera un equipo de desarrollo.

A nivel de hardware, nuestra aplicación solo va a necesitar un portátil para su desarrollo. En nuestro caso, es un portátil ya adquirido, un HP eq0003ns al que se le han añadido 32GB de RAM, sustituyendo los 8GB que trae por defecto. Aunque el portátil ya esta adquirido, se incluirá en el presupuesto.

Se ha elegido solo este hardware porque nuestra aplicación no necesita muchos recursos para su desarrollo, así que teniendo un ordenador con bastante espacio de almacenamiento y una buena cantidad de memoria RAM, como es el caso, el desarrollo se va a realizar sin problemas.

3.2. Requisitos Software

Todo el software que se va a emplear en el proceso de desarrollo es software libre, por lo que no va a suponer un coste para el desarrollo de la aplicación. En estas sección se va a explicar en software más importante que se va a emplear el el desarrollo y el motivo de su elección.

El sistema operativo elegido para el equipo donde se va a desarrollar el software es **Kubuntu 24.04LTS**. Se ha elegido Linux, y en concreto está distribución, porque es un sistema muy estable, con una gran cantidad de herramientas para el desarrollo de aplicaciones y que se actualiza frecuentemente. Además es gratuito y de código abierto.

Para la parte de **frontend** se ha decidido emplear **React.js**, en concreto la **versión 19**, que ha sido publicada el 5 de Diciembre de 2024. [10] Esta librería de Javascript o Typescript esta enfocado en el desarrollo **orientado a componentes**, lo que nos va a permitir desarrollar un cliente modular, con componentes altamente reutilizables. Actualmente es una de las librerías más ampliamente utilizas en el desarrollo frontend.

Para pa parte de **backend** se va a emplear **Node 22.14LTS**, que es un **entorno de ejecución para Javascript** que nos va a permitir desarrollar nuestra aplicación en el servidor empleando este lenguaje o, como es nuestro caso, Typescript. Junto con Node, emplearemos **Express 4.21**, un framework para node que permite el desarrollo de aplicaciones web en la parte del servidor de forma, rápida y segura, motivo por el que se ha elegido para el desarrollo. Tanto Node como Express tienen un enfoque modular lo que va a proporcionarnos gran versatilidad y alta performance en nuestra aplicación.

Además de este software, como **base de datos** se va a emplear **PostgreSQL**, ya que aporta más flexibilidad en cuanto a tipos de datos, escalabilidad, simultaneidad e integridad de los datos que otras bases de datos basadas en SQL. Es además una base de datos de código abierto y gratuita.

Para el testeo de aplicaciones se van a usar varias herramientas, como **Jest** para el testeo unitario, **Github Actions** para la integración continua y **Cypress** para el testing e2e.

Por último, en lo relativo a la **gestión de proyecto**, se va a emplear por un lado **GitHub** (Git), como repositorio de código que nos ayudará a llevar un control y un historial de los cambios realizados en

nuestra base de código, además de **Trello**, una aplicación tipo **Kanban** que nos permitirá organizar de forma eficiente las tareas que se deben realizar, añadiendo subtareas, fecha de entrega, etc..

3.3. Requisitos de Personal

El personal necesario para desarrollar la aplicación va a ser mínimo, reduciéndose a **1 desarrollador** que va a ocuparse de todo el proceso de desarrollo de la aplicación. Su sueldo se incluirá en el presupuesto durante los 2 meses que va a llevar el desarrollo del proyecto.

3.4. Recursos de Interconexión y Hosting

Los recursos de interconexión de nuestra aplicación van a ser mínimos, especialmente durante el proceso de desarrollo. En este aspecto, lo único que vamos a necesitar es una **conexión a internet estable**, que contrataremos con algún proveedor que nos facilitará el router y todo lo necesario para establecer la conexión. Nuestro equipo se conectara por **red inalámbrica** a la red, por lo que no necesitamos cables de red.

Respecto al Hosting, ahora mismo nuestra primera opción es **Railway**, un proveedor de hosting bastante completo que permite el despliegue de aplicaciones full stack. En principio, el **plan Hobby** puede servir para nuestra aplicación, aunque si vemos que el nivel de usuario es grande se puede cambiar al plan Pro. En la página web de Railway podemos ver con más detalle todos los planes.

3.5. Alternativas de Diseño

En el caso de que el **equipo de desarrollo creciera**, sería conveniente montar **un servidor**, donde realizar el desarrollo de aplicación, para que todos los programadores pudieran conectarse, albergando nuestra base de datos y el servidor web, así como el repositorio de código. Esta opción **encarecería mucho el presupuesto**, pero sería una opción mucho más escalable y segura. Para nuestras circunstancias, el incremento en el coste no merece la pena.

4. Casos de Uso

En esta sección se ha realizado una extracción de todos los casos de uso para todos los usuarios de nuestro sistema. Una vez que se hayan obtenido todos los casos de uso, se van a representar en primer lugar en un **diagrama general de casos de uso**, mostrando todos los casos de uso y su relación entre ellos y con cada actor. Además, los **casos de uso más importantes** se van a detallar en tablas, explicando los pasos a seguir, las precondiciones y postcondiciones.

4.1. Diagrama General de Casos de Uso

En esta sección se muestra el diagrama general de casos de uso, elaborado con UML y que incluye todos los casos de uso y sus respectivas relaciones.

Figura 4.1: Diagrama de Casos de Uso General

Aunque se han intentado incluir todos los casos de uso, hay alguno se ha quedado fuera, por el bien de la legibilidad del diagrama, aunque han sido casos genéricos.

En concreto, el **CU: Consultar Incidencias**, se ha dejado fuera del diagrama, ya que si inclusión disminuía considerablemente la legibilidad de la parte inferior del diagrama, y si rol, dentro de este, es el mismo que el de otros casos de uso similares como **Consultar Tablón** ó **Consultar Zona Común**.

4.2. Descripción Casos de Uso Principales

En esta sección se van a **describir**, mediante tablas, los **casos de uso principales** de nuestro software.

Hay que tener en cuenta que la **numeración de los casos de uso** que vemos en las tablas puede no

empezar desde 0 o tener números faltantes entre una tabla y otra. Esto se debe a que no todos los casos de uso se van a describir en las tablas y la numeración de estos va acorde a su aparición en el diagrama. Por ejemplo, el primer caso de uso va a ser el **CU03 - Registrarse**, ya que los primeros 2 casos de uso, **CU01 - Consultar Información** y **CU02 - Contactar Equipo** no se van a incluir en esta sección.

<table>
<tr><th colspan="2">CU03 - Registrarse</th></tr>
<tr><td>Versión</td><td>1.0</td></tr>
<tr><td>Dependencias</td><td>• RF3 - Registrarse en la Aplicación
• RF4 - Generar Solicitud
• RNF4 - Almacenado de Contraseñas
• RNF6 - Validación de Datos
• RNF14 - Mensajes de Error Descriptivos</td></tr>
<tr><td>Precondición</td><td>El usuario ha accedido al formulario para iniciar sesión y ha pulsado en el enlace para registrarse facilitado en ese formulario.</td></tr>
<tr><td>Descripción</td><td>El sistema deberá validar los datos del usuario y registrarlo, generando una solicitud de incripción en la comunidad con la misma dirección del usuario, si existe alguna.</td></tr>
<tr><td>Secuencia Normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario accede a nuestra página principal</td></tr>
<tr><td>2</td><td>El usuario pulsa en el enlace para iniciar sesión</td></tr>
<tr><td>3</td><td>El usuario pulsa en el enlace para registrarse del formulario de inicio de sesión</td></tr>
<tr><td>4</td><td>El usuario introduce sus datos de registro.</td></tr>
<tr><td>5</td><td>El sistema valida los datos de registro.</td></tr>
<tr><td>6</td><td>El sistema almacena los datos del usuario en la base de datos.
<table>
<tr><td>6.1</td><td>Se realiza el caso de uso Generar Solicitud</td></tr>
</table></td></tr>
<tr><td>6</td><td>El sistema redirige al usuario a la pantalla de inicio de sesión para que inicie sesión con sus datos.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El usuario queda registrado en el sistema y se ha generado una solicitud de inscripción para la comunidad que concuerde con su dirección.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>5</td><td>Los datos introducidos no son correctos.
<table>
<tr><td>E.1</td><td>El sistema informa del error y muestra el formulario para su corrección.</td></tr>
<tr><td>E.2</td><td>Se cancela el caso de uso</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

Figura 4.2: Caso de Uso - Registrarse

CU04 - Iniciar Sesión	
Versión	1.0
Dependencias	• RF6 - Iniciar Sesión • RNF6 - Validación de Datos • RNF14 - Mensajes de Error Descriptivos
Precondición	El usuario esta previamente registrado en el sistema.
Descripción	El sistema deberá permitir al usuario inciar sesión, empleando su correo y contraseña como credenciales.
Secuencia Normal	Paso: Acción 1: El usuario accede a nuestra página principal 2: El usuario pulsa en el enlace para iniciar sesión 3: El sistema valida los datos de sesión. 4: El sistema redirige al usuario a su página de inicio.
Postcondición	*El usuario queda registrado en el sistema y se ha generado una solicitud de inscripción para la comunidad que concuerde con su dirección.*
Excepciones	**Paso: Acción** 3: Los datos introducidos no son correctos. E.1: El sistema informa del error y muestra el formulario para su corrección. E.2: Se cancela el caso de uso

Figura 4.3: Caso de Uso - Inicio de Sesión

CU08 – Crear Incidencia	
Versión	1.0
Dependencias	• RF6 - Iniciar Sesión • RF12 - Cargar Comunidad • RF17 - Gestionar Incidencias
Precondición	El usuario ha iniciado sesión y se ha cargado el panel de control de la comunidad permitiéndole acceder a la sección de incidencias.
Descripción	El sistema deberá permitir al usuario crear una incidencias en su comunidad.
Secuencia Normal	Paso: Acción 1: El usuario accede a la sección indicencias de la página de comunidad. 2: El usuario pulsa en el bóton de Crear Incidencias para deplegar el formulario 3: El usuario introduce los datos de la incidencias. 4: El sistema registra la incidencias del usuario para la comunidad en la que se ha creado.
Postcondición	*Se ha registrado una nueva incidencias en el sistema para la comunidad con el usuario como creador.*
Excepciones	**Paso: Acción** 5: Los datos introducidos no son correctos. E.1: El sistema informa del error y muestra el formulario para su corrección. E.2: Se cancela el caso de uso

Figura 4.4: Caso de Uso - Crear Incidencia

<table>
<tr><th colspan="2">CU10 – Cambiar Estado Incidencia</th></tr>
<tr><td>Versión</td><td>1.0</td></tr>
<tr><td>Dependencias</td><td>• RF6 - Iniciar Sesión
• RF12 - Cargar Comunidad
• RF17 - Gestionar Incidencias
• RF20 - Modificar Estado Incidencia</td></tr>
<tr><td>Precondición</td><td>El administrador ha iniciado sesión y la comunidad se ha cargado, permitiéndole cargar la sección de incidencias de la comunidad.</td></tr>
<tr><td>Descripción</td><td>El sistema deberá permitir al administrador cambiar el estado de una incidencia.</td></tr>
<tr><td>Secuencia Normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El administrador accede a la sección indicencias de la página de comunidad.</td></tr>
<tr><td>2</td><td>El administrador pulsa en el bóton de Cambiar estado para deslegar el formulario.</td></tr>
<tr><td>3</td><td>El administrador selecciona el nuevo estado de la incidencias de la lista.</td></tr>
<tr><td>4</td><td>El sistema registra el cambio y establece la incidencia con el nuevo estado.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha modificado el estado de la incidencia en la base de datos.</td></tr>
<tr><td>Excepciones</td><td>No hay excepciones en este proceso.</td></tr>
</table>

Figura 4.5: Caso de Uso - Cambiar Estado Incidencia

<table>
<tr><th colspan="2">CU11 - Reservar Zona Común</th></tr>
<tr><td>Versión</td><td>1.0</td></tr>
<tr><td>Dependencias</td><td>• RF6 - Iniciar Sesión
• RF12 - Cargar Comunidad
• RF21 – Reservar Espacio Común</td></tr>
<tr><td>Precondición</td><td>El usuario debe haber iniciado sesión y la comunidad se debe haber cargado, permietiéndole acceder a la sección de zonas comunes.</td></tr>
<tr><td>Descripción</td><td>El sistema deberá permitir al usuario reservar un espacio común en una hora y día que esté libre.</td></tr>
<tr><td>Secuencia Normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario pulsa en el icono del espacio común para ver los detalles.
<table>
<tr><td>1.1</td><td>El caso de uso Consultar Espacio Común se ejecuta</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>El sistema muestra un calendario con los días libres en los que se puede realizar la reserva.</td></tr>
<tr><td>3</td><td>El usuario selecciona el día y la hora que quiere realizar la reserva y pulsa el botón reservar.</td></tr>
<tr><td>4</td><td>El sistema registra la reserva en la BD y marca ese día y hora como no disponible para el siguiente usuario.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha registrado la reserva del usuario para esta zona común en el día y la hora elegidos, marcando ese día y hora ya no disponible para esta zona común.</td></tr>
<tr><td>Excepciones</td><td>No hay expcepciones para este caso de uso.</td></tr>
</table>

Figura 4.6: Caso de Uso - Reservar Espacio Común

<table>
<tr><th colspan="2">CU15 - Añadir Registro</th></tr>
<tr><td>Versión</td><td>1.0</td></tr>
<tr><td>Dependencias</td><td>• RF6 - Iniciar Sesión
• RF26 - Gestionar Finanzas
• RF27 - Calcular Finanzas</td></tr>
<tr><td>Precondición</td><td>El administrador debe haber iniciado sesión y la comunidad debe haberse cargado, para poder acceder a la sección de finanzas.</td></tr>
<tr><td>Descripción</td><td>El sistema deberá permitir al administrador añadir un registro a las finanzas, recalculando el balance anual de forma automática.</td></tr>
<tr><td>Secuencia Normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El administrador pulsa en el botón de añadir registro.</td></tr>
<tr><td>2</td><td>El administrador rellena el formulario con los datos del registro, incluyendo su tipo.</td></tr>
<tr><td>3</td><td>El sistema almacena el registro.
<table>
<tr><td>3.1</td><td>Se ejecuta el caso de uso Calcular Finanzas</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>El sistema registra la reserva en la BD y marca ese día y hora como no disponible para el siguiente usuario.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha almacenado el nuevo registro en el sistema de finanzas y se ha recalculado el balance, mostrando el resultado.</td></tr>
<tr><td>Excepciones</td><td>No hay expcepciones para este caso de uso.</td></tr>
</table>

Figura 4.7: Caso de Uso - Añadir Registro

5. Base de Datos

En este apartado vamos a describir el **diseño de la base de datos** que vamos a emplear en nuestra aplicación, mostrando el **diagrama E/R**, así como su **paso a tablas**, explicado los puntos que sean necesarios. Además se creará un script para la creación de la base de datos y otro para poblarla con información.

La base de datos de nuestra aplicación es una **base de datos relacional** que emplea **SQL** para la realización de las consulta y la creación de la base de datos. La sistema de base de datos elegidos ha sido **PostgreSQL**, un sistema de gestión de bases de datos objeto-relacional y que pone especial énfasis en la extensibilidad y compatibilidad con SQL. [11]

PostgreSLQ, además, ofrece atomicidad, consistencia, aislamiento, propiedades ACID, entre otras cosas. Además es soportada por la mayoría de sistemas operativos y se **integra perfectamente** con Apache, soportando las versiones de PosgreSQL 7.2.8 o superior. [12]

5.1. Diagrama Entidad/Relación

En esta sección se muestra el **diagrama entidad relación** de nuestra base de datos. Como podemos ver en el diagrama, inicialmente hay un total de **5 tablas**, pero como veremos en la siguiente sección, las relaciones entre estas tablas dan lugar a **3 tablas más**, que sumarán un total de **8 tablas**, pero esto ya lo veremos con más detalle en la siguiente sección con el paso a tablas.

Se ha intentado crear la base de datos de la forma más simple posible, **sin que haya sido necesario normalizarla**.

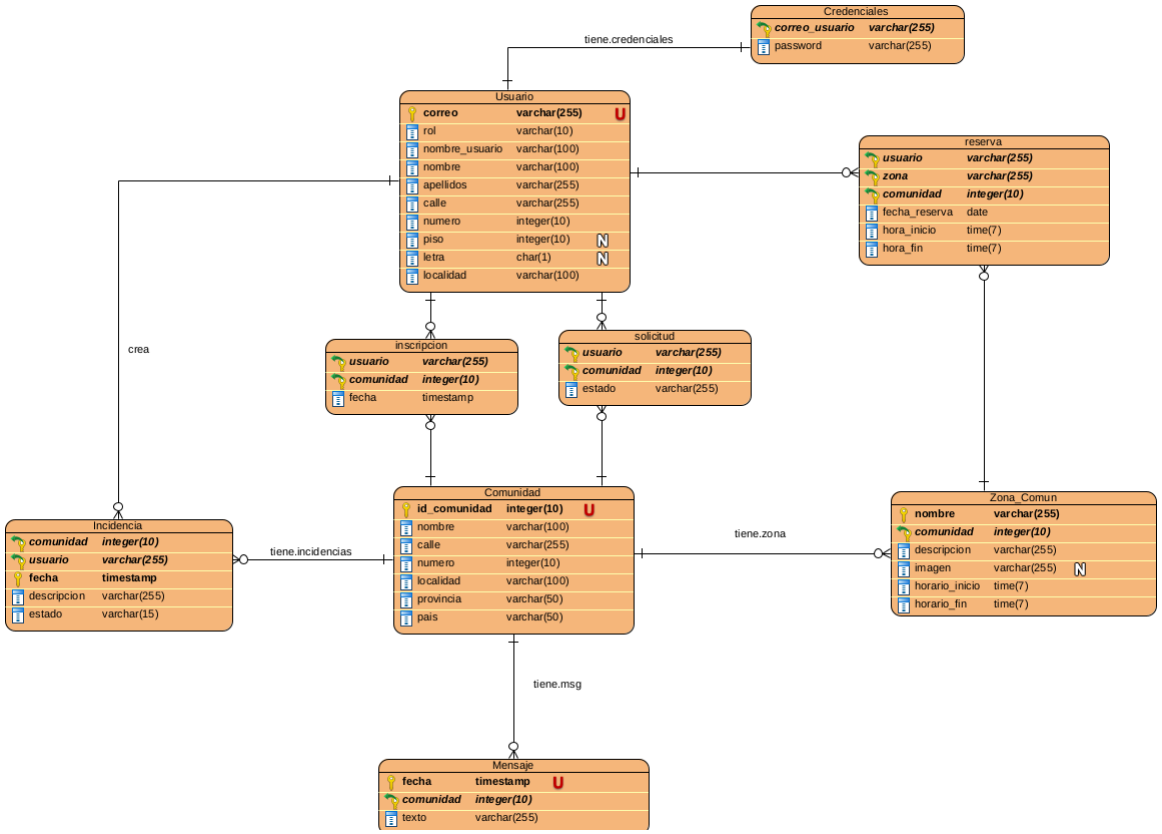

Figura 5.1: Diagrama Entidad/Relación

Como vemos, es un diagrama sencillo, donde se ha intentado **atomizar cada atributo**, además de evitar las referencias circulares. En la siguiente sección veremos el paso a tablas, aunque con este diagrama, donde ya se incluyen como tablas las relaciones, el paso va a ser más directo aún.

5.2. Paso a Tablas

En primer lugar, vamos a pasar las tablas principales, que en nuestro caso serían **Comunidad** y **Usuario**, que no dependen de ninguna otra tabla.

- **Comunidad**(id_comunidad, nombre, calle, numero, provincia, país)
- **Usuario**(correo, rol, nombre_usuario, nombre, apellidos, calle, número, piso, letra, localidad)

A continuación, vamos a pasar las tablas que dependen de estas para identificarse, y que son **Credenciales**, **Zona_Comun**, **Mensaje** y **Incidencia**. En estas tablas hay una relación de M:1 o 1:1 con las tablas principales, por lo que la clave principal se cede desde las tablas principales a éstas.

- **Credenciales**(correo_usuario, password)
- **Mensaje**(fecha, comunidad, texto)
- **Zona_Comun**(nombre, comunidad, descripción, imagen, horario_inicio, horario_fin)
- **Incidencia**(fecha, comunidad, usuario, descripción, estado)

Por último, vamos a pasar la tablas que se han generado con relaciones, en concreto, con relaciones M:N entre dos tablas. Estas tablas son **inscripción**, **solicitud** y **reserva**.

- **inscripción**(usuario, comunidad)
- **solicitud**(usuario, comunidad, estado)
- **reserva**(usuario, comunidad, zona, fecha_reserva, hora_inicio, hora_fin)

Como vemos, el paso a tablas ha sido prácticamente directo, ya que en el diagrama las relaciones ya se han procesado y se han generado las tablas o los movimientos de claves necesario.

5.3. Scripts de la Base de Datos

Acompañando a este documento se han incluido 2 script, en **SQL**, para crear la base de datos y poblarla con datos, para poder realizar algunas consulta u operaciones de prueba.

Hay que tener en cuenta, que la base de datos que se va a emplear en la aplicación es **PostgreSQL**, por lo que aunque la mayoría de sentencias van a ser totalmente compatibles con SQL, puede ser que haya algunos **tipos de datos**, o algunas **sentencias**, como por ejemplo, para generar campos de id de forma aleatoria, que **no estén soportadas** en MySQL ú Oracle. Por eso, para la correcta creación de la base de datos será **necesario el uso de PostgreSQL**.

Respecto al **tema de los roles**, aunque se han creado y usado, el control de los roles se va a **implementar en la aplicación** propiamente dicha, en el backend y frontend, por lo que no tiene mucha relevancia si solo se utilizan en la base de datos de forma manual.

Por último, destacar que aunque se ha intentado realizar el diseño de la forma más eficiente, **puede que haya cambios** en la arquitectura de la base de datos. De ser así, **se documentará** cada cambio y **se explicará** el motivo de haberlo realizado.

Anexo A Descripción de la Interfaz de Usuario

SolucionesVecinales es una aplicación web, por lo que tendrá una **interfaz web** que podrá ser accedida desde cualquier navegador y prácticamente desde cualquier dispositivo. La interfaz estará compuesta por un conjunto de páginas web que ayudarán a los usuarios a utilizar los diferentes servicios que ofrece la aplicación para gestionar la comunidad de vecinos.

Las **páginas** de las que estará compuesta nuestra interfaz son las siguientes:

- **Página Principal**: está es la página que se carga cuando cualquier usuario entra a nuestra aplicación. La función principal de esta pantalla es la de **ofrecer información sobre la aplicación**, así como proporcionar un **menú con diferentes opciones** que, además de permitir la navegación por las diferentes secciones permita a un invitado **registrarse** o **realizar login**.

  Las secciones de las que estará compuesta esta interfaz serán las siguientes:

  - **Cabecera**: aquí se mostrará una imagen con un eslogan, el nombre de la aplicación y un elemento CTA para que los usuarios se registren o realicen login.
  - **Barra de Menú**: esta barra, que se sitúa inicialmente encima de la cabecera y que debe permanecer en todo momento en la parte superior de la pantalla, esta compuesta del **logo de la aplicación** en el lado izquierdo y de un **menú** en el lado derecho. El menú estará siempre presente en todas las páginas de la web, aunque dependiendo del contexto, la opción del menú pueden cambiar.
  - **Secciones**: esta página contendrá, como mínimo, **3 secciones**, donde se explicará un poco el **propósito** del software, sus principales **características**, lo que nos diferencia de la competencia, etc. Las secciones deben ser **visualmente atractivas**, incluyendo imágenes e iconos y con una disposición adecuada.

    Una sección que deberá ser **obligatoria**, es la sección "**Contacta con nosotros**", compuesta por un formulario y que permitirá a cualquier usuario, registrado o no, contactar con el equipo de desarrollo.
  - **Pie de Página**: en el pie de página se añadirá una lista de enlaces con las diferentes secciones de la página, un mapa del sitio web y un conjunto de enlaces de redes sociales de la aplicación o en su defecto del equipo de desarrollo.

  Tanto la **barra de menú** superior como el **pié de página** se mantendrán **visibles en todas las páginas** de la aplicación, cambiando en ciertos aspectos u ofreciendo diferentes opciones.

- **Página de Login**: el propósito de esta página es permitir que los usuarios ingresen al sistema. La pantalla inicial esta compuesta por un **formulario**, que debe aceptar un **email** y una **contraseña** como entrada, con el botón enviar en la parte inferior. Además, incluirá un **enlace** debajo del botón, centrado, que permitirá al invitado **registrarse**, en caso de que no este registrado, desplegando un nuevo formulario que veremos en detalle a continuación. Solo el **administrador de la web** podrá realizar el ingreso al sistema empleando el **usuario WebAdmin**, el resto de usuarios deberán usar su correo.

- **Página de Registro**: en esta pantalla, que se accede desde la pantalla de login, se muestra un formulario que **permitirá a un invitado registrarse** en el sistema. Este formulario estará compuesto por diferentes campos que permitirán al invitado introducir la información necesaria pare registrarse, como puede ser Nombre, Correo, Dirección, rol, etc. Debajo, aparecerá el **botón registrarse** que permitirá al invitado registrar todos sus datos.

- **Página Principal de Comunidad**: una vez que el usuario haya ingresado al sistema, se mostrará la pantalla de comunidad. El propósito de esta pantalla es servir como **puerta de entrada a la gestión de al comunidad** y como **punto de información** para el usuario. Esta interfaz estará compuesta por los siguientes elementos:

  - **Barra de Menú**: similar a la que vemos en la página principal, pero las opciones de menú "Login/Registrarse" habrá cambiado por el **icono de un usuario**. Este icono mostrará un **menú vertical** con las opciones "**Perfil**" y "**Logout**", que permitirán acceder a la información de perfil y hacer logout respectivamente.
  - **Menú Lateral**: un menú lateral, situado a la izquierda, que mostrará los diferentes servicios que ofrece al aplicación para gestionar la comunidad. Las opciones variarán según si el usuario tiene el rol de administrador o de inquilino, pero la opciones comunes a ambos son las siguientes:
    - **Inicio**
    - **Tablón de Anuncios**
    - **Documentos**
    - **Incidencias**
    - **Espacios Comunes**
    - **Finanzas**

    Adicionalmente, si el usuario tiene el **rol administrador**, se mostrará la opción "**Administrar**" , que permitirá al administrador gestionar los aspectos principales de la comunidad desde una misma pantalla. La cual se describirá más adelante.

    El menú lateral se mostrará solo en caso de que el usuario, independientemente de su rol, este inscrito en una comunidad. En caso contrario no se mostrará. Si es un **administrador**, se mostrará en su lugar un **formulario para añadir una comunidad**. Si es un **inquilino**, se mostrará un **mensaje con el estado** de su solicitud de suscripción.
  - **Zona de Contenido**: en esta zona, que estará ubicada en el centro y es el elemento principal de ésta página. Cuando un **inquilino inscrito en una comunidad** acceda a la página principal de comunidad, aquí se mostrar la **información** de ésta, como nombre, dirección y una imagen. Además, se mostraran la actividad reciente del usuario, así como otros notificaciones, como las reservas realizadas o el estado de las incidencias creadas.

    En esta zona de la interfaz se mostrará todas la información de los diferentes servicios y es donde se permitirá interactuar con la comunidad. Seleccionando alguna de las opciones del menú, los **principales elementos** que se mostrarán en esta zona y que permiten **realizar las acciones sobre la comunidad** son las siguientes:

    - **Inicio**: cuando un usuario pinche en esta opción cargara la información por defecto de la página principal de comunidad, que se ha descrito en el párrafo anterior.
    - **Tablón de Anuncios**: este servicio nos permite acceder a los mensajes que se han publicado en la comunidad y mostrará una lista con todos los mensajes, junto con su fecha. Si **es un administrador**, aparecerá un botón en la parte inferior izquierda, debajo del tablón, con el texto "**Añadir mensaje**", que desplegará un formulario que permitirá añadir un nuevo mensaje al tablón de anuncios.

- **Documentos**: pulsando en esta opción se mostrarán los **documentos relativos a la comunidad**, como pueden ser actas de reuniones, facturas, etc. Los documentos se mostrarán como un conjunto de iconos con un nombre debajo. Tanto el icono como el nombre deberán ser un enlace que al pulsarlo permita a los inquilinos descargar el documento en cuestión.

  En el caso de que el **usuario sea un administrador**, debe aparecer un botón con el texto "**Añadir Documento**"que desplegará una formulario y le permitirá añadir nuevos documentos al directorio.

- **Incidencias**: este servicio permitirá a los inquilinos gestionar las incidencias, mostrando una **lista con todas las incidencias** de la comunidad, junto con su fecha de creación y su estado. Para un **inquilino**, solo se mostrarán **sus incidencias** y todos aquellas que hayan sido **marcadas como públicas** por los inquilinos que las hayan creado. Para el administrador, serán visibles todas las incidencias.

  Debajo de la lista de incidencias, debe haber un **botón** con el texto "**Crear incidencia**", que desplegará un formulario y permitirá a cualquier usuario crear una nueva incidencia. Además, si el **usuario conectado es el administrador**, aparecerán 2 botones a la derecha de cada incidencia, que permitirán tanto **borrar las incidencias** como **cambiar su estado**. En caso de que el usuario no sea administrador, solo aparecerá el botón “Borrar” en las incidencias que hayan sido creadas por él mismo.

- **Zonas Comunes**: se mostrarán las diferentes zonas comunes de la comunidad y se permitirá su reserva. Se mostrará un icono por cada zona, con todos los espacios comunes que hay en al comunidad, que permitirá clickar en ellos y acceder a la página de cada espacio común. Además, para los administradores, se mostrará un botón con el texto “Añadir Espacio” que mostrará un formulario que permitirá añadir un nuevo espacio común, con información como el nombre del espacio común, sus características y el horario de uso, además de la posibilidad de incluir una foto de dicho espacio.

  - **Detalle de Zona Común**: se muestran los **detalles de una zona común**. Aparecerá **una imagen**, si se hubiera incluido. Sino se cargara una imagen por defecto. Encima aparecerá el **nombre** del espacio común y al lado derecho de la imagen, una **descripción** de dicho espacio. Debajo de la descripción aparecerá el horario de uso de dicho espacio.

    Debajo de este bloque, se mostrará un **calendario**. Los **días en los que no se pueda reservar** tendrán un fondo **rojo** y en los que aún **si se pueda** tendrán un **fondo verde**. Al **pinchar** sobre uno de estos días se desplegará una **lista con los posibles horarios** en los que se puede reservar, permitiendo pinchar en uno y pulsar el **botón Reservar"** que habrá también aparecido junto con los horarios.

- **Finanzas**: se mostrará una **tabla con las finanzas** de la comunidad, indicando los ingresos y el tipo de estos, como cuotas, seguros, etc., así como los gastos y su tipo, limpieza del edificio, arreglos, seguros, etc. La información de mostrará para este ejercicio, realizando una estimación, y se irá modificando de forma automática cuando se añadan gastos o ingresos, aunque también se podrá consultar para ejercicios anteriores.

  Solo los **administradores** puede modificar esta tabla, mediante un formulario para añadir un registros en el que se podrá seleccionar el tipo de registro, si es un ingreso o gasto, y el tipo de cada uno, así como la cantidad. Cuando se agregue un nuevo registro, el sistema calculará el balance con los nuevos datos y los mostrará. Este formulario se

mostrará después de pulsar el botón “Añadir” debajo de la tabla de finanzas. Solo se pueden añadir o borrar registros en el ejercicio actual.

- **Administrar**: esta opción solo es elegible por los **administradores** y permite al administrador **gestionar** todos los aspectos de una **comunidad** mediante un panel de control.

  En la zona de contenido se mostrará **información de al comunidad**, que será editable, así como los servicios mencionados en este punto: Incidencias, Documentos, Finanzas, etc., con opciones para editar, consultar añadir o eliminar elementos a cada apartado. Los apartados se mostrarán uno al lado de otro con iconos y al pulsar en ellos se desplegará una lista con todos los elementos que estos contienen, ocultando el resto secciones. Además aparecerá un botón para poder volver hacia atrás a la lista de servicios.

- **Barra de Búsqueda**: cabe destacar que **todas las secciones** incluirán, encima de la tabla o lista donde se muestra la información, una **barra de búsqueda** que permitirá seleccionar solo las entradas coincidentes con los términos que queramos.

- **Página de Añadir Comunidad**: esta pantalla, a la que solo tienen acceso los administradores, permite **añadir una nueva comunidad** de vecinos. La interfaz mostrará un **formulario web** con diferentes datos que deberán rellenarse para dar de alta la comunidad, principalmente la **dirección**, que será única para cada comunidad. En el caso de que se pueda dar de alta la comunidad se mostrará una mensaje de éxito y se redireccionará a la página principal de comunidad. En caso contrario, se mostrará los mensajes de error adecuados.

- **Página de Perfil**: en esta pantalla, que se accede desde el menú vertical del icono de perfil, se mostrará un **formulario con toda la información del usuario** en la zona de contenido. Este formulario se podrá **editar** y actualizar mediante un botón que habrá debajo de él con el texto "Guardar".

  Además, debajo de este aparecerá un **botón** con la opción “**Eliminar usuario**” que permitirá a un usuario eliminar su información, y que al ser pulsado ocultará el formulario con al información y mostrará una ventana de confirmación. Si la confirmación es positiva, se eliminará al usuario, y se redireccionará la pantalla principal.

- **Backoffice**: esta es una interfaz especial, ya que solo es accesible por el **administrador web** o **WebAdmin** y sirve como un punto centralizado para gestionar todos los servicios y consultar datos sobre la aplicación. En la parte superior aparecerá un tabla con **estadísticas** de uso generales de la aplicación. Debajo de esta tendremos un formulario con diferentes opciones para generar informes más específicos que se mostrarán en la tabla de superior.

  Debajo de este apartado, aparecerán **dos iconos**, uno para **las comunidades** y otro para **los usuarios**. Al pulsarlo, aparecerá una pantalla con una lista desplegable, con todas las comunidades registradas en al aplicación. Al pulsar en una comunidad, se desplegará una lista mostrando los usuarios registrados en al comunidad, el administrador y los espacios comunes registrados, el número de documentos, etc. En el caso de los usuarios se mostrará una tabla con todos los usuarios registrados en al aplicación.

  Dentro de las pantallas de comunidad y usuarios, el **WebAdmin** podrá **añadir** o **eliminar** cualquier comunidad o usuario. También se podrán eliminar servicios dentro de una comunidad.

Además de estas páginas, hay que tener en cuenta que la aplicación debe **visualizarse correctamente en cualquier dispositivo**. Aunque se añadirán restricciones más adelante en este aspecto, hay que tener en cuenta que algunos **elementos cambiarán** cuando la aplicación se visualice en **dispositivos pequeños**, como móviles o tablets. Estos elementos son:

- **Barra Superior**: en la barra superior todas las opciones del menú se agruparán en un menú hamburguesa, el cual se situará en la parte derecha de la barra.
- **Menú Lateral**: el menú lateral se ocultará, mostrando la interfaz una flecha en la parte superior izquierda, donde debería estar el menú. Al pulsar esta flecha

Más adelante, en la sección de adecuación al entorno y restricciones de diseño se proporcionará más información sobre el desempeño de la aplicación en dispositivos móviles y tablets.

Anexo B Tablas de Requisitos

En este anexo se incluyen varias tablas con los requisitos funcionales y no funcionales, ya que son muy extensas se ha preferido dividir éstas para no hacer tablas que ocupen varias páginas.

Número y Tipo	**Descripción**
RF1.- Consultar Web Principal	Cualquier usuario podrá acceder a la página web principal para consultar la información sobre nuestra aplicación
RF2.- Contactar Equipo	Cualquier usuario podrá contactar con el equipo de desarrollo usando el formulario de la página principal, dejando un mensaje y su correo
RF3.- Registrarse en la Aplicación	Los invitados podrán registrarse en nuestra aplicación empleando el formulario de la página de registro e introduciendo su correo, nombre, apellidos, nombre de usuario y dirección
RF4.- Generar Solicitud	El sistema deberá generar una solicitud de inscripción en la comunidad que coincida con la dirección que ha especificado el invitado que se ha registrado
RF5.- Aprobar Solicitud:	El administrador de la comunidad deberá poder aceptar o denegar la solicitud de inscripción en la comunidad de un invitado
RF6.- Iniciar Sesión	Un inquilino o administrador podrá ingresar al sistema empleando su correo y contraseña como credenciales
RF7.- Añadir Comunidad	El administrador podrá añadir una nueva comunidad de vecinos, independientemente de que este inscrito ya en alguna otra, especificando un nombre, descripción, imagen y dirección de la comunidad
RF8.- Modificar Perfil	Un inquilino o administrador podrá acceder a su perfil y modificar sus datos en el sistema
RF9.- Eliminar Usuario	Un inquilino podrá eliminar su usuario desde su página de perfil, pudiendo ser también eliminado por el administrador
RF10.- Eliminar Administrador	El administrador podrá eliminar su información desde su página de perfil, siempre y cuando no este adscrito a ninguna comunidad
RF11.- Eliminar Comunidad	El administrador podrá eliminar una comunidad de vecinos, que haya creado, desde la sección para administrar la comunidad
RF12.- Cargar Comunidad	Cuando un inquilino o administrador inicie sesión, se cargará la página principal de la comunidad mostrando los datos de ésta, así como las notificaciones del usuario
RF13.- Notificar Usuario	El sistema creará notificaciones con los documentos descargados en los últimos 10 días, las reservas de espacios comunes y las incidencias de un usuario, mostrándolos en la página principal de la comunidad

Tabla B.1: Requisitos Funcionales (Parte I)

Número y Tipo	Descripción
RF14.- Consultar Tablón	Un inquilino o administrador podrá consultar los mensajes creados en el tablón de anuncios
RF15.- Gestionar Tablón	El administrador podrá crear o eliminar mensajes del tablón de anuncios de la comunidad
RF16.- Consultar Documentos	Un inquilino podrá consular los documentos relativos a la comunidad y descargarlos a su dispositivo
RF17.- Gestionar Documentos	El administrador podrá añadir o eliminar documentos de una comunidad
RF18.- Consultar Incidencias	Un inquilino podrá consultar las incidencias publicas o creadas por el mismo, mientras que una administrador podrá consultar todas las incidencias relativas a una comunidad
RF19.- Gestionar Incidencias	Un inquilino podrá crear incidencias, así borrar o modificar las incidencias creadas por el mismo, mientras que el administrador podrá eliminar las incidencias de cualquier usuario
RF20.- Modificar Estado de Incidencia	El administrador podrá cambiar el estado de las incidencias entre: “Creada”, “En Proceso” y “Resuelta”
RF21.- Añadir Espacio Común	El administrador podrá añadir espacios comunes especificando su nombre, descripción y horario de reserva
RF22.- Reservar Espacio Común	Un inquilino podrá reservar un espacio común seleccionando en un calendario el día y hora que quiere realizar la reserva, siempre y cuando este libre
RF23.- Cancelar Reserva	Un inquilino podrá cancelar una reserva de un espacio común desde la sección de espacios comunes o desde la página principal de la comunidad
RF24.- Gestionar Espacios Comunes	El administrador podrá eliminar o modificar la información relativa a un espacio común
RF25.- Consultar Finanzas	Un inquilino podrá consultar las finanzas de la comunidad en la sección finanzas que se mostrarán en una tabla con los datos estimados para el curso actual, pudiendo también consultar los datos del curso anterior
RF26.- Gestionar Finanzas	El administrador podrá añadir, eliminar o modificar los registros de la tabla de finanzas
RF27.- Calcular Finanzas	El sistema realiza el calculo de las finanzas cada vez que se añada o elimine un pago o un ingreso
RF28.- Cerrar Sesión	Un inquilino o administrador podrán cerrar la sesión en el sistema desde el icono de usuario
RF29.- Administrar Servicios	el administrador web podrá iniciar sesión empleando el usuario WebAdmin y acceder al panel de control para administrar los diferentes servicios web así como para obtener datos de uso de la aplicación

Tabla B.2: Requisitos Funcionales (Parte II)

Número y Tipo	Descripción
RNF1.- Arquitectura	La aplicación se implementará empleando una arquitectura cliente-servidor
RNF2.- Protocolo Comunicación	La aplicación empleará el protocolo HTTPS entre el cliente y el servidor, no permitiendo conexión con otro protocolo que no sea este
RNF3.- Peticiones Limitadas	Las peticiones por segundo al servidor estarán limitadas a 10, para evitar ataques de tipo DOS
RNF4.- Almacenado de Contraseñas	El sistema se encargará de almacenar las contraseñas en la base de datos de forma segura, nunca en texto plano, empleando un algoritmo hash y almacenando el resultado encriptado en la base de datos
RNF5.- Robustez Contraseñas	El sistema solo aceptará contraseña con un mínimo de 12 caracteres y con al menos un número y un signo de puntuación
RNF6.- Validación de Datos	Los datos introducidos por los usuarios serán validados tanto en el cliente como en el servidor
RNF7.- Tiempo de Carga	La aplicación deberá tener un tiempo de carga en el navegador del usuario inferior a 3 segundos
RNF8.- Compresión Archivo Multimedia	Los archivos multimedia empleados en la web deberán estar comprimidos lo máximo posible sin que estos pierdan calidad significativamente
RNF9.- Uso de Memoria	El cliente de nuestra aplicación deberá usar el mínimo de memoria posible para cumplir su función de forma eficiente
RNF10.- Interfaz Adaptable	El cliente de nuestra aplicación deberá tener una interfaz que se adapte a diferentes dispositivos, incluyendo dispositivos móviles y tablets
RNF11.- Alternativa Textual	Cualquier elemento de la interfaz que no sea texto deberá tener una alternativa textual
RNF12.- Contraste	La interfaz tiene que tener un contraste mínimo entre el texto y el fondo de 4.5:1 para textos pequeños y 3:1 para textos grandes
RNF13.- Navegabilidad	La interfaz deberá ser totalmente navegable con el teclado
RNF14.- Mensajes de Error Descriptivos	Cualquier mensaje de error que produzca la aplicación, ya sea para los usuarios o desarrolladores, debe explicar correctamente por qué se ha producido el error
RNF15.- Test Unitarios	Todo componente, función o clase deberá tener implementando su correspondiente test unitario
RNF16.- Cobertura de Tests	Los test deberán cubrir el 100 % del código desarrollado

Tabla B.3: Requisitos No Funcionales (Parte I)

Número y Tipo	Descripción
RNF17.- CD/CI	Los test unitarios, así como los e2e y los de integración deberán ejecutarse con un sistema de integración/despliegue continuo
RNF18.- Calidad Global del Código	La calidad global del código debe garantizarse, empleando para ellos herramientas como Code Climate, eslint y todas aquellas que sean necesarias para que el código se adapte a las buenas prácticas y estándares
RNF19 - Formato del Código	El código se formateará empleando prettier, con una configuración predefinida para que todo el código, de cualquier desarrollador, tenga el mismo formato
RNF20 - LOPD y RGPD	En cumplimiento de las leyes LODP y RGPD, deberán mostrarse a los usuarios los documentos legales siguientes: Política de Privacidad y Política de Cookies

Tabla B.4: Requisitos No Funcionales (Parte I)

Glosario

SSL El protocolo SSL ó Secure Socket Layer es un protocolo de cifrado para comunicaciones en red que proporciona un canal seguro entre dos computadores o dispositivos que se comunican a través de internet o una red local.. 8

HTTP El protocolo HTTP o HyperText Transfer Protocol es un protocolo desarrollado por Tim Berners-Lee en 1989 y que permite la transferencia de archivos entre dos computadores o dispositivos y que es la base de comunicación en la World Wide Web.. 9

idempotente Un método HTTP es idempotente si el effecto que se produce en el servidor es el mismo si se manda una sola solicitud que si se mandan varias solicitudes iguales.. 9

API Una API o Applicacition Programming Interface, es una pieza de código que permite a dos aplicaciones comunicarse entre sí para compartir información y funcionalidades.. 9

frontend En desarrollo web, el frontend es la parte de la aplicación que interactura con los usuarios, incluyendo todos los aspectos gráficos de la web y otros elementos que permiten navegar a los usuario dentro de la página.. 12

backend En desarrollo web, el backend es la parte de la aplicación que reside en el servidor y que implementa toda la lógica de la aplicación para permitir que la aplicación trabaje con la base de datos.. 12

SQL SQL o Structured Query Language es un lenguaje de programación diseñado para administrar y recuperar información de una base datos, empleando el calculo relacional para hacerlo de una forma sencilla.. 12

librerías En informática. una librería es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, y que ofrece una interfaz bien definida para la funcionalidad que aporta.. 12

ORM Un ORM o Object Relational Mapping es una técnica que nos permite interactuar con bases de datos relacionales empleando funciones u objetos en vez de consultas SQL. También se conoce como ORM al software o librería que implementa esta funcionalidad.. 12

e2e Las pruebas e2e (End-to-End) son un tipo de test que prueba el sistema como un conjunto, desde la perspectiva del usuario, probando todos los posibles caminos de la aplicación.. 13

hasing El hashing es una herramienta criptográfica que se emplea para generar un conjunto de datos cifrados a partir de otro conjunto de datos, como una cadena o un archivo... 13

DOS Un ataque DOS o Denial of Service es un tipo de ataque que consiste en saturar de la conexión de una aplicación o ordenador de forma que este no pueda ofrecer lo sevicios que estaba ofreciendo.. 13

Markdown Markdown es un lenguaje de marcado ligero creado por John Gruber y Aaron Swartz cuya finalidad es conseguir la máxima legibilidad tanto en la entrada como en la salida.. 19

Hosting Servicio de alojamiento web que permite publicar una página web o aplicación en internet.. 22

ACID En bases de datos se denomina ACID a las transacciones que son atómicas, consistentes, aisladas y durables.. 27

login Proceso mediante el que se controla el acceso individual a un sistema informático mediante el uso de credenciales.. 30

CTA Un CTA, del inglés Call To Action, es una imagen o frase con un botón que busca persuadir a los usuarios para que realicen una acción concreta.. 30

logout Proceso mediante el que un usuario que tienen una sesión activa en un sistema informático puede cerrar ésta y salir del sistema.. 31

menú hamburguesa Es un tipo de menú web que consiste en un icono rectangular con varias líneas horizontales en su interior, y que se despliega al ser clickado. 34

Referencias

[1] Wikipedia - HTTP.
https://en.wikipedia.org/wiki/HTTP.

[2] Mozilla Developer Network - HTTP Methods.
https://developer.mozilla.org/es/docs/Web/HTTP/Methods.

[3] Mozilla Developer Network - HTTP Methods: GET.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET.

[4] Mozilla Developer Network - HTTP Methods: POST.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST.

[5] Mozilla Developer Network - HTTP Methods: DELETE.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE.

[6] Mozilla Developer Network - HTTP Methods: PUT.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT.

[7] Grupo Atico 34 - Cumplimiento de la LOPD.
https://protecciondatos-lopd.com/empresas/pagina-web/.

[8] W3 - Sumario WCAG 2.
https://www.w3.org/WAI/standards-guidelines/wcag/es.

[9] Licencia MIT.
https://opensource.org/license/mit.

[10] Github - React Repository.
https://github.com/facebook/react/releases.

[11] Wikipedia - PosgreSQL.
https://en.wikipedia.org/wiki/PostgreSQL.

[12] Apache - PostgreSQL.
https://db.apache.org/ddlutils/databases/postgresql.html.