



Proyecto Integrado

Portal del Viajero - Alquiler rural

Martín Candela, Juan

Dorante Lucas, David

Proyecto Integrado Ciclo Superior DAW

Dpto. de Informática – IES AlMudeyne

LISTA DE CAMBIOS

Primera Entrega:

1. Toda la documentación creada hasta el punto 3.4.
2. Creada estructura inicial del backend (entidades jpa, repos, services...)
3. Creada estructura de datos inicial del frontend.

Segunda Entrega:

1. Mejoras en varios puntos de la documentación.
2. Toda la documentación creada hasta el punto 4.
3. Anexo con el sueldo de los trabajadores del proyecto (Estudio coste proyecto) y costes actualizados.
4. Añadida indentaciones a los textos.
5. Cambiados los formatos de listas por tablas en varios apartados de la documentación (pe: requisitos de información).
6. Creación de un diagrama Entidad Relación válido.
7. Creación de un prototipo visual y navegable de la aplicación hecho en Figma.
8. Empezando con la creación de un servicio de autenticación OAuth2.
9. Creando un despliegue de la aplicación haciendo uso de Docker y Kubernetes.
10. Codificación de varias páginas del front (Estáticas y no responsive).
11. Mejora exponencial de la presentación del proyecto en github (creación de tableros kanban, issues, readme...)
12. Todos los docentes han sido invitados al github del proyecto.
13. Creación de nuevos diagramas situados en la carpeta de public-resources.
14. Índice acortado hasta encabezados de nivel 3.
15. Diagramas de Análisis mejorados.

ÍNDICE

LISTA DE CAMBIOS.....	2
Primera Entrega:.....	2
ÍNDICE.....	3
INTRODUCCIÓN.....	5
ESTUDIO DE VIABILIDAD.....	6
Descripción del Sistema Actual:.....	6
Descripción del Sistema Nuevo:.....	7
Identificación de Requisitos del Sistema:.....	8
Requisitos de información:.....	8
Tabla de correspondencia de tipos de datos:.....	8
Alojamiento:.....	9
Perfil:.....	10
AlojamientoComodidadAlojamiento:.....	10
Usuario:.....	11
AlquilerAlojamiento:.....	12
ImagenAlojamiento:.....	12
ComodidadAlojamiento:.....	13
TipoComodidad:.....	14
UbicacionAlojamiento:.....	15
ValoracionAlojamiento:.....	16
Requisitos funcionales:.....	17
Registro de usuarios:.....	17
Inicio de sesión:.....	17
Gestión de casas rurales:.....	17
Gestión de reservas:.....	17
Gestión de clientes:.....	18
Búsqueda y filtro de las casa rurales:.....	18
Notificaciones:.....	18
Otros Requisitos:.....	19
Control de versiones:.....	19
Portabilidad de la aplicación:.....	19
Balanceo de carga:.....	19
Seguridad de la aplicación:.....	19
Descripción de la solución:.....	20
Planificación del proyecto:.....	22
Equipo de trabajo:.....	22
Diseñador Web:.....	22
Desarrollador Web:.....	23
Jefe de Proyecto:.....	24
Planificación temporal:.....	25

Fase 1: Preparación y planificación.....	25
Fase 2: Desarrollo.....	26
Fase 3: Pruebas y Depuración.....	27
Fase 4: Despliegue y puesta en marcha.....	27
Fase 5: Mantenimiento y Soporte.....	28
Estudio del coste del proyecto:.....	29
Costes de Personal.....	29
ANÁLISIS DEL SISTEMA DE INFORMACIÓN.....	30
Identificación del entorno tecnológico:.....	30
Entorno Tecnológico de Desarrollo.....	30
Entorno Tecnológico de Explotación.....	32
Modelado de datos:.....	33
Modelo Entidad Relación y esquema de base de datos:.....	33
Modelo Relacional:.....	34
Identificación de los usuarios participantes y finales:.....	35
Usuarios Participantes.....	35
Usuarios Finales.....	35
Otros sistemas.....	36
Diagramas de Análisis:.....	37
Diagramas de Casos de Uso.....	37
Diagrama de Estados.....	41
Diagrama de Secuencia o Interacción.....	42
Tablas de decisión.....	43
Definición de interfaces de usuario:.....	45
Especificación de principios generales de interfaz:.....	45
Especificación de formatos individuales de la interfaz de pantalla:.....	48
Identificación de perfiles de usuario:.....	56
Especificación de los formatos de impresión:.....	58
Especificación de la navegabilidad entre pantallas:.....	59
CONSTRUCCIÓN DEL SISTEMA.....	63
Control del sistema en Github:.....	63
Despliegue de la aplicación con Docker y Kubernetes:.....	66

INTRODUCCIÓN

La gestión de alojamientos rurales ha sufrido una creciente demanda en los últimos años. Sin embargo, a pesar de la popularidad de las casas rurales, tanto para vacaciones como para eventos especiales, la gestión de reservas de alojamientos rurales ha enfrentado muchos problemas.

Es por ello que en este documento se establecen las bases para el desarrollo de un proyecto integrado cuyo objetivo es crear una aplicación web para la gestión de reservas de casas en entornos rurales. Este proyecto surge con la finalidad y el objetivo de brindar a los usuarios una plataforma que les permita conocer la disponibilidad de casas rurales de acuerdo a sus necesidades específicas, así como realizar reservas de manera efectiva durante períodos determinados.

Para cubrir estas necesidades, se han conseguido identificar requisitos y funcionalidades claves que se abordarán en el desarrollo de la aplicación. Todas estas incluyen la gestión de casas rurales, la disponibilidad de las casas rurales en tiempo real, así también como la gestión de clientes y sus reservas.

Se llevará a cabo este proyecto de manera eficiente, ya que se establecerán etapas claras de desarrollo, que abarcarán desde la preparación y planificación inicial hasta el despliegue y mantenimiento continuo de la aplicación. Además, se emplearán tecnologías modernas, incluyendo contenedores Docker para garantizar la portabilidad y escalabilidad del sistema.

El objetivo final de este proyecto es proporcionar una herramienta que sea sólida, robusta y fácil de usar, y que a la vez satisfaga las necesidades tanto de los usuarios finales como de la empresa, facilitando la gestión de reservas de casas rurales y mejorando la experiencia general de los usuarios en este sector.

ESTUDIO DE VIABILIDAD

Descripción del Sistema Actual:

A continuación, se expondrán algunas de las características que implementa el sistema actual de reservas de casas rurales:

En el sistema actual se puede **iniciar sesión** con una **cuenta** que previamente habremos de haber **registrado**.

El sistema también dispone de un **buscador** con varias opciones, en el cual podremos **filtrar** por **provincia, rango de precio, tamaño, número de habitaciones, número de personas, las características de dicha casa rural, las fechas que el cliente desee...** Cada sistema posee filtros parecidos, muy útiles a la hora de que el consumidor encuentre su alojamiento ideal.

Dispone de **alojamientos** que se pueden **alquilar** y que **cumplen** con los **requisitos solicitados por el cliente**. En caso de **no** ser solicitado **ningún requisito**, se muestran **todos los anuncios disponibles**, junto con las imágenes de la casa rural, la información de dicha casa rural y enlaces de contacto con el propietario o gestor.

También dispone de **sistemas de ordenamiento de búsquedas**, que incluyen un ordenamiento basado en el **precio/noche** del alojamiento. Aunque también se pueden encontrar otros.

El sistema provee a la cuenta que posee el perfil de gestor de alojamientos la opción de **publicar anuncios de alojamientos** a arrendar, en el cual puede poner los datos de dicho alojamiento.

La persona que entre con perfil de cliente o cómo anónimo, tiene la posibilidad de **acceder a los anuncios** de los alojamientos que se pueden alquilar e incluso se le da la opción de alquilarla, pero para ello, previamente el cliente tiene que estar registrado.

La persona que entre como **propietario o gestor** en la página web, tiene la posibilidad de **publicar los alojamientos** que quiera poner en alquiler, pudiendo modificar las publicaciones, añadirlas o incluso eliminarlas.

La persona que acceda a la página web como **administrador**, tiene la posibilidad de dar de baja a clientes o propietarios además.

Descripción del Sistema Nuevo:

Este sistema, se enfocará en un **nicho de alquileres rurales**. Esto quiere decir que toda la aplicación estará **ambientada** en entornos rurales (imágenes de casas rurales, montaña, tonos verdes...). Por lo tanto, este será un punto de diferenciación clave entre las apariencias del sistema actual y el sistema nuevo.

Se pretende que la interfaz de usuario sea lo más **sencilla** posible, para ello se llevará a cabo un desarrollo muy enfocado en la usabilidad de la aplicación, teniendo en cuenta que el objetivo principal de la mayoría de personas que consuman dicha aplicación será encontrar una **casa rentable** en cuanto a **comodidades/precio** para llevar a cabo un **alquiler vacacional**. De esta manera se podrá determinar qué elementos se deben añadir y en qué parte se deben posicionar. También, contaremos con un sistema que indique que clases de instalaciones y comodidades ofrece el alojamiento (nº baños, cocina, etc...).

El sistema contará con la posibilidad de registrarse como **gestor de alojamientos** o **cliente**, siendo los primeros los **propietarios** de las casas que podrán ser **alquiladas** y siendo los **segundos** los que las **podrán alquilar**. **Los gestores de alojamientos no podrán llevar a cabo ningún tipo de alquiler vacacional a los pisos que se consideren de su propiedad** (esto se hace así para evitar conflictos en cuánto a los alquileres). En caso de encontrarse registrado, accederás al sistema como **usuario anónimo**, teniendo la posibilidad de acceder a las opciones de alojamiento pero siendo incapaz de efectuar un alquiler.

Cómo **gestor de alojamientos**, se contará con una **sección única** de la aplicación (que solo será renderizada si el perfil usado es el de gestor) en la que se podrá **administrar las viviendas** que tiene ingresadas para que los clientes puedan alquilarlas. Desde este apartado, se podrá ocultar la visibilidad del alojamiento, editar la descripción, imágenes, precio, eliminarlas, etc...

Se hará uso de un **buscador con varios filtros** para llevar a cabo búsquedas de casas que estén disponibles para ser alquiladas. Se pretenden poner filtros como la **ubicación de la casa**, las **fechas** entre las que se quiere alquilar la casa, **número de personas** y el **precio máximo y mínimo**.

El sistema también contará con una pantalla de inicio desde la que se podrán visualizar las opciones mejor valoradas de alojamientos. Además del buscador anteriormente mencionado y otra información relevante.

Identificación de Requisitos del Sistema:

Requisitos de información:

Para llevar a cabo la recopilación de requisitos de información se tendrán en cuenta las relaciones entre las distintas **entidades** que se presentarán en este proyecto.

La base de datos se generará de manera **automática** con el framework de Spring (que compone gran parte del backend de la aplicación). Esto quiere decir que gracias a la configuración que se le aporta, la aplicación es capaz de **generar tablas** y campos con solo levantar el backend. Lo que quiere decir que la lista de campos será tomada directamente de la lista de atributos que tendrán las clases mapeadas de JPA en la parte backend de la aplicación, **aunque en el diagrama entidad/relación se podrá apreciar con mayor detalle el tipo de dato que se almacenará en BBDD junto con sus restricciones.**

Tabla de correspondencia de tipos de datos:

TIPO EN JAVA	TIPO EN MYSQL SERVER
boolean, java.lang.Boolean	TINYINT(1)
int, java.lang.Integer	INTEGER
long, java.lang.Long	BIGINT
float, java.lang.Float	FLOAT
double, java.lang.Double	DOUBLE
short, java.lang.Short	SMALLINT
byte, java.lang.Byte	SMALLINT
java.lang.Number	DECIMAL(38)
java.math.BigInteger	BIGINT
java.math.BigDecimal	DECIMAL(38)
java.lang.String	VARCHAR(255)
char, java.lang.Character	CHAR(1)
byte[], java.lang.Byte[], java.sql.Blob	BLOB(64000)
char[], java.lang.Character[], java.sql.Clob	TEXT(64000)
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	DATETIME

Alojamiento:

- **Descripción:**

Esta entidad guardará toda la información referente a los alojamientos que se encuentran registrados en la aplicación web.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
txtNombre	String	No
txtDescripcion	String	No
numPlazaMin	Integer	No
numPlazaMax	Integer	No
numPrecioPlaza	Double	No
idAlquileresAlojamiento	Set<AlquilerAlojamiento>	Si
idUsuario	Usuario	No
idImagenesAlojamiento	Set<ImagenAlojamiento>	No
idValoracionesAlojamiento	Set<ValoracionAlojamiento>	Si
idUbicacion	Ubicacion	No
idAlojamientoComodidades	Set<AlojamientoComodidad Alojamiento>	Si

- **Otras observaciones:**

1. Esta entidad es la principal de toda la aplicación.
2. La tabla en BBDD de esta entidad es la que más relaciones tiene con las otras.
3. La asociación de esta tabla con la tabla de usuarios se realiza mediante la tabla de alquileres.

Perfil:

- **Descripción:**

Esta entidad guardará toda la información referente a los perfiles de los usuarios de la página web.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
codPerfil	String	No
txtPerfil	String	No
idUsuarios	Set<Usuario>	Si

- **Otras observaciones:**

1. La función principal de esta entidad será determinar los permisos de los usuarios dentro de la aplicación. Los perfiles de la aplicación serán los que determinen si el usuario es cliente, gestor o administrador.

AlojamientoComodidadAlojamiento:

- **Descripción:**

Esta entidad guardará toda la información referente a las relaciones existentes entre la entidad de Alojamiento y ComodidadAlojamiento.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
idComodidadAlojamiento	ComodidadAlojamiento	No
idAlojamiento	Alojamiento	No

- **Otras observaciones:**

1. Esta tabla es producto de la relación (N,M) entre las entidades de Alojamiento y ComodidadAlojamiento.

Usuario:

- **Descripción:**

Esta entidad guardará toda la información referente a los usuarios de la aplicación web, es indiferente del rol que posea el usuario.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
txtNombreUsuario	String	No
txtDescripcion	String	Si
txtDni	String	No
numTelefono	Integer	No
txtEmail	String	No
txtPassword	String	No
datosImagenUsuario	Byte[]	Si
idPerfil	Perfil	No
idAlquileres	Set<AlquilerAlojamiento>	Si
idAlojamientos	Set<Alojamiento>	Si
idValoracionesAlojamientos	Set<ValoracionAlojamiento>	Si

- **Otras observaciones:**

1. En esta tabla se podrán encontrar todos los usuarios, desde aquellos que poseen permisos de administración hasta aquellos que son meros clientes. Todo es distinguido a través del perfil asignado a cada usuario.

AlquilerAlojamiento:

- **Descripción:**

Esta entidad guardará toda la información referente a los alquileres de alojamientos, guardará tanto el alojamiento cómo el cliente que alquila el alojamiento.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
fechaInicioAlquiler	LocalDate	No
fechaFinAlquiler	LocalDate	No
idUsuario	Usuario	No
idAlojamiento	Alojamiento	No
precioTotalAlquiler	Double	No
numPlazasReservadas	Integer	No

- **Otras observaciones:**

1. El usuario gestor del alojamiento ya es un atributo del alojamiento alquilado, por lo que no es necesario incluirlo explícitamente en esta entidad.

ImagenAlojamiento:

- **Descripción:**

Esta entidad almacenará una serie de imágenes asociadas a los alojamientos.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
datosImagen	Byte[]	No
idAlojamiento	Alojamiento	No
numOrden	Integer	No

- **Otras observaciones:**

1. El atributo "numOrden" indicará el orden dentro de la lista de imágenes que pueda tener un alojamiento.

ComodidadAlojamiento:

- **Descripción:**

Esta entidad almacena una serie de comodidades que se pueden referenciar en los alojamientos. Pueden abarcar desde instalaciones interiores hasta actividades que se pueden realizar.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
codigoComodidad	String	No
txtNombre	String	No
txtDescripcion	String	No
idAlojamientoComodidades	Set<AlojamientoComodidad Alojamiento>	Si
idTipoComodidad	TipoComodidad	No

- **Otras observaciones:**

1. Se pretende que los datos de esta entidad se carguen en el front como una lista de comodidades que le puedes añadir cómo atributo a los alojamientos.

TipoComodidad:

- **Descripción:**

Esta entidad almacena los distintos tipos de comodidades, desde comodidad del tipo instalación hasta comodidad del tipo actividad.

- **Lista de campos:** [nombre_campo | tipo_dato]:

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
codigoTipoComodidad	String	No
txtNombre	String	No
idComodidadAlojamientos	Set<ComodidadAlojamiento>	Si

- **Otras observaciones:**

1. La función principal de esta tabla será diferenciar cuál es el tipo de comodidad que estaremos guardando. De esta manera, nos ahorramos crear tablas independientes para instalaciones exteriores, instalaciones interiores, actividades, paisaje... Entre otros detalles que se pueden añadir como items al anuncio de un alojamiento

UbicacionAlojamiento:

- **Descripción:**

Esta entidad almacena la ubicación del alojamiento.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
idAlojamiento	Alojamiento	No
codigoPostal	Integer	No
ciudad	String	No
provincia	String	No
lineaDireccion	String	No
longitud	String	Si
latitud	String	Si

- **Otras observaciones:**

1. Los valores de “longitud” y “latitud” son opcionales, los gestores pueden ubicar su alojamiento en un mapa que se pueda visualizar por los usuarios. Para ello, estos dos valores son determinantes.

ValoracionAlojamiento:

- **Descripción:**

Esta entidad almacena la información referente a las distintas valoraciones que puede tener un alojamiento.

- **Lista de campos:**

NOMBRE ATRIBUTO	TIPO	OPCIONAL
id	Long	No
idAlojamiento	Alojamiento	No
txtMensaje	String	No
txtAsunto	String	No
idUsuario	Usuario	No
puntuacion	Integer	No

- **Otras observaciones:**

1. Se ha decidido optar por una valoración con un formato asunto-autor-puntuación-mensaje dónde un usuario podrá escribir y puntuar su experiencia en las casas rurales.

Requisitos funcionales:

Para llevar a cabo los requisitos funcionales, se seguirán una serie de criterios los cuáles se verán a continuación:

Registro de usuarios:

Los usuarios podrán registrarse o bien como clientes o bien como gestor de alojamientos, siendo necesario especificar dicho perfil. Para ello se habrá de proporcionar ciertos datos como el correo electrónico, la contraseña, etc...

Inicio de sesión:

Los usuarios podrán iniciar sesión introduciendo su correo electrónico y su contraseña. De este modo, los clientes podrán efectuar sus alquileres vacacionales y los gestores podrán poner en alquiler sus viviendas. A su vez, si no se inicia sesión en la aplicación, se podrá únicamente visualizar el contenido de la aplicación (anuncios, ofertas...) pero no consumirlo.

Gestión de casas rurales:

Los gestores tendrán permisos para agregar nuevas casas rurales disponibles, junto con sus características, ubicación, precio o disponibilidad... Estos, también tendrán permisos para editar y suprimir casas rurales existentes siempre y cuándo sean de su propiedad.

Gestión de reservas:

Una vez el cliente haya realizado un alquiler, se le debe de enviar una confirmación de la reserva al cliente por correo electrónico junto con la factura.

En cuanto al gestor de alojamientos, a este se le notificará, a la hora en la que un usuario haya realizado una reserva sobre algún alojamiento en posesión del gestor. Además, los gestores podrán decidir si aceptar o no la reserva de este alojamiento por parte del cliente, incluyendo la capacidad de poder cancelar cualquier reserva que no se encuentre activa y que tenga una fecha a futuro.

Los administradores, podrán ver una lista de todas las reservas realizadas, incluyendo los detalles de dichos alquileres, tanto por parte del cliente como por parte del gestor de la casa rural, teniendo la opción de cancelar reservas existentes en el caso en el que sea necesario.

Gestión de clientes:

Los gestores podrán ver una lista de todos los alquileres que han sido efectuados sobre sus viviendas. Pudiendo de esta manera obtener los datos de usuario público de esos clientes.

En cuanto a los administradores, estos podrán ver la lista completa de los clientes que se encuentran almacenados en la aplicación, y podrán modificar cada uno de sus atributos.

Búsqueda y filtro de las casa rurales:

Los clientes podrán buscar casas rurales y se les dará la opción de establecer un periodo de tiempo específico, un rango de precio detallado, características de la vivienda y ubicación del alojamiento buscado. Además, los clientes podrán ver las características de estas casas rurales junto con las imágenes que el propietario haya subido.

Notificaciones:

Los clientes recibirán aviso a través de correo electrónico del alquiler de una casa rural o de la modificación de dicho alquiler.

Los gestores recibirán un aviso a través de correo electrónico cuando algún cliente trate de llevar a cabo la reserva de un alojamiento.

Otros Requisitos:

Control de versiones:

Cómo tecnología de control de versiones se usará Git, toda la aplicación y la información referente a esta se encontrará disponible en un repositorio online de Github.

De esta forma, se podrá trabajar de manera colaborativa, además de permitirnos el poder llevar a cabo el desarrollo de funcionalidades independientes de una manera más cómoda. Así, nos aseguramos de que la traza de desarrollo de la aplicación sea visible, se pueda apreciar y se encuentre en un lugar seguro y escalable.

Portabilidad de la aplicación:

La aplicación contará con un soporte para ser desplegada en contenedores docker. Esto quiere decir que el despliegue de la aplicación será notoriamente similar desde cualquier sistema operativo, siendo el único requisito tener docker instalado.

Balanceo de carga:

La aplicación contará con una tecnología que aumentará considerablemente el rendimiento de esta misma. En este caso, se usará Kubernetes para orquestar los distintos contenedores en los que la aplicación se encuentra desplegada.

Seguridad de la aplicación:

En este caso, hemos decidido usar Spring Security para tener un control de usuarios y de perfiles. De esta manera, cada usuario tendrá su propio token de autenticación que determinará los permisos que tiene dentro del sistema.

Descripción de la solución:

Para llevar a cabo un desarrollo óptimo y eficaz de la aplicación se realizarán las siguientes tareas:

1. Llevar a cabo un análisis completo de la aplicación a realizar. Esto incluye:
 - Buscar sistemas similares y comprobar la manera en que funcionan.
 - Plantear un sistema propio basado en el de la competencia.
 - Determinar toda la información que será necesaria almacenar.
 - Determinar una serie de funcionalidades que debe de realizar el sistema.
 - Llevar a cabo un análisis del tiempo que tomará completar el desarrollo de la aplicación.
 - Llevar a cabo un análisis de costes basándose en el tiempo de desarrollo y el puesto laboral de cada integrante juntos con sus horas empleadas en realizar la tarea.
 - Analizar el software requerido para construir la aplicación.
 - Realizar un diagrama que represente la modelización de los datos almacenados para estructurar una base de datos.
 - Realizar un análisis de usuarios participantes y finales.
 - Llevar a cabo diagramas de casos de uso de la aplicación para saber a qué decisiones se enfrentarán los usuarios y cómo estructurar los elementos de la aplicación para que estas decisiones se resuelvan de la manera más sencilla y usable posible.
 - Especificar el diseño, estructuración y navegación de la página
2. Una vez realizado el análisis y teniendo claro cuál es el objetivo, los medios y las herramientas que se emplearán para alcanzar el mismo. Se procederá a maquetar un prototipo de la página que se visualizará en el lado del cliente. Para esto, vamos a usar una herramienta llamada **Figma**. De esta manera, podremos establecer un diseño desde el que partir, para así no encontrarnos con ningún imprevisto a la hora de desarrollar el Front de la aplicación
3. Una vez realizado el prototipo de la página, se procederá a crear la estructura básica de las entidades mapeadas de JPA en el backend. Dicho de otra manera, representaremos a través de clases Java la estructura, relaciones y restricciones de la base de datos haciendo uso del framework de Spring Data y basándonos en el diagrama entidad/relación creado en la fase de análisis. A través de la configuración aportada en el backend, la propia aplicación creará una base de datos directamente a la hora de ser desplegada, por lo que no será necesario la creación de un script adicional (aunque de todas formas lo habrá).
4. Una vez terminada la estructura básica del backend, el prototipo del frontend en Figma y teniendo una base de datos que se ha generado automáticamente por el despliegue de la aplicación de Spring. Se procederá a realizar un desarrollo paralelo del frontend y el backend. Se tendrá un control de los cambios realizados en esta etapa desde Github, ya que se usará la tecnología de Git para controlar las

versiones y cambios realizados en la aplicación. Cabe recalcar que el frontend de la aplicación se desarrollará usando el framework de Angular, teniendo una orientación basada en componentes y se hará uso de la librería de Bootstrap para poder estructurar la página de manera correcta y que se pueda visualizar bien desde cualquier dispositivo.

5. Cuando se termine de desarrollar por completo el sistema, se procederá a probar todas y cada una de las funcionalidades creadas. Para, de esta manera, poder asegurarnos que cada componente de la aplicación se comunica bien con el resto y de la manera que se desea.
6. Una vez terminada completamente la aplicación y sus pruebas, se procederá a usar la tecnología de Docker para desplegar las aplicaciones en distintos contenedores. De esta manera, se dotará al sistema de una mayor portabilidad y escalabilidad.
7. Cuando las configuraciones de Docker se encuentren finalizadas, se realizará una configuración con Kubernetes para poder manejar el balanceo de carga de la aplicación. De esta manera, nos aseguraremos de tener varios servidores funcionando con la misma aplicación que puedan distribuir la carga de peticiones.
8. Por último, se llevará a cabo un análisis de accesibilidad y usabilidad de aplicación final. Concluyendo de esta manera con el desarrollo de la aplicación.

Planificación del proyecto:

Equipo de trabajo:

Diseñador Web:

Nombre del puesto de trabajo:

Diseñador Web.

Descripción de puesto de trabajo:

El diseñador web se encarga de dar vida al aspecto visual de los sitios web. Su trabajo consiste en crear la interfaz de usuario (UI) y la experiencia de usuario (UX). Esto implica asegurarse de que el sitio web sea atractivo, práctico, intuitivo y agradable para los visitantes. Algunas tareas específicas de este puesto son: Crear ideas de diseño y maquetas, escoger colores, fuentes, imágenes y otros elementos visuales, determinar la navegación y la ubicación de los elementos en la página, optimizar el diseño para que sea accesible y funcione bien en diversos dispositivos y navegadores y colaborar con desarrolladores web para implementar el diseño.

Requisitos del puesto:

- Certificación de cursos de diseño.
- Tener un título de Ciclo de Grado Superior en diseño o relacionados.
- Tener un año de experiencia laboral como mínimo.
- Tener experiencia con Figma o herramientas de desarrollo de wireframes y prototipos de páginas web.

Aptitudes Valorables:

- Creatividad.
- Pensamiento Crítico.
- Inteligencia emocional.
- Proactividad.
- Conocimiento de software y aplicaciones informáticas.

Sueldo mensual:

El sueldo mensual promedio del diseñador web será de unos 1500€/mes por trabajar un total de 40 horas semanales.

Trabajadores que ocupan este puesto:

- Juan Martín Candela
- David Dorante Lucas

Desarrollador Web:

Nombre del puesto de trabajo:

Desarrollador Web.

Descripción de puesto de trabajo:

El desarrollador web se encarga de codificar todos los aspectos relacionados con una aplicación web. En este caso, podremos encontrarnos una diferencia entre aquellos que se dedican a programar el aspecto visual y funcional del lado del cliente y aquellos que se dedican a desarrollar código para ser ejecutado en un servidor. Deben de trabajar en equipo y colaborar para que todas las partes de la aplicación se comuniquen de manera correcta.

Requisitos del puesto:

- Tener conocimientos de lenguajes como Javascript y Java.
- Tener conocimientos en el uso del framework de Spring.
- Tener conocimientos en el uso de Angular.
- Tener conocimientos en el uso de HTML y CSS.
- Tener un título de Ciclo de Grado Superior en Desarrollo de Aplicaciones Web o similar.

Aptitudes Valorables:

- Resolutivo.
- Resolución de problemas.
- Cooperativo.
- Pensamiento Crítico.
- Proactividad.
- Conocimiento de software y aplicaciones informáticas.

Sueldo mensual:

El sueldo mensual promedio del desarrollador web será de unos 1300€/mes por trabajar un total de 40 horas semanales.

Trabajadores que ocupan este puesto:

- Juan Martín Candela
- David Dorante Lucas

Jefe de Proyecto:

Nombre del puesto de trabajo:

Jefe de Proyecto.

Descripción de puesto de trabajo:

El Jefe de Proyecto se encarga de coordinar a un equipo de trabajo. De cara al producto final, él es quien se encarga de escoger qué tecnologías serán usadas para el desarrollo de una aplicación. También se encarga de hablar con los clientes finales de dicha aplicación y presentarles el progreso del proyecto.

Requisitos del puesto:

- Conocimientos de herramientas de gestión de proyectos
- Experiencia técnica y laboral dentro del puesto de trabajo de desarrollador muy avanzadas.
- Habilidades comunicativas efectivas.
- Tener un título de Ciclo de Grado Superior en Desarrollo de Aplicaciones Web o similar o un título de grado universitario.

Aptitudes Valorables:

- Análisis de datos.
- Capacidad de adaptación
- Liderazgo.
- Resolución de problemas.
- Cooperativo.
- Pensamiento Crítico.
- Proactividad.
- Conocimiento de software y aplicaciones informáticas.

Sueldo mensual:

El sueldo mensual promedio del desarrollador web será de unos 2000€/mes por trabajar un total de 40 horas semanales.

Trabajadores que ocupan este puesto:

- Juan Martín Candela
- David Dorante Lucas

Planificación temporal:

En cuanto a la planificación temporal del proyecto, se prevé que el tiempo total invertido en proyecto sea de unos tres meses, contando con un margen adicional para posibles retrasos o contratiempos que se den durante su producción y desarrollo.

Fase 1: Preparación y planificación.

Esta fase abarca las dos primeras semanas de trabajo y tendrá una cantidad de dos periodos en el que se llevarán a cabo diferentes tareas.

Periodo 1:

Tendrá lugar los primeros 3 días.

Tareas:

- Se llevarán a cabo revisiones exhaustivas y un análisis detallado de los requisitos del proyecto, para comprender al completo las necesidades del cliente.
- Se programan reuniones con el cliente que sirven para aclarar las expectativas del mismo y establecer objetivos claros.

Responsables:

Los responsables de llevar a cabo estas tareas son los Jefes de Proyecto.

Periodo 2:

Tendrá lugar los días restantes próximos a los días del *periodo 1*.

Tareas:

- Se procede a la selección de tecnologías y herramientas adecuadas para llevar a cabo el desarrollo del proyecto, teniendo en cuenta los requisitos de los clientes y las capacidades que presenta el equipo.
- Se procede a hacer una lista de objetivos que tendrán que ser cumplidos en un periodo de tiempo preestablecido.
- Se realiza un reparto de todas las tareas y de qué personas serán responsables de cada parte de la aplicación.
- Se realizan diagramas de casos de uso, entidad/relación, etc... Todo esto con el objeto de clarificar los pasos a seguir para llegar al producto final.

Responsables:

Los responsables de llevar a cabo este conjunto de tareas son los Jefes de Proyectos, ya que son los capacitados para desempeñar este tipo de tareas en ambos periodos.

Fase 2: Desarrollo.

En esta fase, se llevará a cabo la codificación y desarrollo de la aplicación, tanto del aspecto visual orientado a la parte que con la que va a interactuar el cliente, como en la parte que se ejecutará en el servidor. Además, tendrá una duración de unos 2 meses.

Tareas:

- El Diseñador Web se enfocará en el diseño de la interfaz de usuario (UI) y la experiencia de usuario (UX) para garantizar un diseño lo más atractivo y funcional posible, llevará a cabo la realización de Wireframes y prototipos de la página. Así como la navegación en la misma.
- El Desarrollador Web se encargará de codificar el apartado visual, es decir, el frontend. Para ello se hará uso de herramientas como el framework de Angular.
- El Desarrollador Web también es el responsable de codificar el backend de la aplicación web, utilizando el framework de Spring Boot para garantizar que el funcionamiento de la aplicación sea robusto y eficiente.
- Los Jefes de Proyecto, seguirán teniendo reuniones con el cliente para mostrar cómo va avanzando la aplicación y para actualizar cambios, en el caso del que cliente desee modificar o agregar cambios al producto final.

Responsables:

Los responsables para llevar a cabo esta fase son:

- Diseñadores Web.
- Desarrolladores Web.
- Jefes de Proyectos.

Fase 3: Pruebas y Depuración.

En esta fase se llevarán a cabo los procesos correspondientes para la depuración y pruebas del proyecto, comprobando el correcto funcionamiento de las funcionalidades de la aplicación. Este proceso, conlleva una duración de 1 semana.

Tareas:

- Se llevarán a cabo todo tipos de pruebas exhaustivas, sobre todo, pruebas de seguridad, funcionamiento y rendimiento para garantizar la calidad y fiabilidad de la aplicación.
- Identificación y corrección de cualquier error o fallo de funcionamiento encontrado durante las pruebas que se realizan en este periodo de pruebas.

Responsables:

Los encargados para llevar a cabo esta fase es el equipo de desarrollo al completo:

- Jefes de Proyectos.
- Diseñadores Web.
- Desarrolladores Web.

Fase 4: Despliegue y puesta en marcha.

En esta fase se llevará a cabo el despliegue de la aplicación y se comprobará que funciona todo los elementos que componen la aplicación correctamente. Para esta fase, se estima una duración de 1 semana.

Tareas:

- Se procede al despliegue de la aplicación web en un entorno de producción, asegurando que la aplicación tenga una configuración adecuada de servidores y una configuración adecuada en la base de datos.
- Se llevan a cabo pruebas finales en el entorno de producción para garantizar que el lanzamiento del producto final sea exitoso.

Responsables:

Los responsables para llevar a cabo esta fase es el equipo de desarrollo al completo:

- Jefes de proyectos.
- Diseñadores Web.
- Desarrolladores Web.

Fase 5: Mantenimiento y Soporte.

Esta fase tendrá lugar tras la finalización del proyecto, para seguir mejorando la aplicación y seguir actualizando a petición del cliente. Esta fase se realizará de forma continua tras finalizar con el producto final.

Tareas:

- Se realizan actualizaciones de forma periódica para mejorar funcionalidades y corregir errores, asegurando así que la aplicación tenga un rendimiento óptimo.
- Se proporciona soporte técnico continuo para resolver los problemas que surjan y atender solicitudes de los usuarios de manera continua.

Responsables:

Esta fase la llevará a cabo todo el equipo de desarrollo:

- Jefes de Proyectos.
- Diseñadores Web.
- Desarrolladores Web.

Estudio del coste del proyecto:

El estudio del proyecto implica una evaluación detallada de todos los recursos financieros necesarios para llevar a cabo todas las actividades planificadas y para alcanzar los objetivos establecidos. A continuación, se presentan una serie grupos cada uno con sus respectivos costes:

Costes de Personal.

Antes de comenzar a exponer el coste total del proyecto, se ha de recalcar que toda la información expuesta queda recogida en el documento oficial del BOE:

https://www.boe.es/diario_boe/txt.php?id=BOE-A-2023-17238

Concretamente la **tabla salarial** se encuentra en el **Anexo I**. Los costes del personal han sido calculados teniendo en cuenta **el grupo profesional y los plus convenios** además de los **extras salariales pertinentes**, recogidos en los **artículos 15, 27 y 28**. Además, en la tabla salarial se ha tenido en cuenta el salario total a partir de la **fecha del 01-01-2024**.

Dentro de este grupo se incluyen los salarios y beneficios del equipo de trabajo que se dividen a su vez en tres grupos:

	JEFE DE PROYECTO	DISEÑADOR WEB	DESARROLLADOR WEB
Horas semanales	40 horas	40 horas	40 horas
Salario Mensual (Bruto; Extras añadidos)	2057.30€	1362.66€	1362.66€
Duración del contrato temporal	3 meses	1 mes	3 meses
Número de empleados	2	2	2
Grupo perteneciente	Área 3 - Grupo A.1	Área 3 - Grupo D.1	Área 3 - Grupo D.1

COSTE TOTAL DEL PROYECTO:	23245.08 €
----------------------------------	-------------------

ANÁLISIS DEL SISTEMA DE INFORMACIÓN

Identificación del entorno tecnológico:

Para el diseño, la implementación y el funcionamiento exitoso de la aplicación web de gestión de reservas de casas rurales, es fundamental la identificación del entorno tecnológico, tanto de desarrollo como de explotación del sistema.

A continuación, vamos a mostrar más en detalle cómo será el entorno tecnológico en ambas fases:

Entorno Tecnológico de Desarrollo.

Lenguajes de Programación:

Para el desarrollo de la aplicación, vamos a usar una serie de lenguajes de programación, los cuales vamos a detallar a continuación:

Frontend:

Para la parte del frontend, vamos a usar dos lenguajes de marca: “**HTML**” y “**CSS**” y un lenguaje de programación: “**Javascript**”, los cuales nos van a permitir crear la página web que el cliente verá. Para ello, usaremos HTML para crear los elementos a mostrar en la página web, CSS para aplicar estilos a los elementos que creemos con HTML y por último, usaremos Javascript, para darle funcionalidad a aquellos elementos que hayamos creado previamente, usando como framework para aplicar todos estos lenguajes, **Angular**. De la misma manera, al hacer uso de **Angular** estaremos proveyendo a la aplicación una **orientación modularizada**.

Backend:

Para la parte del backend, vamos a usar un solo lenguaje el cual nos va a permitir llevar a cabo el desarrollo de la parte del servidor. Para ello, usaremos “**Java**” para llevar a cabo el desarrollo del backend. Además, estaremos haciendo uso del framework de “**Spring**”, en este caso, para facilitar procesos como el mapeo de objetos, creación de repositorios para las entidades, servicios, controladores, seguridad de la aplicación, despliegue, etc...

Frameworks y Bibliotecas:

Son el conjunto de herramientas, bibliotecas y convenciones que usaremos para llevar a cabo el desarrollo de la aplicación, ofreciendo una base sólida y reutilizable, proporcionando estructuras, patrones y componentes comunes.

Angular:

Usamos **Angular** para llevar a cabo el desarrollo del frontend de la aplicación web, permitiendo la creación de interfaces de usuario para que sean dinámicas y altamente interactivas. Además, este framework hace que la aplicación sea exponencialmente más mantenible y escalable, aportando un enfoque orientado a módulos.

Spring:

Usamos **Spring** para llevar a cabo el desarrollo del backend de la aplicación web, proporcionando un entorno de desarrollo que sea más ágil y eficiente. Además, **Spring** proporciona múltiples frameworks que nos ayudarán a la hora de conectar la aplicación con la base de datos, el frontend, el despliegue y para aportar seguridad a la aplicación. Para desarrollar la parte del backend con el framework de **Spring**, vamos a utilizar el IDE de **Spring Tools Suite 4**.

Base de Datos:

En nuestra aplicación web, necesitamos de una base de datos, para poder guardar la información de los clientes, la información de las casas rurales y de las distintas reservas que se realicen. Para ello vamos a usar **MYSQL**, usando **DBEAVER**, para comprobar que los datos se manipulan de forma correcta.

Herramientas de Desarrollo:

Son las herramientas que vamos a usar para llevar a cabo el desarrollo de la aplicación, y son las siguientes, dividiéndolas según sus características:

Entorno de Desarrollo Integrado:

Vamos a usar **Spring Tool Suite 4** para el desarrollo de la parte del backend de la aplicación web, y **Visual Studio Code** para el desarrollo del frontend de la aplicación de reservas de casas rurales.

Herramientas de control de versiones:

Vamos a usar **GIT** y **GITHUB** para el control de versiones del código fuente, y guardar los cambios que se produzcan en la nube, de forma que todos los miembros del grupo de trabajo puedan tener el código actualizado.

Servidores de aplicaciones:

Son los servidores que usaremos para ejecutar y desplegar la aplicación web en el entorno de desarrollo local, usando en nuestro caso, **Apache Tomcat**.

Entorno Tecnológico de Explotación.

Para el entorno tecnológico de **Explotación** de la aplicación web del alquiler de casas rurales, vamos a usar **Docker**, obteniendo una mayor flexibilidad, portabilidad y eficiencia en la gestión y el despliegue de la aplicación, facilitando así su mantenimiento y escalabilidad en un entorno de producción.

Contenedores Docker:

Empaquetamos la aplicación web en contenedores **Docker** para garantizar la portabilidad y la consistencia del entorno de ejecución.

Imágenes Docker:

Para el frontend y el backend crearemos imágenes **Docker**. Estas imágenes van a contener todos los archivos y dependencias necesarias para ejecutar la aplicación de manera aislada.

Orquestador de Contenedores y balanceo de carga:

Gestionamos y orquestamos contenedores Docker en un entorno de producción usando **kubernetes**. Estas herramientas facilitan la administración y el despliegue de la aplicación en múltiples y diversos contenedores, además de garantizar una comunicación efectiva entre todos los contenedores.

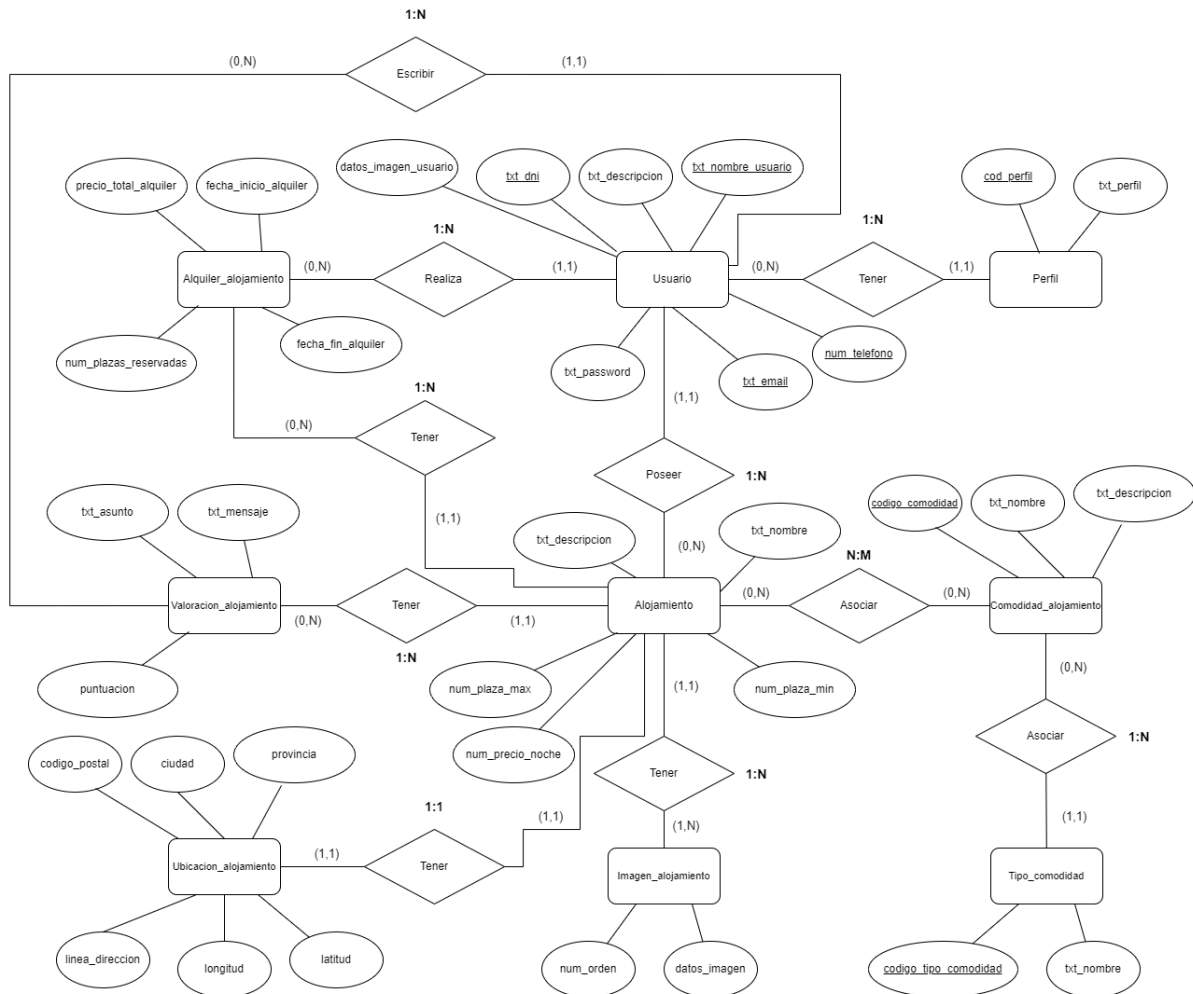
Seguridad:

Se aplicarán las mejores prácticas de seguridad para contenedores Docker, como el uso de imágenes oficiales, el escaneo de vulnerabilidades y la gestión adecuada de credenciales, junto con permisos de accesos.

Además de todas estas medidas de seguridad, la propia aplicación también tendrá su propia capa de seguridad aportada por **Spring Security**.

Modelado de datos:

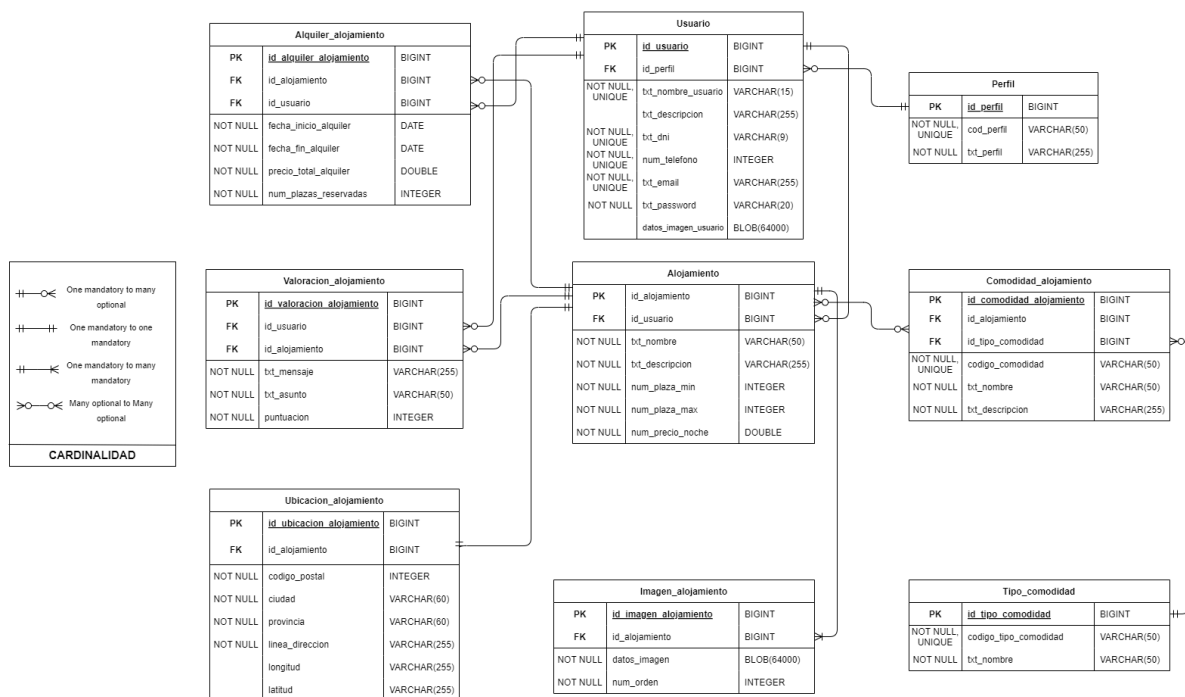
Modelo Entidad Relación y esquema de base de datos:



A continuación, se aportarán links para poder descargar y visualizar los archivos del modelo de entidad relación y el esquema completo de la Base de Datos:

https://github.com/jmarcan286/portal_del_viajero-PI/tree/develop/public-resources

Modelo Relacional:



Identificación de los usuarios participantes y finales:

Para la gestión de reservas de casas rurales en el contexto de la aplicación web, es importante identificar a los usuarios participantes y finales que interactúan con la aplicación. A continuación se mostrará una identificación y explicación de cada uno de ellos, crucial para poder entender cómo interactúan con el sistema y qué funciones realizan:

Usuarios Participantes.

Los usuarios participantes son aquellos que participan en la aplicación manteniendo la aplicación en funcionamiento.

Administrador del Sistema:

- Estos usuarios son responsables de administrar y mantener el sistema en funcionamiento.
- Se encargan de realizar funciones, como incluir la gestión de las propiedades disponibles, la atención a las reservas, la actualización de la disponibilidad o también se encargan de incluir la gestión de los clientes.
- Además, interactúan directamente con la interfaz de administración del sistema.

Usuarios Finales.

Los usuarios finales son aquellos que interactúan directamente con la página web, pudiendo realizar reservas o obtener información sobre las casas rurales y sus propietarios, para poder contactar con ellos.

Gestores de Casas Rurales:

- Estos usuarios son los propietarios de las casas rurales que se encargan de ponerlas en alquiler.
- Se encargan de realizar funciones, como gestionar las casas rurales o propiedades disponibles, prestar atención a las reservas, actualizar la disponibilidad o no de las casas rurales y se encargan también de la gestión de los clientes, con los cuáles podrán llegar a un acuerdo o no para el alquiler.
- Además, interactúan con la interfaz de administración para gestionar las casas rurales que hay en la página web y las reservas.

Clientes:

- Los clientes son aquellos usuarios finales que desean alquilar una casa rural que se encuentre dentro de sus requisitos a través de nuestra página web.
- Se encargan de realizar funciones, como realizar una búsqueda casas rurales que estén disponibles, realizar reservas, actualizar la información personal y visualizar el historial de reservas.
- Además, interactúan con la interfaz pública de la aplicación web para buscar, seleccionar y reservar casas rurales.

Otros sistemas.

Cuando hablamos de otros sistemas, hacemos referencia a la utilización de herramientas que no funcionan dentro de la aplicación, es decir, que su sistema por el cual funciona es externo.

Sistema de Pago en Línea:

- Es un sistema externo el cual proporciona servicios de procesamiento de pagos en línea.
- Además, la aplicación web puede integrarse con este sistema para permitir a los clientes realizar pagos de reservas utilizando tarjetas de crédito, transferencias bancarias u otros métodos de pago en línea.

Servicios de Geolocalización:

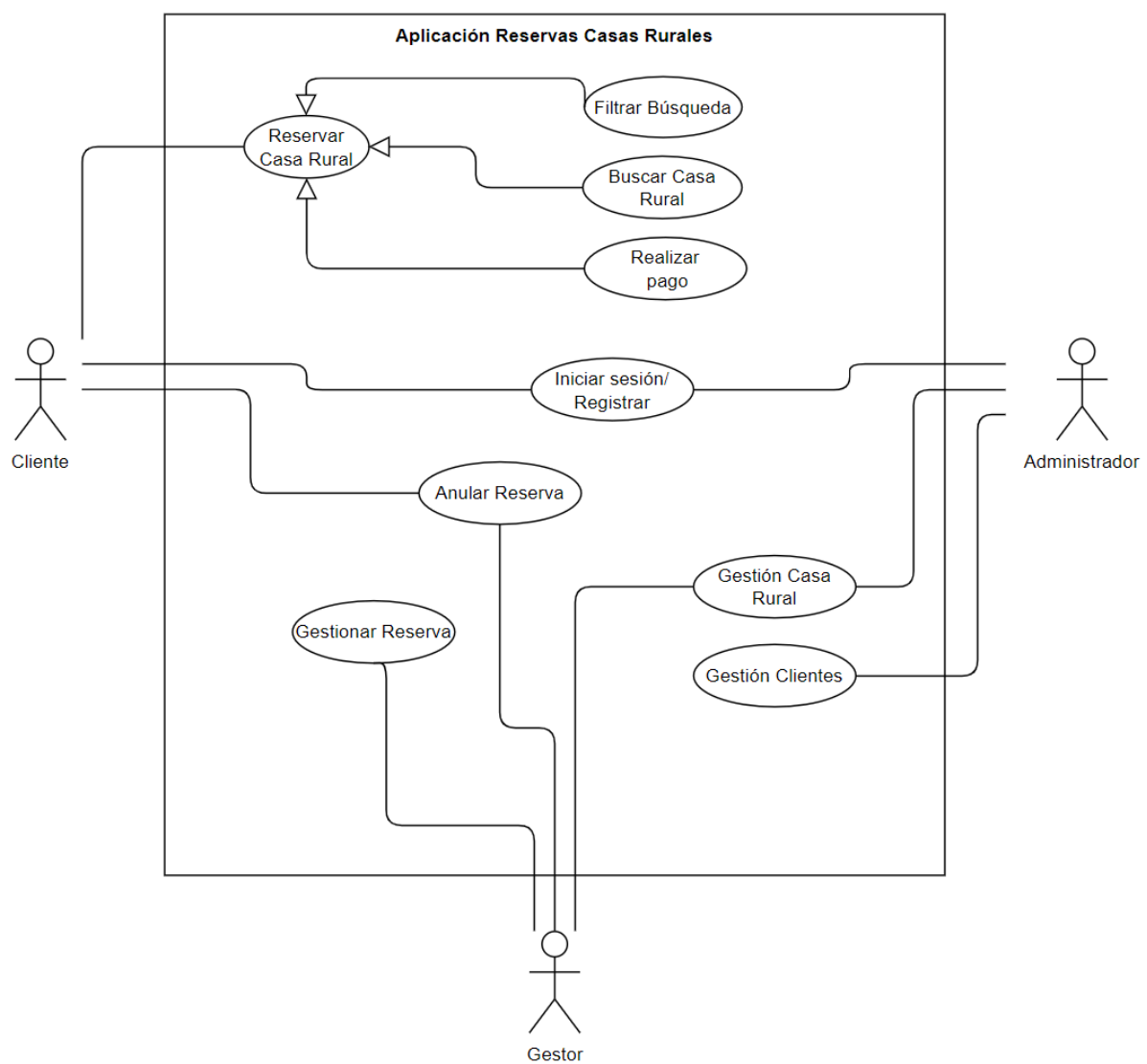
- Los servicios de geolocalización pueden ser utilizados para proporcionar funcionalidades de búsqueda que estén basadas en una ubicación especificada, permitiendo a los clientes encontrar casas rurales que estén cerca de ellos, o en la ubicación que los clientes especifiquen.
- La aplicación web podría integrarse con servicios como por ejemplo Google Maps para mostrar la ubicación de las casas rurales y proporcionar indicaciones de cómo llegar a dichas casas rurales.

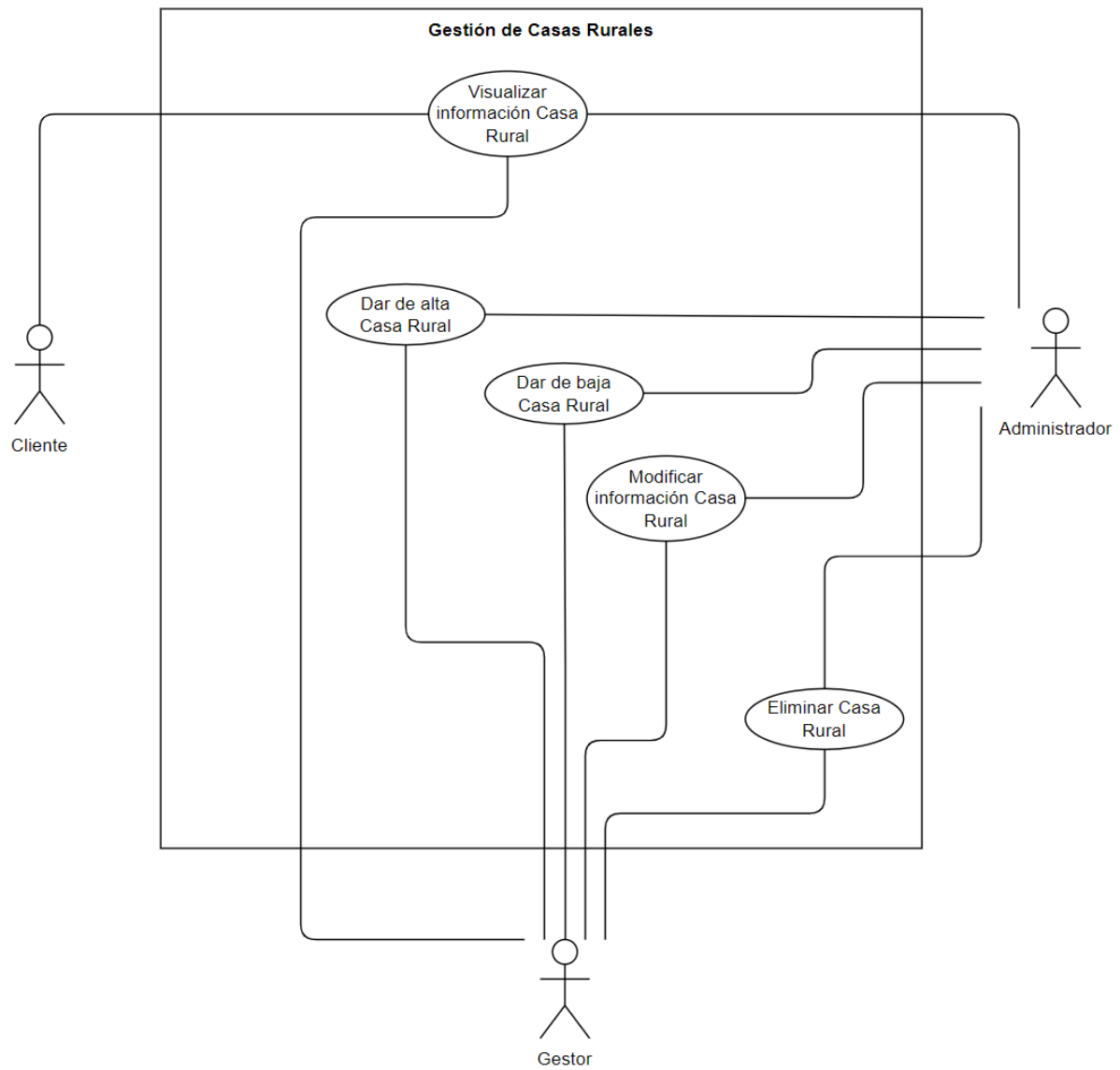
Plataforma de Correo Electrónico:

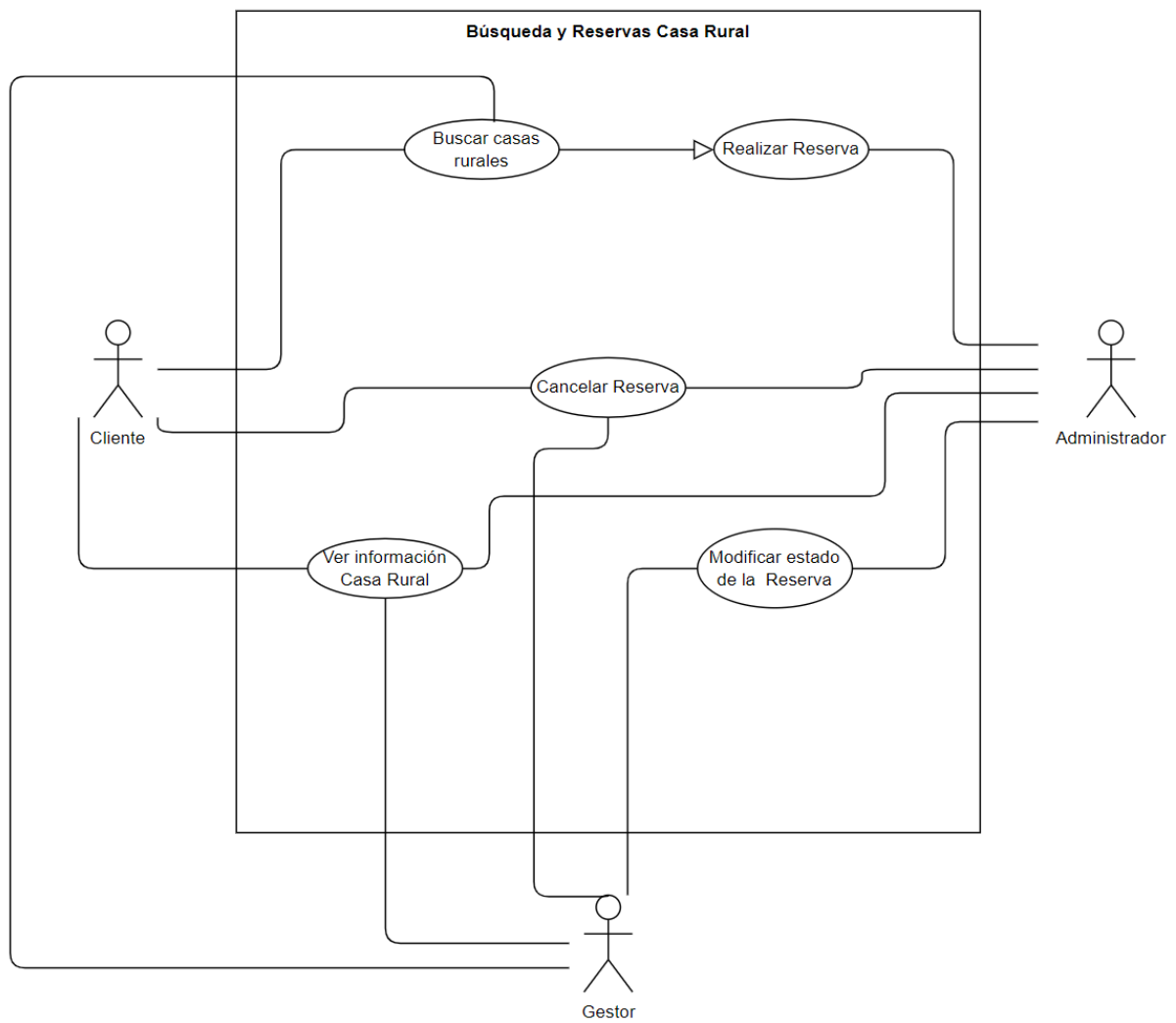
- Estas plataformas, se utilizan para enviar notificaciones por correo electrónico a los clientes, pueden ser confirmaciones de reservas, recordatorios de pago o actualizaciones sobre el estado de las reservas.
- Además, la aplicación web puede integrarse con servicios de correo electrónico como Gmail, Outlook o Hotmail, pudiendo así enviar automáticamente correos electrónicos a los clientes.

Diagramas de Análisis:

Diagramas de Casos de Uso.







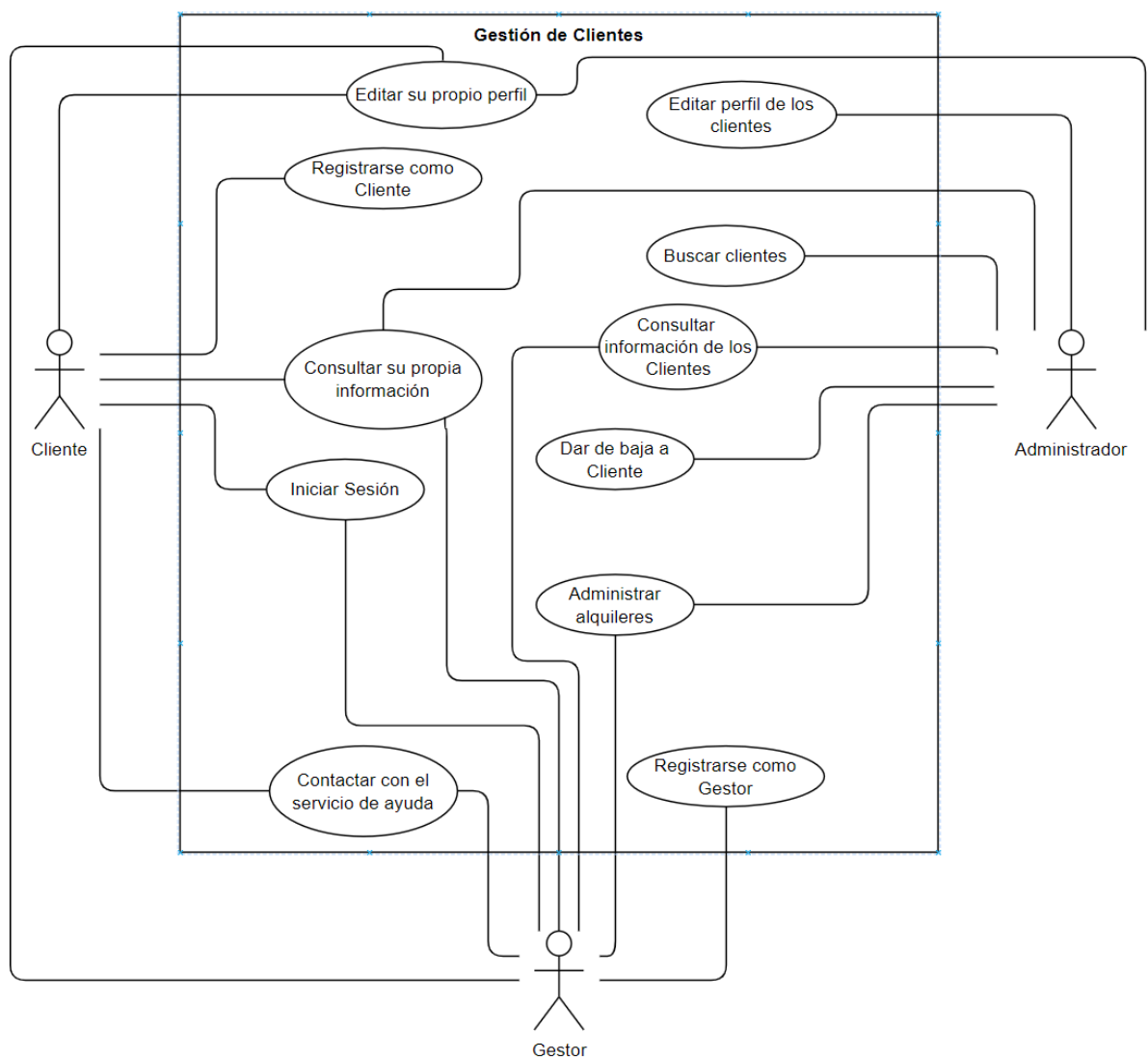


Diagrama de Estados.

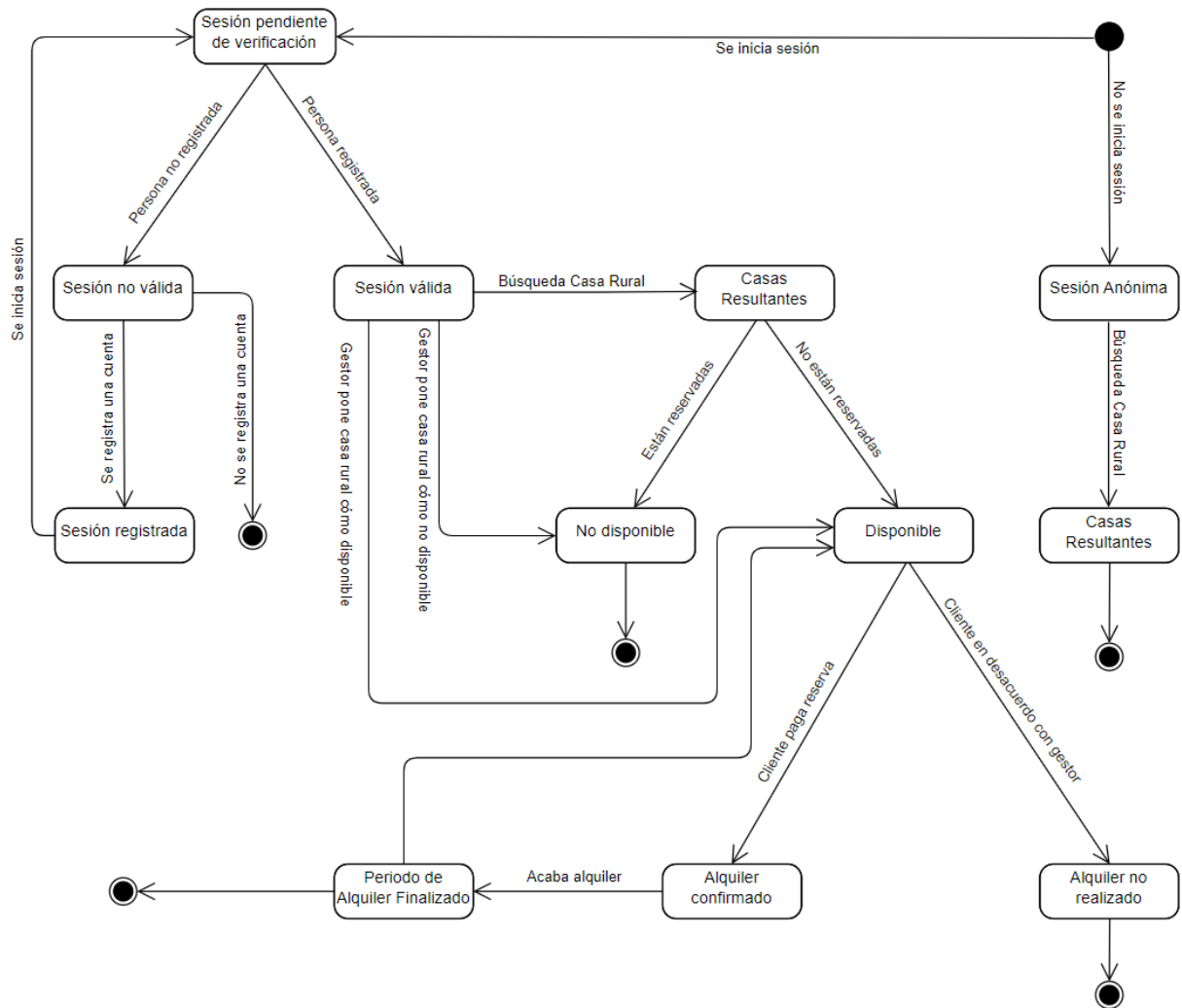
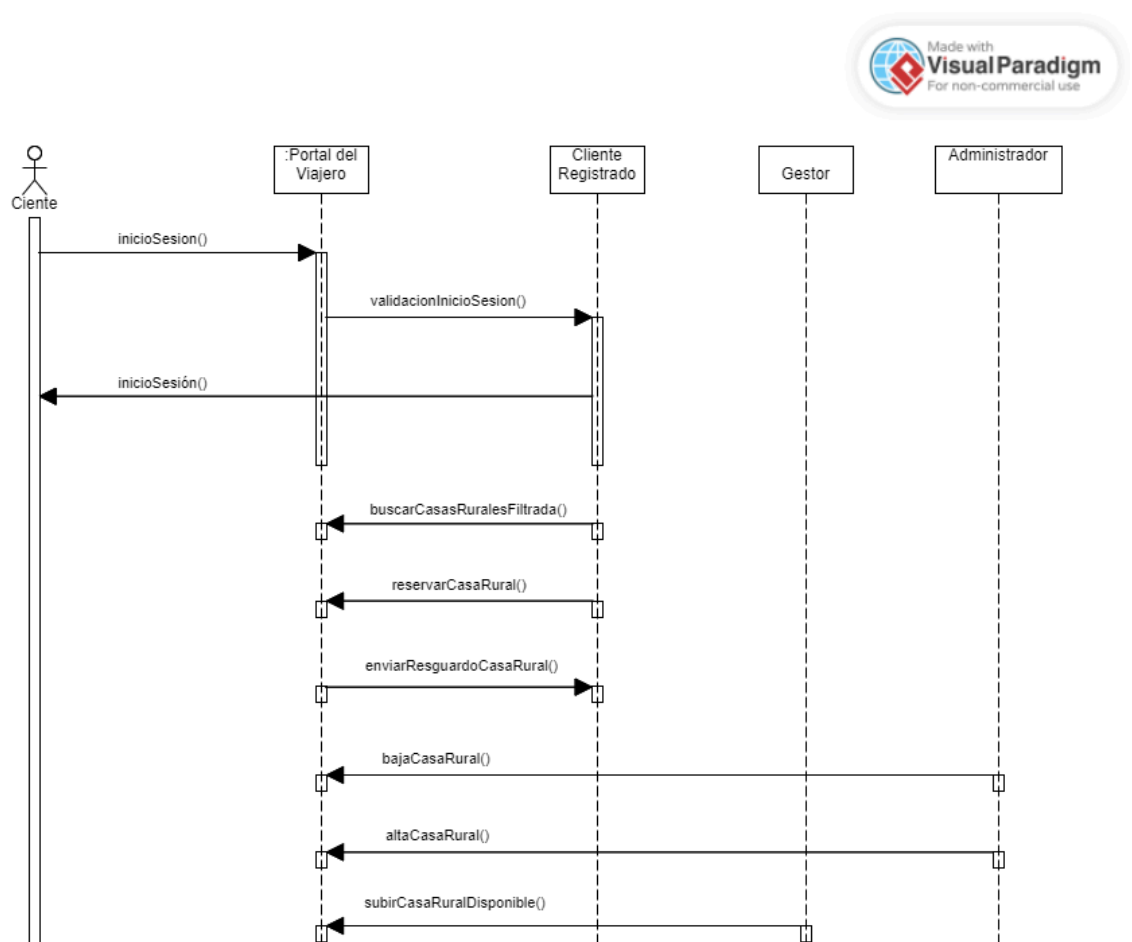


Diagrama de Secuencia o Interacción.

El diagrama de secuencia es un tipo de diagrama usado para modelar la interacción entre objetos en un sistema según UML, aunque no están pensados para mostrar las lógicas de procedimientos que son complejos.

Portal del Viajero:



Tablas de decisión.

Tabla de decisión para Gestión de Casas Rurales:

Condición	Acción
Casa Rural nueva agregada	Actualizar la página web para que se muestre las nuevas casas rurales agregadas recientemente.
Casa Rural desactivada por mantenimiento u ocupada en el lapso de tiempo en el que un cliente trata de realizar el alquiler.	Ocultar la Casa Rural de la lista de disponibles para alquilar.
Cliente solicita más información sobre Casa Rural	Mostrar detalles de la casa rural seleccionada al cliente.
Cliente alquila una Casa Rural	Actualizar la disponibilidad de la casa rural en la base de datos. Generar un alquiler en base a los datos de la casa, el cliente y el gestor. Además de enviar un resguardo al cliente.

Tabla de decisión para Gestión de Clientes:

Condición	Acción
Nuevo cliente registrado	Almacenar los datos del cliente en la base de datos.
Cliente busca casas rurales disponibles	Aplicar filtros según las preferencias del cliente. En el caso que no aplique filtros, se hará una búsqueda de todas las casas rurales que hay en la página web.
Cliente alquila casa rural	Registrar el alquiler del cliente en la base de datos y modificar en la página web el anuncio de dicha casa rural para que no esté disponible para las fechas que estén alquiladas.
Cliente visualiza información sobre casa rural	En el anuncio de la casa rural podrá ver la información sobre la casa rural, pudiendo obtener más detalles si pincha en el anuncio.
Cliente modifica la información de su perfil	Reflejar los cambios en la base de datos.

Tabla de decisión para Gestión de Usuarios y Perfiles:

Condición	Acción
Nuevo usuario registrado como gestor	Otorga permisos de gestor en la aplicación y tendrá disponible todas las opciones del gestor de viviendas.
Gestor realiza una modificación en una casa	Registrar la modificación en la base de datos y actualizar la información que se visualiza por pantalla.
Gestor elimina una casa	Eliminar la vivienda de la base de datos y de la página web.
Registra el perfil del usuario	Guardar perfil de usuario en la base de datos, ya que cada uno tendrá unas opciones disponibles o no.

Definición de interfaces de usuario:

Antes de comenzar con la definición de la interfaz de usuario, se recomienda tener en cuenta que todos estos principios se pueden ver aplicados de manera visual en el prototipo que se ha llevado a cabo haciendo uso de la herramienta de Figma:

<https://www.figma.com/file/RtIIT1WYMnWOQv93a8xAEv/Proyecto-Integrado---Portal-Del-Viajero?type=design&node-id=0%3A1&mode=design&t=HMAAc5MnJQ96fN8k-1>

Especificación de principios generales de interfaz:

Tipografía:

La elección de la tipografía adecuada desempeña un papel crucial para la legibilidad y la estética de la interfaz de la aplicación. En este proyecto, se ha seleccionado la fuente **“Inter”** debido a su excelente legibilidad tanto en pantallas digitales como en impresiones (por si se diese el caso en el que se necesitase implementar un sistema para generar archivos) .

Fondo de Pantallas:

Color: Para los fondos de pantallas de cualquiera de las páginas de la aplicación, **excepto** para la de inicio de sesión, registro de usuarios y convertir en gestor se utilizará el color blanco (**#FFFFFF**).

Utilizaremos el color blanco para el fondo de las páginas de la aplicación ya que proporciona un aspecto limpio y moderno, resaltando el contenido y facilitando mucho la legibilidad de la misma.

Enlaces:

Color: Los enlaces dentro del texto o botones tendrán el color gris (**#767272**).

Estilo de Subrayado: Los enlaces se encontrarán subrayados para indicar a los usuarios que son clickables.

El color gris oscuro lo utilizaremos para los enlaces dentro del texto o botones para diferenciarlos claramente del texto normal. Este color es lo suficientemente distintivo como para destacar los enlaces, pero lo bastante sutil como para no abrumar al usuario.

Botones de Acciones:

Color para botones con acciones de Reservar, Aceptar Operaciones: Los botones que realicen **acciones importantes** como **“Reservar”, “Aceptar”, “Registrar”, “Iniciar Sesión”, “¡Si quiero!”, “Guardar”, “Editar”, “Añadir”, “Añadir Casa Rural”** o ver los **“Detalles”** de las casas rurales tendrán el **color morado (#673DE6)**.

Color para botones con acciones de Eliminar o Cancelar Operaciones: Los botones que tengan acciones de cancelar operaciones o eliminarlas tendrán un color rojo (**#C60000**).

Estilo: Los botones tendrán esquinas redondeadas y sombras suaves para resaltar su interactividad, al hacer “hover” sobre ellos esta sombra cambiará.

Tamaño: Se mantendrá un tamaño uniforme para todos los botones de acción, intentando mantener una cierta coherencia visual.

Usamos el color morado vibrante para resaltar los botones que realizan acciones de gran relevancia, este color contrasta bien con el fondo blanco y llama la atención del usuario, indicando dónde se encuentran las funciones principales de la aplicación.

Para botones que denotan la eliminación o cancelación de alguna acción usamos el color rojo, ya que se asocia comúnmente con la alerta y la advertencia. Este color distintivo ayuda al usuario a identificar rápidamente las acciones que pueden tener consecuencias irreversibles.

Formularios e inputs:

Bordes e inputs: Los bordes de los campos de entrada de los datos tendrán un color gris claro (**#9B9898**) para diferenciar claramente cuáles son las áreas de entrada.

Estilo de Texto: El texto dentro de los inputs será de color negro (**#000000**) y tendrá un tamaño y fuente legibles para facilitar la entrada de datos.

Placeholder: Todos los inputs de la aplicación están dotados con placeholders que indican en todo momento con qué tipo de dato se han de rellenar los correspondientes inputs.

Utilizaremos el gris claro para los bordes de los campos de entrada para diferenciar las áreas donde el usuario puede ingresar datos. Este color es lo suficientemente sutil como para no distraer al usuario, pero que a su vez también sea bastante visible para entender los campos con los que deben ser rellenados.

Texto Descriptivo:

Color: El texto descriptivo que está relacionado con las características de las casas rurales, títulos y otros detalles será de color negro (#000000).

Estilo de fuentes: Se utilizará una fuente legible y moderna para garantizar la fácil lectura del contenido.

El color negro se usa para el texto descriptivo relacionado con las características o detalles de las casas rurales, títulos y otros detalles importantes. El negro ofrece un **alto contraste** con el fondo en blanco, lo que garantiza una excelente legibilidad y claridad visual.

Iconos y Elementos Visuales:

Estilo de Iconos: Utilizaremos iconos que sean intuitivos, descriptivos y reconocibles para mejorar la experiencia del usuario..

Consistencia: Todos los elementos visuales, incluidos iconos y gráficos, seguirán un estilo coherente para mantener la consistencia en toda la interfaz.

Espaciado y Alineación:

Espaciado: Mantendremos un espaciado uniforme entre los elementos de la interfaz que mejorará la legibilidad y la organización visual, creando una apariencia limpia y ordenada.

Alineación: La correcta alineación de los elementos garantiza una apariencia ordenada y profesional, lo que facilita la navegación y la comprensión del contenido por parte del usuario.

Responsividad:

Adaptabilidad: La interfaz responsive garantiza una experiencia de usuario óptima en dispositivos de diferentes tamaños y resoluciones, lo que mejora la accesibilidad y la usabilidad de la aplicación.

Especificación de formatos individuales de la interfaz de pantalla:

Página de Inicio:

Descripción: Esta pantalla será la primera que vea el usuario al acceder a la aplicación, incluso sin iniciar sesión. El objetivo de esta página es dar la bienvenida al usuario y proporcionar una visión general de las funciones disponibles, así como para que el usuario pueda ver las casas rurales que hay disponibles, sin tener la opción de poder llevar a cabo una reserva si no se encuentra con la sesión iniciada.

Elementos:

- Encabezado con el nombre y el logotipo de la aplicación. Lo más remarcable es el buscador que se ubica en la parte superior de la pantalla, en el que se podrá especificar el destino, la fecha de llegada y la fecha de salida. También hay un link, el cuál permitirá a un usuario registrarse como gestor para poner una casa rural en alquiler.
- En la parte superior a la derecha, podremos encontrar un menú desplegable en el cuál se le permitirá a cualquier usuario registrarse, editar perfil, iniciar sesión, cerrar sesión, ver reservas realizadas, administrar comodidades o ver usuarios registrados, entre otras opciones.
- Filtros de búsqueda por precio por día, fechas de estancia, destino, instalaciones, comodidades. Estos filtros serán aplicables desde un menú lateral ubicado a la izquierda del lugar dónde salen las casas rurales disponibles.
- Listado de resultados de búsqueda con tarjetas de anuncio en los cuáles se mostrarán las casas disponibles y que cumplan con los filtros de búsqueda del usuario, los cuáles incluyen título, ubicación, precio por noche, una miniatura de la propiedad (imagen) y un botón en el cuál se pueden ver más detalles de la casa rural, y a través del cual se puede llegar a la página para reservar una casa rural.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará y se optimizará para adaptarse a diferentes tamaños de pantalla y resoluciones.
- El diseño de la página de inicio será minimalista y atractivo. Tanto en grandes como pequeñas resoluciones.
- Se utilizarán fuentes y colores consistentes con la identidad de marca de la aplicación.

Página de Inicio de Sesión:

Descripción: En esta pantalla, los usuarios pueden iniciar sesión con sus cuentas existentes. Los usuarios podrán acceder con su usuario y contraseña o podrán acceder mediante su cuenta de Google (servicio OAuth no implementado para la versión actual).

Elementos:

- Formulario de inicio de sesión con campos para nombre de usuario y contraseña.
- Enlace para recuperar la contraseña si el usuario la olvidó.
- Casilla para marcar si el usuario quiere que se le recuerde las credenciales o no.
- Botón para iniciar sesión.
- Enlace para mandar al usuario a la pantalla de registro, en el caso de que no se haya registrado previamente.

Aspectos Dinámicos:

- La pantalla se adaptará para garantizar una experiencia de inicio de sesión fluida en todos los dispositivos.

Página de Registro:

Descripción: En esta página, los nuevos usuarios pueden crear una cuenta en la aplicación.

Elementos:

- Formulario de registro con campos en los cuáles el usuario tendrá que ingresar información personal, como usuario, contraseña, DNI, Teléfono, correo electrónico.
- Opciones para seleccionar el tipo de cuenta, si usuario estándar o gestor.
- Casilla para marcar si el usuario acepta los términos y condiciones de política y privacidad, siendo obligatoria marcarla, para efectuar el registro.
- Botón para completar el registro.
- Enlace para redirigir a la página de inicio de sesión en el caso de que el usuario ya se haya registrado previamente.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para facilitar el registro en dispositivos de diferentes tamaños.

Página de Conversión a Gestor:

Descripción: Esta pantalla permite a los usuarios convertirse en gestores para poder listar y gestionar casas rurales.

Elementos:

- Información sobre los requisitos y beneficios de convertirse en gestor.
- Botón para enviar la solicitud y registrarse como gestor (para ello debe tener la sesión de usuario ya iniciada previamente).

Aspectos Dinámicos:

- La pantalla se ajustará facilitando la conversión a gestor en todos los dispositivos.

Página de Detalles de la Casa Rural:

Descripción: Esta pantalla proporciona a los usuarios información detallada y completa sobre una casa rural específica, permitiendo tomar decisiones informadas sobre una posible reserva.

Elementos:

- Título y subtítulo atractivos de la casa rural.
- Imágenes de la casa rural, para que el usuario pueda tener una primera impresión.
- Descripción general de la casa rural, incluidas las características, comodidades, ubicación y cualquier información relevante adicional.
- Diseño limpio y fácil de leer, con viñetas y párrafos cortos para obtener así una mejor legibilidad.
- Viñeta que muestra los precios y la disponibilidad de la casa rural para reservar.
- Actualización de la disponibilidad y los precios en tiempo real.
- Sección dedicada a mostrar las reseñas y calificaciones de los usuarios que se han alojado en dicha casa rural.
- Sección dedicada a las valoraciones de los usuarios que se hayan alojado anteriormente en esa casas.
- Botón visible y resaltado del resto de elementos de la página que sirve para iniciar el proceso de reserva de la casa rural.

Aspectos Dinámicos:

- La pantalla se ajustará y optimizará para proporcionar una mejor visibilidad en todos los dispositivos.
- Se utilizarán fuentes y colores consistentes con la identidad de marca de la aplicación.

Página de Gestión de Casas Rurales:

Descripción: Esta pantalla está diseñada específicamente para permitir a los gestores de casas rurales listar y gestionar sus propiedades de alquiler vacacional de una manera eficiente y organizada.

Elementos:

- Acceso rápido a las funciones de edición, eliminación o guardar nueva casa rural.
- Botones o enlaces para acceder a las funciones de edición, eliminación o adición de nuevas casas rurales.
- Cuadro de diálogo o pantalla de edición intuitiva y fácil de usar para actualizar la información de la casa rural, como el título, la descripción o las imágenes de la misma.
- Valoraciones y calificaciones que el gestor podrá visualizarlas todas pero sin editarlas ni eliminarlas.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará y optimizará para proporcionar una experiencia de gestión bastante buena.
- Se proporciona una interfaz de usuario intuitiva y fácil de usar, con accesos directos y opciones claras.
- Se utilizarán fuentes y colores consistentes con la identidad de marca.

Página de Edición de Perfil de Usuarios:

Descripción: Página en la cuál los usuarios pueden editar su información personal y su perfil creado al registrarse.

Elementos:

- Formulario con campo para editar información personal, como nombre, DNI, correo electrónico, descripción, teléfono.
- Botón para guardar los cambios.

Aspectos Dinámicos:

- La pantalla se ajustará para facilitar la edición del perfil en los dispositivos de diferentes tamaños.
- Se utilizarán fuentes y colores consistentes con la identidad de marca.

Página de Usuarios Registrados:

Descripción: Esta pantalla está diseñada para mostrar una lista de todos los usuarios registrados en la aplicación, brindando a los administradores una visión general clara y organizada de la base de usuarios.

Elementos:

- Lista de usuarios con información básica, cómo por ejemplo el nombre del usuario.
- Filtros y opciones de clasificación para ayudar a los administradores a encontrar a los usuarios.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para mostrar la lista de usuarios de manera clara y ordenada en todos los dispositivos.
- Se proporcionará una interfaz de usuario intuitiva y fácil de usar, con accesos directos y opciones claras para realizar tareas comunes relacionadas con la gestión de usuarios registrados.

Página de Perfil de Otro Usuario:

Descripción: En esta pantalla los gestores pueden ver el perfil de los usuarios estándares pero sin editarlo.

Elementos:

- Información básica sobre el usuario, como son el nombre, la imagen de perfil, la descripción, el teléfono o el correo electrónico.
- Un enlace que te lleva a una pestaña donde se muestran todas las casas rurales puestas para alquilar pertenecientes a ese usuario (siempre y cuando sea gestor).

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para mostrar la lista de usuarios de manera clara y ordenada en todos los dispositivos.
- Se proporcionará una interfaz de usuario intuitiva y fácil de usar, con accesos directos y opciones claras para realizar tareas comunes relacionadas con la gestión de usuarios registrados.

Página de Reservas del Usuario:

Descripción: En esta pantalla se muestran todas las reservas que ha realizado un usuario específico (usuario propio).

Elementos:

- Lista de reservas con información detallada, como la casa detallada, el precio total en ese momento y las características con la que disponía el alquiler.
- Opciones para cancelar una reserva (siempre y cuando no se haya realizado o ya se haya disfrutado de ella).
- Botón para ver los detalles de la casa rural.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para mostrar las reservas de manera clara y organizada en todos los dispositivos.

Página de Reservas de las Casas Rurales Puesta en Alquiler por un Gestor:

Descripción: Muestra al gestor todas las casas disponibles que tiene puestas en alquiler.

Elementos:

- Anuncios de las casas rurales puestas en alquiler junto con su información.
- Botón para añadir una nueva casa rural.
- Botón para editar cualquier sobre una casa rural la cuál el propio gestor haya puesto en alquiler.
- Botón para eliminar el anuncio de la casa rural y por tanto, quitarla de disponible.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para mostrar las reservas de manera clara y organizada en todos los dispositivos.
- Se proporcionará una interfaz de usuario intuitiva y fácil de usar, con accesos directos y opciones claras para realizar tareas comunes relacionadas con la gestión de usuarios registrados.

Página de Comodidades:

Descripción: En esta pantalla, los gestores pueden ver una lista de las comodidades disponibles en la web para las casas rurales.

Elementos:

- Lista de comodidades con descripciones breves.
- Opciones para buscar y filtrar comodidades.
- Opción para editar o eliminar comodidad.

Aspectos Dinámicos:

- La disposición de los elementos se ajustará para mostrar las reservas de manera clara y organizada en todos los dispositivos.
- Se proporcionará una interfaz de usuario intuitiva y fácil de usar, con accesos directos y opciones claras para realizar tareas comunes relacionadas con la gestión de comodidades que puede tener un alojamiento.

Página de Creación de Comodidades:

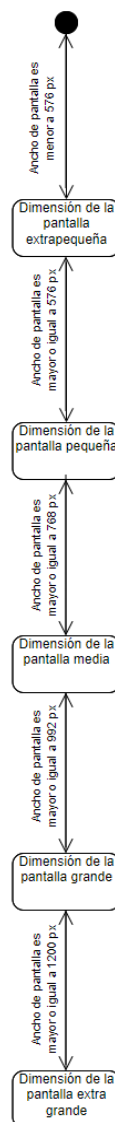
Descripción: Esta pantalla permite a los administradores crear nuevas comodidades para agregar a las casas rurales.

Elementos:

- Formulario con campos para ingresar el nombre y la descripción de la nueva comodidad.
- Opción para cargar un icono representativo de la comodidad.
- Botón para guardar la comodidad y otro botón para cancelar el guardado de la comodidad que se está creando.

Diagrama de los distintos estados por los que puede pasar una pantalla:

Los distintos estados por los que puede pasar la pantalla van a ir cambiando según la resolución de esta misma. Como en nuestro proyecto hemos usado **bootstrap** para estructurar todos los elementos que componen nuestra página, el diagrama que se muestra a continuación se encuentra acorde a las especificaciones responsive aportadas por esta tecnología:



Identificación de perfiles de usuario:

Perfil Administrador:

Este perfil corresponde al que tendría una entidad con la capacidad de realizar cualquier acción dentro de la aplicación.

Al encontrarse la sesión logueada, no podrá visualizar las pantallas de registro y login. Ya que estas pantallas únicamente son visibles cuando el usuario no se encuentra logado (por lo que se encuentra usando un perfil anónimo).

Pantallas que puede visualizar	Acciones que no puede realizar
<ul style="list-style-type: none">Todas las pantallas de la aplicación excepto las de login y registro.	

El perfil de administrador está destinado a mantener el orden dentro de la aplicación. Puede desde eliminar anuncios de casas rurales de otros gestores hasta modificar su información. Siempre se tiene en cuenta que los administradores no realizarán acciones fraudulentas dentro de la aplicación, y que únicamente actuarán en situaciones dónde sea de extrema necesidad (alquileres fraudulentos, detección de bots en las valoraciones, mensajes de odio, etc...).

Perfil Cliente:

Este perfil corresponde al que tendrá el consumidor promedio de la aplicación. Tendrá una experiencia de usuario mucho más completa que la podría tener alguien con perfil anónimo.

Al encontrarse la sesión logueada, no podrá visualizar las pantallas de registro y login. Ya que estas pantallas únicamente son visibles cuando el usuario no se encuentra logado (por lo que se encuentra usando un perfil anónimo).

Pantallas que puede visualizar	Acciones que no puede realizar
<ul style="list-style-type: none">InicioDetalles casa rural (vista cliente)Ver perfiles de otros usuariosVer casas en alquiler de un usuario (accediendo a su perfil).Página para convertirse en gestor.Ver perfil propio.Editar perfil propio.Ver reservas.	<ul style="list-style-type: none">Poner en alquiler una vivienda. Tampoco operaciones relacionadas como modificar viviendas etc...

El perfil de cliente está destinado a realizar alquileres. Por eso mismo, no se le proporcionará ninguna herramienta de gestión de estos. De la misma manera, a la hora de visualizar su perfil no se renderizan opciones cómo las de “Casas en alquiler”.

Perfil Gestor:

Este perfil corresponde al que tendrán aquellos usuarios que quieran poner su casa u otra vivienda que posean en alquiler.

Al encontrarse la sesión logueada, no podrá visualizar las pantallas de registro y login. Ya que estas pantallas únicamente son visibles cuando el usuario no se encuentra logado (por lo que se encuentra usando un perfil anónimo).

Pantallas que puede visualizar	Acciones que no puede realizar
<ul style="list-style-type: none">• Inicio• Detalles casa rural (vista cliente)• Editar detalles casa rural (vista gestor)• Ver perfiles de otros usuarios• Ver casas en alquiler de un usuario (accediendo a su perfil).• Página para convertirse en gestor.• Ver perfil propio (vista gestor).• Editar perfil propio (vista gestor).• Ver reservas.• Casas propietarias en alquiler.	<ul style="list-style-type: none">• Crear comodidades para sus viviendas. (Las comodidades únicamente pueden ser creadas por los administradores. Una vez creadas, los gestores pueden elegir comodidades e instalaciones como hotel, cubiertos, piscina, etc... Que serán cargados en un select múltiple desde dónde podrán añadirlos a sus anuncios.• Buscar usuarios concretos dentro de la aplicación por su nombre de usuario, correo u otra cosa.

El perfil de gestor está destinado a poner anuncios sobre alquileres de sus viviendas para que los demás usuarios puedan llevar a cabo un alquiler. Son los usuarios que más permisos poseen, únicamente sobrepasados por los administradores. Pueden visualizar todas las pantallas que el usuario promedio puede ver e incluso puede visualizar pantallas que tienen que ver directamente con la gestión de viviendas dentro de la aplicación.

Perfil Anónimo (Usuario no registrado - Sin perfil):

El perfil de usuario anónimo corresponde a aquel que no ha llevado a cabo un proceso de registro y logueo en la aplicación.

Es el usuario que más restricciones tiene dentro de la aplicación.

Pantallas que puede visualizar	Acciones que no puede realizar
<ul style="list-style-type: none">• Inicio• Login• Registro• Detalles casa rural (vista cliente)• Ver perfiles de otros usuarios• Ver casas en alquiler de un usuario (accediendo a su perfil).	<ul style="list-style-type: none">• Llevar a cabo el alquiler de una casa.• Poner en alquiler una vivienda.• Editar Perfil• Ver su propio perfil• Ver las reservas que ha llevado a cabo

Para este perfil no se renderizan elementos del menú de navegación como puede ser los enlaces de “cerrar sesión”, “editar perfil”, “ver reservas”, entre otros. Además, tampoco se renderiza la opción de “¿Quiéres poner en alquiler tu casa?”, la cuál permite a un usuario con perfil de “Cliente” elevar su perfil al de Gestor.

En general, la interfaz para el perfil anónimo será de **SOLO LECTURA**, lo que significa que no podrá interactuar de ninguna manera con los datos almacenados, únicamente viéndolos.

Especificación de los formatos de impresión:

En nuestro caso, no usaremos ningún tipo de formato de impresión, ya que la aplicación no dará cabida a imprimir o generar ningún tipo de documento.

Especificación de la navegabilidad entre pantallas:

A continuación, se presentarán una serie de diagramas que representarán la navegabilidad entre pantallas que podrán realizar los distintos perfiles de usuario dentro de la aplicación. A su vez, también se exponen aquellas pantallas que no podrán visitar.

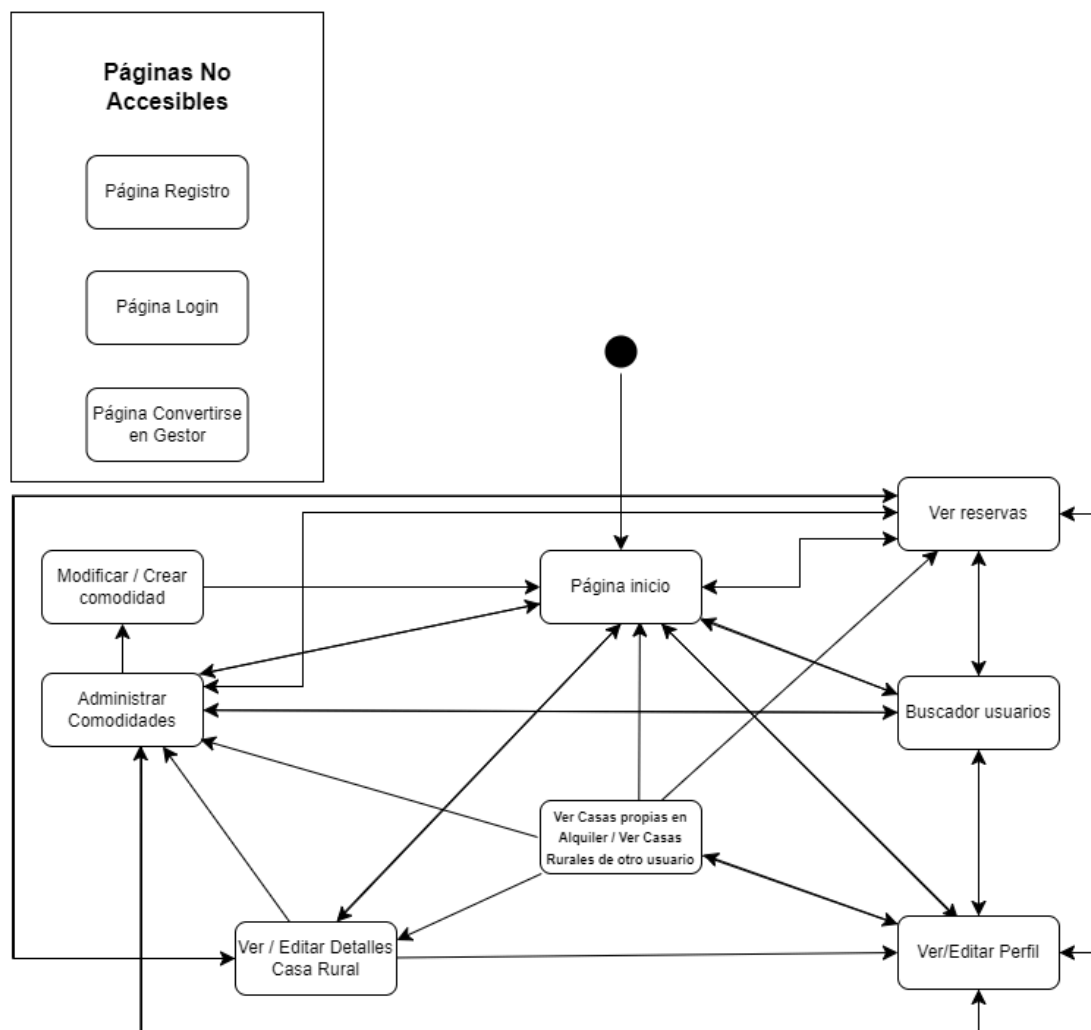
Los diagramas indican mediante flechas la navegación de una página a otra, la cuál puede ser bidireccional o unidireccional.

Aparte de los diagramas, la navegación dentro de la aplicación también podrá ser apreciada en un prototipo de la aplicación realizado en figma, usando la opción de “play” e interactuando con los elementos navegables de la página:

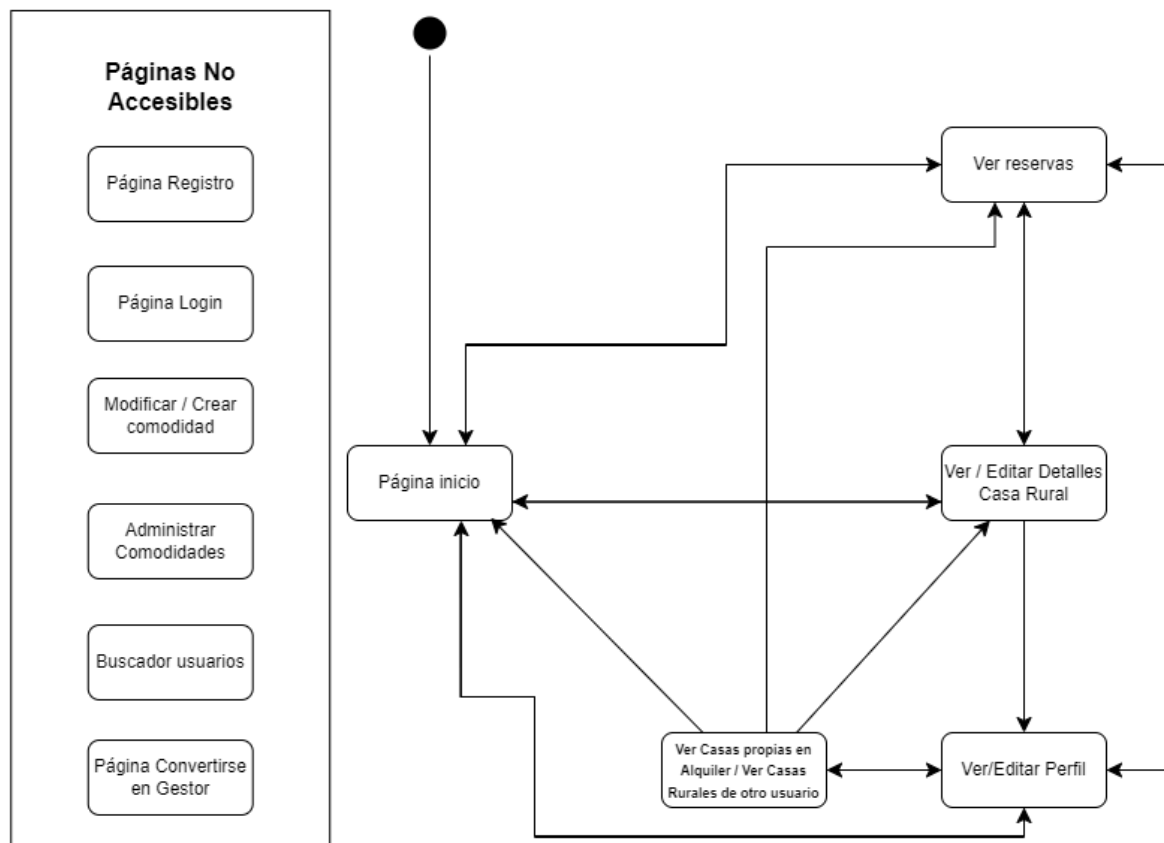
ENLACE A FIGMA:

<https://www.figma.com/file/RtIIT1WYMnWOQv93a8xAEv/Proyecto-Integrado---Portal-Del-Viajero?type=design&node-id=0%3A1&mode=design&t=HMAAc5MnJQ96fN8k-1>

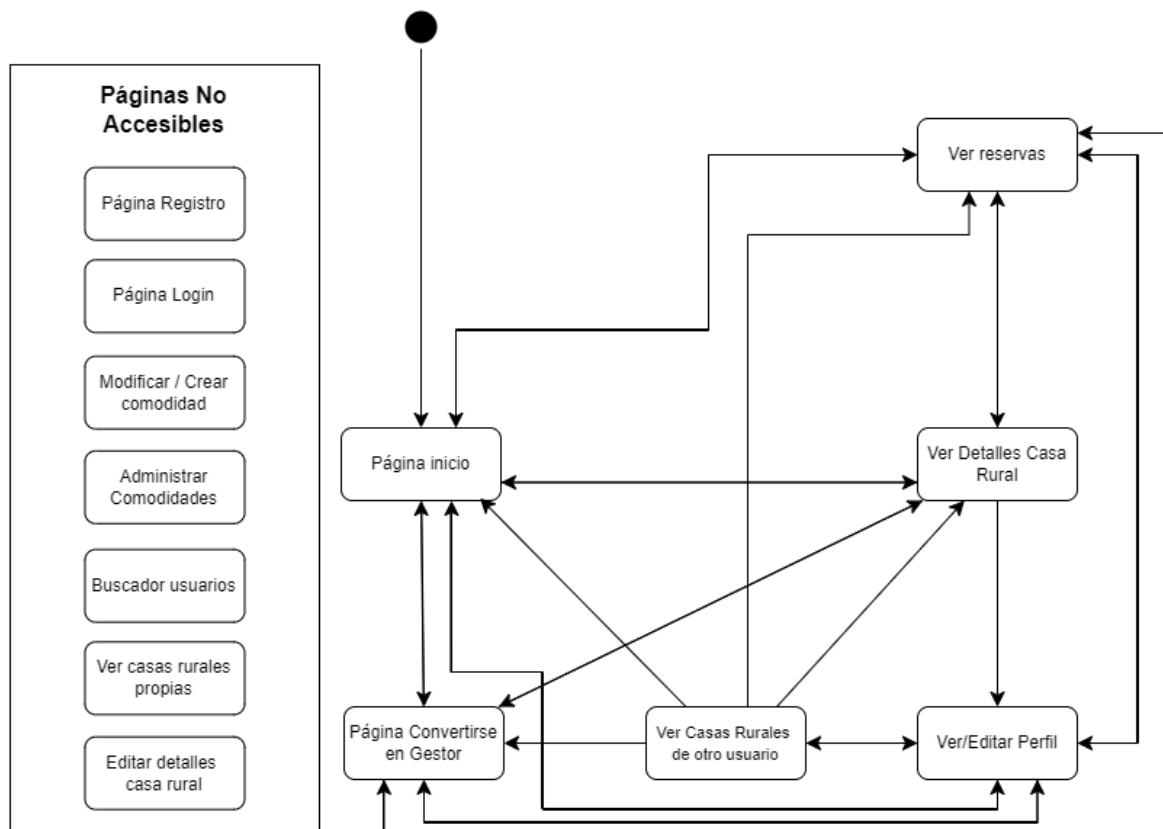
Perfil Administrador:



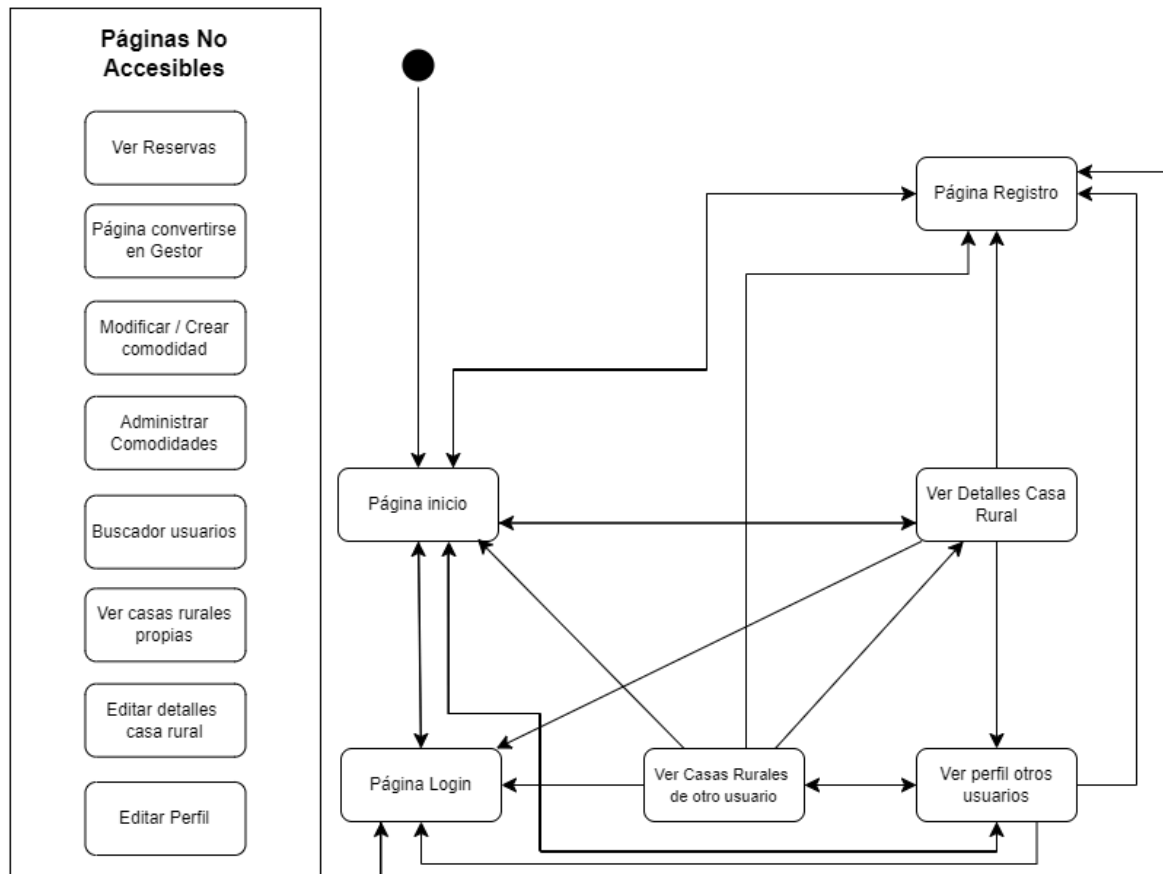
Perfil Gestor:



Perfil Cliente:



Perfil Anónimo:

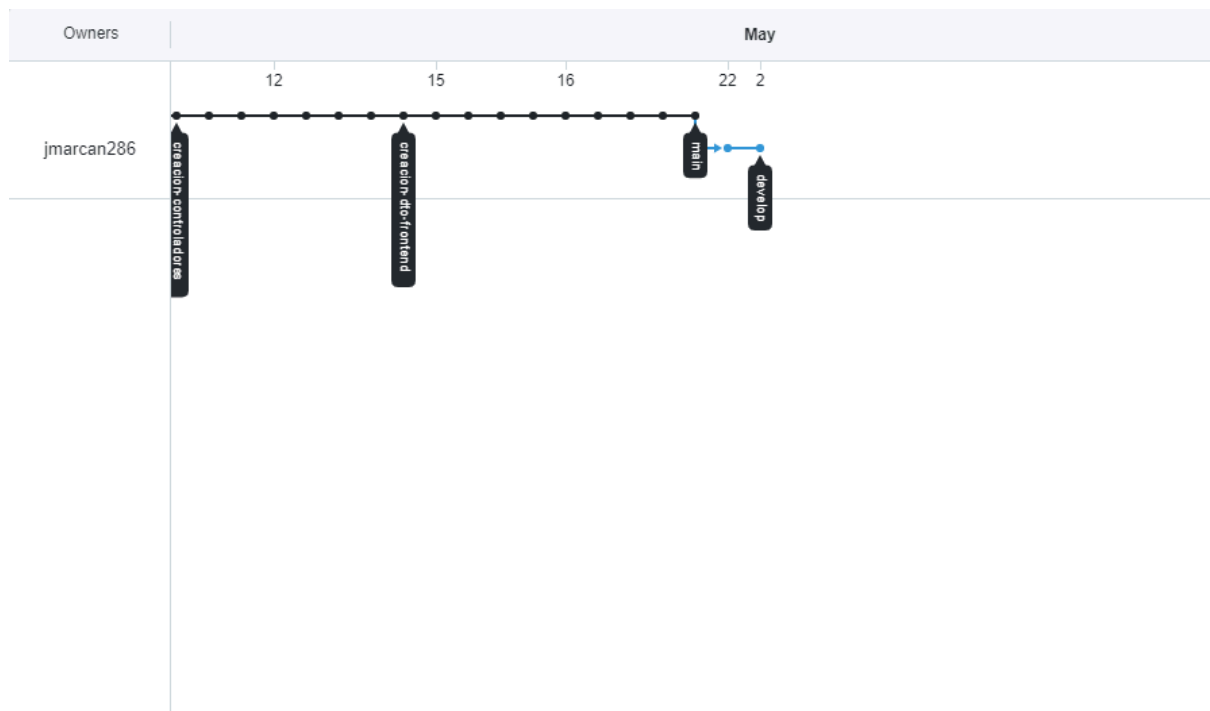


CONSTRUCCIÓN DEL SISTEMA

Control del sistema en Github:

ACLARACIÓN: Se ha elegido que el repositorio sea público porque si es privado no se pueden acceder a acciones como el apartado de Network.

LINK A GITHUB: https://github.com/jmarcan286/portal_del_viajero-PI-



En este apartado se detalla el proceso y las herramientas utilizadas durante la construcción del sistema, así como las prácticas de desarrollo colaborativo implementadas, centrándose en el uso de un repositorio git público en Github para la gestión del código y las fuentes del proyecto.

Uso de Repositorio Git Público en Github:

Uso de Cuentas Personales en Github: En este caso, ambos participantes del proyecto figuran cómo colaboradores en el repositorio de Github, permitiendo una colaboración eficaz y un buen seguimiento del proyecto.

Configuración del Repositorio Público: Se ha establecido el uso de un repositorio git público en la plataforma Github como principal herramienta de control de versiones para el proyecto, para permitir que sea mucho más accesible, escalable y estable.

Colaboradores del Repositorio: Los profesores encargados del seguimiento y evaluación del proyecto también han sido añadidos como colaboradores al repositorio, permitiéndoles revisar el progreso del proyecto, proporcionar retroalimentación y evaluar el trabajo de los estudiantes de manera eficiente.

Estructura de Ramas: Se ha llevado a cabo una estrategia de ramificación para organizar y gestionar el desarrollo del proyecto, facilitando el desarrollo y la integración de nuevas características:

- **main:** Rama principal del repositorio, donde se encuentra la versión estable y desplegable del proyecto.
- **develop:** Rama de desarrollo para la integración y prueba de nuevas características antes de mergearlas con la rama principal.
- **creacion-controladores:** Rama destinada al desarrollo de los controladores en el backend.
- **creacion-dto:** Rama enfocada en la creación de los DTO (Data Transfer Objects) en el backend.
- **creacion-dto-frontend:** Rama destinada a la creación de los DTO en el frontend.
- **creacion-jpa-ddor:** Rama utilizada para la creación de las entidades JPA y relaciones en el backend.
- **creacion-jpa-jmc:** Rama destinada a la creación de las entidades JPA y relaciones en el backend.
- **creacion-repositorios:** Rama para el desarrollo de los repositorios en el backend.
- **creacion-servicios:** Rama enfocada en la creación de los servicios en el backend.
- **Otras ramas:** Durante el desarrollo, se irán implementando ramas que contengan partes del desarrollo independientes, relevantes y de gran proporción de cambios que luego se unirán a la rama principal para así unificar la aplicación de la manera más organizada posible.

Organización de Recursos: Además, se han creado tres estructuras de carpetas para almacenar recursos importantes del proyecto:

- **Pdv-backend:** Contiene el código fuente del backend de la aplicación Portal del Viajero.
- **Pdv-frontend:** Contiene el código fuente del frontend de la aplicación web.
- **public-resources:** Almacena documentos importantes como todos los diagramas de análisis, diagramas de la base de datos y capturas de la misma, y la documentación del proyecto integrado hasta el momento.

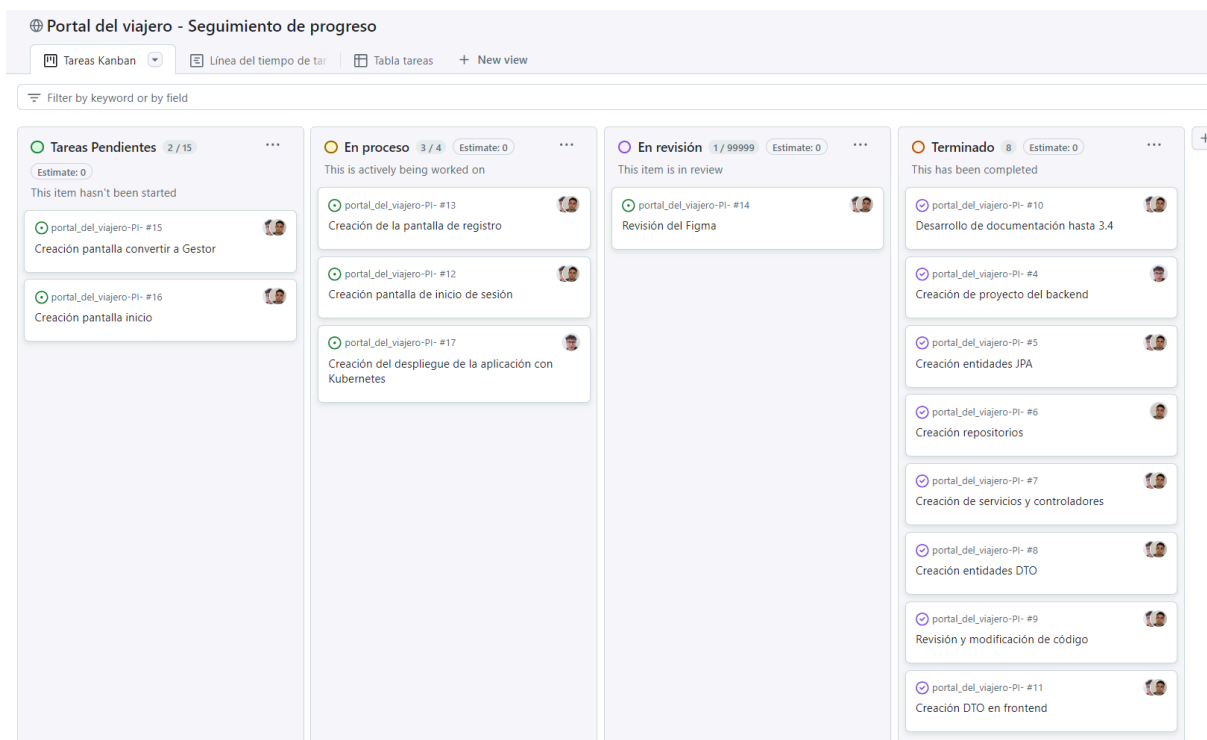
Prácticas de Desarrollo Colaborativo:

- Se han adoptado prácticas de desarrollo colaborativo, como la revisión de código entre compañeros y la asignación de tareas y responsabilidades claras.
- Se ha fomentado la comunicación y colaboración constante entre los miembros del equipo a través de reuniones regulares y herramientas de comunicación como Discord.
- Se ha establecido un flujo de trabajo basado en ramas para facilitar el desarrollo de diferentes funcionalidades y evitar conflictos en el código.

- Se han implementado reuniones de seguimientos periódicas para revisar el progreso del proyecto, identificar posibles problemas y tomar decisiones sobre el rumbo del desarrollo.

Este enfoque de construcción garantiza la eficiencia en el desarrollo, la colaboración efectiva entre los miembros del equipo y la trazabilidad completa de los cambios realizados durante el ciclo de vida del proyecto, permitiendo así una gestión eficiente del código y una mayor calidad en el producto final.

Tableros Kanban y seguimiento del progreso de la aplicación:



Con objeto de mantener una visión global de todos los aspectos que se han ido cambiando, la fecha en la que se han producido los cambios y los tipos de cambios, se ha procedido a crear un tablero Kanban.

Los colaboradores del proyecto van creando “Issues” que se engloban dentro de los distintos estados que se muestran en la imagen.

De esta manera, generamos una traza de tareas que se han ido completando para así poder visualizar, una vez acabado el proyecto, cuánto tiempo nos ha tomado realizar cada tarea en el periodo que abarca el desarrollo de la aplicación.

Despliegue de la aplicación con Docker y Kubernetes:

En este caso, se ha decidido desplegar la aplicación haciendo uso de docker y kubernetes. En este apartado no entraremos mucho en detalle con lo que hace cada línea de los archivos que componen el despliegue, sin embargo, si se explicará lo que hace cada uno de manera superficial.

Usaremos Docker para crear Dockerfiles de las distintas partes de la aplicación, y poder crear así una imagen personalizada del frontend y del backend, que supla todas las necesidades requeridas para poner en producción nuestra app.

Explicación de los Dockerfiles:

```
Pdv-frontend > Dockerfile > ...
1  # Use official node image as the base image
2  FROM node:lts as build
3
4  ENV NG_APP_API=http://localhost:9999
5  ADD ./package.json /tmp/package.json
6  RUN cd /tmp && npm install
7  RUN mkdir -p /usr/local/app && cp -a /tmp/node_modules /usr/local/app/
8  WORKDIR /usr/local/app
9  COPY ./ /usr/local/app/
10 RUN npm run build
11
12 FROM nginx:latest
13
14 COPY --from=build /usr/local/app/dist/build/browser /usr/share/nginx/html
15 COPY ./nginx.conf /etc/nginx/conf.d/default.conf
16 EXPOSE 80 443 6006 4200
17
18 ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

En este caso, este archivo Dockerfile es el que se hará cargo de generar la imagen personalizada del frontend que se desplegará en los pods de kubernetes.

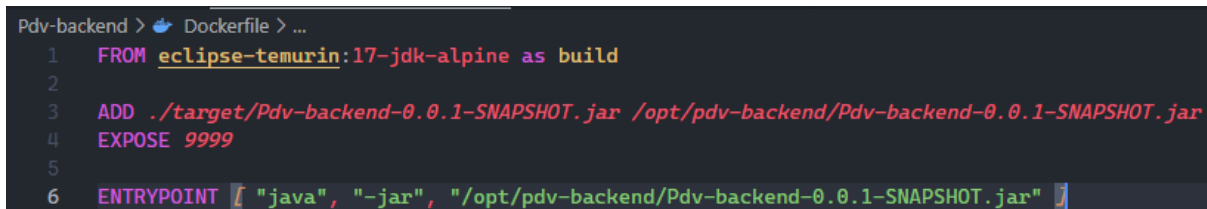
Este archivo hará lo siguiente:

1. Partirá de una imagen de node con la versión lts y le asignará el alias de “build”.
2. Añade la variable de entorno NG_APP_API que controla a qué servicio backend estará haciendo las peticiones (en este caso es localhost porque todavía no está totalmente integrado con kubernetes).
3. Copia el package.json local a la imagen, crea un directorio temporal y ahí monta el proyecto con un “npm run build”, que creará un desplegable de la aplicación de Angular.
4. Ahora partiremos de una imagen de nginx con su última versión, copiaremos el desplegable de la aplicación de angular a la carpeta “/usr/share/nginx/html”.

Expondremos los puertos correspondientes con los que podremos comunicarnos con Nginx. Por último, ejecutaremos el servidor.

Se ha de recalcar que esta imagen ha de ser construida usando el siguiente comando:

```
docker build -t jmarcan286/pdv-frontend .
```

A screenshot of a code editor showing a Dockerfile for 'Pdv-backend'. The file is named 'Dockerfile' and is located in the 'Pdv-backend' directory. The content of the Dockerfile is as follows:

```
1 FROM eclipse-temurin:17-jdk-alpine as build
2
3 ADD ./target/Pdv-backend-0.0.1-SNAPSHOT.jar /opt/pdv-backend/Pdv-backend-0.0.1-SNAPSHOT.jar
4 EXPOSE 9999
5
6 ENTRYPOINT ["java", "-jar", "/opt/pdv-backend/Pdv-backend-0.0.1-SNAPSHOT.jar"]
```

En este caso, este archivo Dockerfile es el que se hará cargo de generar la imagen personalizada del backend que se desplegará en los pods de kubernetes.

1. Partimos de una imagen de eclipse-temurin con la jdk-17.
2. Añadimos el archivo desplegable de nuestra aplicación a la ruta /opt/pdv-backend/pdv-backend-0.0.1-SNAPSHOT.jar
3. Exponemos el puerto 9999 que es el que usa el servidor de spring boot que tenemos configurado.
4. Ejecutamos el desplegable haciendo uso de `java -jar`

Se ha de recalcar que esta imagen ha de ser construida usando el siguiente comando:

```
docker build -t jmarcan286/pdv-backend .
```

La imagen que usaremos para desplegar la base de datos no será personalizada, en este proyecto se hará uso de la imagen oficial de mysql en su última versión.

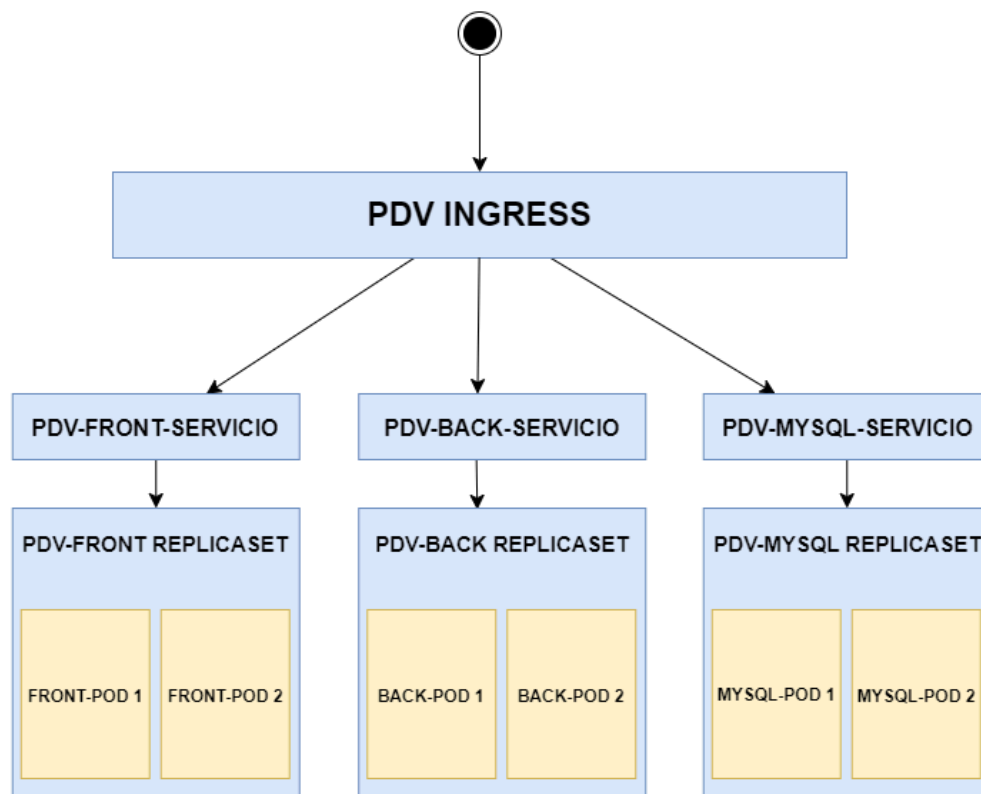
Es muy importante que antes de empezar a desplegar todo el entorno se haya descargado docker desktop.

Despliegue con Kubernetes:

Actualmente el despliegue con kubernetes no se encuentra completamente finalizado (en código).

Antes de comenzar a desplegar la aplicación, debemos tener habilitado en nuestro entorno de producción la opción de “**Enable Kubernetes**” si es que no lo trae activado por defecto (cómo puede ser el caso de Docker Desktop). Estaremos utilizando minikube para desplegar nuestro Cluster de Kubernetes.

A continuación, se expondrán los detalles del despliegue realizado con kubernetes:



Como se puede observar en el diagrama, cada una de las partes de la aplicación tendrá un servicio, que comunicará las peticiones con los distintos pods y de esta manera distribuirá la carga de la aplicación, actuando como un balanceador de carga. A su vez, tendremos un servicio Ingress actuando por encima de todas estas capas, este servicio ingress nos permitirá asociar “paths” a nuestros servicios, de esta manera, podremos obtener una interfaz de comunicación que nos permitirá conectar cada parte de la aplicación de manera más sencilla.

```

Deployment > Y pdv-frontend-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4  labels:
5    app: pdv-frontend
6    name: pdv-frontend-deployment
7  spec:
8    replicas: 3
9    selector:
10   matchLabels:
11     app: pdv-frontend
12   template:
13     metadata:
14       labels:
15         app: pdv-frontend
16     spec:
17       containers:
18       - image: jmarcan286/pdv-frontend
19         name: pdv-frontend
20         ports:
21         - containerPort: 80
22           protocol: TCP
23       restartPolicy: Always
24 ---
25 apiVersion: v1
26 kind: Service
27 metadata:
28   name: pdv-frontend-svc
29 spec:
30   selector:
31     app: pdv-frontend
32   ports:
33   - port: 80
34     targetPort: 80
35     protocol: TCP

```

Para exponer el funcionamiento de kubernetes usaremos de referencia este ejemplo, dónde se encuentra el deployment, el servicio y tres pods del frontend. **No haremos referencia al despliegue de servicios como el backend o la base de datos ya que son similares entre sí, por lo que este ejemplo puede aplicarse a todos.**

1. En primer lugar, usamos la versión de la api de deployment de kubernetes apps/v1.
2. Indicaremos el tipo de archivo que estamos escribiendo es un deployment.
3. Asignaremos como metadatos el label de {app: pdv-frontend} **ESTE SERÁ NECESARIO PARA QUE LA CONEXIÓN CON EL SERVICIO SEA EXITOSA.**
4. Asignaremos como metadato el nombre del deployment

5. Indicamos que el deployment hará uso de tres réplicas donde se alojarán pods (3).
6. Indicamos que el selector que vamos a usar para enlazar los datos sea el label anteriormente descrito.
7. Creamos un template con la información que los pods estarán corriendo.
8. Usaremos la imagen personalizada y exponremos los puertos 80 de los contenedores generados.
9. Añadimos una línea separatoria, esto hace que el .yaml intérprete que lo que hay debajo de esa línea sea tratado como un nuevo fichero yaml (aunque no lo sea).
10. Definimos el servicio que apuntará a nuestros pods generados por el deployment.
11. Haciendo uso de selector le indicamos el label que tendrán los pods a los que debe apuntar.
12. Realizamos el mapeo de puertos.

Para realizar el despliegue de este .yaml, tendremos que iniciar nuestro cluster, en este caso, como estamos usando minikube tendríamos que ejecutar el comando:

```
minikube start
```

Una vez iniciado el cluster, procederemos a situarnos en la carpeta donde se encuentre el .yaml y ejecutar el comando:

```
kubectl apply -f ./pdv-frontend-deployment.yaml
```

De esta manera, ya tendremos desplegados los pods y el servicio correspondiente, si queremos asegurarnos de que todo ha salido bien podemos ejecutar los comandos:

```
kubectl get svc
kubectl get pods
```

```
C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl apply -f pdv-frontend-deployment.yaml
deployment.apps/pdv-frontend-deployment created
service/pdv-frontend-svc created

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get svc
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
kubernetes           ClusterIP   10.96.0.1     <none>       443/TCP    43s
pdv-frontend-svc     ClusterIP   10.99.212.20  <none>       80/TCP     15s

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-45v58  1/1     Running   0          21s
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          21s
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          21s
```

Así podremos asegurarnos de que todo se ha desplegado de manera correcta. Incluso podemos habilitar el addon del dashboard de minikube y asegurarnos de que todo funciona correctamente de manera más visual.

Una vez creado los pods y el servicio, solo tenemos que preocuparnos de cómo acceder a ellos de manera correcta.

Para ello, vamos a implementar lo que se conoce como “Ingress”, que nos permitirá definir una interfaz de comunicación entre las distintas partes de la aplicación, dándonos la opción de utilizar una URL personalizada como host.

```
Deployment > Y pdv-ingress.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: pdv-ingress
5  annotations:
6    nginx.ingress.kubernetes.io/rewrite-target: /
7  spec:
8    rules:
9    - host: portaldelviajero.com
10   http:
11     paths:
12     - pathType: Prefix
13       path: "/"
14     backend:
15       service:
16         name: pdv-frontend-svc
17         port:
18           number: 80
19     - pathType: Prefix
20       path: "/backend"
21     backend:
22       service:
23         name: pdv-backend-svc
24         port:
25           number: 9999
26     - pathType: Prefix
27       path: "/"
28     backend:
29       service:
30         name: pdv-mysql-svc
31         port:
32           number: 3306
```

A continuación, se procederá a dar unas indicaciones de que hace este archivo:

1. Define el tipo a Ingress y selecciona la versión de la api visible en la captura.
2. Asigna como nombre pdv-ingress
3. Asigna una anotación que reemplaza la ruta de todas las peticiones que se hagan al servicio ingress por /.
4. Define las reglas y el nombre del host.
5. Asocia un path, un servicio y un puerto para cada parte de nuestra aplicación, permitiéndonos obtener un alcance completo.

De nuevo, para desplegar este archivo deberemos ejecutar el comando:

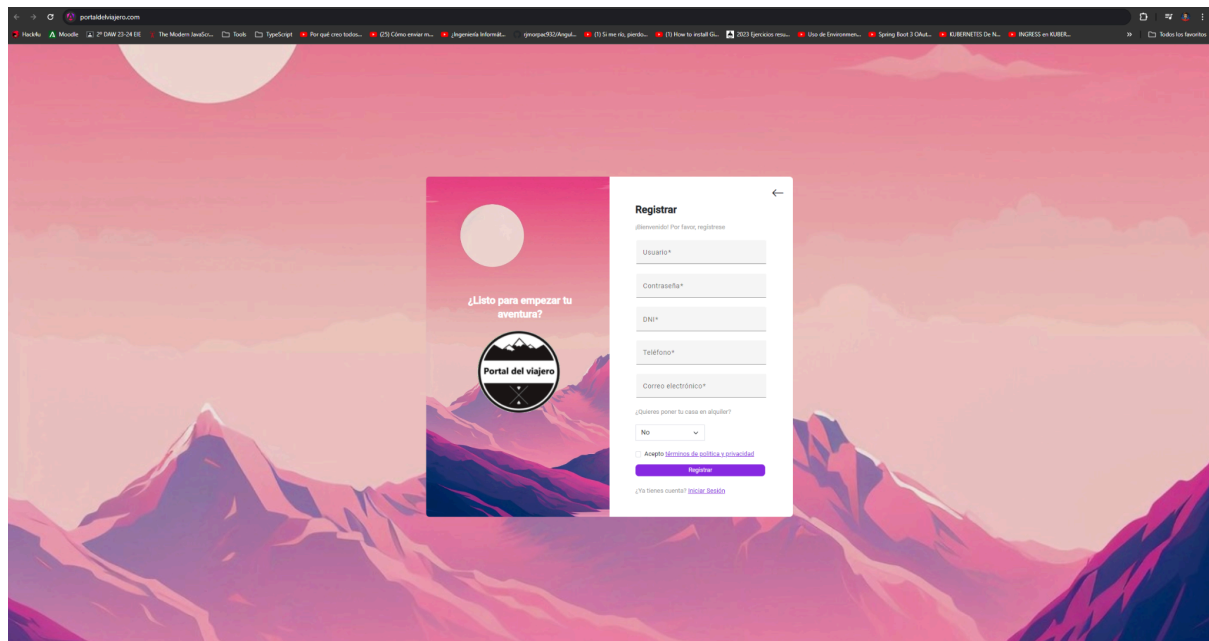
```
kubectl apply -f ./pdv-ingress.yaml
```

Para comprobar que el servicio ingress se ha desplegado correctamente podemos ejecutar el siguiente comando:

```
kubectl get ing
```

```
C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get ing
NAME          CLASS    HOSTS                ADDRESS        PORTS    AGE
pdv-ingress   nginx    portaldelviajero.com 192.168.49.2   80       42s
```

¡Ya tenemos funcionando nuestra aplicación!



Eliminar y añadir pods en kubernetes:

Un pod se encarga de correr la imagen asignada a él. Puede haber momentos dónde el servidor, por ciertas cuestiones, necesite un aumento de capacidad de carga, y puede haber otros momentos donde sobre.

Para esto, existen comandos que nos permitirán aumentar o disminuir esa cantidad de pods que corren la aplicación.

Veamos un ejemplo práctico, imaginemos que tenemos tres pods corriendo del frontend de la aplicación, ahora imaginemos que por ciertas cuestiones queremos eliminar uno de esos pods. Para ello, podrías intentar usar comandos como:

```
kubectl delete pod [nombre_pod]
```

Lo que ocurriría al ejecutar ese comando sería lo siguiente:

```
C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-bsv5g  1/1     Running   0          7s
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          17m
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          17m

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl delete pod pdv-frontend-deployment-54b4f9f699-bsv5g
pod "pdv-frontend-deployment-54b4f9f699-bsv5g" deleted

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-7b2tn  1/1     Running   0          4s
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          20m
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          20m
```

Como se puede observar, el pod se ha borrado, sin embargo, sigue habiendo tres pods de todas formas. Esto ocurre porque el ReplicaSet se encuentra configurado para tener corriendo siempre 3 pods, de manera que si alguno se cae otro se levanta instantáneamente, es una de las ventajas de Kubernetes.

Para solucionar esto cambiaremos la escala de réplicas que tiene nuestro deployment haciendo uso del siguiente comando:

```
kubectl scale --replicas=2 deployment/pdv-frontend-deployment
```

```
C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-7b2tn  1/1     Running   0          4s
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          20m
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          20m

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
pdv-frontend-deployment  3/3     3            3          23m

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl scale --replicas=2 deployment/pdv-frontend-deployment
deployment.apps/pdv-frontend-deployment scaled

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          24m
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          24m
```

De la misma manera, si lo que se quisiese fuera aumentar la escala, bastaría con asignar un número mayor al actual:

```
kubectl scale --replicas=4 deployment/pdv-frontend-deployment
```

```
C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          24m
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          24m

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl scale --replicas=4 deployment/pdv-frontend-deployment
deployment.apps/pdv-frontend-deployment scaled

C:\Users\juanp\OneDrive\Documentos\Proyecto Integrado\Deployment>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
pdv-frontend-deployment-54b4f9f699-gdn92  1/1     Running   0          27m
pdv-frontend-deployment-54b4f9f699-k5q6v  1/1     Running   0          6s
pdv-frontend-deployment-54b4f9f699-sdjm8  1/1     Running   0          27m
pdv-frontend-deployment-54b4f9f699-zbzfn  1/1     Running   0          6s
```