Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Peaker. Viajes compartidos para practicar esquí, snowboard y surf

MEMORIA

D. Álvaro Moyano Riera

TUTOR: Dña. María José Suarez Cabal

FECHA: JULIO 2020

Índice de contenidos

Índice de figuras

1. Introducción

1.1 Antecedentes

En los últimos años ha crecido el interés por realizar deportes como surf, bodyboard, esquí o escalada entre otros. Este tipo de deportes a priori se practican de manera individual, pero la experiencia mejora si se realiza en pareja o en grupo. En este tipo de actividades es frecuente tener que desplazarse al lugar donde se practiquen que generalmente son espacios naturales.

La posibilidad de practicar determinadas actividades requiere de condiciones meteorológicas apropiadas, lo que hace que los deportistas acudan a ellos en sus vehículos en un mismo espacio de tiempo. A menudo estos lugares son el punto de encuentro de personas que practican estos deportes y que en muchas ocasiones son personas de la misma ciudad o personas que se conocen por haber coincidido varias veces practicando alguna actividad.

Para personas que no dispongan de vehículo propio esta condición puede ser un impedimento para realizar estos deportes con la frecuencia que le gustaría pues dependen de una persona que disponga de vehículo para poder desplazarse. Para personas que si dispongan de vehículo el poder realizar la actividad con una o varias personas puede ser una motivación y mejorar la experiencia además de compartir viaje con una o varias personas con tus mismos intereses deportivos.

El hecho de compartir vehículo se ha normalizado en los últimos tiempos con las aplicaciones de vehículo compartido como *Uber* o *BlaBlaCar*. Para esta segunda aplicación he comprobado personalmente que el perfil de los usuarios que la utilizan es muy heterogéneo: desde jóvenes hasta adultos. Por otra parte, existen también aplicaciones para conectar a personas que tengan los mismos gustos, ya sean de ocio o deportivos, y puedan realizar estas actividades en grupo. De esta manera se cubren las necesidades de ocio que tienen las personas que se acaban de mudar a una nueva ciudad o viven en ella, pero no tienen un grupo de conocidos que compartan sus intereses.

Existen variedad de aplicaciones destinadas a cubrir esta necesidad, entre las más extendidas están Meetup o *Timpik,* pero que principalmente están orientadas a utilizarse en grandes ciudades y aprovechar los espacios deportivos o de ocio tanto públicos como privados. Otras aplicaciones cuyo objetivo es aprovechar los espacios naturales para realizar rutas en grupo como *Wikiloc*.

1.2 Motivación

El hecho de crear un producto software que ofrezca solucionas a sus usuarios en un ámbito social y económico ha servido de motivación para el desarrollo de *Peaker*, cuyo nombre es un juego de palabras Peak (cumbre en inglés) y Picker (recogedor). El nombre se basa la idea de recoger a las personas y llevarlas a la cumbre. La aplicación contará con una interfaz de usuario que proporcione una identidad única y cuyas funcionalidades sean simples e intuitivas. Las características anteriores harán que usuarios que no necesariamente estén habituados a utilizar este tipo de aplicaciones puedan desenvolverse fácilmente.

El aspecto social se basa poner en contacto a personas que comparten interés en realizar deportes y actividades en espacios naturales. Además del aspecto social para las personas jóvenes es también relevante el aspecto económico. La aplicación permitirá indicar si se compartirán los gastos que se generan, principalmente de transporte, reduciendo así los costes para los participantes del viaje compartiendo.

Para la elaboración de la aplicación se utilizará Liferay Portal, más adelante detallaremos los aspectos más relevantes de esta tecnología y su arquitectura. La elección se ha visto motivada por ser la herramienta que utilizo actualmente en mi ámbito laboral, el hecho de poder desarrollar por mí mismo una aplicación utilizando esta tecnología y poder utilizar los conocimientos que adquiero en mi trabajo y ampliarlos a través de la realización de esta aplicación, todo esto ha servido de motivación para su elección. Poder construir un producto software por mí mismo que dé solución a problemas reales de la gente con la que me relaciono y tengo contacto resulta muy satisfactorio.

1.1 Objetivo y alcance del proyecto

El propósito del presente proyecto es la creación de una aplicación web, donde personas que realicen actividades deportivas, que requieren desplazamiento a un espacio natural, contacten con personas con gustos deportivos similares. De esta manera podrán realizar la actividad juntos y compartir los gastos que se produzcan. Esto genera un doble beneficio: social y económico; para los participantes de la actividad.

Respecto a los beneficios sociales hacen referencia a poder conocer a personas con tus mismos gustos y poder realizar una actividad deportiva en grupo, con personas que por circunstancias no dispongan de vehículo, no conozcan personas que practiquen dicho deporte o acaben de llegar a una ciudad. En cuanto al económico, está orientado generalmente para personas jóvenes que no dispongan de independencia económica y que les pueda resultar un problema el realizar una actividad repetir varias veces por semana que requieran de desplazamiento. Este problema se soluciona dividiendo los gastos de transporte y resto de gastos que se puedan generar realizando la actividad entre los participantes.

La aplicación será capaz de llevar a cabo una gestión de viajes y solicitudes, de manera que los usuarios que dispongan de vehículo puedan organizar una actividad deportiva en un espacio natural destinado para ello. Los viajes tendrán un itinerario definido y los usuario podrán consultar la información como descripción de la actividad a realizar, planificación horaria, si se van a compartir gastos entre otros datos y otros puedan solicitar el unirse para realizar el deporte juntos y poder compartir los gastos. Proporcionará los mecanismos necesarios para la gestión de las solicitudes enviadas al organizador de un viaje de manera que pueda aceptarlas o rechazarlas además de poder consultar las solicitudes enviadas. Cuando una solicitud sea aceptada la aplicación notificará la resolución al usuario involucrado vía correo electrónico o aplicación de mensajería móvil para que mantengan una comunicación fluida fuera de la aplicación.

Peaker será una aplicación web que utiliza *Liferay Portal 6.2* como entorno donde se integran los componentes modulares desarrollados en Java llamados portlets, que son contenidos y gestionados por el portal. También permite tanto personalización visual del portal con el desarrollo de temas de apariencia como del código nativo del portal. Además de proporcionar mecanismos de seguridad, balanceado de carga y gestión de contenidos. Cada uno de los portlets se puede considerar una aplicación web en sí misma con una interfaz propia, que atienden las solicitudes de un cliente web

generado contenido dinámico y que exponen servicios para que la información que manejan pueda ser consumida y gestionada por otros portlets.

Con el fin de cumplir con los objetivos del proyecto los usuarios de la aplicación web deben de poder realizar las siguientes funciones:

- Registrarse y acceder a la aplicación proporcionando unas credenciales.
- Gestionar y configurar la información de su perfil.
- Añadir la información necesaria para registrar uno o más de vehículos en la aplicación y poder ser utilizado para organizar viajes.
- Crear un viaje, estableciendo el itinerario a través de un mapa y proporcionando la información necesaria para definir una planificación horaria una descripción de la actividad y un vehículo. A la hora de establecer el itinerario el usuario visualizará en el mapa la posición de lugares populares donde realizar actividades llamados *Spots.*
- Buscar viajes, la información de cada viaje se utilizará para la creación de buscadores que segmenten los diferentes tipos de viaje para facilitar al resto de usuarios de la aplicación el encontrar los que más se adecuen a sus necesidades.
- El usuario organizador del viaje podrá gestionar las peticiones recibidas, aceptándolas o rechazándolas.
- Enviar solicitudes a los viajes creados en la aplicación y consultar el estado en la que se encuentran además de recibir información sobre la resolución de su solicitud. Como solicitantes también podrán cancelar si se dan las condiciones adecuadas

2. Documentos del proyecto

La elaboración de los documentos de que complementan a la *Memoria* se ha estructurado siguiendo los procesos definidos en la metodología Métrica Versión 3. Estos procesos se dividen en actividades y a su vez en tareas, los productos generados se mejoran y amplían a medida que se elaboran dichos documentos. Los apartados de los que se componen la memoria son los siguientes:

- **Elaboración del Estudio de Viabilidad del Sistema**: se realiza la definición del alcance del sistema, un estudio de la situación actual analizando sistemas existentes similares al que se va a desarrollar, la definición preliminar del catálogo de requisitos del sistema y una estudio y valoración de alternativas con el fin de encontrar la solución que más se adecue. Los apartados de este documento son los siguientes:

  1. Establecimiento del alcance del sistema
  2. Estudio de la solicitud actual
  3. Definición de requisitos
  4. Estudio y valoración de alternativas
  5. Selección de la solución

- **Análisis del sistema de información:** se delimita el alcance del sistema, de esta manera se define un catálogo de requisitos generales donde se recogen de forma detallada los requisitos funcionales del sistema además de una especificación de la interfaz de usuario. Al tratarse de un sistema orientado a objetos se realiza una descripción de los diferentes módulos que componen el sistema también se elaboran los modelos de casos de uso y de clases. Todo lo anterior proporciona una especificación detallada del sistema

  1. Catálogo de Requisitos
  2. Descripción de Subsistemas de Análisis
  3. Análisis de Casos de Uso
  4. Análisis de Clases
  5. Definición de Interfaces de Usuario
  6. Análisis de Consistencia
  7. Especificación del plan de pruebas

- **Diseño del Sistema de Información:** se define la arquitectura del sistema, se finaliza la elaboración del catálogo de requisitos del sistema. Se determina el comportamiento entre las diferentes clases y el diseño de la interfaz de usuario obteniendo así una definición del entorno tecnológico que dará soporte a los diferentes subsistemas.

    1. Diseña de la Arquitectura del Sistema
    2. Especificación de Subsistemas
    3. Modelo de Datos
    4. Diseño de Interfaz de Usuario
    5. Diseño del Plan de Pruebas

- **Plan de pruebas:** se presenta un catálogo con las pruebas a realizar y los resultados esperados de la ejecución de estas.

- **Construcción del sistema de información:** se obtiene el producto software compuesto por el código fuente de los diferentes módulos que forman el sistema. También se realiza una carga inicial de datos para llevar a cabo la ejecución de distintos tipos de pruebas.

- **Aceptación del sistema:** se realizan pruebas de implantación que tienen mayor complejidad que las unitarias y se asemejaran al uso real que se dará al sistema.

3. Estudio de Viabilidad del Sistema

1. Establecimiento del alcance del sistema

1.1 Estudio de la solicitud

Se desarrollará una aplicación web donde los usuarios, previo registro y autenticación, puedan personalizar su perfil que consta de la información básica de contacto y tengan la capacidad de añadir la información de su vehículo. Los usuarios que dispongan de vehículo tendrán la capacidad de crear viajes definiendo un itinerario de viaje a través de una aplicación de mapas como *Google Maps* y una planificación temporal. También tendrá que completar un formulario con la información de la actividad a realizar. Toda esta información será utilizada por diferentes buscadores de viajes que segmenten dicha información y faciliten a los usuarios la búsqueda de los viajes que se adapten mejor a sus necesidades. El sistema también permitirá a los usuarios gestionar las solicitudes enviadas o recibidas y notificará su resolución para que si fuera necesario los usuarios puedan mantener una comunicación fuera de la aplicación web.

Los usuarios de esta aplicación serán personas de más de 18 años que estén dispuestos a compartir vehículo para realizar actividades deportivas en espacios naturales. Se cubren dos necesidades de los usuarios, principalmente un aspecto social por conocer a nuevas personas que compartan tus mismos gustos e intereses. En segundo lugar y de manera opcional una necesidad económica solucionada compartiendo los gastos de desplazamiento y demás gastos que se puedan generar.

1.2 Alcance del sistema

Se definirá el alcance general del sistema abordando el alcance de cada uno de los módulos que lo forman. En primer lugar, el módulo de gestión de registro, los usuarios han de ser capaces de registrarse en el sistema y acceder a él mediante un proceso de login. Un módulo de gestión de usuarios donde los usuarios puedan completar la información de su perfil. La información proporcionada tiene que ser accesible y poder ser modificada por el usuario. Este módulo incluye a su vez la gestión de vehículos, donde se podrá guardar la información de su vehículo y adjuntar fichero de imagen. Esta información al igual que la de usuario ha de ser accesible y modificable.

El módulo de gestión de viajes se encargará tres tareas fundamentales creación de viajes para usuarios con vehículos registrados. La interfaz de creación ha de ser simple y ha de tener integrada una aplicación de mapas, preferiblemente *Google Maps*, para poder definir un

itinerario de viaje a través de la posición de marcadores. Se ha de incluir un buscador de viajes que segmenten los viajes según su origen, actividad a realizar y periodo de fechas entre los que se realice el viaje.

La información sobre los viajes que muestran los buscadores se mostrará de dos formas, en un mapa y en un listado de viajes. En el mapa se mostrará la información del itinerario de los viajes a través de marcadores. Los destinos de los viajes podrán ser elegido libremente por el usuario, pero también se le ofrecerán un listado de destinos populares llamados *Spots* propios de cada zona geográfica. También se mostrarán los viajes encontrados a través de un listado con una interfaz que muestre la información más relevante de cada viaje permitiendo a los usuarios con un vistazo decidir si el viaje se adapta a sus requisitos.

Al publicar un viaje se creará una ficha del viaje, mediante un mapa se mostrará la información del itinerario y también se mostrará la información detallada del viaje. Se han de consumir los servicios de alguna API que proporcione datos meteorológicos del lugar de destino, con el fin de ofrecer la mayor cantidad de información al solicitante. Por último, desde la ficha del viaje los usuarios podrán enviar solicitudes al organizador para unirse al viaje y se notificará el envío a través de correo electrónico .

Por último, el módulo de gestión de solicitudes permitirá al usuario acceder a la información de las solicitudes tanto enviadas como recibidas para cada uno de los viajes. Las enviadas podrán ser canceladas por el solicitante si se cumplen determinadas situaciones que se explicarán más adelante. Para las solicitudes recibidas el usuario tendrá la capacidad de aceptarla o rechazarla en cualquiera de los casos se notificará la resolución al usuario implicado. Se debe de garantizar que se cumple con las restricciones de número de plazas establecidas en la creación del viaje.

2. Estudio de la situación actual

2.1 Valoración de la situación actual

En la actualidad existen aplicaciones que abarcan los dos principales ámbitos de nuestro sistema, poner en contacto a gente con los mismos intereses y la utilización de vehículo compartido para abaratar costes de desplazamiento. Actualmente no existe ninguna aplicación web o móvil que abarque estos dos ámbitos en un mismo sistema. Esto puede deberse a que el acto de compartir vehículo con otras personas es relativamente nuevo y extendido en los últimos años pero que en países como España ha producido resultados satisfactorios, para personas de todo tipo de edades. También que la popularidad de la aplicación en cada provincia depende la diversidad de espacios naturales destinados a actividades deportivas.

Las personas que realizan actividades que requieren desplazarse a ciertos lugares para poder practicarlas y no dispongan de vehiculo propio, tendrán que utilizar transporte público. Esta opción en muchas ocasiones no es viable ya que este tipo de transporten no llega a determinados lugares de la costa o la montaña y transportar el equipamiento necesario para realizar la actividad puede no ser muy práctico. Para estas personas, lo comentado anteriormente puede ser un aliciente muchas veces para no practicar dicho deporte. Sin embargo, puede haber otras personas que practican el mismo deporte y disponen de vehículo con plazas libres y estarían dispuestos a realiza la actividad junto a otras personas. La intención de la aplicación es servir como lugar virtual de confluencia para estos deportistas y facilitar el hecho de conocerse y ponerse en contacto.

2.2 Identificación de usuarios participantes

Todos los usuarios registrados en la aplicación podrán realizar las mismas acciones en la aplicación. En el contexto de creación de un viaje los tipos de usuarios que contempla el sistema son dos:

- **Usuario organizador:** persona que dispone de vehículo y que es el creador del viaje, definirá el itinerario y toda la información necesaria. Recibirá las solicitudes del resto de usuarios y decidirá su resolución. Deberá contactar con los usuarios cuyas solicitudes hayan sido aceptadas.

- **Usuario solicitante:** persona que puede disponer o no de vehículo, a través de la información proporcionada en los buscadores decidirá que viaje se adecua mejor a sus necesidades y enviará solicitudes a los usuarios organizadores. Si su solicitud es aceptada podrán contactar con el organizador.

2.3 Descripción de los sistemas existentes

En este apartado se analizarán las aplicaciones existentes cuyo alcance incluye los aspectos a cubrir por nuestra aplicación que son el desplazamiento utilizando vehículo compartido con el fin de minimizar los gastos de desplazamiento y el ser un punto de encuentro virtual donde conocer a personas con los mismos intereses con el fin de realizar actividades deportivas.

2.3.1 Aplicaciones de vehículo compartido

- **Uber:** la aplicación más popularizada mundialmente para el uso de vehículo compartido, que pone a disposición de hasta un máximo de 4 usuarios un vehículo con conductor para desplazarse a un destino previamente establecido. El coste del desplazamiento se establece previamente y se divide entre los pasajeros.

  En el sistema a desarrollar, un usuario pondrá a disposición de otros su vehículo y podrá opcionalmente solicitar a los pasajeros la aportación de parte de los gastos de desplazamiento

- **BlaBlaCar:** en España es la aplicación más aceptada y que más ha normalizado el hecho compartir vehículo para desplazarse a un destino, el usuario organizador establece el itinerario del viaje y la tarifa a abonar por cada pasajero. Está orientada principalmente a disminuir los costes de desplazamiento de viajes entre ciudades.

  Nuestro sistema también tiene como objetivo disminuir los costes económicos, pero no es algo prioritario puesto que se puede dar el caso que el usuario organizador asuma los costes de movilidad para facilitar que otros usuarios se unan a su viaje además por el hecho de realizar una actividad deportiva se creará una relación de amistad entre los participantes del viaje.

2.3.2 Aplicaciones para la realización de actividades en grupo

- **MeetUp:** aplicación de uso muy extendido en grandes ciudades cuyo objetivo es poner en contacto a personas para que realicen actividades culturales, sociales o deportivas. Los usuarios se pueden unir a grupos destinados a realizar un tipo de actividades dentro de la misma ciudad.

  Las actividades deportivas que realizar a través de nuestra aplicación se llevan a cabo fuera de las ciudades o en otras ciudades de manera que necesariamente los usuarios requieran de desplazamiento en un vehículo. Los buscadores de viajes facilitarán a los usuarios el encontrar viajes que se adapten a sus intereses. A pesar de que el ámbito de esta aplicación difiere del nuestro, añadir a nuestra aplicación funcionalidades como la de creación de grupos pueden ser interesante.

- **Wikiloc:** aplicación hibrida de origen español donde los usuarios se pueden publicar las rutas que van a realizar como puntos de interés dentro de ellas. Las rutas pertenecen a diferentes categorías como senderismo, ciclismo, kayak entre otras. El punto fuerte es el mapa satélite integrado con toda la información sobre la ruta como duración, distancia o desnivel.

  Entre nuestros objetivos estará la personalización del mapa integrado en el buscador para que proporcione de manera simple y visual toda la información sobre los itinerarios de los viajes.

3. Descripción de funcionalidades

A continuación, se describirá las funcionalidades principales que el sistema debe de proporcionar a los usuarios y que se especificarán en detalle en los documentos de análisis y diseño del sistema.

Acceso

Para acceder a la aplicación el usuario deberá haberse registrado en el sistema proporcionando la siguiente información nombre, correo electrónico y contraseña. El correo ha de ser único en el sistema.

Una vez registrado el usuario podrá acceder utilizando sus credenciales, correo electrónico y contraseña. En caso de haber olvidado la contraseña el usuario podrá restablecerla.

Gestión del perfil de usuario

Una vez haya accedido al sistema, el usuario tendrá acceso a su perfil donde deberá complementar la información del registro con la siguiente: provincia, comunidad autónoma y opcionalmente número de teléfono y una imagen. Toda la información excepto el correo electrónico ha de ser modificable.

El usuario podrá añadir uno o más vehículos detallando su marca, modelo, color, tipo de combustible e información adicional además de poder adjuntar una imagen. En cualquier momento un usuario podrá editar la información o borrar el vehículo

Creación de viajes

Los usuarios que tengan registrado al menos un vehículo podrán crear viajes donde definirán en un mapa el itinerario de este a través de diferentes marcadores, al menos el origen y el destino serán obligatorios. El resto de información del viaje como deporte a realizar, descripción de la actividad, planificación horaria y vehículo a través de un formulario.

Tras la creación del viaje se le mostrará una ficha con toda la información de este tal y como la verán el resto de los usuarios, la información no se podrá modificar. El usuario podrá cancelar el viaje siempre que lo haga antes de la fecha de realización.

Buscadores de viajes

El sistema dispondrá de buscadores de viajes para facilitar a los usuarios encontrar viajes que se adapten a sus necesidades. Habrá tres tipos de buscadores: por provincia de origen, por actividad y por rango de fechas. Todos los buscadores mostrarán la información de dos formas, en el mapa se indicará a través de marcadores los destinos de los viajes y desde ellos se podrá acceder a la información detallada de cada viaje; y a través de un listado visualmente atractivo la información más importante de cada viaje además de poder consultar su itinerario en el mapa y acceder a la información del viaje.

Envío de solicitudes

Las solicitudes se enviarán a través de la ficha de información detallada de cada viaje. El sistema deberá realizar las comprobaciones necesarias para garantizar que no se envían solicitudes duplicadas y no se excede el número máximo de solicitudes además de notificar a los usuarios implicados vía correo electrónico.

Gestión de solicitudes

Los usuarios podrán consultar tanto las solicitudes que han enviado como las que cada uno de sus viajes ha recibido. Las solicitudes enviadas se podrán cancelar siempre que el organizador no la haya aceptado o rechazado, si la ha aceptado se le mostrará al solicitante la información de contacto del organizador y recibirá un correo informándole. Los viajes que haya creado un usuario podrán recibir como máximo 10 solicitudes, pero se podrán aceptar tantas como plazas haya indicado en el viaje. Siempre que se acepte o rechace una solicitud se enviará un correo electrónico al solicitante, en el caso que sea aceptada se le informará de la forma de contactar con el organizador.

4. Estudio y valoración de alternativas

Arquitectura de desarrollo

El desarrollo de los componentes del sistema se puede llevar a cabo siguiendo dos enfoques: desarrollo de microservicios o desarrollo monolítico.

Siguiendo una arquitectura de microservicios los módulos serán independientes, pero se comunicarán entre sí exponiendo servicios para que puedan ser consumidos por el resto de los módulos.

- **Pros:** La independencia de módulos software facilitan el mantenimiento de código y el control de errores se realizan de forma más efectiva. Proporciona mayor escalabilidad y flexibilidad a la hora de implementar nuevas funcionalidades. Desarrollo de código ágil.

- **Contras:** Mayor complejidad a la hora de configurar e implementar los componentes. Al tratarse de componentes distribuidos se consumen más recursos y la ejecución de pruebas es más compleja.

Por otra parte, si se siguiera un enfoque de desarrollo monolítico todas las funcionalidades del sistema estarían definidas en un único componente, esto deriva en un acoplamiento alto de las diferentes unidades software. Los sistemas construidos que siguen esta arquitectura son eficientes y proporcionan una buena gestión de errores.

- **Pros:** Los sistemas construidos son eficientes. La ejecución de pruebas es más sencilla. Menor consumo de recursos que los microservicios. Al no comunicarse con otros componentes existen menos puntos de fallo. Al tratarse de un único componente las tareas de configuración y puesta en marcha de la aplicación se realizan de manera rápida.

- **Contras:** Al desarrollar un único componente cada cambio supone tener que construir y desplegar todas las funcionalidades, aunque no se hayan modificado, las tareas de desarrollo y mantenimiento de software no son tan ágiles como en los microservicios. Este tipo de arquitectura dificulta el mantenimiento y escalabilidad del sistema. Un error puede suponer la caída total del sistema.

Tecnologías lado del cliente

Respecto a las tecnologías que se utilizarán en el lado del cliente, al tratarse de una aplicación web es imprescindible el uso de *JavaScript* y *CSS* para definir una interfaz de usuario que cumpla con los requisitos establecidos. Se utilizará *Ajax* para realizar llamadas asíncronas al servidor y así mejorar la usabilidad de la aplicación.

Tecnologías lado del servidor

En cuanto al lenguaje de programación del lado del servidor los más aceptados son *PHP* y *Java*. Ambos tienen características compartidas como buen rendimiento, mecanismos de concurrencia o portabilidad, el código se puede ejecutar en si el entorno de una máquina virtual Java, por el contrario, PHP es un lenguaje interpretado que solo depende del servidor web para ser ejecutado.

A continuación, se valorará los pros y contras del uso de Java:

- **Pros:** Java proporciona mejores mecanismos de seguridad y concurrencia, la utilización de multihilos da mejor rendimiento que el uso de multiprocesos de PHP. Dispone de herramientas de empaquetado y despliegue del código como ANT o Maven. Las herramientas de desarrollo Java como Eclipse dispone de depuración de código

- **Contras:** Por lo general el coste de desarrollo de un proyecto en Java es superior a un proyecto en PHP. El desarrollo de una aplicación Java es más complejo.

Los pros y contras de PHP son los siguientes:

- **Pros:** PHP es más fácil de aprender y flexible lo que implica que los desarrollos son más rápidos y menores costes de desarrollo. Es un lenguaje más ligero que proporciona mejor rendimiento y velocidad.

- **Contras:** El rendimiento empeora en aplicaciones con muchos usuarios concurrentes. Aplicaciones menos seguras y más vulnerables. Es más difícil integrar componentes ya desarrollados y de libre distribución.

Si bien es cierto que es *PHP* es un tipo de lenguaje más fácil de aprender y que permite desarrollos rápidos no es un factor determinante para su elección, el uso de frameworks como *Spring* para Java o *Laravel* para *PHP* proporcionan patrones de arquitectura que permiten desarrollos agiles en ambos lenguajes. A continuación, se describirán las características más relevantes de estos frameworks:

- **Spring:** las aplicaciones desarrolladas son flexibles y escalables. Las unidades software que se generan tienen bajo acoplamiento y se utiliza inyección de dependencias, esto proporciona un fácil mantenimiento y reutilización de código además de minimizar el impacto en el resto del código si se produce un cambio. Dispone de módulos como Spring Security que proporciona mecanismos de seguridad y autenticación para los servicios web o Spring Cloud proporciona herramientas para la construcción de patrones distribuidos, llamada de servicios o balanceo de carga

- **Laravel:** Utiliza *Blade* como sistema de plantillas para la creación de vistas donde se ejecuta código PHP, al igual que las JSP de Spring. El sistema de gestión de rutas de Laravel es simple y fácil de implementar. *Eloquent* es el ORM (mapeo objeto-relacional) que proporciona mecanismos sencillos de acceso a base de datos. Cuenta con una gran comunidad y documentación.

Base de datos

La base de datos ha de ser de tipo relacional puesto que el modelo de datos lo requiere. Se utilizará una base de datos de código abierto, teniendo en cuenta esto existen varias alternativas como: *Open Source* como *MySQL*, PostgreSQL, *MariaDB* o *MongoDB*.

A continuación, se valorarán los pros y contras del uso de MySQL vs PostgreSQL.

- **MySQL** Ideal para base de datos pequeñas y medianas ofreciendo mejores resultados en aplicaciones con poca concurrencia en operaciones de escritura y muchas lecturas en BD. Utilización de vistas, que son tablas constituidas por los resultados de varias consultas, para mejorar el rendimiento a la hora de gestionar bases de datos de gran tamaño. Bajo consumo de recursos. A pesar de ser propiedad de Oracle dispone de soporte comunitario, así como comercial. MySQL dispone de una de aplicación de interfaz gráfica llamada *MySQL Workbench* que facilita en gran medida la visualización y manejo de base de datos.

- **PostgreSQL:** Orientado al manejo de base de datos grandes y consultas complejas. Ofrece mejores mecanismos para consultas concurrentes y protección de la integridad de los datos. Altamente escalable dando la posibilidad de crear tipos de datos, operadores y tipos de índice. Ideal para entornos que requieren de alta disponibilidad. Es totalmente de código libre e impulsada por la comunidad, al contrario que *MySQL* que es propiedad de Oracle. Al igual que *MySQL*, dispone de una herramienta con interfaz gráfica llamada *pgAdmin* para facilitar la gestion de la base de datos.

5. Selección de solución

Se ha optado por la utilización de *Liferay Portal*, un framework de código abierto escrito en Java para el desarrollo de aplicaciones web. Proporciona mecanismos que nos facilitan en gran medida la gestión de usuarios, permisos y roles. También dispone de herramientas que facilitan la gestión de servidores de correo o posicionamiento web.

Permite la integración y gestión de componentes modulares escritos en Java llamados, *portlets,* que son pequeñas aplicaciones web integradas en el portal y que sirven contenido de forma dinámica. *Liferay* actúa como contenedor de portlets proporcionándoles un entorno de ejecución y controlando el procesamiento de solicitudes y su ciclo de vida. Cualquier funcionalidad o interfaz propia de *Liferay* es personalizable a través de *Hooks* componentes que personalizan o extienden las funcionalidades de los componentes propios de *Liferay.*

Para facilitar el desarrollo de los portlets se utilizará *Maven*, una herramienta software que facilita la construcción y despliegue de los portlets además de facilitar en gran medida la gestión de dependencias y librerías. También proporciona mecanismos como *Service Builder* para la creación de servicios web de manera sencilla y rápida a través de un descriptor *XML*.

5.1 Servidor web

Es el componente que recibe las peticiones *HTTP* de los navegadores web y les da respuesta. Durante la etapa de construcción del sistema se instalará en el equipo de desarrollo. Para la implantación de la aplicación en un entorno de producción se montará en un servidor dedicado ya que proporciona mecanismos de disponibilidad, mantenimiento y respaldo para la base de datos.

Liferay Portal se puede montar sobre la mayoría de servidores web como *Tomcat*, *Glassfish* o *Websphere* entre otros, pero nos decantaremos por Apache Tomcat pues existen *bundles* de este servidor web con *Liferay* instalado y facilita las tareas de configuración que es uno de los aspectos más tediosos en la puesta en marcha del servidor. Apache Tomcat que actúa como contenedor de servlets y da soporte a Java Server Pages (JSP) además de ser es el más extendido para aplicaciones basadas en Java y ser el más conocido por los desarrolladores. Es ideal para el desarrollo de la aplicación por su fácil configuración e instalación y en entornos de producción tiene un buen rendimiento para los niveles de tráfico y disponibilidad que manejaremos.

5.2 Base de datos

Se ha optado por la utilización de MySQL como sistema de gestión de base de datos relacional, a pesar de que estrictamente no es de código abierto por ser propiedad de Oracle y disponer de módulos de código cerrado es considerada como tal por tener una licencia de código abierto. Es la base de datos de open source más popular entre los desarrolladores por su fácil utilización, configuración e instalación. Es compatible con sistemas operativos Windows, Linux y Unix. *MySQL* proporciona mecanismos de escalabilidad, seguridad e integridad de datos necesarios para los volúmenes de operaciones que se van a manejar. También ofrece buenos tiempos de respuesta en operaciones CRUD y consumo de recursos en comparación con otras bases de datos.

5.3 Lenguajes de programación

Como se ha mencionado anteriormente, las tecnologías que se utilizarán del lado del cliente serán Bootstrap para definir la apariencia de la aplicación. Jquery y AJAX proporcionarán las funciones necesarias para mejorar la interacción del usuario con la aplicación web a través del control de eventos o creación de animaciones entre otras funcionalidades.

En el lado del servidor, Java será el lenguaje de programación elegido puesto que los componentes modulares a desarrollar que serán integrados en *Liferay Portal* han de estar basados en Java. La utilización de *Spring MVC* proporciona un patrón Modelo-Vista-Controlador que ofrece desarrollo flexible y altamente configurable y dota de una estructura a los componentes conocida por el desarrollador.

4. Análisis del Sistema de Información

1. Catálogo de requisitos

Basándonos en la descripción de requisitos de usuario realizada en el documento de *Estudio de Viabilidad del Sistema* se elaborará el catálogo de requisitos del sistema. En este producto se identificarán los requisitos mediante un identificador, se realizará una breve descripción de estos y se les priorizará. Se establecerán los requisitos para cada uno de los subsistemas, los cuales se definirán a continuación. Finalmente se validará qué subsistema cubrirá cada uno de los requisitos definidos.

1.1 Requisitos funcionales

Identificador	Descripción
RF 1	**Registro de usuarios e inicio de sesión**
RF1.1	El sistema permitirá al usuario deberá introducir los siguientes datos; nombre, apellidos, correo electrónico y contraseña para ser registrado en el sistema.
RF 1.2	El usuario accederá al sistema a través de un proceso de acceso a la aplicación donde las credenciales son: correo electrónico y contraseña.
RF 1.3	Se permitirá al usuario recuperar su contraseña si la ha olvidado enviándole un email a su dirección de correo.
RF 2	**Gestión del perfil del usuario y de vehículos**
RF 2.1	El usuario dispondrá de una página de perfil donde su información es editable a excepción del correo electrónico y podrá añadir una imagen de perfil.
RF 2.2	El sistema permitirá al usuario registrar varios vehículos, se debe de proporcionar la siguiente información: marca modelo, plazas, tipo de combustible, color, información extra y una imagen del vehículo.
RF 2.3	El usuario podrá elegir como quiere que se contacte con él una vez haya aceptado una solicitud, mediante aplicación de mensajería o correo electrónico.
RF 2.4	El sistema debe incluir todos las marcas de vehículos y sus modelos correspondientes para evitar que se puedan introducir tipos de vehículos inexistentes.
RF 2.5	El usuario podrá editar la información e imagen del vehículo.
RF 2.6	El usuario podrá eliminar cualquiera de sus vehículos.
RF 3	**Creación y gestión de viajes**
RF3.1	Si un usuario sin vehículo intenta crear un viaje, se mostrará un aviso al usuario y no se le permitirá crear el viaje.

RF3.2	En la creación del viaje el usuario podrá establecer en un mapa su itinerario de viaje mediante distintos marcadores donde el origen y el destino son obligatorios.
RF3.3	El usuario podrá determinar el tipo de actividad a realizar, dentro de unas previamente establecidas, divididas en dos categorías: costa y montaña.
RF3.3	El usuario podrá definir la siguiente información durante la creación del viaje: descripción sobre la actividad, fecha de la actividad, hora de salida y llegada e información extra.
RF3.4	El sistema permitirá al usuario seleccionar uno de sus coches registrados para realizar la actividad
RF3.5	Tras la creación del viaje el sistema mostrará al usuario una ficha con toda la información del viaje.
RF3.6	El sistema deberá consumir los servicios de una API de información meteorológica cuya información será formateada y mostrada a los usuarios en la ficha del viaje.
RF 3.7	El sistema permitirá al usuario cancelar el viaje en cualquier momento y enviará un correo electrónico informando a los usuarios que se hayan unido o hayan enviado una solicitud.
RF 3.8	El sistema proporcionará a los usuarios distintos tipos de buscadores de viajes, en los que se muestre la información de los viajes y permitirá a los usuarios consultar la ficha de información del viaje.
RF 3.9	En los buscadores el usuario podrá ver el itinerario de cada viaje en el mapa, teniendo la opción de que se vuelvan a mostrar los marcadores destino de todos los mapas sin necesidad de recargar el buscador.
RF 3.10	El sistema permitirá a los usuarios enviar como máximo una solicitud por viaje, si el usuario ya ha enviado una se le mostrará un aviso.
RF 3.11	Se podrán enviar un máximo de 10 solicitudes a cada viaje.
RF 3.12	A través de un dashboard el usuario podrá acceder a la página de Crear Viaje o al Buscador de viajes
RF 4	**Sistema de gestión de solicitudes**
RF 4.1	El sistema debe de permitir a los usuarios las solicitudes enviadas a otros viajes y las solicitudes recibidas de los viajes activos.
RF 4.2	Si una solicitud no ha sido resuelta por el organizador del viaje, el solicitante podrá cancelar la solicitud.
RF 4.3	El sistema mostrará al usuario un listado con los viajes que ha creado pudiendo acceder a las solicitudes recibidas y a la ficha de información del viaje. También informará de los viajes con solicitudes pendientes de resolución.
RF 4.4	El sistema informará al usuario de los viajes activos con solicitudes pendientes.
RF 4.4	El sistema mostrará al usuario las solicitudes recibidas para cada viaje.
RF 4.5	Se permitirá al usuario aceptar o cancelar una solicitud, esta acción será definitiva y se notificará al solicitante mediante correo electrónico de la resolución.
RF 4.6	Se permitirá al usuario rechazar una solicitud, esta acción será definitiva y se notificará al solicitante mediante correo electrónico de la resolución.
RF 4.7	Si la solicitud se ha aceptado, se mostrará a ambos usuarios el método de contacto elegido por ambos para que establezcan una comunicación fuera de la aplicación.

1.2 Requisitos no funcionales

Identificador	Descripción
RNF 1	**Eficiencia**
RNF1.1	El sistema tiene que ser capaz de gestionar grandes volúmenes de usuarios con sesiones concurrentes.
RNF 1.2	Las acciones de modificación de información de la base de susceptibles de ser realizadas por varios usuarios deben de tener carácter transaccional.
RNF 1.3	Las funcionalidades del sistema que ejecuta el usuario deben de dar respuesta menor de 5 segundos.
RNF 2	**Seguridad**
RNF 2.1	Las comunicaciones externas tendrán que realizarse de manera seguro.
RNF 2.2	Las contraseñas de los usuarios deben de estar almacenadas en el sistema codificadas por algún algoritmo criptográfico.
RNF 2.3	Todos los módulos que componen el sistema deben de tener una copia de seguridad en un software de control de versiones.
RNF 2.4	Se realizará periódicamente una copia de respaldo de la información multimedia de los usuarios.
RNF 3	**Usabilidad**
RNF 3.1	El sistema debe de proporcionar al usuario mensajes de información y de error.
RNF 3.2	El interfaz del sistema debe de ser intuitiva para facilitar la curva de aprendizaje de los usuarios.
RNF 3.3	El usuario podrá revertir cualquiera de las que pueda realizar.
RNF 3.4	El sistema se comunicará con el usuario correo electrónico para mantenerle informado de los resultados de determinadas acciones.

2. Descripción de subsistemas de análisis

Basándonos en el catálogo de requisitos previamente definido y teniendo en cuenta que la definición de los modelos de clases y datos pueden implicar ajustes, se definirán los subsistemas que componen la aplicación y se elaborará un gráfico para representarlos de manera visual y facilitar la comprensión del sistema de información. Por último, se realizará una comprobación de los requisitos funcionales a cumplir por cada uno de ellos de manera que todos se encuentren cubiertos

- **Subsistema de Registro y Acceso**: se encarga de dos funcionalidades básicas del sistema, registro de nuevos usuarios en el sistema guardando su información para posteriormente poder validar las credenciales de acceso y permitir al usuario acceso a la parte privada de la aplicación. Se cubren las funcionalidades definidas en el RF1.

- **Subsistema de Gestión de Perfil:** se encarga de gestionar la información personal del usuario y los aspectos de configuración disponibles. Se cubren parte de las funcionalidades RF2, las relacionadas con los usuarios.

- **Subsistema de Gestión de Vehículos:** proporciona al usuario funcionalidades para gestionar sus posibles vehículos permitiéndole crear un nuevo vehículo, editar la información o borrarlo del sistema. Se cubren el resto de los requisitos definidos en RF2, los relacionados con la gestión de vehículos.

- **Subsistema de Creación de Viajes:** ofrece al usuario las funcionalidades necesarias para la creación de viajes, permitiéndole establecer un itinerario de viaje y definir las características del viaje para posteriormente registrarlo en el sistema y hacerlo visible al resto de usuarios. Se cubren los requisitos definidos en el apartado RF3 que implican a la creación de viajes

- **Subsistema de Búsqueda de viajes:** proporciona las funcionalidades necesarias para que los usuarios puedan buscar viajes estableciendo unos requisitos. Los viajes encontrados se mostrarán al usuario a través de una interfaz que ofrezca información de manera simple y visual. Permitiéndoles acceder a la información completa de cada viaje y realizar acciones como enviar solicitudes o cancelar el viaje. Cubre las funcionalidades definidas en RF3 que se refieren a la búsqueda de viajes.

- **Subsistema de Gestión de solicitudes:** las funciones a llevar a cabo por son permitir a los usuarios resolver las solicitudes enviadas o recibidas por otros usuarios. La resolución de una solicitud puede ser de tres maneras distintas: aceptándola, rechazándola o cancelándola. También establece una comunicación con los usuarios a través de email informándoles del estado de sus resoluciones pendientes. Se cubren las funcionalidades definidas en RF4.

2.1 Diagrama de subsistemas

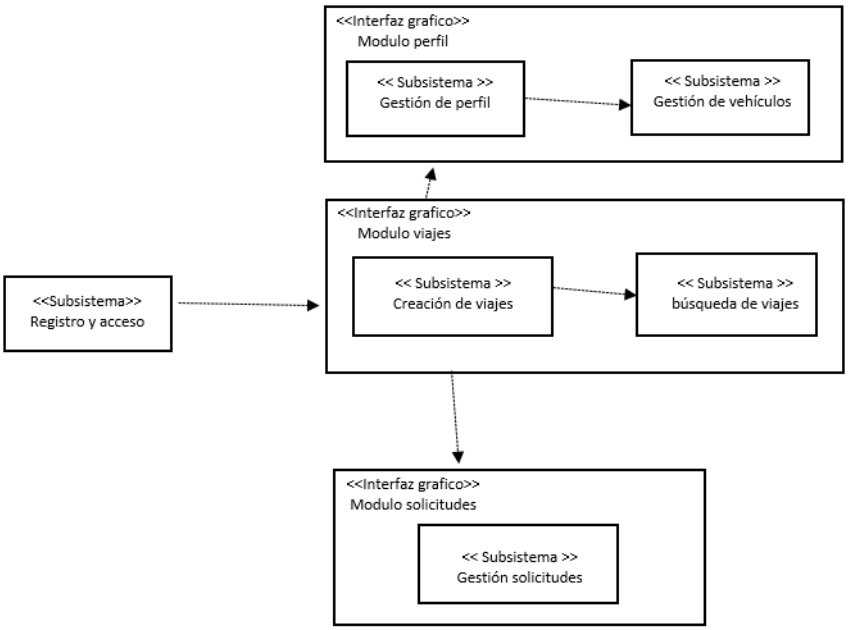


Figura 1 Diagrama de subsistemas

3. Análisis de casos de uso

3.1 Acceso al sistema

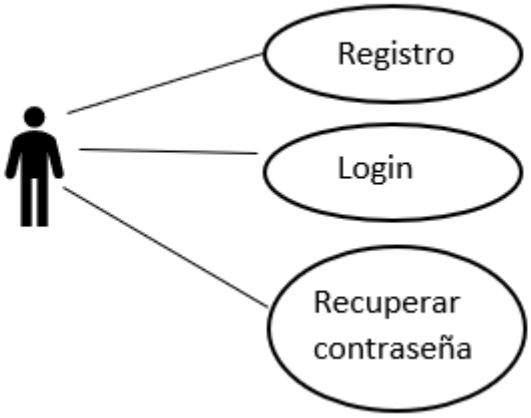


Figura 2 Caso de uso Acceso al sistema

Caso de Uso 1: Registro	
Actor	Usuario.
Precondiciones	El usuario no ha iniciado sesión en el sistema y se encuentra en la página pública del sistema.
Postcondiciones	Se registra al usuario en el sistema
Flujo Principal	1. El usuario completa y envía el formulario de registro con la siguiente información: nombre, apellidos, email electrónico y contraseña. 2. Se registra correctamente en el sistema.
Flujos alternativos	
El email ya está registrado en el sistema.	1. Se realiza la acción 1 del flujo principal. 2. Se muestra un aviso informándole de que ese email ya está registrado en el sistema.
La contraseña no cumple con los requisitos.	1. Se realiza la acción 1 del flujo principal. 2. Se muestra un aviso informándole de que la contraseña no cumple con los requisitos.
El email no tiene un formato válido.	1. Se realiza la acción 1 del flujo principal. 2. Se muestra un aviso informándole de que el email no tiene un formato válido.
Caso de uso 2: Login	
Actor	Usuario.
Precondiciones	El usuario está registrado en el sistema.

Postcondiciones	Se accede a la página de viajes, en la parte privada del sistema.
Flujo principal	1. El usuario introduce sus credenciales, email y contraseña, en el formulario de Login. 2. Se comprueba en la BD que el email esta registrado en el sistema y que la contraseña es la correcta. 3. Accede a la parte privada de la aplicación.
Flujos alternativos	
La contraseña introducida no es correcta.	1. Se realiza la acción 1 y 2 del flujo principal. 2. La contraseña introducida no coincide con la guardada en BD. 3. Se muestra un aviso al usuario informándole de que la contraseña no es correcta.
Caso de uso 3: Recuperar contraseña	
Precondiciones	El usuario está registrado en el sistema.
Postcondiciones	Se envía un correo para que pueda establecer una nueva contraseña.
Flujo principal	1. El usuario pulsa en el botón de recuperar contraseña. 2. Se le muestra un formulario donde deberá escribir el su email. 3. Se envía un email al usuario de recuperación de contraseña.
Flujos alternativos	
El email no tiene un formato válido.	1. Se realiza la acción 1 y 2 del flujo principal 2. Se muestra un aviso informándole de que el email no tiene un formato válido.

3.2 Perfil usuario

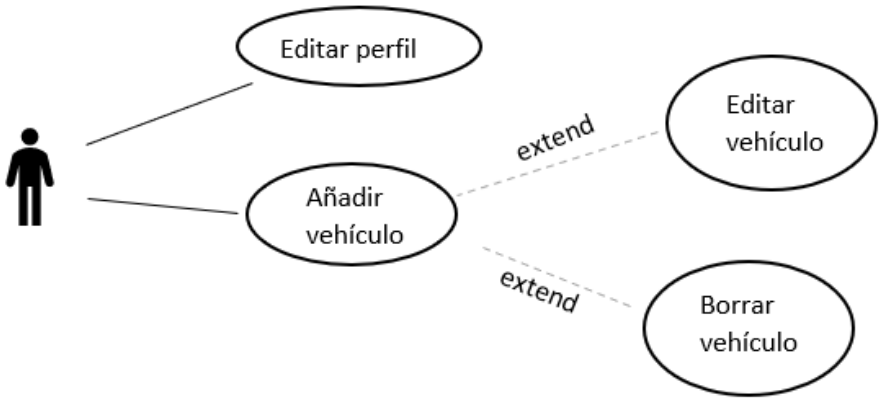


Figura 3 Caso de uso Editar perfil usuario

Caso de Uso 4: Editar perfil	
Actor	Usuario
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de gestión de su perfil.
Postcondiciones	La información del usuario se actualiza en BD.
Flujo Principal	1. El usuario modifica y envía el formulario con la información de su perfil y su imagen. 2. Se actualiza correctamente la nueva información en BD. 3. Se muestra un mensaje al usuario informándole de que se ha actualizado correctamente.
Flujos alternativos	
Alguno de los campos requeridos está vacío.	1. Se realiza la acción 1 del flujo principal. 2. Se avisa al usuario de que debe completar los campos requeridos vacíos.
El archivo adjuntado en el campo imagen no es de un formato válido.	1. Se realiza la acción 1 del flujo principal 2. Se muestra un aviso al usuario de que el tipo del archivo adjuntado no es válido.
Caso de uso 5: Añadir vehículo	
Actor	Usuario
Precondiciones	El usuario ha accedido la parte privada de la aplicación El usuario se encuentra en la página de gestión de vehículos.

Postcondiciones	Se registra el vehículo del usuario en BD. Se le envía a la página de gestión de vehículos donde podrá consulta la ficha de su nuevo vehículo.
Flujo principal	1. Se completa el formulario de registro de vehículo y se envía. 2. La información del vehículo de guarda correctamente en BD. 3. Se redirige al usuario a la página de gestión de vehículos y se muestra un mensaje al usuario informándole de que se ha guardado correctamente.
Flujos alternativos	
Alguno de los campos requeridos está vacío	1. Se realiza la acción 1 del flujo principal. 2. Se avisa al usuario de que debe completar los campos requeridos vacíos.
El archivo adjuntado en el campo imagen no es de un formato válido.	1. Se realiza la acción 1 del flujo principal. 2. Se muestra un aviso al usuario de que el tipo del archivo adjuntado no es válido.
Caso de uso 6: Editar vehículo	
Precondiciones	El usuario ha accedido la parte privada de la aplicación El usuario se encuentra en la página de gestión de vehículos El usuario tiene al menos un vehículo registrado en el sistema.
Postcondiciones	Se edita la información el vehículo en BD Se le envía a la página de gestión de vehículos.
Flujo principal	1. El formulario de contiene la información actual de BD y se muestra la imagen del vehículo. 2. El usuario modifica y envía el formulario con la información de su vehículo. 3. Se actualiza correctamente la nueva información en BD 4. Se muestra un mensaje al usuario informándole de que se ha actualizado correctamente.
Flujos alternativos	
El archivo adjuntado en el campo imagen no es de un formato válido.	1. Se realiza la acción 1 y 2 del flujo principal. 2. Se muestra un aviso informándole de que el email no tiene un formato válido.

Caso de uso 7: Borrar vehículo	
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de gestión de vehículos. El usuario tiene al menos un vehículo registrado en el sistema.
Postcondiciones	Se elimina la información el vehículo en BD.
Flujo principal	1. El usuario pulsa sobre el botón borrar de la ficha del vehículo a eliminar. 2. Se elimina la información del vehículo de BD. 3. Se elimina la ficha del vehículo de la página de gestión de vehículos y se le muestra un mensaje informativo al usuario.
Flujos alternativos	
El archivo adjuntado en el campo imagen no es de un formato válido.	4. Se realiza la acción 1 y 2 del flujo principal. 5. Se muestra un aviso informándole de que el email no tiene un formato válido.

3.3 Gestión Viajes

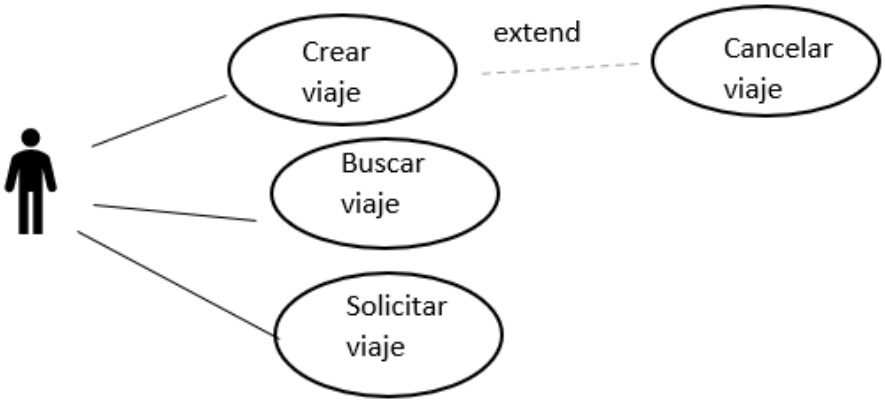


Figura 4 Caso de uso Gestión de viajes

Caso de Uso 8: Crear viaje	
Actor	Usuario.
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de creación de viaje.
Postcondiciones	La información del viaje se guarda en BD y se publica para el resto de los usuarios. Se redirige al usuario a la página de información del viaje.
Flujo Principal	1. El usuario indica en el mapa mediante marcadores el itinerario del viaje a realizar. 2. Se completa el resto de información del formulario de creación y se envía. 4. Se guarda la información del viaje en BD. 5. Se redirige al usuario a la página de información del vehículo y se publica para el resto de los usuarios.
Flujos alternativos	
El usuario no tiene registrado ningún vehículo en el sistema.	1. Al acceder al formulario de creación se le muestra un aviso de que debe añadir al menos un vehículo y se le oculta el botón de envió del formulario.
No se ha indicado alguno de los marcadores requeridos en el mapa.	1. Se realiza la acción 1,2 del flujo principal. 2. Se avisa al usuario del tipo de marcador que es necesario indicar en el mapa.
Alguno de los campos requeridos está vacío.	1. Se realiza la acción 1,2 del flujo principal.

	2. Se muestra un mensaje al usuario con un listado de los campos a completar.
Algún campo no tiene formato válido.	1. Se realiza la acción 1,2 del flujo principal 2. Se muestra un mensaje al usuario con un listado de los campos que no tienen un formato válido.
Caso de uso 9: Buscar viaje	
Actor	Usuario
Precondiciones	El usuario ha accedido la parte privada de la aplicación Se encuentra en una de las secciones del buscador.
Postcondiciones	Se muestran los viajes que cumplen los requisitos.
Flujo principal	1. El usuario accede a uno de los buscadores de viajes e introduce los parámetros de búsqueda si es necesario. 2. Se muestran en el mapa los marcadores con los destinos de cada viaje y los tickets con la información más relevante. 3. El usuario podrá acceder a la ficha de información del viaje o visualizar en el mapa el itinerario de un viaje especifico.
Flujo alternativo	
No hay ningún viaje que cumpla los parámetros de búsqueda.	1. Se realiza la acción 1 del flujo principal 2. Se muestra un aviso al usuario de que no hay ningún viaje con las características que busca.
Caso de uso: 10 Cancelar viaje	
Actor	Usuario organizador.
Precondiciones	El usuario ha accedido la parte privada de la aplicación Se encuentra en la ficha del viaje y es el organizador del viaje.
Postcondiciones	El viaje pasa a estado cancelado y se envía un email a los usuarios solicitantes.
Flujo principal	1. El usuario pulsa sobre el botón cancelar de la ficha del viaje 2. Confirma la acción en la ventana de aviso 3. Se pasa el viaje a estado cancelado y se avisa a los usuarios solicitantes a través de email.
Caso de uso 11: Solicitar viaje	

Actor	Usuario solicitante.
Precondiciones	El usuario ha accedido la parte privada de la aplicación Se encuentra en la ficha del viaje.
Postcondiciones	Se registra la solicitud en BD y se envía un mensaje al usuario organizador.
Flujo principal	1. El usuario accede a la ficha de información del viaje y pulsa el botón solicitar. 2. Se guarda la solicitud en BD. 3. Se muestra un mensaje al solicitante de que se ha creado correctamente y un email al organizador.
Flujo alternativo	
El viaje tiene el número máximo de solicitudes pendientes.	1. Al acceder a la ficha del viaje se informa al usuario de que el viaje no acepta más solicitudes por el momento.
El viaje tiene el número máximo de solicitudes aceptadas.	2. Al acceder a la ficha del viaje se informa al usuario de que el viaje está completo.

3.4 Gestión de solicitudes

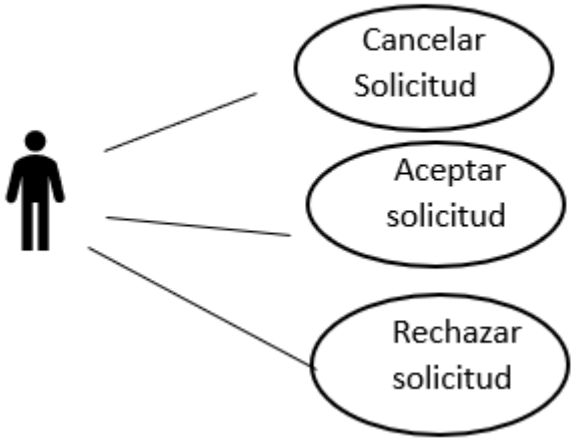


Figura 5 Caso de uso Gestión de solicitudes

Caso de Uso 12: Cancelar solicitud	
Actor	Usuario solicitante.
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de solicitudes enviadas.
Postcondiciones	El estado de la solicitud cambia a CANCELADA.
Flujo Principal	1. La solicitud se encuentra en estado PENDIENTE. 2. El usuario pulsa sobre el botón cancelar. 3. Se confirma la acción pulsando en el botón aceptar el pop up. 4. La solicitud pasa a estado CANCELADA.
Flujos alternativos	
Solicitud en estado ACEPTADA	1. Se realiza la acción 2,3 del flujo principal. 2. Se notifica al usuario organizador y se libera una plaza del viaje.
Solicitud en estado RECHAZADA.	1. No se puede ejecutar ninguna acción sobre la solicitud.
Caso de uso 13: Aceptar solicitud	
Actor	Usuario organizador.
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de solicitudes que ha recibido el viaje

Postcondiciones	El estado de la solicitud cambia a ACEPTADA
Flujo principal	1. La solicitud se encuentra en estado PENDIENTE. 2. El usuario pulsa sobre el botón Aceptar. 3. Se confirma la acción pulsando en el botón aceptar el pop up. 4. La solicitud pasa a estado ACEPTADA, se resta una plaza disponible al viaje y se notifica al solicitante.
Flujos alternativos	
Solicitud en estado CANCELADA	1. No se puede ejecutar ninguna acción sobre la solicitud.
Solicitud en estado RECHAZADA	1. No se puede ejecutar ninguna acción sobre la solicitud.
Caso de uso 14: Rechazar solicitud	
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de gestión de vehículos. El usuario tiene al menos un vehículo registrado en el sistema.
Precondiciones	El usuario ha accedido la parte privada de la aplicación. El usuario se encuentra en la página de solicitudes que ha recibido el viaje.
Postcondiciones	El estado de la solicitud cambia a RECHAZADA.
Flujo principal	1. La solicitud se encuentra en estado PENDIENTE. 2. El usuario pulsa sobre el botón Aceptar. 3. Se confirma la acción pulsando en el botón aceptar el pop up. 4. La solicitud pasa a estado CANCELADA y se notifica al solicitante.
Flujos alternativos	
Solicitud en estado ACEPTADA	No se puede ejecutar ninguna acción sobre la solicitud.
Solicitud en estado CANCELADA	No se puede ejecutar ninguna acción sobre la solicitud.

4. Análisis de Clases

Los diferentes objetos del sistema y sus relaciones se representarán a través de un diagrama de clases utilizando notificación UML.

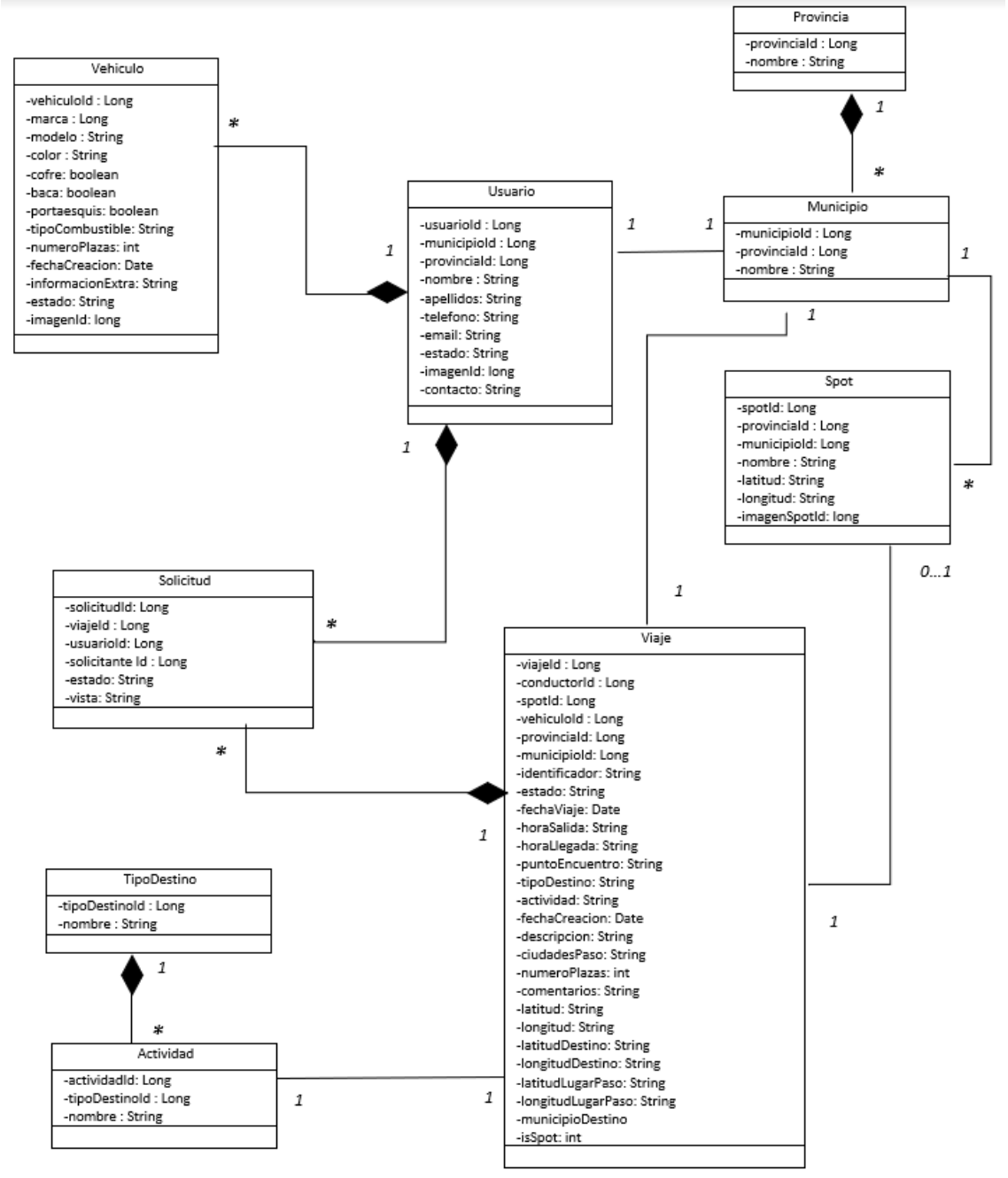


Figura 6 Diagrama Entidad-Relación

4.1 Diagrama de Transición de Estados

A continuación, se representa el comportamiento de los diferentes componentes del sistema en función del evento que se produzca.

DTE Inicio de sesión y perfil usuario

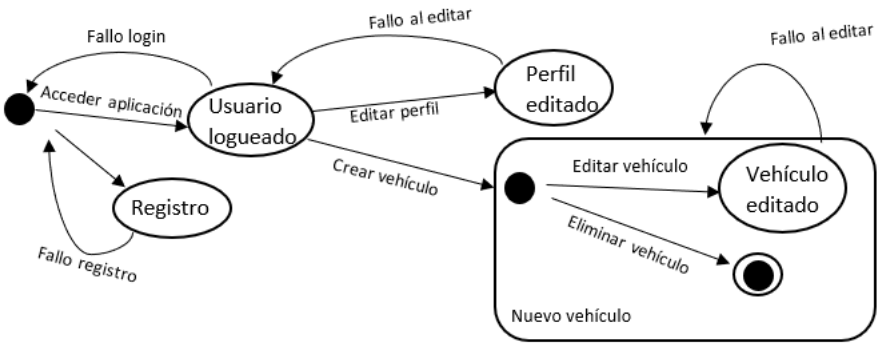


Figura 7 DTE Inicio de sesión y perfil usuario

DTE Gestión de viajes y de solicitudes

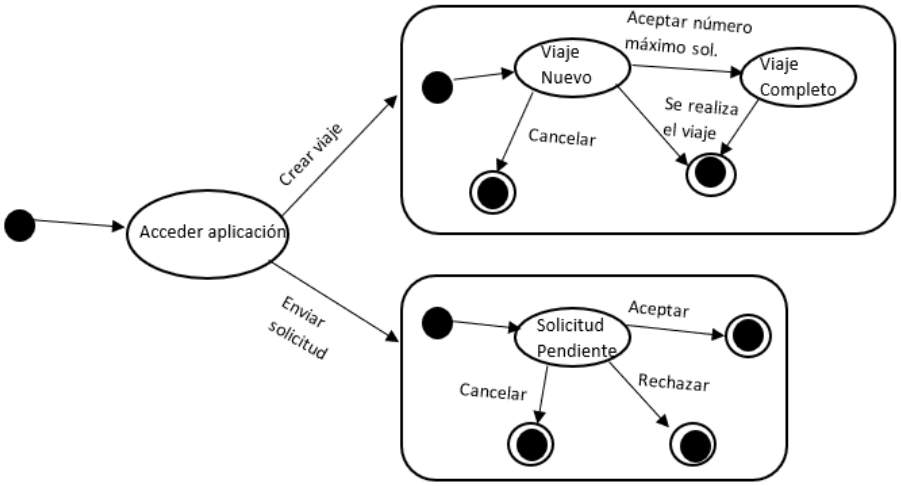


Figura 8 DTE Gestión de viajes y solicitudes

5. Definición de interfaces de usuario

En esta parte del documento se describirán los tipos de usuarios que participan en el sistema y las acciones a realizar por cada uno de ellos.

A continuación, se representará de manera esquemática la navegación entre las diferentes pantallas de la aplicación y se simulará el aspecto visual de las interfaces de los diferentes módulos que componen el sistema. Se describirán las acciones a realizar en cada una de ellas de manera que se cumplan con los requisitos establecidos en el catálogo de requisitos.

5.1 Perfiles de usuario

En primera instancia los usuarios se dividirán en dos tipos: usuarios no registrados y usuarios registrados. Los usuarios no registrados solo tendrán acceso a la parte publica de la aplicación, donde se podrá registrar para ser registrado en el sistema y poder acceder a la parte privada a través de un proceso de login.

Todos los usuarios registrados podrán realizar las mismas acciones en el sistema, editar su perfil, añadir vehículos, crear viajes o enviar solicitudes. En el contexto del viaje se dividirán los usuarios en dos tipos:

- Usuario organizador
- Usuario solicitante

El usuario organizador es el creador del viaje y recibirá las solicitudes de los usuarios solicitantes, podrá aceptarlas o rechazarlas además de cancelar el viaje. Los usuarios solicitantes podrán cancelar las solicitudes enviadas.

5.2 Mapa de prototipos

Antes de definir los prototipos de la interfaz, se establecerá la navegación entre las diferentes interfaces mediante el siguiente mapa de prototipos.

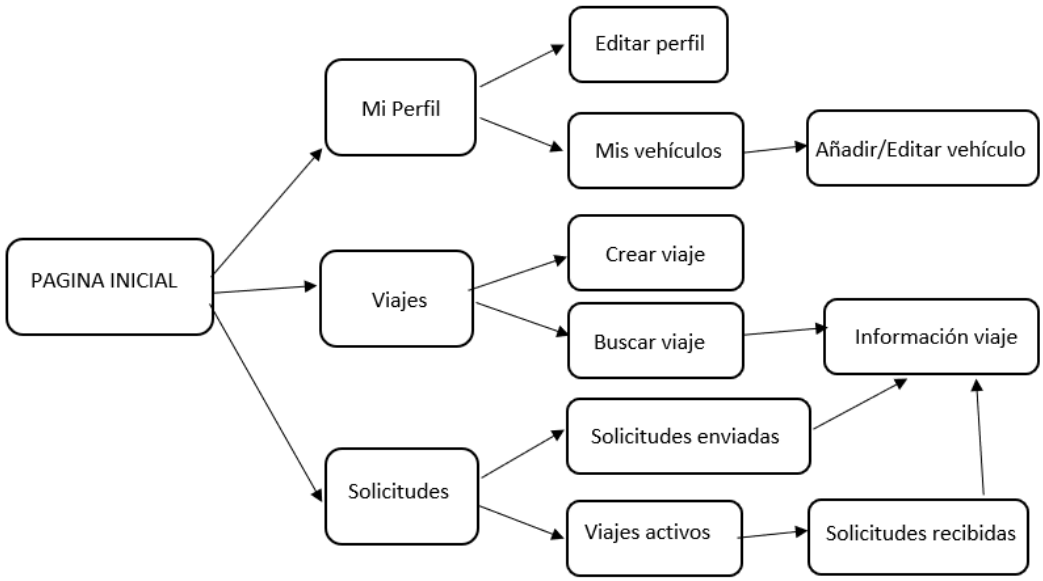
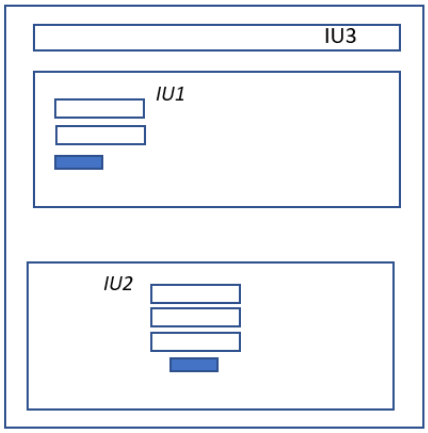


Figura 9 Diagrama navegación entre prototipos de interfaz

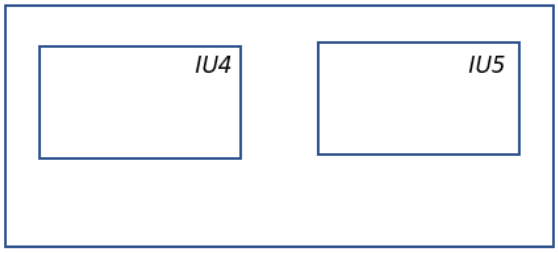
5.3 Formato de interfaz de pantalla

Pantalla inicial

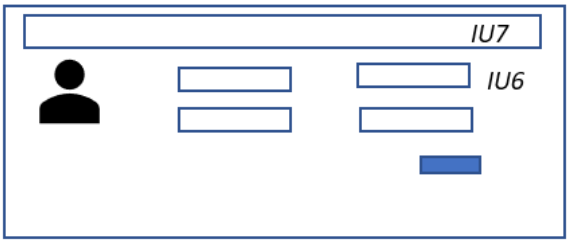


Pagina inicial aplicación		
Ident.	**Descripción**	**Acción**
IU1	Formulario de acceso al sistema, se debe introducir email y contraseña.	Login
IU2	Formulario de registro en el sistema, los campos a completar son nombre, apellido, email y contraseña.	Registro
IU3	Se muestra información sobre los errores.	Mostrar avisos

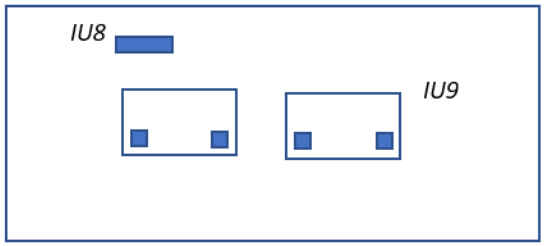
Mi perfil



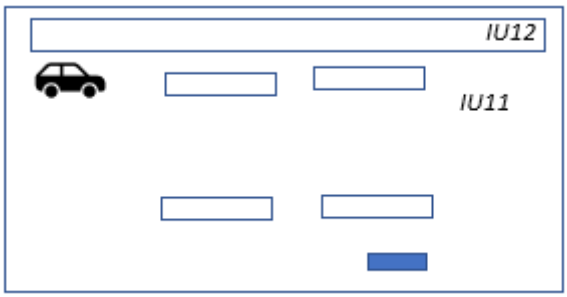
Pagina inicial Mi perfil		
Ident.	**Descripción**	**Acción**
IU4	Redirige al usuario a la página de edición de su perfil.	Acceso editar perfil
IU5	Redirige al usuario a la página Mis Vehículos.	Acceso mis vehículos



Página inicial Mi perfil		
Ident.	**Descripción**	**Acción**
IU6	Formulario con la información del perfil del usuario, podrá añadir o cambiar su foto de perfil y seleccionar la forma de contacto con otros usuarios.	Editar perfil
IU7	Se muestra información sobre el resultado de la acción editar.	Mostrar información

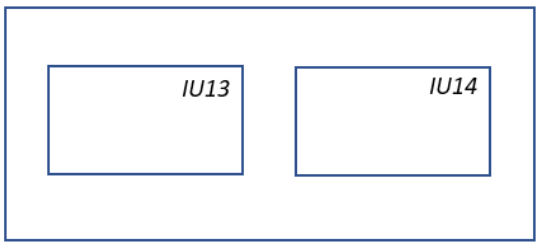


Pagina Mis vehículos		
Ident.	**Descripción**	**Acción**
IU8	Redirige a la página de creación de vehículo.	Acceso a Crear vehículo
IU9	Se muestra la información sobre los vehículos registrados. Permite editar la información o borrarlo.	Editar vehículo

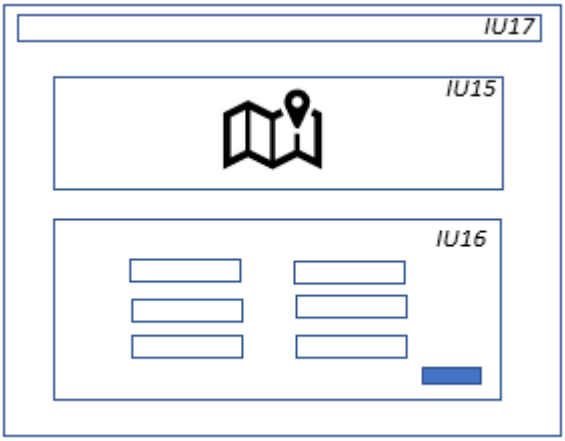


Página crear vehículo		
Ident.	**Descripción**	**Acción**
IU11	Formulario para completar con la información del vehículo, se puede adjuntar una imagen. Al enviarlo se registra el vehículo.	Crear vehículo
UI12	Se muestra información sobre el resultado de la creación del vehículo.	Mostrar información

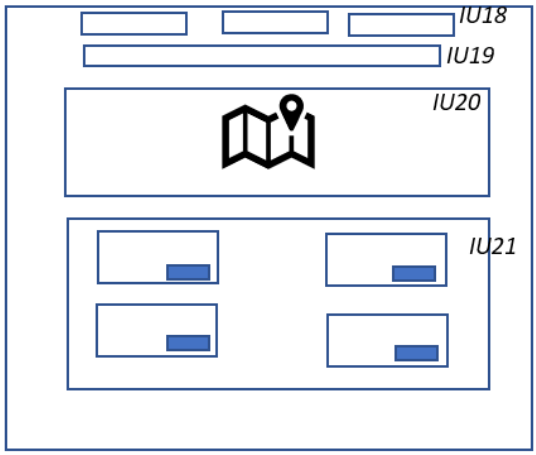
Viajes



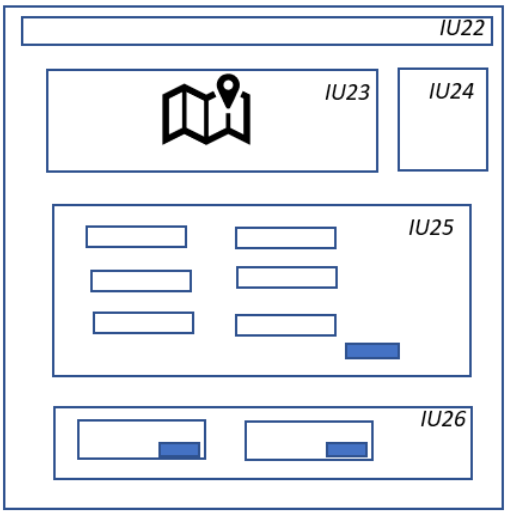
Página inicial Viajes		
Ident.	**Descripción**	**Acción**
IU13	Redirige al usuario a la página Crear Viaje.	Acceso crear viaje
IU14	Redirige al usuario a los Buscador de viajes.	Acceso a buscadores



Página Crear viaje		
Ident.	**Descripción**	**Acción**
IU15	Mapa donde se especifica el itinerario de viaje a través de marcadores.	Definir marcadores viaje
IU16	Formulario para completar con el resto de la información del viaje al enviar se crea el viaje y redirige a la página de información.	Crear viaje
IU17	Se muestra información sobre el resultado de la acción crear viaje.	Mostrar información

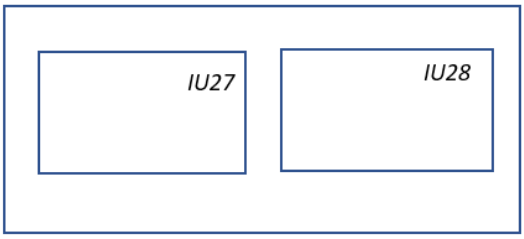


Página Buscador de viajes		
Ident.	**Descripción**	**Acción**
IU18	Redirige al usuario a los diferentes buscadores de viajes.	Acceso a buscadores
IU19	Mapa donde se especifica el itinerario de viaje a través de marcadores. Redirigen a la página información del viaje.	Acceso página creación. Acceso
IU20	Mapa donde se muestran los marcadores de los viajes que cumplan con los requisitos.	Mostrar marcadores
IU21	Se muestra información sobre los viajes que cumplan los requisitos.	Mostrar información viajes

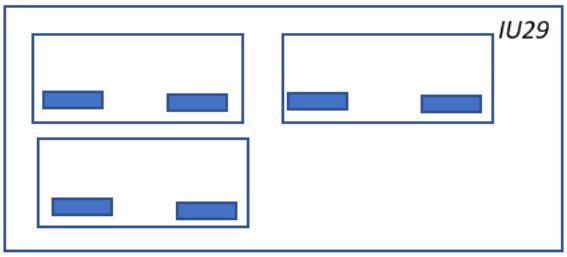


Página información viaje		
Ident.	**Descripción**	**Acción**
IU22	Se muestra información sobre el resultado de la acción Enviar Solicitud.	Mostrar información
IU23	Mapa donde se visualiza el itinerario de viaje a través de marcadores.	Mostrar itinerario
IU24	Muestra al usuario información meteorológica en el lugar destino.	Mostrar información
IU25	Se muestra información sobre el viaje. El usuario solicitante podrá enviar solicitud y el organizador cancelar el viaje.	Enviar solicitud / Cancelar viaje
UI26	Muestra viajes relacionados con el que se está consultando. Redirige a la página de información del viaje.	Acceso a información viaje

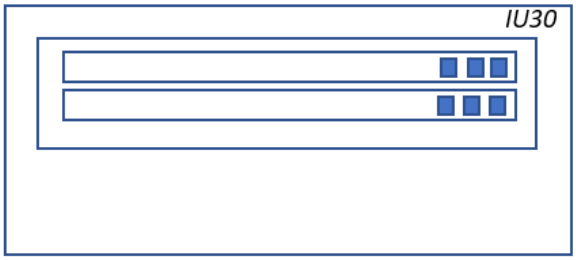
Solicitudes



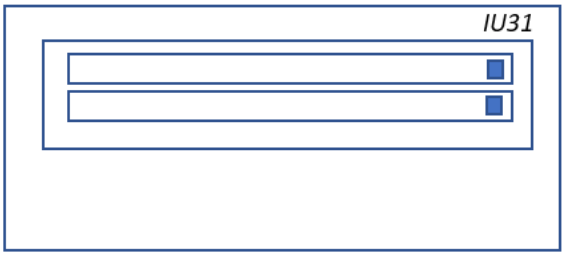
Página inicial Solicitudes		
Ident.	**Descripción**	**Acción**
IU27	Redirige al usuario a la página de Mis Solicitudes Enviadas.	Acceso Mis Solicitudes Enviadas
IU28	Redirige al usuario a la página Mis Viajes Activos.	Acceso Mis viajes activos



Página viajes activos		
Ident.	**Descripción**	**Acción**
IU29	Se muestra a los usuarios los viajes creados que se encuentren activos, podrá ir a la página de solicitud del viaje o acceder a las solicitudes que ha recibido el viaje.	Acceso a información viaje. Acceso a solicitudes recibidas del viaje.



Página solicitudes recibidas		
Ident.	**Descripción**	**Acción**
IU30	Se muestra al usuario la información de las solicitudes recibidas para ese viaje. Sobre una solicitud podrá aceptarla o rechazarla.	Aceptar solicitud. Rechazar solicitud



Página solicitudes enviadas		
Ident.	**Descripción**	**Acción**
IU31	Se muestra al usuario la información de las solicitudes enviadas y el estado de estas. Se podrá cancelar la solicitud si no ha sido resuelta por el organizados	Cancelar solicitud.

6. Análisis de consistencia

El objetivo de esta tarea es garantizar el cumplimiento de los requisitos especificados en el catálogo de requisitos de este documento. Esto se conseguirá realizando la trazabilidad de los requisitos contra diferentes modelos definidos en este documento de manera garantice la calidad de estos.

6.1 Consistencia Requisitos-Objetivos

		Objetivos				
		Registro e inicio de sesión	Gestión de perfil	Gestión de vehículos	Gestión de viajes	Gestión de solicitudes
Requisitos	RF 1.1	X				
	RF 1.2	X				
	RF 1.3	X				
	RF 2.1		X			
	RF 2.2			X		
	RF 2.3		X			
	RF 2.4			X		
	RF 2.5			X		
	RF 3.1				X	
	RF 3.2				X	
	RF 3.3				X	
	RF 3.4				X	
	RF 3.5				X	
	RF 3.6				X	
	RF 3.7					X
	RF 3.8				X	
	RF 3.9				X	
	RF 3.10				X	
	RF 4.1					X
	RF 4.2					X
	RF 4.3					X

	RF 4.4					X
	RF 4.5					X
	RF 4.6					X
	RF 4.7					X
	RF 4.8					X

6.2 Consistencia Requisitos-Casos de uso

Requisitos	Objetivos													
	CU Registro	CU Login	CU Recuperar contraseña	CU Editar perfil	CU Añadir vehículo	CU Editar vehículo	CU Borrar vehículo	CU Crear viaje	CU Buscar viaje	CU Cancelar viaje	CU Solicitar viaje	CU Aceptar solicitud	CU Rechazar Solicitud	CU Cancelar solicitud
RF 1.1	X													
RF 1.2		X												
RF 1.3			X											
RF 2.1				X										
RF 2.2					X									
RF 2.3				X										
RF 2.4					X									
RF 2.5						X								
RF 2.6							X							
RF 3.1								X						
RF 3.2								X						
RF 3.3								X						
RF 3.4								X						
RF 3.5								X						
RF 3.6								X						
RF 3.7										X				
RF 3.8									X					
RF 3.9									X					
RF 3.10											X			
RF 4.1												X		

	RF 4.2													X
	RF 4.3											X		
	RF 4.4											X		
	RF 4.5											X		
	RF 4.6												X	
	RF 4.7											X		
	RF 4.8										X			

7. Especificación del Plan de Pruebas

A continuación, se describirán los diferentes tipos de pruebas que se ejecutarán sobre el sistema y que servirá como guía para la ejecución las mismas. El objetivo de la ejecución de diferentes tipos de pruebas es detectar posibles errores para que sean corregidos y garantizar que se cumplen todas las funcionalidades definidas en el catálogo de requisitos

Los resultados de su ejecución se presentarán en el apartado de *Desarrollo del Plan de Pruebas.* Las pruebas que se ejecutarán son las siguientes. Los diferentes tipos de pruebas que se van a realizar son los siguientes:

- **Pruebas Unitarias:** en este tipo de pruebas se verificará la lógica y correcto funcionamiento de los métodos desarrollados de manera independiente. Se ejecutarán estas pruebas sobre las clases Controlador, Manager y Útil.

  Maven es la herramienta software que se utilizará para empaquetar y desplegar el código desarrollado. También nos facilita la ejecución de pruebas unitarias sobre el código que se ha desarrollado a través del comando *mvn test.* Cada vez que se cree un nuevo método en un portlet se ejecutará este comando para realizar test automáticos JUnit.

- **Pruebas de Integración:** el objetivo de este tipo de pruebas será comprobar la correcto funcionamiento e interacción de los módulos que forman el sistema.

  Estas pruebas se realizarán de manera manual, una vez se obtenga una versión funcional de todos los componentes integrados en *Liferay Portal*. Para realizar estas pruebas se registrarán tres usuarios en la aplicación y se realizarán las acciones como edición de perfil, gestión de vehículos, creación de viaje, búsqueda de viajes y gestión solicitudes y creación de vehículo. De manera que se ejecuten los distintos casos de prueba definiendo para cada una de las pruebas las entradas y las salidas esperadas.

- **Pruebas de Usabilidad:** tienen como objetivo obtener la aceptación de los usuarios de la aplicación, los cuales podrán validar las diferentes funcionalidades del sistema.

  Se llevarán a cabo a través de un cuestionario que tendrá que ser completado por los usuarios, estará orientado a valorar de cada una de las funcionalidades principales de la aplicación.

- **Pruebas de carga**: su objetivo es valorar el comportamiento de la aplicación bajo distintos volúmenes de usuarios concurrentes para determinar a partir de que cantidad de peticiones la aplicación empeora su rendimiento.

Estas pruebas se realizarán cuando se haya implementado la aplicación en un entorno productivo por lo que sus resultados no se encuentran en el documento de Desarrollo del plan de pruebas. Su ejecución se llevará a cabo midiendo los tiempos de respuesta, a través de un cliente web, de las transacciones importantes de la aplicación. De esta manera se determinará a partir de que cantidad de usuarios simultáneos los tiempos de respuesta no cumplen con los definidos en los requisitos no funcionales.

5. Diseño del Sistema de Información

1. Diseño de la Arquitectura del Sistema

A continuación, se describirá la infraestructura tecnológica que dará soporte al sistema de información. A pesar de utilizar el software Liferay Portal CE, la aplicación a desarrollar se asemeja a una aplicación web más que a un portal web. La arquitectura general de la aplicación sigue el patrón Cliente-Servidor, está compuesta por un cliente web que realiza peticiones al sistema, un servidor web que trata las peticiones y envía la respuesta al cliente y por último una BD donde se almacenan y actualiza la información que maneja el sistema.

El patrón de arquitectura software elegido para los diferentes portlets que se desplegarán en el servidor web y se desarrollarán las interfaces de usuario y las funcionalidades, será MVC (Modelo-Vista-Controlador). Este tipo de arquitectura proporciona mecanismos para facilitar el desarrollo y mantenimiento de los componentes software a desarrollar. Los portlets se integran en el portal web ejecutándose como aplicaciones web independientes y a su vez el portal se encarga de proporcionarles un entorno de ejecución, recursos necesarios para su ejecución, controla su ciclo de vida y redirige las peticiones recibidas al portlet correspondiente

1.1 Entorno de trabajo

En esta parte se describen las características del equipo donde se realizará el desarrollo e implementación del sistema de información y que da soporte a la arquitectura previamente definida.

- Sistema Operativo: Windows 10 Pro
- Servidor Web: Apache Tomcat / 7.0.62
- Portal Web : Liferay Portal 6.2 CE ga6
- Versión Java: 1.8.0_171
- Versión Apache Maven: 3.6.1. Herramienta software que se utilizará para el empaquetado y despliegue de portlets en el servidor.
- Entorno de desarrollo integrado: Eclipse IDE 4.11.0
- Base de datos: MySQL 5.7

1.2 Diagrama de la arquitectura general

A continuación, se describirá brevemente la funcionalidad de los diferentes elementos de la arquitectura:

- Cliente web: a través de los diferentes navegadores web los clientes realizarán peticiones HTTP de tipo GET para obtener y visualizar la información y tipo POST para realizar envíos de datos. Los navegadores web recibirán la respuesta del servidor y le presentarán

- Servidor web: recibirá las peticiones de los clientes web y las redirigirá al módulo correspondiente para realizar el tratamiento de la petición y construir una respuesta que mostrará al usuario la interfaz

- Base de datos: sistema donde se guarda la información y se permite el acceso a ella de forma segura. Interactúa con el servidor de aplicaciones a través de la capa de persistencia de los diferentes módulos del servidor de aplicaciones.

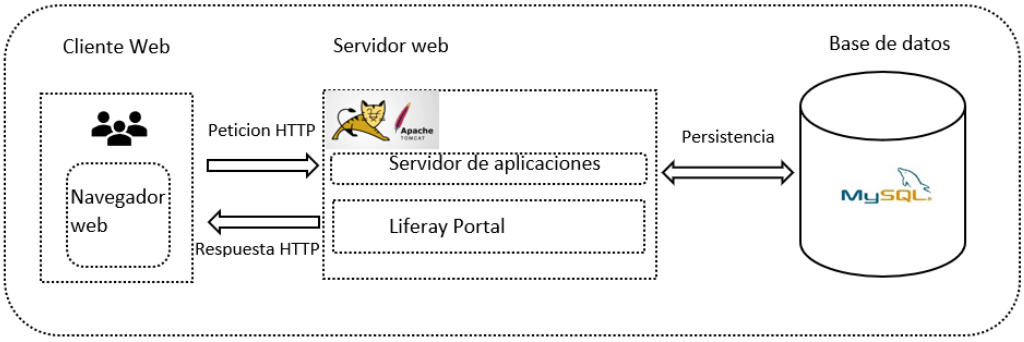


Figura 10 Diagrama de la arquitectura de la aplicación

1.3 Estructura de las clases autogeneradas

Estas clases son generadas en todos los portlet que requieran de acceso a la BD y se generarán a través de Maven que con un comando (*mvn liferay:build-service*) se generarán las clases e interfaces necesarias para realizar operaciones básicas sobre las entidades definidas. Se representarán las clases autogeneradas en base al fichero service.xml a través de diagramas de clases. Los métodos definidos en estas clases serán accesibles por cualquier portlet que las incluya como dependencia y que proporcionan acceso funciones búsqueda, creación, actualización y borrado sobre el modelo de datos. Además, se generan las clases necesarias para permitir acceso a servicios remotos SOAP o JSON. Tendremos todo lo anterior en cuenta cuando se representen los diagramas de secuencia de los diferentes portlets.

Tomaremos como ejemplo las clases generadas a partir de la entidad Viaje, aunque las clases generadas son iguales para todas. Las clases e interfaces generadas se dividen en tres capas:

modelo, persistencia y servicio. Hay que destacar que se pueden realizar ampliaciones sobre el código generado para personalizar la capa de persistencia sin ocasionar problemas en el resto de las capas.

- Modelo

En primer lugar, se realizará una pequeña descripción de las funciones de cada una de las clases:

  - ViajeModel: se definen los métodos get y set de los atributos de la clase.
  - ViajeModelImpl: implementación de los métodos de ViajeModel.
  - ViajeImpl: en esta clase se puede extender el comportamiento de la clase ViajeModel
  - Viaje: a la que se hace referencia en el código que desarrollaremos.

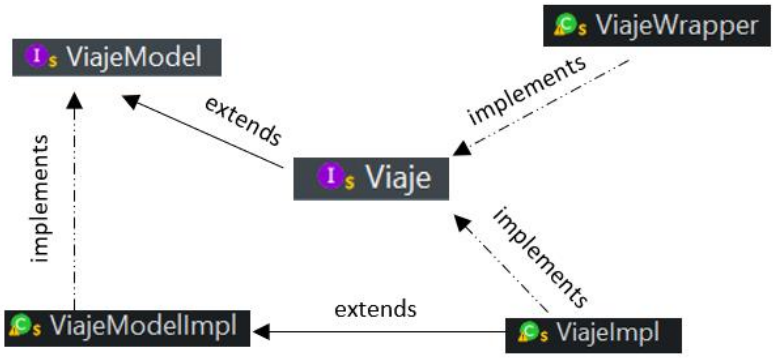


Figura 11 Diagrama de clases del modelo autogeneradas

- Persistencia

En primer lugar, se realizará una pequeña descripción de las funciones de cada una de las clases:

  - ViajePersistence: definición de los métodos de búsqueda por atributos del modelo
  - ViajePersistenceImpl: implementación de los métodos de la interfaz ViajePersistence.
  - ViajeUtil: implementación de los métodos que realizan operaciones CRUD sobre el modelo

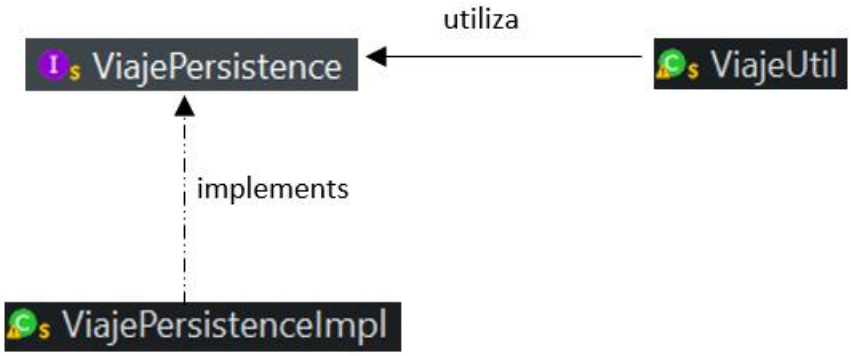


Figura 12 Diagrama de clases de persistencia autogeneradas

- Servicio

En primer lugar, se realizará una pequeña descripción de las funciones de cada una de las clases:

  - ViajeLocalServiceImpl: en esta clase se puede añadir código para añadir nuevos métodos.
  - ViajeLocalServiceBaseImpl: implementación de los métodos de Servicio.
  - ViajeLocalServiceUtil: a partir de esta clase se accede a los servicios definidos en esta capa.

Figura 13 Diagrama de clases de servicio autogeneradas

2. Especificación de Subsistemas

A continuación, se abordará en detalle la implementación de los diferentes portlets desarrollados que componen el sistema. Los portlets son componentes modulares gestionados y visualizados por nuestra aplicación web, para cada uno de ellos se detallarán los métodos de los que dispone y de las clases implicadas. Los portlets que se han desarrollado son los siguientes:

- Administración-Provincias-Municipio
- Administración-Solicitudes
- Administración-Spots
- Administración-Usuarios
- Administración-vehículos
- Administración-Viajes
- Registro-Usuarios

Cada uno de estos portlets se compone de dos submódulos con el siguiente nombrado: nombrePortlet-portlet y nombrePortlet-service. Tomemos como ejemplo el módulo Administración-Usuarios para explicar su estructura y comportamiento, que se repite para el resto de portlets desarrollados.

- Administración-Usuarios-portlet: en este módulo se encuentran las implementaciones de los servicios que gestionan el modelo de datos Usuario, la construcción de estas implementaciones se lleva a cabo a través del fichero service.xml donde se define la entidad Usuario utilizando el comando *mvn liferay build-service* que nos proporciona Maven. En este módulo también se encuentran las implementaciones propias para el manejo de la interfaz de usuario y la gestión de las peticiones recibidas.

- Administración-Usuarios-portlet-service: este módulo contiene las clases model, wrapper, soap entre otras, construidas a partir de los modelos definidos en service.xml que serán consumidas por el submódulo *Administración-Usuarios-portlet* y se detallarán a continuación.

2.1 Portlet Registro usuarios

En este portlet se implementan los métodos necesarios para el registro de un nuevo usuario y se visualiza a través de la página inicial de la aplicación. Dispone de una única interfaz de usuario que es un formulario donde el usuario lo completa con su información personal y se encarga de gestionar todas las operaciones realizadas sobre la tabla *pk_au_usuarios,* la tabla donde se guardará la información del usuario pero al registrar un usuario también se realizará una inserción en la tabla *user_* propia de Liferay ya que el acceso al sistema realizará a través del login propio de Liferay para beneficiarnos de los mecanismos de seguridad para esta funcionalidad.

Clases portlet Registro-Usuarios

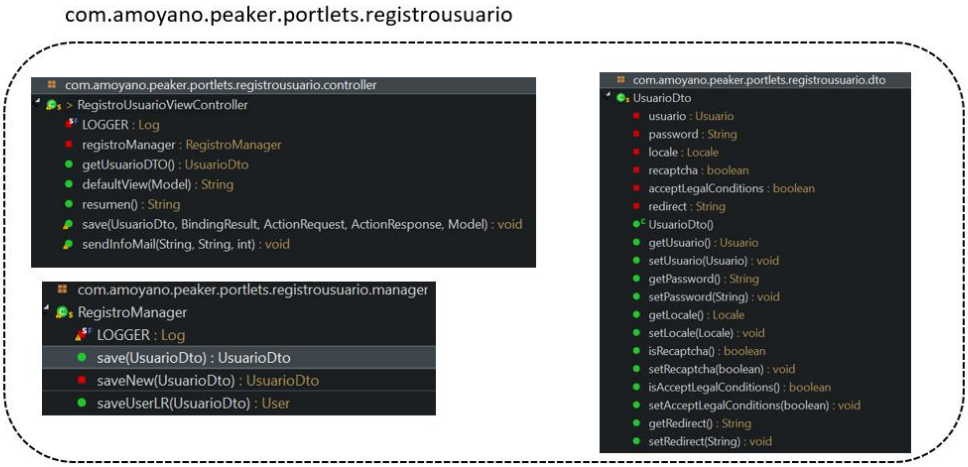


Figura 14 Diagrama de clases paquete registrousuario

Diagrama de secuencia

A continuación, se ilustra la interacción de las diferentes clases desarrolladas en el portlet Registro-Usuarios

cuando un usuario realiza una interacción con la interfaz de usuario

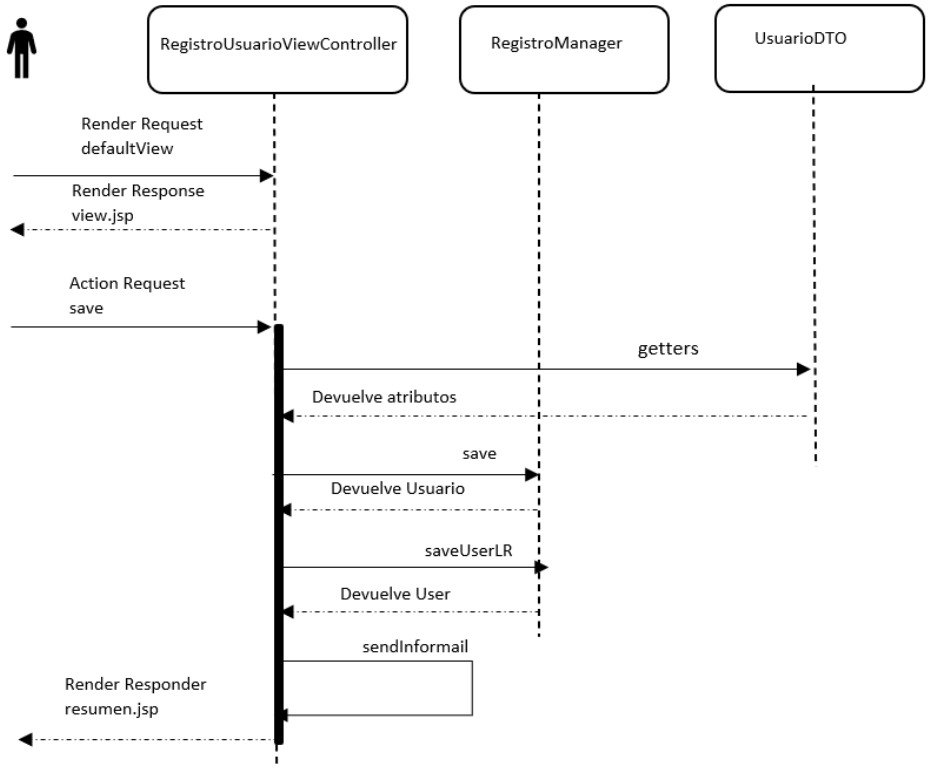


Figura 15 Diagrama de secuencia portlet Registro-Usuario

2.2 Portlet Administración-Provincias-Municipios

En este portlet se implementan los métodos para la gestión de los modelos Provincias y Municipios, no dispone de interfaz de usuario por lo que su función será proveer al resto de portlets que lo tengan como dependencia, los servicios necesarios para la consulta de provincias y municipios. Es un portlet muy básico que dispone de una única interfaz en la parte de administración del portal donde se puede importar un fichero JSON con la información de provincias y municipios, la carga de información en estas tablas se ha llevado a cabo de esta manera.

Para este portlet no se realizará un *diagrama de secuencia* ya que el usuario no tiene interacción con él, es ejecutado por el administrador cuando se realice la carga de datos de provincias y municipios.

Clases portlet Administracion-Provincias-Municipios

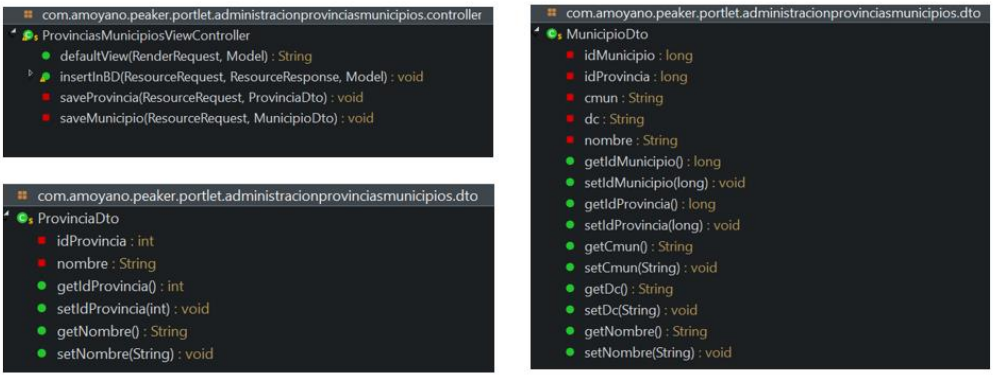


Figura 16 Diagrama de clases portlet Administración-Provincias-Municipios

2.3 Portlet Administración-Spots

En este portlet, al igual que el anterior no dispone de interfaz gráfica y únicamente se implementan los métodos para la gestión en este caso de los objetos Spot , su función será proveer al resto de portlets que lo tengan como dependencia, los servicios necesarios para la

Las clases que componen este portlet son las autogeneradas, que son básicamente la consulta y obtención de información por cada uno de los campos del modelo por lo que no se definirá *diagrama de clases* ni *diagrama de secuencia* en esta ocasión.

2.4 Portlet Administración-Usuarios

Uno de los portlets principales del proyecto, se encuentra desplegado en la página Mi perfil de la aplicación web. Proporciona los métodos necesarios para la gestión del perfil del usuario y la gestión de los vehículos, consume los servicios de los portlets Administración-Provincias-Municipios, Administración-Vehículos y Administración-Viajes. A continuación, se describen las funcionalidades:

- Editar la información del usuario a través de un formulario cuyos campos son los siguientes ; nombre, apellidos, provincia, municipio, teléfono y contraseña. Permite añadir una imagen de perfil del usuario y establecer el método de contacto con el resto de los usuarios. Muestra avisos con el resultado de la acción realizada.

- Añadir un vehículo definiendo los siguientes campos: marca, modelo, tipo de combustible, color, plazas disponibles ,información extra e imagen del vehículo.

- Permite visualizar la información de los vehículos registrados y realizar acciones sobre ellos como editar la información o borrarlos.

- Gestionar la información de los viajes relacionados en caso de que un vehículo sea borrado

La navegación de las diferentes vistas que genera este portlet se representará a través de un diagrama de páginas jsp.

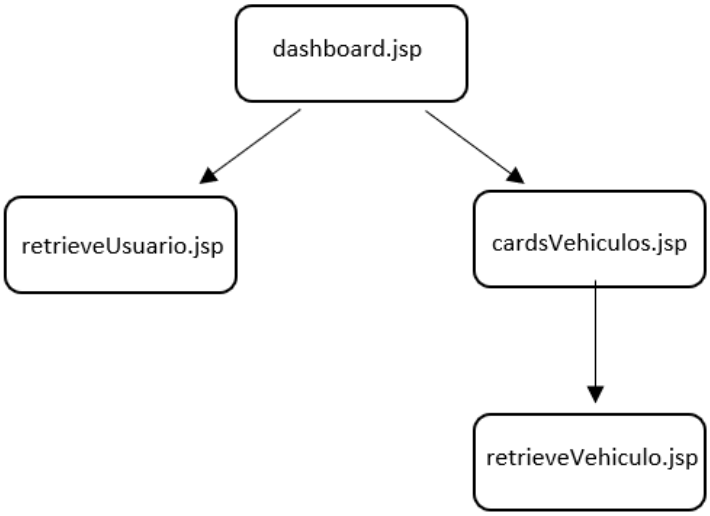
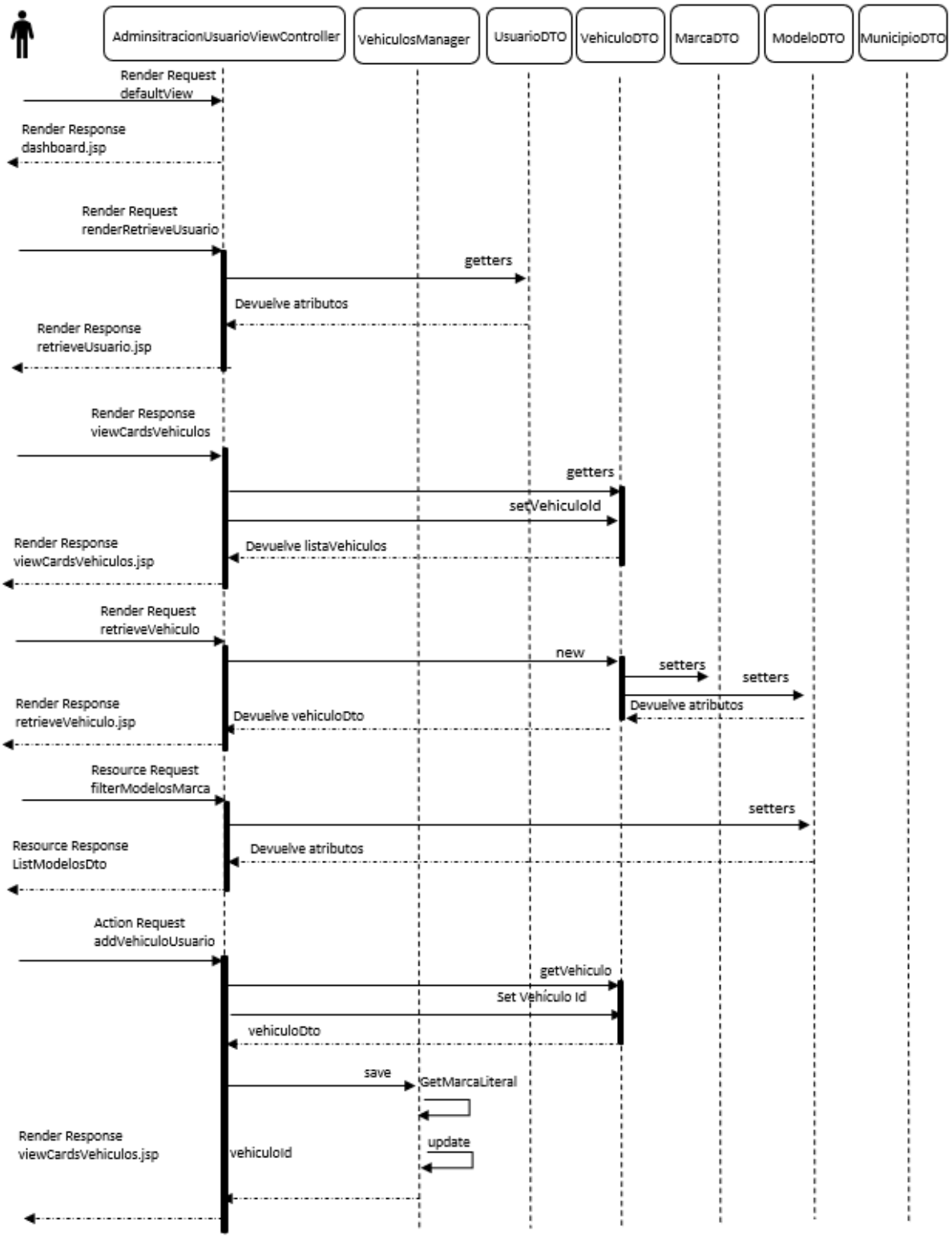


Figura 17 Diagrama de .jsp del portlet Administración-Usuarios

Clases portlet Administracion-Usuarios

Figura 18 Diagrama de clases del portlet administración-usuarios

Diagrama de secuencia



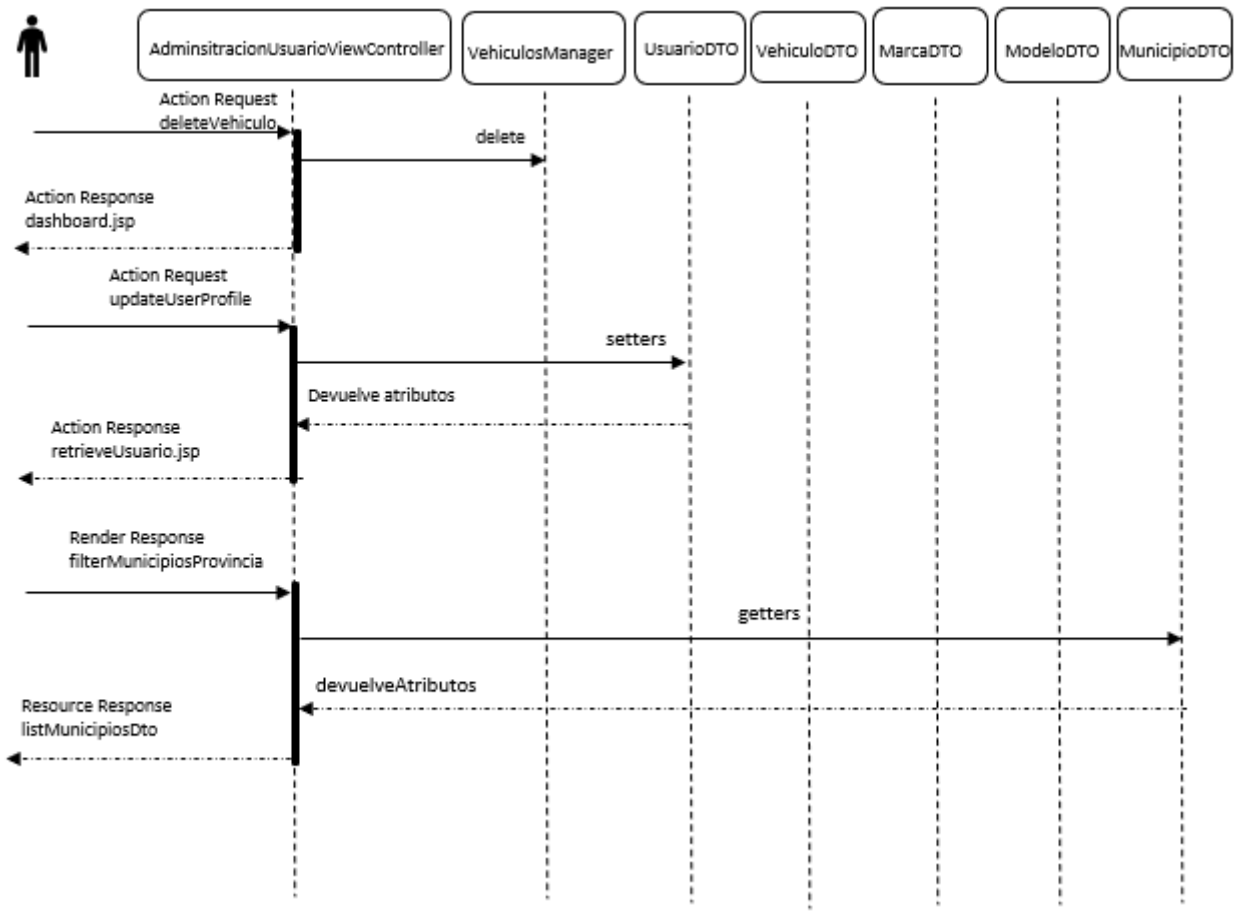


Figura 19 Diagrama de secuencia del portlet Administración-Usuarios

2.5 Portlet Administración-Vehiculos

En este portlet, como en otros casos no dispone de interfaz gráfica y únicamente se implementan los métodos para la gestión en este caso de los objetos Vehículo, su función será proveer al resto de portlets que lo tengan como dependencia, los servicios necesarios para la consulta y manipulación de objetos Vehículo. Las clases que componen este portlet son las autogeneradas, que son básicamente la consulta y obtención de información por cada uno de los campos del modelo por lo que no se definirá *diagrama de clases* ni *diagrama de secuencia* en esta ocasión.

2.6 Portlet Administración-Viajes

Otro de los portlets principales del proyecto, se encuentra desplegado en la página *Viaje* de la aplicación web. Proporciona los métodos necesarios para la creación y eliminación de viajes, además de proporcionar las funcionalidades necesarias a la interfaz de búsqueda de viajes. Las funcionalidades que realiza este portlet son las siguientes:

- Creación de viajes donde el itinerario se define a través de diferentes marcadores en un mapa y la información del viaje a través de un formulario cuyos campos son: tipo de destino, actividad, descripción de la actividad, municipio, provincia, punto de encuentro, fecha del viaje, hora salida, hora llegada, vehículo e información extra.

- Búsqueda de viajes a través de tres tipos de buscadores: provincia origen, actividad y rango de fechas. Los tres buscadores mostrarán la información de los viajes que cumplan los requisitos de búsqueda de dos maneras, a través de un mapa donde se mostrarán los marcadores destino de cada uno de los viajes y listado de los viajes mostrando la información más relevante de cada uno.

- Mostrar la información completa del viaje y de su itinerario además de permitir solicitar o cancelar el viaje dependiendo del tipo de usuario que acceda a la vista. También muestra viajes relacionados con el que se está consultando

- Realizar peticiones a la API OpenWeatherMaps para recibir información meteorológica en diferentes lugares para mostrar la información en la fiche del viaje.

- Permitir al usuario cancelar un viaje y gestionar el aviso de la acción a los solicitantes del viaje a través de un email.

- Crear una solicitud para un viaje a través de la página de información de este y notificarlo a los usuarios implicados a través de un email.

Teniendo en cuenta que este será es portlet con mayor número de clases y métodos simplificaremos el diagrama de paquetes incluyendo los flujos de los métodos más representativos de cada tipo.

A continuación, se adjunta la parte del *pom* donde se especifican las dependencias del resto de portlets que consume.

```xml
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionviajes</groupId>
    <artifactId>Administracion-Viajes-portlet-service</artifactId>
    <version>1.0.2</version>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionvehiculos</groupId>
    <artifactId>Administracion-Vehiculos-portlet-service</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionusuarios</groupId>
    <artifactId>Administracion-Usuarios-portlet-service</artifactId>
    <version>1.0.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionprovinciasmunicipios</groupId>
    <artifactId>Administracion-Provincias-Municipios-portlet-service</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionsolicitudes</groupId>
    <artifactId>Administracion-Solicitudes-portlet-service</artifactId>
    <version>1.0.1</version>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionspots</groupId>
    <artifactId>Administracion-Spots-portlet-service</artifactId>
    <version>1.0.0-SNAPSHOT</version>
</dependency>
```

Figura 20 Fichero pom.xml del portlet Administración-Viajes

La navegación de las diferentes vistas que genera este portlet se representará a través de un diagrama de páginas jsp.

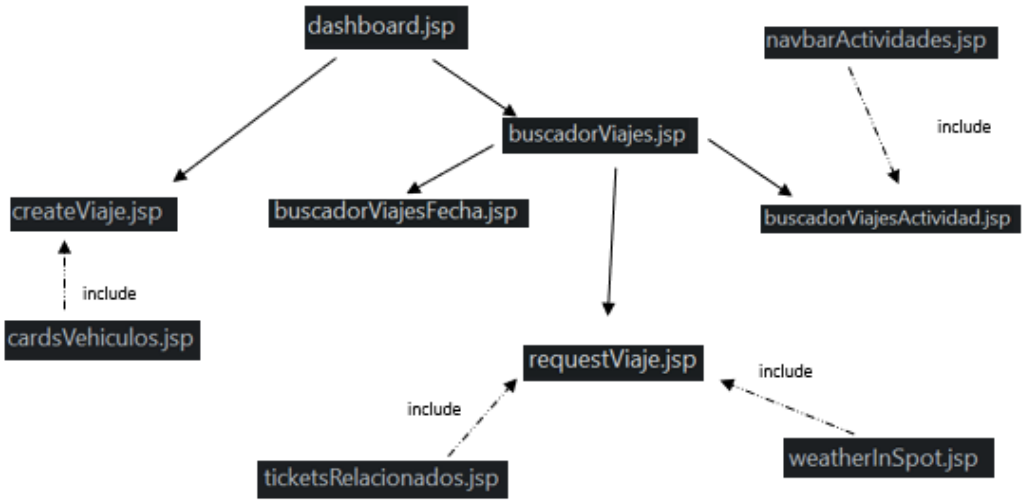


Figura 21 Diagrama de .jsp del portlet Administración-Viajes

Figura 22 Diagrama de clases del portlet Administración-Viajes

Diagrama de secuencia

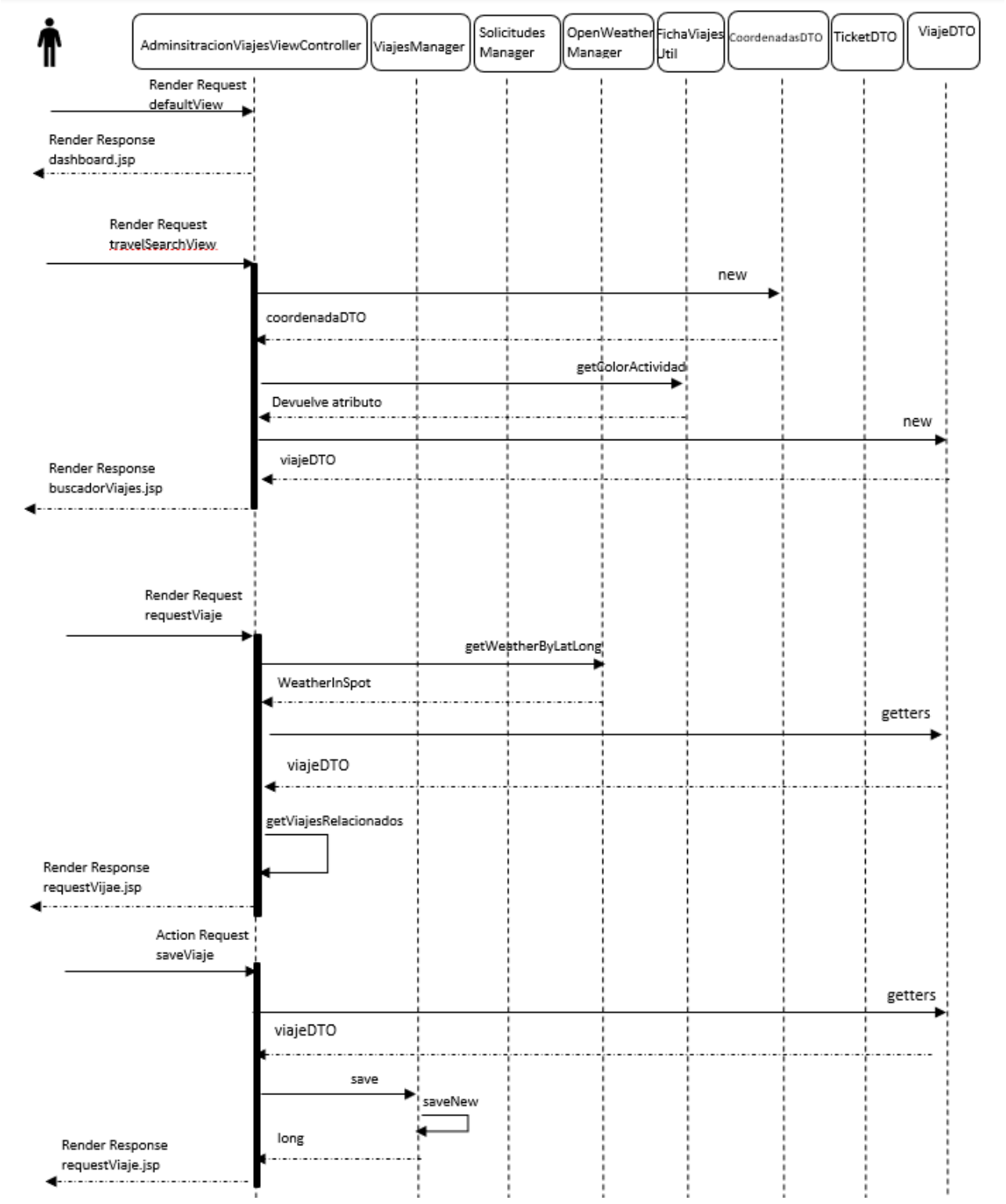


Figura 23 Diagrama de secuencia del portlet Administración-Viajes

2.7 Portlet Administración-Solicitudes

Este último portlet se encuentra desplegado en la página *Solicitudes* de la aplicación web. Proporciona los métodos necesarios para la gestión de solicitudes por parte de los dos tipos de usuarios, organizadores y solicitantes. Las funcionalidades que realiza este portlet son las siguientes:

- Visualización a través de interfaces de usuario de solicitudes recibidas y enviadas.
- Métodos necesarios para la gestión y control de solicitudes, esto se refiere a aceptarlas rechazarlas o cancelarlas además de ejecutar los métodos necesarios sobre los viajes.
- Envío de correos electrónicos a los usuarios pertinentes informándoles de la resolución de las solicitudes

En el diagrama de paquetes se representará únicamente la llamada la función aceptarSolicitud pues las llamadas para las funciones aceptarSolicitud, rechazarSolicitud y cancelarSolicitud son iguales.

Clases portlet Administracion-Solicitudes

Figura 24 Diagrama de clases del portlet Administración-Solicitudes

Diagrama de paquetes

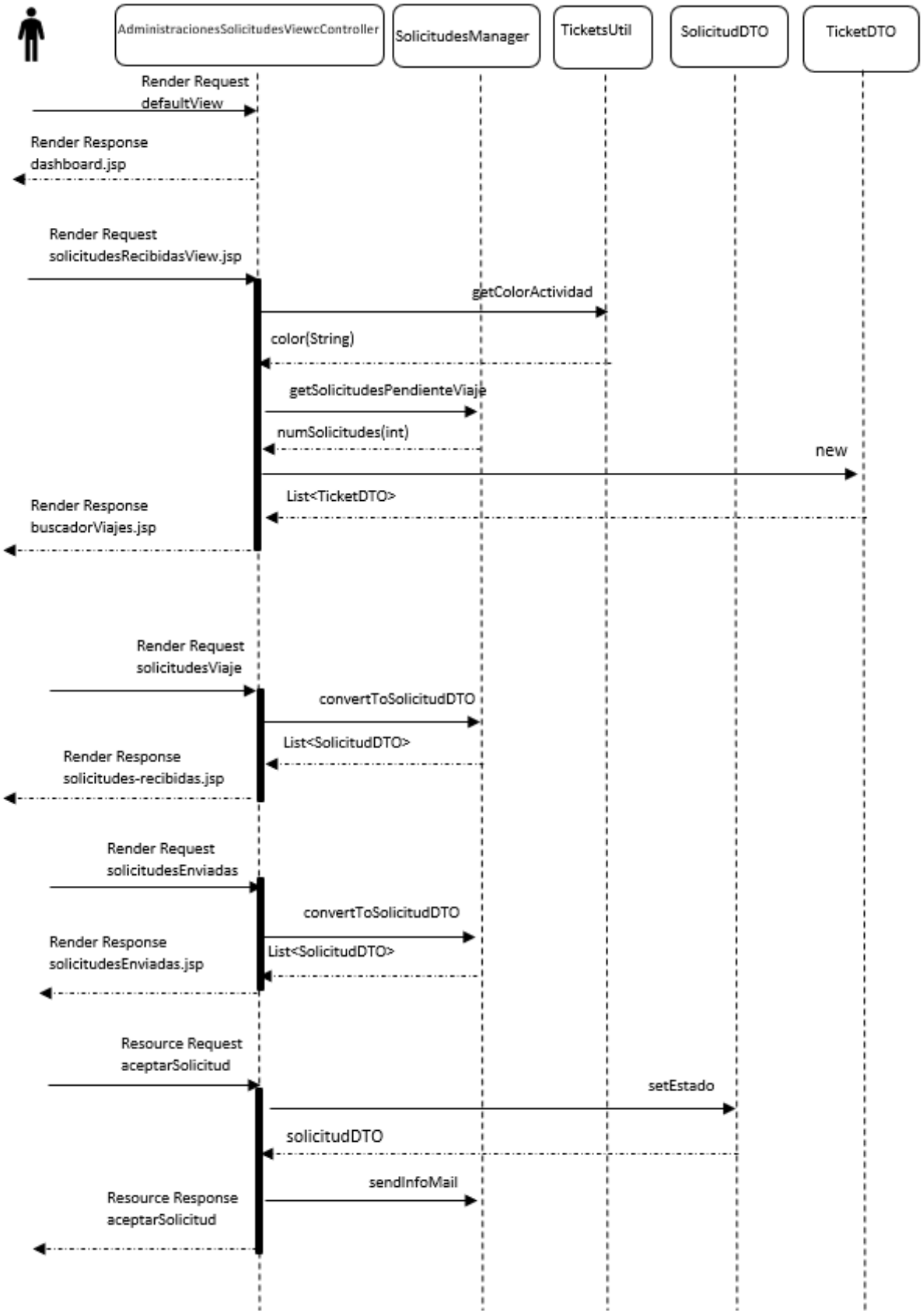


Figura 25 Diagrama de secuencia del portlet Administración-Solicitudes

3. Modelo de datos

La elaboración del modelo de datos tiene como objetivo representar y definir todos los datos que se almacenan, crean y gestionan en el sistema. El modelo de datos se proporciona una visión general que facilita la comprensión de los datos y funcionamiento del sistema. Lo que se conseguirá son las estructuras de datos del sistema, independientemente de las restricciones tecnológicas y físicas del entorno. Se utilizará un modelo entidad-relación:

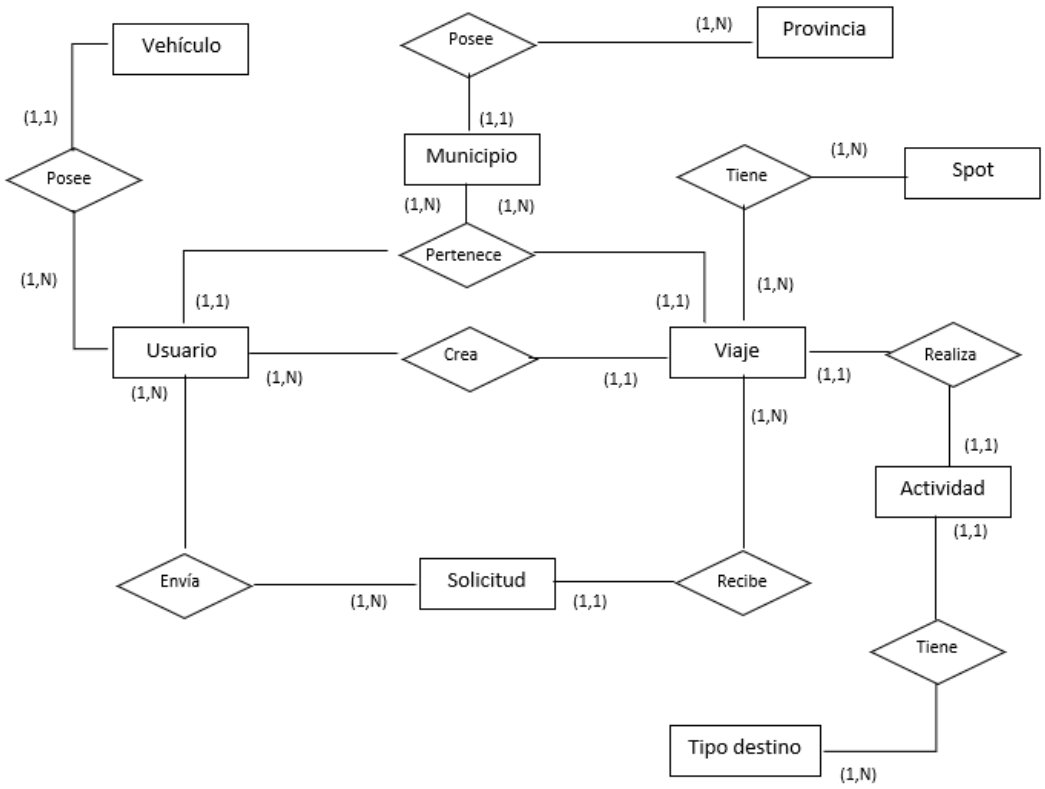


Figura 26 Diagrama entidad-relación

4. Diseño de Interfaz de usuario

A continuación, se presentarán las interfaces de usuario que conforman la aplicación acompañadas de una descripción de los elementos de la interfaz y los *Casos de prueba* cuyos flujos de acciones quedan cubiertos por las funcionalidades que ofrece la interfaz.

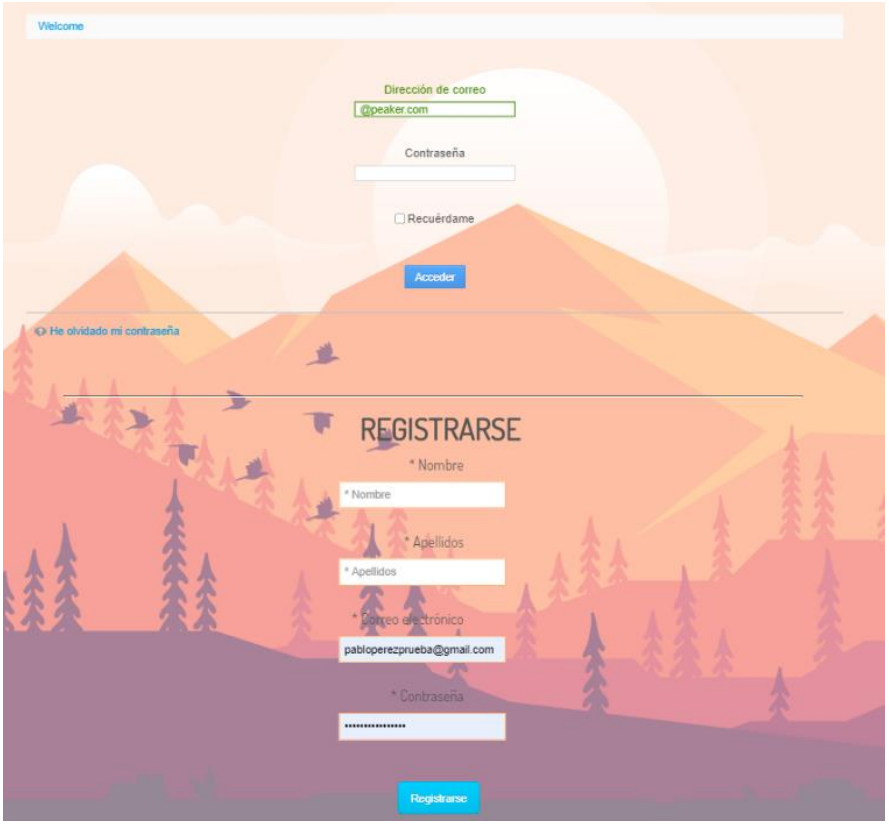


Figura 27 Interfaz Inicio

PI 1 Inicio
Casos de uso cubiertos
CU1 Registro, CU2 Login, CU3 Recuperar contraseña
Descripción de la interfaz
La pantalla está formada por dos formularios, el superior es el formulario de acceso a la aplicación donde se deberá de indicar la siguiente información • Email • Contraseña El formulario inferior es el de registro donde se deberá especificar los siguientes campos • Nombre • Apellidos • Email • Contraseña Al pulsar el botón He olvidado mi contraseña el formulario de login se sustituirá por este otro formulario con los siguientes campos: • dirección de correo • Texto de verificación Tras pulsar el botón *Envía la nueva contraseña* se enviará un email al usuario para que pueda resetarla

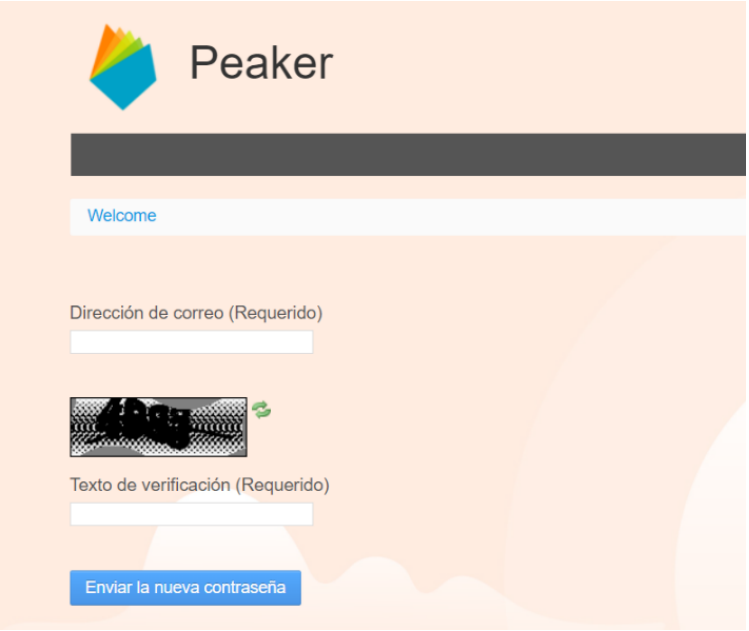


Figura 28 Interfaz Recuperación contraseña

P1 Inicio
PI 1.1 Recuperar Contraseña
Descripción de la interfaz
La pantalla está formada por las secciones, Crear viaje redirige al usuario a *PI Crear Viaje* y Buscar viajes redirige al usuario a PI Buscador Viajes provincia Origen

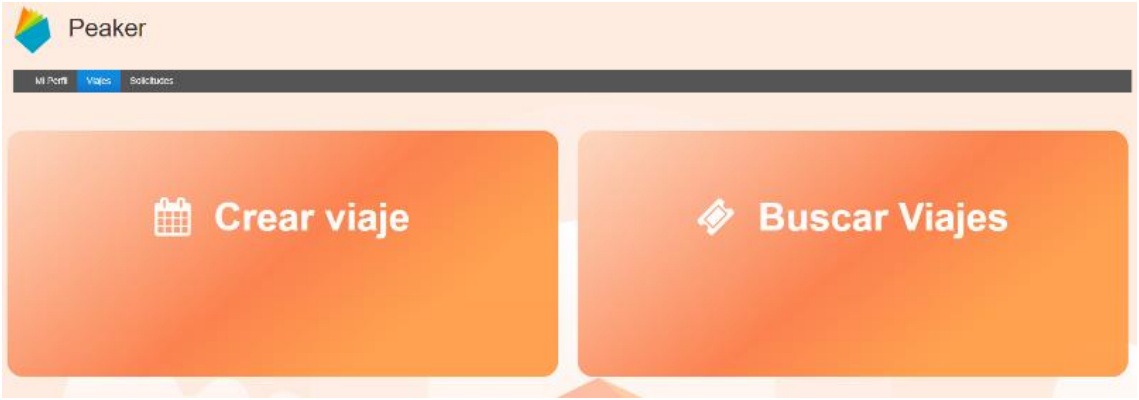


Figura 29 Interfaz Dashboard Viajes

P2 Viajes
PI 2.1 Dashboard Viajes
Descripción de la interfaz
La pantalla está formada por las secciones, Crear viaje redirige al usuario a *PI Crear Viaje* y Buscar viajes redirige al usuario a PI Buscador Viajes provincia Origen

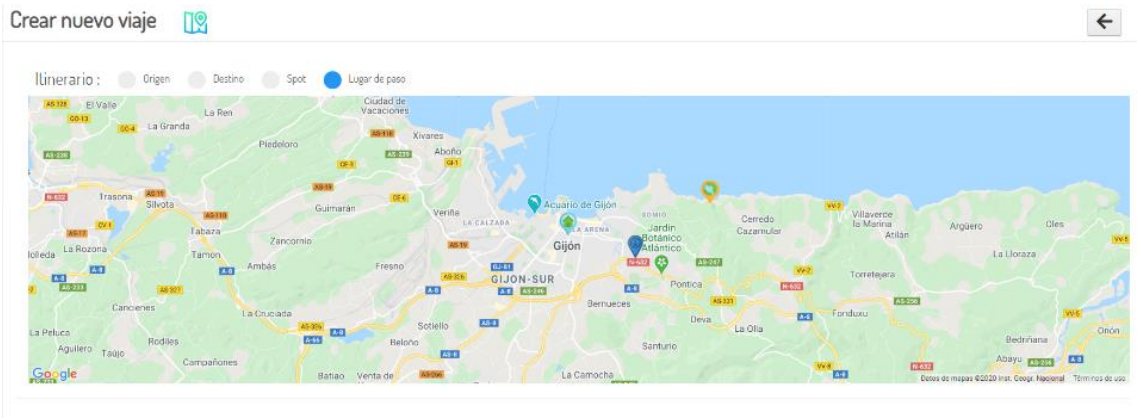


Figura 30 Interfaz Crear Viaje

PI 2.2 Crear viaje
Casos de uso cubiertos
CU8 Crear viaje
P.2.2.1 Mapa itinerario
Descripción de la interfaz
En el mapa se definirá el itinerario a través de diferentes tipos de marcadores, el tipo de marcador se selecciona a través de los radio buttons encima del mapa. Los tipos de marcadores son los siguientes • Origen: determina el punto de partida del viaje. • Destino: determina el destino del viaje. • Spot: muestra los Spots definidos para tu provincia. • Lugar paso: lugar por el que se puede pasar a recoger a las personas que no puedan quedar en el *Origen* Un Spot es un lugar habitual donde se realiza una actividad en dentro de una provincia, la utilización de Spots servirá para facilitar a los usuarios la búsqueda de destinos y facilitar una organización de los destinos de los viajes.

Figura 31 Interfaz Seleccionar Spot

P.2.2.2 Seleccionar Spot como destino
Descripción de la interfaz
Al seleccionar el botón Spots se mostrarán en el mapa diferentes marcadores, al pulsar sobre uno de los marcadores se mostrará una ventana de información con el nombre del spot y un botón para seleccionarlo como destino. El marcador Spot es equivalente a destino por lo que si previamente hay seleccionado un marcador destino se borrará y viceversa

Figura 32 Interfaz Actividades Tipo Costa

P.2.2.3 Seleccionar actividad
Descripción de la interfaz
Inicialmente el tipo de destino de las actividades que se muestran es Costa, a la derecha se muestran los iconos de las diferentes actividades a realizar. Al pulsar sobre uno de los iconos de actividad quedará seleccionada el deporte que se va a realizar.

Figura 33 Interfaz Actividades Tipo Montaña

P.2.2.4 Cambiar tipo destino
Descripción de la interfaz
Al pulsar sobre uno de los iconos de tipos de destino se mostrarán las actividades correspondientes

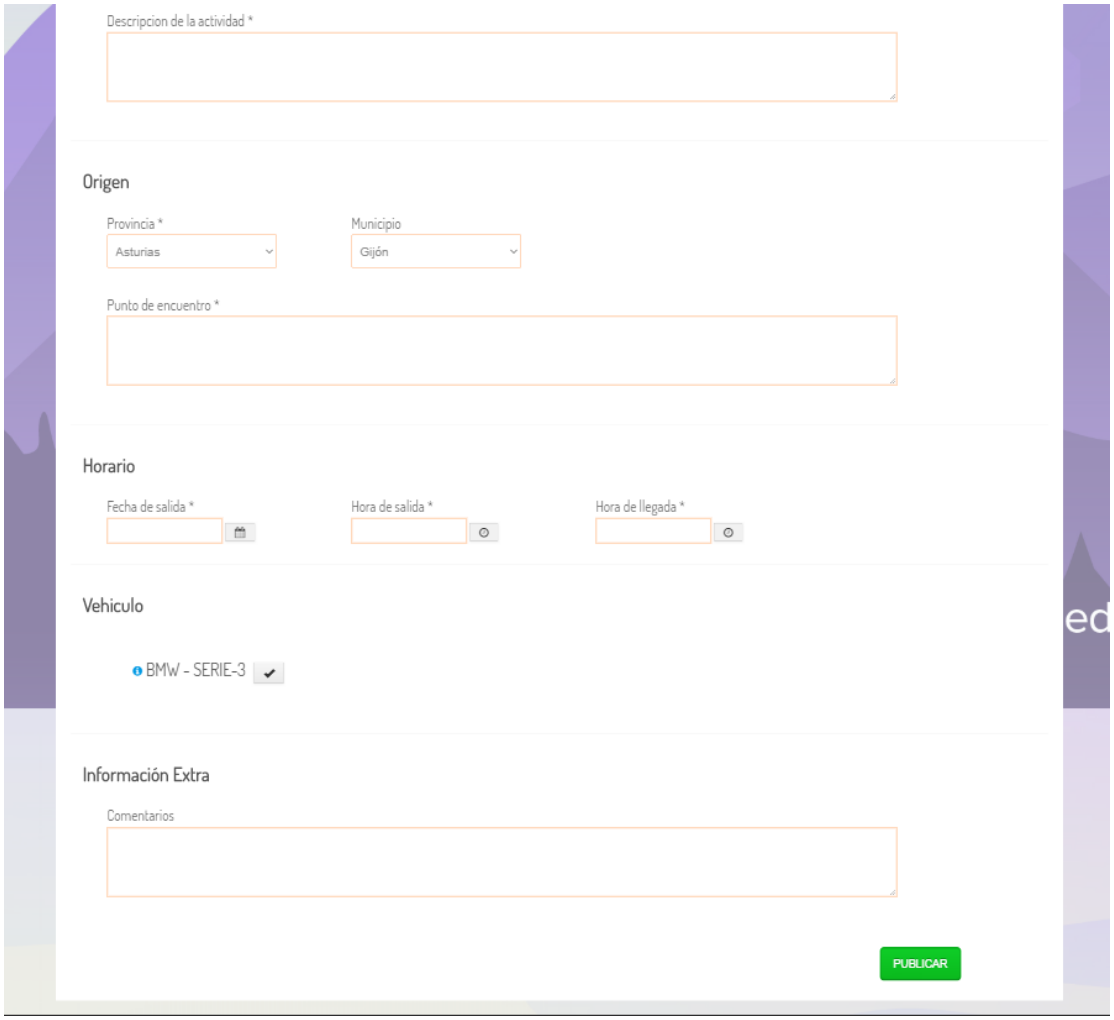


Figura 34 Interfaz Formulario Crear Viaje

PI 2.2 Crear viaje
P.2.2.5 Formulario información viaje
Descripción de la interfaz
El formulario de información del viaje consta de los siguientes campos: • Descripción de la actividad • Provincia • Municipio • Punto de encuentro: descripción del lugar de encuentro en el municipio origen. • Fecha del viaje • Hora de salida • Hora de llegada • Vehículos: en esta parte se mostrará la información relevante de los vehículos registrados por el usuario y permitirá elegir uno de ellos • Comentarios: información extra sobre el viaje Cuando el formulario este completo se registrará el viaje al pulsar en Crear Viaje

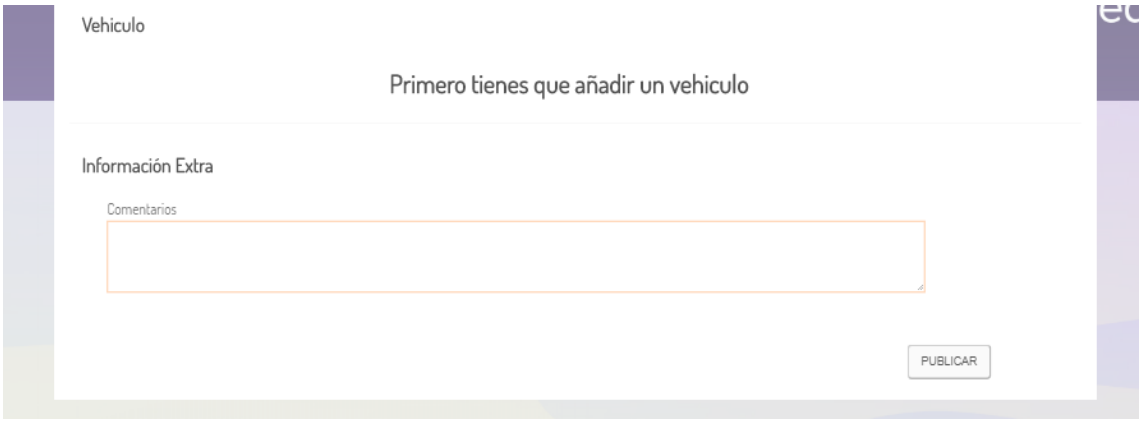


Figura 35 Interfaz Crear Viaje sin vehículo

P.2.2.6 Crear viaje sin vehículo registrado
Descripción de la interfaz
Al acceder a la página de Crear viaje se mostrará un aviso en la parte superior y en la sección vehículos además de tener deshabilitado el botón de Crear

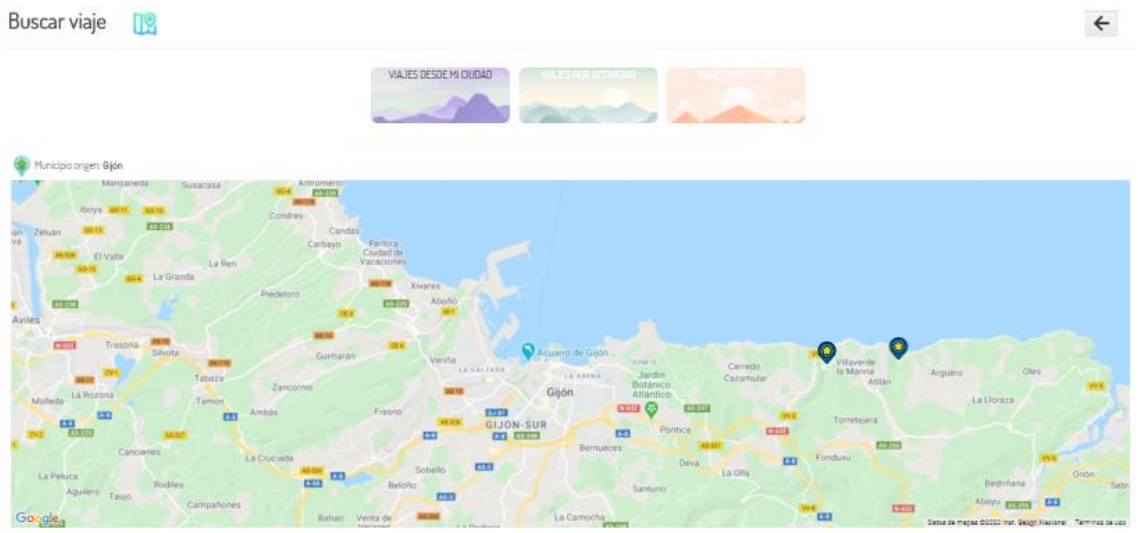


Figura 36 Buscador viajes provincia origen

PI 2.3 Buscador viajes provincia origen
Casos de uso cubiertos
CU9 Buscar viaje
P.2.3.1 Navegación entre buscadores
Descripción de la interfaz
Todos los buscadores tienen una parte común compuesta por 3 pestañas donde se accede al tipo de buscador correspondiente.
P.2.3.2 Mapa
Descripción de la interfaz
Se mostrarán los destinos de los viajes que cumplan los requisitos de búsqueda. Se visualizarán dos tipos de marcadores: • Destino: marcadores individuales con los destinos de los viajes. Al pulsar sobre un marcador *Destino* se mostrará información sobre el viaje • Spot: marcadores donde se agrupan todos los viajes cuyo destino sea el Spot. Al pulsar sobre un marcador *Spot* se mostrará información sobre el viaje En los dos tipos de marcadores al pulsar sobre el botón Mas Información se enviará al usuario a *PI Ficha Viaje*

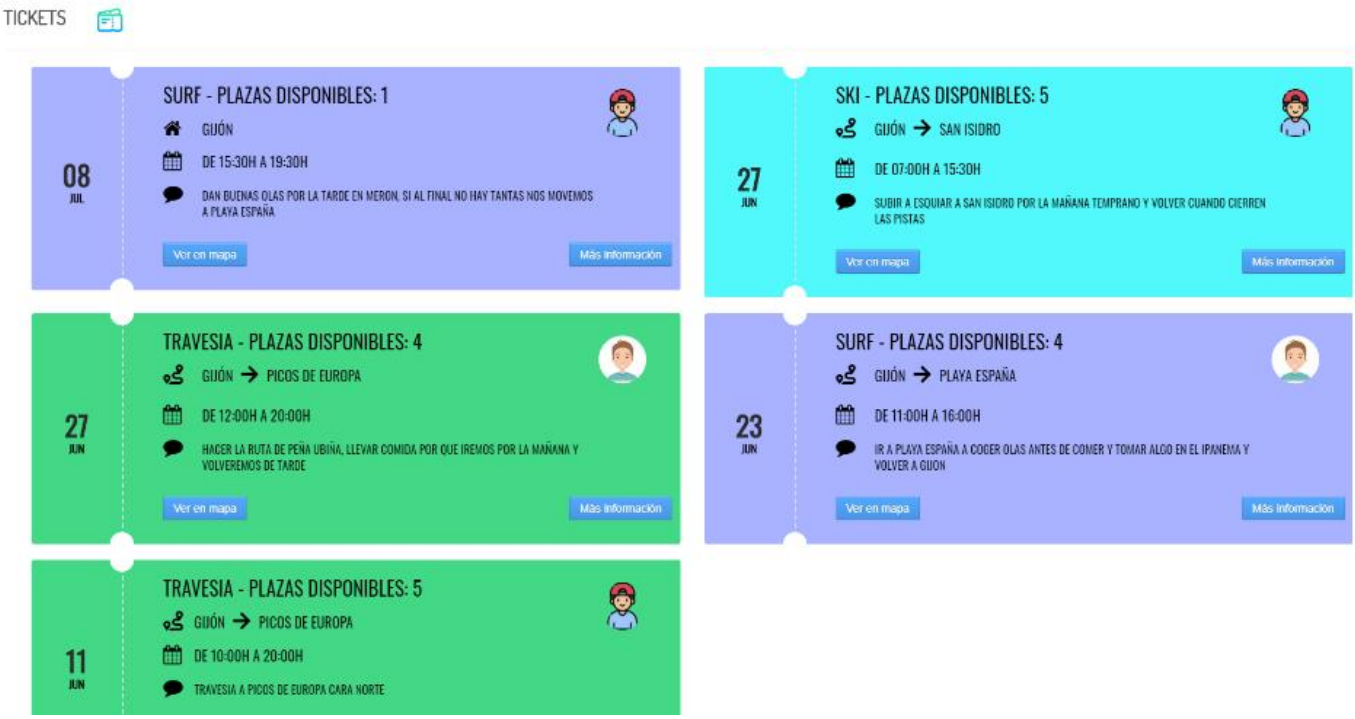


Figura 37 Tickets viajes

PI 2.3 Buscador viajes provincia origen
P.2.3.3 Tickets
Descripción de la interfaz
La información de los viajes se presentará en forma de ticket donde se mostrará la siguiente información: • Actividad: Cada tipo de actividad tiene asignado un color de ticket diferente • Plazas disponibles • Fecha viaje • Origen / Destino • Hora salida / Hora llegada • Información sobre la actividad • Imagen del usuario organizador Al pulsar sobre el botón Mas Información se enviará al usuario a *PI Ficha Viaje*.

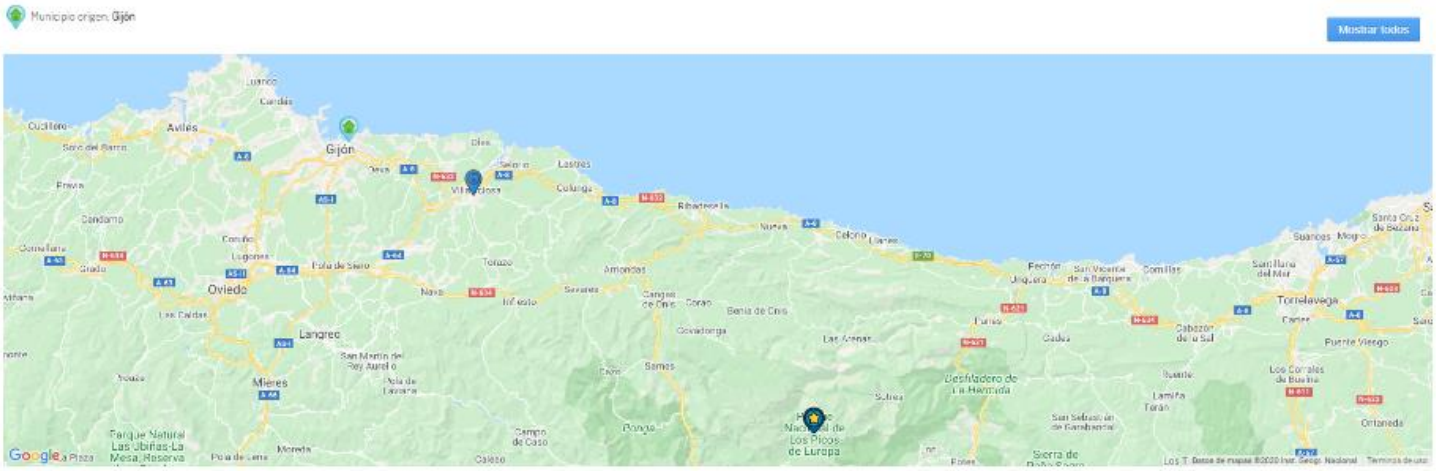


Figura 38 Interfaz itinerario del viaje

PI 2.3 Buscador viajes provincia origen
P.2.3.4 Ver itinerario de un viaje
Descripción de la interfaz
Al pulsar sobre el botón Ver en mapa de uno de los tickets de viaje, se mostrará en el mapa todos los marcadores definidos para el itinerario del viaje. Se mostrará el botón Mostrar todos, al pulsar sobre él se volverán a visualizar todos los marcadores de los viajes que cumplan los requisitos.

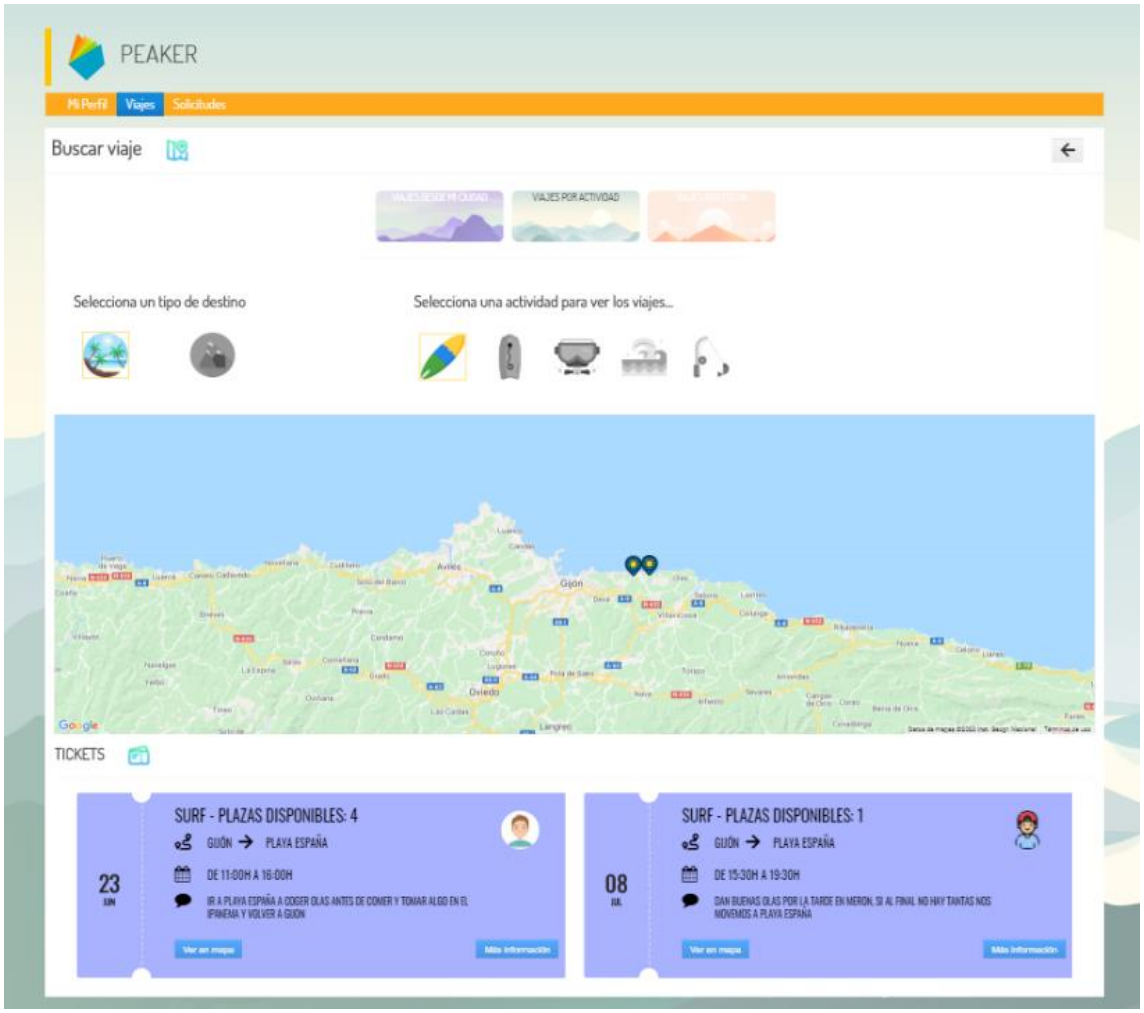


Figura 39 Interfaz Buscador viajes por actividad

PI 2.4 Buscador viajes actividad
P.2.4.1 Resultados viajes por actividad
Casos de uso cubiertos
CU9 Buscar viaje
Descripción de la interfaz
Nota: el comportamiento de los tres tipos de buscadores es similar por lo que solo se abordarán los aspectos diferentes al resto Inicialmente se encuentra seleccionado el tipo de destino Costa y se mostrarán las actividades correspondientes. Al pulsar sobre una actividad se mostrará como seleccionada y se visualizarán los marcadores y tickets de las actividades que cumplan con los requisitos. Si no existen viajes se mostrará un aviso**.**

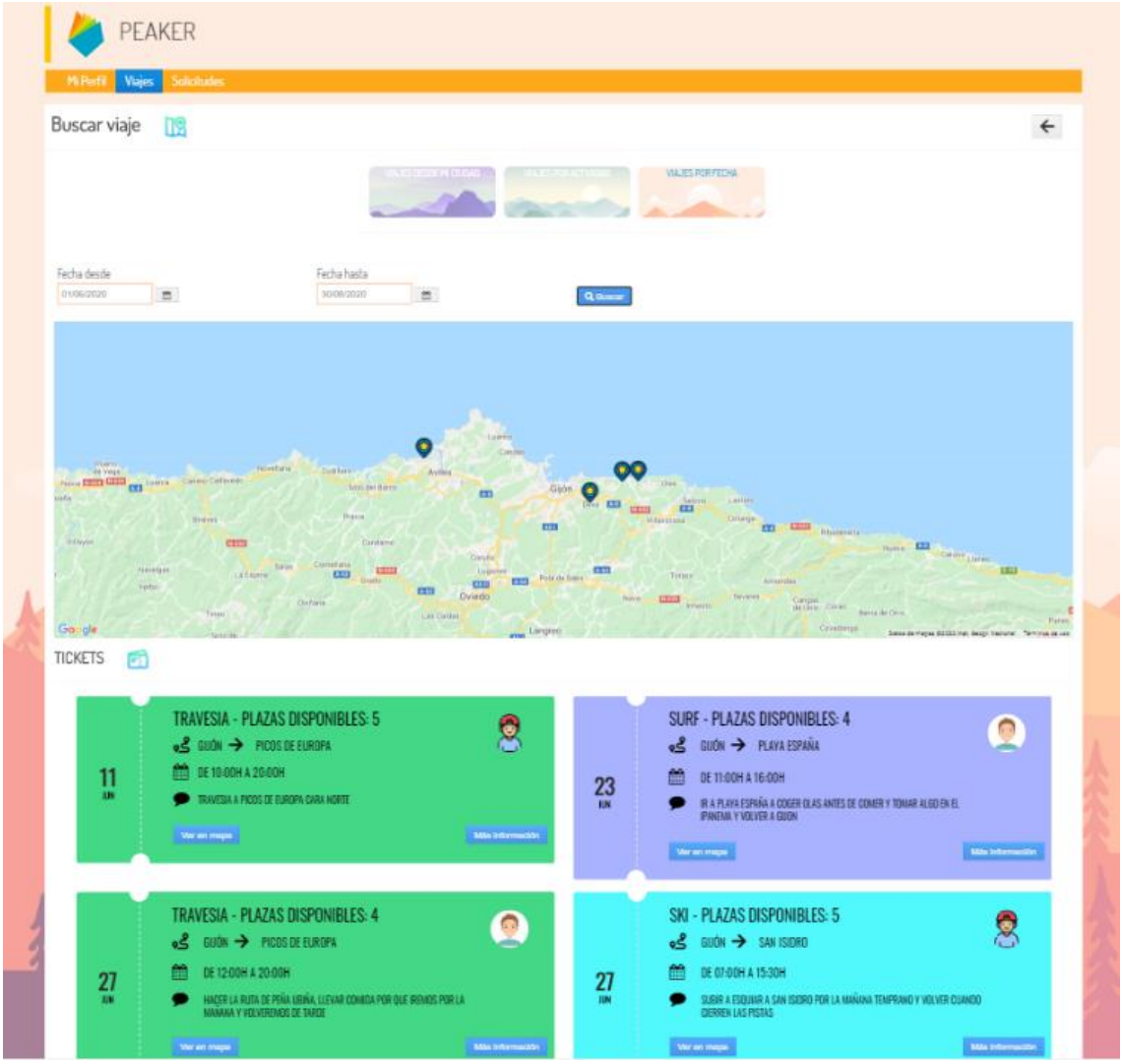


Figura 40 Interfaz Buscador viajes por fecha

PI 2.5 Buscador viajes por fecha
P.2.5.1 Ver itinerario de un viaje
Casos de uso cubiertos
CU9 Buscar viaje
Descripción de la interfaz
Nota: el comportamiento de los tres tipos de buscadores es similar por lo que solo se abordarán los aspectos diferentes al resto En la parte superior del mapa hay un formulario con dos campos: Fecha desde y Fecha hasta. Al pulsar sobre el botón Buscar se mostrarán los marcadores y tickets de los viajes cuya Fecha viaje se encuentre en el rango **de fechas establecido**

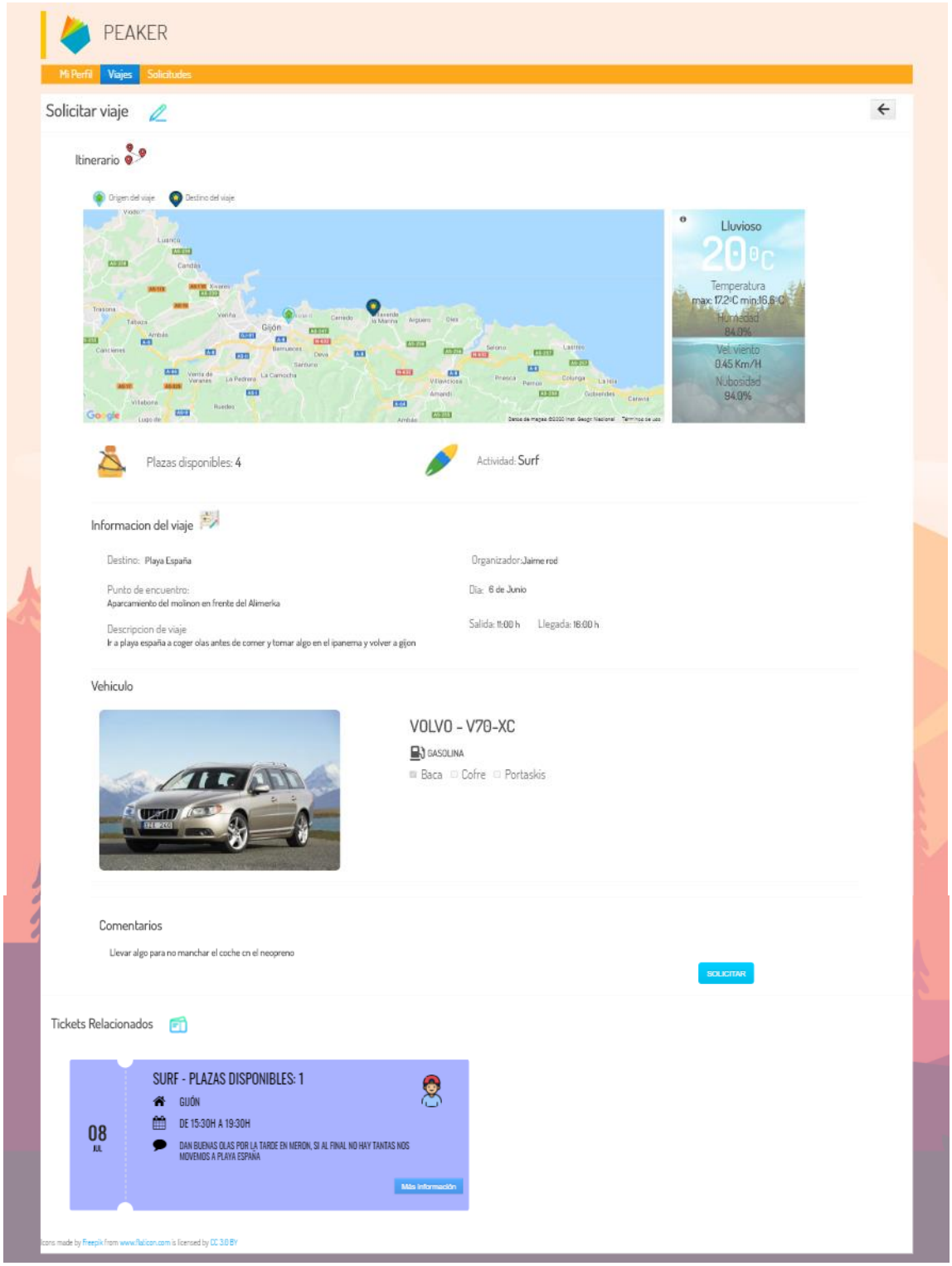


Figura 41 Interfaz Información del viaje

PI 2.6 Ficha del viaje
P.2.6.1 Información del viaje
Casos de uso cubiertos
CU10 Cancelar viaje, CU11 Solicitar viaje
Descripción de la interfaz
La información que se visualiza en esta página se dividirá en 4 secciones: • Mapa: donde se muestra el itinerario del viaje. • Información meteorológica: es consultada a tiempo real en el lugar de destino del viaje y se muestra la información más relevante • Información del viaje: la información completa definida por el usuario • Tickets relacionados: se muestra un listado de tickets que están relacionados, mismo origen del viaje y actividad a realizar, con el viaje que se está consultando. Al pulsar sobre el botón Más información se lleva al usuario a la Ficha del viaje seleccionado.

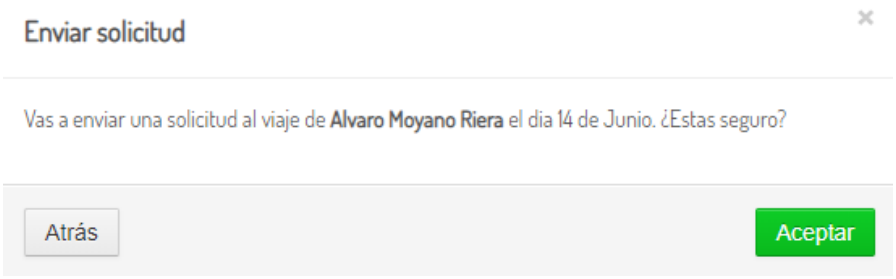


Figura 42 Interfaz Enviar solicitud

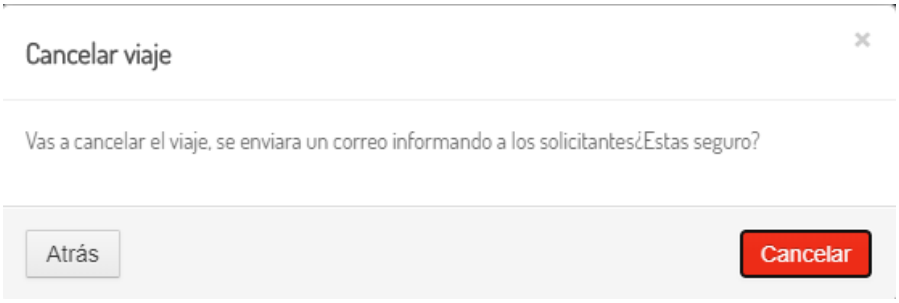


Figura 43 Interfaz Cancelar Viaje

P.2.6.2 Solicitar viaje
Descripción de la interfaz
Si el usuario, que no sea el organizador, accede al viaje podrá enviar una solicitud pulsando sobre el botón Solicitar y confirmando la acción en el pop up que se le mostrará. Si el usuario ha completado sus plazas disponibles o ha alcanzado el número máximo de solicitudes recibidas no se visualizará este botón y se mostrará un aviso.
P.2.6.3 Cancelar viaje
Descripción de la interfaz
Si el usuario organizador, accede al viaje en lugar del botón solicitar se le mostrará el botón Cancelar. Podrá cancelar el viaje pulsando sobre el botón y confirmando la acción en el pop up que se le mostrará.

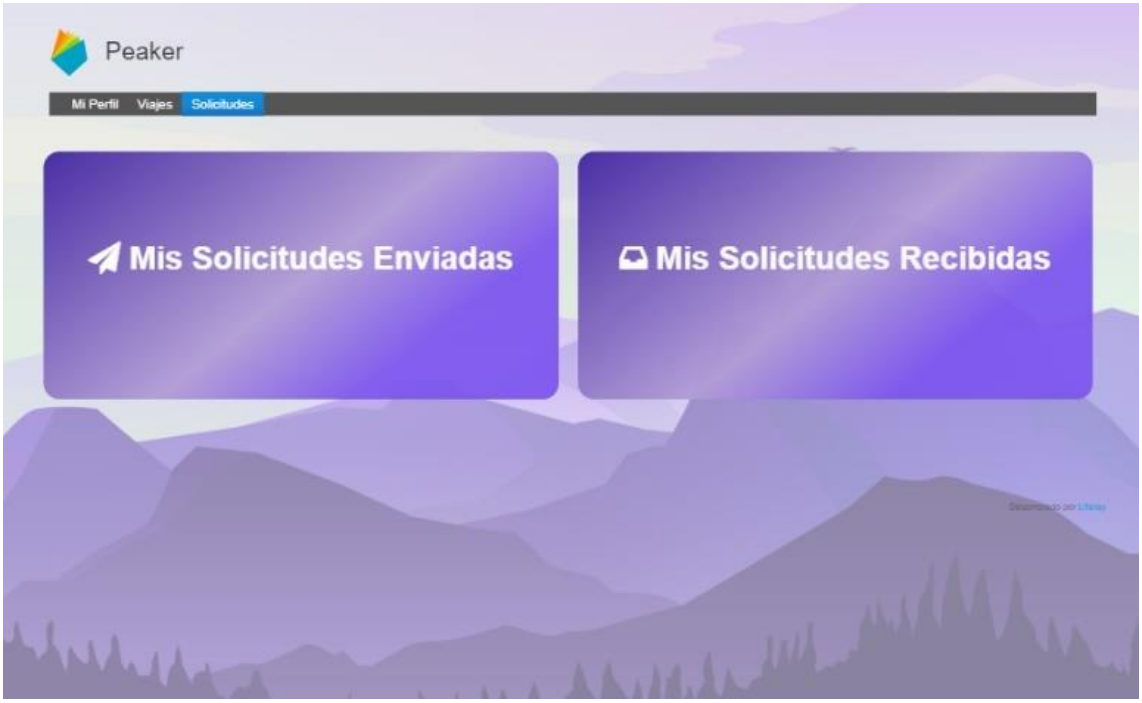


Figura 44 Interfaz Dashboard solicitudes

PI 3 Solicitudes
PI 3.1 Dashboard Viajes
Descripción de la interfaz
La pantalla está formada por las secciones, Mi Solicitudes Enviadas que viaje redirige al usuario a *PI Solicitudes Enviadas* y Mis Solicitudes Recibidas redirige al usuario a *PI Viajes Activos.*

Figura 45 Interfaz Solicitudes Enviadas Viaje

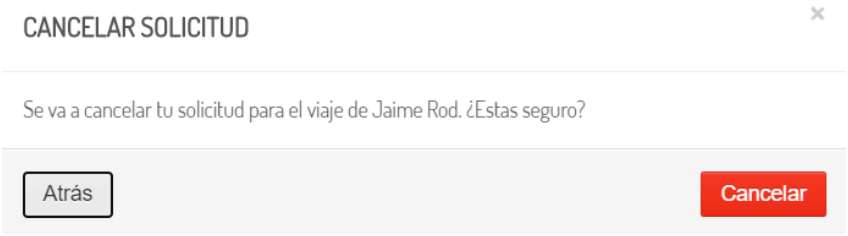


Figura 46 Interfaz Cancelar Solicitud

PI 3.2 Solicitudes Enviadas
Casos de uso cubiertos
CU12 Cancelar solicitud
Descripción de la interfaz

Se muestra una tabla con todas las solicitudes enviadas ordenadas por fecha de envío. Los campos que muestra la tabla son los siguientes
• Identificador Viaje
• Organizador viaje
• Actividad
• Destino
• Hora salida
• Hora llegada
• Fecha del viaje
• Contacto: campo que solo será visible si la solicitud se encuentra en estado *ACEPTADA*
• Estado solicitud:
  ○ *PENDIENTE* si aún no ha sido resuelta por el usuario organizador.
  ○ *ACEPTADA* o *RECHAZADA*: por el usuario organizador.
  ○ *CANCELADA* : por el usuario solicitante.
Al pulsar sobre el botón información se llevará al usuario a PI Ficha Viaje.

PI 3.2.1 Cancelar solicitud
Descripción de la interfaz
Al pulsar sobre el botón Cancelar y confirmar la acción en el pop up, la solicitud pasará a estado *CANCELADA* y se notificará al organizador del viaje

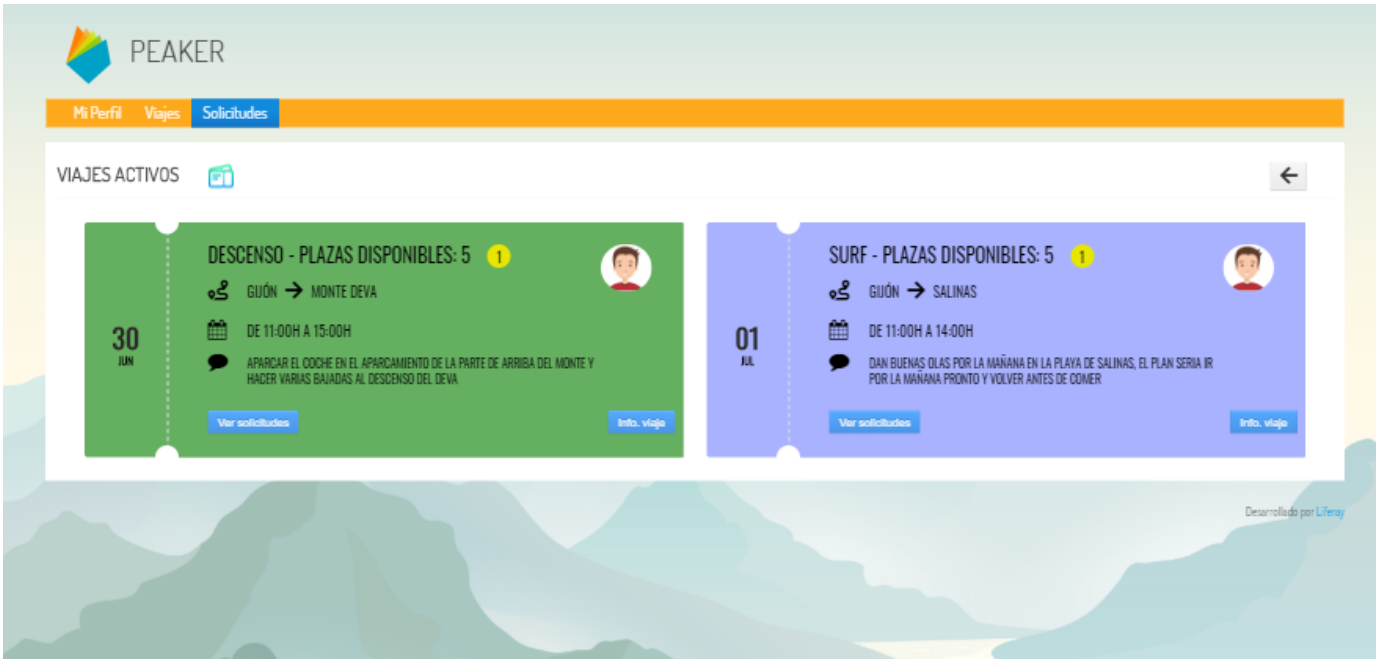


Figura 47 Interfaz Viajes activos

PI 3.3 Viajes activos
Descripción de la interfaz
Nota: los viajes se presentan con el formato de ticket utilizado en otras partes de la aplicación, solo mencionaremos los aspectos diferentes en esta interfaz. A la derecha de las plazas disponibles se mostrará un círculo amarillo con un número que indica el número de solicitudes pendientes de resolución que ha recibido el viaje. Al pulsar el botón *Info. Viaje* llevará al usuario a *PI Ficha viaje* . Al pulsar sobre Ver Solicitudes se llevará al usuario a *PI Solicitudes Recibidas.*

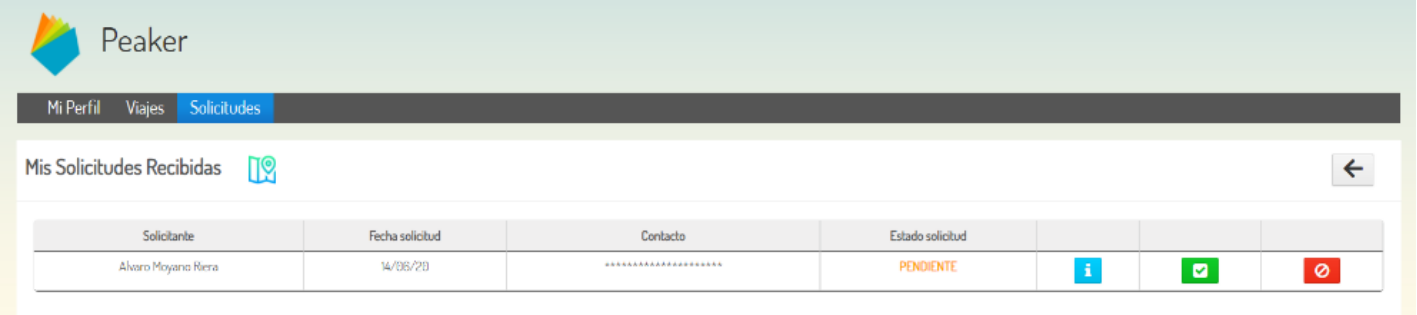


Figura 48 Interfaz Solicitudes Recibidas Viaje

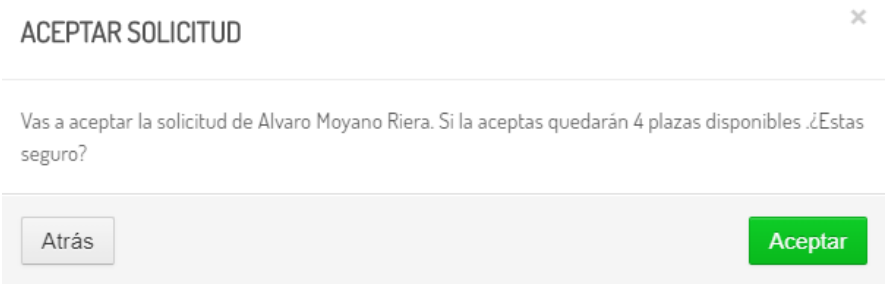


Figura 49 Interfaz Aceptar solicitud

PI 3.4 Solicitudes Recibidas
Casos de uso cubiertos
CU13 Aceptar solicitud, CU14 Rechazar solicitud
Descripción de la interfaz

Se muestra una tabla con todas las solicitudes enviadas ordenadas por fecha de envío. Los campos que muestra la tabla son los siguientes
• Solicitante
• Fecha solicitud
• Contacto
• Estado solicitud:
  ○ *PENDIENTE* si aún no ha sido resuelta por el usuario organizador.
  ○ *ACEPTADA* o *RECHAZADA*: por el usuario organizador
  ○ *CANCELADA* : por el usuario solicitante
Al pulsar sobre el botón información se llevará al usuario a PI Ficha **Viaje**

PI 3.4.1 Aceptar solicitud
Descripción de la interfaz
Al pulsar sobre el botón Cancelar y confirmar la acción en el pop up, la solicitud pasará a estado *ACEPTADA* y se notificará al organizador del viaje. También ambos usuarios visualizarán el campo contacto para poder contactar fuera de la aplicación

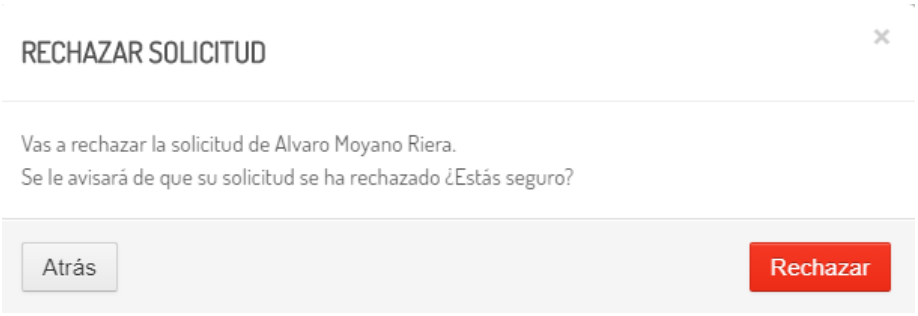


Figura 50 Interfaz Rechazar Solicitud

PI 3.4.2 Rechazar solicitud
Descripción de la interfaz
Al pulsar sobre el botón Rechazar y confirmar la acción en el pop up, la solicitud pasará a estado *RECHAZADA* y se notificará al solicitante del viaje.

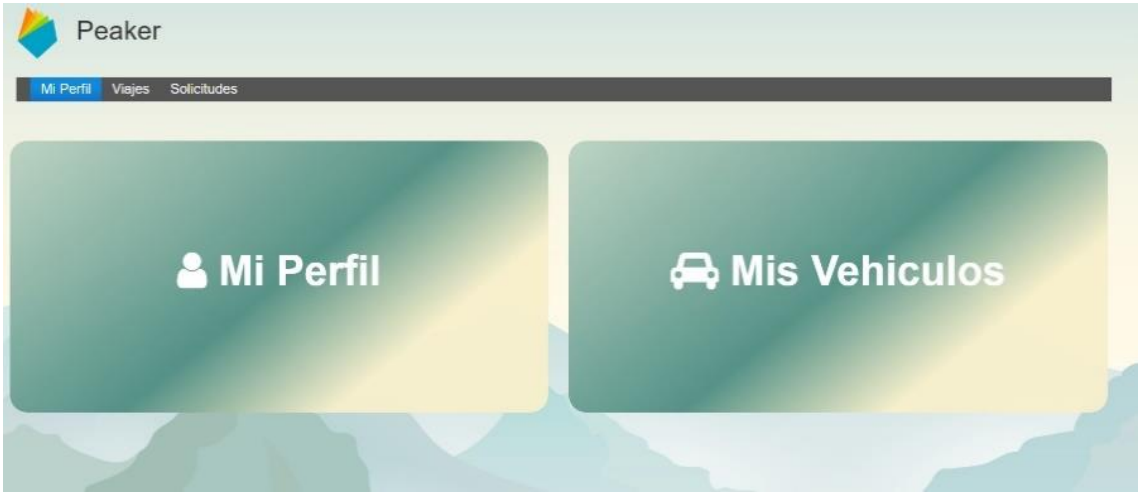


Figura 51 Interfaz Dashboard Mi Perfil

PI 4 Mi perfil
PI 4 Dashboard Mi perfil
Descripción de la interfaz
La pantalla está formada por las secciones, Mi Perfil que viaje redirige al usuario a *PI Perfil Usuario* y Mis Vehículos redirige al usuario a *PI Listado vehículos*

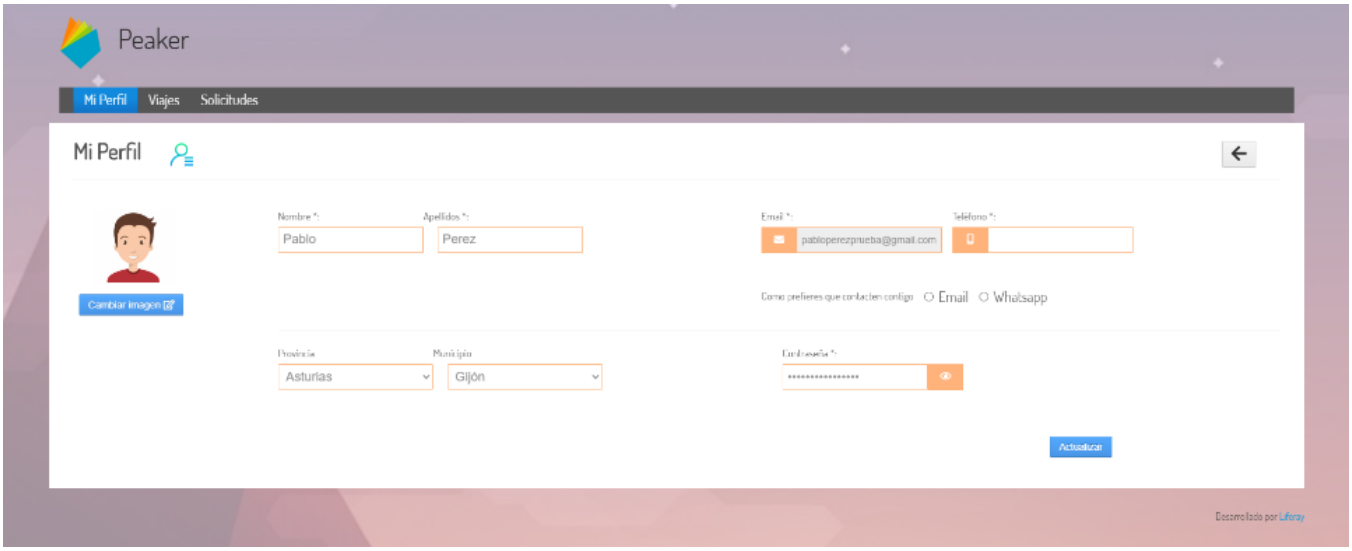


Figura 52 Interfaz Mi perfil

PI 4.2 Formulario edición del perfil
Casos de uso cubiertos
CU4 Editar perfil
Descripción de la interfaz
El formulario de información del viaje consta de los siguientes campos: • Nombre • Apellidos • Email (no editable) • Teléfono • Provincia • Municipio • Contraseña • Imagen usuario Cuando el formulario este completo actualizará la información de perfil al pulsar sobre *Actualiza,* si hay algún error en los datos del formulario se mostrará un aviso al usuario

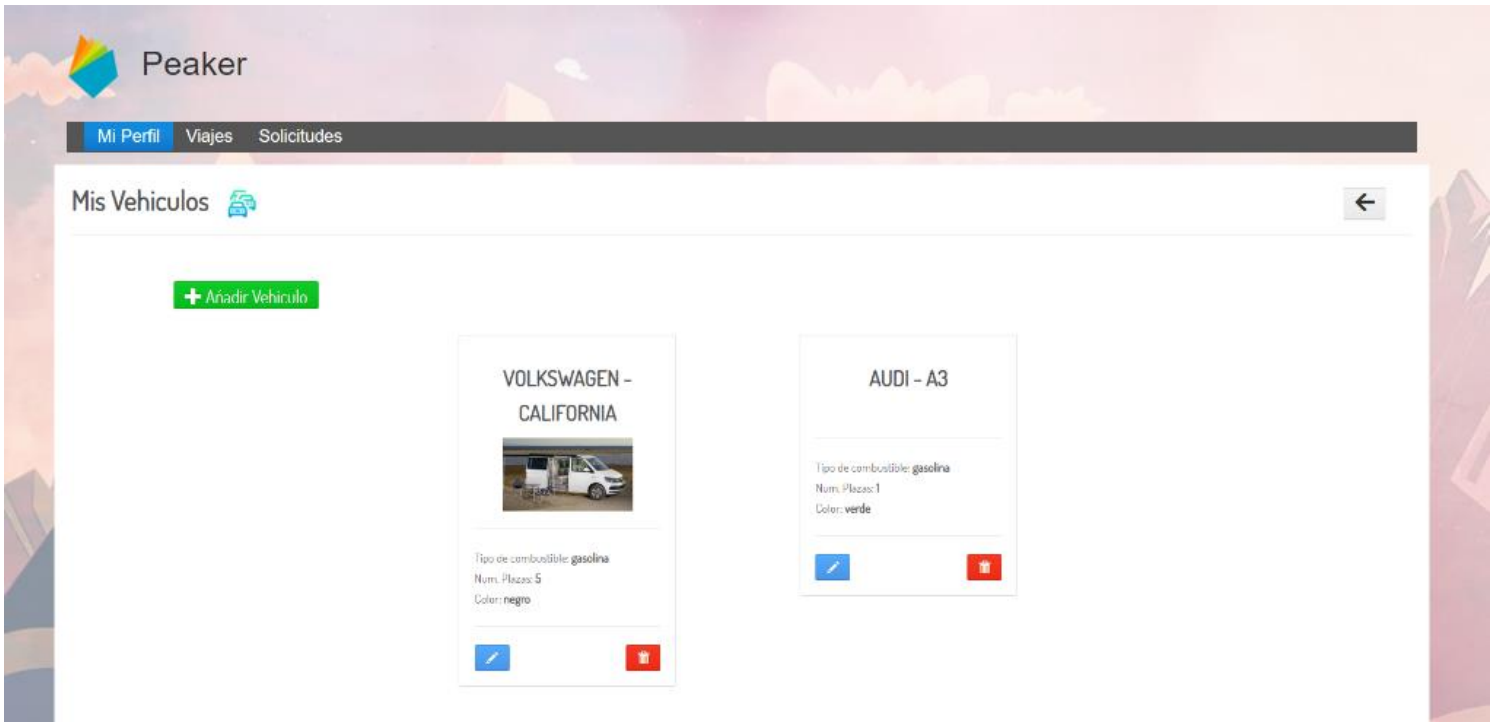


Figura 53 Interfaz Mis vehículos

PI 4.3 Mis vehículos
PI 4.3.1 Listado vehículos
Casos de uso cubiertos
CU5 Añadir vehículo, CU6 Editar vehículo, CU7 Borrar vehículo
Descripción de la interfaz
Se muestran los vehículos registrado a través de una ficha con la siguiente información: • Marca • Modelo • Tipo combustible • Plazas • Color • Imagen (si dispone de ella) *Al pulsar sobre el botón Añadir vehículo llevará al usuario a PI Añadir vehículo* Cada ficha dispondrá de dos botones: • Editar: redirige a *PI Editar vehículo.* • Borrar: elimina el vehículo, pasa a estado cancelado los viajes relacionados con él y notifica a los usuarios con notificaciones pendientes si los hubiera.

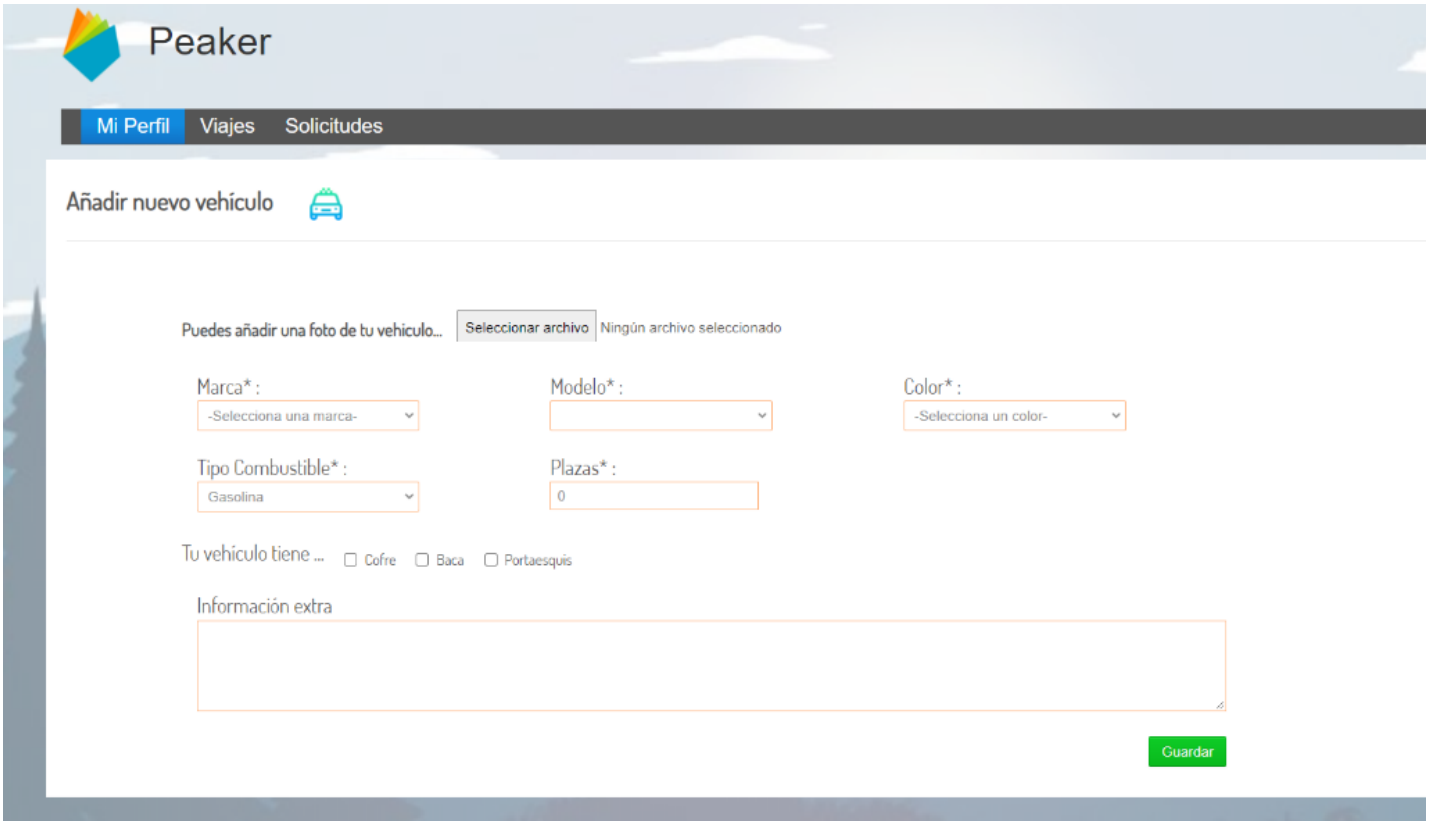


Figura 54 Interfaz Añadir nuevo vehículo

PI 4.3.2 Añadir vehículo
Descripción de la interfaz

Se muestra un formulario con la siguiente información a completar:
• Marca
• Modelo
• Tipo combustible
• Plazas
• Color
• Imagen
• Información extra
• Si el coche dispone de
  ○ Porta esquís
  ○ Vaca
  ○ Cofre
Al pulsar sobre el botón *Guardar* se guardar el vehículo y se redirige al usuario *a PI Mis vehículos* o bien se mostrará un aviso con la información incorrecta

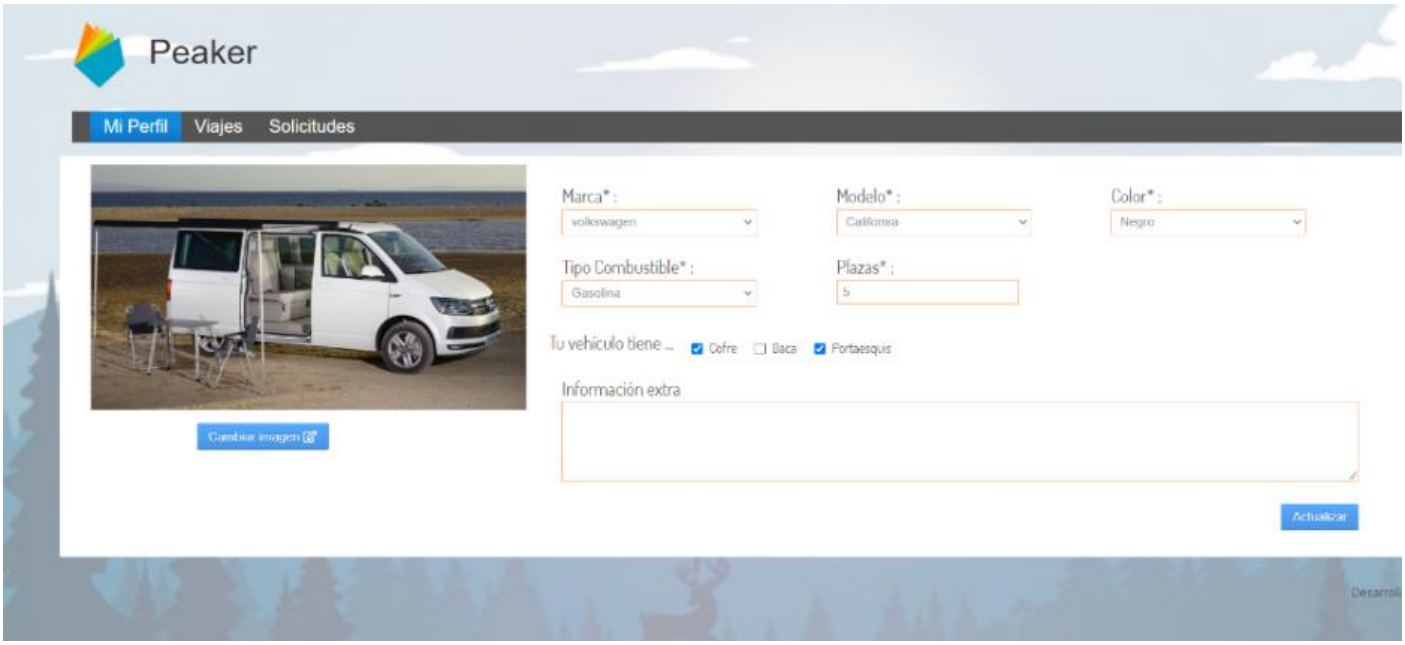


Figura 55 Interfaz Editar vehículo

PI 4.3.3 Editar vehículo
Descripción de la interfaz
El formulario inicialmente mostrará la información actual del vehículo, cualquier campo se podrá modificar y al pulsar sobre el botón actualiza se actualizará la información del vehículo o se mostrará un aviso si algún campo es incorrecto.

6. Implementación de la aplicación

El desarrollo de los portlets de la aplicación se ha llevado a cabo utilizando la plataforma de desarrollo Eclipse que es la más utilizada para el desarrollo de proyectos Java. A pesar de ser un IDE que acepta diversos lenguajes de programación, proporciona al desarrollador herramientas de desarrollo Java, una interfaz gráfica que facilita la navegación entre las clases del proyecto y asistentes para la gestión de proyectos además de un potente y visual depuradora que es de gran importancia para el desarrollo del proyecto.

Cabe la posibilidad de añadir plugins como *Liferay IDE* que es una herramienta para realizar dentro de eclipse el compilado, empaquetado y despliegue de portlets, a pesar de esto no se ha utilizado pues personalmente prefiero realizar estas acciones a través de línea de comandos utilizando *Maven*. Respecto a la arquitectura lógica que ha dado soporte a la aplicación en el entorno de desarrollo es la siguiente:

- Sistema Operativo: Windows 10 Pro
- Servidor Web: Apache Tomcat / 7.0.62
- Portal Web : Liferay Portal 6.2 CE ga6
- Base de datos: MySQL 5.7

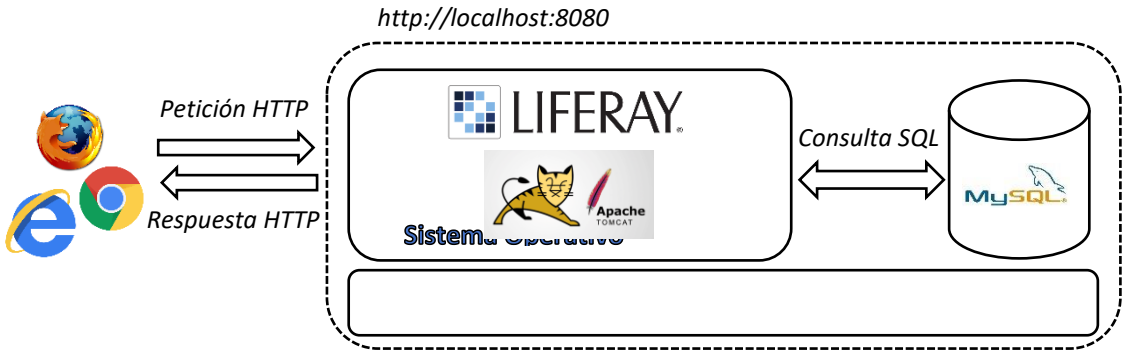


Figura 56 Arquitectura lógica de la aplicación

Por el momento la aplicación solo se ha implementado en un equipo personal, si se quisiera llevar a un entorno de producción será necesario implementar la misma arquitectura en un servidor dedicado Posteriormente será necesario montar la misma arquitectura que en el entorno de desarrollo y realizar las configuraciones de red y seguridad necesarias para exponer nuestro servidor apache en el puerto 80 para poder recibir peticiones HTTP.

Se ha simulado la contratación de uno con la siguiente configuración:

- Windows Server 2012 Standard
- Procesador Intel Xeon E3
- 32 GB RAM
- 2 x 2 TB de almacenamiento SATA
- Ancho de banda 500 Mb/s

Una se obtenga el mínimo producto viable la aplicación piloto se publicará en Asturias, un lugar que dispone de gran variedad de espacios naturales donde practicar diferentes tipos deportes tanto de costa como montaña. De esta manera se podrá estudiar en un contexto real la viabilidad de la solución propuesta e implementar las mejoras necesarias para posteriormente implantar el sistema en más lugares.

6.1 Tecnologías empleadas

A continuación, se describirán las diferentes tecnologías que se han utilizado para construir Peaker. La característica que comparten todas las tecnologías empleadas es que son de código libre, por este carácter existen grandes comunidades donde que dan soporte a todo tipo de necesidades.

Liferay Portal 6.2

La primera tecnología que detallaremos es la utilización de *Liferay Portal* para la elaboración de esta aplicación web, ya su uso ha condicionado la elección de otras tecnologías empleadas. Liferay es un software para la gestión de portales de código abierto y escrito en Java, se puede ejecutar en cualquier sistema operativo que tenga instalada una Máquina Virtual Java (JVM).

La utilización de *Liferay Portal* nos facilita gran medida el desarrollo de la aplicación pues nos ofrece funcionalidades para la gestión de contenidos, documentos, permisos de usuarios a través de una interfaz gráfica sencilla pero potente.

En un portal gestionado por *Liferay* se pueden integrar distintos tipos de componentes, los más relevantes en el desarrollo de la aplicación son los portlets, se puede considerar una aplicación web en sí misma con una interfaz propia gestionados por el portal que los contiene, atienden las solicitudes de un cliente web generado contenido dinámico y que exponen servicios para que la información que manejan pueda ser consumida y gestionada por otros portlets.

A continuación, se describen los diferentes tipos de componentes y su funcionalidad:

- **Portlet**: componentes principales, Liferay nos proporciona de manera nativa hasta 60 portlets con diferentes funcionalidades como Calendario, Visualización de contenido web, Comentarios, Buscador de contenidos entre otros muchos. Para desarrollar la aplicación hemos generado nuestros propios portlets adaptados para cumplir con

todos los requisitos establecidos, estos están escritos en Java y las diferentes interfaces que presentan al usuario la información de manera dinámica son Java Server Pages (JSP). Los portlets desarrollados están basados en los estándares JSR 168 y JSR 286, por lo que garantiza una integración totalmente compatible con cualquier portal web que siga estos estándares.

- **Hook**: componentes que permiten la modificación del código nativo del portal permitiendo ampliar o modificar su comportamiento. Las modificaciones que se contemplan son:
    - Personalización de las JSP propias de *Liferay*.
    - Modificación o ampliación de los servicios de *Liferay*
    - Modificar o añadir nuevas propiedades al portal
    - Añadir ficheros .*properties* para realizar la internacionalización del portal

- **Layout**: componente que permite definir una nueva disposición para los elementos que se encuentren en una página. La disposición de los componentes de cada página está definida por una Layout. Nuestra aplicación no utilizará ningún componente de este tipo.

- **Themes**: componente que permite modificar la apariencia general del portal o de una página concreta. Para nuestra aplicación no se necesita desarrollar un componente de este tipo pues se usará el Theme por defecto de Liferay y los aspectos de personalización del aspecto recaerá sobre los portlets.

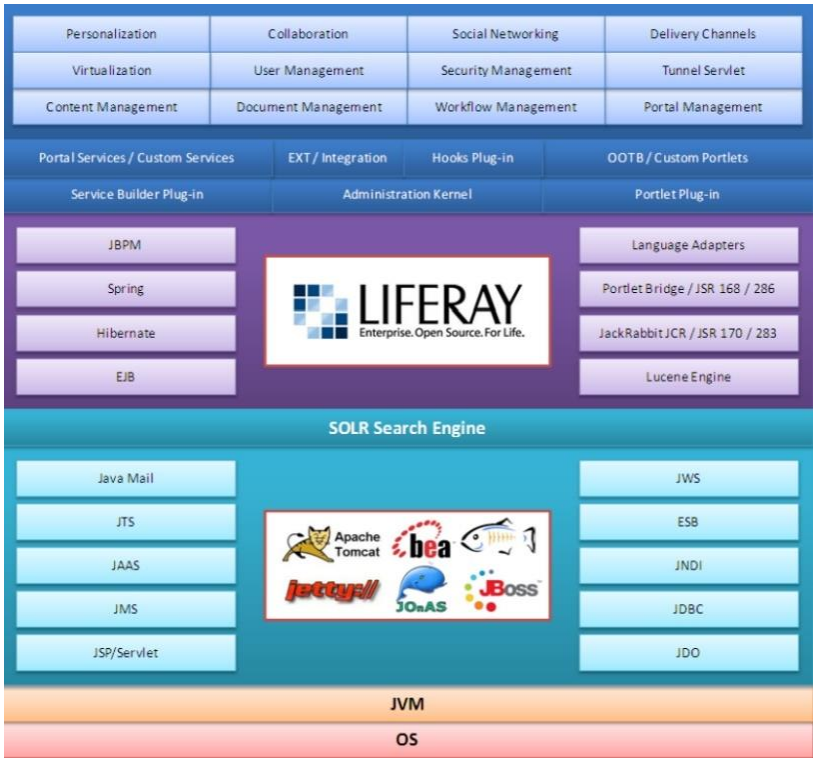


Figura 57 Arquitectura lógica Liferay Portal 6.2

Spring MVC

Es el módulo del framework Spring que proporciona los mecanismos necesarios para el desarrollo de aplicaciones java basadas en el patrón Modelo-Vista-Controlador, que proporciona desarrollos flexibles, reutilización de código y facilita las tareas de configuración. Las características principales de este framework son:

- Acoplamiento débil, separación clara de las diferentes clases: controlador, modelo, validador.
- Reutilización de objetos de negocio para agilizar el desarrollo.
- Soporte para JSPs y otro tipo de motores de plantillas como Velocity o Freemarker.
- Gestión de dependencias entre clases a través de Spring IoC.
- Flexibilidad para la resolución de la visualización de las vistas

Este será el framework a utilizar por los portlets de desarrollo propio, para utilizarlo habrá que definirlo para cada uno de ellos en el fichero de configuración *portlet.xml.*

```
<?xml version="1.0"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <portlet>
        <portlet-name>Administracion-Viajes</portlet-name>
        <display-name>Administracion-Viajes</display-name>
        <portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
        <init-param>
            <name>contextConfigLocation</name>
            <value>/WEB-INF/spring-context/portlet/Administracion-Viajes.xml</value>
        </init-param>
        <expiration-cache>0</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
            <portlet-mode>EDIT</portlet-mode>
        </supports>
        <resource-bundle>content.administracion-viajes.Language</resource-bundle>
        <portlet-info>
            <title>Administracion-Viajes</title>
            <short-title>Administracion-Viajes</short-title>
            <keywords>Administracion-Viajes</keywords>
        </portlet-info>
        <security-role-ref>
            <role-name>administrator</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>guest</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>power-user</role-name>
        </security-role-ref>
        <security-role-ref>
            <role-name>user</role-name>
        </security-role-ref>

    </portlet>
</portlet-app>
```

Figura 58 Fichero de configuración portlet.xml

El tratamiento de peticiones realizadas a uno de nuestros portlets se realiza a través de un portlet central que recibe todas las solicitudes, el DispatcherPortlet. La aplicación java recibe peticiones que son capturadas por dispatcherPortlet y redirigidas al controlador el cual debe ejecutar las acciones lógicas pertinentes y las llamadas a los servicios necesarios. Posteriormente construirá la respuesta con Modelo informado, esto será gestionado por el dispatcherPortlet que asociará la respuesta con la vista a renderizar y enviará la respuesta.

El esquema de procesamiento de peticiones por el DispatcherPortlet se muestra en la siguiente figura.

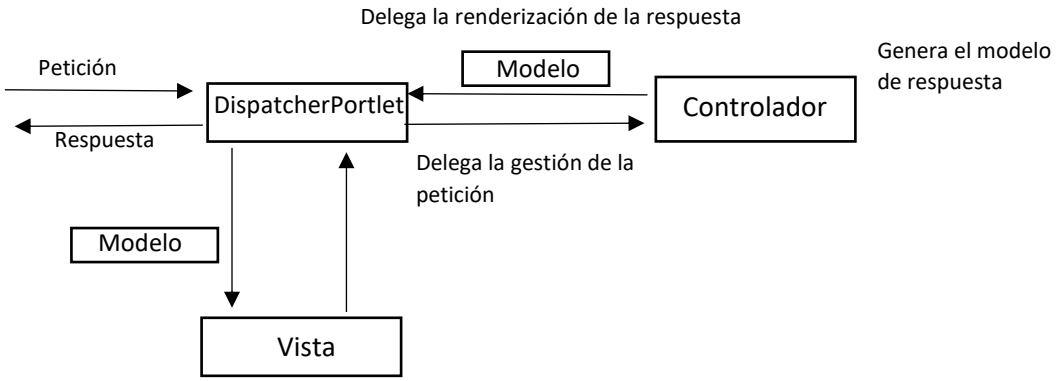


Figura 59 Diagrama procesamiento peticiones Dispatcher Portlet

Maven

Maven es una herramienta software de código abierto que nos facilita la gestión de dependencias y compilación del código Java de los portlets a desarrollar, utiliza un fichero de configuración llamado *pom.xml* donde se incluyen las librerías y las dependencias entre módulos. Cabe destacar que se ha una herramienta Liferay llamada *Service Builder* la cual genera automáticamente las clases y métodos básicos necesarios para el acceso a los diferentes modelos de datos. Para realizar se basa en el fichero service.xml de cada portlet donde se especifica la entidad con sus respectivas columnas y métodos *Finder.*

```
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionsolicitudes</groupId>
    <artifactId>Administracion-Solicitudes-portlet-service</artifactId>
    <version>1.0.1</version>
</dependency>
<dependency>
    <groupId>com.amoyano.peaker.portlet.administracionspots</groupId>
    <artifactId>Administracion-Spots-portlet-service</artifactId>
    <version>1.0.0-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.2.4</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.3</version>
</dependency>
<dependency>
    <groupId>org.codehaus.jackson</groupId>
    <artifactId>jackson-mapper-asl</artifactId>
    <version>1.9.2</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.1</version>
</dependency>
```

Figura 60 Dependencias del portlet Administración-Viajes

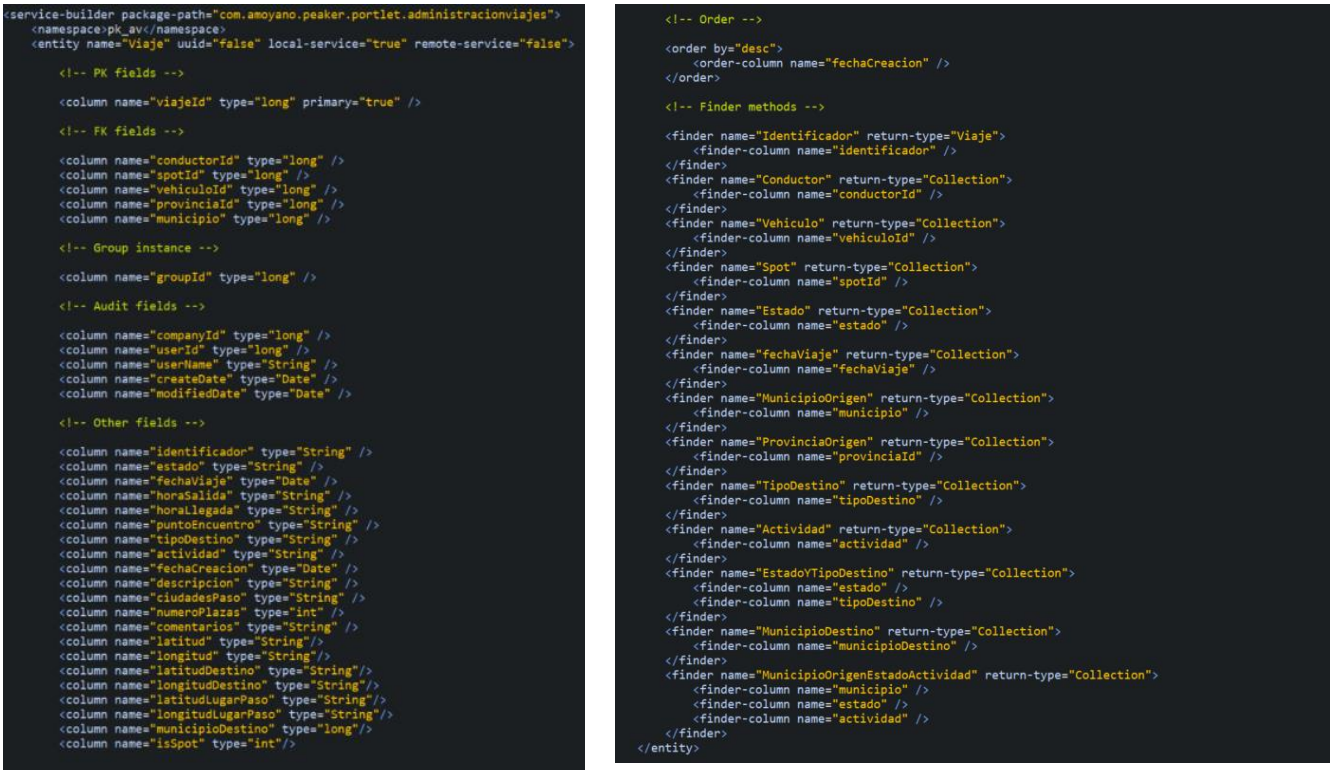


Figura 61 Archivo service.xml del portlet Administración-

Maven utiliza dos repositorios, uno local *.m2* donde guarda una copia de todas las versiones generadas por nuestros portlets para poder incluirlas como dependencias en los pom.xml de nuestros otros proyectos y un repositorio central remoto donde se encuentran prácticamente todas las librerías que se utilizan en los proyectos java y que se descargan cuando es necesario.

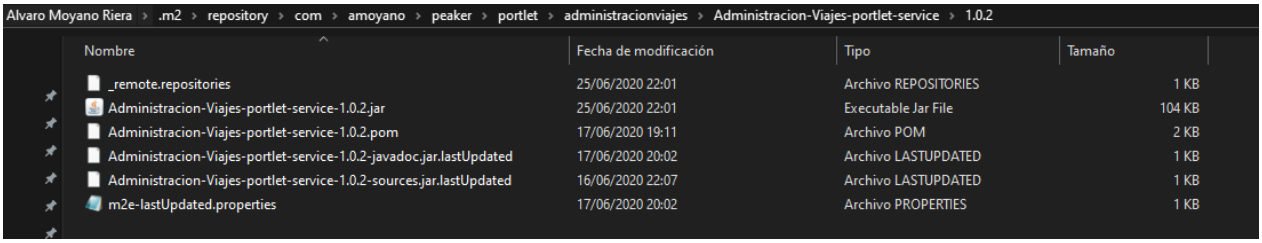


Figura 62 Archivos generados en el repositorio .m2

Los ciclos de vida de Maven son los siguientes:

- *compile*: compila las clases. Java para generar los archivos .class
- *test*: ejecutando pruebas unitarias sobre el código fuente.
- *package:* empaqueta el código compilado para generar un fichero .war o .jar
- *install:* copia el fichero .jar y lo guarda en nuestro repositorio local para hacerlo accesible al resto de proyectos.
- *deploy:* despliega el código en un entorno.

GIT

La herramienta software que se ha utilizado para el control de versiones de los diferentes portlets que se han desarrollado es GIT que es el software más común en el ámbito laboral frente a sistemas de control de versiones centralizado como *SVN*. A pesar de que todo el código ha sido desarrollado por una sola persona, ha facilitado la gestión eficiente de los componentes desarrollados además de mantener el control sobre los cambios realizados y deshacerlos fácilmente. De manera anecdótica quiero comentar que el desarrollo de los diferentes portlets se ha llevado a cabo en tres equipos diferentes: un MacbookPro, un portátil sony VAIO y el portátil DELL que utilizo en mi trabajo.

Cada portlet dispone en el repositorio de dos ramas: *master* y *develop*. En *develop* se lleva a cabo las funcionalidades nuevas que se están desarrollando, cuando se ha verificado la calidad del código desarrollado a través de pruebas, la nueva funcionalidad se lleva a la rama *master* para que solo contenga el código definitivo. De hecho, el desarrollo de los portlets se ha llevado a cabo en 2 portátiles diferentes y un ordenador de sobre mesa.

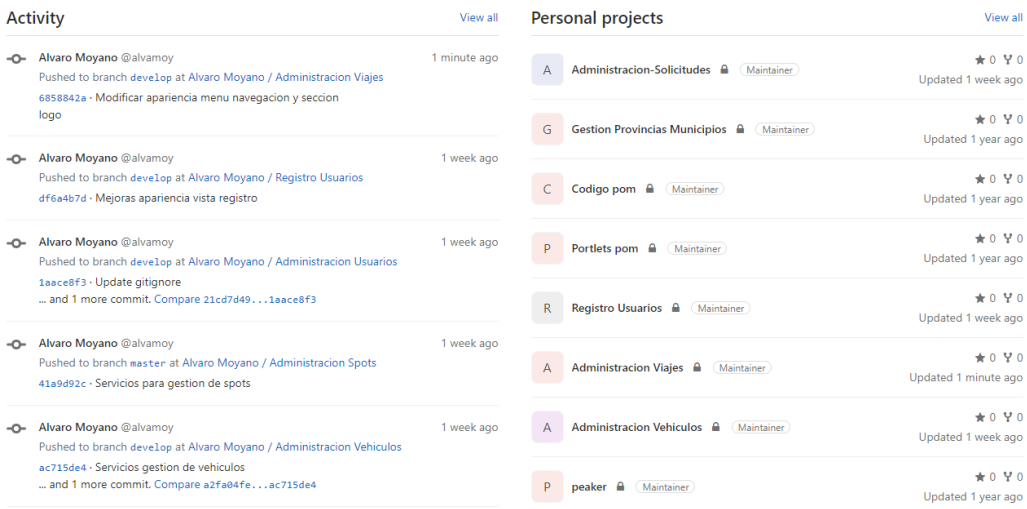


Figura 63 Repositorio GIT de la aplicación

7. Planificación y Presupuesto

7.1 Diagrama de Gantt

La planificación de la elaboración de este proyecto se ha estructurado siguiendo los procesos definidos en la metodología Métrica Versión 3. Estos procesos se dividen en actividades y a su vez en tareas, son estas últimas las que se representan en el siguiente Diagrama de Gantt y donde se muestra el tiempo de dedicación y las dependencias entre ellas. A continuación, se destacan los aspectos relevantes de la planificación del proyecto:

- Personas implicadas: un único programador.
- Días de trabajo semanales: lunes a viernes.
- Duración de cada sesión: 5 horas.
- Fecha de inicio del proyecto: 3 de febrero de 2020.
- Fecha de fin del proyecto: 4 de julio de 2020.
- Días totales: 153.
- Horas totales: 765.

Nombre	Fecha de inicio	Fecha de fin	Duración
Estudio de Viabilidad del Sistema	3/02/20	7/02/20	5
Analisis del Sistema de Informacion	8/02/20	17/02/20	10
Diseño del Sistema de Informacion	18/02/20	29/02/20	12
Plan de pruebas	8/02/20	11/02/20	4
Construccion del Sistema de Informacion	6/03/20	21/06/20	108
Creacion Entorno de Trabajo	6/03/20	7/03/20	2
Modulo de Registro	8/03/20	12/03/20	5
Modulo Gestion de Usuarios	13/03/20	9/04/20	28
Modulo Gestion Vehiculos	27/03/20	7/04/20	12
Modulo Gestion Provincias	20/03/20	23/03/20	4
Modulo Gestion Viajes	10/04/20	8/06/20	60
Modulo Gestion Spots	30/04/20	9/05/20	10
Modulo Gestion Solicitudes	9/06/20	21/06/20	13
Implantacion y Aceptacion del Sistema	22/06/20	26/06/20	5
Manual de Usuario	27/06/20	27/06/20	1
Memoria	28/06/20	4/07/20	7
TOTAL	3/02/20	4/07/20	153

Figura 64 Planificación temporal de las tareas

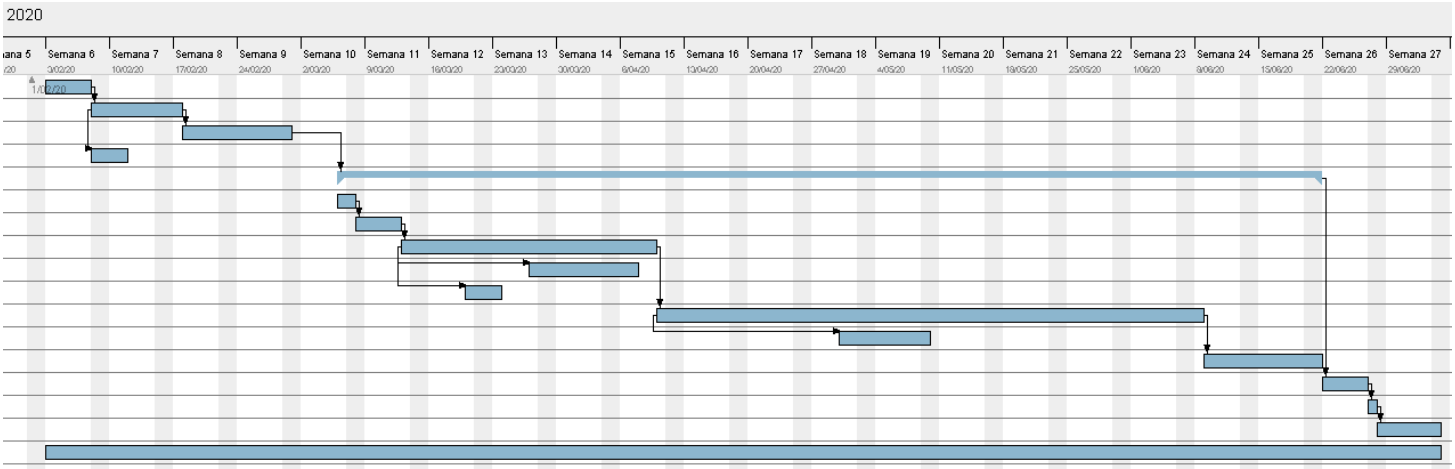


Figura 65 Diagrama de Gantt

7.2 Presupuesto

A continuación, se cuantificará el coste económico que supondría llevar este proyecto a cabo dividiendo los gastos en recursos hardware, recursos software y contratación del personal.

El desarrollo del proyecto lo llevaría a cabo un programador senior. Según el convenio colectivo estatal para empresas de consultoría publicado en el documento *BOE-A-2009-5688* del Boletín Oficial del Estado el salario base para este puesto es de 15.442,56 € al año, teniendo en cuenta las diferentes bonificaciones que tenga el trabajador la media salarial es de 26.208 € lo que se traduciría en 13,23 € / h.

En cuanto a los recursos software, se ha priorizado el uso de herramientas de código abierto cuya licencia es gratuita y cuentan con comunidades muy activas donde se comparte conocimiento y soluciones a problemas habituales.

Respecto a los recursos hardware, será necesaria la compra de un ordenador de sobremesa para llevar a cabo el desarrollo de la aplicación web cuyo coste será de 1200 €. La vida útil para un producto de este tipo es de 9 años lo que supondría una amortización del 11,11% anual. El resto del coste hardware implica el coste de contratación de un servidor dedicado, se ha simulado la contratación de uno en la plataforma *OVHCloud* y el resultado es de 52,24€ mensuales.

7.2.1 Recursos Hardware

Descripción recurso	Ud. medida	Cantidad	Precio / ud	Precio total
Ordenador de sobremesa	Unidad	1	1200 €/ud	1200,00 €
Conexión de internet	Mes	6	35 €/mes	210,00 €
Windows Server 2012	Mes	12	52,24 €/mes	626,88 €

7.2.2 Recursos Software

Descripción recurso	Ud. medida	Cantidad	Precio / ud	Precio total
SO Windows 10	Licencia	1	145 €/licencia	145 €
Liferay Portal (CE) 6.2	Licencia	1	0 €/licencia	0 €
Apache Maven	Licencia	1	0 €/licencia	0 €
Apache Tomcat 6.0.53	Licencia	1	0 €/licencia	0 €
Eclipse IDE	Licencia	1	0 €/licencia	0 €
Spring Framework 4.2.4	Licencia	1	0 €/licencia	0 €
MySQL 5.7	Licencia	1	0 €/licencia	0 €
Certificado SSL	Mes	12	14 €/mes	168 €

7.2.3 Recursos humanos

Tarea	Días	Horas Totales	Precio total
Evaluación de Sistema de Información	5	25	330,75 €
Análisis del sistema de Información	10	50	661,50 €
Diseño del Sistema de Información	12	60	793,80 €
Plan de Pruebas	4	20	264,60 €
Construcción del Sistema de Información	108	540	7.144,20 €
Implantación y Aceptación del sistema	5	25	330,75 €
Manual de usuario	1	5	66,15 €
Memoria	7	35	463,05 €

7.2.4 Presupuesto total

Tipo de recurso		Precio
Recursos Hardware		2036,88 €
Recursos Software		313,00 €
Recursos Humanos		10.054,80 €
	Subtotal:	12.404,68 €
	Gastos generales:	13 %
	Beneficio industrial:	6 %
	Total, sin IVA:	14.761,57
	IVA:	21 %
	Total, con IVA:	17.861,50 €

8. Desarrollo del Plan de Pruebas

1. Diseño de pruebas

A continuación, se definen los requisitos de prueba que servirán como guía para la ejecución de las pruebas de integración. Se establecerán unos Objetivos de Prueba (*OP*) basados en los casos de uso del documento de análisis del sistema y para cada uno de estos objetivos se definirán la entrada y la salida esperada para los distintos requisitos de prueba (*RP*).

Identificador	Entrada	Salida esperada
OP1 Usuario se registra correctamente		
RP1.1	Alguno de los campos está vacío.	Error al registrarse.
RP1.2	Email ya está registrado en el sistema.	Error al registrarse.
RP1.3	Contraseña no cumple con requisitos.	Error al registrarse.
RP1.4	El formato del email no es correcto.	Error al registrarse.
RP1.5	Todos los campos están completos y el formato de email y contraseña son correctos.	Se registra correctamente en el sistema.
OP2 El usuario inicia sesión en el sistema		
RP2.1	Alguno de los campos está vacío.	Error al iniciar sesión.
RP2.2	Email no registrado en el sistema.	Error al iniciar sesión.
RP2.3	El email y la contraseña no coinciden.	Error al iniciar sesión.
RP2.4	Email y contraseña correctos.	Acceso a parte privada de la aplicación.
OP3 Editar información del perfil		
RP3.1	Alguno de los campos requeridos está vacío.	Perfil no se actualiza.
RP3.2	El tipo de imagen no es correcto.	Perfil no se actualiza.
RP3.3	La contraseña no cumple con los requisitos.	Perfil no se actualiza.
RP3.4	Se cambia el valor de la opción de contacto con otros usuarios.	Se actualiza el modo de contacto con otros usuarios.
RP3.5	Se actualiza la imagen de perfil y se pulsa el botón actualizar.	Se actualiza la imagen de perfil del usuario.
RP3.6	Todos los campos son completados, el formato es correcto y se pulsa en el botón actualizar.	Se actualiza la información del perfil.
OP4 Añadir/Editar vehículo		
RP4.1	Alguno de los campos requeridos está vacío.	Error al guardar vehículo.
RP4.2	Formato de imagen no es correcto.	Error al guardar vehículo.
RP4.3	Todos los campos se informan y el formato de imagen es el correcto.	Vehículo se guarda correctamente.
RP4.4	Se completan los campos de información del viaje, pero no se adjunta imagen.	Vehículo se guarda correctamente.
OP5 Crear viaje		
RP5.1	Se pulsa en el botón Spots para seleccionar uno de ellos como destino.	Se visualizan correctamente los spots cargados en el sistema.
RP5.2	Se selecciona uno de los Spots del mapa.	Se visualiza como seleccionado y se ocultan el resto de Spots.
RP5.3	Se pulsa el botón Crear sin marcar un Origen en el mapa.	Se muestra un aviso y el viaje no se crea.

RP5.4	Se pulsa el botón Crear sin marcar un Destino en el mapa.	Se muestra un aviso y el viaje no se crea.
RP5.5	Se pulsa el botón Crear sin marcar un Punto de encuentro en el mapa.	Se muestra un aviso y el viaje no se crea.
RP5.6	Se pulsa sobre uno de los iconos de tipo de destino.	Se visualizan correctamente los deportes del tipo seleccionado.

Identificador	Entrada	Salida esperada
OP5 Crear viaje		
RP5.7	Se envía el formulario sin elegir uno de los vehículos registrados.	Se muestra un aviso y el viaje no se crea.
RP5.8	Se envía el formulario sin completar uno de los campos requeridos.	Se muestra un aviso de los campos necesarios y el viaje no se crea.
RP5.9	Se intenta crear un viaje sin tener un vehículo registrado.	Se muestra un aviso y no se permite crear el viaje.
RP5.10	Se pulsa el botón *Origen* del mapa y se selecciona un punto en el mapa.	Se crea el marcador Origen.
RP5.11	Se pulsa el botón *Lugar de Paso* y se selecciona un punto en el mapa.	Se crea el marcador lugar paso.
RP5.12	Se seleccionan los marcadores de origen y destino, se completa la información del viaje y se pulsa en el botón Crear.	Se crea el viaje correctamente y redirige a la página de información del viaje.
RP5.13	Se seleccionan los marcadores de origen, lugar de paso y destino, se completa la información del viaje y se pulsa en el botón Crear.	Se crea el viaje correctamente y redirige a la página de información del viaje.
OP6 Buscar viaje		
RP6.1	Se accede al buscador de viajes desde mi provincia.	Se visualizan automáticamente los marcadores y los tickets.
RP6.1.2	Se pulsa sobre uno de los marcadores del mapa.	Se muestra en un pop up la información sobre el viaje.
RP6.1.3	Se pulsa el botón Más Información en el pop up de uno de los marcadores del mapa.	Se redirige a la página de información del viaje.
RP6.1.4	Se pulsa sobre el botón Ver en Mapa de uno de los tickets de los viajes.	Se eliminan los marcadores existentes y se muestran los marcadores del itinerario del viaje.
RP6.2	Se accede al buscador de viajes por actividad.	Se accede correctamente al buscador de tickets por actividad.
RP6.2.1	En la página de buscador por actividad se pulsa sobre uno de los tipos de destino.	Se visualizan el filtro por actividades.
RP6.2.2	Se selecciona una actividad del filtro.	Se visualizan los marcadores destino de los viajes en el mapa y los tickets.
RP6.2.3	Se pulsa el botón Más Información en el pop up de uno de los marcadores del mapa.	Se redirige a la página de información del viaje.
RP6.2.4	Se pulsa sobre el botón Ver en Mapa de uno de los tickets de los viajes.	Se eliminan los marcadores existentes y se muestran los marcadores del itinerario del viaje.

RP6.3	Se accede al buscador de viajes por fecha	Se accede correctamente al buscador de tickets por fecha.
RP6.3.1	En la página de buscador por fecha se selecciona un rango de fechas donde no hay viajes	Se muestra un aviso de que no hay viajes entre las fechas seleccionadas.
RP6.3.2	En la página de buscador por fecha se selecciona un rango de fechas donde hay viajes.	Se visualizan los marcadores destino de los viajes en el mapa y los tickets.
OP7 Solicitar viaje		
RP 7.1	Se accede a la página de información de uno de los viajes y se pulsa el botón Solicitar.	Se muestra un pop up de confirmación.
RP 7.2	Se pulsa el botón Aceptar del pop up.	Se crea la solicitud y se recibe un email con la información del viaje.
RP 7.3	Se pulsa el botón Rechazar del pop up.	Se vuelve a la página de información del viaje.
OP8 Ver solicitudes		
RP 8.1	Se accede a la página de solicitudes enviadas sin haber enviado ninguna.	Se muestra un aviso informando de que no se han enviado solicitudes.
RP 8.2	Se accede a la página de solicitudes enviadas.	Se visualizan las solicitudes enviadas correctamente.
RP 8.3	Se accede a la página de Solicitudes Recibidas sin tener ningún viaje creado.	Se muestra un aviso.
RP 8.4	Se accede a la página de Solicitudes Recibidas	Se visualizan los tickets de los viajes activos.
RP 8.5	Se pulsa el botón Ver Solicitudes.	Se redirige a la página de solicitudes recibidas del viaje.
OP9 Cancelar viaje		
RP9.1	Se accede a la página de Solicitudes Recibidas.	Se visualizan los tickets de los viajes activos.
RP9.2	Se pulsa el botón Información Viaje.	Se redirige a la información del viaje.
RP 9.2.1	Se pulsa el botón Cancelar y el viaje tiene solicitudes recibidas PENDIENTES o ACEPTADAS.	El viaje cambia a estado CANCELADO y se envía un mensaje a los solicitantes.
RP 9.2.2	Se pulsa el botón Cancelar y el viaje no ha recibido solicitudes.	El viaje cambia a estado CANCELADO.
OP 10 Cancelar solicitud		
RP 10.1	Se intenta cancelar una solicitud resuelta.	No se permite cancelar
RP 10.2	Se pulsa sobre el botón cancelar de una solicitud enviada.	Se muestra un pop up de confirmación.
RP 10.2.1	Se pulsa el botón Aceptar.	La solicitud pasa a estado CANCELADA.
RP 10.2.2	Se pulsa el botón Rechazar.	Se vuelve a la vista de solicitudes recibidas.

Identificador	Entrada	Salida esperada
RP 11 Aceptar Solicitud		
RP 11.1	Se pulsa el botón Aceptar de una solicitud y se pulsa el botón Confirmar del pop up. El viaje tiene más de 1 plaza libre.	Se muestra aviso, se envía un mensaje al solicitante y se resta una plaza disponible. Se visualiza el modo de contacto con el usuario.
RP 11.2	Se pulsa el botón Aceptar de una solicitud y se pulsa el botón Confirmar del pop up. El viaje tiene 1 plaza libre	Se muestra aviso, se envía un mensaje al solicitante y se resta una plaza disponible. El viaje cambia a estado COMPLETO Se visualiza el modo de contacto con el usuario. El resto de las solicitudes cambian a RECHAZADAS y se envía mensaje a los usuarios
RP 11.3	Se pulsa el botón Aceptar de una solicitud y se pulsa el botón Confirmar del pop up. El viaje no tiene plazas libres.	Se muestra aviso de que no se puede aceptar solicitud.
RP 11.4	Se pulsa el botón Aceptar de una solicitud y se pulsa el botón Atrás del pop up.	Se vuelve a la página de solicitudes recibidas
RP 11.5	Se pulsa el botón aceptar solicitud y el viaje está completo.	Se muestra aviso y la solicitud no cambia de estado.
OP 12 Rechazar solicitud		
RP 12.1	Se pulsa el botón Rechazar de una solicitud que está en estado Pendiente y se pulsa el botón Confirmar del pop up.	Se muestra aviso, se envía un mensaje al solicitante.
RP12.2	Se pulsa el botón Rechazar de una solicitud que está en estado Confirmada y se pulsa el botón Confirmar del pop up.	Se muestra aviso, se envía un mensaje al solicitante. Se suma 1 al número de plazas disponibles.
RP12.3	Se acepta una solicitud y se pulsa el botón Atrás del pop up.	Se vuelve a la página de solicitudes recibidas.
OP 13 Recuperar Contraseña		
RP 13.1	Se pulsa el botón recuperar contraseña, en el formulario de recuperación se introduce el email del usuario registrado.	Se envía un email al usuario para poder resetear la contraseña.
RP 13.2	Se pulsa el botón recuperar contraseña, en el formulario de recuperación se introduce el email erróneo.	Se muestra aviso informando que el email no está registrado en el sistema.

2. Pruebas Funcionales

Las pruebas que realizar se dividirán en tres tipos: unitarias, de integración y de aceptación. En las pruebas unitarias se ejecutarán llamadas con diferentes entradas a los métodos de los paquetes *LocalServiceUtil* de cada entidad, *Manager* y *Útil.*

Las pruebas de integración se realizarán de forma manual, es decir se crearán varios usuarios con información real y se simulará flujos de acciones que realizará un usuario real de la aplicación. De esta manera se probarán todos los módulos de la aplicación de manera individual y su interactuación con el resto de los componentes.

Por último, para las pruebas de aceptación que servirán para determinar si el sistema cumple con el funcionamiento esperado y permitirán al usuario valorar si el alcance del proyecto se ha cumplido. Para ello se diseñará un formulario de valoración para diferentes funcionalidades de la aplicación.

2.1 Pruebas Unitarias

Identificador	Entrada	Salida esperada	Resultado
Clase registroManager			
UsuarioDTO save(UsuarioDTO usuario)			
P.U 1.1	save(null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente
P.U 1.2	save(UsuarioDTO)	UsuarioDTO	OK
P.U 1.3	Save(UsuarioDTO) Alguno de los campos sin informar	Exception	KO Se captura la excepción y se muestra un mensaje al usuario
UsuarioDTO saveUserLR(UsuarioDTO usuario)			
P.U 1.4	saveUserLR(null)	NullPointerException	KO Se captura la excepción, el usuario liferay no se crea y se muestra un mensaje al usuario
P.U 1.5	saveUserLR(usuarioDTO)	User	OK
P.U 1.6	saveUserLR(usuarioDTO) Alguno de los campos sin informar	Exception	KO Se captura la excepción, el usuario liferay no se crea y se muestra un mensaje al usuario

UsuarioDTO saveNew(UsuarioDTO usuario)			
P.U 1.7	saveNew(null)	NullPointerException	KO Se captura la excepción, el usuario no se crea y se muestra un mensaje al usuario
P.U 1.8	saveNew(UsuarioDTO)	User	OK
P.U 1.9	saveNew(UsuarioDTO) Alguno de los campos sin informar	Exception	KO Se captura la excepción, el usuario no se crea y se muestra un mensaje al usuario
void sendRegistroMail(String to, String subject)			
P.U 1.10	sendRegistroMail (“usuario@gmail.com”,”Registro Peaker”)	Envío de correo electrónico al usuario	OK
P.U 1.11	sendRegistroMail (“prueba@prueba.com”,”Registro Peaker”)	AddressException	KO Se captura la excepción y se muestra un mensaje al usuario
Clase UsuarioLocalServiceUtil			
Usuario findByEmail(String email)			
P.U 2.1	findByEmail(jaimerod@gmail.com)	Usuario	OK
P.U 2.2	findByEmail(null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente
P.U 2.3	findByEmail(prueba@prueba.com) email no registrado en BD	NoSuchUsuarioException	KO La excepción se captura y se muestra un mensaje al usuario
Usuario updateUsuario(Usuario usuario)			
P.U 2.4	updateUsuario (usuario)	Usuario	OK
P.U 2.5	updateUsuario(null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente
Usuario getUsuario(User usuario)			
P.U 2.6	getUsuario (usuario)	Usuario	OK
P.U 2.7	getUsuario (null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente

P.U 2.8	getUsuario (usuario) el usuario no está registrado en la tabla pk_au_usuario	PortalException	KO Se captura la excepción y se trata adecuadamente
Clase AdministracionUsuarioViewController			
void filterMunicipiosProvincia(String provinciaId)			
P.U 3.1	filterMunicipiosProvincia(5)	JSON	OK
P.U 3.2	filterMunicipiosProvincia(-10) provId no corresponde a ninguna provincia	SystemException	KO La excepción se captura y trata adecuadamente
void filterModelosMarca(String marcaId)			
P.U 3.3	filterModelosMarca(1)	JSON	OK
P.U 3.4	filterModelosMarca(0) marcaId es valor numérico, pero no corresponde a ninguna marca	JSON vacío	OK
P.U 3.5	filterModelosMarca(“prueba”) marcaId no corresponde a ninguna marca	NumberFormatException	KO La excepción se captura y se muestra un mensaje al usuario
Clase vehiculosManager			
long save(VehiculoDTO vehículo, UsuarioDTO usuario)			
P.U 4.1	save(vehículo, usuario)	long vehiculoId	OK
P.U 4.2	save(null, usuario)	PortalException	KO Se captura la excepción, el vehículo no se crea y se muestra un mensaje al usuario
P.U 4.3	save(vehículo, null)	NullPointerException	KO Se captura la excepción, el vehículo no se crea y se muestra un mensaje al usuario
long saveNew(VehiculoDTO vehículo, UsuarioDTO usuario)			
P.U 4.4	saveNew(vehículo, usuario)	long vehiculoId	OK
P.U 4.5	saveNew(null, usuario)	PortalException	KO Se captura la excepción, el vehículo no se crea y se muestra un mensaje al usuario
P.U 4.6	saveNew(vehículo, null)	NullPointerException	KO Se captura la excepción, el vehículo no se crea y se muestra un mensaje al usuario

long update(VehiculoDTO vehículo, UsuarioDTO usuario)			
P.U 4.7	update(vehículo, usuario)	long vehiculoId	OK
P.U 4.8	update(null, usuario)	PortalException	KO Se captura la excepción, el vehículo no se actualiza y se muestra un mensaje al usuario
P.U 4.9	update(vehículo, null)	NullPointerException	KO Se captura la excepción, el vehículo no se actualiza y se muestra un mensaje al usuario
String getMarcaLiteral(String marcaId)			
P.U 4.10	getMarcaLiteral (1)	String nombreMarca	OK
P.U 4.11	getMarcaLiteral (0) marcaId es valor numérico, pero no corresponde a ninguna marca	" "	OK
P.U 4.12	filterModelosMarca("prueba") marcaId no corresponde a ninguna marca	NumberFormatException	KO Se captura la excepción y se trata adecuadamente
Clase MunicipioLocalServiceUtil			
Municipio getMunicipio(long municipioId)			
P.U 5.1	getMunicipio(1)	Municipio	OK
P.U 5.2	getMunicipio(0) municipioId no está en BD	PortalException	KO Se captura la excepción y se trata adecuadamente
List<Municipios> getMunicipiosByProvinciaId(long provinciaId)			
P.U 5.3	getMunicipiosByProvinciaId(1)	List<Municipios>	OK
P.U 5.4	getMunicipiosByProvinciaId(0) municipioId no está en BD	PortalException	KO Se captura la excepción y se trata adecuadamente
Clase ProvinciaLocalServiceUtil			
Provincia getProvincia(long provinciaId)			
P.U 6.1	getProvincia(1)	Provincia	OK
P.U 6.2	getProvincia(0) municipioId no está en BD	PortalException	KO La excepción se captura y trata adecuadamente
List<Provincia> getProvincias(int start, int end)			
P.U 6.3	getProvincias(0, lastId)	List<Provincia> Listado de todas las provincias	OK

P.U 6.4	getProvincias(0,10)	List<Provincia> Listado de las 10 primeras provincias	OK
P.U 6.5	getProvincias(0, null) Alguno de los valores no es del tipo correcto	SystemException	KO Se captura la excepción y se trata adecuadamente

Identificador	Entrada	Salida esperada	Resultado
Clase VehiculoLocalServiceUtil			
Vehiculo getVehiculo(long vehiculoId)			
P.U 7.1	getVehiculo (1)	Vehiculo	OK
P.U 7.2	getVehiculo(1200) vehiculoId no está registrado en BD	PortalException	KO Se captura la excepción y se trata adecuadamente
List<Vehiculo> findByUsuarioId(long usuarioId)			
P.U 7.3	findByUsuarioId (1)	List<Vehiculo>	OK
P.U 7.4	findByUsuarioId (-1)	SystemException	KO Se captura la excepción y se trata adecuadamente
Vehiculo addVehiculo(Vehiculo vehiculo)			
P.U 7.5	addVehiculo (vehiculo)	Vehiculo	OK
P.U 7.6	addVehiculo (null)	SystemException	KO Se captura la excepción y se trata adecuadamente
Clase actividadLocalServiceUtil			
actividad getActividad(long actividadId)			
P.U 8.1	getActividad (jaimerod@gmail.com)	Usuario	OK
P.U 8.2	getActividad (null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente
P.U 8.3	findByEmail(prueba@prueba.com) email no registrado en BD	NoSuchUsuarioException	KO Se captura la excepción y se trata adecuadamente
List<actividad> getActividades(int start, int end)			
P.U 8.4	getActividades (0, lastId)	List<actividad> Listado de todas las actividades	OK
P.U 8.5	getActividades (0,10)	List<actividad> Listado de las 10 primeras actividades	OK
P.U 8.6	getActividades (0, null)	SystemException	KO

	Alguno de los valores no es del tipo correcto		Se captura la excepción y se trata adecuadamente
List<actividad> getActividadesByTipoDestinoId(long tipoDestinoId)			
P.U 8.7	getActividadesByTipoDestinoId (1)	List<actividad> Listado de todas las actividades con mismo tipoDestinoId	OK
P.U 8.8	getActividadesByTipoDestinoId (5) tipoDestinoId no corresponde a ninguna actividad	[ ] Lista vacía	OK
void filterModelosMarca(String marcaId)			
P.U 8.9	filterModelosMarca(1)	JSON	OK
P.U 8.10	filterModelosMarca(0) marcaId es valor numérico, pero no corresponde a ninguna marca	JSON vacío	OK
P.U 8.11	filterModelosMarca("prueba") marcaId no corresponde a ninguna marca	NumberFormatException	KO Se captura la excepción y se trata adecuadamente
Clase AdministracionViajesLocalServiceUtil			
void retrieveTicketsViajesBetweenFechas (String fechaDesde, String fechaHasta)			
P.U 9.1	retrieveTicketsViajesBetweenFechas ("01/07/20", "20/07/20")	JSON de listado de tickets entre esas fechas	OK
P.U 9.2	retrieveTicketsViajesBetweenFechas ("01/07/22", "20/07/22") Rango de fechas donde no hay viajes registrados	JSON vacío	OK
P.U 9.3	retrieveTicketsViajesBetweenFechas ("01/10/20", "20/07/20") fecha inicio mas tarde que fecha hasta	JSON vacío	OK
void getMarkersViajesBetweenFechas(String fechaDesde, String fechaHasta)			
P.U 9.4	getMarkersViajesBetweenFechas ("01/07/20", "20/07/20")	JSON de listado de marcadores entre esas fechas	OK
P.U 9.5	getMarkersViajesBetweenFechas ("01/07/22", "20/07/22") Rango de fechas donde no hay viajes registrados	JSON vacío	OK
P.U 9.6	getMarkersViajesBetweenFechas ("01/10/20", "20/07/20") fecha inicio mas tarde que fecha hasta	JSON vacío	OK
void filterModelosMarca(String vehículoId)			
P.U 9.7	filterModelosMarca (1)	JSON con listado de modelos	OK
P.U 9.8	filterModelosMarca (0) vehiculoId no corresponde a ningún vehículo en BD	SystemException	KO La excepción se captura y trata adecuadamente
P.U 9.10	filterModelosMarca (null)	NullPointerException	KO

			La excepción se captura y trata adecuadamente
void getTicketsActividad(int actividadId)			
P.U 9.11	getTicketsActividad (1)	JSON Con listado de tickets	OK
P.U 9.12	getTicketsActividad (-1) actividadId no valido	JSON vacío	OK
P.U 9.13	getTicketsActividad (2) actividadId valido, pero no hay ningún ticket asociado a ese actividadId	JSON vacío	OK
void getMarkersActividad(long municipioId)			
P.U 9.14	getMarkersActividad (1)	JSON Con listado de marcadores	OK
P.U 9.15	getMarkersActividad (-1) actividadId no valido	JSON vacío	OK
	actividadId valido, pero no hay ningún ticket asociado a ese actividadId	JSON vacío	OK
List<Spots> getSpotsProvincia(long provinciaId)			
P.U 9.16	getSpotsProvincia (1)	JSON Con listado Spots	OK
P.U 9.17	getSpotsProvincia (-1) municipioId no está en BD	JSON Vacío	OK
Provincia getCoordinatesViaje(String viajeId)			
P.U 9.18	getCoordinatesViaje (5)	JSON Con listado de coordenadas del viaje	OK
P.U 9.19	getCoordinatesViaje (-1) municipioId no está en BD	JSON vacío	OK
List<Provincia> filterMunicipioProvincia(String provinciaId)			
P.U 9.20	filterMunicipioProvincia (1)	JSON Con listado de municipios de la provincia	OK
P.U 9.21	filterMunicipioProvincia (0) provinciaId no corresponde con ninguna provincia en BD	JSON vacío	OK
void filterActividades(String tipoDestinoId)			
P.U 9.22	filterMunicipioProvincia (1)	JSON Con listado de municipios de la provincia	OK
P.U 9.23	filterMunicipioProvincia (0) provinciaId no corresponde con ninguna provincia en BD	JSON vacío	OK
void getInfoViajeTicket(String viajeId)			
P.U 9.24	getInfoViajeTicket(3)	JSON Con información del viaje	OK
P.U 9.25	getInfoViajeTicket(0)	JSON vacío	OK
void sendSolicitud(String viajeId)			

P.U 9.26	sendSolicitud(5)	Mensaje sendSolicitudSuccess	OK
P.U 9.27	sendSolicitud(5) pero se produce algún error al enviar la solicitud	Mensaje sendSolicitudError	OK
List<TicketDTO> getViajesRelacionados(Viaje viaje)			
P.U 9.28	getViajesRelacionados(viaje)	List<TicketDTO> Con viajes relacionados	OK
P.U 9.29	getViajesRelacionados(viaje) pero no tiene ningún viaje relacionado	[ ] Listado vacío	OK
Clase ViajesManager			
long save(ViajeDTO viaje, Usuario usuario)			
P.U 10.1	save(viaje, usuario)	long viajeId	OK
P.U 10.2	save(null, usuario)	PortalException	KO Se captura la excepción, el viaje no se crea y se muestra un mensaje al usuario
P.U 10.3	save(viaje, null)	NullPointerException	KO Se captura la excepción, el viaje no se crea y se muestra un mensaje al usuario
long saveNew(ViajeDTO viaje, Usuario usuario)			
P.U 10.4	saveNew(viaje, usuario)	long viajeId	OK
P.U 10.5	saveNew(null, usuario)	NullPointerException	KO Se captura la excepción, el viaje no se crea y se muestra un mensaje al usuario
P.U 10.6	saveNew(viaje, null)	NullPointerException	KO Se captura la excepción, el viaje no se crea y se muestra un mensaje al usuario
long update(ViajeDTO viaje)			
P.U 10.7	update(viaje)	long viajeId	OK
P.U 10.8	update(null, usuario)	NullPointerException	KO Se captura la excepción, el viaje no se actualiza y se muestra un mensaje al usuario
boolean delete(long viajeId)			

P.U 10.9	delete(3)	true	OK
P.U 10.10	delete(0)	False	OK
Clase SolicitudesManager			
boolean saveNew(Viaje viaje, Usuario usuario)			
P.U 11.1	saveNew(viaje, usuario)	true	OK
P.U 11.2	saveNew(null, usuario)	false	OK
P.U 11.3	saveNew(viaje, null)	false	OK
Clase OpenWeatherManager			
WeatherInSpot getWeatherByLatLong(String latitud, String longitud)			
P.U 12.1	getWeatherByLatLong("43.54", "-5.63")	JSON Con objeto WeatherInSpot	OK
P.U 12.2	getWeatherByLatLong("0","0")	JSON vacío	OK

Identificador	Entrada	Salida esperada	Resultado
Clase DataUtil			
String weatherInES(String descripcion)			
P.U 13.1	weatherInEs("clouds")	"Nuboso"	OK
P.U 13.2	weatherInEs ("prueba")	" "	OK
String monthInES(int mes)			
P.U 13.3	monthInES(2)	"Febrero"	OK
P.U 13.4	monthInES(20)	" "	OK
Clase FichaViajesUtil			
String getColorActividad(String email)			
P.U 14.1	getColorActividad ("surf")	"#A9B2FF"	OK
P.U 14.2	getColorActividad("prueba")	"#FED2EA"	OK
String getImgActividad(String actividad)			
P.U 14.3	getImgActividad ("surf")	"/documents/20182/25606/tabla-de-surf.png"	OK
P.U 14.4	getImgActividad ("prueba")	" "	OK
String getCardWeatherImage(String tiempo)			
P.U 14.5	getCardWeatherImage("rain")	String imagenLluvioso	OK
P.U 14.6	getCardWeatherImage("prueba")	String imagenDespejado	OK
Clase ViajesLocalServiceUtil			
Viaje getViaje(Long viajeId)			
P.U 15.1	getViaje(5)	Viaje	OK
P.U 15.2	getViaje(-10) viaje no corresponde a ningún viaje	PortalException	KO La excepción se captura y se muestra un mensaje al usuario
List<Viajes> getViajesByProvinciaOrigen (long provinciaId)			
P.U 15.3	getViajesByProvinciaOrigen (1)	List<Viajes>	OK
P.U.15.4	getViajesByProvinciaOrigen (-1) provinciaId no corresponde a ninguna provincia	PortalException	KO Se captura la excepción y se

			trata adecuadamente
List<Viajes> getViajesByActividadYProvinciaOrigen (long actividad, long provinciaId)			
P.U 15.5	getViajesByActividadYProvinciaOrigen (1,1)	List<Viajes>	OK
P.U 15.6	getViajesByActividadYProvinciaOrigen (null, 1)	PortalException	KO Se captura la excepción y se trata adecuadamente
P.U 15.7	getViajesByActividadYProvinciaOrigen (1, null)	PortalException	KO Se captura la excepción y se trata adecuadamente
List<Viaje> getViajesDisponiblesBetweenFechas(String fechaDesde, String fechaHasta)			
P.U 15.8	getViajesDisponiblesBeetweenFechas (“01/07/20”, “20/07/20”,1)	List<Viaje>	OK
P.U 15.9	getViajesDisponiblesBetweenFechas (“01/10/20”, “20/07/20”,1) fechaDesde posterior a fechaHasta	[ ]	OK
P.U 15.9	getViajesDisponiblesBeetweenFechas (“01/07/20”, “20/07/20”,-1) provinciaId no valido	PortalException	KO Se captura la excepción y se trata adecuadamente
List<Viajes> getViajesBySpotId(long spotId)			
P.U 15.10	getViajesBySpotId (1)	List<Viajes>	OK
P.U 15.11	getViajesBySpotId (-1)	PortalException	KO La excepción se captura y trata adecuadamente
Clase SpotLocalServiceUtil			
Spot getSpot(long Id)			
P.U 16.1	getSpot(1)	Spot	OK
P.U 16.2	getSpot(0) spot no está en BD	PortalException	KO Se captura la excepción y se trata adecuadamente

Identificador	Entrada	Salida esperada	Resultado
Clase AdminsitracionSolicitudesController			
void aceptarSolicitud(String solicitudId)			
P.U 17.1	aceptarSolicitud(1)	JSON	OK
P.U 17.2	aceptarSolicitud (-1)	Exception	KO Se captura la excepción y se muestra un mensaje al usuario
void rechazarSolicitud(String solicitudId)			
P.U 17.4	rechazarSolicitud(1)	JSON	OK
P.U 17.5	rechazarSolicitud (-1)	Exception	KO

			Se captura la excepción y se muestra un mensaje al usuario
void cancelarSolicitud(String solicitudId)			
P.U 17.6	rechazarSolicitud(1)	JSON	OK
P.U 17.7	rechazarSolicitud (-1)	Exception	KO Se captura la excepción y se trata adecuadamente
Clase SolicitudesManager			
List<SolicitudDTO> convertToSolicitudDTO(List<Solicitud> listSolicitudes)			
P.U 18.1	convertToSolicitudDTO (listSolicitudes)	List<SolicitudDTO>	OK
P.U 18.2	convertToSolicitudDTO (null)	NullPointerException	KO Se captura la excepción y se trata adecuadamente
Int getSolicitudesPendientesViaje (long viajeId)			
P.U 18.4	getSolicitudesPendientesViaje (1)	numSolicitudesPendientes	OK
P.U.18.5	getSolicitudesPendientesViaje (-1)	0	OK
void sendInfoMail (String to, String subject, Solicitud solicitud)			
P.U 18.6	sendInfoMail (“usuario@gmail.com”,”Solicitud aceptada”, Solicitud)	Envío de correo electrónico al usuario	OK
P.U 18.7	sendInfoMail (“prueba@prueba.com”,”Solicitud aceptada”, Solicitud)	AddressException	KO Se captura la excepción y se muestra un mensaje al usuario

2.2 Pruebas de Integración

Para llevar a cabo este tipo de pruebas simularemos las acciones reales de tres usuarios de prueba, de esta manera se comprobará la interacción de todos los módulos del sistema. Los casos de prueba (C.P) se basarán en los casos de uso definidos en el documento de análisis. Para cada uno de ellos se establecerán los requisitos de prueba (R.P) cubiertos, las precondiciones, los diferentes caminos de prueba y por último la salida esperada y la salida final. A continuación, se detalla la información de los tres usuarios de prueba

Usuario 1

Email: alvamoy94@gmail.com

Nota: El usuario no está registrado en el sistema.

Usuario 2

Información de perfil:

- Nombre: Jaime
- Apellidos: Rodríguez
- Email: jaimerodprueba@gmail.com
- Teléfono: 618847971
- Forma de contacto: móvil
- Provincia: Asturias
- Municipio: Gijón
- Contraseña: jaimerodprueba

Información de vehículo:

- Marca: VOLVO
- Modelo: V70 XC
- Imagen: VolvoV70.jpg
- Color: Gris
- Tipo combustible: Gasolina
- Número de plazas: 4
- Cofre: No
- Baca: Si
- Portaesquis: No
- Información extra:

Información de viajes creados:

Viaje 1

- Identificador W842.
- Origen: (43.5396, -5.6272)
- Destino: (43.1871, -4.8724)
- Tipo destino: Montaña.
- Actividad: Travesía.
- Descripción de la actividad: Hacer la ruta de Peña Ubiña, llevar comida por que iremos por la mañana y volveremos de tarde.
- Provincia: Asturias
- Municipio: Gijón
- Punto de encuentro: enfrente del banco Santander de la plazuela San Miguel.
- Fecha Viaje: 27-06-20
- Hora salida: 12:00
- Hora llegada: 20:00
- Número de plazas: 4
- Comentarios: Llevar el equipamiento de montaña necesario, se aceptan perros
- Vehículo: Volvo V70 Xc.
- IsSpot: true.

Viaje 2

- Identificador: 1B23.
- Origen: (43.5352, -5.6386)
- Destino: (43.4524, -5.5292)
- Tipo destino: Costa.
- Actividad: Surf.
- Descripcion de la actividad: Ir a Playa España a coger olas antes de comer y tomar algo en el Ipanema y volver a Gijón
- Provincia: Asturias
- Municipio: Gijón
- Punto de encuentro: Aparcamiento de El Molinón en frente del Alimerka.
- Fecha Viaje: 23-06-20
- Hora salida: 11:00
- Hora llegada: 16:00
- Número de plazas: 4
- Comentarios: Llevar algo para no manchar el coche con el neopreno,
- Vehículo: Volvo V70 Xc.
- IsSpot: true.

Email: pabloperezprueba@gmail.com

Contraseña: pabloperezprueba

Información de perfil:

- Nombre: Pablo
- Apellidos: Perez
- Email: pabloperezprueba@gmail.com
- Teléfono: 672814532
- Forma de contacto: Email
- Provincia: Asturias
- Municipio: Gijón
- Contraseña: pabloperezprueba

Información de vehículo:

- Marca: Mercedes-Benz
- Modelo: Viano.
- Imagen: mercedes_viano.jpg
- Color: Blanco
- Tipo combustible: Diesel
- Número de plazas: 5
- Cofre: No
- Baca: No
- Porta esquís: No
- Información extra: Amplio espacio en el maletero para llevar equipamiento

Información de viajes creados:

Viaje 3

- Identificador R32P.
- Origen: (43.5366, -5.6603)
- Destino: (43.5100, -5.5988)
- Tipo destino: Montaña.
- Actividad: Descenso.
- Descripción de la actividad: Aparcar el coche en el aparcamiento de la parte de arriba del monte y hacer varias bajadas al descenso del Monte Deva
- Provincia: Asturias
- Municipio: Gijón
- Punto de encuentro: Parada de autobuses de enfrente del Alimerka de Begoña.
- Fecha Viaje: 30-06-20
- Hora salida: 11:00
- Hora llegada: 15:00
- Número de plazas: 5

- Comentarios: Se permiten perros.
- Vehículo: Mercedes-Benz Viano
- IsSpot: true.

Viaje 4

- Identificador: KAM9.
- Origen: (43.5353, -5.6603)
- Destino: (43.18, -5.6380)
- Tipo destino: Costa.
- Actividad: Surf.
- Descripción de la actividad: Dan buenas olas por la mañana en la playa de salinas, el plan sería ir por la mañana pronto y volver antes de comer.
- Provincia: Asturias
- Municipio: Gijón
- Punto de encuentro: Aparcamiento de El Molinón, enfrente del Alimerka.
- Fecha Viaje: 01-07-20
- Hora salida: 11:00
- Hora llegada: 14:00
- Número de plazas: 5
- Vehiculo: Mercedes-Benz Viano.
- IsSpot: true.

C.P Registro en la aplicación	
R.P Cubiertos	1.1, .12, 1.3, 1.4, 1.5
Precondición	Usuario con correo alvamoy94@gmail.com no registrado en el sistema Usuario en pantalla Inicio
Caminos de prueba	1. Se introducen los siguientes valores • Nombre: Alvaro • Apellidos Moyano • Email: alvamoy94@gmail.com • Contraseña: alvamoyPeaker2020 2. Se introducen los siguientes valores • Nombre: alvaro • Apellidos: moyano • Email: alvamoy94prueba.com • Contraseña: alvamoyPeaker2020 3. Se introducen los siguientes valores • Nombre: alvaro • Apellidos: moyano • Email: alvamoy94@gmail.com • Contraseña : alvamoy#Peaker#2020
Salida esperada	1. Se registra al usuario correctamente 2. No se registra al usuario. Se muestra mensaje de correo inválido 3. No se registra al usuario. Se muestra mensaje de contraseña inválida
Salida final	Se obtienen los resultados esperados.

C.P Recuperar contraseña	
R.P Cubiertos	RP 13.1, RP 13.2
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com Usuario en pantalla Inicio
Caminos de prueba	1. Se pulsa sobre el botón recuperar contraseña, se muestra el formulario de recuperación. Se introducen los siguientes valores • Email: alvamoy94@gmail.com 2. Se pulsa sobre el botón recuperar contraseña, se muestra el formulario de recuperación • Email: alvamoy94prueba.com
Salida esperada	1. Se envía el email de recuperación de contraseña 2. El email no se envía y se muestra mensaje de email no registrado en el sistema
Salida final	Se obtienen los resultados esperados.

C.P Login	
R.P Cubiertos	2.1, 2.2, 2.3, 2.4
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com Usuario en pantalla Inicio
Caminos de prueba	1. Se pulsa sobre el botón recuperar contraseña, se muestra el formulario de recuperación. Se introducen los siguientes valores • Email: alvamoy94@gmail.com. • Contraseña: alvamoyPeaker2020 2. Se pulsa sobre el botón recuperar contraseña, se muestra el formulario de recuperación • Email: alvamoy94prueba.com. • Contraseña: alvamoyPeaker2020 3. Se pulsa sobre el botón recuperar contraseña, se muestra el formulario de recuperación • Email: alvamoy94@gmail.com. • Contraseña: prueba.
Salida esperada	1. Se accede correctamente a la aplicación. 2. Se muestra aviso de que el email no es correcto. 3. Se muestra aviso de que la contraseña es incorrecta.
Salida final	Se obtienen los resultados esperados.

C.P Editar Perfil	
R.P Cubiertos	3.1, 3.2, 3.3, 3.4, 3.5, 3.6
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com Usuario en pantalla Mi perfil. Ya se encuentra completada la información que indicó en el registro.
Caminos de prueba	1. Se introducen los siguientes valores • Provincia: Asturias. • Municipio: Gijón. • Telefono: 691764953. 2. Se pulsa sobre el botón actualizar sin completar ningún campo. 3. Se introducen los siguientes valores • Provincia: Asturias. • Municipio: Gijón. 4. Se actualiza la contraseña con el siguiente valor: alvaro#Peaker#2020 5. Se pulsa en el botón Cambiar imagen y se adjunta una nueva imagen .png
Salida esperada	1. Se actualiza la información del usuario correctamente. 2. Se muestra aviso de que es necesario completar los campos Provincia y Municipio. 3. Se actualiza la información del usuario correctamente. 4. Se muestra aviso de formato de contraseña incorrecta. 5. Se actualiza la imagen del usuario correctamente.
Salida final	Se obtienen los resultados esperados para todos los caminos de prueba

C.P Crear Vehiculo	
R.P Cubiertos	4.1, 4.2, 4.3, 4.4
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com . Usuario en pantalla Mis vehículos. Ya se encuentra completada la información que indicó en el registro.
Caminos de prueba	1. Se pulsa el botón añadir vehiculo. Se redigire al usuario a la página Añadir vehiculo. Se introducen los siguientes valores • Marca: BMW. Al seleccionar una marca se cargan los diferentes modelos en el campo Modelo. • Modelo: Serie 3. • Plazas: 3. • Tipo de combustible: diesel. • Color: Blanco. Se pulsa sobre el botón Seleccionar archivo y se adjunta una imagen .png 2. Se pulsa el botón añadir vehiculo. Se redigire al usuario a la página Añadir vehiculo. Se introducen los siguientes valores • Marca: BMW. Al seleccionar una marca se cargan los diferentes modelos en el campo Modelo. • Plazas: 3. • Tipo de combustible: diesel. • Color: Blanco. 3. Se pulsa el botón añadir vehiculo. Se redigire al usuario a la página Añadir vehiculo. Se introducen los siguientes valores • Marca: BMW. Al seleccionar una marca se cargan los diferentes modelos en el campo Modelo. • Modelo: Serie 3. • Plazas: 3.
Salida esperada	1. Se crea el vehículo correctamente. 2. Se crea el vehículo correctamente sin imagen. 3. Se muestra aviso de los campos requeridos.
Salida final	Se obtienen los resultados esperados.

C.P Editar Vehiculo	
R.P Cubiertos	4.1, 4.2, 4.3, 4.4
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com . Usuario en pantalla Mis vehículos. Ya se encuentra completada la información que indicó en el registro.
Caminos de prueba	1. Se pulsa el botón EDITAR. Se redigire al usuario a la página Editar vehiculo. La información del vehiculo ya se encuentra informada en el formulario. Se introducen los siguientes valores • Plazas: 4. Se pulsa sobre el botón Seleccionar archivo y se adjunta una imagen .png. Se pulsa el botón Actualizar. 2. Se pulsa el botón EDITAR. Se redigire al usuario a la página Editar vehiculo. La información del vehiculo ya se encuentra informada en el formulario. Se introducen los siguientes valores • Información extra: maletero amplio. Se pulsa el botón Actualizar. 3. Se pulsa el botón añadir vehiculo. Se redigire al usuario a la página Añadir vehiculo. Se introducen los siguientes valores • Se pulsa sobre el botón Seleccionar archivo y se adjunta una imagen .png .
Salida esperada	1. Se actualiza el número de plazas del vehiculo. 2. Se actualiza el campo Información extra del vehiculo. 3. Se actualiza la imagen del vehiculo correctamente.
Salida final	Se obtienen los resultados esperados.

C.P Solicitar Viaje	
R.P Cubiertos	6.1, 6.1.2, 6.1.3, 6.1.4, 6.2, 6.2.1, 6.2.3, 6.2.4, 6.3, 6.3.1, 6.3.2, 7.1, 7.2, 7.3
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com . El usuario se encuentra en la página dashboard viajes.
Caminos de prueba	1. Se pulsa en el botón Buscar Viajes. Se accede al buscador por municipio origen. El municipio del usuario es Gijón. • Se muestran los siguientes viajes: W842, 1B23, R32P y KAM9 • Se pulsa sobre el botón ver en el mapa del viaje. Se visualizan correctamente los marcadores del itinerario • Se pulsa en el botón Más información del viaje **1B23**. • Se accede a la Ficha del viaje **1B23**con toda su información • Se pulsa el botón solicitar para enviar una solicitud al viaje **1B23**y se confirma la acción en la ventana modal. 2. Se pulsa en el botón Buscar Viajes. Se pulsa el botón viajes por actividad. Se redirige al usuario a la página buscador viajes por actividad. • Se selecciona surf • Se visualizan en el mapa los marcadores destino de los viajes y los tickets de los siguientes viajes: 1B23 y KAM9 . • Se pulsa el botón más información del viaje • Se accede a la Ficha del viaje **KAM9** con toda su información • Se pulsa el botón solicitar para enviar solicitud al viaje **KAM9** y se confirma la acción en la ventana modal. 3. Se selecciona Buscar Viajes. Se pulsa el botón Viajes por fecha • Se introduce el valor 26/06/20 en el campo fecha desde y 29/06/20 en el campo fecha hasta • Se visualizan en el mapa los marcadores y los tickets de los siguientes viajes: W842 y R32P • Se pulsa el botón Más información del viaje **R32P** • Se accede a la Ficha del viaje **R32P** • Se pulsa el botón solicitar y se confirma la acción en la ventana modal
Salida esperada	1. Se crea correctamente la solicitud para el viaje. Se envía al organizador un email informándole de la solicitud . Se envía un email al solicitante con la información del viaje. 2. Se crea correctamente la solicitud para el viaje. Se envía al organizador un email informándole de la solicitud . Se envía un email al solicitante con la información del viaje. 3. Se crea correctamente la solicitud para el viaje. Se envía al organizador un email informándole de la solicitud . Se envía un email al solicitante con la información del viaje.
Salida final	Se obtienen los resultados esperados.

C.P Crear Viaje	
R.P Cubiertos	5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 ,5.8, 5.9, 5.10, 5.11
Precondición	Se ha accedido a la aplicación con el email alvamoy94@gmail.com . El usuario se encuentra en la página dashboard viajes.
Caminos de prueba	1. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con la siguiente información. • Se selecciona origen y se indica un marcador en el mapa • Se selecciona Spots. Se muestran en el mapa los marcadores de los Spots de Asturias. • Se pulsa sobre el marcador Playa España y se selecciona como Spot. • Se selecciona la actividad Surf. • Se completa el campo Descripción de la actividad • Se informan el campo Punto de encuentro • Se informan los campos del horario con la siguiente información: o Fecha viaje: 13/07/20 o Hora salida: 09:00 o Hora llegada: 15:00 • En la sección Vehiculo se visualiza el vehiculo añadido. Se selecciona el vehiculo BMW-Serie3. • Se pulsa el botón Publicar 2. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 y además se selección un marcador Lugar de paso. 3. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 pero no se selecciona Destino ni Spot. 4. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 pero no se selecciona ningún vehículo. 5. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 pero no se selecciona Origen. 6. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 pero no se selecciona Actividad. 7. Se pulsa en la Crear Viaje. Se accede a la página Crear Viaje. Se completa el formulario con de la misma forma que el camino 1 pero no se informan los siguientes ni campos: fecha viaje, hora desde y hora hasta.

Salida esperada	1. Se crea el viaje correctamente con identificador **CG1Z** y se redirige a la Ficha del viaje. 2. Se crea el viaje correctamente con identificador **FU2J** y se redirige a la Ficha del viaje . 3. El viaje no se crea y se muestra un aviso informando que es necesario seleccionar un marcador destino. 4. El viaje no se crea y se muestra un aviso informando que es necesario seleccionar un vehiculo. 5. El viaje no se crea y se muestra un aviso informando que es necesario seleccionar un marcador origen. 6. El viaje no se crea y se muestra un aviso informando que es necesario seleccionar una actividad. 7. El viaje no se crea y se muestra un aviso informando que es necesario informar los siguientes campos: fecha viaje, hora salida y hora llegada.
Salida final	Se obtienen los resultados esperados.

C.P Gestionar solicitudes recibidas	
R.P Cubiertos	8.1, 8.3, 8.4, 8.5, 9.1, 11.1, 11.2, 11.3, 11.4, 11.5, 12.1, 12.2, 12.3,
Precondiciones	El usuario registrado se encuentra en la página principal de Solicitudes.
Caminos de prueba	Se accede al sistema con el usuario jaimerod@gmail.com. .Se procede a aceptar solicitud del usuario alvamoy94@gmail.com. • Se accede a la sección Solicitudes Recibidas • Se visualizan los dos viajes que ha recibido solicitudes • Se pulsa en el botón Ver solicitudes del viaje con destino Playa España para acceder a la página de solicitudes recibidas del viaje con identificador **1B23**. • Se pulsa el botón Aceptar solicitud y se confirma la acción en la ventana modal 1. Se accede al sistema con el usuario pabloperezprueba@gmail.com . Se procede a rechazar solicitud del usuario alvamoy94@gmail.com • Se accede a la sección Solicitudes Recibidas • Se visualizan los dos viajes que ha recibido solicitudes • Se pulsa en el botón Ver solicitudes del viaje con destino Monte para acceder a la página de solicitudes recibidas del viaje con identificador **R32P**. • Se pulsa el botón Rechazar solicitud y se confirma la acción en la ventana modal
Salida esperada	1. Se muestra un mensaje informando de que la solicitud se ha enviado correctamente y en el campo contacto se muestra el telefono del usuario, como tiene indicado en su perfil el usuario jaimerod@gmail.com. Se desactivan todas las opciones excepto la de consultar la información del viaje. El número de plazas disponible se reduce a 3 para el viaje con identificador **1B23**. 2. Se muestra un mensaje informando de que la solicitud se ha rechazado y se desactivan el resto de las opciones excepto la de consultar viaje. Se envía un email a pabloperezprueba@gmail.com Informándole de que la solicitud ha sido rechazada.
Salida final	Se obtienen los resultados esperados en ambos caminos de prueba.

C.P Gestionar enviadas	
R.P Cubiertos	8.2 ,10.2, 10.2.1, 10.2.2
Precondiciones	Se accede al sistema con el usuario alvamoy94@gmail.com. El usuario alvamoy94@gmail.com ha enviado una solicitud al viaje con identificador **1B23**. El usuario jaimerodprueba@gmail.com la ha aceptado el número de plazas disponibles se reduce a 3. El usuario alvamoy94@gmail.com ha recibido un email informándole. El usuario alvamoy94@gmail.com ha enviado una solicitud al viaje con identificador **KAM9.** El usuario registrado se encuentra en la página principal de solicitudes
Caminos de prueba	1. Se accede a la página Solicitudes recibidas para consultar información solicitud aceptada 2. Cancelar solicitud del viaje con identificador • Se accede a la página solicitudes recibidas • Se pulsa el botón cancelar para el viaje
Salida esperada	1. En el campo contacto del viaje se visualiza el email, como ha seleccionado el usuario jaimerodprueba@gmail.comen su perfil 2. La solicitud se cancela correctamente y se envía un mensaje al usuario pabloperezprueba@gmail.com informándole
Salida final	Se obtienen los resultados esperados para ambos caminos de prueba

2.3 Pruebas de aceptación

Para verificar que la aplicación cumple con el funcionamiento esperado, se ha diseñado un cuestionario que será completado tres usuarios con diferentes perfiles. De esta manera serán los usuarios quien valoren la usabilidad y funcionalidades de la aplicación. El perfil de los usuarios es el siguiente:

- Usuario A
    - Edad: 22
    - Deporte favorito: surf.
- Usuario B
    - Edad: 38
    - Deporte favorito: esquí.
- Usuario C
    - Edad: 54
    - Deporte favorito: travesía.

2.3.1 Diseño del cuestionario

Preguntas

¿Ha compartido vehiculo con otras personas con alguna finalidad?¿Qué opinión tiene de ello?

☐ Lo he hecho varias veces y he tenido buenas experiencias.

☐ Lo he hecho ocasionalmente y me parece algo normal hoy en día.

☐ Nunca lo he hecho, pero lo probaría.

☐ Nunca lo he hecho y no lo haría.

¿Realiza actividades deportivas en solitario? Si es así ¿con que frecuencia?

☐ Si, varias veces por semana.

☐ Si, de manera esporádica .

☐ No, solo si es con una o varias personas.

¿Estaría dispuesto a compartir vehículo con otras personas para repartir gastos de transporte?

☐ Si, estaría dispuesto.

☐ No lo haría.

¿Estaría dispuesto a utilizar la Peaker para compartir vehiculo con otras personas y realizar juntos una actividad deportiva?

- ☐ La utilizaría siempre que fuera a realizar alguna actividad deportiva
- ☐ La utilizaría únicamente para reducir los gastos de desplazamiento
- ☐ La utilizaría de manera ocasional.
- ☐ No la utilizaría.

Respecto a la usabilidad de la aplicación, puntue de 1 a 5 las siguientes afirmaciones. Donde 1 es "estoy en desacuerdo" y 5 es "estoy totalmente de acuerdo".

	1	2	3	4	5
El proceso de registro y acceso a la aplicación me ha resultado rápido y sencillo					
La información de Mi Perfil					
La forma de añadir, editar y borrar vehículos me ha parecido intuitiva.					
La información necesaria para la creación de un viaje me ha parecido adecuada.					
Me ha resultado sencillo utilizar los buscadores de viajes y he podido encontrar los viajes fácilmente					
El proceso de envío de solicitudes me ha resultado intuitivo.					
El sistema de gestión de gestión para solicitudes enviadas y recibidas me ha.					

La interfaz de las diferentes pantallas y los avisos del sistema me han facilitado el uso de la aplicación.				
Me ha resultado sencillo navegar en las diferentes pantallas de la aplicación.				
Estoy de acuerdo en que la comunicación entre usuarios una vez se acepte una solicitud sea fuera de la aplicación				
Mi experiencia usando la aplicación es buena.				

2.3.2 Resultados del cuestionario

Usuario A

¿Ha compartido vehiculo con otras personas con alguna finalidad?¿Qué opinión tiene de ello?

- [x] Lo he hecho varias veces y he tenido buenas experiencias.
- [ ] Lo he hecho ocasionalmente y me parece algo normal hoy en día.
- [ ] Nunca lo he hecho, pero lo probaría.
- [ ] Nunca lo he hecho y no lo haría.

¿Realiza actividades deportivas en solitario? Si es así ¿con que frecuencia?

- [ ] Si, varias veces por semana.
- [x] Si, de manera esporádica .
- [ ] No, solo si es con una o varias personas.

¿Estaría dispuesto a compartir vehículo con otras personas para repartir gastos de transporte?

- [x] Si, estaría dispuesto.
- [ ] No lo haría.

¿Estaría dispuesto a utilizar la Peaker para compartir vehiculo con otras personas y realizar juntos una actividad deportiva?

- [x] La utilizaría siempre que fuera a realizar alguna actividad deportiva .
- [ ] La utilizaría únicamente para reducir los gastos de desplazamiento.
- [ ] La utilizaría de manera ocasional.
- [ ] No la utilizaría.

Respecto a la usabilidad de la aplicación, puntue de 1 a 5 las siguientes afirmaciones. Donde 1 es "estoy en desacuerdo" y 5 es "estoy totalmente de acuerdo"

	1	2	3	4	5
El proceso de registro y acceso a la aplicación me ha resultado rápido y sencillo					X
La información de Mi Perfil me parece adecuada				X	
La forma de añadir, editar y borrar vehículos me ha parecido intuitiva.				X	
La información necesaria para la creación de un viaje me ha parecido adecuada.				X	
Me ha resultado sencillo utilizar los buscadores de viajes y he podido encontrar los viajes fácilmente					X
El proceso de envío de solicitudes me ha resultado intuitivo.					X

El sistema de gestión de gestión para solicitudes enviadas y recibidas me ha resultado fácil de utilizar.			**X**		
La interfaz de las diferentes pantallas y los avisos del sistema me han facilitado el uso de la aplicación.				**X**	
Me ha resultado sencillo navegar en las diferentes pantallas de la aplicación.		**X**			
Estoy de acuerdo en que la comunicación entre usuarios una vez se acepte una solicitud sea fuera de la aplicación			**X**		
Mi experiencia usando la aplicación es buena.			**X**		

Usuario B

¿Ha compartido vehiculo con otras personas con alguna finalidad?¿Qué opinión tiene de ello?

- ☐ Lo he hecho varias veces y he tenido buenas experiencias.
- ☒ Lo he hecho ocasionalmente y me parece algo normal hoy en día.
- ☐ Nunca lo he hecho, pero lo probaría.
- ☐ Nunca lo he hecho y no lo haría.

¿Realiza actividades deportivas en solitario? Si es así ¿con que frecuencia?

- [x] Si, varias veces por semana.
- [ ] Si, de manera esporádica .
- [ ] No, solo si es con una o varias personas.

¿Estaría dispuesto a compartir vehículo con otras personas para repartir gastos de transporte?

- [x] Si, estaría dispuesto.
- [ ] No lo haría.

¿Estaría dispuesto a utilizar la Peaker para compartir vehiculo con otras personas y realizar juntos una actividad deportiva?

- [ ] La utilizaría siempre que fuera a realizar alguna actividad deportiva .
- [x] La utilizaría únicamente para reducir los gastos de desplazamiento.
- [ ] La utilizaría de manera ocasional.
- [ ] No la utilizaría.

Respecto a la usabilidad de la aplicación, puntue de 1 a 5 las siguientes afirmaciones. Donde 1 es "estoy en desacuerdo" y 5 es "estoy totalmente de acuerdo"

	1	**2**	**3**	**4**	**5**
El proceso de registro y acceso a la aplicación me ha resultado rápido y sencillo					**X**
La información de Mi Perfil me parece adecuada				**X**	

La forma de añadir, editar y borrar vehículos me ha parecido intuitiva.				**X**
La información necesaria para la creación de un viaje me ha parecido adecuada.				**X**
Me ha resultado sencillo utilizar los buscadores de viajes y he podido encontrar los viajes fácilmente				**X**
El proceso de envío de solicitudes me ha resultado intuitivo.			**X**	
El sistema de gestión de gestión para solicitudes enviadas y recibidas me ha resultado fácil de utilizar.			**X**	
La interfaz de las diferentes pantallas y los avisos del sistema me han facilitado el uso de la aplicación.				**X**
Me ha resultado sencillo navegar en las diferentes pantallas de la aplicación.		**X**		
Estoy de acuerdo en que la comunicación entre usuarios una vez se acepte una solicitud sea fuera de la aplicación			**X**	
Mi experiencia usando la aplicación es buena.			**X**	

Usuario C

¿Ha compartido vehiculo con otras personas con alguna finalidad?¿Qué opinión tiene de ello?

- [ ] Lo he hecho varias veces y he tenido buenas experiencias.
- [ ] Lo he hecho ocasionalmente y me parece algo normal hoy en día.
- [x] Nunca lo he hecho, pero lo probaría.
- [ ] Nunca lo he hecho y no lo haría.

¿Realiza actividades deportivas en solitario? Si es así ¿con que frecuencia?

- [ ] Si, varias veces por semana.
- [ ] Si, de manera esporádica .
- [x] No, solo si es con una o varias personas.

¿Estaría dispuesto a compartir vehículo con otras personas para repartir gastos de transporte?

- [x] Si, estaría dispuesto.
- [ ] No lo haría.

¿Estaría dispuesto a utilizar la Peaker para compartir vehiculo con otras personas y realizar juntos una actividad deportiva?

- [ ] La utilizaría siempre que fuera a realizar alguna actividad deportiva.
- [ ] La utilizaría únicamente para reducir los gastos de desplazamiento.
- [x] La utilizaría de manera ocasional.
- [ ] No la utilizaría.

Respecto a la usabilidad de la aplicación, puntue de 1 a 5 las siguientes afirmaciones. Donde 1 es "estoy en desacuerdo" y 5 es "estoy totalmente de acuerdo"

	1	2	3	4	5
El proceso de registro y acceso a la aplicación me ha resultado rápido y sencillo.				**X**	
La información de Mi Perfil me parece adecuada.			**X**		
La forma de añadir, editar y borrar vehículos me ha parecido intuitiva.			**X**		
La información necesaria para la creación de un viaje me ha parecido adecuada.			**X**		
Me ha resultado sencillo utilizar los buscadores de viajes y he podido encontrar los viajes fácilmente			**X**		
El proceso de envío de solicitudes me ha resultado intuitivo.			**X**		
El sistema de gestión de gestión para solicitudes enviadas y recibidas me ha resultado fácil de utilizar.				**X**	
La interfaz de las diferentes pantallas y los avisos del sistema me han facilitado el uso de la aplicación.					**X**
Me ha resultado sencillo navegar en las diferentes pantallas de la aplicación.		**X**			
Estoy de acuerdo en que la comunicación entre usuarios una vez se acepte una solicitud sea fuera de la aplicación					**X**
Mi experiencia usando la aplicación es buena.			**X**		

9. Manual de Instalación

En este apartado se detallarán los pasos a seguir para instalar Liferay 6.2 y desplegar en el portal todos los componentes, contenidos web y estructura de páginas de la aplicación.

El manual detalla la instalación en un equipo con sistema operativo Windows de 64-bit, los requisitos recomendables de instalación son 4GB de memoria RAM y una CPU con al menos 2.0 GHz.

1. Instalación de herramientas

Será necesaria la instalación las siguientes herramientas:

- Java JDK 8
- MySQL 5.7
- Apache Maven 3.6.3

1.1 Java JDK 8

Se puede descargar a través de la página de Oracle
https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html

Se deberá seleccionar la distribución para Windows 64-bit

Linux x64 RPM Package	121.53 MB	jdk-8u261-linux-x64.rpm
Linux x64 Compressed Archive	136.48 MB	jdk-8u261-linux-x64.tar.gz
macOS x64	203.94 MB	jdk-8u261-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.77 MB	jdk-8u261-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.72 MB	jdk-8u261-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.23 MB	jdk-8u261-solaris-x64.tar.Z
Solaris x64	92.47 MB	jdk-8u261-solaris-x64.tar.gz
Windows x86	154.52 MB	jdk-8u261-windows-i586.exe
Windows x64	166.28 MB	jdk-8u261-windows-x64.exe

Figura 66 Página descarga Java JDK 8

A continuación, será necesario añadir la variable de entorno de Java. Esto se realiza en Panel de control -> Sistema y Seguridad -> Sistema -> Opciones avanzadas. Para definirla, pulsar sobre Variables de entorno

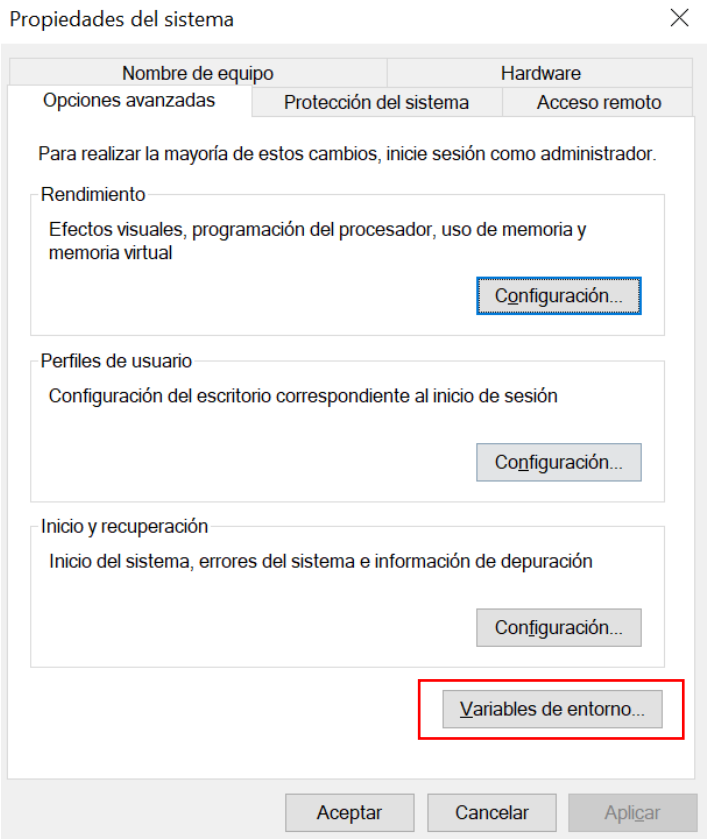


Figura 67 Página opciones avanzadas del sistema

En la ventana de Variables de entorno será necesario:

- En la sección variables del sistema crear una nueva variable
- El nombre de la variable ha de ser JAVA_HOME, el valor de la variable es la ruta donde se haya instalado localmente el jdk (java developer kit) , en el caso del equipo de desarrollo C:\desarrollo\java\install\jdk1.8.0_171. Finalmente pulsar en el botón aceptar
- La nueva variable JAVA_HOME debería aparecer en el listado de variables del sistema

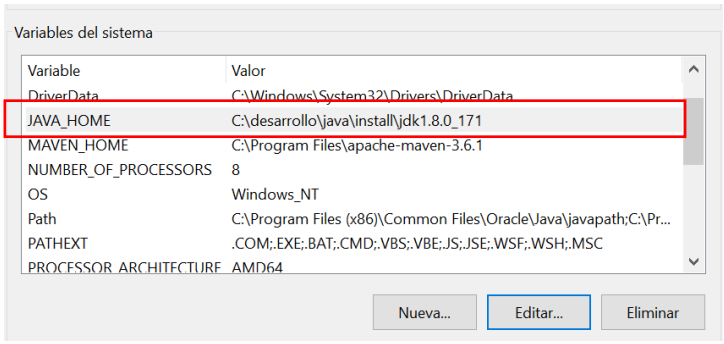


Figura 68 Variable de entorno JAVA_HOME

Añadir la nueva variable a PATH

- Seleccionar la variable Path del listado de variables del sistema, pulsar sobre Editar
- En la ventana de edición pulsar sobre Nuevo y escribir la siguiente ruta %JAVA_HOME%/bin
- Pulsar en el botón Aceptar.
- Comprobar que se ha añadido al listado

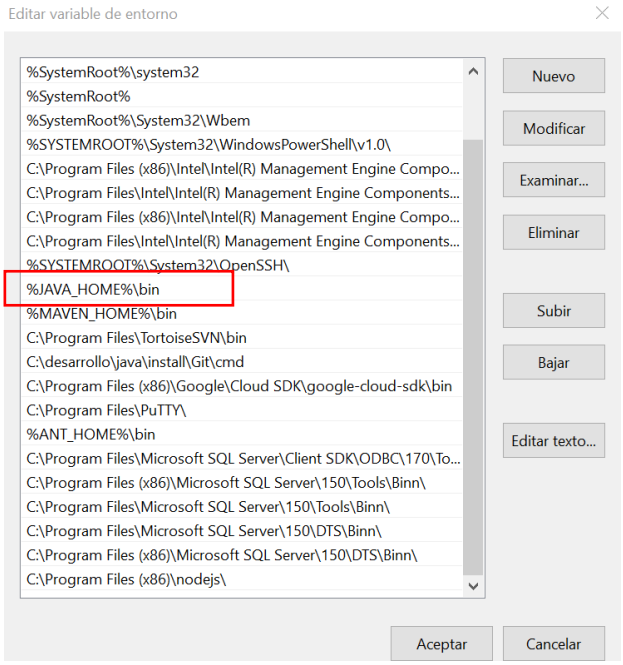


Figura 69 Variable de entorno Path

Para comprobar que se ha instalado correctamente

- Abrir la consola de comandos
- Escribir el siguiente comando: *java -version*
- El resultado obtenido ha de ser el siguiente

```
C:\Users\amoyano>java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```

Figura 70 Versión de Java

1.2 MySQL 5.7

A continuación, se describen los pasos para instalar MySQL 5.7:

- Acceder a la siguiente página https://dev.mysql.com/downloads/mysql/
- Pulsar sobre el botón Looking for previous GA versions?
- Seleccionar las siguientes opciones en los desplegables

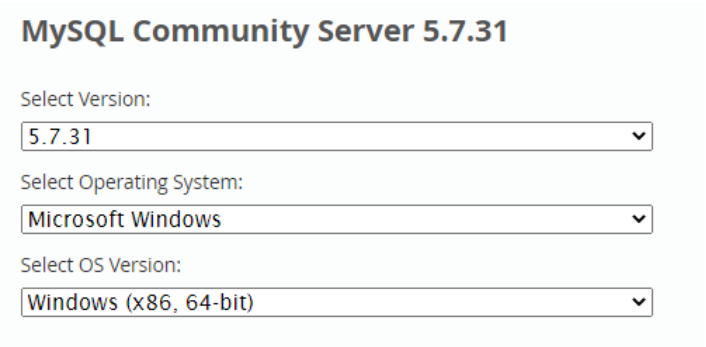


Figura 71 Configuración MySQL Community Server

- Descargar el archivo ZIP y ejecutar el instalador.
- Seleccionamos la configuración personalizada para seleccionar los productos que nos interesan
- Seleccionar los siguientes productos: MySQL Server 5.7.2, MySQL Workbench 6.3.2, MySQL Shell 1.0.5, Connector/J 6.0.6

Figura 72 Productos MySQL a instalar

- En el apartado Type and Networking seleccionar Standalone MySQL Server
- Dejar seleccionado el puerto por defecto, Port Numer: 3306
- En la sección Account and Roles se define la contraseña de Root, por ejemplo root
- En el apartado Windows Server dejar la configuración predeterminada y pulsar Next
- Finalizar la configuración

Por último, crearemos la base de datos de la aplicación, para ello:

- Ejecutamos la aplicación MySQL Workbench
- Creamos una nueva conexión con los siguientes parámetros

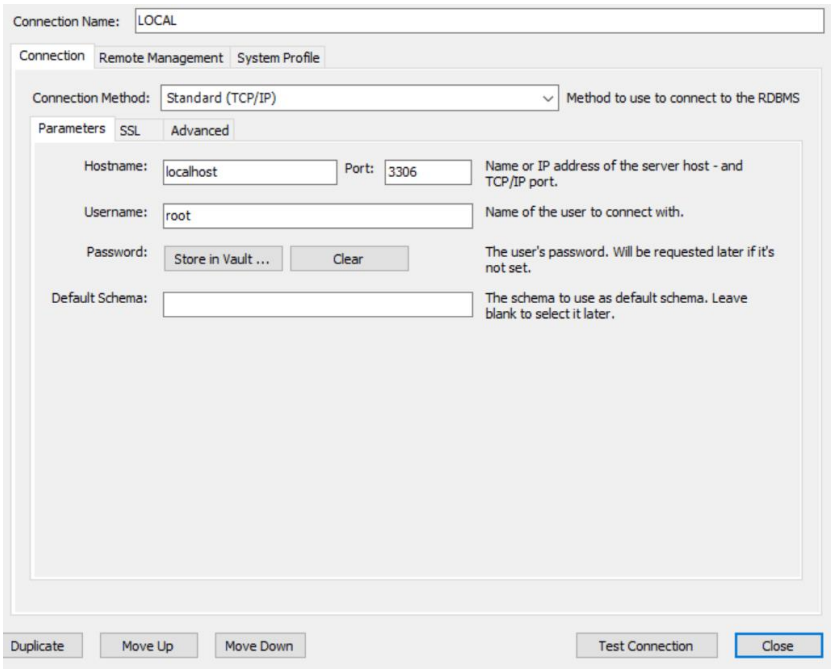


Figura 73 Creación conexión local de BD

- Introducir la contraseña definida en el proceso de instalación.
- En el apartado SCHEMAS hacer click derecho, seleccionar Create schema… y escribir *peaker* como nombre del esquema

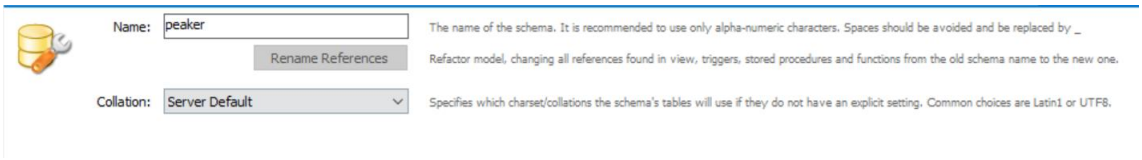


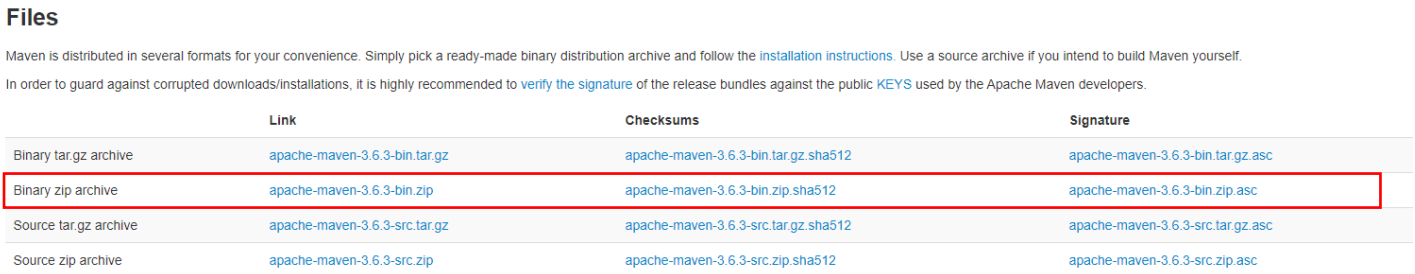
Figura 74 Esquema de BD

- Con todo esto habremos finalizado la instalación de la BD

1.3 Apache Maven 3.6.3

Se detallarán los pasos para instalar Maven, la herramienta de empaquetado y despliegue de componentes software. *Nota: es necesario previamente haber instalado Java y definido la variable JAVA_HOME. En el equipo de desarrollo la versión de Maven instalada es la 3.6.1, en el manual se instalará la 3.6.3*

- El archivo comprimido con el código fuente se obtiene en la siguiente página https://maven.apache.org/download.cgi



Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.6.3-bin.tar.gz	apache-maven-3.6.3-bin.tar.gz.sha512	apache-maven-3.6.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.6.3-bin.zip	apache-maven-3.6.3-bin.zip.sha512	apache-maven-3.6.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.6.3-src.tar.gz	apache-maven-3.6.3-src.tar.gz.sha512	apache-maven-3.6.3-src.tar.gz.asc
Source zip archive	apache-maven-3.6.3-src.zip	apache-maven-3.6.3-src.zip.sha512	apache-maven-3.6.3-src.zip.asc

Figura 75 Página de descarga Maven

- Extraer el directorio de ZIP por ejemplo, en la siguiente ruta C:/Archivos de programa/
- A continuación, será necesario añadir la variable de entorno de Java. Esto se realiza en Panel de control -> Sistema y Seguridad -> Sistema -> Opciones avanzadas. Para definirla, pulsar sobre Variables de entorno

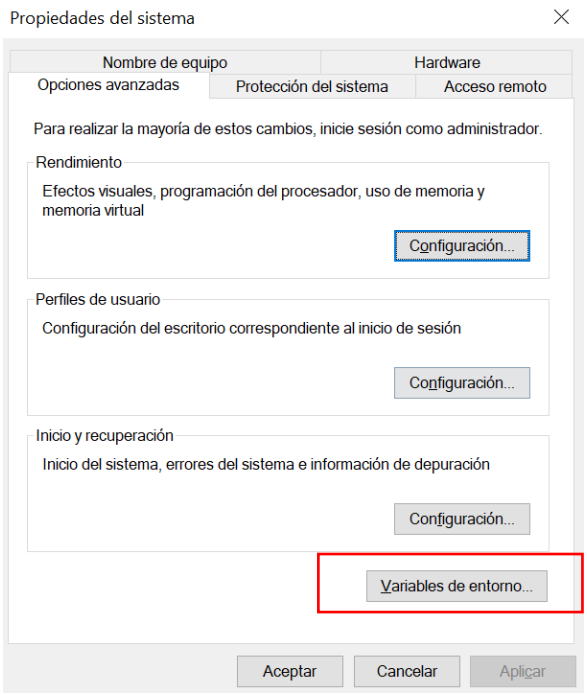


Figura 76 Página opciones avanzadas del sistema

En la ventana de Variables de entorno será necesario:

- En la sección variables del sistema crear una nueva variable
- El nombre de la variable ha de ser MAVEN_HOME, el valor de la variable es la ruta donde situado Apache Maven , en este caso C:/Archivos de programa/apache-maven-3.6.1 Finalmente pulsar en el botón aceptar
- La nueva variable MAVEN_HOME debería aparecer en el listado de variables del sistema

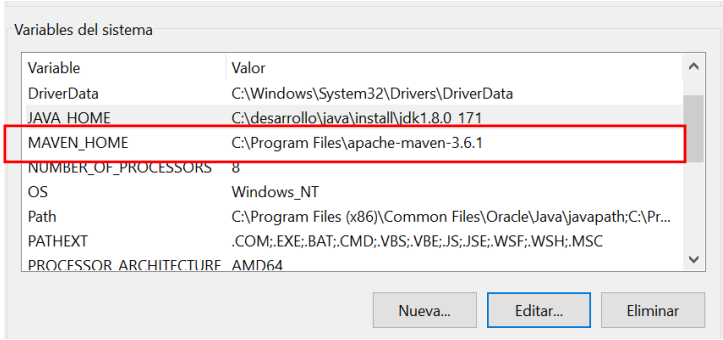


Figura 77 Variable de entorno MAVEN_HOME

Añadir la nueva variable a PATH

- Seleccionar la variable Path del listado de variables del sistema, pulsar sobre Editar
- En la ventana de edición pulsar sobre Nuevo y escribir la siguiente ruta %MAVEN_HOME%/bin
- Pulsar en el botón Aceptar.
- Comprobar que se ha añadido al listado

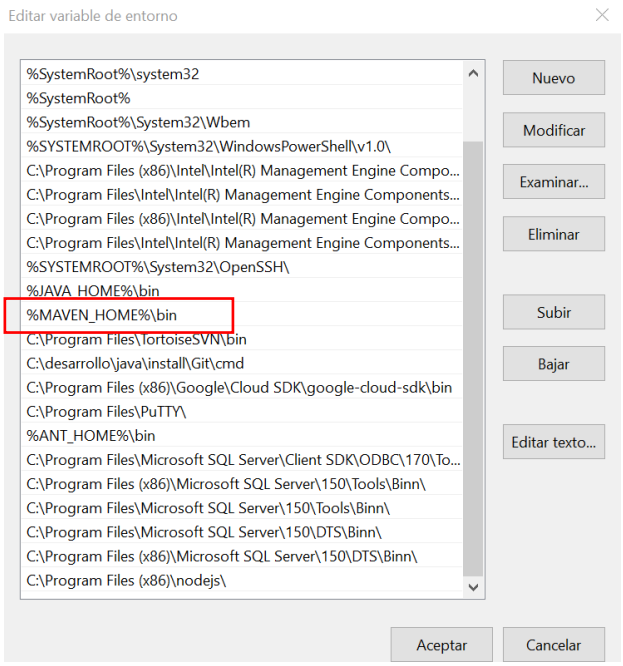


Figura 78 Variable de entorno Path

Para comprobar que se ha instalado correctamente

- Abrir la consola de comandos
- Escribir el siguiente comando: *mvn -version*
- El resultado obtenido ha de ser el siguiente

```
C:\Users\amoyano>mvn --version
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T21:00:29+02:00)
Maven home: C:\Program Files\apache-maven-3.6.1\bin\..
Java version: 1.8.0_171, vendor: Oracle Corporation, runtime: C:\desarrollo\java\install\jdk1.8.0_171\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Figura 79 Versión Maven desde cmd

Nota en el equipo de desarrollo la versión instalada es 3.6.1 en el caso de este proceso de instalación debería de ser 3.6.3

2. Instalación bundle Liferay Portal 6.2 con Apache Tomcat

Antes de comenzar con la instalación se ha de preparar la estructura de carpetas, en el caso del equipo de desarrollo ha sido:

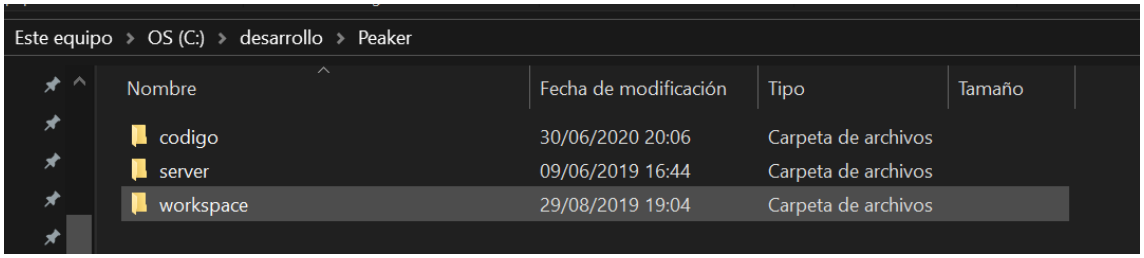


Figura 80 Estructura directorios Peaker

- Server: es el directorio donde se instalará el servidor web
- Código: carpeta donde se encuentran los portlets desarrollados
- Workspace: carpeta creada por eclipse para poder trabajar con los proyectos Java.

A continuación se detalla el proceso de instalación y configuración de Liferay Portal y Apache Tomcat:

- El paquete de instalación se obtiene en el siguiente enlace https://sourceforge.net/projects/lportal/files/Liferay%20Portal/6.2.5%20GA6/
- Seleccionamos la siguiente distribución

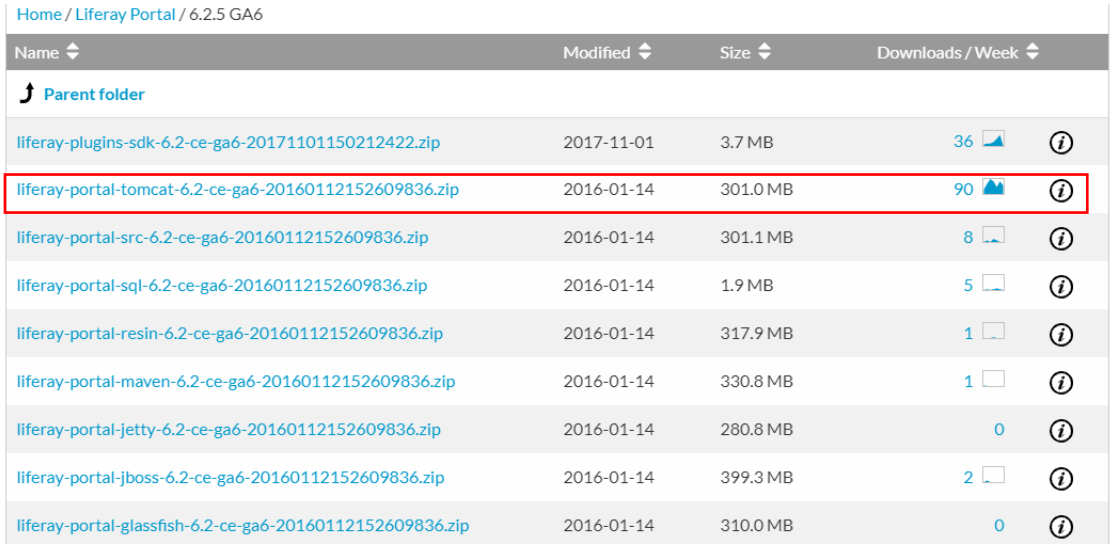


Figura 81 Página de descarga bundle Liferay y Tomcat

- Descomprimimos el fichero en la siguiente ruta C:/desarrollo/Peaker/server , la estructura de directorios es la siguiente.

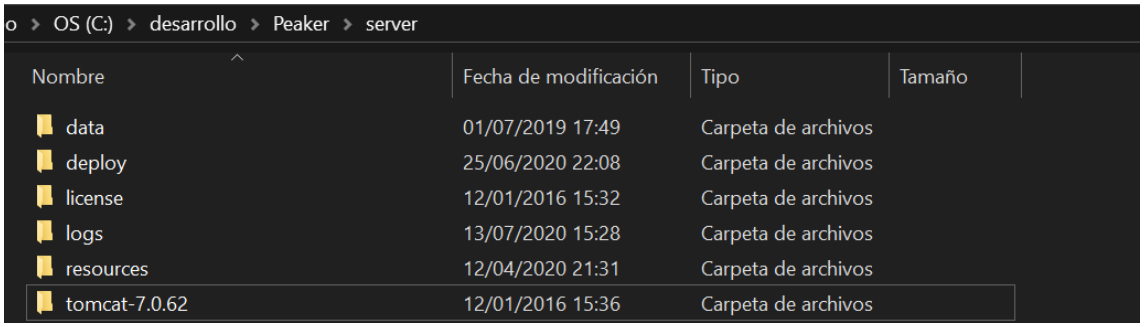


Figura 82 Estructura directorios Tomcat

- Creamos dos nuevos ficheros .properties con el siguiente contenido.
    - Portal-ext.properties: Se definen la variable para la conectividad con la base de datos, las variables de configuración para el envio de correos y la clave para conectar con la API de Google maps.

```
jdbc.default.jndi.name=jdbc/LiferayPool

setup.wizard.enabled=false

journal.article.types=announcements,blogs,general,news,press-release,test,FAQ


# To remove Password reminder question on first login
users.reminder.queries.enabled=false
users.reminder.queries.custom.question.enabled=false
terms.of.use.required=false

users.image.check.token=false

google.maps.license=AIzaSyA9yXOwg56_Vo78kG9TnGs_718RH2QrvC0


## SMTP Google
mail.session.mail.smtp.host=smtp.gmail.com
mail.session.mail.smtp.password=Alvaro35707
mail.session.mail.smtp.port=465
mail.session.mail.smtp.auth=true
mail.session.mail.smtp.user=travelwithpeaker@gmail.com
mail.session.mail.transport.protocol=smtp

theme.portlet.decorate.default=false

##
# Urls to access reCaptcha services
##
captcha.engine.recaptcha.url.script=http://api.recaptcha.net/challenge?k=
captcha.engine.recaptcha.url.noscript=http://api.recaptcha.net/noscript?k=
captcha.engine.recaptcha.url.verify=https://www.google.com/recaptcha/api/siteverify
```

Figura 83 Fichero portal-ext.properties

- portal-setup-wizard.properties: se definen la dirección y nombre del usuario por defecto y la ruta de instalación de Liferay.

```
admin.email.from.address=test@peaker.com
admin.email.from.name=Test Test
liferay.home=C:/desarrollo/Peaker/server
setup.wizard.enabled=false
```

Figura 84 Fichero portal-setup-wizard.properties

- Establecer conexión con la base de datos. Editamos el fichero ROOT.xml situado en C:/Peaker/server/tomcat-7.0.62/conf/Catalina/localhost.

```
<Context path="" crossContext="true">
    <Resource
        name="jdbc/LiferayPool"
        auth="Container"
        type="javax.sql.DataSource"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost/peaker?useUnicode=true&amp;characterEncoding=UTF-8"
        username="root"
        password="admin"
        defaultTransactionIsolation="READ_COMMITTED"
        connectionProperties="useUnicode=true;characterEncoding=UTF-8;autoReconnect=true;allowMultiQueries=true"
        maxActive="50"
        maxIdle="30"
        initialSize="20"
        minIdle="10"
        maxWait="10000"
        testOnBorrow="true"
        testOnReturn="false"
        testWhileIdle="true"
        validationQuery="SELECT 1"
        validationQueryTimeout="10"
        timeBetweenEvictionRunsMillis="5000"
        minEvictableIdleTimeMillis="60000"
        removeAbandoned="true"
        removeAbandonedTimeout="60"
        validationInterval = "30000"
        maxAge = "0"
    />
```

Figura 85 Fichero ROOT.xml

- El siguiente paso será arrancar el servidor para que se creen las tablas de Liferay en BD. Para arrancar el servidor ejecutamos el fichero startup.bat situado en C:/Peaker/server/tomcat-7.0.62/bin.

- Copiamos la carpeta recursos.zip proporcionada en el código de la aplicación en la siguiente ruta C:/Peaker/server/, esta carpeta contiene recursos estáticos como imágenes o archivos JSON.

- Comprobamos que las tablas se han creado correctamente en base de datos.

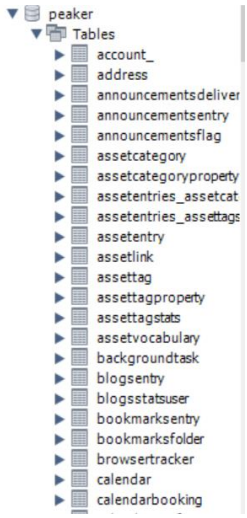


Figura 86 Tablas de Liferay en base de datos

3. Configuración inicial de Liferay

- Abrimos un navegador web y accedemos a *http://localhost:8080* , al ser el primer acceso será necesario realizar la configuración inicial de Liferay Portal

- Nos logueamos en el sistema con el siguiente usuario test@liferay.com y la contraseña elegida en el proceso de configuración del portal.

- Accedemos a la parte de administración del portal desde el menú de la parte superior derecha. Pulsamos sobre Panel de control y seleccionamos Sitios Web.

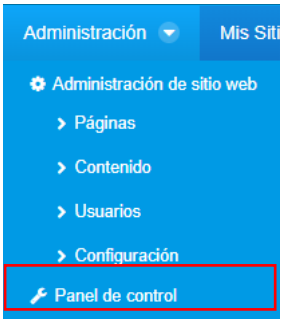


Figura 87 Menú de administración

- Pulsamos sobre Configuración e introducimos los siguientes valores.

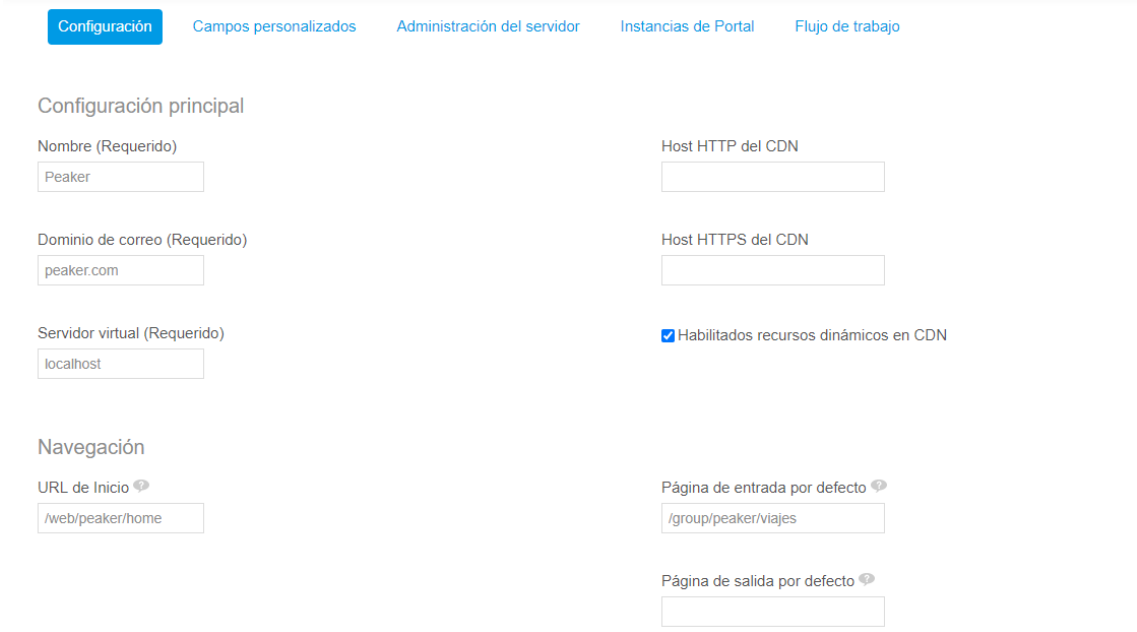


Figura 88 Configuración general de Liferay

- A continuación, pulsamos sobre Sitios Web en el menú situado en la parte superior. Accedemos al sitio web Peaker.

- Accedemos a la página de ajuste del sitio situada en el menú de Configuración

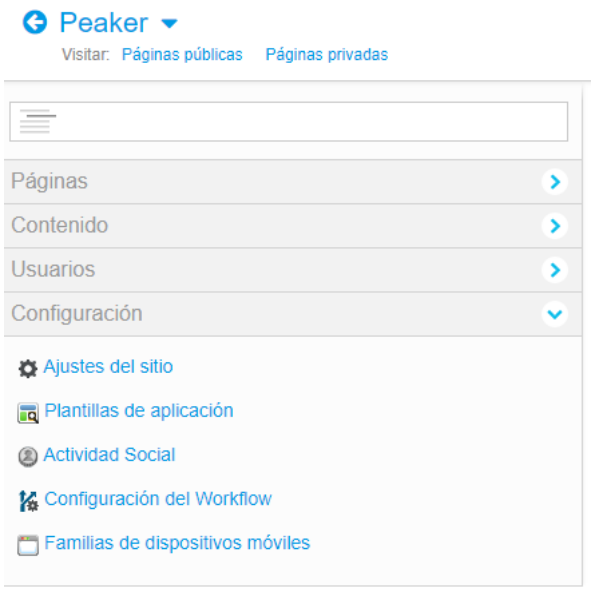


Figura 89 Menú lateral del sitio web apartado Configuración

- Nos situamos en la sección URL del sitio y cambiamos el valor del campo URL amigable

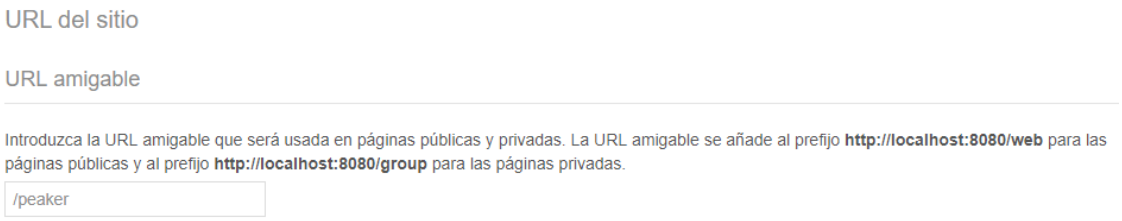


Figura 90 Configuración URL amigable

- El siguiente paso es importar los contenidos, estructura de carpetas y páginas de la aplicación. Para ello será necesario importar los archivos .lar del directorio Contenido proporcionado en el código fuente.

    - Para importar las páginas privadas es necesario situarse en la sección Paginas privadas y pulsar en el botón Importar. Adjuntar el archivo paginas-privadas.lar

Figura 91 Importar páginas privadas

- Las paginas públicas no será necesario importarlas porque es solo una, bastará con cambiar el nombre de la página a *Welcome*

- Para importar las páginas privadas es necesario situarse en la sección Contenido -> Documentos y multimedia. Pulsamos sobre el icono de Documentos y Multimedia situado en la parte derecha y seleccionamos Exportación/Importación. Adjuntar el archivo documentos-multimedia.lar.

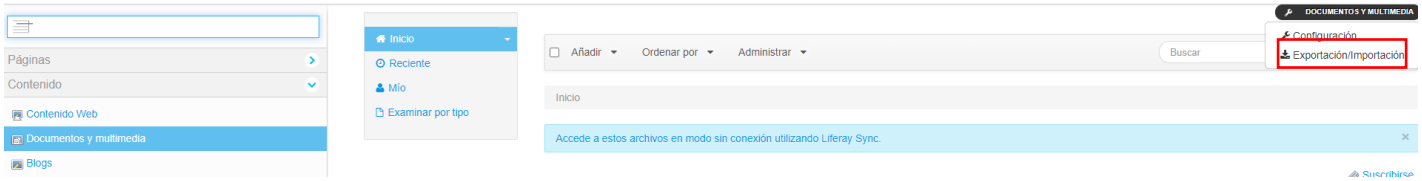


Figura 92 Importar documentos multimedia

4. Despliegue de portlets

Para facilitar el despliegue de los portlets se han facilitado en el directorio de código fuente los ficheros .war y .jar.

- Con el servidor parado, copiar los ficheros jar en la siguiente ruta C:/Peaker/server/tomcat-7.0.62/lib/ext

- Arrancar el servidor ejecutando el archivo startup.bat.

- Cuando el servidor haya arrancado completamente copiar de uno en uno en la carpeta C:/Peaker/server/deploy. Comprobar en el terminal del servidor que el componente se ha desplegado correctamente.

- Tras desplegar todos los portlets comprobamos en base de datos que se han creado las tablas correspondientes.

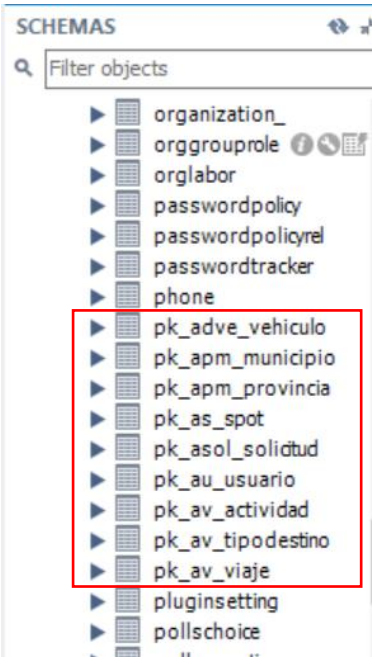


Figura 93 Tablas de los portlets en base de datos

- Finalmente añadiremos los portlets a las paginas previamente importadas. Para añadirlos será necesario loguearse en la aplicación con el usuario test@liferay.com. Accediendo con este usuario tendremos visible un menú lateral donde se encuentran los portlets desplegados.

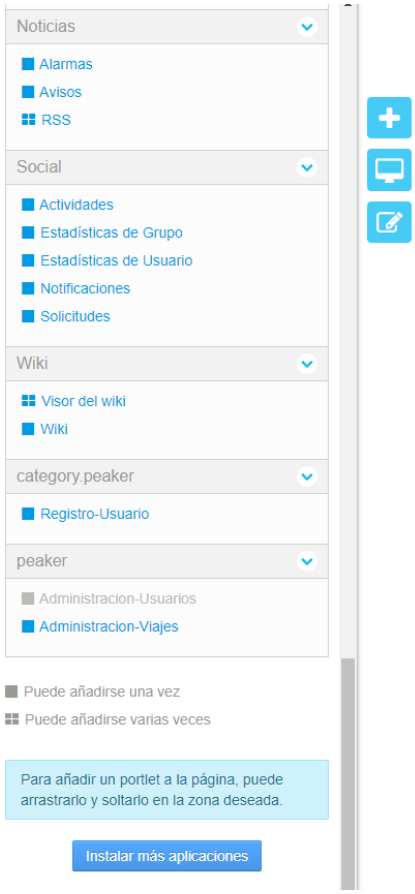


Figura 94 Menú lateral para añadir componentes

- A continuación, detalla un listado de que portlet se despliega en cada página

  - Página: Welcome — Portlet: Registro-Usuarios
  - Página: Mi Perfil — Portlet: Adminsitracion-Usuarios
  - Página: Viajes — Portlet: Administracion-Viajes
  - Página: Mis Solicitudes — Portlet: Administracion-Solicitudes

10. Conclusiones

El objetivo de este proyecto era la creación de una aplicación web donde personas que realicen ciertas actividades deportivas puedan ponerse en contacto con otras que compartan esos gustos y puedan realizar la actividad juntos y repartir los gastos que se generen. La aplicación generada cumple con los requisitos establecidos en los documentos de análisis y diseño del sistema y es viable la publicación de una versión piloto en un entorno real.

Uno de los objetivos inicialmente era la utilización de software de código abierto para el desarrollo de este proyecto, hecho que se ha llevado a cabo en las diferentes capas de la aplicación. Como sistema de gestión de base de datos se ha utilizado *MySQL* herramienta en la que ya tenía experiencia previa por lo que su configuración y utilización ha resultado sencilla. Como servidor de aplicaciones la utilización de Apache Tomcat por ser el servidor web idóneo para los módulos java que se han desarrollado además la existencia de *bundles* en los que con una sencilla configuración tendremos integrado Liferay Portal en el servidor web.

Teniendo en cuenta que se decidió orientar el desarrollo software a microservicios, la utilización de *Spring MVC* para el desarrollo de los módulos Java ha sido acertada por las siguientes razones. Por su propio diseño modular que facilita en gran medida la creación y mantenimiento de los componentes que se han desarrollado. Su fácil configuración para los seis módulos que se han desarrollado, este aspecto puede ser tedioso en otros frameworks de desarrollo. Los mecanismos que utiliza Spring para agilizar el desarrollo como la inyección de dependencias o las anotaciones para facilitar la programación basada en MVC. Por último, al ser el framework más utilizado para desarrollo de aplicaciones java cuenta con una comunidad muy activa que genera mucha documentación a la que acudir.

El hecho de realizar la aplicación web Peaker utilizando *Liferay Portal 6.2* que es la herramienta de código abierto que venía utilizando a diario en mi ámbito laboral, me ha demostrado que si es la herramienta utilizada en entornos productivos ofreciendo buenos resultado también me serviría para la elaboración de la aplicación. Este hecho me ha permitido una fácil adaptación al entorno de trabajo además de ampliar mi conocimiento sobre esta tecnología. Aunque soy consciente de que el uso que se le ha dado en este proyecto no es el habitual que se le da a un portal web, el hecho de desarrollar los portlets por completo y el alto nivel de personalización que ofrece Liferay a los desarrolladores me han permitido beneficiarme de las herramientas que proporciona para adaptar su funcionamiento a mis propósitos y crear una aplicación que cumple con los requisitos definidos.

10.1 Posibles ampliaciones

El sistema resultante cumple satisfactoriamente con los requisitos establecidos, pero se podría en un futuro realizar mejoras y ampliaciones sobre los diferentes módulos que lo componen. También se valorará el desarrollo de módulos que ofrezcan nuevas funcionalidades y que podrán ser integrados fácilmente en Liferay. A continuación, se propondrán diferentes ampliaciones sobre los portlets ya desarrollados

- **Valoración de viajes**: una funcionalidad que inicialmente se había pensado incluir en el sistema desarrollado, aunque finalmente no se ha añadido, es un sistema de valoración de viajes a través del cual los usuarios puedan posteriormente a la realización del viaje establecer una valoración de este a través de un formulario de satisfacción puntuando diferentes aspectos del viaje y una opinión sobre el mismo. Esta información se ofrecería al resto de usuarios de manera que les ayude en su elección. Se podría enfocar de dos maneras:

    - La valoración se realiza sobre el usuario que organiza el viaje, aunque pienso que esto no es lo más adecuado para no convertir el sistema de valoración en un condicionante a la hora de la elección del viaje que organiza cierto usuario. Si finalmente se realizara de esta forma esta ampliación la forma más adecuada sería añadir una nueva entidad que se gestionaría desde el portlet Administración-Usuarios, y mostrar esta información en una página de este portlet que sea accesible para el resto de los usuarios ya que actualmente las páginas de este portlet solo muestran información del usuario registrado en ese momento.

    - Valoración de Spots, la opción más deseable ya que hemos desarrollado el portlet Administración-Spots y solo tendríamos que ampliar si funcionalidad, aunque también se podría realizar esta ampliación en el portlet Administración-Viajes. No se desea convertir a este portlet en un componente monolítico, sino que nuestro sistema se orienta más bien al desarrollo de componentes pequeños cuyas funcionalidades puedan ser consumidas fácilmente por el resto. Para llevar esto a cabo sería necesario añadir una nueva entidad donde se guarde la información de la valoración del viaje en función del *Spot* y desarrollar una interfaz de usuario donde se pueda consultar la información y opinión de viajes realizados a cada uno de los lugares. Se tendría que crear una página nueva en el portal llamada Valoración Spots, pero ocultarla para los usuarios logueados de manera que solo se pueda acceder a ella a través de uno de los portlets ya existentes con la idea de mantener la simplicidad y añadir nuevas funcionalidades a los portlets ya existentes. Actualmente poder consultar las valoraciones y opiniones de los usuarios sobre productos o experiencias es de gran interés.

- **Integración con un sistema de mensajería,** actualmente la aplicación comunica a los usuarios la resolución de solicitudes o información de las acciones realizadas a través de un email. Si bien es cierto que actualmente el correo electrónico es el canal de comunicación para multitud de aplicaciones y donde se reciben facturas, estado de compras realizadas entre otras muchas notificaciones. esto hace que sea consultado prácticamente a diario por lo que es una vía de comunicación muy importante. Sería interesante para nuestra aplicación integrarnos con algún sistema de mensajería como *WhatsApp* o *Telegram*, de manera que se pueda notificar al usuario mediante estas aplicaciones; las solicitudes recibidas, canceladas o recordatorios del viaje.

- **Permitir la creación de Spots,** dado que como destino del viaje se puede marcar cualquier punto en el mapa, es fácil que se dé el caso de que varios marcadores con latitudes y longitudes diferentes pero muy próximas hagan referencia al mismo destino. La creación del concepto Spot, que hace referencia a un lugar habitual donde se realizan cierto tipo de actividades, nos permite unificar la información de los viajes y simplificar el proceso de búsqueda de viajes mostrando al usuario todos los viajes que comparten destino. Teniendo en cuenta que para la primera versión del producto está pensada para utilizar en Asturias, se ha realizado una carga inicial a través de inserciones en BD de algunos de los lugares más habituales de la provincia para la realización de actividades. Para proporcionar un valor añadido sería interesante poner a disposición de los usuarios un formulario donde puedan añadir la información necesaria para la creación de un nuevo Spot de manera que la gestión de Spots sea una funcionalidad colaborativa entre los usuarios de la aplicación, pero con la supervisión de un administrador para evitar añadir datos erróneos. Será necesario diseñar una interfaz de usuario visualizable desde la parte privada de la aplicación donde el administrador pueda consultar la información enviada por los usuarios y estudiar si finalmente se añade a la aplicación.

- **Desarrollo de un tema de apariencia propio**: actualmente la aplicación utiliza el *Theme* propio de *Liferay*, que es modificado a través de código *CSS* y *JS* que se encuentran en los portlets. Aunque simplemente con eso se consigue una interfaz uniforme un desarrollo que nos ayudaría a mejorar el aspecto gráfico de nuestra aplicación es el desarrollo de un *Theme* propio para poder centralizar las modificaciones que se realizan sobre la apariencia del portal. Pese a que Liferay Portal proporciona mecanismos para hacer responsivas las diferentes páginas del portal, actualmente nuestra aplicación no cuenta con un diseño resposive, si bien es cierto que entre nuestros objetivos no se encuentra que el diseño se adapte correctamente a la visualización desde dispositivos móviles, sería interesante para que un usuario pueda utilizar nuestra aplicación desde su dispositivo móvil.

11. Bibliografía

What is Maven. https://maven.apache.org/what-is-maven.html

Oracle https://www.oracle.com/es/database/what-is-a-relational-database/

Liferay in Action. Autor Rickar Sezov

Maven: The Definitive Guide. Autor OReilly

Liferay Portal 6.2 https://help.liferay.com/hc/es/categories/360000929231

MVC en Spring http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html

Liferay's Architecture https://liferay.dev/blogs/-/blogs/liferay-s-architecture-the-service-layer

Spring framework https://docs.spring.io/spring/docs/2.0.x/reference/mvc.html

Servicios de redes de área local. Autor: Miguel Angel Palomares Ortega

JSR 286 https://jcp.org/en/jsr/detail?id=286

Apache Tomcat http://tomcat.apache.org/

Spring in Action Autor C.Walls.

MVC Portlet in Liferay http://liferay-article.blogspot.com/2017/11/spring-mvc-portlet-in-liferay-with.html

Maven Service Builder https://www.opensource-techblog.com/2017/09/liferay-maven-service-builder.html

Google Maps Documentation https://developers.google.com/maps/documentation

Apuntes de Ingeniería del Software. Autores Javier Tuya y José García Fanjul.

Apuntes Ingeniería de Servicios.

Convenio 2009 para empresas de consultoría https://www.boe.es/buscar/doc.php?id=BOE-A-2009-5688