UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

Desarrollo de un sitio Web para una agencia de viajes

Proyecto Final de Carrera

[Ingeniería Técnica Informática de Sistemas]

Autor: Patricia Baixauli Sánchez

Director: Félix Buendía García

Septiembre 2015

Resumen

En los últimos años ha habido un aumento de los viajes de ocio en todo el mundo, siendo uno de los negocios más en auge. Cada día existen más agencias de viaje o sitios web en los que el usuario puede encontrar el viaje soñado y siempre al mejor precio. Por eso, nosotros queremos ofrecer una herramienta para que el usuario sea capaz de descubrir cuál es su destino perfecto para unas vacaciones inolvidables. Para ello, hemos creado un sitio web que les permitirá recorrer el mundo según las preferencias de cada uno de ellos desde cualquier lugar del mundo.

Palabras clave: viajes, destino, país, recomendador.

Tabla de contenidos

Contenido

1 Introducción

En este documento se resume la memoria del Proyecto Final de Carrera (a partir de ahora PFC) realizado por Patricia Baixauli Sánchez, perteneciente a la titulación de Ingeniería Técnica Superior en Informática de Sistemas para la Universidad Politécnica de Valencia.

1.1 Objetivos

El principal objetivo marcado con la realización de este proyecto será desarrollar un sitio web para la recomendación de destinos turísticos que permita:

- Ofrecer al usuario la posibilidad de filtrar destinos según sus necesidades
- Buscar destinos según zonas geográficas
- Consultar sus últimas búsquedas
- Recomendar viajes a los destinos seleccionados

Como ya existen muchos sitios web en la red que ofrecen viajes organizados a destinos, se ha querido dar un enfoque diferente a nuestro sitio web para poder realizar algo distinto a lo que el mercado ya ofrece. Por lo tanto, no es objetivo competir con los sitios de viajes ya existentes sino más en la idea de poder ofrecer un catálogo de destinos al usuario según sus gustos y necesidades.

Otros de los objetivos del proyecto es que sea compatible y usable desde la mayoría de dispositivos que hoy en día están en pleno auge, como puedan ser los *smartphones* o las *tablets*. Por eso, este sitio web se programará con tecnologías específicas para el desarrollo móvil.

1.2 Contexto

Como ya se ha comentado, existen multitud de sitios web para la reserva de viajes pero pocos *recomendadores* para buscar destinos no conocidos o destinos que se adapten a nuestras necesidades.

Existen algunos sitios web para la recomendación de destinos o para búsquedas según gustos y necesidades, así que en esos ejemplos nos hemos apoyado para la realización de este proyecto.

Para nuestro sitio web nos hemos basado en dos sitios especialmente, que son way?away y Busco un viaje. Los ejemplos de sus sitios web los podemos ver en la ilustración 1 y 2.

Ilustración 1. www.way-away.es

Ilustración 2. www.buscounviaje.com

1.3 Estructura de la memoria del proyecto

La siguiente memoria se estructura siguiendo las distintas fases por las que ha pasado el proyecto del portal web hasta la implementación final del mismo.

La memoria se divide en los siguientes capítulos:

- **Capítulo 1.** Introducción
  En este punto se describen los objetivos, motivaciones y contexto en el que se propone y se realiza este proyecto de sitio web.
- **Capítulo 2.** Especificación de requisitos
  En la especificación de requisitos se recogen todos los puntos a tener en cuenta para la realización del proyecto. Desde la motivación inicial hasta las funciones finales del producto.
- **Capítulo 3.** Análisis
  En este punto se tratan los casos de uso, las clases que tendrá el proyecto final y los diagramas de flujo de las diferentes funcionalidades.
- **Capítulo 4.** Diseño
  Arquitectura y diseño de la aplicación, tanto a nivel superficial como en otros niveles.
- **Capítulo 5.** Implementación
  Se describen todos los servicios implementados en las diferentes capas, las herramientas utilizadas, la estructura global del sitio web y las pruebas realizadas.
- **Capítulo 6.** Conclusión
  Conclusiones después de realizar el trabajo y problemas y objeciones encontrados durante el proceso.
- **Capítulo 7.** Bibliografía
  Textos y material utilizados para la redacción y realización del proyecto.

Al final de la memoria se incluyen diversos anexos para complementar la información de los apartados anteriores.

2 Especificación de requisitos

2.1 Introducción

La especificación de requisitos de software (ERS) se utiliza para describir completamente el comportamiento del sistema que se va a desarrollar. Aquí se van a describir los requisitos funcionales y las características de diseño de interfaz y de uso que tendrá nuestra página web.

2.1.1 Ámbito

Este Proyecto está orientado a la programación e implementación de un portal web dedicado a la búsqueda de destinos para viajar. La idea para llevar a cabo este proyecto nació de la falta de opciones a la hora de encontrar un destino que se adapte a nuestras necesidades tanto como a nuestros gustos. Por ello, este proyecto puedo ayudar a la gente a encontrar un destino de viajes introduciendo muy pocos datos.

En nuestra página se permitirá elegir entre 7 zonas diferentes del globo terráqueo para delimitar un poco la búsqueda. Se podrá después afinar más la búsqueda introduciendo diferentes criterios esenciales para el viajero, tal como puede ser que sea un destino de playa o para hacer turismo por grandes ciudades. Después de la elección del destino se podrán contemplar los diferentes tipos de vacaciones que se ofrecen para ese destino, todo dependiendo de la búsqueda que haya hecho el usuario. Las búsquedas de los usuarios registrados quedarán guardadas en un histórico que podrá consultar cuando necesite.

Habrá dos tipos de usuario para el acceso a la web, el invitado y el registrado, y cualquiera de ellos podrá realizar búsquedas en el portal, sin que las del usuario invitado queden guardadas para posterior consulta. El usuario administrador podrá además insertar un nuevo destino desde la propia página web.

Existen además dos módulos extra que son un *recomendador* aleatorio de destinos para aquellos usuarios que no quieran hacer una búsqueda, así como un apartado de últimas búsquedas para ver los últimos destinos buscados (en caso de ser un usuario registrado).

2.1.2 Definiciones, acrónimos y abreviaturas

Las definiciones que aparecen durante el documento se encuentran aquí resumidas:

- Portal web: Página de inicio que permite el acceso a las diferentes opciones de un sitio web.
- Usuario: Persona que hace uso de la aplicación/web.
- Navegador: Aplicación software que permite visualizar páginas escritas en lenguaje HTML.
- Sitio web: Conjunto de páginas web sobre el mismo tema que tienen el mismo nombre de dominio.
- Dominio: Nombre con el que se conoce un sitio web.
- Proceso: Acciones que se desencadenan después de un evento en la aplicación
- Evento: Cualquier cambio que ocurra en la aplicación, ya sea por parte del usuario como de la aplicación misma.
- Aplicación: Programa/web implementado para un fin específico. En nuestro caso un sitio web para encontrar el destino perfecto.
- Interfaz: Medio en el cual el usuario se puede comunicar con un lenguaje no verbal con los programas o aplicaciones.
- UML: Sus siglas, lenguaje unificado de modelado. Lenguaje gráfico para visualizar, especificar y construir un modelo de un sistema software.

2.1.3 Referencias

- Consejos para el desarrollo de tu proyecto final de carrera (PDF). Francisco J. Abad Cerdá. Año 2010.

2.1.4 Visión global

La sección de especificación de requisitos consta de tres partes. En la primera de ellas se realiza una introducción al proyecto y se proporciona una visión general de los requisitos del sistema.

En la segunda parte tenemos se realiza la descripción global del producto, detallando las principales funciones que esté tendrá, así como los datos a utilizar, las características de los usuarios o las restricciones que pueda tener el proyecto.

En la tercera parte encontraremos los requisitos técnicos, tanto a nivel de interfaz con el usuario como a nivel interno con el contenido y las funciones.

2.2 Descripción general

2.2.1 Perspectiva del producto

Nuestro sitio web será un portal que debe proporcionar al usuario una amplia gama de destinos según sus gustos, cualidades o necesidades. El mercado de sitios web para reserva de vacaciones es ya muy amplio, por eso nuestro sitio web se va a centrar más en la decisión anterior a la de contratar un viajes, que es saber adónde.

Esta opción permitirá al usuario conocer destinos que tal vez no conociera, o escoger un destino por lo que éste le pueda ofrecer.

La aplicación o portal web estará alojada en un servidor web. La base de datos será MySql por ser esta de uso libre, una base de datos robusta y gran almacenamiento que nos permitirá almacenar toda la información perteneciente a los destinos, las últimas búsquedas así como los usuarios registrados.

Se utilizará jQuery Mobile para la implementación del sitio web para asegurarnos que la web es visible y usable desde la mayoría de dispositivos móviles. También, al ser un sitio web nos aseguramos la accesibilidad desde cualquier navegador web, lo que implica que usuarios de cualquier sistema operativo puedan utilizar el sitio web.

2.2.2 Funciones del producto

En este sitio web se va a ofrecer al usuario una forma de encontrar el destino que más se adapta a sus necesidades, y dentro de éste, las diferentes opciones que podrá realizar en el destino.

Las funciones principales de la página serán, ofrecer el destino según las siguientes características.

Se podrá hacer una búsqueda de destinos a gran escala utilizando la zona del mundo a la que nos gustaría ir. Según la zona elegida se mostrarán los destinos que pertenezcan a esa zona. Las zonas serán los continentes físicos.

A parte de poder elegir a qué zona del mundo queremos viajar, se podrá elegir entre tipos de vacaciones que se pueden disfrutar en cada uno de los destinos. Los filtros incluirán vacaciones tipo playa, aventura, relax, gastronomía o grandes ciudades, pudiendo combinarse entre ellos o elegirse de uno en uno.

El formulario se enviará y la búsqueda devolverá los destinos que estén dentro de los criterios proporcionados por el usuario.

Habrá sección de últimas búsquedas para los usuarios registrados en las que podrán tener un recordatorio de los últimos destinos buscados.

Para poder distinguir entre usuarios registrados e invitados tendremos también un formulario de registro para nuevos usuarios en los que serán necesarios un nombre, contraseña y dirección de e?mail.

También habrá una parte para el acceso de los usuarios, ya sean simples usuarios de la web o administradores del sitio.

A los administradores del sitio se les estará permitido añadir un destino desde el sitio web y por tanto habrá también un formulario donde se rellenarán los campos necesarios para la creación de un destino nuevo.

2.2.3 Características del usuario

Los usuarios que accedan al sitio web serán personas con una conexión a internet y que tengan las mínimas nociones sobre cómo funciona la red. En nuestro sitio web hay tres tipos de usuarios distintos, que son los siguientes:

- **Usuarios invitados**
  Son los usuarios que no se han registrado en el sitio web y que, por tanto, menos privilegios tienen. Pueden hacer consultas de destinos y ver los destinos aleatorios del día, sin posibilidad de guardar los resultados de la búsqueda.

- **Usuarios registrados**
  Son los usuarios que se han registrado en el sitio web y por tanto tienen una cuenta de usuario. Pueden consultar destinos, ver los destinos aleatorios y recuperar sus últimas búsquedas.

- **Usuario administrador**
  Es el usuario que se encarga de gestionar y mantener el sitio web. Pueden hacer lo mismo que cualquier usuario registrado en cuanto a

consultas y búsquedas. Gozan de más privilegios que los usuarios registrados y se les permite actualizar y ampliar los destinos desde el sitio web.

2.2.4 Restricciones generales

La finalidad de este proyecto es puramente académica por lo tanto, los servicios ofertados en el sitio web pueden no corresponder con la realidad o ser más limitados de lo que serían en cualquier página web a la disposición del público. Puede que los datos no sean exactos o que aparezcan menos resultados debido al tamaño de la base de datos.

Los datos requeridos para el registro en el sitio web no son comprobados y no serán en ningún caso parte de una base de datos pública.

2.3 Requisitos específicos

2.3.1 Interfaz

La interfaz de nuestro sitio web utilizará el navegador web para mostrarse al público, pudiéndose utilizar gracias esto en cualquier sistema operativo.

El sitio se dividirá en 3 partes bien diferenciadas. Cabecera, información general de la página y pie de página.

La ilustración 3 nos muestra el boceto para el diseño de la interfaz de inicio.

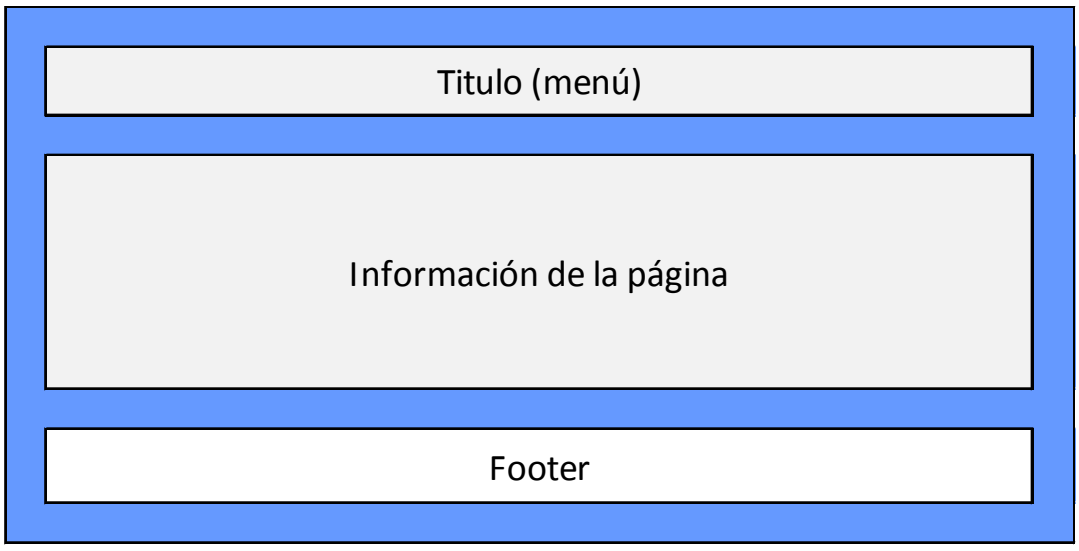

Ilustración 3. Diseño página de inicio

La parte superior contendrá el encabezado en el cual se podrá ver el nombre o la descripción del sitio. También puede que disponga de un pequeño menú para el registro y acceso de los diferentes tipos de usuarios y un mensaje de bienvenida al usuario para saber si éste está registrado en el sitio web o actúa como invitado. También habrá un botón de *Logout*, para finalizar la sesión del usuario en nuestro sitio web.

En la parte central tenemos la información principal que nos ofrece el sitio web. Al ser un diseño pensado para dispositivos móviles, tanto el índice o página principal del sitio como los resultados de la primera búsqueda de destinos, tendrán el mismo modelo.

En el índice podremos ver un menú diseñado en acordeón para las diferentes secciones que tenemos en el sitio. En esta parte nos encontraremos con la sección ¿Dónde voy?, que es la función principal de nuestro sitio web, los destinos aleatorios del día y la función de recordatorio de las últimas búsquedas de cada usuario.

Cuando utilizamos la sección ¿Dónde voy?, los destinos aleatorios o las últimas búsquedas, llegamos otra página con un diseño similar a esta.

En ella nos encontraremos con la cabecera y luego una lista de destinos para el usuario registrado (en el caso de últimas búsquedas) o el resto de destinos también para invitados.

Cuando elijamos un destino, llegaremos a una página con un diseño un tanto diferente. Se ha elegido este diseño para darle protagonismo al destino solicitado.

En la ilustración 4 se puede observar el boceto de lo que será el diseño para la página del destino solicitado:

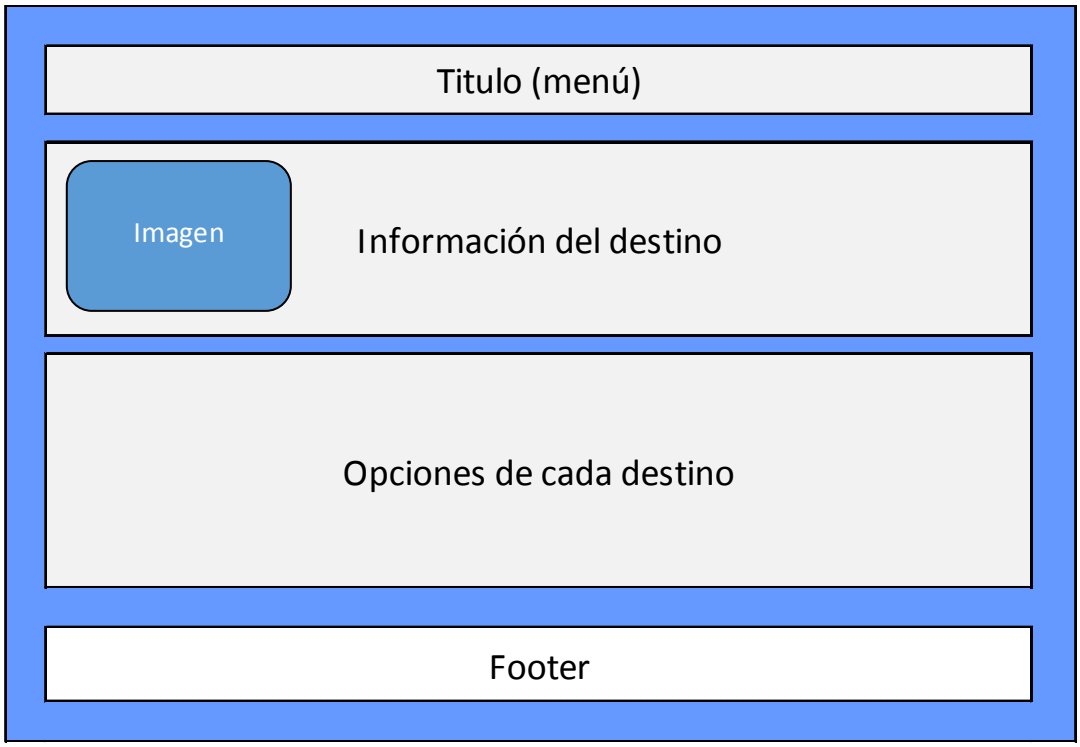

Ilustración 4. Diseño página destino elegido.

En la parte perteneciente a la información del destino se podrá ver una imagen del destino ocupando la mayor parte del espacio. A continuación el nombre del destino, la zona donde se encuentra y los diferentes tipos de vacaciones que ofrece.

En la sección siguiente habrá espacio para las diferentes actividades o viajes organizados que se ofertan desde el sitio web con información referente a estos.

En la última parte de la página web irá el *footer*. En cualquier página web esto se rellenará con la información de la empresa, el contacto y las declaraciones de privacidad. En nuestro caso será simplemente diseño.

2.3.2 Contenido

El contenido del sitio web estará formado por los destinos que hayan sido previamente incluidos en la base de datos.

En la parte dedicada a mostrar los destinos seleccionados con las peticiones del usuario se verá el destino, con una imagen de la ciudad, el país y la zona a la que pertenece.

Una vez elegido el destino, se podrá ver una imagen en un tamaño mayor que el anterior, el nombre del destino, la zona y los requisitos que cumple ese destino para el usuario.

Aparte de la información del destino, se verán las diferentes actividades o excursiones que se pueden realizar en el destino o cerca de él, mostrando información tal cómo los días, el precio y en qué tipo de actividad se puede encasillar.

El contenido también incluirá una página de últimas búsquedas realizadas por los usuarios (sólo usuarios registrados). En ella se mostrarán las 10 últimas búsquedas realizadas por el usuario, mostrándose los datos del formulario que se envió para encontrar destinos. Desde esta parte se puede acceder directamente a la búsqueda y volver a tener los destinos que ya se vieron en la búsqueda anterior.

Existirá también un apartado de destinos aleatorios para los más indecisos donde no hay que introducir ningún dato. Estos destinos se generarán de un modo aleatorio cada vez que un usuario pulse el botón.

2.3.3 Funciones

En este punto se describen las funciones y servicios que ofrecerá el sitio web *recomendador* de destinos. Algunos de ellos ya se han mencionado en puntos anteriores.

Para mejorar la comprensión se han clasificado dependiendo del tipo de usuario que accede al sitio web, dependiendo del nivel de accesibilidad que le pertenezca.

Las funciones se pueden contemplar en el Anexo A, al final del documento.

3 Análisis

Este capítulo tratará de hacer una unión entre el punto anterior de los requisitos marcados para el sitio web y la futura implementación que haremos.

Se intentará traducir las necesidades y especificaciones del sitio web a un lenguaje que nos facilite la implementación del sistema. Para ello crearemos diferentes modelos y diferentes flujos de información según las opciones elegidas por el usuario.

Estos modelos y flujos de información nos permitirán comprender mejor desde y hacia donde debe moverse la información de la que disponemos, lo que nos facilitará la implementación así como ampliaciones o mejoras en caso de que fuera así.

Para definir estos modelos nos ayudaremos de UML, el lenguaje unificado de modelado. Aunque UML define una gran cantidad diferente de diagramas, en nuestro caso nos centraremos en los siguientes dos tipos de diagramas:

- Diagrama de clases
- Diagrama de casos de usos

Cada uno de estos diagramas cubre una parte diferente del análisis de la aplicación lo que nos permitirá cubrir distintos aspectos del desarrollo del sitio.

3.1 Diagrama de clases

Este tipo de diagrama nos describe la estructura de un sistema mostrando las entidades, atributos y las relaciones existentes entre ellos. Estos diagramas se utilizan durante el análisis y diseño de aplicaciones o sitios web para tener una idea conceptual de la información que manejará la aplicación, por dónde se ha de mover, los componentes necesarios para el correcto funcionamiento y la relación entre ellos.

Este diagrama nos permite conocer que información manejará la aplicación y qué podremos obtener de ella a grandes rasgos.

En la ilustración 5 podemos ver el diagrama de clases inicial.

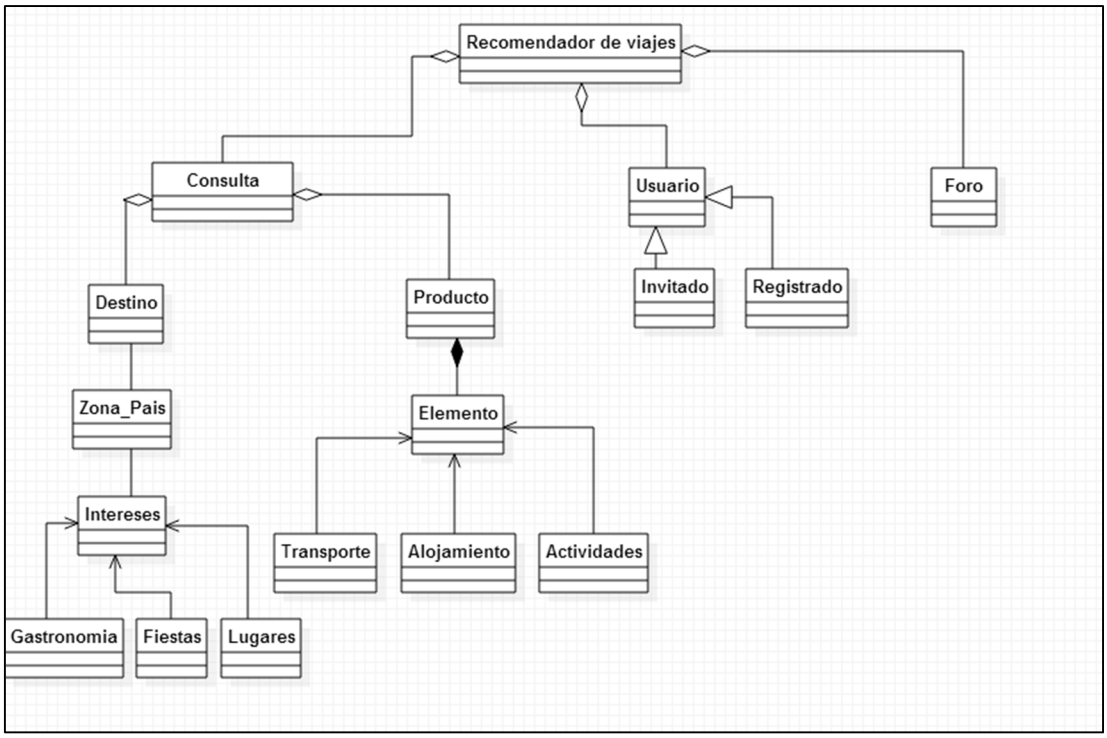

Ilustración 5. Diagrama de clases inicial

Después de revisar el diseño anterior, convenimos que era mejor cambiar la estructura de entidades para aligerar el diseño y facilitar las consultas. La ilustración 6 muestra el diseño final del diagrama de clases.

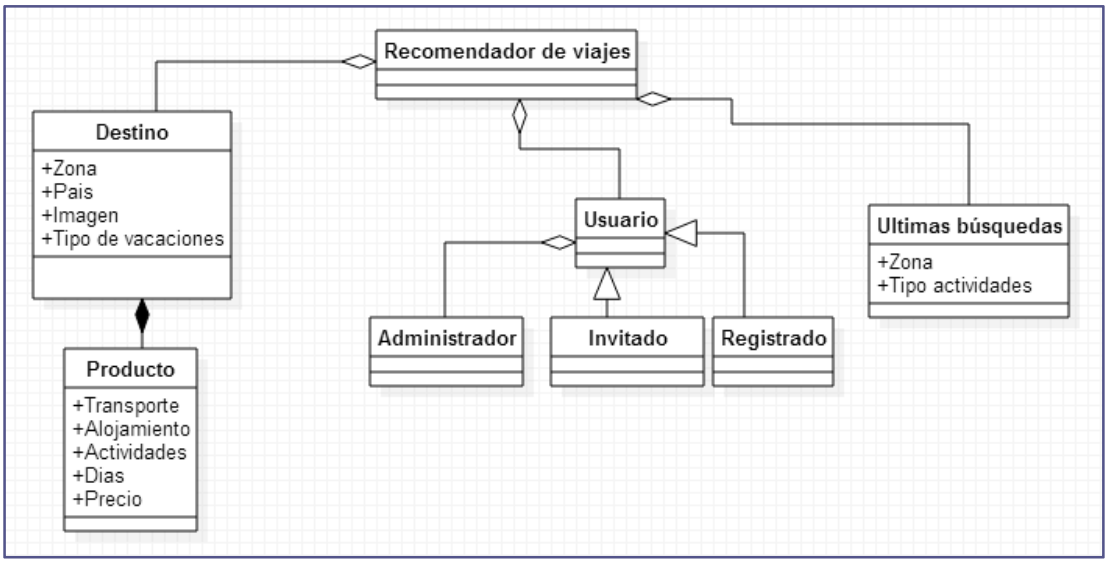

Ilustración 6. Diagrama de clases final

En el diagrama final nos encontramos cuatro entidades importantes para el sitio web. El propio *recomendador* de viajes que actúa de entidad madre de la que todos dependen.

Destino: Esta es la entidad que recoge todos los destinos disponibles en nuestro sistema. Es la entidad principal de nuestro sitio web ya que tiene la información más importante que se puede ver en él.

Producto: Esta entidad tiene una relación directa con destino. Cada uno de los destinos puede tener entre 0 y muchos productos que ofrecer. Cuando elegimos un destino, podemos ver todo lo que se ofrece en el mismo.

Usuario: En esta clase podemos registrarnos como usuarios o loguearnos en el sistema como administradores o usuarios comunes.

Últimas búsquedas: En esta entidad lo que hacemos es unir el usuario con las búsquedas realizadas, pudiendo mostrárselas y permitiendo nuevas búsquedas desde ahí mismo.

3.2 Diagramas de casos de uso

El diagrama de caso de uso es usado para identificar los elementos primarios y los procesos que forman el sistema. Los elementos primarios son llamados "actores" y los procesos "casos de uso". Los diagramas de casos de uso muestran qué actores interactúan con cada caso de uso.

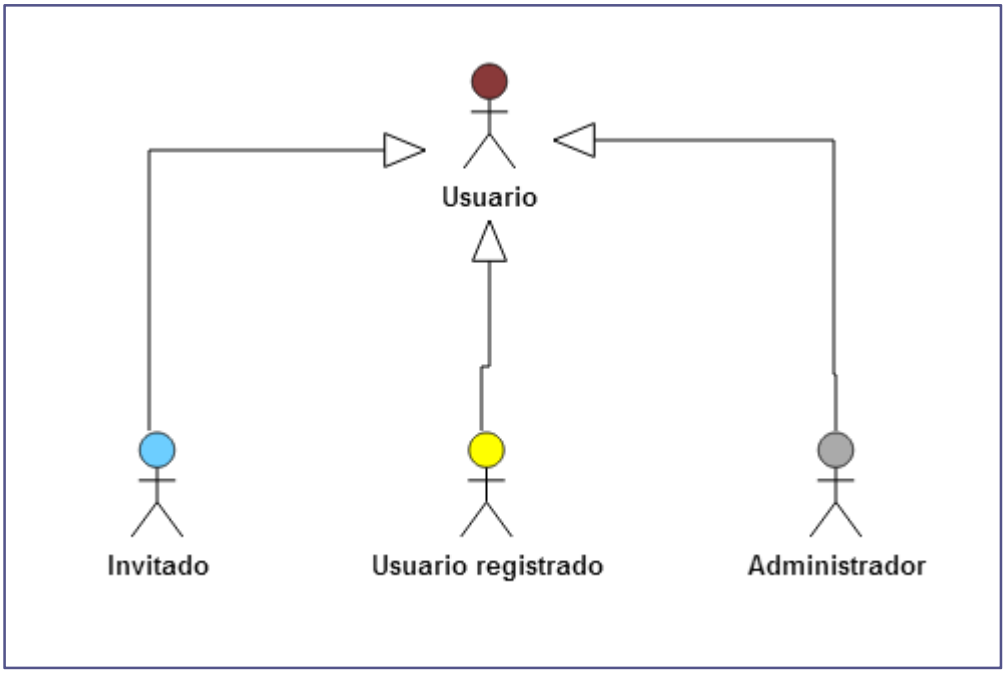

Ilustración 7. Actores del sistema

La ilustración 7 nos muestra los distintos actores que tendrán acceso al sitio web.

A continuación describiremos las acciones que puede realizar cada uno de ellos.

Usuario: El actor principal usuario puede realizar la búsqueda de destinos y los destinos aleatorios. Además también puede registrarse en el sistema. El resto de actores tendrán unas características propias que sólo podrán realizar ellos. En este caso, todas las acciones que pueda realizar el actor usuario, también las podrá realizar el actor invitado.

En la ilustración 8 podemos ver los casos de uso del actor "usuario".

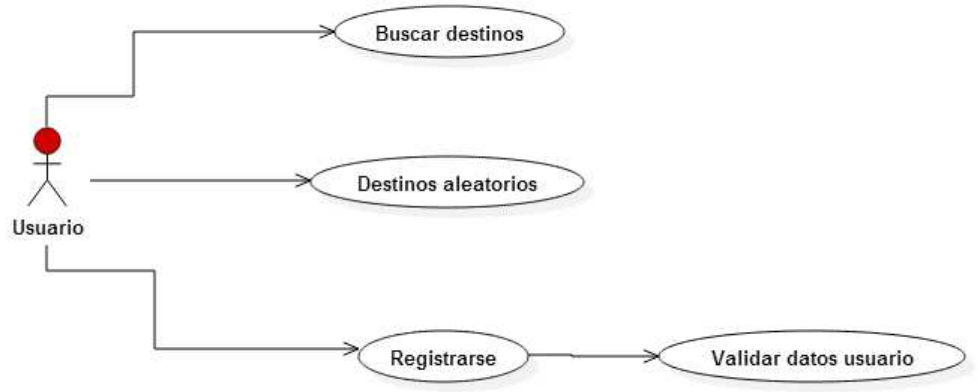

Ilustración 8. Casos de uso del actor usuario

Usuario registrado: El actor usuario registrado es un usuario invitado que se ha registrado en el sistema. Las acciones permitidas a un usuario registrado son:

- *Loguearse* en el sistema
- Guardar todas las búsquedas realizadas
- Ver las últimas búsquedas realizadas y relanzarlas

En la siguiente ilustración, la 9, podemos ver el diagrama de casos de uso del actor usuario registrado.

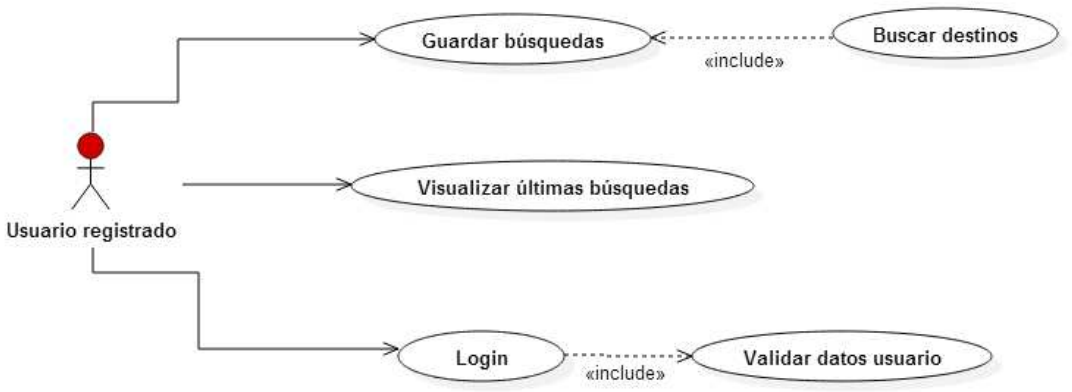

Ilustración 9. Casos de uso de usuario registrado

Administrador: El actor administrador puede además de todas las acciones anteriores, alguna más que solo su condición de administrador del sistema le permite.

La ilustración 10 nos muestra su diagrama de casos de uso.

Ilustración 10. Casos de uso de actor administrador

4 Diseño

4.1 Introducción

En nuestro sitio web se ha optado por un modelo de programación por capas. Las capas que contiene nuestro diseño son la capa de presentación, la de negocio y la de datos. Este modelo propone la separación de la lógica de negocio de la de diseño o la de datos.

Optar por este modelo nos permite tener claramente separados las tres funcionalidades diferentes. Esto nos permite ahorrarnos inconvenientes en cuanto a escalabilidad, disponibilidad y seguridad del sistema, tanto como para futuras mejoras o ampliaciones, permitiendo un avance mucho más rápido en el conocimiento de la aplicación.

4.2 Arquitectura del sistema

La arquitectura de tres capas por la que hemos optado en nuestro sitio web separa la implementación y diseño de la aplicación en 3 capas o niveles. Por una parte tendremos la capa donde se gestiona la información del sistema, es decir la base de datos. Seguidamente tendremos la lógica de negocio y por último, tenemos la última capa, la interfaz gráfica o capa de presentación.

A continuación un breve resumen de lo que contiene cada una de las capas.

- **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

- **Capa de negocio:** es donde residen las funciones que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

- **Capa de presentación:** la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de

proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

La ilustración 11 nos muestra un diagrama de la arquitectura de tres capas.

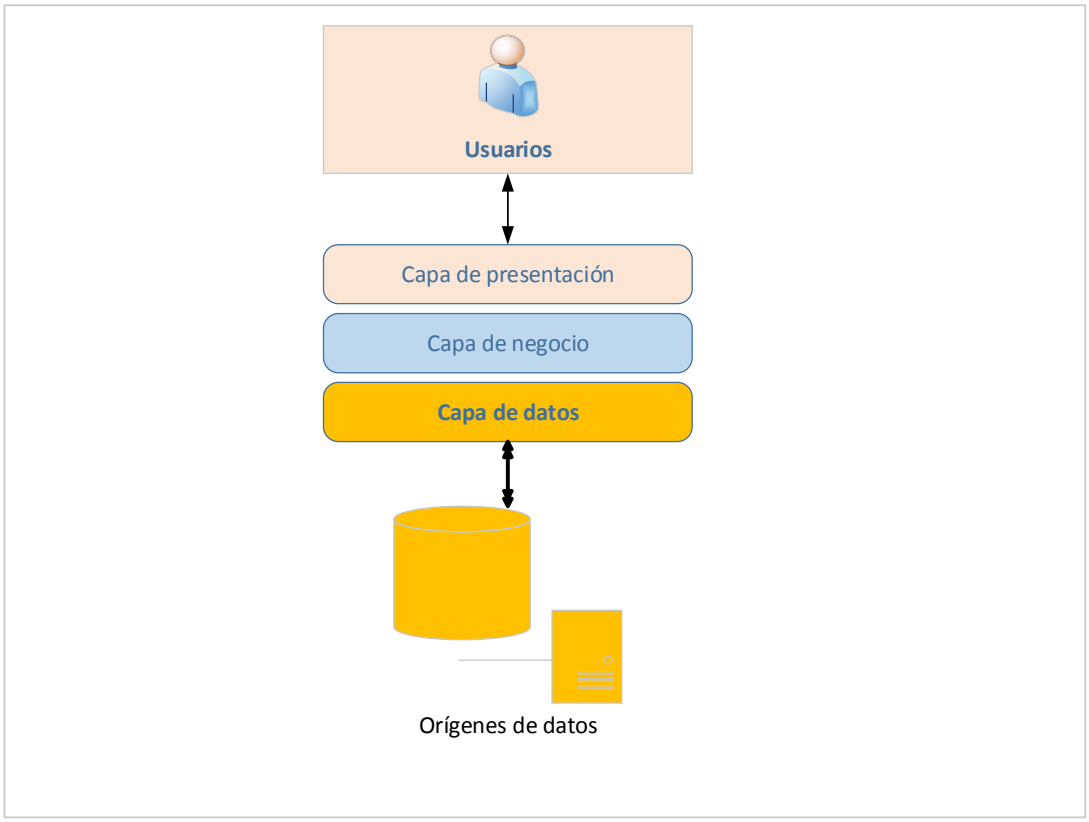

Ilustración 11. Arquitectura de tres capas.

Como ventajas de esta arquitectura destacaremos las siguientes:

- Las llamadas de la interfaz de usuario en la estación de trabajo al servidor de capa intermedia son más flexibles que el diseño de dos capas, ya que la estación solo necesita transferir parámetros a la capa intermedia.

- Con la arquitectura de tres capas la interfaz del cliente no es requerida para comprender o comunicarse con el receptor de los datos, por lo tanto, esa estructura de datos puede ser modificada sin cambiar la interfaz de usuario.

- El código de la capa de negocio puede ser reutilizado por otras funciones si está diseñado en formato modular.

- La separación de roles en tres capas hace más fácil reemplazar o modificar una capa sin afectar a los módulos restantes.

4.2.1 Capa de presentación

Como ya hemos comentado en el punto anterior esta capa se encarga de proveer una interfaz entre el sistema y el usuario. Esta capa es la responsable de la comunicación de información al usuario por parte del sistema y viceversa, manteniendo ésta a su vez una comunicación exclusiva con la capa de negocio.

Este apartado va a describir el diseño elegido para la capa de presentación del sitio web del *recomendador*. El sitio web dispone de una interfaz sencilla y funcional para que el usuario pueda actuar con el sistema de una forma fácil y amigable.

Para el menú inicial se ha creado un HTML desde cero, utilizando elementos jQuery para darle dinamismo y usabilidad.

La página inicial contiene los siguientes elementos:

Ilustración 12. Cabecera del sitio web.

Una cabecera (ilustración 12) donde según en la página del sitio web que nos encontremos, nos mostrará un título u otro. En el caso de la página inicial, tendremos el título del sitio web.

Ilustración 13. Mensaje de bienvenida.

Después del título nos encontramos con el mensaje de bienvenida que permite saber al usuario si se encuentra registrado en ese momento o solo

está accediendo como invitado al sitio web. Lo podemos ver en la ilustración 13.

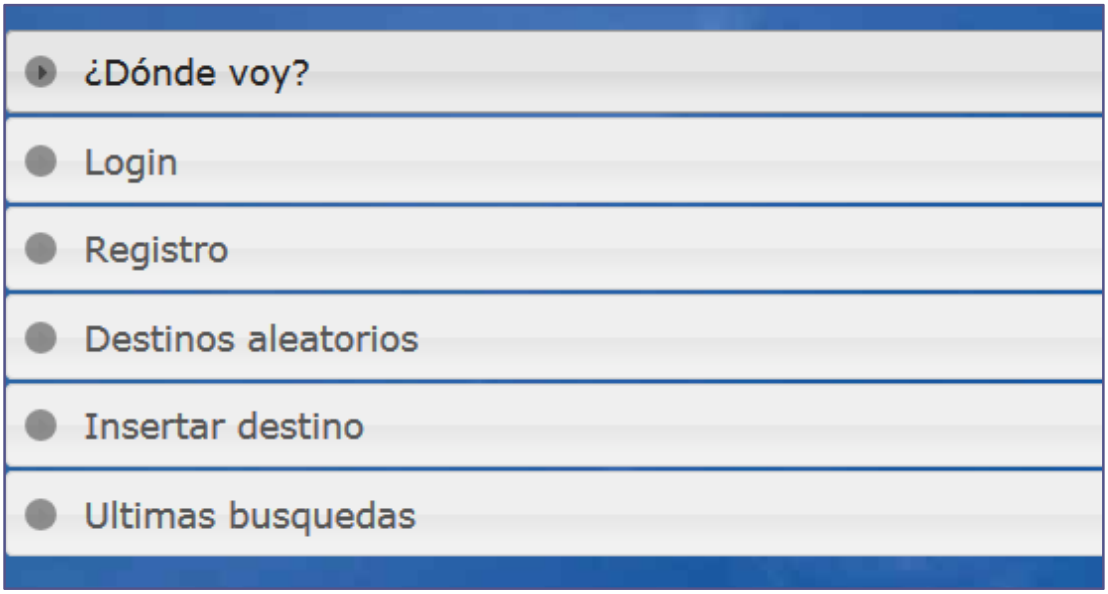

Ilustración 14. Menú de la página inicial.

El menú principal se presenta en un acordeón. Se puede contemplar en las ilustración 14. Desde el tenemos acceso a todas las funciones que se permiten en el sitio web. Esto nos permite tener un menú agrupado y sencillo que ayuda al usuario a conocer en cada momento qué funciones puede ejecutar.

El diseño escogido para mostrar los datos obtenidos de la base de datos está basado en una aplicación móvil[1] donde se utiliza jQuery Mobile con PhoneGap. De esta aplicación móvil hemos reutilizado el diseño para las listas de destinos y el destino elegido.

En la ilustración 15 tenemos el ejemplo de diseño de la aplicación módelo.

[1] Aplicación móvil Cristophe Coenraets

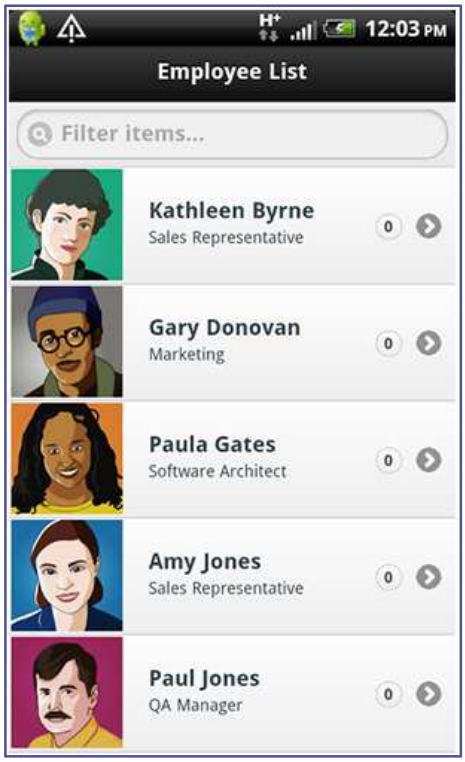

Ilustración 15. Ejemplo de diseño1.

A partir de este diseño, el sitio web mostrará un diseño como el presentado en la ilustración 16:

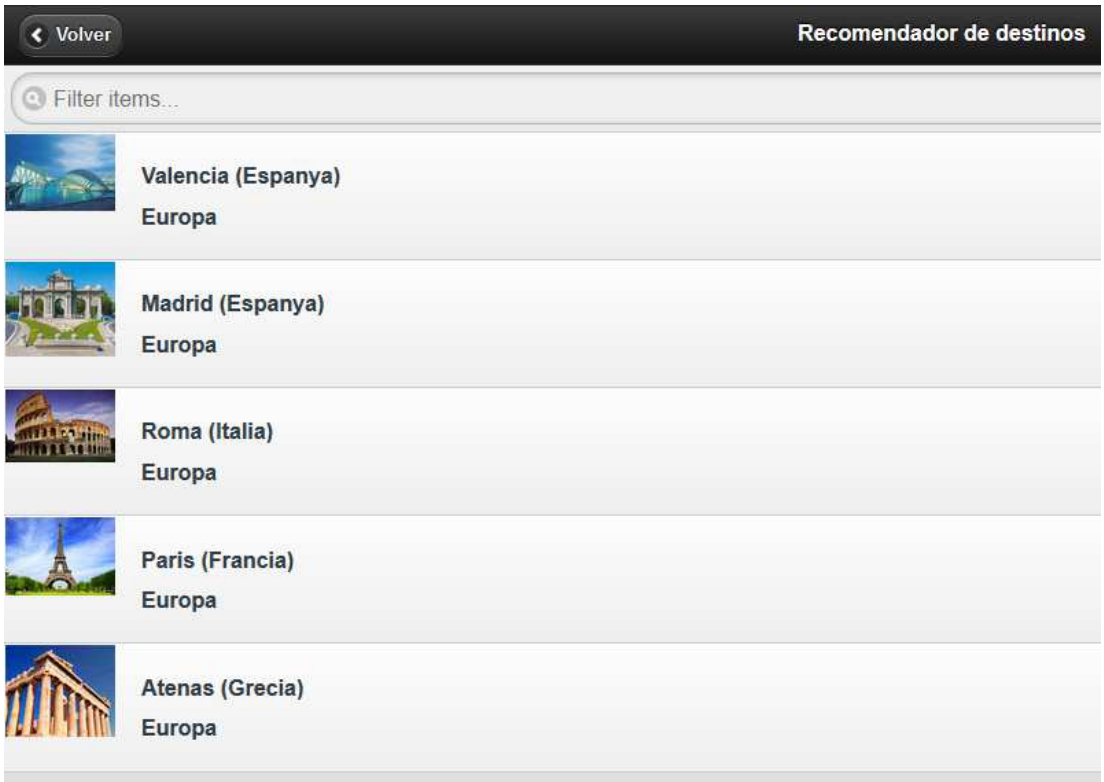

Ilustración 16. Diseño de lista de destinos.

Tenemos la cabecera que tendremos en todas las páginas del sitio web, con el botón Volver, que nos permitirá desde cualquier página volver a la página anterior, y la lista de los destinos seleccionados.

El diseño para el destino elegido, lo hemos basado en el diseño móvil para cada uno de los empleados. Aquí, ilustración 17, vemos el diseño modelo.

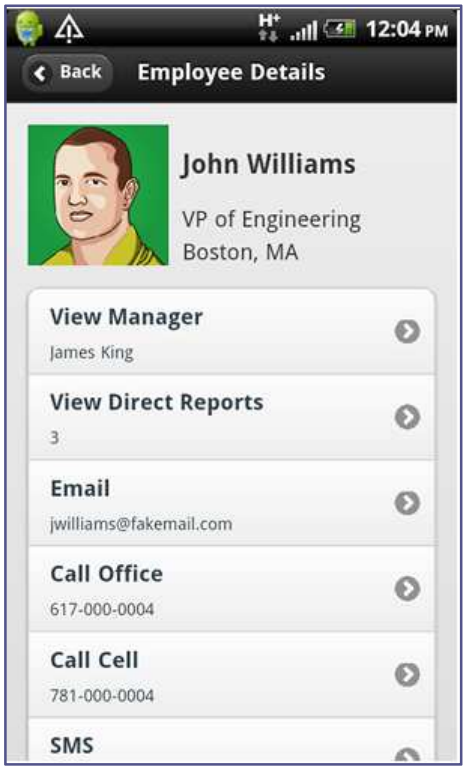

Ilustración 17. Diseño aplicación móvil.

Este diseño lo hemos adaptado a nuestras necesidades y así nos queda nuestra página para el diseño elegido. Ilustración 18.

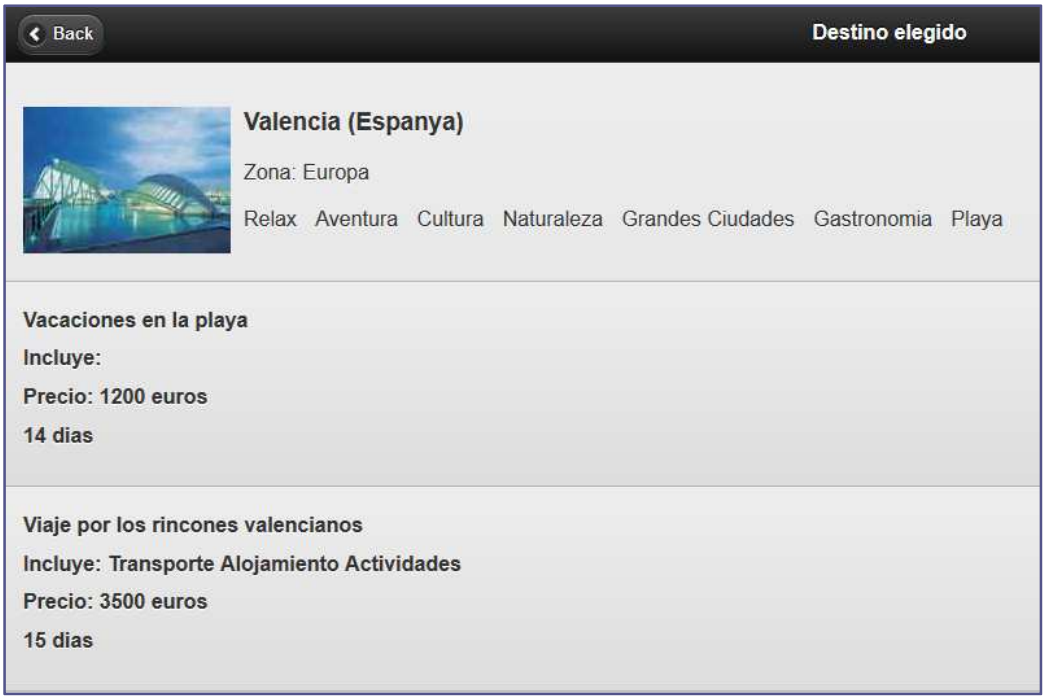

Ilustración 18.Diseño de la página Destino elegido

4.2.2 Capa de negocio

Esta capa contiene los procesos y funciones a realizar. La capa de negocio se encuentra en medio de las otras dos capas y esto hace que comparta la información con estas. Esto significa que debe recoger la información introducida por el usuario en la capa de presentación, realizar las funciones necesarias, preguntarle a la capa de datos por esa información y devolverá en forma que la capa de presentación la entienda y la pueda mostrar.

La capa de negocios en nuestro sitio web contiene las siguientes funciones:

- Administra las variables de sesión de todo el sitio web. Las inicializa, las guarda, las muestra o hace comprobaciones con la información recibida.
- Recoge los parámetros de las consultas enviadas desde la capa de presentación por método GET o POST.
- Realiza la conexión a la base de datos.
- Ejecutar sentencias SQL como consultas, inserciones o borrado de datos directamente en la base de datos.
- Empaqueta la información recibida de la base de datos en arrays.
- Hace la gestión de errores del sitio.
- Programación de algoritmos para diferentes funciones.
- Comprobación y verificación de usuarios.

- Registro de nuevos usuarios en la base de datos.
- Eliminar las variables de sesión al terminar la sesión de usuario.
- Insertar nuevos registros en algunas tablas.

4.2.3 Capa de datos

Por último, en la capa de datos es donde se almacenan todos los datos. Desde aquí salen los datos que llegarán, a través de la capa de negocio, hasta el usuario a través de la capa de presentación.

Nosotros utilizaremos un modelo de base de datos SQL y podremos tanto acceder a él mediante el gestor de la base de datos como desde las propias funciones que nos ofrece el sitio web.

El modelo que mostramos a continuación describe de una manera gráfica la base de datos que se encuentra detrás de nuestro sitio web. En él se detalla la composición de la base de datos en sus tablas como su estructura.

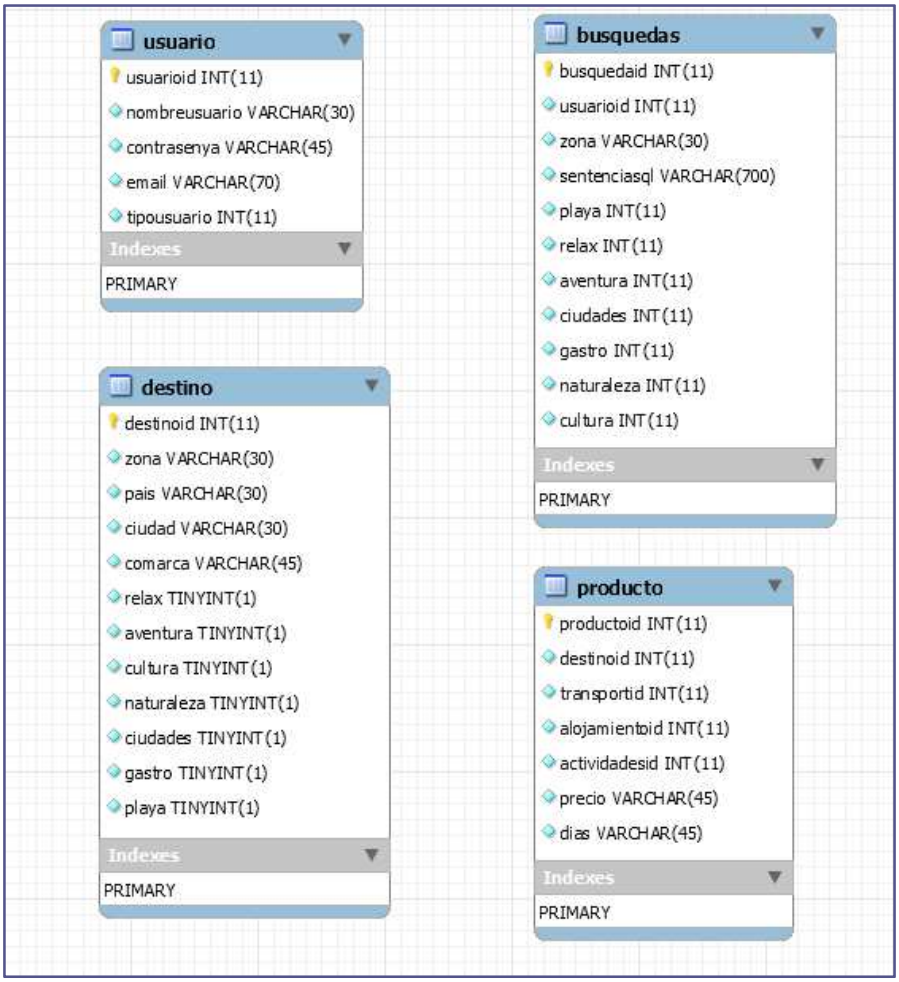

Ilustración 19. Estructura base de datos.

En la ilustración 19 se muestra la base de datos completa que se encuentra en nuestra capa de datos. Se trata de una base de datos relacional donde se pueden encontrar varios componentes, como pueden ser las entidades, las relaciones entre tablas y los atributos.

Podemos diferencias cuatro tablas con sus diferentes atributos que se relacionan entre sí.

Nos encontramos con la tabla “usuario” que será la encargada de guardar los usuarios que se registren en el sitio web. En esta misma tabla se realizará la comprobación para los usuarios que quieran *loguearse* o para saber si el usuario es administrador o solo un usuario registrado.

Tenemos también la tabla “destino” que corresponde a la información que tenemos de cada destino en sí. Esta tabla se relaciona con la tabla “producto” a través del destino id, permitiendo así saber que productos se ofrecen en cada destino. La relación es de 1 a muchos desde “destino” hasta “producto”.

La tabla destino es la entidad más importante de nuestro sitio web, ya que alrededor de ella gira toda la programación del sitio. Desde aquí se devuelven los destinos que el usuario ha requerido, la información sobre el destino elegido e incluso los destinos aleatorios.

La tabla “producto” se compone de los productos turísticos que ofrece cada destino. Como ya hemos comentado, tiene una relación con la tabla “destino”. De ella recogemos la información relativa a todos los productos que se pueden encontrar en ese destino como alguna información extra relativa a estos.

Por último, la tabla “búsquedas” que se relaciona con la tabla “usuario” a través del atributo usuarioid. A través de éste podemos recoger las búsquedas que ha realizado ese usuario mientras estaba logueado.

Para entender mejor el modelo relacional de la base de datos, se detalla a continuación el esquema lógico de la composición de las tablas que componen la base de datos de este sitio web.

Usuario: (usuarioid:integer not null, nombreusuario varchar(30) , contrasenya varchar(45), email varchar(70), tipousuario integer)

CP: {usuarioid}
VNN: {nombreusuario, contrasenya, email, tipousuario}

Destino: (destinoid: interger not null, zona varchar(30), país varchar(30), ciudad varchar(30), imagen varchar(45), relax: integer, aventura: integer,

cultura:integer, naturaleza: integer, ciudades: integer, gastro: integer, playa: integer)

CP: {destinoid}
VNN:{zona, país, ciudad, imagen, relax, aventura, cultura, naturaleza, ciudades, gastro, playa}

Producto: (productoid: integer not null, destinoid: integer not null, transporte integer, alojamiento integer, actividades integer, precio varchar(45), días varchar(45))

CP: {productoid}
CAj:{destinoid}
VNN:{tranporte, alojamiento, actividades, precio, dias}

Búsquedas: (búsquedaid: integer not null, usuarioid: integer not null, zona varchar(30), sentenciasql varchar(700), playa integer, relax integer, aventura integer, ciudades integer, naturaleza integer, gastro integer, cultura integer)

CP: {busquedaid}
CAj: {usuarioid}
VNN: {zona, sentenciasql, playa, relax, aventura, ciudades, naturaleza, gastro, cultura}

5 Implementación

En el siguiente apartado se van a describir las tecnologías, herramientas y los detalles en la implementación de la página web

5.1 Tecnologías

Para la construcción de este sitio web hemos utilizados diferentes tecnologías para las diferentes capas que lo componen.

En la capa de presentación se han utilizado el lenguaje HTML, así como CSS, para mantener el aspecto en las diferentes páginas del sitio web. También se han utilizado los lenguajes Javascript y Jquery, así como Jquery mobile, para la mejora de la navegabilidad y dinamismo de la página web.

En la parte de servidor se ha utilizado el lenguaje PHP junto con el motor de base de datos MySQL.

A partir de aquí vamos a profundizar un poco más en los diferentes lenguajes utilizados y en qué capa son importantes.

5.1.1 Capa de presentación

HTML

HTML, en inglés HyperText Markup Language, es un lenguaje de marcado para la elaboración de páginas web. Este lenguaje permite crear la estructura básica de la web y él código HTML, a su vez, definir el contenido de la página, ya sea como texto, imágenes o videos. El lenguaje HTML es un estándar de la W3C, una organización que se encarga de la estandarización de muchas de las tecnologías que tienen que ver con la web, ya sea referente a su escritura como a la interpretación que se hace del mismo. Es considerado el lenguaje web más importante ya que es estándar para la visualización de páginas web y todos los navegadores actuales lo han adoptado.

Este lenguaje se basa en las referencias. Esto significa que para añadir un elemento externo a la página web, lo que se hace es una referencia a la ubicación del elemento mediante texto. Por lo tanto, la página web solo contendrá texto que un navegador web deberá interpretar.

A día de hoy todos los navegadores entienden y son capaces de interpretar el código HTML en cualquiera de sus versiones, aunque esto cause que veamos

diferencias entre los diferentes navegadores y sus versiones a la hora de interpretar el código.

Al ser un lenguaje de referencias y de etiquetas, no se necesita de un compilador o un programa especial para su programación. Se pueden utilizar los editores de texto de cualquier sistema operativo, ya se Bloc de Notad de Windows, como Vi de Unix.

En nuestro caso el HTML nos sirve para la parte visual de nuestro sitio web. Lo utilizamos para crear una página inicial con las diferentes opciones que se le ofrecen al usuario y luego dos páginas con una estructura similar para mostrar la búsqueda de destinos y los destinos en sí.

CSS

CSS, en inglés Cascading Style Sheets o en español Hoja de estilos en cascada es un lenguaje utiliza para definir la parte visual de un documento estructurado escrito en HTML o XML. En este caso es también el W3C el responsable de estandarizar la especificación de las hojas de estilos tanto para los agentes de usuario como para los navegadores.

La idea principal del CSS es la de separar la estructura del documento de la presentación del mismo.

Las ventajas que nos supone la utilización de CSS en la web son, por un lado la reutilización de estilos para diferentes páginas sin tener que volver a reescribir el mismo estilo para diferentes elementos y la mejora de la accesibilidad al documento al poder prescindir de las tablas, así como de otros elementos de diseño, que ya están desfasados y que algunos navegadores no pueden interpretar.

JavaScript

JavaScrip es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de una navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque también existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en otras aplicaciones diferentes a la web, como por ejemplo en documentos PDF o Widgets de escritorio, es también significativo.

El lenguaje JavaScript se diseñó con una sintaxis similar al lenguaje de programación C, aunque adopta nombres y convenciones del lenguaje Java. Sin embargo, no existe relación entre Java y JavaScript y sus propósitos son diferentes.

Todos los navegadores modernos interpretan el código JavaScript que viene integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM o Modelo de Objetos del Documento.

JavaScript se ha utilizado tradicionalmente en páginas web para realizar operaciones en la aplicación cliente, sin acceso a funciones del servidor. Hoy en día es ampliamente utilizado para enviar y recibir información del servidor junto con la ayuda de otras tecnologías como AJAX. JavaScript se interpreta en la aplicación al mismo tiempo que las sentencias se van descargando junto con el código HTML.

En el sitio web utilizamos javascript para aquellas funciones en la que no es necesaria la intervención del servidor o para dar un poco de dinamismo a la web. Validar campos en formularios, expandir el menú acordeón de la página inicial o la transición entre las páginas de búsqueda son también tarea del javascript.

jQuery

jQuery no es una tecnología en sí, sino una biblioteca de JavaScript, creada por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es la biblioteca de JavaScript más utilizada, ya que permite cambiar el contenido de una página web sin necesidad de recargarla.

jQuery es software libre y código abierto, permitiendo su uso tanto en proyectos libres como privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.

La biblioteca jQuery es compatible con los siguientes navegadores: Mozilla Firefox 2.0+, Internet Explores 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

Como ya hemos leído, se trata de una biblioteca de Javascript por lo tanto, nos va a ayudar con la parte dinámica de la web. Con añadir la librería

code.jquery tenemos acceso a muchas funciones jquery que simplifican el proceso de programación e implementación de cualquier página web.

jQuery Mobile

jQuery Mobile es un framework optimizado para dispositivos táctiles que está siendo desarrollado por el equipo de proyecto de jQuery. El desarrollo se centra en la creación de un Framework compatible con la gran variedad de smartphones y tablets. El framework de jQuery Mobile es compatible con otros frameworks móviles y plataformas como PhoneGap y Worklight entre otros.

jQuery Mobile es compatible con la mayoría de navegadores actuales y también con las principales plataformas móviles como Android, IOS, Windows Phone, blackberry, WebOS y Symbian.

Con jQuery Mobile nos aseguramos que nuestro sitio web es visible y perfectamente utilizable por la casi totalidad de smartphones o tablets. Con él conseguimos un aspecto sencillo y claro para poder leer y tener una mayor usabilidad en dispositivos con pantalla táctil y una zona de acción reducida.

En nuestro caso podemos ver que las páginas en las que se muestran la búsqueda de destinos así como los destinos en sí, los campos y los enlaces mostrados son elementos grandes donde se puede hacer click con el dedo sin necesidad de utilizar un lápiz táctil o tener que ampliar la pantalla para conseguir nuestro objetivo.

5.1.2 Capa de negocio

PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente al documento HTML en lugar de llamar a un archivo externo que procesase los datos.

El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. Se puede usar en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Aunque fue creado por Rasmus Lerdorf, la implementación principal de PHP actualmente es producida por The PHP Group y sirve como el estándar para PHP al no haber una especificación formal. Está publicado bajo la PHP License a la que la Free Software Foundation considera como software libre.

El PHP es el encargado de todas las funciones que se realizan en nuestro sitio web. Al tener que recuperar los datos de la base de datos necesitamos un lenguaje para poder devolver los datos requeridos.

En este caso el PHP se encarga de lanzar las consultas SQL a la base de datos, devolver los resultados o simplemente insertar información del sitio web en la base de datos. Con el PHP nos aseguramos también mantener el usuario registrado durante toda la visita gracias a las variables de sesión.

Debido a la arquitectura de nuestro sitio web, nuestro documentos PHP devuelven arrays de datos codificados en JSON para que la capa de presentación los analice y los pueda mostrar correctamente.

5.1.3 Capa de datos

MySQL

MySQL es un sistema de administración de bases de datos (*DBMS*) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

La historia del MySQL se remite a principios de la décadade 1980. Programadores de IBM lo desarrollaron para contar con un código de programación que permitiera generar múltiples y extensas bases de datos para empresa y organizaciones de diferente tipo.

Una de las características más interesantes de MySQL es que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a diferentes necesidades y requerimientos.

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente.

La base de datos nos sirve como base de nuestro sitio web en todo momento. Tanto para el registro o el login de usuarios, como para devolver las búsquedas de destinos, o relacionar una búsqueda con un usuario determinado y así poder presentar las últimas búsquedas de ese usuario en pantalla.

5.2 Herramientas

En esta sección se describirán las herramientas utilizadas para la programación y creación de esta página web.

5.2.1 Notepad++

Notepad++ es un editor de texto y de código fuente, así como un reemplazo para el Bloc de notas, que soporta muchos lenguajes. Sólo funciona bajo Microsoft Windows y su uso está regulado por la licencia GPL.

Basado en el potente componente Scintilla, Notepad++ está escrito en C++ y usa la API Win32 y STL lo que asegura una alta velocidad de ejecución y un tamaño reducido del programa. La ilustración 20 muestra una captura de pantalla de la aplicación Notepad++.

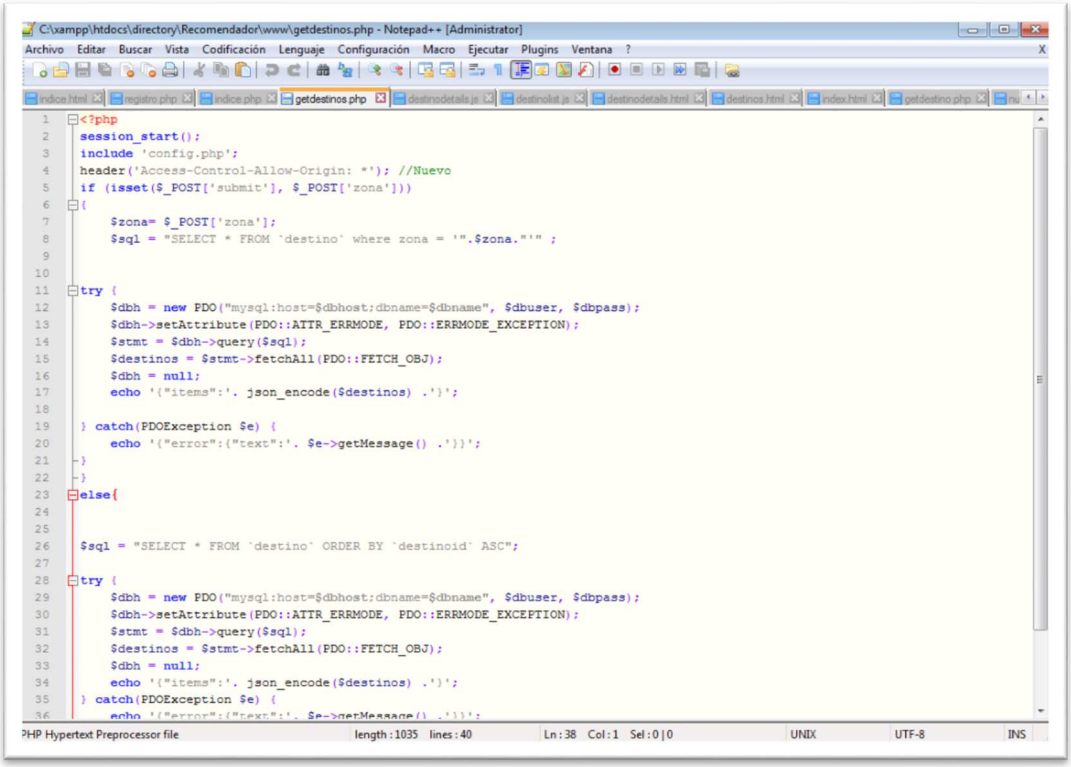

Ilustración 20. Captura Notepad++.

5.2.2 XAMPP Server

XAMPP es un servidor independiente de plataforma que consiste principalmente en el sistema de gestión de base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script, PHP y Perl.

El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

Se puede utilizar como servidor web al uso, pero en nuestro caso ha sido utilizado para el testeo en local de la página web. Los módulos FileZilla FTP Server y PHPMyadmin también han sido utilizados. En la ilustración siguiente, 21, vemos una captura del Control Panel de la aplicación XAMPP.

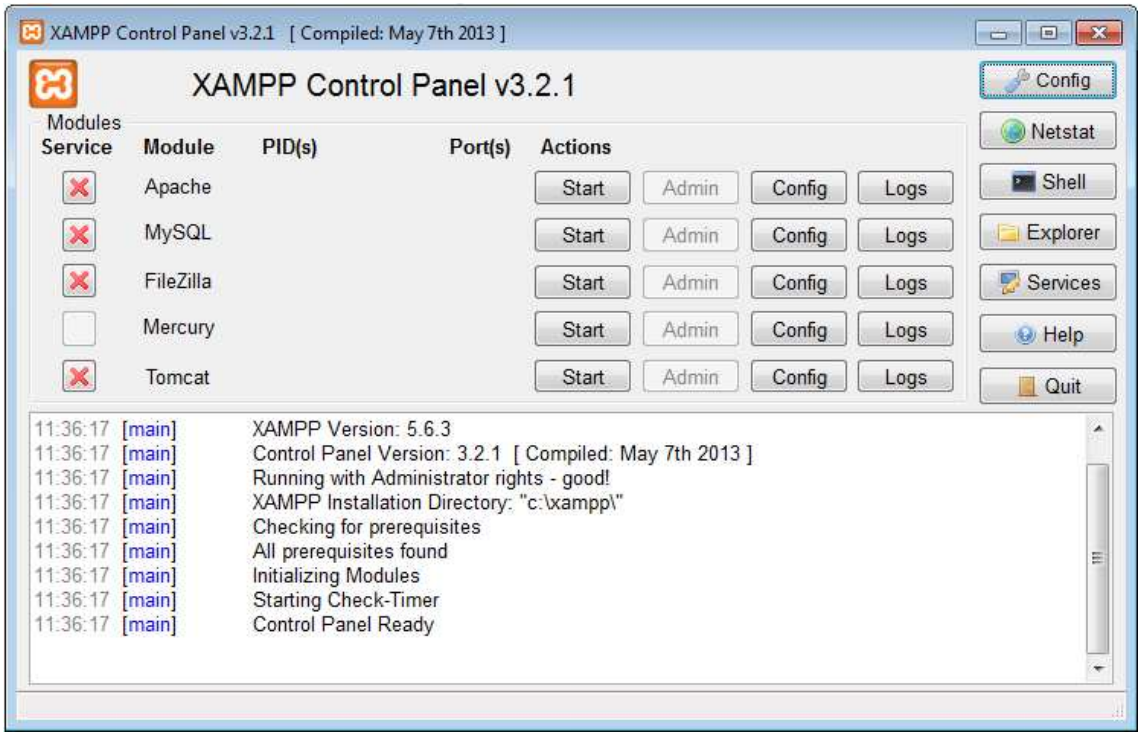

Ilustración 21. Captura XAMPP Control Panel.

5.2.3 Apache

Apache es un servidor de páginas web HTTP de código abierto para plataformas Unix, Windows, Mac OS X y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular HTTPd de la NCSA (creadores del navegador Mosaic), pero más tarde fue reescrito por completo. El servidor

Apache se desarrolla dentro del proyecto HTTP Server (httpd) de Apache Software Foundation.

La misión del servidor Apache es aceptar las peticiones de páginas que provienen de los visitantes que acceden a nuestro sitio web y gestionar su entrega o denegación, de acuerdo a las políticas de seguridad establecidas.

5.2.4 Adobe Photoshop

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incoporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente “taller de fotos”. Fue comercializado inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows.

En la ilustración 22 vemos una captura del programa de edición fotográfica Adobe Photoshop.

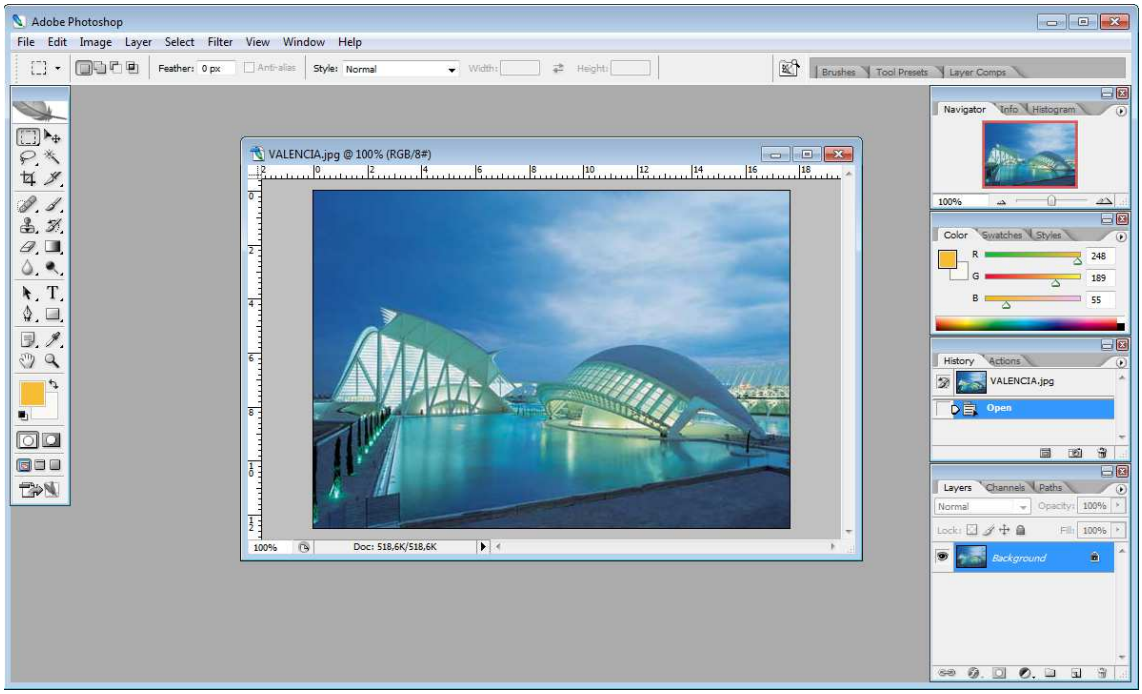

Ilustración 22. Captura de Adobe Photoshop.

5.3 Detalles de implementación

En este apartado se van a describir los pasos seguidos para la implementación de este sitio web.

5.3.1 Estructura de la página web

El sitio web está estructurado de forma que la misma plantilla nos sirva para diferentes secciones del sitio web. Al hacerlo de esta forma, nos ahorramos código y le damos mayor usabilidad al sitio web, proporcionándole al usuario una sensación de familiaridad con el sitio y con los diferentes escenarios.

Se ha dividido el sitio web en tres plantillas de interfaz diferentes según el contexto en el que nos encontremos. Tenemos un menú en la página principal, una página máster para mostrar la lista de destinos, tanto buscados como aleatorios, y las últimas búsquedas realizadas por el usuario. La última de las plantillas es aquella que nos sirve para presentar los detalles del destino y los tipos de actividades o productos que nos puede ofrecer.

Para ello hemos creado una página principal o índice donde se mostrará el menú y desde donde se realizarán todas las acciones posibles en el sitio web. En esta página encontramos un mensaje de bienvenida que nos sirve también como ayuda para saber si el usuario es un invitado o es un usuario previamente registrado en el sitio.

Después tenemos un menú en forma de acordeón donde se encuentran todas las funciones disponibles en el sitio. ¿Dónde voy?, Login, Registro, Destinos aleatorios, Insertar destino y Últimas búsquedas. Esto lo podemos observar en la ilustración 23.

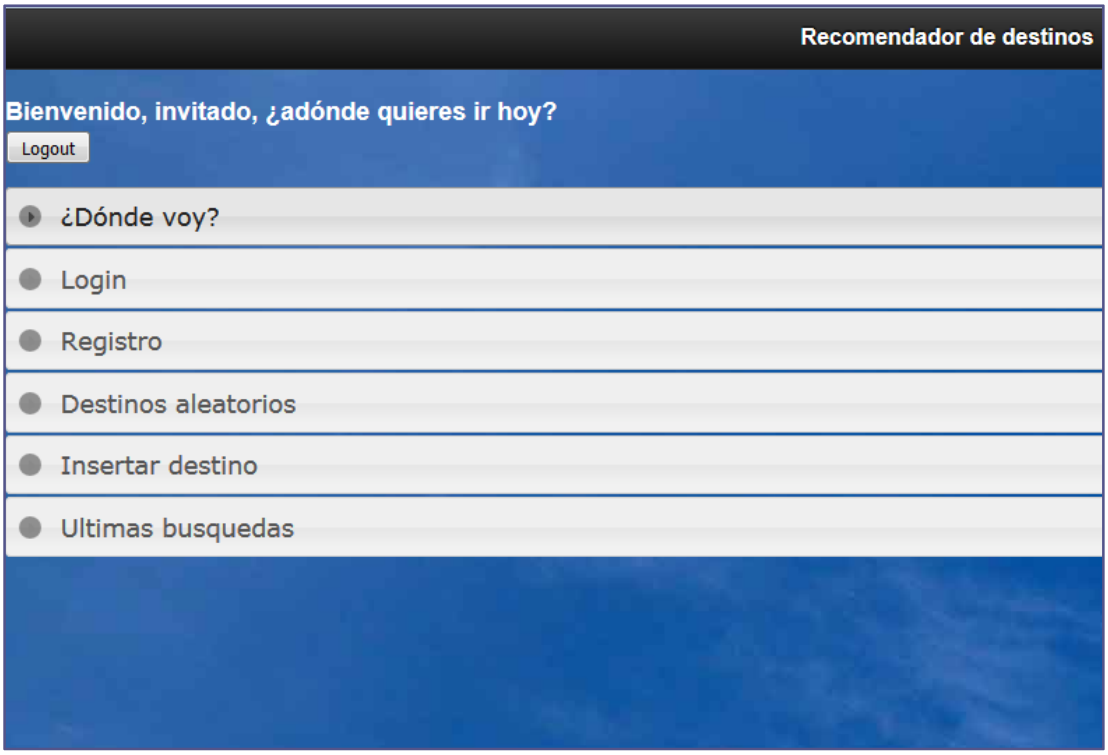

Ilustración 23. Página de inicio.

Las funciones que se realizan en *Login*, Registro o insertar nuevo destino no necesitan de interfaz gráfica para su ejecución y por lo tanto quedan fuera de esta descripción.

El resto de funciones del menú (¿Dónde voy?, Destinos aleatorios y Últimas búsquedas) nos llevan a una interfaz que es la misma para las tres. Se trata de la página master que se utiliza en jQuery Mobile para cargar todas las interfaces y scripts que se utilizarán más tarde. Desde esta página se puede acceder a las páginas siguientes, pero sin cargar esta página no cargará ninguna otra.

La plantilla máster contiene una cabecera, el botón para volver atrás y una lista que se cargará con los diferentes scripts que hemos ejecutaremos al cargar la página.

El código HTML de la plantilla máster lo podemos ver en la siguiente imagen (ilustración 24).

```
<body>

<div id="destinoListPage" data-role="page" data-add-back-btn="true">

    <div data-role="header" data-position="fixed">
        <h1>Recomendador de destinos</h1>
        <a data-theme="a" href="indice.html"
        class="ui-btn-left ui-btn ui-btn-icon-left ui-btn-corner-all ui-shadow ui-btn-up-a"
        data-rel="back"
        data-icon="arrow-l">
        <span class="ui-btn-inner ui-btn-corner-all">
        <span class="ui-btn-text">Volver</span>
        <span class="ui-icon ui-icon-arrow-l ui-icon-shadow"></span></span></a>
    </div>

    <div data-role="content">
        <ul id="destinoList" data-role="listview" data-filter="true"></ul>
    </div>

</div>

<script src="js/jquery.js"></script>
<script src="js/jquery.mobile-1.0rc1.min.js"></script>
<script src="js/destinolist.js"></script>
<script src="js/destinodetails.js"></script>
<script src="js/reportlist.js"></script>

</body>
```

Ilustración 24. Plantilla máster en HTML para la lista de destinos.

La página máster tiene un diseño como el siguiente (ilustración 25).

Ilustración 25. Diseño de la lista de destinos.

A la tercera plantilla se accede desde la página anterior cuando pinchamos en alguno de los destinos para conocer más en detalle que nos ofrece. El diseño lo vemos en la ilustración 26.

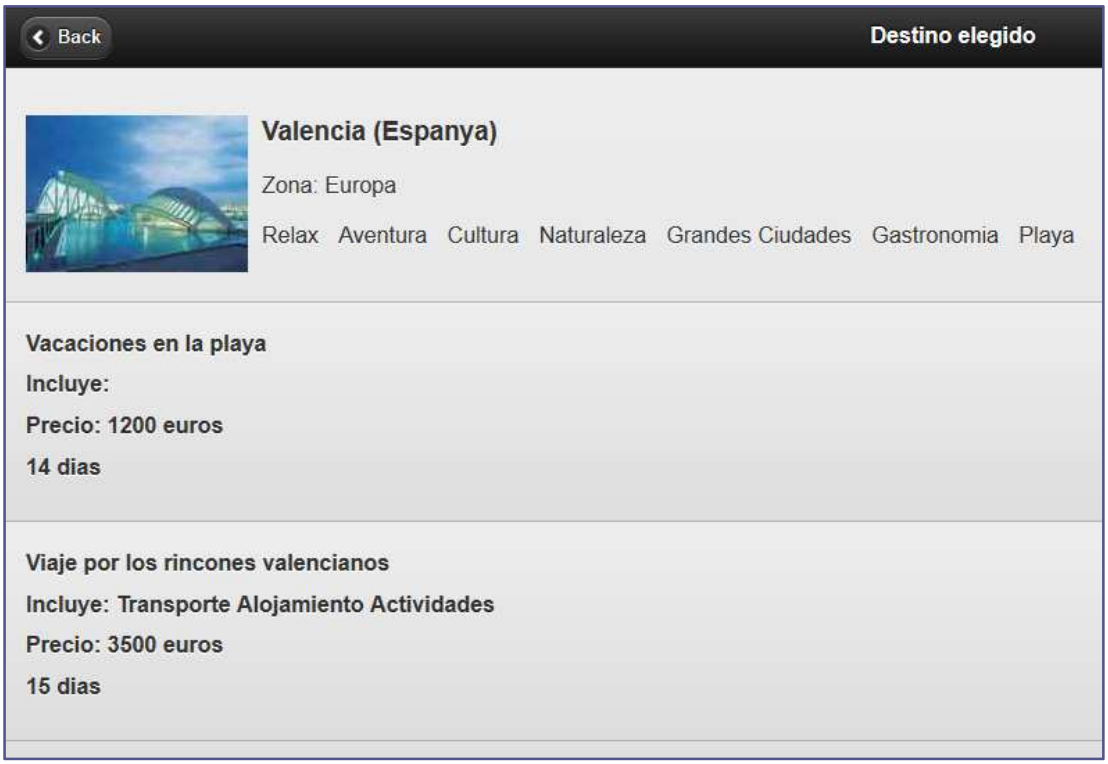

Ilustración 26. Diseño página destino elegido.

La plantilla del detalle del destino está compuesta por los datos del destino elegido más una lista que no sabemos cuántos elementos tendrá, por eso haremos una carga dinámica según el destino (ilustración 27).

```
<body>
<div id="detailsPage" data-role="page" data-add-back-btn="true">

<div data-role="header">
    <h1>Destino elegido</h1>
</div>

<div data-role="content">
    <div>
    <img id="destinoPic"/>
    <div id="destinoDetails">
    <h3 id="ciudad"></h3>
    <p id="zona"></p>
     <ul>
     <li style="display:inline;padding-right:10px" id="relax"></li>
     <li style="display:inline;padding-right:10px" id="aventura"></li>
     <li style="display:inline;padding-right:10px" id="cultura"></li>
     <li style="display:inline;padding-right:10px" id="naturaleza"></li>
     <li style="display:inline;padding-right:10px" id="ciudades"></li>
     <li style="display:inline;padding-right:10px" id="gastro"></li>
     <li style="display:inline;padding-right:10px" id="playa"></li>
     </ul>

 </div>
 <p><br></p>
 </div>
 <div>
   <ul id="productList" data-role="listview"></ul>
</div>
</div>

</div>
</body>
```

Ilustración 27. Plantilla HTML para el destino elegido.

Todas las plantillas excepto la de la página de inicio tienen un botón Volver para siempre poder volver a la página anterior. En el caso del destino seleccionado, el botón vuelve a la página de la búsqueda, pero en el caso de las últimas búsquedas el botón nos devuelve a la página de inicio.

5.3.2 Todos los usuarios

Todos los usuarios (invitados, registrados y administradores) tienen la posibilidad de buscar su destino ideal. Esto se hace a través del menú de la página inicial, en el área ¿Dónde voy?

Ahí nos aparece un formulario en el que el usuario ha de rellenar los campos referidos al destino que está buscando. Lo podemos ver en la ilustración 28.

En este caso tenemos la zona, que es la zona geográfica del mundo dónde el usuario quiere realizar su búsqueda y luego tenemos una serie de tipos de vacaciones que el usuario puede escoger, que son Playa, Aventura, Grandes ciudades, Gastronomía, Relax, Cultura y Naturaleza.

Ilustración 28. Formulario búsqueda de destinos.

La zona es un campo obligatorio que hay que rellenar, por eso hay una opción marcada por defecto (Europa) que nos obliga a elegir una zona del mundo para que la consulta no se extienda demasiado. Los tipos de vacaciones son opcionales, si no se elige ninguno de ellos, se mostrarán todos los destinos pertenecientes a la zona elegida. Si se eligen varios de ellos se devolverán solo los resultados de los destinos que cumplan con todos los requisitos marcados por el usuario.

La consulta que le enviamos al servidor pasa primero por el filtro del javascript, donde se recogen los valores introducidos por el usuario a través de la URL del sitio web. Una vez recogidos los valores y formada la URL definitiva, llamamos al PHP getdestinos donde devolveremos un JSON con la información que le pasaremos a la función displayDestinoList que formará el HTML que mostraremos por pantalla finalmente.

En la siguiente ilustración (29) se muestra un ejemplo de código javascript para la carga dinámica de destinos.

```
$('#destinoListPage').bind('pageinit', function(event) {
    var zona = getUrlVars()["zona"];
    var playa = getUrlVars()["playa"];
    var relax = getUrlVars()["relax"];
    var aventura = getUrlVars()["aventura"];
    var ciudades = getUrlVars()["ciudades"];
    var gastro = getUrlVars()["gastro"];
    var naturaleza = getUrlVars()["naturaleza"];
    var cultura = getUrlVars()["cultura"];
    var sentenciasql = getUrlVars()["sentenciasql"];

    if (zona != undefined)
    {
        var consulta = "";
        if (playa == 1){ consulta += "&playa="+playa;}
        if (relax == 1) {consulta += "&relax="+relax;}
        if (aventura == 1) {consulta += "&aventura="+aventura;}
        if (ciudades == 1) {consulta += "&ciudades="+ciudades;}
        if (gastro == 1) {consulta += "&gastro="+gastro;}
        if (naturaleza == 1) {consulta += "&naturaleza="+naturaleza;}
        if (cultura == 1) {consulta += "&cultura="+cultura;}


        $.getJSON(serviceURL + 'getdestinos.php?zona='+zona+''+consulta, displayDestinoList);

    }
```

Ilustración 29. Ejemplo de código destinos.js

En getdestinos.php consultamos las variables GET que hemos pasado por la URL y generamos la consulta SQL que enviaremos a la base de datos para recibir los destinos (ilustración 30).

```
if (isset($_GET['zona']))
{
    $consultasql= "";
    if (isset($_GET['playa'])){$playa = 1; $consultasql .= " and playa = ".$playa;}
    else{ $playa = 0;}
    if (isset($_GET['relax'])){$relax = 1; $consultasql .= " and relax = ".$relax;}
    else{$relax = 0;    }
    if (isset($_GET['aventura'])){$aventura = 1; $consultasql .= " and aventura = ".$aventura;}
    else{$aventura = 0; }
    if (isset($_GET['ciudades'])){$ciudades = 1; $consultasql .= " and ciudades = ".$ciudades;}
    else{$ciudades = 0;}
    if (isset($_GET['gastro'])){$gastro = 1; $consultasql .= " and gastro = ".$gastro;}
    else{$gastro = 0;}
    if (isset($_GET['naturaleza'])){$naturaleza = 1; $consultasql .= " and naturaleza = ".$naturaleza;}
    else{$naturaleza = 0;}
    if (isset($_GET['cultura'])){$cultura = 1; $consultasql .= " and cultura = ".$cultura;}
    else{$cultura = 0;}

    $zona= $_GET['zona'];

    $sql = "SELECT * FROM `destino` where zona = '".$zona."'".$consultasql;
```

Ilustración 30. Ejemplo de código PHP Getdestinos.

En la ilustración 31 mostramos cómo abrimos la conexión con la base de datos y con la consulta SQL previamente guardada, devolvemos los valores de los destinos en una lista JSON, que devolveremos al javascript para que se puede crear el HTML.

```php
try {
$dbh = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$stmt = $dbh->query($sql);
$destinos = $stmt->fetchAll(PDO::FETCH_OBJ);
$dbh = null;
echo '{"items":'. json_encode($destinos) .'}';

if (isset($_SESSION['user_id']))
{

}
catch(PDOException $e) {
echo '{"error":{"text":'. $e->getMessage() .'}}';
}
```

Ilustración 31. Ejemplo de código destinos.js

Finalmente, llamamos a la función displayDestinoList (ilustración 32) que con la lista JSON que habíamos recogido del PHP forma la lista con los destinos que se mostrarán en el HTML.

```javascript
function displayDestinoList(data) {
    var destinos = data.items;
    $('#destinoList li').remove();
        destinos = data.items;
        $.each(destinos, function(index, destino) {
            $('#destinoList').append('<li><a href="destinodetails.html?id=' + destino.destinoid + '">' +
                    '<img src="pics/' + destino.imagen + '"/>' +
                    '<h4>' + destino.ciudad + ' ('+ destino.pais +')</h4>' +
                    '<h3>' + destino.zona + '</h3>' +
                    '</a></li>');
        });
        $('#destinoList').listview('refresh');
}
```

Ilustración 32. Carga de datos en javascript.

Dándonos como resultado la siguiente vista en el sitio web que podemos observar en la ilustración 33.

Ilustración 33. Resultado de la búsqueda de destinos.

Cada uno de los elementos de la lista es a su vez un link a la descripción pormenorizada de cada uno de los destinos, por eso si pinchamos en uno de los destinos, obtendremos las actividades que podemos realizar en dicho destino.

Para ello, se hará una llamada a destinodestails.html?id=X siendo el id el número del destino elegido en nuestra base de datos.

Al final, obtendremos una vista como la mostrada en la ilustración 34.

Ilustración 34. Vista del destino elegido.

En este caso llamamos primero al HTML que cargará el javascript que viene incrustado en la página. Lo primero que haremos será recoger el ID que le hemos pasado para lanzar nuestra petición al PHP (ilustración 35).

```
$('#detailsPage').live('pageshow', function(event) {
    var id = getUrlVars()["id"];
    $.getJSON(serviceURL + 'getdestino.php?id='+id, displayDestino);
});
```

Ilustración 35. Recoger parámetros GET.

El PHP recoge la variable ID y lanza dos consultas distintas a la base de datos. Una de ellas para recoger la información del destino y la otra para recoger los diferentes tipos de vacaciones (producto) que ofrece ese destino.

Una vez que hemos lanzado las consultas a la base de datos, recogemos los valores y formamos un array donde irán tanto los destinos como los productos para devolverlo al javascript que se encargará de montar el HTML final. Este proceso se puede observar en la ilustración 36.

```
include 'config.php';
header('Access-Control-Allow-Origin: *'); //Nuevo

$sql = "SELECT * FROM  destino where destinoid = :id ";

$sql2 = "SELECT * FROM `producto` where destinoid = :id ";
$id = $_GET['id'];

try {
    $arr = array();
    $dbh = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $dbh->prepare($sql);
    $stmt->bindParam(":id", $id);
    $stmt->execute();
    $destino = $stmt->fetchObject();

    $stmt2= $dbh->prepare($sql2);
    $stmt2->bindParam(":id", $id);
    $stmt2->execute();
    $productos = $stmt2->fetchAll(PDO::FETCH_OBJ);
    $arr['destino'] = $destino;
    $arr['productos'] = $productos;
    $dbh = null;

    echo json_encode($arr);

    }
    catch(PDOException $e) {
    echo '{"error":{"text":'. $e->getMessage() .'}}';
    }
```

Ilustración 36. Consultas BBDD para obtener destino elegido.

Llamamos a la función DisplayDestino y lo primero que hacemos es recoger el tipo de destino que es y los tipos de vacaciones que ofrece para mostrarlo como información (ilustración 37).

```
function displayDestino(data) {
    var destino = data['destino'];
    var productos = data['productos'];
    console.log(destino);
    $('#destinoPic').attr('src', 'pics/' + destino.imagen);
    $('#ciudad').text(destino.ciudad + ' (' + destino.pais + ')');
    $('#zona').text("Zona: " +destino.zona);
    if (destino.relax == 1){
    $('#relax').text("Relax");
    }
    if (destino.aventura == 1){
    $('#aventura').text("Aventura");
    }
    if (destino.cultura == 1){
    $('#cultura').text("Cultura");
    }
    if (destino.naturaleza == 1){
    $('#naturaleza').text("Naturaleza");
    }
    if (destino.ciudades == 1){
    $('#ciudades').text("Grandes Ciudades");
    }
    if (destino.gastro == 1){
    $('#gastro').text("Gastronomia");
    }
    if (destino.playa == 1){
    $('#playa').text("Playa");
    }
```

Ilustración 37. Mostrar información del destino elegido.

Para después, recoger la lista de productos y mostrarla como productos ofertados para ese destino (ilustración 38).

```
$.each(productos, function(index, producto) {
    if (producto.transporte == 1) {var transporte ="Transporte ";} else { transporte="";}
    if (producto.alojamiento == 1) {var alojamiento ="Alojamiento ";} else {alojamiento="";}
    if (producto.actividades == 1) {var actividades ="Actividades ";} else {actividades="";}
    $('#productList').append('<li><h4>' + producto.titulo + '</h4>' +
            '<h3>Incluye: ' + transporte + ''+ alojamiento +''+actividades+'</h3>' +
            '<h3>Precio: ' + producto.precio + ' euros</h3>' +
            '<h2>' + producto.dias + ' dias<h2>' +
            '</a></li>');
});
$('#productList').listview('refresh');
```

Ilustración 38. Carga de la lista de productos ofertados por destino elegido.

Los usuarios anónimos o invitados también pueden acceder a los destinos aleatorios. Aquí se mostrará una lista de hasta tres destinos diferentes que siempre tendrán en común la zona geográfica del mundo y un tipo de actividad que se pueda realizar en ellos.

Para hacerlo de forma aleatoria utilizamos la función *rand()* de PHP para crear dos números aleatorios, que serán los tipos de actividades y la zona geográfica. Las dos consultas se muestran a continuación, en sendas ilustraciones 39 y 40.

```
$randomC = rand(1,7);
switch ($randomC) {
case 1:
    $consulta = " and playa = 1";
    break;
case 2:
    $consulta = " and relax = 1";
    break;
case 3:
    $consulta = " and aventura = 1";
    break;
case 4:
    $consulta = " and cultura = 1";
    break;
case 5:
    $consulta = " and naturaleza = 1";
    break;
case 6:
    $consulta = " and ciudades = 1";
    break;
case 7:
    $consulta = " and gastro = 1";
    break;
}
```

Ilustración 39. Consulta aleatoria para tipo de destino.

```php
$randomZ = rand(1,3);
switch ($randomZ) {
case 1:
    $zonaR = "zona = 'Europa'";
    break;
case 2:
    $zonaR = "zona = 'Asia'";
    break;
case 3:
    $zonaR = "zona = 'Africa'";
    break;

}
```

Ilustración 40. Consulta aleatoria para zona de destino.

Así montamos la consulta completa que lo que hace es buscar en la lista de destinos, aquellos con ese tipo de actividad en esa zona determinada. La consulta SQL se muestra completa en la ilustración 41.

```php
$sql = "SELECT * FROM `destino` where ".$zonaR."".$consulta." limit 3";
```

Ilustración 41. Consulta SQL con datos aleatorios.

Al utilizar la misma plantilla index.html que en la lista de destinos anterior, lo que hacemos es crear otra función para la lista de destinos aleatorios. La función javascript getRandomList() devolverá la lista de destinos aleatorios. La función de los destinos aleatorios la podemos ver en la ilustración 42.

```javascript
function getRandomList() {
    $.getJSON(serviceURL + 'getrandom.php', function(data) {
        $('#destinoList li').remove();
        destinos = data.items;
        $.each(destinos, function(index, destino) {
            $('#destinoList').append('<li><a href="destinodetails.html?id=' + destino.destinoid + '">' +
                    '<img src="pics/' + destino.imagen + '"/>' +
                    '<h4>' + destino.ciudad + ' ('+ destino.pais +')</h4>' +
                    '<h3>' + destino.zona + '</h3>' +
                    '</a></li>');
        });
        $('#destinoList').listview('refresh');
    });
}
```

Ilustración 42. Mostrar lista aleatoria de destinos.

Los usuarios invitados tienen la posibilidad de registrarse en el sistema para poder tener acceso a, por ejemplo, sus últimas búsquedas.

Para ello, deben rellenar un formulario de registro que se encuentra en el menú de la página inicial bajo el título Registro (ilustración 43).

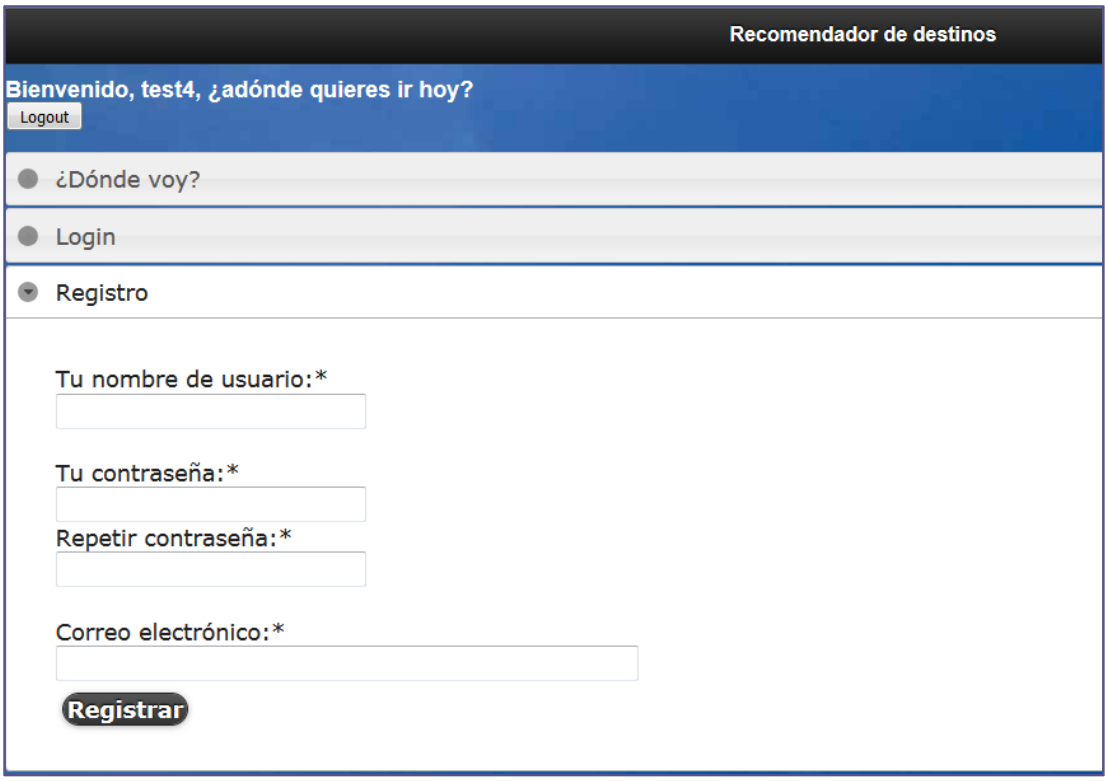

Ilustración 43. Formulario de registro para usuarios.

El usuario debe rellenar un nombre de usuario, una contraseña y una dirección de correo electrónico.

Los asteriscos nos indican que son campos obligatorios para realizar el registro y, en el caso de no rellenarse, el sistema no nos dejaría avanzar con el registro y veríamos el formulario con los campos en rojo, como vemos en la ilustración 44.

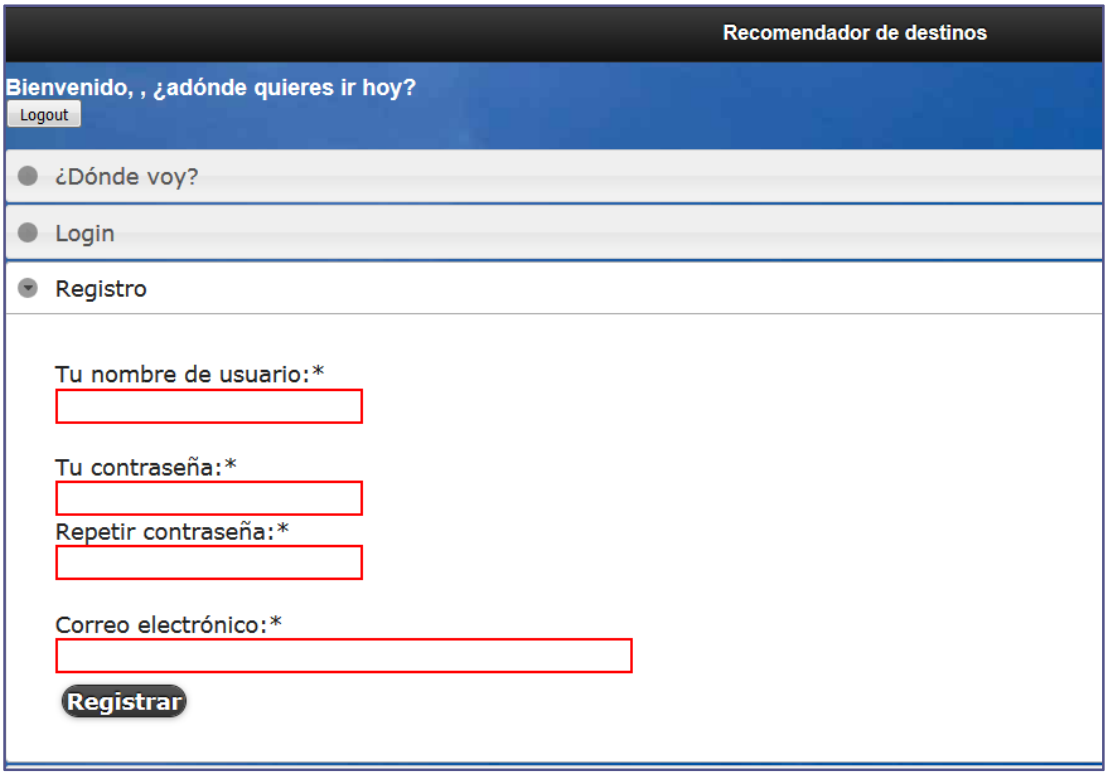

Ilustración 44. Formulario de registro sin validar.

Para conseguir este efecto utilizamos un script (ilustración 45) que se ejecuta antes de enviar el formulario a la parte del servidor.

```
<script>
    function validar() {
        var valido = true;
        if(registro.username.value == "") { registro.username.style.border="2px solid red"; valido=false; }
        if(registro.passwort.value == "") { registro.passwort.style.border="2px solid red"; valido=false; }
        if(registro.passwort2.value == "") {registro.passwort2.style.border="2px solid red"; valido=false; }
        if(registro.email.value == "") {registro.email.style.border="2px solid red"; valido=false;}
        return valido;
        }
</script>
```

Ilustración 45. Script de validación de campos.

Una vez que hemos enviado los datos al servidor, lo primero que hacemos es comprobar que el nombre de usuario no existe en nuestra base de datos (ilustración 46).

```
$conn = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$sql = "SELECT * FROM usuario WHERE `nombreusuario` LIKE :user";
$stmt = $conn->prepare($sql);
$stmt->bindValue(':user', $user);
$stmt->execute();
$ifuser = $stmt->fetch(PDO::FETCH_ASSOC);
```

Ilustración 46. Consulta SQL existe usuario.

Si esta función nos devuelve un conjunto vacío, sabemos que el usuario no existe y por tanto podemos proceder a insertar los datos en la base de datos.

En la ilustración 47 mostramos la consulta de inserción en la base de datos y la redirección a la página inicial después de ésta.

```
if($ifuser == false)
{
    $conn2 = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
    $conn2->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $qth = $conn2->prepare("INSERT INTO usuario (nombreusuario, contrasenya, email, tipousuario) VALUES (?,?,?,?)");
    $qth->execute(array($user, $passwordCRYPT, $email, $tipouser));


    //Redirigir a la página principal indice.html
    header( "refresh:1; url=indice.html");
    exit;
}
```

Ilustración 47. Introducir usuario en BBDD.

La contraseña nunca se guardará en texto plano en la base de datos, por eso, al recibir los datos del usuario se codifica con la función md5() propia de PHP.

Si existe el usuario, devolvemos el error de que el usuario ya existe (ilustración 48) y volvemos a la página inicial, desde donde podemos volver a iniciar el proceso.

```
else
{
    echo "El nombre de usuario ya existe.Por favor, elija otro nombre de usuario.
    <a href=\"indice.html\">Volver al inicio</a>";
}
```

Ilustración 48. Error al insertar usuario ya existente.

5.3.3 Usuario registrado

Los usuarios registrados tienen, además las tres opciones anteriores, la opción de recordar sus últimas búsquedas una vez están *logueados* en el sistema.

Para *loguearse* en el sistema pueden ir a la página inicial y bajo el epígrafe de *Login* pueden rellenar un formulario donde se les pide su nombre de usuario y contraseña. Para ello tienen que haberse registrado con anterioridad en el sistema.

En la figura 49 se muestra el formulario para el *login* de usuario.

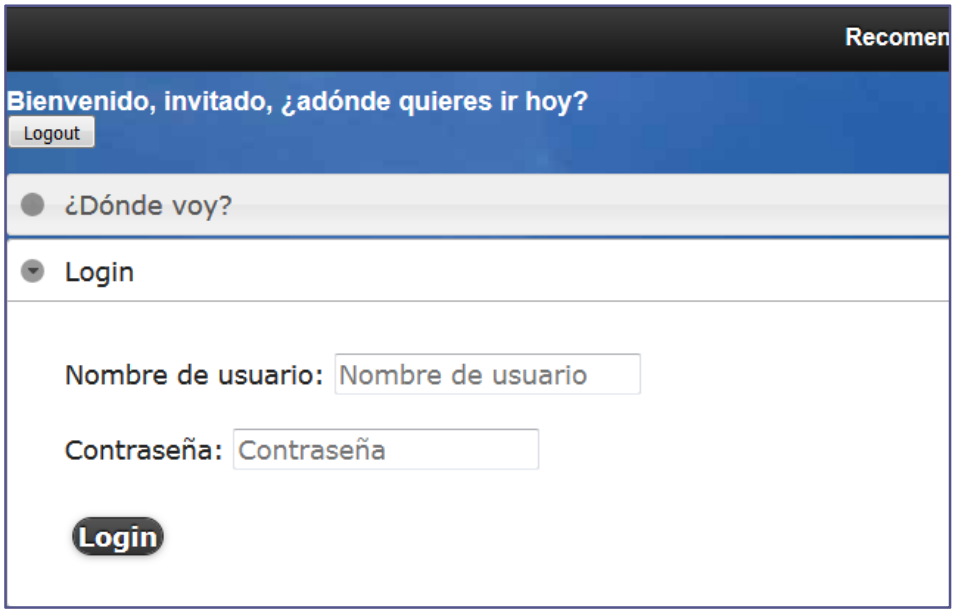

Ilustración 49. Formulario de Login.

Una vez efectuado el login, veremos que el saludo inicial cambia para saludar a usuario por su nombre. El saludo personalizado se muestra en la ilustración 50.

Ilustración 50. Mensaje de bienvenida al usuario logueado.

Una vez que el usuario se ha logueado, podemos ir a la parte de las últimas búsquedas. Lo que hacemos aquí es mostrar una lista con las 10 últimas búsquedas realizadas por el usuario, ya sean en esta sesión o en sesiones anteriores. Para ello, solo tenemos que ir al menú de la página de inicio del sitio bajo el nombre Últimas búsquedas (ilustración 51).

Ilustración 51. Menú últimas búsquedas.

Aquí nos llevará a una página donde nos mostrará la lista con un máximo de 10 búsquedas. Las búsquedas incluyen la zona y los tipos de vacaciones que se seleccionaron (ilustración 52).

Ilustración 52. Lista últimas búsquedas.

Cada una de las búsquedas es un vínculo a la lista con los destinos que devolvieron esas búsquedas, para poder volver a hacerlas o para recordar lo que se buscó. La lista que se devuelve se presenta en el mismo formato que una búsqueda normal desde el menú ¿Dónde voy?

5.3.4 Usuario administrador

El usuario administrador tiene, además de todas las opciones anteriores, la posibilidad de añadir destinos desde el propio sitio web. Para esto debe estar *logueado* con un usuario administrador que le permita insertar un nuevo destino en la base de datos.

Para saber que usuarios pertenecen al grupo de los administrado se ha creado un campo tipo_user en la base de datos que diferencia entre los dos tipos de usuarios, usuario registrado y administrador.

El usuario tiene que rellenar un pequeño formulario (ilustración 53) con los campos obligatorios para la creación de un destino nuevo. La zona en la que se encuentra, el país, el nombre de la ciudad y el tipo de actividades o productos que puede ofrecer a los visitantes.

Ilustración 53. Formulario insertar nuevo destino.

Cualquier usuario puede acceder a rellenar el formulario, por eso, al intentar insertar un nuevo destino, lo primero que hacemos es comprobar que el usuario que tiene la sesión iniciada es un administrador. La comprobación de inicio de sesión por parte de administrador se muestra en la ilustración 54.

```
if ($_SESSION['tipouser'] == 1)
```

Ilustración 54. Comprobar si es administrador.

Después, recogemos todos los valores que el usuario ha introducido y, si el usuario tiene los permisos necesarios, insertamos el destino en la base de datos. Este proceso lo podemos observar en la ilustración 55.

```php
try
    {
        $zona      = $_POST['zona'];
        $pais    = $_POST['pais'];
        $ciudad = $_POST['ciudad'];
        if (isset($_POST['playa'])) {$playa = 1;} else  {$playa = 0;}
        if (isset($_POST['relax'])) {$relax = 1;} else  {$relax = 0;}
        if (isset($_POST['aventura']))  {$aventura = 1;} else   {$aventura = 0;}
        if (isset($_POST['ciudades']))  {$ciudades = 1;} else   {$ciudades = 0;}
        if (isset($_POST['gastro']))    {$gastro = 1;} else {$gastro = 0;}
        if (isset($_POST['naturaleza']))    {$naturaleza = 1;} else {$naturaleza = 0;}
        if (isset($_POST['cultura']))   {$cultura = 1;} else    {$cultura = 0;}

        $conn2 = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
        $conn2->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $qth = $conn2->prepare("INSERT INTO destino (zona, pais, ciudad, relax, aventura,
        cultura, naturaleza,  ciudades, gastro, playa) VALUES (?,?,?,?,?,?,?,?,?,?)");
        $qth->execute(array($zona, $pais, $ciudad, $relax, $aventura, $cultura, $naturaleza,
        $ciudades, $gastro, $playa));


        //Redirigir a la página principal indice.html
        header( "refresh:1; url=indice.html");
        exit;
    }

    catch (Exception $e)
    {
        echo 'Excepción: ',  $e->getMessage(), "\n";
    }
```

Ilustración 55. Insertar nuevo destino.

6 Evaluación

6.1 Introducción

La fase de evaluación tiene como fin comprobar la respuesta de nuestro sitio web tienen frente a diferentes entornos, dispositivos o navegadores.

Son pruebas básicas que todos los sitios web deben superar con éxito para permitir la usabilidad de cualquier usuario en cualquier tipo de situación. Mostramos a continuación los resultados obtenidos en las siguientes pruebas.

6.2 Pruebas

6.2.1 Pruebas de resolución

Las pruebas de resolución son necesarias para que los usuarios puedan ver correctamente la página con independencia del navegador o dispositivo que estén utilizando para ello.

Como nuestro sitio web es además un sitio web pensado para ser visto desde dispositivos móviles, este aspecto es fundamental para nuestro sistema. Por eso hemos hecho pruebas de resolución para todos los tamaños de pantallas estándar y para resoluciones típicas en dispositivos móviles, siendo este el resultado de ellas.

Las pruebas realizadas muestran la compatibilidad del sitio web con una resolución de incluso 640x480 para pantallas con una baja resolución hasta los 1920x1080 de grandes monitores.

Para estas pruebas se han utilizado las aplicaciones web Screenfly y TestSize.

En la ilustración 66 y 67 a continuación veremos el ejemplo del sitio web en una resolución de 640x480.

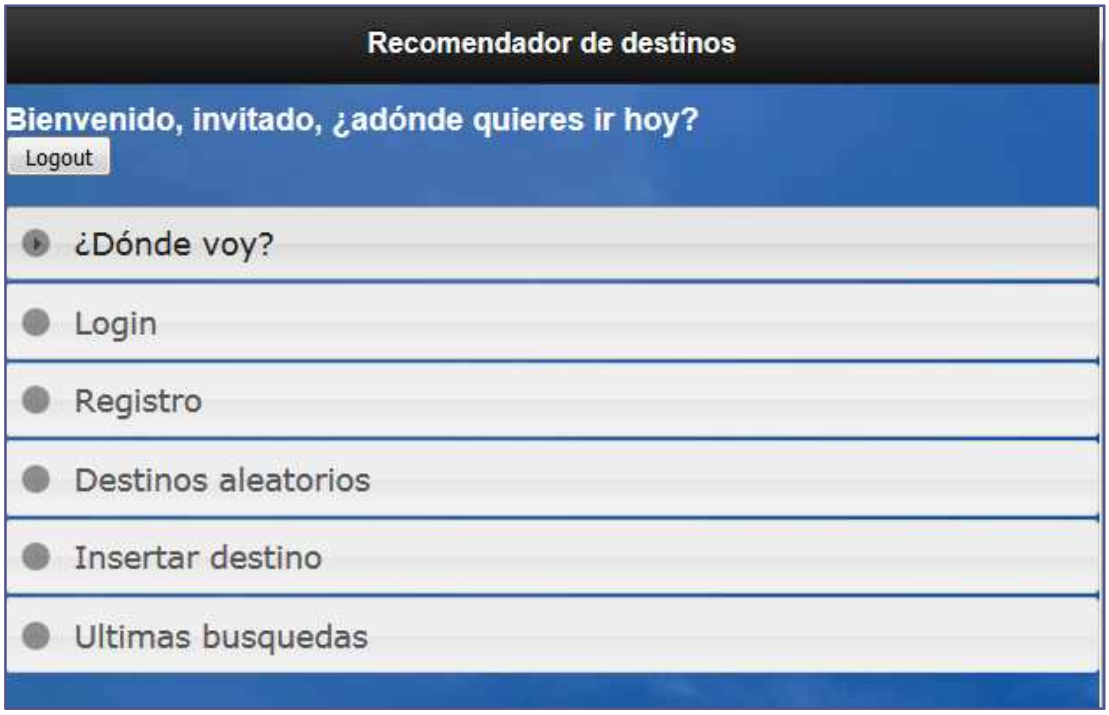

Ilustración 56. Página inicio 640x480.

Ilustración 57. Lista de destino 640x480.

En la siguiente ilustración, la ilustración 58, vemos la resolución de la página de inicio para una pantalla 1920x1080.

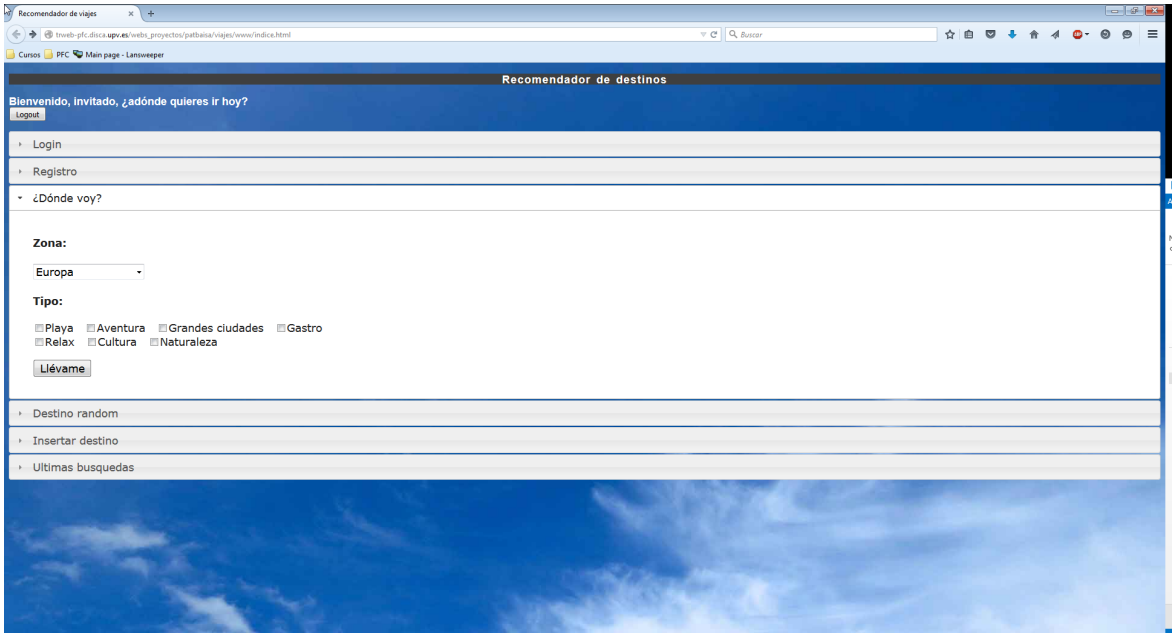

Ilustración 58. Página inicio 1920x1080.

En los ordenadores portátiles de medio tamaño es muy apropiada la resolución 1366x768. En la ilustración 59 vemos un ejemplo de la página de inicio de nuestro sitio web mostrada en esa resolución.

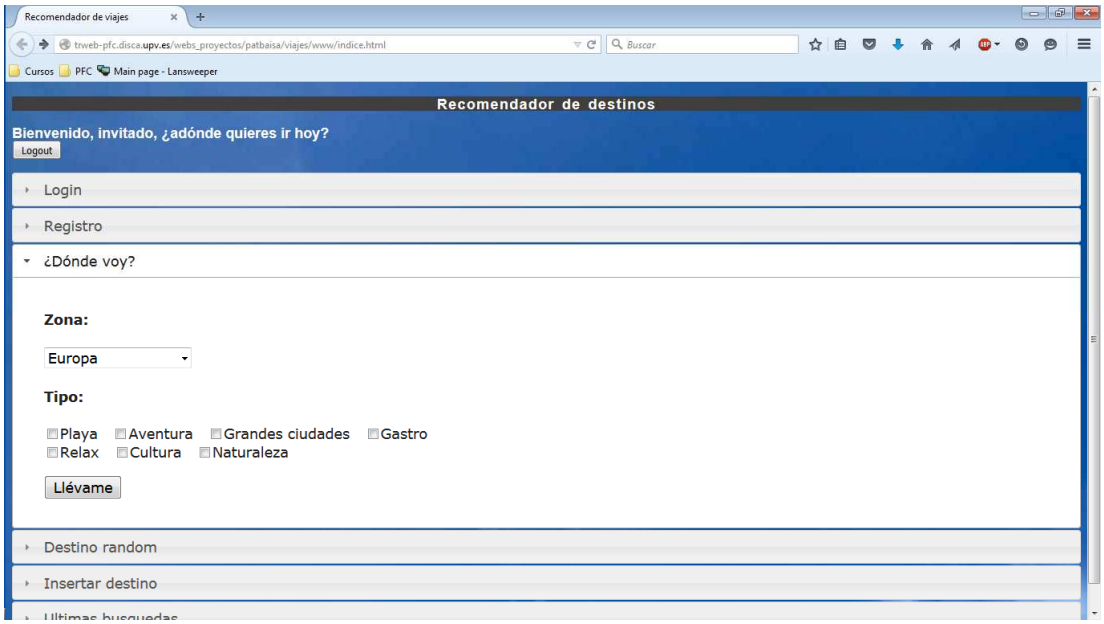

Ilustración 59. Página inicio 1366x768.

Los monitores pequeños o más antiguos también disponen de la resolución menos panorámica, como es la 800x600. En las siguientes ilustraciones mostramos el ejemplo de la página de inicio y la lista de destinos en esa resolución (ilustraciones 60 y 61).

Ilustración 60. Página inicio 800x600.

Ilustración 61. Lista de destino 800x600.

Las pruebas realizadas en móviles muestran que la resolución es correcta y que se puede utilizar el sitio web con facilidad. La ilustración 62 nos muestra la pantalla de inicio en una pantalla móvil con resolución 320x480, que es una resolución que adaptan muchos dispositivos móviles.

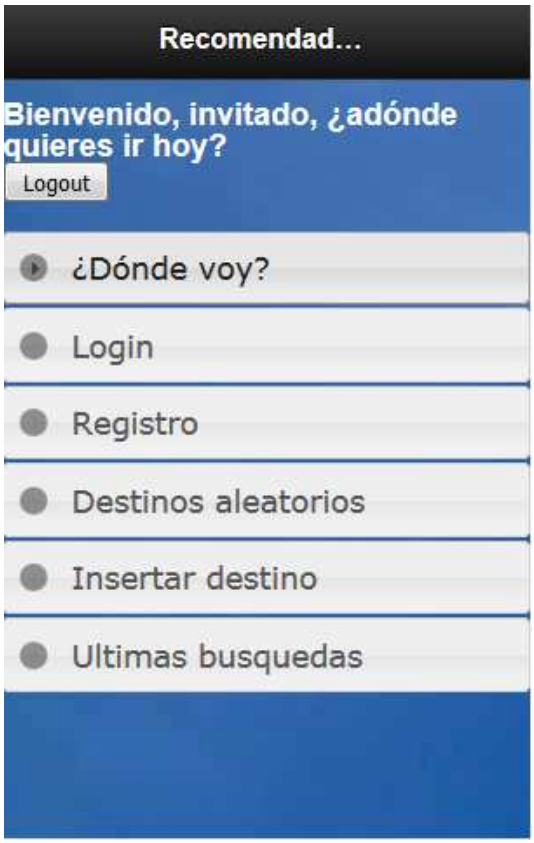

Ilustración 62. Página inicio 320x480.

En la ilustración 63 vemos la página con el listado de destinos también en una resolución 320x480. Como se puede observar, la usabilidad en dispositivos móviles también es total.

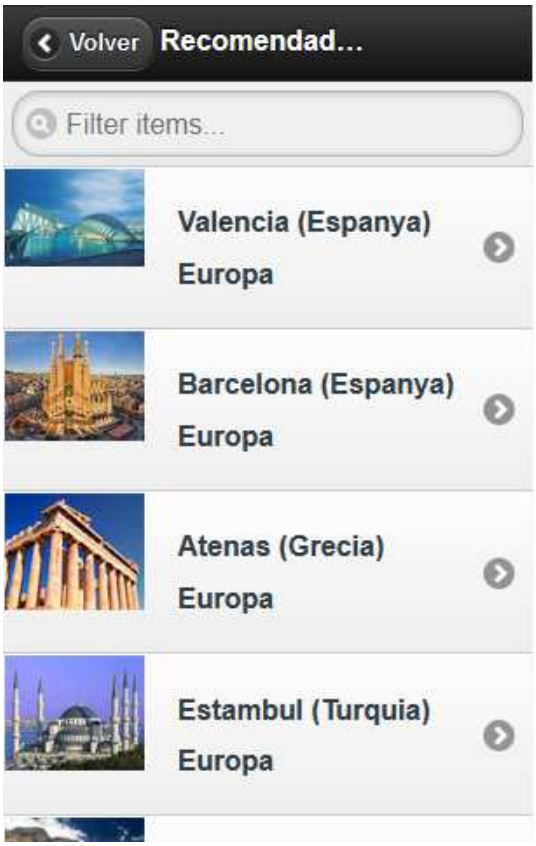

Ilustración 63. Lista destinos 320x480.

Al tener elementos de jQuery Mobile la usabilidad en dispositivos de pantalla reducida es mejor que otros sitios web no adaptados.

6.2.2 Pruebas con navegadores

Esta prueba consiste en comprobar el correcto funcionamiento y usabilidad del sitio web en los diferentes navegadores existentes. Para ello se han elegido los más representativos tanto visibles desde un ordenador como desde un dispositivo móvil.

Las pruebas se han realizado con Mozilla Firefox, Internet Explorer y Google Chrome. En los tres navegadores se ha observado un normal comportamiento de los elementos CSS, no siendo necesario el utilizar diferentes versiones para cada navegador.

Mozilla Firefox

Aquí vemos la página de inicio mostrada en el navegador Mozilla Firefox (ilustración 64).

Ilustración 64. Página inicio Mozilla Firefox.

La lista de destinos también se muestra correctamente en el navegador. Lo podemos ver en la ilustración 65.

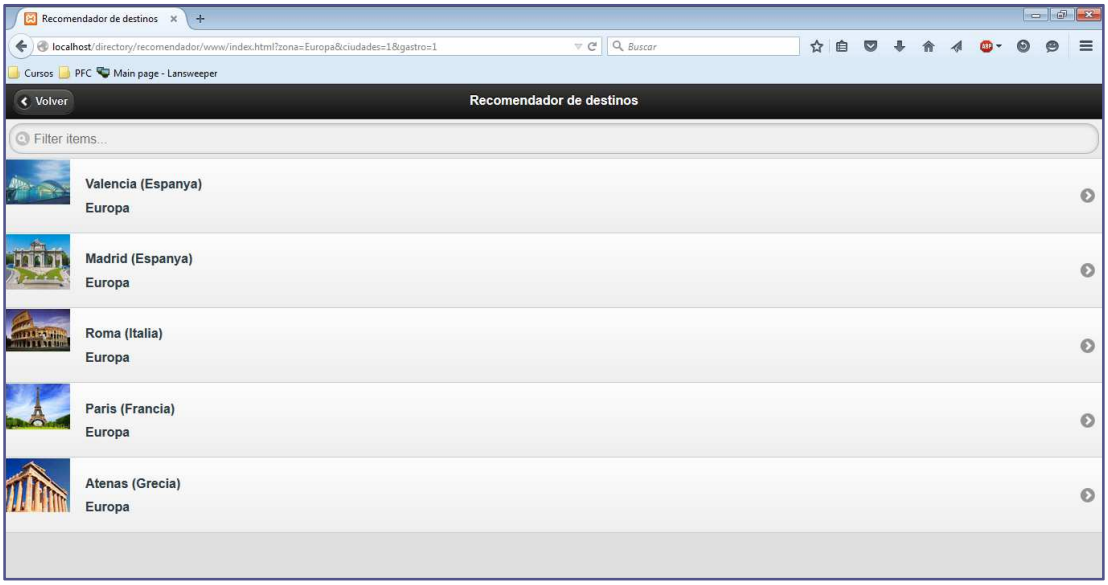

Ilustración 65. Lista de destino Mozilla Firefox.

Internet Explorer

Las pruebas realizadas en Internet Explorer también nos devuelven un resultado correcto. La lista de destinos de nuestro sitio web se puede observar en la ilustración 66.

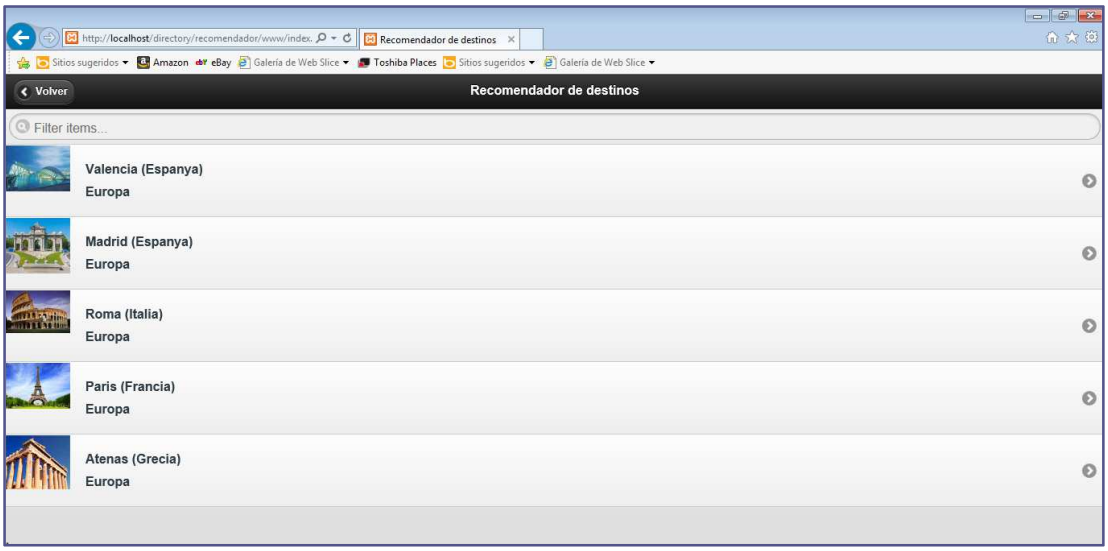

Ilustración 66. Lista de destinos Internet Explorer.

Google Chrome

La ilustración 67 muestra el ejemplo de la página de destinos seleccionados, mostrada en el navegador Google Chrome.

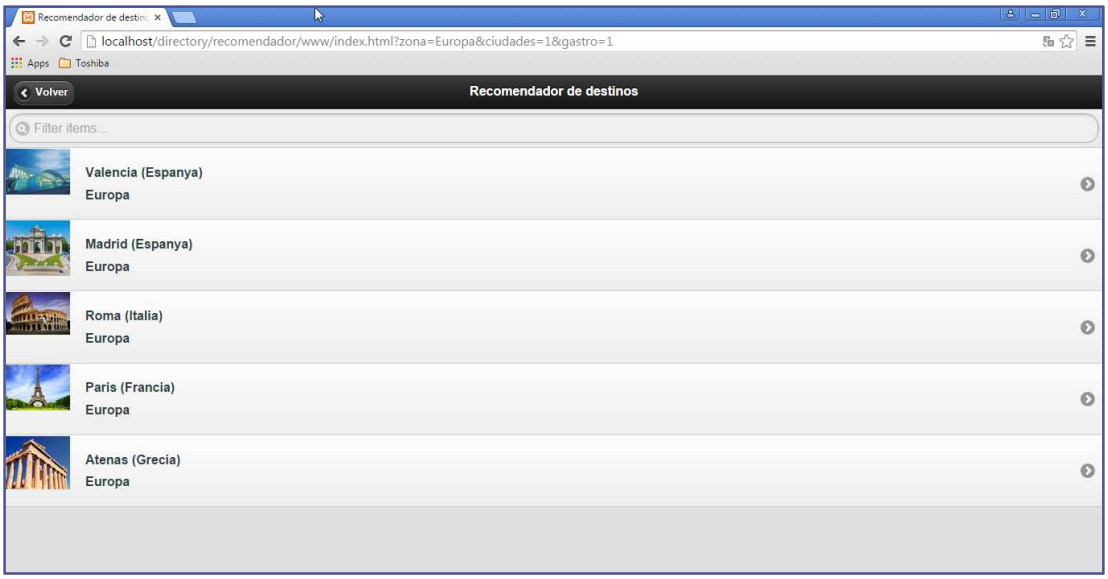

Ilustración 67. Lista de destinos Google Chrome.

7 Conclusiones

En este apartado comentaré las conclusiones que he extraído durante el proceso de realización de este proyecto final de carrera.

Desde el momento que supe que tenía que realizar el proyecto final de carrera, sabía que quería programar una aplicación o sitio web. Debido a mi trabajo como administradora de sistemas, no estoy muy familiarizada con el mundo web, pero me seducía la idea de probar la programación para internet con algo de diseño. El tema de hacer un portal de viajes vino recomendado por Félix Buendía en uno de los proyectos que él ofrecía para este curso, y como los viajes son una de mis pasiones, pensé que podía hacer algo nuevo e interesante para mí y que a la vez, el tema me motivara para dar lo mejor. Así que el proyecto de desarrollo de un sitio web para una agencia de viajes era el proyecto ideal.

En las primeras reuniones que tuvimos decidimos que no íbamos a hacer una agencia de viajes al uso ya que internet ya está lleno de páginas que ofrecen esos servicios, por eso nuestro sitio web iba a ser algo diferente e iba a ofrecer destinos a personas que no supieran dónde ir pero sí lo que esperaban de sus vacaciones y así es como surgió la idea del *recomendador* de destinos.

Al no ser la programación web mi área de especialización el proceso de estudio de las distintas tecnologías a utilizar fue mucho más costoso de lo que preveía. La programación por capas y la inclusión de javascript o jquery mobile hicieron que el trabajo de programación se retrasara hasta el final, después de tener unas nociones básicas de estos lenguajes de programación.

Al principio el proyecto iba a ser un proyecto en equipo. Mi parte del trabajo se iba a centrar en la base de datos y todo lo que dependiera de ella, devolviendo la información al sitio web para que se mostrase de una forma limpia y clara. Mi compañero, que se tenía que ocupar de la interfaz, se desvinculó del proyecto al final lo que hizo que el diseño del sitio web no se corresponda con lo que se planeó en la especificación de requisitos. Algunas de las funciones que al principio pensamos que serían útiles para el sitio web también se descartaron por falta de tiempo o por pensar que no eran necesarias. Algunas de éstas se propondrán como mejoras al final del documento.

Al realizar este proyecto he constatado que las fases de un proyecto ayudan a que el desarrollo marche de una forma ordenada, si es verdad que no han sido implementadas todas las funciones que se pensaron o de la forma en que se pensaron, pero ha ayudado a mantener un control y a seguir un orden

en la realización del proyecto. También he visto que detrás de un proyecto informático siempre hay más trabajo del que uno ve desde el principio, por eso es necesario seguir un esquema de trabajo y saber hasta dónde nos podemos desviar del camino o lo que hay que de dejar fuera.

La finalización de este proyecto ha sido un reto para mí, ya que como dije antes está fuera del ámbito de trabajo que yo realizo el día a día. Ha sido difícil y costoso llegar hasta el final pero el resultado me deja con una buena sensación. El trabajo no habría sido posible sin la ayuda de Félix Buendía que me ha guiado durante todo el proceso y siempre ha estado ahí, con paciencia, para cualquier problema o duda que surgiera.

Las posibles mejoras o ampliaciones para el sitio web son muchas así que las vamos a resumir en la lista siguiente.

- Opción de reserva de las vacaciones incluidas a los usuarios registrados en el sitio web.
- Ampliar opciones de búsqueda para destinos, como podría ser por país, por temporada del año, por proximidad al lugar de residencia.
- Foro de opiniones o preguntas sobre los destinos para los usuarios registrados del sitio web.
- Conexión del sitio web con las redes sociales para mostrar fotos de otros viajeros a los destinos ofertados.
- Lista de destinos visitados para los usuarios.

8 Bibliografía

- Enciclopedia online Wikipedia https://www.wikipedia.org/
- Christophe Coenraets http://coenraets.org/blog/2011/10/sample-application-with-jquery-mobile-and-phonegap/
- Stack Overflorw. http://stackoverflow.com
- W3 Schools http://www.w3schools.com/
- https://amitgharat.wordpress.com/2011/09/17/how-to-load-content-in-jqueryui-accordion-dynamically/
- http://tutorialzine.com/2009/10/cool-login-system-php-jquery/
- Esepe studio. Especialistas 10.0 http://www.esepestudio.com

Anexo A

Todos los usuarios.

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Buscar destino ideal	Se accede desde el menú de la página inicial. Área ¿Dónde voy?		
Entradas	El usuario tiene que introducir los siguientes datos: Zona del mundo y el tipo de actividad que le gustaría practicar en su destino		
Fuente	Formulario a rellenar por el usuario	**Salida**	Lista con todos los destinos que correspondan con la búsqueda que ha realizado el usuario
Destino	Página con los resultados de la consulta hecha por el usuario	**Restricciones**	Ninguna. Todos los usuarios pueden hacer la búsqueda del destino ideal
Proceso	El usuario introduce los parámetros y se realiza una búsqueda en la base de datos que devuelve los destinos que coinciden con las exigencias del usuario.		

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Destinos aleatorios	Se accede desde el menú de la página incial. Área Destinos aleatorios		
Entradas	El usuario solo tiene que pinchar el botón sin la necesidad de tener que rellenar ningún formulario		
Fuente	Base de datos	**Salida**	Se muestran los destinos aleatorios del día
Destino	Página con los resultados de los destinos predeterminados para ese día calculados por un algoritmo aleatorio	**Restricciones**	Ninguna. Todos los usuarios pueden ver los destinos aleatorios.
Proceso	Al ir al área de destinos aleatorios, se hace una consulta de los destinos utilizando un algoritmo random para que cada vez que el usuario pincha el botón, se devuelva un resultado diferente		

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Registro de usuario	Se accede desde el menú de la página inicial. Área Registro de usuarios.		
Entradas	El usuario tiene que rellenar un formulario en el que se pide. Nombre de usuario, contraseña (2 veces) y dirección de correo electrónico.		
Fuente	Formulario a rellenar por el usuario.	**Salida**	El usuario queda registrado en el sistema.
Destino	El destino es la misma página pero siendo el usuario ya un usuario registrado y no un invitado. Los datos se guardan en la base de datos para futuros accesos.	**Restricciones**	Ninguna. Todos los usuarios pueden hacer uso del registro.
Proceso	El usuario invitado introduce sus datos en el formulario y estos se introducen en la base de datos dándole un número de usuario registrado nuevo.		

Usuarios registrados

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Últimas búsquedas realizadas	Se accede desde el menú de la página inicial. Área Últimas búsquedas.		
Entradas	El usuario tiene que estar logueado para que el sistema sepa cuáles son sus últimas búsquedas.		
Fuente	Base de datos.	**Salida**	La lista de las últimas búsquedas realizadas por ese usuario
Destino	Página que muestra la lista de las últimas búsquedas realizadas por el usuario que está logueado en ese momento.	**Restricciones**	Sólo los usuarios registrados y logueados en ese momento pueden ver sus búsquedas.
Proceso	El sistema hace una consulta a la base de datos de las últimas búsquedas del usuario que está logueado en ese momento y las devuelve en forma de lista.		

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Login de usuario	Se accede desde el menú de la página inicial. Área Login.		
Entradas	El usuario debe introducir su nombre de usuario y contraseña.		
Fuente	Formulario rellenado por el usuario con sus credenciales	**Salida**	El usuario verá inmediatamente en la parte superior ha dejado de ser "Invitado" para ver su nombre de usuario en el mensaje de bienvenida.
Destino	Base de datos. La página se recarga y se puede leer el nombre del usuario en el mensaje superior.	**Restricciones**	Solo los usuarios previamente registrados podrán loguearse. El resto verán un error al no encontrarse su usuario.
Proceso	El usuario introduce su usuario y contraseña y la base de datos hace una búsqueda que coincida con esas credenciales. Si coincide, el usuario es inmediatamente logueado en la página como usuario registrado.		

Usuarios administradores

Especificación de requisitos funcionales			
Nombre	**Descripción**		
Insertar destino	Se accede desde el menú de la página inicial. Área Insertar destino.		
Entradas	El usuario administrador tiene que rellenar un formulario con todos los campos que corresponden al destino. La Zona en la que se encuentra, el país, el nombre de la ciudad y el tipo de actividades que ofrece.		
Fuente	El formulario rellenado por el usuario administrador.	**Salida**	El destino se inserta en la base de datos. No se verá hasta la próxima consulta.
Destino	Base de datos. Los datos introducidos por el usuario administrador son almacenados en la base de datos para formar parte de los destinos	**Restricciones**	Solo los usuarios administradores y logueados en ese momento pueden insertar destino. La imagen del destino ha de subirse al servidor separadamente.
Proceso	El usuario administrador rellena el formulario con los datos pertenecientes al destino. Se guardan en la base de datos y a partir de la consulta siguiente ya aparecerán como un destino más.		