Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Plataforma web para aprendizaje de idiomas

Autor: Carlos Romero Martínez
Tutor: Federico José Barrero García

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de la Telecomunicación

Plataforma web para aprendizaje de idiomas

Autor:
Carlos Romero Martínez

Tutor:
Federico José Barrero García
Profesor titular

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Plataforma web para aprendizaje de idiomas

Autor: Carlos Romero Martínez

Tutor: Federico José Barrero García

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

Agradecimientos

Agradecer a mis padres y a mis abuelos por ayudarme a llegar hasta aquí. Agradecer a mis tíos, que más que tíos son hermanos y que en gran parte tienen la culpa de que sea como soy, y a mis primos que han hecho las veces de hermanos pequeños. Agradecer a Julián y Araceli por tratarme como a un hijo más. Y, sobre todo, agradecer a Mari por estar conmigo todos estos años, ya que sin su apoyo incondicional no habría llegado tan lejos.

Carlos Romero Martínez

Autor del Proyecto

Sevilla, 2016

Resumen

El proyecto trata sobre la creación de una plataforma web que pondrá en contacto a personas interesadas en el aprendizaje de diferentes idiomas.

En este documento se detallan las tecnologías utilizadas para la realización de la web, la estructura de los documentos que se han escrito para el correcto funcionamiento de la misma y un manual de usuario acompañado de capturas que hacen de guía.

Además de esto, se hablará de posibles mejoras que se podrán introducir en futuras actualizaciones.

Abstract

The project deals with the development of a website that will connect people interested in learning different languages.

This document describes the technologies used for the development of the web, the structure of the documents that have been written for the proper operation and a user manual with pictures.

Following this, we will talk about possible updates that could be included in the future.

Índice

ÍNDICE DE FIGURAS

1 MOTIVACIÓN Y OBJETIVOS

En la actualidad y debido a la globalización, hablar más de un idioma es una cualidad cada vez más necesaria tanto para el mundo laboral como para el ámbito social. Ante esta necesidad surge la idea de poner en contacto de una forma fácil y rápida a personas que deseen intercambiar, aprender o mejorar un idioma, y que se encuentren en un lugar geográfico en concreto.

Comenzando con el envío del primer correo electrónico en 1971 a manos de Ray Tomlinson, y la creación de la primera página web en 1991 por Tim Berners-Lee, el desarrollo de internet y las telecomunicaciones en estas pocas décadas de existencia ha sido considerable, hasta tal punto de poder acceder a cualquier contenido desde la palma de nuestra mano gracias a los smartphones.

Figura 1-1. Primera página web creada en 1991.

Que internet ha revolucionado la informática y las comunicaciones en los últimos tiempos es un hecho en sí, siendo una herramienta a nivel mundial con la que poder difundir cualquier tipo de información de forma instantánea.

Por lo tanto, ¿qué mejor lugar que una plataforma web en internet para ubicar nuestra herramienta? Un lugar accesible desde la mayoría de paises del mundo y al alcance de cualquier persona.

2 INTRODUCCIÓN

Si bien es cierto que actualmente se han puesto de moda los CMS o sistemas de gestión de contenidos puesto que facilitan de forma considerable el trabajo a los desarrolladores de sitios web, para este proyecto se ha optado por no utilizar ninguno de ellos y trabajar desde un lienzo en blanco. Esta forma de programar permite el control total sobre el código de tal forma que siempre se comportará como uno ha escrito. No obstante, haremos una breve descripción sobre el funcionamiento y las ventajas que pueden obtenerse al utilizar un gestor de contenidos. En este capítulo también cabe resaltar la evolución Internet sin el cual no se concibe el desarrollo de estas herramientas, y por último la importancia del posicionamento web.

2.1 Evolución de internet

Desde la creación de ARPANET en 1969, una red que conectaba varios ordenadores a través de la línea telefónica, la evolución de internet hasta nuestros días se puede considerar un elemento clave en el desarrollo y la historia reciente del ser humano.

No podríamos considerar ARPANET como el concepto que tenemos actualmente sobre Internet, ya que ni siquiera utilizaba los protocolos TCP/IP y se trataba de una red privada creada por encargo del Departamento de Defensa de los Estados Unidos. En la primera fase del proyecto ARPANET se unieron entre sí cuatro nodos a través de líneas telefónicas alquiladas utilizando el primer protocolo desarrollado NCP (Network Control Protocol). Cada nodo estaba unido a otros dos ofreciendo mayor robustez a la conexión. Estos nodos fueron cuatro universidades: Utah, Stanford, Santa Barbara y Los Ángeles.

En 1971 se publican las RFC de los primeros protocolos del nivel de aplicación que fueron desarrollados: FTP para la transferencia de ficheros, TELNET para la ejecución de comandos en otros equipos y Mail Box Protocol, que fue el primer protocolo para el envío de mensajes de correo electrónico.

Años después, en 1985 entra en servicio lo que se considera el auténtico nacimiento de internet, llamado NSFNET, una red que ya utilizaría los protocolos TCP/IP y cuya misión es la de ser troncal para la conexión de redes. Esta red fue creada por la NSF (Fundación nacional de ciencia) y estaba subvencionada por el gobierto de los Estados Unidos, con la idea de interconectar todas las universidades americanas para compartir datos y resultados de investigaciones.

En Europa y el resto del mundo la situación fue diferente. La ISO publicó en 1984 el modelo OSI, por lo que la entrada de los protocolos TCP/IP fueron frenados inicialmente para que la conexión de redes se llevase a cabo utilizando este modelo. Pero la ralentización en el desarrollo de los protocolos OSI definitivos y la progresiva implantación de sistemas basados en UNIX que incluían de forma nativa los protocolos TCP/IP hizo que finalmente triunfaran estos últimos.

Entre finales de los años 80 y principio de los 90 otros países fueron conectándose a la red NSFNET con carácter académico, siendo estas conexiones gestionadas por las universidades.

Fue en 1991 cuando se creó la primera web, siendo el nacimiento del WWW (World Wide Web). El éxito de este sistema fue inmediato ya que permitía la publicación de documentos en internet con la posibilidad de añadir enlaces a otros documentos. No sería hasta 1993 cuando se ejecutó Mosaic, el primer navegador en un entorno gráfico.

En 1995, la NSF transfiere el control de Internet a cuatro operadoras norteamericanas, que constituyen los llamados NAP (Network Access Point) y que serían los encargados de proporcionar conectividad a las empresas que ofrecían servicios de conexión a Internet, los llamados ISP.

Tras esto, el proceso de descentralización de la red sigue avanzando, desapareciendo los NAP y estableciéndose la arquitectura definitiva y que se mantiene en la actualidad.

2.2 Posicionamiento web

El SEO (Search Engine Optimization) es un aspecto realmente importante para conseguir visitas en nuestro sitio web. Con una buena estrategia seremos capaces de atraer a un mayor número de visitas y posicionarnos por delante de otros servicios que puedan competir con el nuestro.

Los motores de búsqueda como Yahoo o Google tienen en cuenta diversos factores a la hora de mostrar sus resultados. Entre otros, influyen la popularidad y la relevancia del contenido de los sitios web, ordenándolos en una lista empezando por el más popular o apropiado para el usuario.

Para obtener un buen posicionamiento podemos recurrir a dos técnicas: aspectos relacionados con el contenido de nuestro sitio y aspectos relacionados con elementos externos a él.

2.2.1 Aspectos internos

Podemos utilizar algunas buenas prácticas a la hora de escribir nuestro documento para lograr un buen posicionamiento web. Uno de las más importantes es el uso de palabras clave o keywords en nuestros documentos. Estas palabras clave debemos ponerlas en diferentes lugares de nuestro sitio, como puede ser el título, subtítulo, varias veces dentro de la etiqueta BODY, en la descripción, en la descripción alternativa de las imágenes.

Otro de los aspectos básicos a tener en cuenta para obtener un buen posicionamiento en los buscadores es no utilizar direcciones excesivamente largas para que nuestra página no sea tratada como SPAM. En la URL, incluir la palabra clave también lograría un mejor posicionamiento.

2.2.2 Aspectos externos

Hasta ahora hemos visto la forma en la que los algoritmos de los buscadores identifican nuestra página web como una web interesante para ser mostrada a los usuarios pero, además de esto, la popularidad de nuestro sitio también influye a la hora de conseguir las posiciones más altas en los buscadores.

Para ello, uno de los aspectos externos que podemos tratar es el alcance que tiene nuestro sitio web en las redes sociales. Si hay páginas que citan a la nuestra, los buscadores también tendrán este factor en cuenta y lograremos posiciones altas en las búsquedas.

Por lo tanto una buena difusión de nuestro sitio web, hará que tengamos un buen posicionamiento, logrando así un mayor número de visitas.

2.3 Sistemas de Gestión de Contenido (CMS)

Los gestores de contenido ofrecen módulos con funcionalidades específicas para ser insertados en cualquier lugar de una página web, pero hasta que no están todos operativos, uno no sabe si serán compatibles entre sí o si cumplirán completamente con el propósito que uno quiere, ya que en muchas ocasiones han sido desarrollados por personas diferentes.

Figura 2-1.Logos de dos gestores de contenido populares.

Generalmente, un CMS se instala en el servidor proporcionando al desarrollador una interfaz basada en formularios con los que se pueden dar de alta los contenidos de una manera sencilla, y al que se accederá habitualmente utilizando un navegador web.

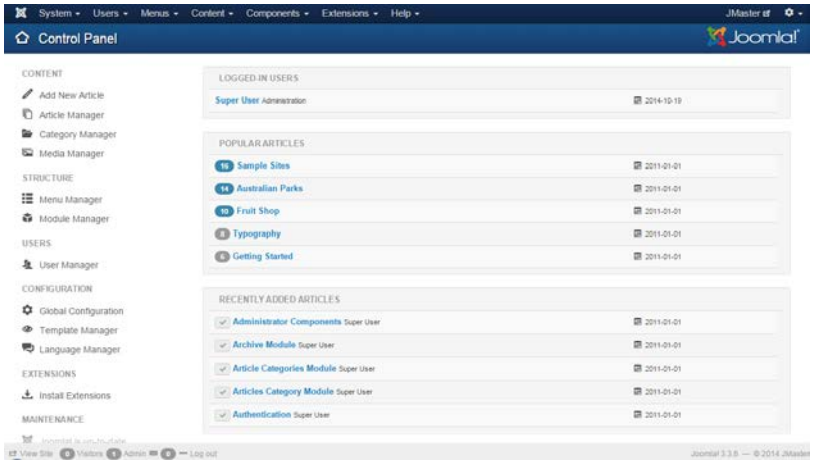

Figura 2-2. Ejemplo back-end de Joomla!

Los contenidos añadidos por el administrador aparecerán en la página en el lugar donde fueron dados de alta, y que serán utilizados por los usuarios finales. Podemos, por lo tanto, diferenciar dos partes en los CMS, una en la que los administradores añaden información, y otra con la que interactúan los visitantes de la web, llamadas back-end y front-end respectivamente.

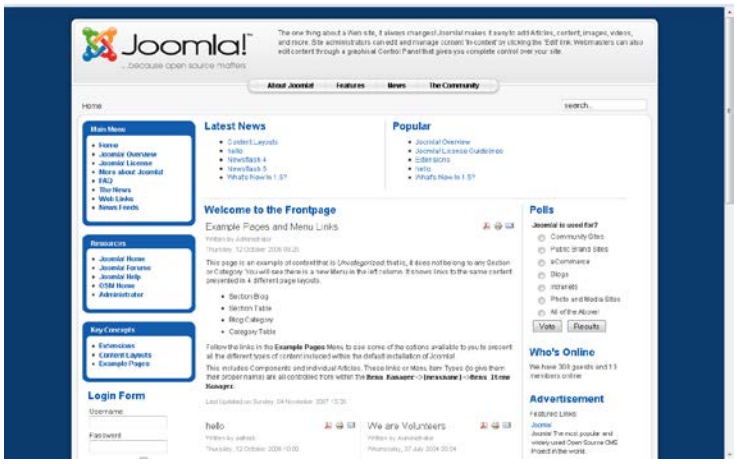

Figura 2-3. Ejemplo front-end de Joomla!

A nivel estructural, los gestores de contenido pueden dividirse en 3 aspectos:

- **Diseño:** relacionado con los lenguajes de etiqueta HTML y CSS, y con lenguajes dinámicos en el lado cliente como javascript. Este aspecto del programa define la maquetación sobre la que se inserta el contenido. Los gestores de contenido suelen traer un diseño básico y diferentes plantillas sobre las que poder modificar diferentes aspectos para adaptarlas a las necesidades del desarrollador.
- **Bases de datos:** este es uno de los pilares básicos de los gestores de contenido, puesto que es donde se almacena todo el contenido publicado en la web, la configuración de la misma, los usuarios, etc.
- **Programación:** este aspecto es el encargado de extraer la información que los usuarios han solicitado, mostrándosela de forma estructurada según el diseño. Esta parte de los CMS suele estar escrito en

PHP.

Desarrollar contenido web utilizando gestores de contenido puede suponer grandes ventajas frente a no utilizar uno, como puede ser agilizar el mantenimiento de todas las páginas que componen el proyecto, el cambio de estética cambiando el diseño sin que el funcionamiento se vea afectado, la reutilización de contenido almacenado, el control de acceso, etc.

Pese a todos estos beneficios, se ha optado por no utilizarlos para el desarrollo de este proyecto y así enfrentarnos directamente a la tarea de tener que programar una página web desde cero, teniendo que aplicar todos los conocimientos de programación y seguridad informática adquiridos durante la carrera

2.4 Contenido de la memoria

En este documento se expondrán las tecnologías utilizadas para la realización de la plataforma web para el intercambio de idiomas, la estrutura de páginas que se han utilizado para realizar el programa, el manual de usuario y futuras mejoras que podrían llevarse a cabo. Todo ello se organizará en capítulos de la siguiente forma:

- **Tecnologías utilizadas:** donde se abordarán los lenguajes utilizados para la realización de la plataforma web. También se hablará sobre la encriptación utilizada para la seguridad de las contraseñas y sobre el programa que se ha utilizado como servidor web que será el encargado de mantener la página accesible a los usuarios y mediante el cual se podrán gestionar las bases de datos.
- **Modelo estructural del código:** en esta sección se mostrará qué archivos componen la web y cómo interaccionan entre ellos. También se hablará en detalle sobre cada documento, las funciones que se han creado y los errores que pueden detectarse durante el envío de los formularios y el procesamiento de los datos.
- **Manual de usuario: e**ste apartado detallará el funcionamiento de la interfaz que se encontrará el usuario e irá acompañado de capturas para la mejor comprensión del mismo.
- **Conclusiones y futuros trabajos:** como toda plataforma web alojada en internet, la página que se trata en este proyecto será un punto de partida para la idea del intercambio de idiomas. Aquí se hablará de extras que podrían añadirse en futuras actualizaciones de este servicio web.

3 TECNOLOGÍAS UTILIZADAS

Como hemos dicho anteriormente, este apartado tratará la parte técnica de la plataforma web. Se expondrán los lenguajes utilizados y las funciones externas, se hablará sobre la encriptación utilizada para la seguridad de las contraseñas y sobre el programa que se ha utilizado como servidor web que será el encargado de mantener la página accesible a otros equipos y mendiante el cual se podrán gestionar las bases de datos.

3.1 Lenguajes utilizados

Se han utilizado diferentes lenguajes de programación que cumplen con diferente propósito. Algunos solo se ejecutan en el servidor, otros en el cliente y otros simplemente formatean la información para darle estilos a la interfaz de usuario.

3.1.1 HTML5

Hyper Text Markup Language es un lenguaje de etiquetas que se utiliza para estructurar y presentar el contenido de una web.

HTML5 es el nuevo estándar que añade nuevas funciones al antiguo HTML. Hace posible la inserción de contenido multimedia en los sitios web sin que el usuario tenga que tener instalado plugins externos como Adobe Flash, lo que hace que los navegadores consuman menos recursos obteniendo el mismo resultado.

Además de esto se han añadido nuevas etiquetas que facilitan la programación al desarrollador, como por ejemplo evitando que se tenga que utilizar javascript para comprobar formularios básicos.

3.1.2 CSS3

Cascade Style Sheets es un lenguaje que se utiliza para definir la presentación de un documento estructurado como puede ser HTML o XML. Se utiliza para separar la estructura de un documento de su presentación, pudiendo escribirse en un mismo documento o en dos documentos separados.

Los estilos CSS definen la estética entera de un sitio web, pudiendo ser cambiada totalmente sin alterar líneas de código en el documento HTML al que haga referencia.

3.1.3 PHP

PHP Hypertext Preprocessor. A diferencia de HTML, este código es ejecutado en el servidor, generando código HTML y enviándolo al cliente. El cliente recibe el resultado de la ejecución pero no tendrá forma de saber el código subyacente escrito en PHP.

3.1.4 JavaScript

JS o JavaScript es un lenguaje de programación interpretado, orientado a objetos y dinámico. Su uso es principalmente del lado cliente permitiendo mejoras dinámicas en la interfaz de usuario, aunque también puede ser utilizado en la parte del servidor.

3.2 Librerías externas y seguridad

Se ha hecho uso de algunas librerías o API externas para la realización de varias funciones que se utilizan en la plataforma web, como por ejemplo la indentificación del tipo de dispositivo desde el que el usuario navega, el cajón de búsqueda de lugares de Google, la conversión de lugares a coordenadas y el envío de correos electrónicos desde el lenguaje PHP.

A continuación algunos detalles sobre el funcionamiento de dichas librerías y las posibles utilidades que se le podrían dar:

3.2.1 Mobile-Detect

Utilizar esta librería permite la identificación del tipo de dispositivo que está utilizando la persona que está navegando por la web. Esto puede utilizarse para la adaptación del tamaño de los elementos en pantalla o para guardar estadísticas sobre los dispositivos más utilizados para navegar.

Con estas funciones podemos saber si el dispositivo desde el que se accede a la web es un smartphone o una Tablet (o ninguno de ellos), e incluso saber la marca o modelo del mismo.

3.2.2 API de Google Maps Geocoding

Esta API da acceso a funciones de los mapas de Google. Creando un search box o cajón de búsqueda que enlaza con la base de datos de lugares de Google, puede mostrar sugerencias al usuario que vaya a insertar o que esté buscando un lugar en la base de datos propia de este proyecto.

La geocodificación es un proceso que convierte direcciones en coordenadas geográficas que se pueden utilizar para posicionar un cursor sobre un mapa, o para guardar en una base de datos la latitud y longitud del lugar

buscado.

Para empezar a utilizar este servicio de Google lo primero que hay que hacer es registrarse como usuario y desarrollador de aplicaciones. Una vez registrado se debe obtener una clave o key acorde a las necesidades del sitio web que se esté desarrollando. Esta clave dependerá del número de consultas que se pueden cursar, de dónde va a ser ejecutada la aplicación (cliente, servidor), etc., siendo la más básica gratuita y teniendo diferentes cuotas para las siguientes.

La consulta a la base de datos de Google devuelve una matriz "results" JSON. Este resultado está compuesto por los siguientes campos:

- **types[ ]:** una matriz que proporciona el tipo de resultado devuelto. Contiene un grupo de 0 o más etiquetas que identifican el tipo de función devuelto en el resultado. Estas etiquetas identifican si el valor devuelto corresponde a una calle, una carretera, una intersección, una ciudad, un país, etc.
- **formatted_address:** esta cadena contiene la dirección de la ubicación en lenguaje natural. Normalmente dicha dirección equivale a una dirección postal que difiere según el país. Cada dirección contiene varios componentes con información adicional como se indica a continuación.
- **address_components[ ]:** esta matriz guarda la información de cada componente de la dirección en varios campos, siendo los siguientes algunos de los muchos que puede contener:
    - **types[ ]:** matriz que indica el tipo de componente de la dirección. Los tipos devueltos pueden corresponder a un piso, un punto de interés, una casilla de correo específica, un conjunto de áreas geográficas, etc.
    - **long_name:** descripción textual o nombre completo del componente de la dirección, tal y como lo devuelve el geocodificador.
    - **short_name:** nombre textual abreviado, siempre que esté disponible. Por ejemplo para "Alaska", su long_name sería Alaska y su short_name sería AK.
- **postcode_localities[ ]:** dicha matriz contiene todas las localidades que dependan de un código postal común.
- **geometry:** puede contener la siguiente información:
    - **location:** este campo contiene los valores de longitud y latitud del lugar buscado. Suele ser el campo más importante.
    - **location_type:** guarda datos adicionales sobre la ubicación especificada. Hace referencia al tipo de precisión devuelto como puede ser el centro geométrico del lugar, indicar que las coordenadas son una aproximación, indicar que el valor devuelto es el geocódigo exacto para esa ubicación, etc.
    - **viewport:** contiene el viewport recomendado para mostrar el usuario un resultado devuelto,

especificando dos valores de latitud y longitud que definen la esquina sudoeste y noreste del cuadro límite del viewport. El viewport se usa para enmarcar un resultado que se mostrará al usuario.

  - **bounds:** guarda el cuadro de límite que puede contener por completo el resultado devuelto. Estos límites no tienen por qué coincidir con el viewport recomendado.

- **partial_match:** este campo indicará que el geocodificador no ha devuelto una coincidencia exacta para la solicitud original, pero pudo encontrar una coincidencia parcial para la dirección solicitada.
- **place_id:** place_id es un identificador único que se puede usar con otras API de Google para obtener información detallada de un lugar en concreto como puede ser un negocio, obteniendo datos como su número de teléfono, horario de apertura, etc.

3.2.3 Google Maps Geocoder PHP

Como hemos comprobado en el subapartado anterior, las búsquedas de lugares en Google devuelven matrices algo complicadas de utilizar directamente. Para no tener que implementar funciones propias que podrían ser bastante engorrosas, se ha hecho uso de una librería externa de código libre.

Esta librería se utiliza para trabajar de una forma sencilla en PHP con la API de Google. Provee funciones que manejan de manera intuitiva los datos devueltos en la matriz resultado de la consulta.

En general, el uso de estas funciones durante el proyecto ha sido el siguiente:

- Creación de una variable $Geocoder del tipo GoogleMapsGeocoder al que se le ha pasado la dirección en forma de cadena buscada en el search box de Google.
- Creación de una variable $respuesta que utiliza la función $Geocoder->geocode() y que contiene todos los valores “results” descritos anteriormente.
- En este punto ya podemos acceder al contenido de results con funciones como getAddress(), setAddress(), getLatitude(), etc., o accediendo directamente (en este caso para obtener la latitud de la dirección buscada) a través de $respuesta[‘results’][0][‘geometry][‘location][‘lat’].

3.2.4 Clase PHPMailer

Enviar un correo electrónico sin acceder a un cliente web o utilizar un programa de escritorio puede resultar algo complicado dependiendo de lo complejo que sea el contenido del email. Con esta clase tenemos un conjunto de funciones para PHP que facilita enormemente este trabajo. El proceso para enviar un email desde PHP resulta muy sencillo de esta manera, limitándose a una función llamada Send( ) que realizará el envío utilizando una serie de parámetros que se le han pasado previamente al objeto mail.

El proceso para enviar un email es sencillo tras incluir dicha clase. El primer paso es instanciar un objeto de la clase PHPMailer. Tras esto debemos definir el cuerpo del mensaje que puede ser un documento html. Debemos añadir la dirección y nombre del remitente del mensaje, la dirección de respuesta al correo (que debería ser la misma que el remitente para no ser filtrado como SPAM por algunos servidores) y la dirección del destinatario junto a su nombre. También podremos añadir un cuerpo alternativo del mensaje que contenga solo texto plano. Después de esto podríamos añadir un archivo adjunto al mensaje y por último haríamos uso de la función Send( ).

3.2.5 Encriptación de contraseñas

Para la seguridad del usuario, las contraseñas no se almacenan directamente en la base de datos, sino que son previamente encriptadas utilizando un algoritmo de un solo sentido llamado Blowfish para calcular el hash. Este método de encriptación sirve para proteger las contraseñas de los usuarios ante posibles accesos no autorizados que supongan el robo de información de la base de datos.

De esta forma una contraseña como “Carlos12345678” sería almacenada en la base de datos como “$2y$10$4NvPuImwK3L52kPup7r0quc9s7Xzr3NpPdWnyomXT8BdYkLBGDyOO”, no pudiéndose obtener Carlos12345678 partiendo de su encriptación.

3.3 Programa servidor web

Para la creación y comprobación del funcionamiento de la plataforma web, se ha utilizado el programa WAMPServer (Windows Apache MySQL PHP Server). Dicho programa es un servidor gratuito muy completo para el desarrollo web en Windows con toda la potencia del servidor Apache, compatible con el lenguaje de programación PHP y que puede gestionar bases de datos MySQL.

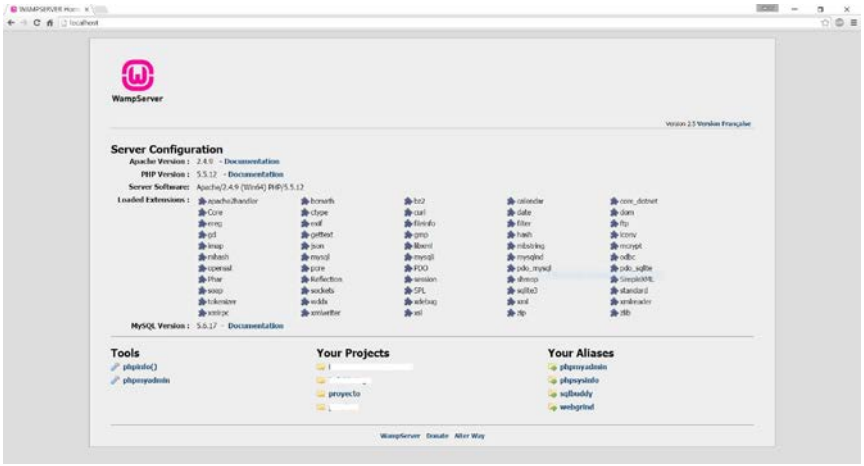

Figura 3-1. Página inicial del programa WAMPServer.

Su utilidad es importante ya que puedes trabajar con páginas web en proceso de elaboración de manera offline en una red local, como si estuvieras accediendo a ella a través de internet estando alojada en un servidor web.

Una de las ventajas de este programa es que cuenta con un administrador de bases de datos accessible vía web llamado PHPMyAdmin. Este programa facilita enormemente la tarea de administración de un desarrollador dando acceso a todas las bases de datos que componen los proyectos que esté desarrollando, pudiendo cambiar entre uno y otro sin necesidad de editar los demás.

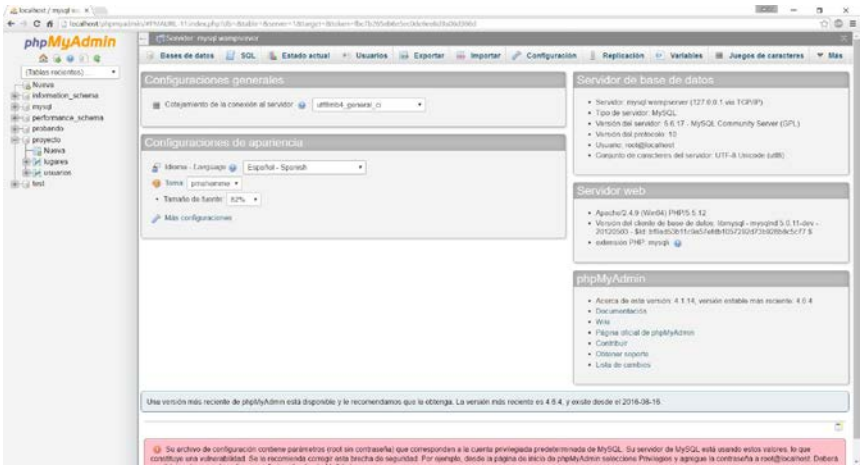

Figura 3-2. Página principal PHPMyAdmin.

Con esta herramienta podemos crear bases de datos, introducir o editar datos en ella, realizar consultas y generar scripts SQL, además de importar y exportar bases de datos ya creadas.

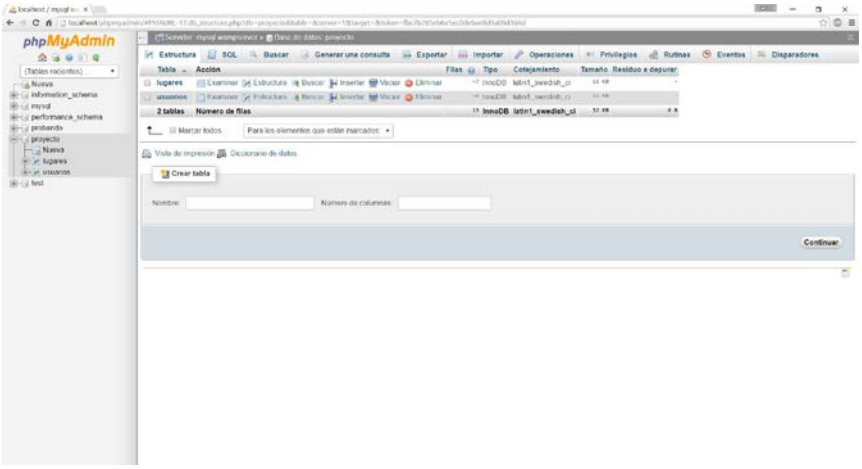

Figura 3-3. Administrador de “proyecto” en el que vemos 2 tablas.

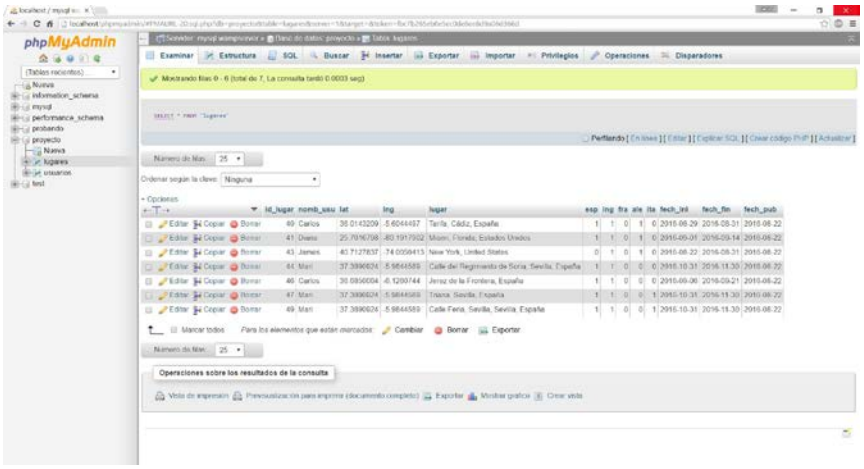

Figura 3-4. Administrador de la tabla “lugares”.

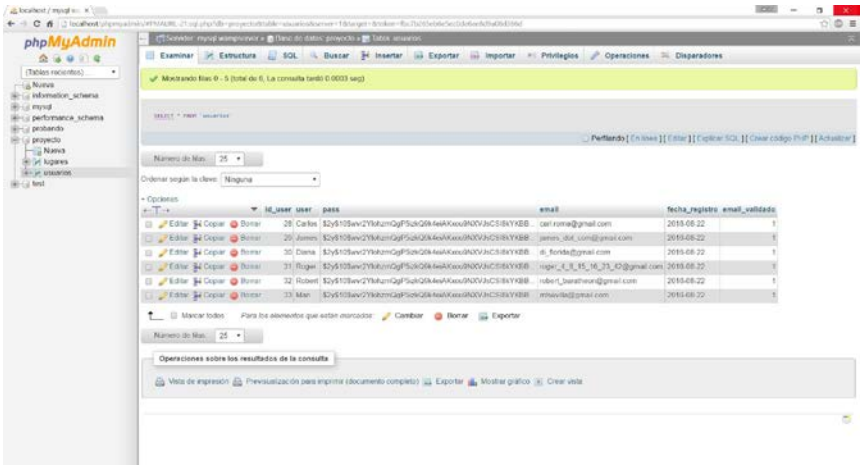

Figura 3-5. Administrador de la página “usuarios”.

4 MODELO ESTRUCTURAL DEL CÓDIGO

Como la mayoría de sitios en internet, esta plataforma web se compone de varios archivos con diferentes contenidos por los que el usuario podrá navegar para interactuar con la misma.

Este apartado está compuesto por dos secciones. En el primero hablaremos sobre la organización de los documentos que componen la plataforma y en el segundo entraremos algo más en detalle sobre dichos documentos de forma independiente, sobre las funciones que se han creado, para qué sirven y qué códigos de error pueden lanzar, además de hablar sobre la base de datos y las dos tablas que se han creado para el proyecto.

4.1 Estructura de la plataforma

La plataforma web se compone de 18 archivos propios que contienen todo el código, tanto html, como php, javascript y estilos css. Además de estos 18 documentos, y como se ha mencionado en el apartado 2.2, también se han incluído 3 librerías o clases externas, una para la detección del tipo de dispositivo, otra para la geocodificación y otra para el envío de correos electrónicos en php.

De los 18 archivos propios no todos son accesibles para el usuario, ya que algunos solo ejecutan código en el servidor que puede servir para la validación de los datos enviados en los formularios, o simplemente para conducir al usuario a otro documento en función del procesamiento de los datos que se hayan tratado.

Separaremos el mapa de la plataforma web en 2 grupos, siendo el primero el de un usuario que no está registrado en ella, al que llamaremos “invitado” y que tendrá limitaciones a la hora de interactuar con la página, y el segundo el de un usuario registrado y logueado que tendrá acceso a todas las funciones de la web. Como las páginas web no almacenan información se hace uso de cookies para guardar la sesión del usuario, almacenando en el navegador su nickname o nombre de usuario con el que se registró.

4.1.1 Usuario “invitado”

Como hemos dicho antes, el invitado es el usuario que no se ha registrado o que no ha iniciado sesión en la página. Este usuario no logueado tendrá limitaciones a la hora de interactuar con la web, podrá ver las publicaciones del resto de usuarios pero no podrá publicar sus propios destinos para que los demás puedan ver los suyos propios.

El mapa de los archivos navegables que tendrá el usuario invitado, partiendo de un archivo *.php[1] será el siguiente:

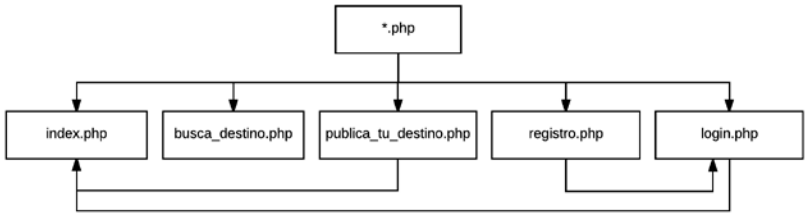

Figura 4-1. Mapa de archivos del tipo de usuario "invitado".

4.1.2 Usuario logueado

El usuario logueado tendrá pleno acceso a la plataforma web. Podrá ver publicaciones de otros usuarios y también publicar las suyas propias. Los documentos "visibles" partiendo de *.php[2] por los que podrá moverse son:

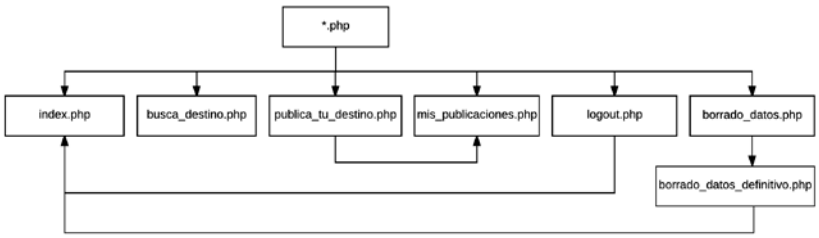

Figura 4-2. Mapa de archivos de un usuario logueado.

Como podemos observar, la principal diferencia es que un usuario logueado puede añadir sus propias publicaciones y tiene una sección en la que podrá administrar todo lo que haya publicado anteriormente.

También se han sustituido registro.php y login.php por logout.php y borrado_datos.php ya que no tendría sentido que un usuario logueado se registrase de nuevo o intentase iniciar sesión por segunda vez sin haberla cerrado previamente.

4.2 Detalle de los documentos y base de datos

Como hemos dicho anteriormente, la plataforma web se compone de 18 archivos propios y 1 base de datos con 2 tablas. Esta sección detalla el funcionamiento de cada archivo por orden alfabético, qué funciones utiliza

[1] Al poner *.php se hace referencia a cualquier otro documento de los mostrados en la figura.
[2] En este caso *.php hace referencia al resto de archivos mostrados en la figura exceptuando a logout.php, que elimina la sesión de usuario y cambia directamente a index.php.

y cómo interactúa con los demás. Todos los documentos por los que el usuario pueda navegar tienen en común los elementos de menú, así que se omitirán en la descripción de cada uno. En todos los documentos que lo requieran (para mostrar el nombre del usuario, para poder publicar, para borrar una publicación, etc), se comprueba que la cookie del usuario está activa ya que puede caducar tras un tiempo.

4.2.1 borrado_datos.php

Este archivo muestra al usuario un campo en el que debe introducir su contraseña. Al pulsar sobre borrar se envía con el formulario la cookie de usuario y la contraseña hacia el documento borrado_datos_definitivo.php.

4.2.2 borrado_datos_definitivo.php

Cuando este documento recibe información del formulario del archivo borrado_datos.php, ejecuta la función comprueba_credenciales(usuario, contraseña) del archivo DataBase.php. Si los credenciales del usuario son correctos se ejecuta la función borra_todo(usuario), que eliminará todas las publicaciones del usuario en cuestión, además de toda la información que haya sobre él en la base de datos (email, usuario, contraseña, etc).

Independientemente de si los datos fueron o no correctos, se volverá a la página index.php enviando por GET (mediante la URL) el estado de la ejecución.

4.2.3 borrado_publicacion_definitivo.php

Este documento se comporta de forma similar a borrado_datos_definitivo.php. Cuando se recibe información del formulario del archivo borrar_publicacion.php, ejecuta la función comprueba_credenciales(usuario, contraseña), si los datos coincidieron se ejecuta la función borra_publicación(lugar), donde lugar es el identificador de la publicación que se va a eliminar de la base de datos. Se creará una variable de estado que indicará si el proceso de ha llevado a cabo o no.

Independientemente de si se ha borrado o no, se mandará al usuario a la página mis_publicaciones.php añadiendo por GET la variable de estado que indicará el resultado de la ejecución.

4.2.4 borrar_publicacion.php

Este archivo es equivalente a borrado_datos.php. Muestra un campo para introducir la contraseña que se enviará junto a la cookie de usuario en el formulario tras hacer click en Borrar y con destino al documento borrado_publicacion_definitivo.php.

El requisito de la contraseña a la hora de borrar algo sirve por si el usuario olvidó cerrar sesión en un ordenador compartido, y otra persona intentase utilizarla de forma fraudulenta para eliminar sus datos.

4.2.5 busca_destino.php

Este documento contiene un formulario con un campo tipo texto que se autocompleta con el buscador de lugares de google, un campo tipo fecha y un checkbox para cada idioma. El formulario se envía a sí mismo, ejecutando la función destinos_encontrados(lugar, idioma, fecha) para mostrar las publicaciones que coincidan con los filtros introducidos por el usuario que está utilizando el buscador.

Tanto si se ha recibido un formulario como si todavía no se ha hecho, se ejecuta la función destinos_publicados() del archivo DataBase.php.

Este documento no manda al usuario a ninguna otra página salvo que él mismo utilice los botones de menú para desplazarse hacia otro lugar.

4.2.6 comprueba_usuario.php

Lo primero que realiza este archivo es la ejecución de la función borra_no_validados() para hacer limpieza de la base de datos de usuarios. Este documento recibe información de un formulario desde el archivo login.php y ejecuta la función comprueba_credenciales(usuario, contraseña).

Si el contraste con la base de datos es positivo, se setea la cookie de usuario con el nombre del usuario y se vuelve al a página index.php. En caso contrario, no se hace nada con la cookie y se vuelve a la página login.php enviando el estado de la operación en la URL mediante GET para informar al usuario de que algo ha ido mal.

4.2.7 DataBase.php

Este documento es el más extenso de todo el programa. Contiene todas las funciones que hacen uso de la base de datos ya sea para comprobar información, como para introducir datos nuevos en la misma. A continuación se explicará la utilidad de cada función.

Todas las funciones que utilizan la base de datos deben conectarse a la misma antes de hacer uso de ella, y cerrar la conexión al terminar de utilizarla. Como esto es común a todas las funciones de este documento, se omitirá en la descripción individual de cada una de ellas.

Para lanzar una orden a la base de datos, se utiliza la función PHP mysqli_query($link, “sentencia SQL”)

- **insertar_destino(usuario, latitud, longitud, lugar, [idiomas], fecha inicio, fecha fin):** inserta en la base de datos una nueva publicación de un usuario.
- **destinos_publicados( ):** para que la página web elimine automáticamente las publicaciones cuya fecha de finalización es anterior al día actual, esta función ejecuta una sentencia SQL que las elimina.

Tras esto ejecuta una sentencia SQL que devuelte todas las publicaciones que haya en la base de datos, y le muestra al usuario las 5 publicaciones más próximas temporalmente.

- **comprueba_email(email):** comprueba si la dirección de correo electrónico que se está intentando registrar ya existe en la base de datos para que una misma dirección no se pueda utilizar más de una vez.
- **comprueba_user(usuario):** igual que comprueba_email(email) pero con el usuario.
- **insertar_usuario(usuario, hash, email):** inserta un nuevo usuario en la base de datos y setea la fecha de registro en ese mismo instante.
- **comprueba_credenciales(usuario, contraseña):** comprueba si hay algún usuario validado (con el bit de validación a 1). Si encuentra un usuario, contrasta la contraseña que se le ha pasado a la función con la almacenada en la base de datos. Si toda la verificación es correcta, devuelve true, en caso contrario, la función devuelve false.
- **destinos_encontrados(lugar, idioma, fecha inicio):** haciendo uso de la clase GoogleMapsGeocoder busca la ciudad correspondiente a la dirección que se le ha pasado mediante la variable "lugar".
  Para salvar el problema del idioma en el que se escriba el nombre de la ciudad o dirección (London distinto de Londres), el programa primero obtiene la latitud y longitud de la ciudad a la que corresponda el lugar.
  Esto se hace recorriendo el valor devuelto por la clase GoogleMapsGeocoder hasta encontrar un tipo igual a 'locality' (explicado en el apartado 2.2.2). En caso de introducir una comunidad autónoma, país, estado, o algo que sea de un tipo superior a ciudad, el programa devolverá un error indicándole al usuario que no ha introducido un lugar válido.
  Si el lugar introducido es correcto, se comprueban los idiomas que ha marcado el usuario y con esto y la fecha de inicio se hace la consulta a la base de datos. Si se encuentran resultados se muestran al usuario ordenándolos por la fecha de finalización más próxima.
- **valida_final(email, usuario, contraseña):** cuando un usuario se registra queda pendiente el proceso de validación del correo electrónico. Si la validación no se hace antes del día siguiente, este usuario será eliminado de la base de datos. Esta función comprueba que el email, el usuario y la contraseña que existen en la base de datos se corresponden con las que se le han pasado. En caso positivo actualiza el bit de validación de dicho usuario poniendo su valor a 1 (originalmente está a 0) y devolviendo el valor true. En caso negativo no se actualiza nada en la base de datos y se devuelve false.
- **borra_no_validados( ):** elimina de la base de datos todos los usuarios que no hayan sido validados antes del día en el que se ejecute la función.
- **borra_todo(usuario):** elimina de las tablas usuarios y lugares todo el contenido publicado por el usuario que ha solicitado el borrado de sus datos.

- **muestra_mis_publicaciones(usuario):** esta función se encarga de hacer la consulta a la base de datos y mostrar al usuario todas las publicaciones que tiene activas ordenadas por fecha de finalización. Si el usuario todavía no ha publicado nada, esta función le mostrará el mensaje de que todavía no ha publicado nada.
- **borra_publicacion(lugar):** la variable lugar que se le pasa a esta función no es la dirección, sino el identificador de la publicación en la tabla lugares. Con este identificador, la función elimina de la base de datos la publicación que el usuario ha solicitado borrar.
- **cuenta_publicaciones( ):** cuenta las filas que tiene la tabla lugares para saber el número total de publicaciones que están activas en la plataforma web. Este número es el que se muestra en la página principal index.php.

4.2.8 index.php

Este documento contiene una imagen y una breve descripción de la utilidad de la plataforma web junto al número de publicaciones activas que se han contado en el momento de cargar la página. En función de si el usuario se ha logueado o no, se mostrarán diferentes elementos en el menú, como se ha explicado en los apartados 3.1.1 y 3.1.2. Estos menús son comunes al resto de páginas visibles por el usuario.

4.2.9 inserta_destino.php

Este documento no es visible por el usuario. Su función es comprobar que los datos que se han introducido en el formulario procedente de publica_tu_destino.php son válidos para ser introducidos en la base de datos. Para ello, el documento comienza comprobando si se han recibido todos los datos necesarios para la publicación. Si se han recibido todos los datos necesarios comprueba que la fecha de inicio se anterior a la fecha de fin, puesto que no tendría sentido el caso contrario. Una vez hecho esto, el grueso de este documento se centra en comprobar que se ha introducido una ubicación válida para ser publicada (que el lugar se encuentre en la base de datos de google y que no sea un lugar de un tipo mayor que una ciudad).

Si el lugar es correcto, se extrae de él la latitud y la longitud correspondiente a su ciudad, se extraen los idiomas que se han marcado en el formulario y se llama a la función insertar_destino(usuario, latitud, longitud, lugar, [idiomas], fecha inicio, fecha fin), que se encarga de introducir los datos en la tabla lugares.

Tanto si la publicación ha sido exitosa como si no lo ha sido, el documento finaliza enviando al usuario a la URL correspondiente a resumen_publicacion.php y añadiendole al final la variable de estado correspondiente al resultado de la ejecución del documento. Esto lo obtendrá el nuevo documento mediante GET.

4.2.10 login.php

En este archivo mostrará al usuario un formulario para darle acceso a la plataforma web. En el formulario están los campos obligatorios para introducir el nombre de usuario y contraseña. Cuando se rellenan los datos y se hace click sobre el botón Iniciar, se cambia al documento comprueba_usuario.php.

Como los campos son obligatorios (etiqueta required en HTML5), si alguno de ellos no se rellena antes de pinchar en el botón se mostrará un aviso al usuario indicando que debe rellenar todos los campos antes de poder inciar sesión.

4.2.11 logout.php

La única función de este documento es eliminar la cookie de usuario almacenada en el navegador del usuario seteándola a un valor vacío y con tiempo de caducidad -1. Después de esto lo devuelve a la página principal index.php.

4.2.12 mis_publicaciones.php

Se encarga de mostrar al usuario las publicaciones que ha realizado anteriormente. Para ello, si la cookie de usuario está activa, llama a la función muestra_mis_publicaciones(usuario) que se encarga de todas las comprobaciones necesarias y de imprimir por pantalla el resultado.

Además de esto, este documento también recibe mediante la variable $_GET el estado de un intento de borrado de una publicación. Si el usuario introdujo correctamente la contraseña mostrará un aviso con la función alert(string) de javascript, que le notifica que la publicación ha sido eliminada con éxito. En caso contrario mostrará que la contraseña introducida no era la correcta, invitándolo a que lo intente de nuevo.

4.2.13 publica_tu_destino.php

Este documento contiene el formulario necesario para que un usuario realice una nueva publicación. Todos los campos del formulario son obligatorios. El lugar se autocompleta con la base de datos de lugares de google. Haciendo uso de HTML5 se consigue que el usuario no pueda introducir una fecha anterior al día actual en los campos fecha inicio y fecha fin. Además de estos campos, este documento tiene un checkbox para cada idioma, debiendo marcar al menos uno de ellos para que la publicación sea válida. Cuando el usuario pinche en el botón Publicar, se le mandará al documento inserta_destino.php que se encargará de insertar en la base de datos la nueva publicación, siempre que todos los datos sean válidos.

A continuación del formulario, se hace uso de la función destinos_publicados( ) para sugerir al usuario las 5 publicaciones que caducarán próximamente.

4.2.14 registra_usuario.php

Lo primero que hace este archivo es una limpieza de usuarios no validados en la base de datos llamando a la función borra_no_validados( ).

Una vez completada la limpieza, este documento se encarga de manejar los datos recibidos en el formulario 'registro' del documento registro.php. Para ello comprueba si se ha recibido el propio formulario, si el formulario contiene una dirección de correo electrónico, un nombre de usuario y una contraseña. Si se han recibido correctamente todos los datos, comienza el proceso de validación de los datos. Lo primero es comprobar si no existía un usuario asociado a la misma cuenta de correo electrónico o con el mismo nombre de usuario. Esto se hace llamando a las funciones comprueba_email(email) y comprueba_user(usuario) que ya han sido explicadas en 3.2.7.

Si no ha habido problemas, lo siguiente es hacer uso de la clase PHPMailer para enviar un correo de validación a la dirección con la que el usuario se ha registrado. Para ello se ha creado una cuenta de GMAIL específica que será la encargada de enviar los emails, evitando tener que crear un servidor de correo electrónico propio. El funcionamiento de la clase PHPMailer ya ha sido explicado en el apartado 2.2.4, así que omitiremos aquí la explicación de esta parte del documento.

Con la contraseña utilizada por el usuario, se crea un hash utilizando la función password_hash(contraseña, algoritmo de encriptacion, opciones) de PHP, haciendo uso del algoritmo blowfish y con opciones de coste de 10 (este coste indica en potencia de 2 cuantas iteraciones utilizará el algoritmo para calcular el hash, cuanto más alto, más seguro pero más trabajo para el servidor, luego hay que buscar un punto óptimo para no saturar al servidor).

Si todo ha salido bien, se llama a la función insertar_usuario(usuario, hash, email) y se enviará al usuario a la página login.php para que se loguee tras haber validado su cuenta. Si algo ha ido mal, se enviará al usuario de nuevo a la página de registro para que vuelva a rellenar el formulario. En cualquiera de los dos casos, en la URL se añade la variable de estado para que la página destino obtenga el resultado de la ejecución de este documento mediante $_GET y sepa qué ha salido mal para notificárselo al usuario.

4.2.15 registro.php

Este documento contiene el formulario que se enviará a registra_usuario.php donde todos los campos son obligatorios. Contiene 4 campos, uno de tipo email que gracias a HTML5 verificará que el formato de la dirección de correo electrónico introducida es correcto, otro de tipo texto donde el usuario introducirá el nombre con el que quiere ingresar en la página web y dos campos tipo contraseña, el primero será el que se envíe, y el segundo se utilizará para que el usuario compruebe si la contraseña que quería ingresar es correcta. Esto se hará utilizando la función propia compara_pass( ), que mostrará cada vez que el usuario teclee algo en en cualquiera de los campos de contraseña un texto en verde si las dos contraseñas coinciden, o en rojo en caso

contrario.

El formulario sólo se enviará si todos los campos se han rellenado y si las dos contraseñas coinciden.

4.2.16 resumen_publicacion.php

Al intentar hacer una publicación, el usuario será redirigido a este documento. Básicamente este archivo es un switch que, en función del contenido de la variable de estado que recibe en la variable $_GET, avisa al usuario mediante la función alert(string) del resultado de su publicación.

Si la publicación ha sido exitosa, el mensaje mostrado indica que todo ha ido bien y redireccionará al usuario hacia la página mis_publicaciones.php donde podrá ver la nueva publicación junto a todas las anteriores.

Si la publicación no se ha completado, se indicará el motivo por el que no ha podido realizarse y mandará al usuario nuevamente al formulario contenido en publica_tu_destino.php para que lo intente de nuevo.

4.2.17 style.css

Este documento contiene los estilos utilizados para darle formato a todos los documentos que forman la plataforma web. Los estilos se han dividido en 5 partes en función de qué tipo de elementos formateen y de qué parte de la visualización del navegador modifiquen:

- **Etiquetas de los documentos:** modifican los estilos por defecto de las etiquetas propias del lenguaje html. En el caso de este proyecto se han modificado los estilos de las etiquetas “html”, “body” y “a”. El motivo de modificar las etiquetas html y body ha sido para poder situar en el orden deseado los bloques que luego componen el resto de las páginas, además de para insertar una imagen en el fondo del documento.
- **Bloque de cabeceras:** en esta parte se han situado los estilos que dan formato a la cabecera de todos los documentos. Sirven para organizar los elementos en la parte superior de los documentos y para dar estilo a los elementos de menú.
- **Bloque central:** esta parte contiene los estilos de los elementos que se mostrarán en el centro del navegador, como pueden ser las publicaciones o los formularios.
- **Bloque pie:** contiene los estilos de la parte inferior de los documentos. Sirve para organizar los elementos que se mostrarán al final de las páginas en el navegador.
- **Elementos de librerías externas:** esta parte formatea los elementos que se utilizan de librerías externas, como puede ser el searchbox de google y los resultados que se muestran para autocompletar el campo del formulario.

4.2.18 valida_email.php

A este documento se accede a través de un enlace que se envía a la dirección de correo electrónico con la que se ha registrado el usuario.

Aquí encontramos un formulario donde habrá que introducir el nombre de usuario que se indicó en el formulario de registro y la misma contraseña, asociados al email que se va a validar. Cuando se hace click en el botón Validar del formulario, se envía al usuario al mismo documento, que en esta ocasión detectará que se le ha enviado el formulario de validación y pasará a comprobar que los datos introducidos son correctos haciendo uso de la función valida_final(email, usuario, contraseña).

El estado de la validación se le mostrará al usuario con la función alert(string) de javascript. Si el resultado de la validación fue positivo, se le enviará a la página login.php para que pueda iniciar sesión ya como usuario registrado en la plataforma web, y si la validación no se ha podido completar con éxito, se le enviará a la página principal, index.php, indicando que intente de nuevo el proceso de validación.

4.2.19 Base de datos

Para este proyecto se ha utilizado una base de datos llamada “proyecto” que contiene dos tablas, “usuarios” y “lugares”. A su vez, cada tabla contiene una serie de campos que es donde se almacena la información de los usuarios y sus publicaciones.

La tabla usuarios contiene los siguientes campos:

- **id_user:** este campo es de tipo entero (int) y es autoincremental, siendo 0 el primer usuario que se registre, y creciendo de 1 en 1 según se van insertando usuarios en la tabla.
- **user:** en este campo se almacena el nombre del usuario en forma de caracteres (varchar).
- **pass:** aquí va almacenado el hash calculado a partir de la contraseña del usuario (varchar).
- **email:** este campo contiene el email con el que se ha registrado un usuario. Al igual que el nombre del usuario, el email se almacena en forma de caracteres (varchar).
- **fecha_registro:** campo de tipo fecha (date), se rellena automáticamente con la fecha del servidor cuando un usuario se registra en la página web.
- **email_validado:** el tipo de este campo es entero de un solo bit (tinyint(1)). Este campo se inicializa a 0 en el momento de registro y se actualiza a 1 cuando se ha completado el proceso de validación del email de un usuario.

La tabla lugares está formada por los siguientes campos:

- **id_lugar:** es el identificador de una publicación. Al igual que id_user, es de tipo entero (int) y es autoincremental, aumentando se número cuando se inserta una nueva fila en la tabla.

- **nomb_usu:** al igual que el campo user de la tabla usuarios, esta columna almacena el nombre del usuario que publicó el lugar. (varchar)
- **lat:** guarda la latitud de la ciudad asociada al lugar (varchar).
- **lng:** guarda la longitud de la ciudad asociada al lugar (varchar).
- **lugar:** almacena la dirección original que el usuario utilizó a la hora de realizar la publicación. (varchar)
- **esp:** entero de un solo bit (tinyint(1)) que está a 1 si el usuario ha marcado que habla español, o a 0 si no lo ha hecho.
- **ing:** entero de un solo bit (tinyint(1)) que está a 1 si el usuario ha marcado que habla inglés, o a 0 si no lo ha hecho.
- **fra:** entero de un solo bit (tinyint(1)) que está a 1 si el usuario ha marcado que habla francés, o a 0 si no lo ha hecho.
- **ale:** entero de un solo bit (tinyint(1)) que está a 1 si el usuario ha marcado que habla alemán, o a 0 si no lo ha hecho.
- **ita:** entero de un solo bit (tinyint(1)) que está a 1 si el usuario ha marcado que habla italiano, o a 0 si no lo ha hecho.
- **fech_ini:** campo de tipo fecha (date) que guarda la fecha de comienzo de una publicación.
- **fech_fin:** campo de tipo fecha (date) que guarda la fecha de fin de una publicación.
- **fech_pub:** campo de tipo fecha (date) que guarda la fecha en la que se publicó un lugar en la base de datos.

5 MANUAL DE USUARIO

En este apartado veremos el funcionamiento de la plataforma web de una forma guiada a través de capturas. Para la toma de las capturas se han creado varios usuarios ficticios y publicaciones específicas para explicar el comportamiento de la página ante varias situaciones.

Para acceder a la página principal, ejecutamos el programa WAMPServer en Windows, abrimos un navegador web y accedemos a la dirección del proyecto (en este caso se ha utilizado Google Chrome, pero se ha escrito el código para ser compatible con la mayoría de navegadores actuales). Como estamos accediendo a la plataforma web desde el mismo ordenador en el que se está ejecutando el servidor, podemos acceder mediante "localhost/proyecto/".

Comenzaremos la guía partiendo de un usuario "invitado" que no tiene acceso a publicar contenido en la web.

Figura 5-1. Vista de index.php desde usuario no logueado.

En la cabecera podemos observar los elementos de menú para navegar por las diferentes páginas de la plataforma web. En el centro vemos una imagen junto a una breve descripción del objetivo del proyecto, además del número de publicaciones activas que hay en ese momento.

Figura 5-2. Campos del formulario de registro.php.

Aquí se muestra el formulario de registro donde habrá que introducir 4 campos obligatorios antes de enviarlo.

Figura 5-3. Comienzo de escritura de segunda contraseña.

Cuando comenzamos a escribir el segundo campo contraseña, vemos que aparece un texto en color rojo que indica que las contraseñas son distintas. Este mensaje se mantiene hasta que las contraseñas coinciden, como se muestra en la siguiente captura.

Figura 5-4. Cambio de mensaje cuando las contraseñas coinciden.

Una vez completados los 4 campos y teniendo el mensaje verde que confirma que las contraseñas son idénticas, pinchamos sobre el botón Registrar.

Figura 5-5. Alerta de envío de correo electrónico.

Si la dirección de correo electrónico y el nombre de usuario que hemos introducido no existían en la plataforma, se mostrará una alerta indicando que se ha enviado el correo electrónico para la validación de la cuenta.

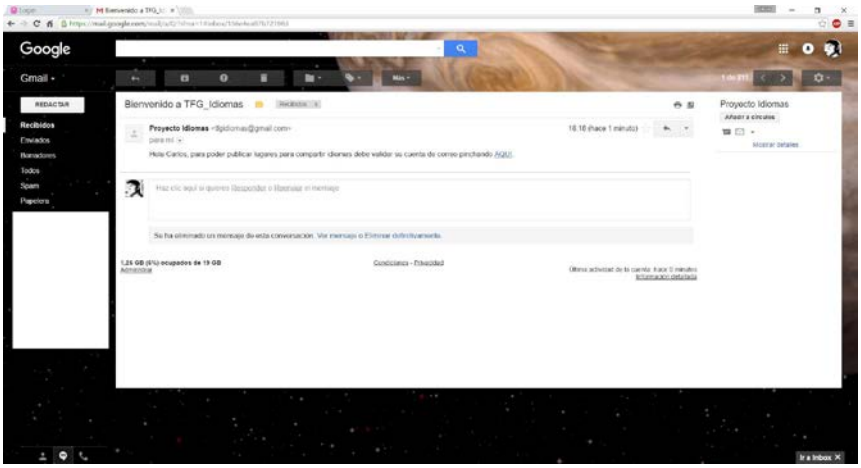

Figura 5-6. Correo electrónico recibido con enlace de validación.

En el correo electrónico se nos indica un enlace que tendremos que seguir para poder validar la cuenta.

Figura 5-7. Formulario de validación de email.

Al seguir el enlace que recibimos por correo se nos pide que introduzcamos nuevamente el nombre de usuario y la contraseña.

Figura 5-8. Cuenta de correo validada correctamente.

Si los datos asociados a esa cuenta de correo son correctos, se mostrará un mensaje que indica que la validación de la cuenta ha sido completada con éxito.

Figura 5-9. Formulario de login.php.

Una vez validado ya podremos iniciar sesión con nuestro usuario desde el formulario de la página login.php.

Figura 5-10. Vista de index.php desde usuario logueado.

Cuando iniciamos sesión de forma correcta volvemos a la página index.php. Al estar logueados se muestra el nombre de usuario que estamos utilizando, además de tener acceso a “Mis publicaciones” y a poder borrar los datos del usuario, además de poder cerrar sesión.

Ya tenemos un usuario válido que pueda utilizar por completo la plataforma web. Seguiremos navegando por la plataforma, expondremos varias situaciones y veremos cómo se comporta la página en cada caso.

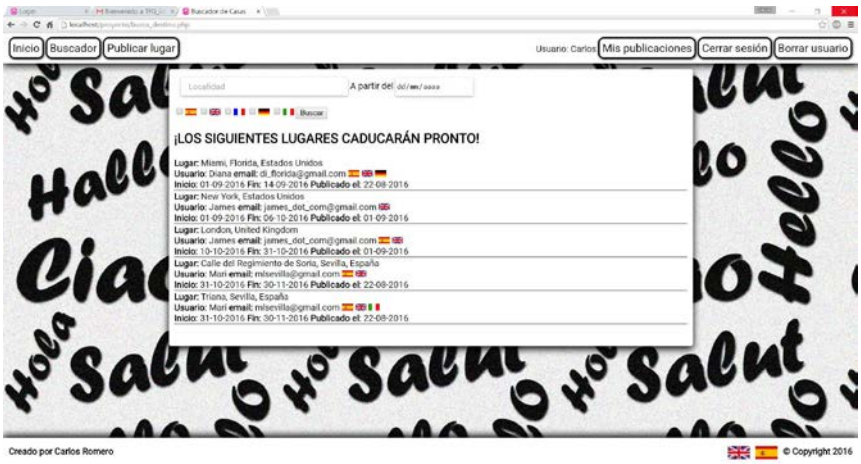

Figura 5-11. Buscador del documento busca_destino.php.

En el buscador vemos el searchbox de Google para introducir una dirección o ciudad, un campo donde tendremos que insertar la fecha a partir de la cual queremos buscar un lugar, y varios checkbox para marcar los idiomas. Tendremos que marcar al menos uno de los idiomas para que la página nos muestre algún resultado.

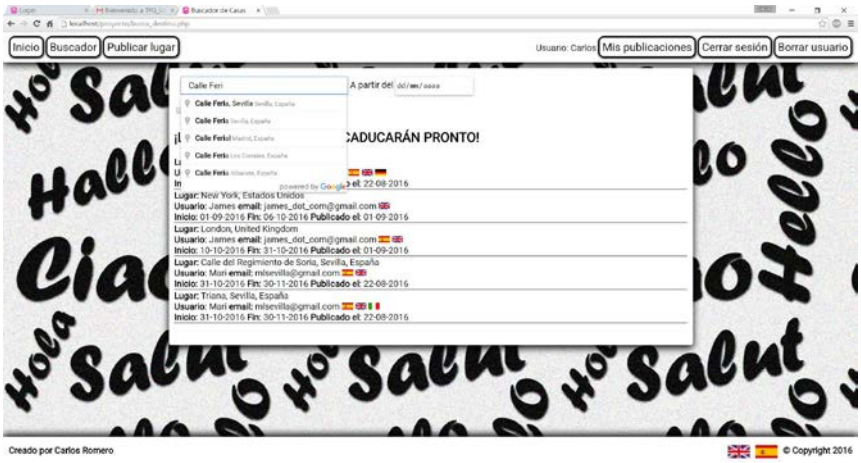

Figura 5-12. Sugerencias desde la base de datos de Google Places.

En la figura 5-12 vemos el menú desplegable con las sugerencias del searchbox que hace uso de la base de datos de Google Places. Podemos observar que al empezar a escribir el nombre de una calle se muestran 5 sugerencias correspondientes a diferentes lugares.

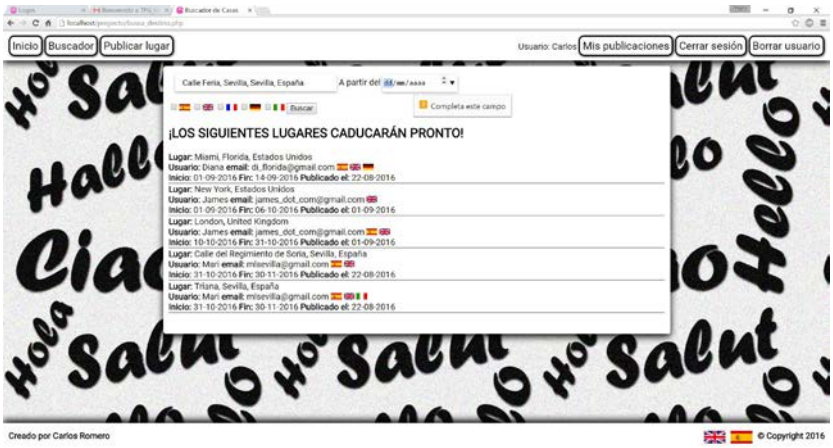

Figura 5-13. Mensaje HTML5 ante un campo obligatorio no rellenado.

En la figura 5-13 vemos que si no rellenamos un campo obligatorio (etiqueta HTML5 required), se muestra un mensaje indicando que se debe completar ese campo antes de realizar el envío del formulario.

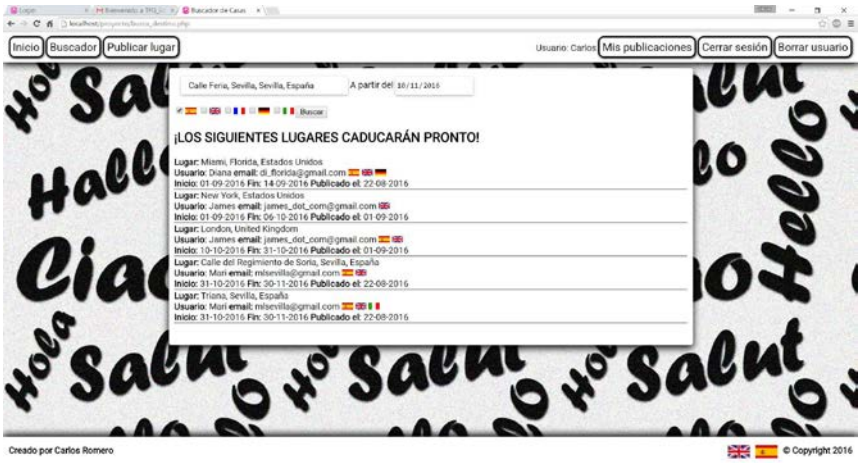

Figura 5-14. Todos los campos completados.

Completamos todos los campos y enviamos formulario para obtener resultados de la ciudad de Sevilla en la fecha seleccionada y donde se hable al menos uno de los idiomas marcados.

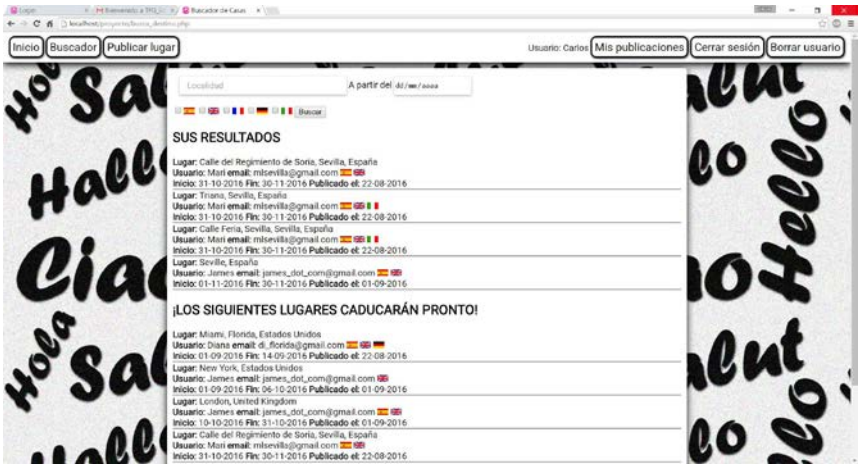

Figura 5-15. Resultado de la búsqueda de Calle Feria.

Vemos en la figura 5-15 que al buscar “Calle Feria, Sevilla”, en los resultados mostrados obtenemos todos los que corresponden a la ciudad de Sevilla. Esto se hace para no limitar el filtro a una única calle, ya que la probabilidad de encontrar a alguien con un filtro tan específico sería muy baja.

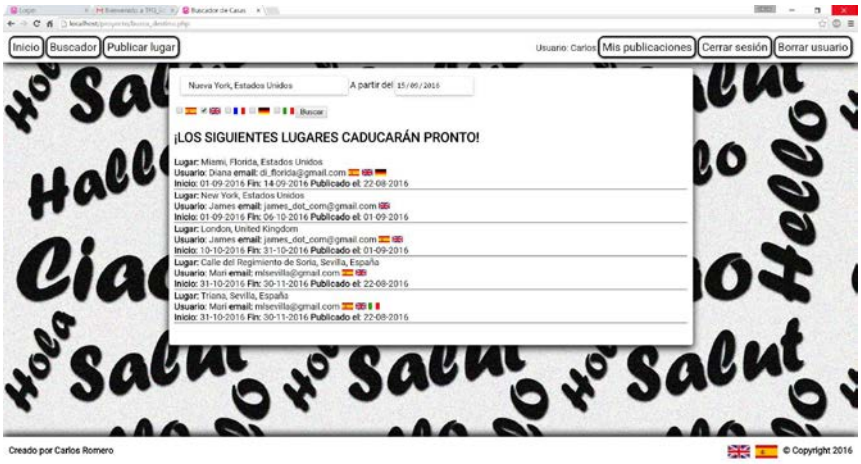

Figura 5-16. Búsqueda de Nueva York (en español).

Como dijimos en apartados anteriores, los lugares publicados se guardan con la latitud y longitud de su ciudad. Esto se hace para salvar el problema de que una ciudad tenga nombres diferentes en varios idiomas, de esta forma, si un usario realiza una búsqueda escribiendo NUEVA YORK como en la figura 5-16, obtendrá todos los resultados que correspondan con esa ciudad en cualquier otro idioma.

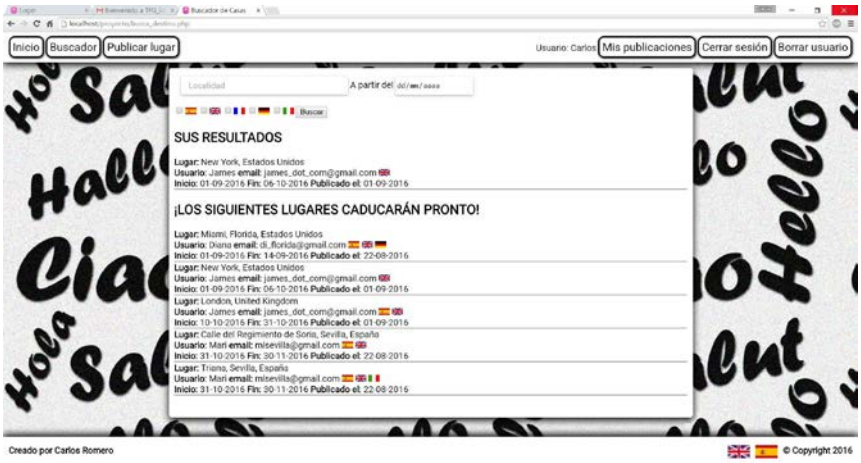

Figura 5-17. Resultado New York (en inglés).

En la figura 5-17 podemos ver que al buscar NUEVA YORK obtenemos también NEW YORK como resultado.

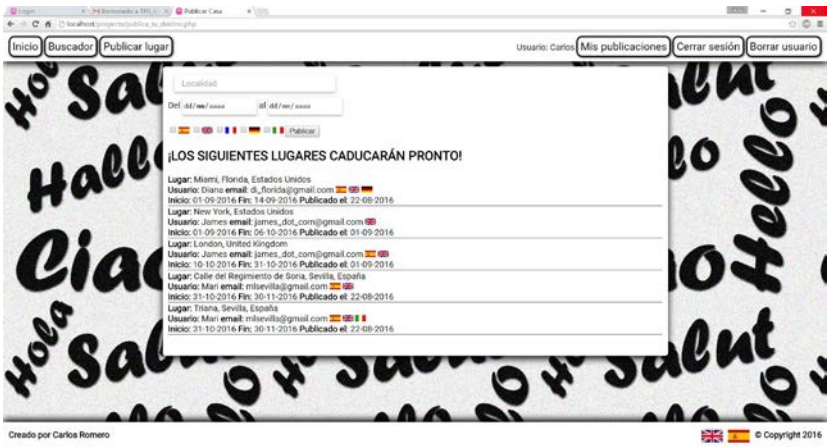

Figura 5-18. Vista de publica_tu_destino.php

El formulario para publicar es similar al buscador. En este caso disponemos de dos campos para fechas, siendo el primero de ellos para la fecha de comienzo y el segundo para la fecha de finalización.

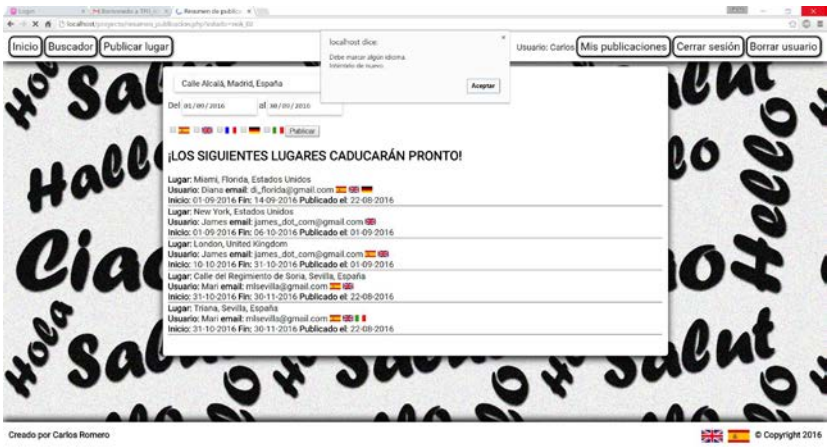

Figura 5-19. Error por no marcar idiomas.

Aquí también tendremos que marcar al menos uno de los idiomas para poder crear nuestra publicación.

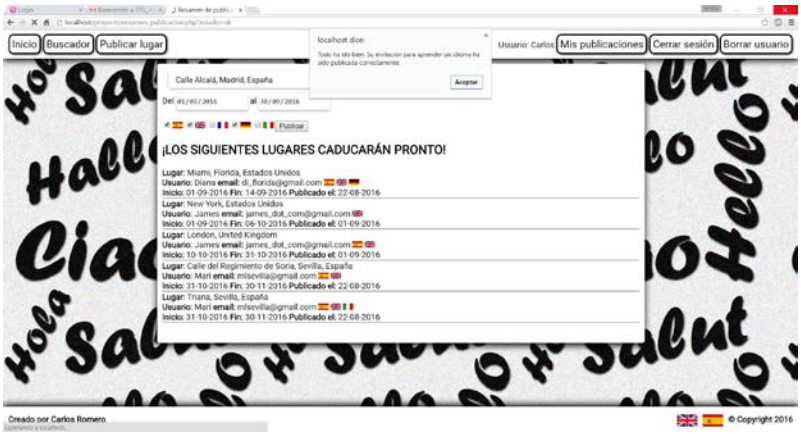

Figura 5-20. Publicación correcta.

Cuando hemos completado todos los campos correctamente y hacemos click sobre Publicar, se inserta en la base de datos nuestra invitación para aprender un idioma, recibiendo el mensaje de la figura 5-20.

Figura 5-21. Vista de mis_publicaciones.php.

Tras publicar una invitación, seremos redireccionados a la vista de “Mis publicaciones”, donde podremos comprobar que nuestra publicación ha sido insertada correctamente.

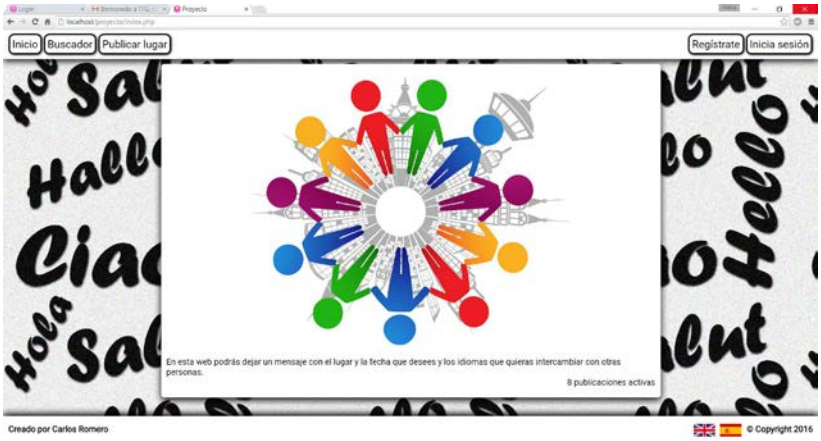

Figura 5-22. Cierre de sesión.

Al cerrar sesión volvemos automáticamente a index.php. Aquí podemos comprobar cómo ha aumentado el contador de publicaciones activas, ya que ahora tiene en cuenta la que acabamos de insertar.

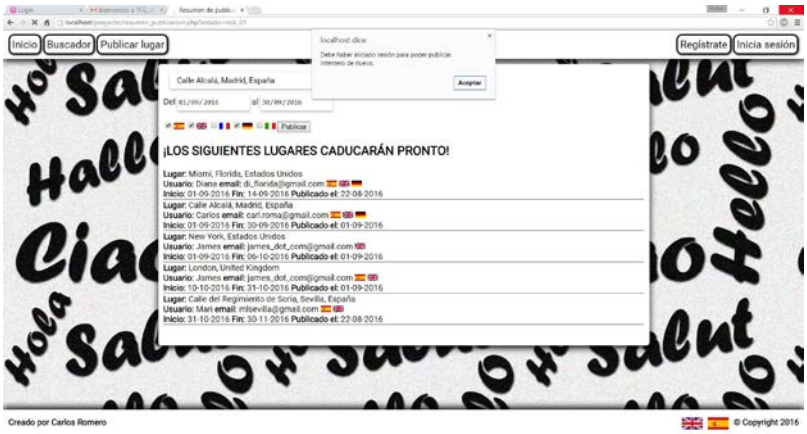

Figura 5-23. Intento de publicación como invitado.

Si intentamos añadir una publicación sin haber iniciado sesión previamente, recibiremos el mensaje que se muestra en la figura 5-23. Volvemos a iniciar sesión para seguir con la siguiente captura e iniciar el proceso de borrado de usuario.

Figura 5-24. Vista de borrado_datos.php.

En el formulario de borrado únicamente se nos pide la contraseña del usuario con el que estamos logueados.

Figura 5-25. Usuario eliminado correctamente.

Si la contraseña es correcta, todas las publicaciones y datos personales del usuario que estaban almacenados en la base de datos son eliminados sin posibilidad de recuperación.

Ahora hacemos click sobre la bandera del Reino Unido para cambiar el idioma al inglés, terminando el manual de usuario con la demostración del borrado de una publicación.

Figura 5-26. Vista de la versión en inglés de index.php.

Ahora iniciaremos sesión con otro usuario, veremos sus publicaciones y borraremos una de ellas.

Figura 5-27. Vista de “Mis publicaciones” en inglés.

Para comenzar el proceso de borrado de una localización, simplemente tendremos que hacer click con el ratón sobre una de las X que se encuentran a la derecha de cada publicación. De esta forma seremos redirigidos al formulario de la siguiente captura.

.

Figura 5-28. Formulario para la eliminación de una publicación.

Al igual que ocurría al intentar borrar un usuario, si la contraseña que se ha introducido es correcta se elimina de la tabla de lugares la publicación correspondiente a la ID que tuviera la opción seleccionada, y se muestra el mensaje que aparece en la figura 5-29.

Figura 5-29. Publicación elminiada con éxito.

6 CONCLUSIONES Y FUTUROS TRABAJOS

Como hemos podido observar, la utilidad actual de la plataforma web es un servicio básico para poner en contacto a personas que tengan interés en compartir sus conocimientos sobre los idiomas que quieren practicar o aprender. En esta primera versión del programa la única forma de ponerse en contacto con el resto de usuarios es a través del correo electrónico validado que proporcionan al registrarse, así que todavía queda mucho trabajo por hacer para mejorar los servicios que se pueden ofrecer.

Dejando pendiente como futuros trabajos, las mejoras que se podrían introducir serían:

- Personalización del perfil de usuario, pudiendo añadir un avatar o imagen de usuario.
- Descripción personal, profesión, intereses, etc., para que el resto de usuarios puedan encontrar personas más afines.
- Nivel de cada idioma.
- Adaptación de la plataforma web a más idiomas además del español y el inglés.
- Incluir más idiomas en los checkbox.
- Incluir mensajes personalizados en las publicaciones creadas por los usuarios.
- Introducir un sistema de valoración entre usuarios que ya se hayan puesto en contacto y que fuera visible para el resto.
- Incluir mensajes privados entre usuarios.
- Crear un método de suscripción para recibir notificaciones ante eventos seleccionados.

Estos futuros trabajos, además de añadir nuevos servicios a la plataforma web, estarían orientados a generar confianza entre los usuarios que inicialmente pudieran sentirse reticentes ante personas desconocidas.

La realización de este proyecto, además de una gran satisfación personal, me ha aportado una gran soltura con los lenguajes de programación web y con el manejo de bases de datos, además de tener que pensar en aspectos de seguridad para la protección de los usuarios. El hecho de utilizar APIs y funciones realizadas por terceras personas, hace que aprendas a buscar mucha información entre la documentación que aporta el creador y, sobre todo, en foros donde otras personas ya se han encontrado con problemas que podrían darse. Todos estos aspectos se trabajan en mayor o menor medida durante la carrera, pero hasta que no realizas un proyecto propio no quedan totalmente asentados.

7 BIBLIOGRAFÍA

Manuales y ejemplos HTML y CSS

http://www.w3schools.com/

Manuales y ejemplos PHP

http://php.net/manual/es/

Documentación Google Geocoding

https://developers.google.com/maps/documentation/geocoding/intro?hl=es - Results

http://jstayton.github.io/GoogleMapsGeocoder/classes/GoogleMapsGeocoder.html

PHPMailer

http://www.desarrolloweb.com/articulos/phpmailer.html

Mobile-Detect

https://github.com/serbanghita/Mobile-Detect