

# MIS VIAJES. APLICACIÓN WEB

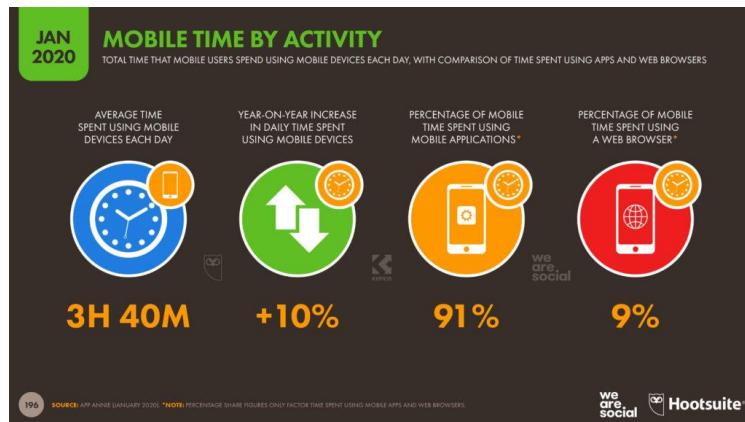
ALEJANDRO CALVO TORIBIO

## Contenido

Introducción.....	2
Objetivos del proyecto.....	3
Viabilidad del proyecto.....	5
Análisis de costes .....	8
Solución planteada .....	9
Medios Materiales Utilizados.....	9
Planificación del proyecto .....	10
Base de Datos.....	11
Planificación.....	11
Creación.....	12
Seguridad.....	15
Despliegue.....	18
GeoJSON.....	21
Desarrollo.....	23
Estructura .....	23
index.html.....	25
introduccion.html .....	28
mapa.html .....	29
destino.html .....	32
Cultura .....	34
login.html.....	36
registro.html.....	39
Responsive.....	41
Funcionalidad .....	48
index.html e introduccion.html .....	48
mapa.htm .....	49
destino.html .....	55
cultura.html .....	56
Funcionalidad de las APIs .....	58
Paises.php.....	58
DBPaises.php .....	60
DB.php .....	63
Destino.php .....	65
Usuarios.php.....	65
DBUsuarios .....	68
Estilo.....	70
Conclusiones .....	73
Bibliografía .....	75
Anexos .....	79

## Introducción

El primer cuarto de siglo XXI se está enmarcando como la era de la digitalización. El 68% de la población mundial cuenta con un teléfono inteligente en la actualidad<sup>1</sup> y, además, las personas utilizan, de media, casi 4 horas al día su *smartphone*.



Además, en la actualidad más de la mitad población mundial utiliza internet<sup>2</sup>. Esto está generando una necesidad exponencial de desarrollo de aplicaciones de *smartphone* y aplicativos web para ofrecer a la ingente cantidad de usuarios, que además utilizan cada vez más tiempo estos medios digitales, una serie de servicios que puedan utilizar en su beneficio o plataformas en las que puedan pasar sus momentos de ocio.

En este contexto nace este proyecto, como desarrollo de un sitio web relacionado con otro previo (aplicación de *smartphone*), que busca meterse en este creciente mercado de entretenimiento: **Mis Viajes Web**. El principal objetivo de esta aplicación web es llenar el deseo de realización de todos los viajeros que existen en el mundo y que disfrutan de su afición de forma continua y reiterada. El turismo, con el gran desarrollo económico, cultural y social de los últimos años, ha crecido exponencialmente y *Mis Viajes* quiere ofrecer un servicio a todos estos potenciales usuarios.

Si tenemos en cuenta los datos de los últimos años<sup>3</sup>, motivados por el surgimiento de la pandemia asociada a la SARS-CoV-2 (COVID-19), existe un gran decrecimiento generalizado motivado por varias circunstancias ineludibles como el miedo o la incertidumbre; sin embargo, hay otras circunstancias eludibles, como pudiera ser el conocimiento de las medidas exigidas por cada uno de los países. De este modo mucha gente que desconoce si puede visitar un país o no, o si debe permanecer 2 semanas en cuarentena a su llegada, es capaz de conocer la circunstancia real y decidir si desea realizar el viaje o no. Además, el conocimiento es poder y no podemos obviar toda la gente que tiene “miedo” de enfermar, pero que confía en las vacunas desarrolladas hasta la fecha y en las medidas gubernamentales como los pasaportes COVID.

Por este motivo **Mis Viajes** tiene un amplio nicho de mercado debido a los 1400 millones<sup>4</sup> de turistas que viajaron en el pasado 2019 pero que es ampliamente superior al

<sup>1</sup><https://mktefa.ditrendia.es/blog/todas-las-estad%C3%ADsticas-sobre-m%C3%B3viles-que-deber%C3%ADas-conocer-mwc19>

<sup>2</sup> <https://datos.bancomundial.org/indicator/IT.NET.USER.ZS>

<sup>3</sup> <https://es.statista.com/temas/3612/el-turismo-en-el-mundo/>

<sup>4</sup> <https://www.epdata.es/datos/turismo-espana-mundo-datos-graficos/272>

adaptarse a los momentos convulsos en los que vivimos para ofrecer información sobre las exigencias COVID y cómo se verían afectados los usuarios.

## Objetivos del proyecto

Como ya se ha mencionado previamente, *Mis Viajes* pretende captar un nicho de mercado establecido por los *wanderlust*<sup>5</sup>, *travelling fugue* o dromomaníacos (en lengua castellana). Estos turistas necesitados de trasladarse continuamente y de conocer todos los rincones del mundo posibles; a los que había que añadir, desgraciadamente, los que tienen la necesidad patológica de aparentar en las redes sociales de manera real o, incluso, falsa<sup>6</sup>.

Teniendo claro el objetivo que pretende el sitio web, habría que plantear el manejo de cómo se va a llevar a cabo y establecer las funcionalidades principales y secundarias.

El eje central del sitio será “**mapa**”, que es una página del sitio en el que se mostrará un mapamundi interactivo en el que se identifican los 195 países reconocidos mundialmente por la ONU<sup>7</sup> (para no entrar en posibles rivalidades territoriales como Kosovo, Taiwán, Sáhara, etc.). El mapa tiene un interés cultural evidente y es que, al ser interactivo, el usuario observa continuamente sobre qué país está situado; sin embargo, la funcionalidad principal es la interacción directa con los países, pudiéndolos marcar el usuario como países en los que ha residido, países en los que ha hecho turismo o, por el contrario, países a los que todavía no ha podido ir, pero tiene claro que desharía hacerlo en un futuro. Es decir, será como un mapa de “rascar” en los que el usuario establece sus viajes.



Si *Mis Viajes* no tuviera otro objetivo principal, sería una aplicación que podría ser “desechada” bastante rápido, puesto que puede pasar un largo tiempo en los que el usuario no realice ningún viaje. Sin embargo, gracias a una segunda funcionalidad principal, no tendría por qué ser así: **destino**. No es lo mismo disfrutar un viaje que conocer qué país debería ser visitado por mi perfil; incluso, pudiera llegar el momento en el que muchos turistas ya no son capaces de decidirse por un nuevo destino.

Este segundo objetivo principal (en cuanto a funcionalidad) es la creación de un formulario que debería cumplimentar el usuario para que se le faciliten las opciones concordantes (posibles futuros destinos del usuario) y pueda elegir el nuevo país que quiere visitar. Es decir, *Mis Viajes* ofrece un formulario con campos que deben ser rellenados (o no, porque no se quiere filtrar por ello) sobre las características de los países, tales como:

<sup>5</sup> <https://lamenteesmaravillosa.com/el-sindrome-wanderlust-la-obsesion-por-viajar/>

<sup>6</sup> <https://www.psicologialflexible.com/es/falsa-apariencia-redes-sociales/>

<sup>7</sup> <https://www.un.org/en/about-us/member-states>

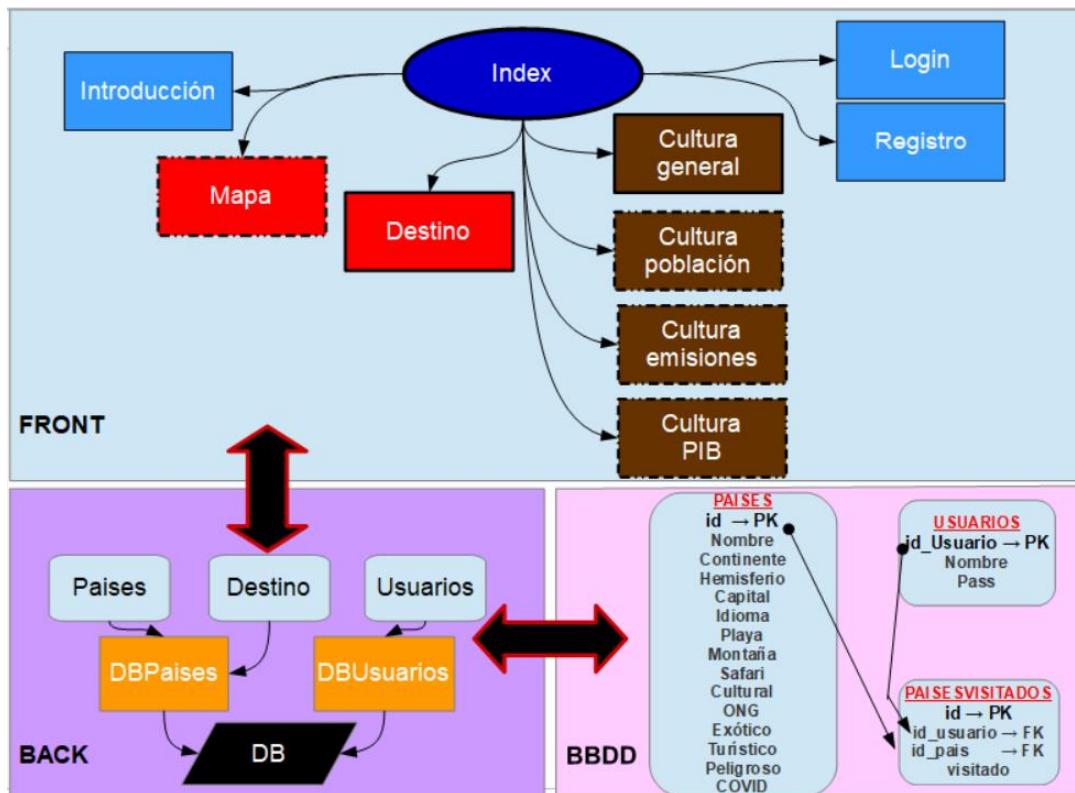
hemisferio, continente, idioma, situación ante la COVID-19 o si cuenta con playa, montaña, si es un país turístico, etc. Cuando el usuario finalice se le devolverán los resultados coincidentes para que elija su nuevo deseo de viaje. Ya sea porque sea un indeciso o porque se le acaben las opciones, **destino** ayudará al usuario.

Las funciones principales del sitio son las dos establecidas *ut supra*, pero también cuenta con una secundaria, que sería el aprendizaje a través del *hobbie*. Es probable que haya mucha gente que haya viajado, pero desconozca completamente la geografía mundial, la situación de los países, si es un país ecológico o no, etc. *Mis Viajes* pretende que a través del interés que genera la pasión por viajar, los usuarios del sitio puedan aprender más sobre el mundo que nos rodea. Para ello existen 2 formas: la primera sería la recopilación de información del país y la muestra de ella en una página de forma estática para la lectura del usuario y, por el contrario, la utilización del motor que se utilizará en “*mapas*” para que, de forma dinámica, el usuario aprenda nueva información como la población mundial, número de emisiones de cada país o la riqueza de los mismos a través del PIB.

Como objetivos del sitio, que no de la funcionalidad, también es importante para *Mis Viajes* que el sitio sea *responsive* y sea visible en todo tipo de dispositivos. Para ello se utilizará *Bootstrap*, que facilita mucho la realización de estos tipos de sitios *responsive*.

Por último, habría que mencionar la parte del *backend* que el usuario nunca llegará a ver pero que es vital que sea desarrollada para la funcionalidad porque de poco serviría que el usuario pudiera marcar los países que ha visitado si cuando cerrara el navegador se perdiera esa información. Para ello se han establecido unas *APIs* en PHP que relacionan mediante AJAX las páginas del *front-end* con una base de datos en las que se encuentran los países almacenados con sus características, y se irán almacenando, insertando o eliminando los usuarios de la aplicación y las visitas que quieran almacenar.

De este modo tenemos un sitio web que sigue este esquema.



## Viabilidad del proyecto

*Mis Viajes* tiene como objetivo la marca de los países visitados de un usuario y la elección de un futuro destino ajustándose a las preferencias que él elija. Para saber la viabilidad del proyecto se debe conocer el **entorno** (microentorno) en el que se va a encontrar.

Si se analiza, en este entorno, la competencia actual se puede observar que, claramente, no existe ese tipo de aplicación específica. Es algo fácilmente comprobable con una sencilla búsqueda en un motor de búsqueda online como *Google*. No hay webs que te ayuden a elegir un país atendiendo a unas características, lo más parecido es atendiendo a la personalidad del usuario, y sin poder marcar la opción en un mapa.

The screenshot shows a Google search bar with the query "a qué país debería ir?". Below the search bar, there are tabs for "Todo", "Noticias", "Imágenes", "Videos", "Maps", and "Más". The "Todo" tab is selected. The search results show approximately 3,380,000,000 results found in 0.46 seconds. The first result is a link to "https://www.traveler.es › Viajeros › Curiosidades" titled "19 países que todo viajero debería visitar | Traveler". The second result is a link to "https://www.nacionrex.com › test › test-a-donde-debo-v..." titled "TEST: ¿Cuál es el país ideal al que debes viajar según tu ...". The third result is a link to "https://zoo.com › what-country-the-world-spanish-zoo" titled "¿Qué país del mundo se ajusta más con tu personalidad? | Zoo". The fourth result is a link to "https://pandemicquiz.com › de-que-pais-eres-de-corazon" titled "¿De qué país eres de corazón? - PandemicQuiz".

Tampoco encontramos la opción de marcar los países que se han visitado, salvo en algún ejemplo básico, antiguo, sin interfaz amigable y sin poder almacenar los datos.

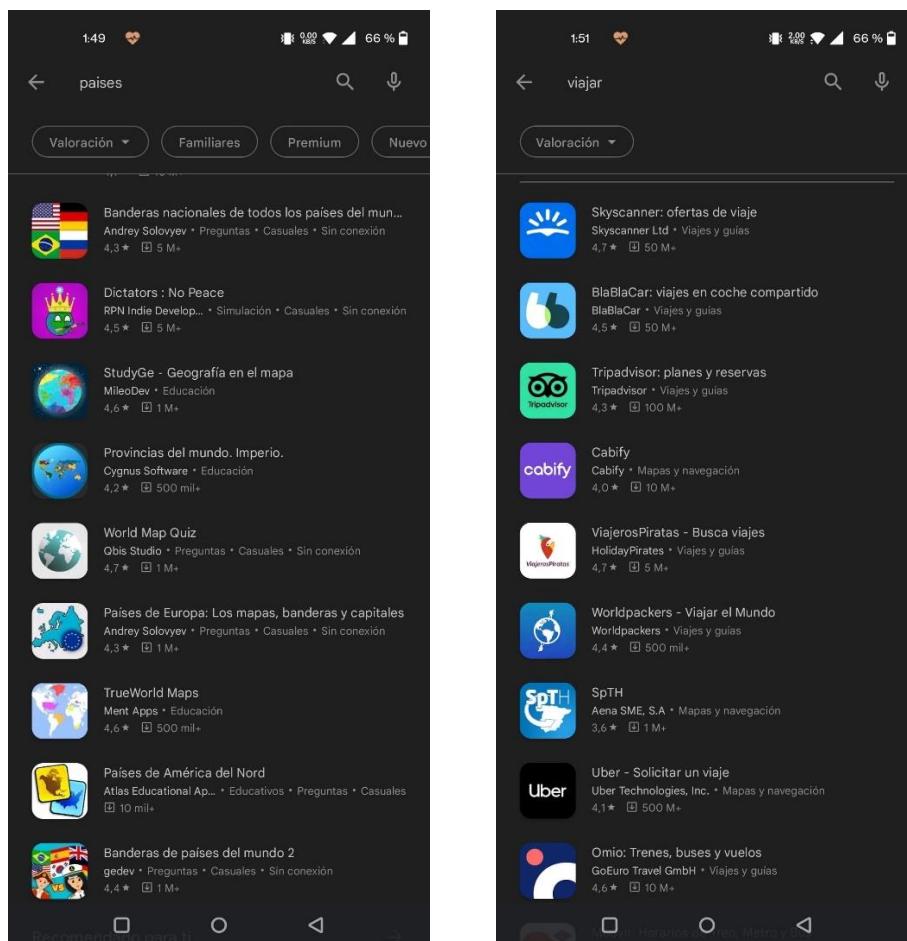
The screenshot shows a Google search bar with the query "seleccionar paises en mapa". Below the search bar, there are tabs for "Todo", "Imágenes", "Noticias", "Maps", "Videos", and "Más". The "Todo" tab is selected. The search results show approximately 20,100,000 results found in 0.63 seconds. The first result is a link to "https://www.genbeta.com › web › con-esta-pagina-podr..." titled "Con esta página podrás crear tus mapas del mundo con ...". The second result is a link to "https://conlamochila.com › crear-mapa-paises-visitados" titled "Cómo crear un mapa de los países visitados - Con La Mochila". The third result is a link to "https://online.seterra.com › vgp" titled "Mundo: Grandes países - Juego de Mapas - Online Seterra ...". A sidebar titled "Otras personas también buscan" lists related queries: "colorear mapas online", "mapa para marcar lugares visitados app", "mapamundi para llenar online", "generador de mapas geográficos", "mapa lugares visitados google maps", and "mapchart".

Aquí la opción más similar:

Simply select in the table below the countries you have been to and copy the html code below.

visited 2 states (0.88%)				
<input type="checkbox"/> Africa		<input type="checkbox"/> Americas	<input type="checkbox"/> Asia	<input type="checkbox"/> Australia and Pacific
<input checked="" type="checkbox"/> Virgin Islands		<input type="checkbox"/> Mexico	<input type="checkbox"/> Costa Rica	
<input type="checkbox"/> Venezuela		<input type="checkbox"/> Martinique	<input type="checkbox"/> Colombia	
<input type="checkbox"/> Uruguay		<input type="checkbox"/> Jamaica	<input type="checkbox"/> Chile	
<input checked="" type="checkbox"/> United States		<input type="checkbox"/> Honduras	<input type="checkbox"/> Cayman Islands	
<input type="checkbox"/> Turks and Caicos Islands		<input type="checkbox"/> Haiti	<input type="checkbox"/> Canada	
<input type="checkbox"/> Trinidad and Tobago		<input type="checkbox"/> Guyana	<input type="checkbox"/> British Virgin	

Podemos concluir, pues, que el microentorno es bastante favorable en cuanto a la competencia actual y, además, es bastante positivo en una futura debido a que no parece que haya desarrollos similares; ni siquiera para dispositivos móviles:



Es vital para conocer la viabilidad de un proyecto conocer las Amenazas y Debilidades para evitarlas, mejorarlas o solucionarlas y las Fortalezas y Oportunidades para explotarlas al máximo.

Por este motivo conviene recalcar, aunque ya se ha mencionado, que *Mis Viajes* puede pecar de ser un sitio al que apenas se visite, sin generar tráfico ni muchos beneficios por publicidad, visitas, etc. Sin embargo, se debe tener en cuenta que, al no existir una competencia real y directa, existen muchas posibilidades de crecimiento y consolidación. Además, al adaptarse a los tiempos para ofrecer información sobre una situación de actualidad como es la de la actual pandemia, cuenta con una gran fortaleza sobre la que basar sus cimientos y comenzar el crecimiento.

La base del sitio es muy fuerte pero fácilmente ampliable, por lo que cuenta con grandes oportunidades. Existen muchas posibles opciones de ampliación que se verán en las conclusiones, como, por ejemplo, un enlace a APIs de grandes portales web de viajes como *Booking*, para que el usuario pueda recibir ofertas de los países a los que desea viajar.

Para su desarrollo es necesaria la fundación de una pequeña *startup* en un vivero de empresas en la localidad del creador (Zamora) con el nombre Calvo S.L. El nombre comercial será Mis Viajes y tendrá el siguiente logotipo:



Para el desarrollo del sitio se debería hacer una estimación del tiempo del desarrollo. La forma más sencilla suele ser por analogía (comparación con proyectos previos), sin embargo, al ser el primer desarrollo de esta *startup*, se debería realizar por descomposición en tareas. De este modo podríamos estimar:

- Organización del desarrollo y relaciones → 15 horas
- Creación del sitio básico → 30 horas
- Creación de los mapas interactivos → 40 horas
- Dar un estilo propio y distinguido al sitio → 10 horas

Es importante tratar de ajustar lo máximo a la realidad porque si estimamos de menos tendremos que realizar un trabajo “que no se va a pagar” puesto que estaba estipulado que iban a trabajarse menos horas (muchas consultoras pierden grandes cantidades de dinero por esta razón). Si, por el contrario, se estiman horas de más, el presupuesto será mucho más grande que otras empresas y no obtendríamos el desarrollo.

## Análisis de costes

Habiendo definido los objetivos del sitio y su viabilidad, con la estimación de las horas de desarrollo incluidas, el siguiente paso es el análisis de los costes. Tener un sitio web operativo tiene unos costes asociados tanto de desarrollo como de despliegue y mantenimiento.

El lugar para el desarrollo tiene un coste bajo, **230€/mes<sup>8</sup>** por el alquiler del local donde se va a desarrollar y mantener *online* la web.

El gasto más importante será la mano de obra del desarrollo que se estima de la siguiente manera (atendiendo a la viabilidad del proyecto).

	Horas	Precio/Hora	Total
Analista	15	40	600
Developer	70	25	1750
Diseñador gráfico	10	27	270
<b>TOTAL</b>			<b><u>2620 €</u></b>

El paso siguiente sería el hosting, que en este caso sería gratuito (salvo el mantenimiento del servidor) y el dominio para identificar nuestro sitio. Como los dominios más importantes ya se encuentran ocupados, la opción más viable sería recurrir al dominio **.info**; por un coste de **15 euros/año** con una oferta importante el primer año en el que casi sería gratis.

**misviajes.es ya está registrado , misviajes.com ya está registrado ,**

✓ misviajes.info      2,25 €/año por 1 año  
después 15 €/año      Seleccionar

Teniendo en cuenta que si un sitio desea estar bien posicionado en el mercado y ser bien visto por los usuarios (especialmente los más inexpertos), se debería intentar comprar los dominios <http://www.misviajes.com> y <http://www.misviajes.es>; teniendo a los precios medios<sup>9</sup>, este coste serían de 20€/año por el .com y 10€/año por el .es; es decir 35€/año.

También habría que contratar un certificado SSL por **32,26€/año** (SSL.com).

Como mantenimiento, establecer sistemas de copias de seguridad, seguridad en el sistema, etc., se pueden calcular unas 20 horas de manos de obra para un administrador de sistemas a un precio de 25 € la hora, es decir **500€**.

El presupuesto final sería: 2620 € de mano de obra de desarrollo, 500 € mantenimiento y seguridad, 15 € dominio y 32,26 de SSL más los 230 de alquiler, osea **3.397,26€** (el primer año).

Atendiendo a los ingresos medios con 3 banners<sup>10</sup>, necesitaríamos 2.3 millones de visitas para rentabilizar la web.:  $(2.3 \text{ millones de visitas} * 3 \text{ banners} * 0.5 \text{ CPM}) / 1000 = 3.500 \text{ €}$ . Lo cual sugiere un inicio complicado pero viable con el paso del tiempo, especialmente con unas [posibles ayudas del 80%](#).

<sup>8</sup> <https://www.camarazamora.com/emprendimiento/vivero-de-empresas>

<sup>9</sup> <https://www.ionos.es/digitalguide/dominios/consejos-sobre-dominios/cuanto-cuesta-un-dominio-web/>

<sup>10</sup> <https://www.marketingguerrilla.es/lo-que-una-web-con-100-000-paginas-vistas-ingresa-con-publicidad-online-al-mes/>

## Solución planteada

Para realizar todo lo mencionado hasta ahora es necesario contar con:

### Medios Materiales Utilizados

#### *Hardware*

Para el desarrollo del proyecto se ha utilizado un ASUS de la serie ROG modelo *GL552VW-DM141*. Cuenta con las siguientes características: procesador Quad-Core i7 6700-HQ a 3.5 Ghz., 16 GB DDR4 de RAM a 1600 Mhz. y 256 GB de memoria secundaria (SSD).

Por su parte, para el despliegue se ha utilizado un ordenador de sobremesa con un procesador dual core a 2.9 Ghz con 8 GB de RAM y un disco SSD de 256 GB.

#### *Software*

El Sistema Operativo utilizado como base ha sido **Windows 10 Pro N**.

En cuanto a los programas utilizados para el desarrollo, el principal ha sido **NetBeans**, porque, aunque no es mi IDE favorito, ha sido el que más hemos utilizado a lo largo del curso y porque la depuración de código PHP desde NetBeans me parece muy sencilla. La versión utilizada ha sido: *12.5 de septiembre de 2021* con el **xDebug** para depuración en su versión 3.0.

También ha sido utilizado el **Visual Studio Code**. Me gusta mucho el desarrollo web con él, pero no ha sido el principal debido a la depuración PHP. La versión de Visual Studio Code era la *1.63.0 user setup*.

El entorno en el que se va a desplegar la aplicación elegido ha sido **XAMPP**. La elección del mismo ha sido por la sencillez de uso y su simpleza. La versión utilizada ha sido la v3.3.0.

La BBDD ha sido desarrollada a mano a través de **PhpMyAdmin**, accediendo a la misma a través de un navegador web. La BBDD tiene un tipo de servidor **MariaDB** en su versión 10.4.21.

Para el procesamiento de textos, análisis, etc., se ha utilizado el programa **Notepad++**. Ha sido especialmente muy útil en el manejo de grandes cantidades de información, como el archivo con todos los polígonos, o en la modificación de varias líneas de manera simultánea y la comparación de archivos.

Un programa que no es estrictamente necesario pero ha sido utilizado para el procesamiento de imágenes ha sido el **Photoshop** en su versión 6.

Por último, y no por ello menos importante, unos programas necesarios para este desarrollo web han sido los **navegadores**. Tanto para comprobar la correcta visualización de la web como para su desarrollo se han utilizado los siguientes: Mozilla Firefox en su versión 94.0.1 (64-bit), Google Chrome en su versión 95.0.4638.69 de 64 bits, Internet Explorer en su versión 11 y Microsoft Edge en su versión 95.0.1020.44 de 64 bits.

### *Medios humanos*

La aplicación ha sido desarrollada íntegramente, como no puede ser de otra manera, por Alejandro Calvo Toribio, el alumno que presenta el proyecto. Sin embargo, hay una serie de personas que han tenido participación en la fase de pruebas y han aportado sus apreciaciones en lo que sería una fase beta, para el *testeo*, del sitio web.

## Planificación del proyecto

La planificación de un proyecto es harto complicada porque es muy difícil en informática ajustar los tiempos por fallos que se deben depurar, el uso de tecnologías o *frameworks* que no se dominan completamente, etc.

Hay que tener en cuenta que ha sido muy difícil ajustar la planificación (real, la ajustada al desarrollo de este proyecto) y seguir los pasos establecidos en el anteproyecto. Como trabajador a jornada completa en un sector que no es el de desarrollo es muy difícil ajustar el tiempo que se le dedica al proyecto. Además, aunque he estado trabajando de desarrollador web, mis conocimientos se encontraban un poco oxidados.

Pese a todo ello, se comenzó por la parte que se había establecido en el anteproyecto, la BBDD. El desarrollo de la misma ha llevado muchas horas debido a que, aunque es ficticia, se ha intentado que sea lo más aproximada a la realidad posible.

Posteriormente se estableció el servidor donde se alojaría el sitio, lo cual no llevo mucho tiempo al ser uno propio. Pese a que el desarrollo del sitio iba a ser en otro equipo, se dejó el entorno de despliegue montado en este momento.

El grueso del trabajo fue el desarrollo propiamente dicho y la adaptación de los objetos de tipo GeoJSON. No es sólo como utilizar la librería que permite su manejo (*Leaflet*) y el desarrollo de la funcionalidad propia (como dotar a los polígonos de colores atendiendo a la BBDD), sino que hay que desarrollar y dibujar los países del mundo que pueden ser manipulados. Es cierto que existen multitud de opciones de mapas realizados con GeoJSON, incluso de propios organismos oficiales como el [gobierno autonómico de Canarias](#), pero siempre hay que modificar muchos polígonos, eliminar otros que la ONU no estima como país, añadir ciertas islas administradas por un país, etc. Además, en este desarrollo se han añadido una serie de datos a los propios polígonos: ID, nombre, nombre formal, abreviatura, hemisferio, continente, subcontinente, población, emisiones y PIB; para que se pudieran formar diversos tipos de mapas atendiendo a la característica establecida (funcionalidad dinámica de las páginas de cultura).

Cuando se estaba finalizando el desarrollo de los polígonos se comenzó, a la par, el desarrollo del sitio propiamente dicho: estructura de carpetas, creación de las páginas, etc. Según se iban creando, iban surgiendo algunas pequeñas modificaciones fácilmente asumibles.

Cuando el sitio funcionaba perfectamente e, incluso, ya era accesible desde internet, se comenzó, por último, la etapa en la que se iba mejorando el proyecto con una mejor estética (se comprobaba en diversos dispositivos; *responsive*) y configuraciones o desarrollos nuevos atendiendo a las sugerencias de la gente que estuvo en la fase de pruebas.

Atendiendo a esta planificación que se siguió (otorgada en el anteproyecto), se explica la solución pormenorizada del sitio.

## Base de Datos

### Planificación

En un primer momento había que tener en cuenta que datos se querían almacenar y cuales no de cada país → Tabla **países**. Es la tabla principal de la BBDD, almacena los datos de forma estática y sólo pueden ser modificados por el administrador fuera del sitio web. Almacena las características de cada país, en principio estáticas:

- ID. Que identificará como Clave Primaria al mismo
- Nombre
- Capital
- Continente
- Hemisferio
- Idioma principal

Booleanos con características del país (se tiene o no).

- Playa
- Montaña
- Safari
- Cultural (si el país merece la pena ser visitado por su cultura. Por ejemplo, el Reino Unido es cultural por el *Big Ben, British Museum*, etc.).
- ONG (si el país puede considerarse para realizar un voluntariado).
- Exótico (desde un punto de vista sur-europeo).
- Turístico
- Peligroso (nivel de riesgo en el viaje).

Campo COVID. Este campo es importante para tener una APP actualizada a nuestros días. Con este campo el usuario tiene una idea, dentro de la complejidad y la gran variabilidad de estos datos, de si se puede viajar al país, si no tiene ninguna restricción, si necesita pasaporte Covid o si debe guardar cuarentena a la llegada.

- Covid.
  - 0 → No se tienen datos claros del país.
  - 1 → El país exige PCR negativa o pauta completa de vacunación.
  - 2 → El viajante debe guardar cuarentena al llegar.
  - 3 → Prohibición de viaje al país.
  - 4 → Sin restricciones.

La tabla países es la primera y principal (la que llevó el gran % de horas de desarrollo), pero no es la única, existen las tablas **usuarios** y **paisesVisitados**. Para que la web sea accesible por más de un usuario, se debe crear una tabla que almacene el nombre de usuario y su contraseña (se podían haber puesto una serie de campos opciones como email, teléfono, etc.; pero se intentó un desarrollo simple). El campo contraseña, como resulta evidente, está codificado en md5 y el campo “nombre” está establecido como único e irrepetible.

La tabla países y usuarios no están relacionadas (recordemos que países sólo es modificable por el administrador), pero ambas sí que estarán relacionadas con paisesVisitados. Esto es debido a que es una tabla que recibe el id del usuario y el id del país y almacena el estado “visitado” que introduce el propio usuario sobre un país. Es decir, por ejemplo, el usuario 1 (Alejandro) quiere almacenar el país 3 (Alemania) como visitado 2 (un país que ha visitado).

## Creación

Se comienza, como no puede ser de otra manera, con la creación de la propia base de datos → **paisesDB**, aprovechando para crear la primera tabla → **Países**.

```

1 CREATE DATABASE paisesDB;
2 USE paisesDB;
3 CREATE TABLE paises (
4     id INTEGER NOT NULL,
5     nombre TEXT NOT NULL,
6     continente TEXT NOT NULL,
7     hemisferio TEXT NOT NULL,
8     capital TEXT NOT NULL,
9     idioma TEXT NOT NULL,
10    playa INTEGER NOT NULL,
11    montana INTEGER NOT NULL,
12    safari INTEGER NOT NULL,
13    cultural INTEGER NOT NULL,
14    ong INTEGER NOT NULL,
15    exotico INTEGER NOT NULL,
16    turistico INTEGER NOT NULL,
17    peligroso INTEGER NOT NULL,
18    covid INTEGER NOT NULL,
19    PRIMARY KEY(id)
20 );

```

Tras la creación tenemos el mensaje de confirmación.

 MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0039 segundos.)

`CREATE DATABASE paisesDB;`

[ [Editar en línea](#) ] [ [Editar](#) ] [ [Crear código PHP](#) ]

El momento “crítico” del desarrollo de la BBDD no fue la planificación ni la creación de la estructura, sino el que viene a continuación, que es la recopilación de los datos de cada uno de los países, **hecho a mano** recopilando la información a través de internet. Sin duda fue el trabajo más tedioso y menos técnico. Habría que recordar, aunque ya se había mencionado, que cuesta hasta saber cuántos países hay en el mundo, puesto que no existe una idea única.

Con el nombre de los 195 países que se van a establecer, había que ir rellenando el resto de campos. En ocasiones es muy sencillo, todo el mundo conoce que el idioma oficial de los Estados Unidos de América es el inglés, que su capital es Washington D.C. o que se encuentra en América en el hemisferio Norte, pero, ¿Podríamos decir lo mismo de Suazilandia? Para llenar los campos se ha recurrido a diversas webs, siendo la más visitada la *Wikipedia*. Habría que hacer una mención especial al campo **COVID** puesto que la recopilación de datos ha sido especialmente difícil. Se han seguido [esta web](#), en sus versiones también para [África](#) y [Sudamérica](#); también se ha recurrido a un periódico regional español para conocer la situación de [Europa](#). Existen multitud de contradicciones, incluso en la propia web, donde en la representación en el mapa aparece de una forma y “escrito” de otra. Si la aplicación fuera real y estuviera lanzada en producción, tendría que haber todo un equipo que fuera visitando 1 por 1 todas las webs oficiales de los 195 países reconocidos y actualizando los datos prácticamente a diario. Además, no se establecerían simplemente 4 niveles en un campo de una BBDD, sino que establecerían claramente las restricciones (o no) que pudiera tener dicho país.

La parte más sencilla fue, una vez recopilada la información, realizar los 195 INSERTS que se añadirán a la tabla (1 por país). Se ponen los 5 primeros países como ejemplos (el resto, como todo el código, se anexará al final del proyecto).

```

1 INSERT INTO paises VALUES (1,'Afganistán','Asia','Norte','Kabul','Dari',0,0,0,0,0,0,0,1,0,0);
2 INSERT INTO paises VALUES (2,'Albania','Europa','Norte','Tirana','Albanés',1,0,0,0,0,0,0,0,0,4);
3 INSERT INTO paises VALUES (3,'Alemania','Europa','Norte','Berlín','Alemán',1,0,0,1,0,0,1,0,0,0);
4 INSERT INTO paises VALUES (4,'Andorra','Europa','Norte','Andorra la Vieja','Francés',0,1,0,0,0,0,1,0,0,1);
5 INSERT INTO paises VALUES (5,'Angola','Africa','Sur','Luanda','Portugués',0,0,1,0,1,0,0,0,0,0);

```

### Comprobación.

	→ T →	▼ id	nombre	continente	hemisferio	capital	idioma	playa	montaña	safari	cultural	ong	exotico	turistico	peligroso	visitado			
<input type="checkbox"/>		Editar		Copiar		Borrar	1	Afganistán	Asia	Norte	Kabul	Dari	0	0	0	0	0	1	0
<input type="checkbox"/>		Editar		Copiar		Borrar	2	Albania	Europa	Norte	Tirana	Albanés	1	0	0	0	0	0	0
<input type="checkbox"/>		Editar		Copiar		Borrar	3	Alemania	Europa	Norte	Berlín	Alemán	1	0	0	1	0	1	0
<input type="checkbox"/>		Editar		Copiar		Borrar	4	Andorra	Europa	Norte	Andorra la Vieja	Francés	0	1	0	0	0	1	0
<input type="checkbox"/>		Editar		Copiar		Borrar	5	Angola	Africa	Sur	Luanda	Portugués	0	0	1	0	1	0	0
<input type="checkbox"/>		Editar		Copiar		Borrar	6	Antigua y Barbuda	America	Norte	Saint John	Inglés	1	0	0	0	0	0	0

Realizados los 195 INSERT tenemos la BBDD creada para poder trabajar con ella.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> paises	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	195	InnoDB	utf8mb4_general_ci	16.0 KB	-
1 tabla	Número de filas	195	InnoDB	utf8mb4_general_ci	16.0 KB	0 B

Podemos hacer alguna prueba para comprobar en un principio su integridad. Por ejemplo, vamos a comprobar si la BBDD actúa correctamente ante una posible consulta de información del usuario: país de habla inglesa y, posteriormente, precisando la búsqueda a un país de habla inglesa europeo y que tenga playa.

nombre
Antigua y Barbuda
Australia
Bahamas
Barbados

Precisando la búsqueda:

```
SELECT nombre FROM paises WHERE idioma="ingles";
```

nombre
Irlanda
Malta
Reino Unido

La siguiente tabla a crear es **usuarios**. Una tabla muy sencilla con el campo del nombre del usuario y el campo en el que se almacenará la contraseña codificada (que debe ser VARCHAR (32)). Para su manejo sencillo se crea un campo *id\_usuario* que será AUTO\_INCREMENT y clave primaria de la tabla. Al no ser el nombre la PK, se establece como único, para que no existan duplicidades en el nombre de usuario. Ningún campo puede ser nulo.

```

1 -- Seleccionamos la BBDD
2 use paisesdb;
3
4 -- Creamos la tabla
5 CREATE TABLE usuarios(
6     id_usuario INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
7     nombre VARCHAR(16) NOT NULL,
8     pass VARCHAR(32) NOT NULL,
9     UNIQUE(nombre)
10 )ENGINE=INNODB;
```

Se crean dos usuarios de prueba.

			id_usuario	nombre	pass	
<input type="checkbox"/>				1	root	63a9f0ea7bb98050796b649e85481845
<input type="checkbox"/>				2	alex	534b44a19bf18d20b71ecc4eb77c572f

Se comprueba el funcionamiento de UNIQUE (no repetir nombre de usuario):

```

INSERT INTO usuarios(nombre,pass) VALUES('alex',md5('voyAFallar'));
MySQL ha dicho: ⚒

#1062 - Entrada duplicada 'alex' para la clave 'nombre'
```

La última tabla necesaria es la que relaciona a los usuarios con los países, almacenando la elección del usuario sobre el país → **paisesVisitados**. De este modo los campos principales son *id\_usuario* e *id\_pais* que son FK de las PK de las tablas previas. El resto de campos son un id que será AUTO\_INCREMENT y PK y el campo visitado que es el que almacenará la “estancia” del usuario en el país. Ningún campo podrá ser nulo.

```

1 -- Seleccionamos la BBDD
2 use paisesdb;
3
4 -- Creamos la tabla
5 CREATE TABLE paisesVisitados(
6     id INTEGER AUTO_INCREMENT PRIMARY KEY,
7     id_usuario INTEGER NOT NULL,
8     id_pais INTEGER NOT NULL,
9     visitado INTEGER NOT NULL,
10    FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario),
11    FOREIGN KEY (id_pais) REFERENCES paises(id)
12 ) ENGINE = INNODB;
```

Como siempre, se probaba la funcionalidad con unos registros de prueba.

```

1 -- Campo de prueba visita a Alemania, vivir en España y desear ir a Francia
2 INSERT INTO paisesvisitados(id_usuario, id_pais, visitado) VALUES (1,3,1);
3 INSERT INTO paisesvisitados(id_usuario, id_pais, visitado) VALUES (1,59,2);
4 INSERT INTO paisesvisitados(id_usuario, id_pais, visitado) VALUES (1,66,3);

```

Por último, se hace una prueba de integración, relacionando las tablas.

The screenshot shows a phpMyAdmin interface with a query results page. The query displayed is:

```
SELECT nombre FROM paises WHERE id = (SELECT id_pais FROM paisesvisitados WHERE id_usuario = 1 AND visitado = 1);
```

Below the query, there are several buttons: Perfilando [ Editar en línea ], [ Editar ], [ Explicar SQL ], [ Crear código PHP ], and [ Actualizar ]. There are also filters for 'Mostrar todo' (Show all), 'Número de filas' (Number of rows) set to 25, and a search bar 'Filtrar filas' (Filter rows) with 'Buscar en esta tabla' (Search this table). At the bottom, there are buttons for '+ Opciones' (More options), 'nombre' (name), 'Edit' (Edit), 'Copiar' (Copy), 'Borrar' (Delete), and 'Alemania' (Germany).

### Seguridad

Antes de comenzar con el enlace de la BBDD con la aplicación web hay que darle seguridad. Por defecto la BBDD se conecta al usuario *root* sin contraseña lo cual es una brecha de seguridad enorme.

### Vista global de las cuentas de usuario

Nombre de usuario	Nombre del servidor	Contraseña	Privilegios globales	Grupo de usuario	Conceder	Acción
<input type="checkbox"/> cualquiera	%	No	USAGE		No	Editar privilegios  Exportar
<input type="checkbox"/> pma	localhost	No	USAGE		No	Editar privilegios  Exportar
<input type="checkbox"/> root	127.0.0.1	No	ALL PRIVILEGES		Sí	Editar privilegios  Exportar
<input checked="" type="checkbox"/> root	127.0.0.1	No	ALL PRIVILEGES		Sí	Editar privilegios  Exportar
<input type="checkbox"/> root	localhost	No	ALL PRIVILEGES		Sí	Editar privilegios  Exportar

Para solucionar este problema le otorgamos una contraseña a todas las entradas de *root*.

### Editar los privilegios: Cuenta de usuario 'root'@'127.0.0.1'

The screenshot shows a 'Cambio de contraseña' (Change password) dialog. It has two radio button options: 'Sin contraseña' (No password) and 'Contraseña:' (Password). The 'Contraseña:' option is selected, and there are fields for 'Ingresar:' (Enter) containing a masked password and 'Debe volver a escribir:' (Must be re-entered) also containing a masked password. Below these fields is a 'Strength:' (Strength) bar which is green and labeled 'Fuerte' (Strong). A dropdown menu 'Hashing de la contraseña:' (Password hashing) is set to 'Autenticación de MySQL nativo' (Native MySQL authentication). At the bottom, there are 'Generar contraseña' (Generate password) and 'Generar' (Generate) buttons.

Nos expulsará al no tener privilegios para acceder a la BBDD y debemos incluir la contraseña en el archivo *config.inc.php*.

! phpMyAdmin intentó conectarse con el servidor MySQL, y el servidor rechazó esta conexión. Deberá revisar el host, nombre de usuario y contraseña.

Le ponemos las nuevas credenciales a dicho archivo para solventarlo.

```
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = 'Viajes.2021';
```

Y podremos tener acceso a la BBDD de nuevo.

<input type="checkbox"/> root	127.0.0.1	Sí
<input type="checkbox"/> root	::1	Sí
<input type="checkbox"/> root	localhost	Sí

Aun así, vamos a crear un usuario que no sea *root* que sea el que se va a conectar a la BBDD desde la aplicación para darle una seguridad extra.

## Agregar cuenta de usuario

Información de la cuenta

Nombre de usuario:	<input type="text"/> viajesDB
Nombre de Host:	<input type="text"/> Cualquier servidor %
Contraseña:	<input type="password"/> Strength: <span style="background-color: green;">Fuerte</span>
Debe volver a escribir:	<input type="password"/>
Authentication plugin	<input type="text"/> Autenticación de MySQL nativo

Con todos los permisos.

<input checked="" type="checkbox"/> Datos	<input checked="" type="checkbox"/> Estructura	<input checked="" type="checkbox"/> Administración
<input checked="" type="checkbox"/> SELECT <input checked="" type="checkbox"/> INSERT <input checked="" type="checkbox"/> UPDATE <input checked="" type="checkbox"/> DELETE <input checked="" type="checkbox"/> FILE	<input checked="" type="checkbox"/> CREATE <input checked="" type="checkbox"/> ALTER <input checked="" type="checkbox"/> INDEX <input checked="" type="checkbox"/> DROP <input checked="" type="checkbox"/> CREATE TEMPORARY TABLES <input checked="" type="checkbox"/> SHOW VIEW <input checked="" type="checkbox"/> CREATE ROUTINE <input checked="" type="checkbox"/> ALTER ROUTINE <input checked="" type="checkbox"/> EXECUTE <input checked="" type="checkbox"/> CREATE VIEW <input checked="" type="checkbox"/> EVENT <input checked="" type="checkbox"/> TRIGGER	<input checked="" type="checkbox"/> GRANT <input checked="" type="checkbox"/> SUPER <input checked="" type="checkbox"/> PROCESS <input checked="" type="checkbox"/> RELOAD <input checked="" type="checkbox"/> SHUTDOWN <input checked="" type="checkbox"/> SHOW DATABASES <input checked="" type="checkbox"/> LOCK TABLES <input checked="" type="checkbox"/> REFERENCES <input checked="" type="checkbox"/> REPLICATION CLIENT <input checked="" type="checkbox"/> REPLICATION SLAVE <input checked="" type="checkbox"/> CREATE USER

Aunque no se ha comenzado el desarrollo del sitio ya se puede comprobar la funcionalidad de la BBDD desde una página de prueba para ver si se obtiene el resultado esperado.

```
try {
    $opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
    $bbdd = new PDO("mysql:host=localhost;dbname=paisesDB", "viajesDB", "Viajes.2021", $opciones);
    $bbdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    $error = $e->getCode();
    $mensaje = $e->getMessage();
}

$sql = "SELECT * FROM paises WHERE nombre = 'España'";
$resultado = $bbdd->query($sql);

while($row=$resultado->fetch()) {
    echo "Soy el país número $row[id] con nombre $row[nombre]";
}

unset($resultado);
unset($bbdd);
```

Resultado:

Soy el país número 59 con nombre España

## Despliegue

El despliegue de todo el sitio web no se produce hasta prácticamente días previos a la entrega, pero desde esta segunda fase del desarrollo se prepara todo para que el sitio esté accesible ya que va a ser en un entorno “privado” y no es un sitio web real. De este modo, desde muy temprano se preparan 2 servicios en el equipo “servidor”. El primero y más evidente es la instalación de XAMPP con Apache y MySQL para que la web sea correctamente servida; el segundo es un servidor FTP para ir subiendo los cambios de manera paulatina.

Con **XAMPP** se instala tanto **Apache** como servidor web como **MySQL** como sistema gestor de BBDD, son los mismos, con la misma versión, que los que existen en el equipo que se encuentra desarrollando el sitio. Por su parte se ha montado un servidor FTP a través del IIS propio de Windows para replicar los cambios a distancia. El servidor FTP está deshabilitado desde la “*puesta en producción*” al 100%, por seguridad.

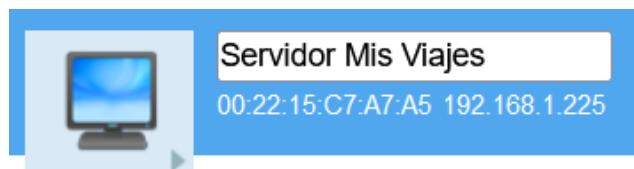
Para conseguir todo ello, lo primero que se debe hacer (si eres de una compañía *LowCost*), es pedir que te establezcan una IP Pública, debido a que muchas te otorgan una CG-NAT, por lo que en mi caso fue lo primero que hice, obteniendo la dirección (en ese momento, puede haberse alterado): **170.253.5.234**, como se puede ver en esta captura del router.

Access type:	Ethernet
IPv4 status:	Connected
IPv4 address:	170.253.5.234

A partir de ese momento ya tenemos una ubicación “real” en internet, por lo que podríamos acceder a la red desde fuera. Sin embargo, el interés principal es el acceso a un equipo específico, que será el que esté sirviendo tanto el servicio web como MySQL, por lo que el siguiente paso es crear una zona desmilitarizada (DMZ). Para comenzar a realizar esta configuración, lo primero que se debe realizar es otorgar una IP estática (y alta, para evitar posibles conflictos) al equipo servidor; se eligió la 225 → **192.168.1.225**. No se eligió la más alta posible del rango (254) debido a que se suele utilizar para servidores y se le quiere poner una ligera traba a un posible *hacker* en la fase de descubrimiento de activos.

Comprobación en el equipo y en el router.

```
Dirección IPv4. . . . . : 192.168.1.225
Máscara de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . : 192.168.1.1
```



Establecida la IP estática, se debe establecer una configuración en el router que permita redirigir las peticiones exteriores a la IP establecida a través del mencionado DMZ.

Host address: AsusROG

Enable DMZ:

[Save](#)

A partir de este momento el equipo ya es accesible desde el exterior de la red, aunque **no desde la red interna**. Para realizar la prueba se realiza con los datos del teléfono. Pese a que se debería poder entrar, no está de más abrir el puerto 80 de conexión HTTP. Por lo que procedo a ello.

Nombre de host	Last Update	IP / Objetivo	Type
misviajes.ddns.net <small>Active</small>	Dec 31, 2021 00:37 CET	170.253.5.234	A

El sitio es accesible, pero a través de una interfaz muy poco amigable (escribir la IP directamente). En este momento es donde entran en juego *dyndns* y *no-ip*. Ambos me van a dar un “nombre de dominio” gratuito para subir la web (si fuera una página profesional no sería correcto) y, además, van a estar a la escucha de mi IP Pública por si cambiara actualizarla y que siempre puedan entrar los usuarios.

Tras hacer las configuraciones pertinentes para que funcione 30 días, este es el resumen de la configuración:

Host name	Domain name	Provider	Status
misviajes	misviajes.ddns.net	No-IP.com	Failed to synchronize

Ahora está la opción de configurar el router o el equipo para que actualice la IP ya que es dinámica. Para ello en “servicios de internet” activamos el *DDNS (dynamic DNS)* con la configuración establecida en *no-ip*.

Enable DDNS:

Status: Failed to synchronize

Provider: No-IP.com

Host name: misviajes

Domain name: misviajes.ddns.net

Username: [REDACTED]

Password: [REDACTED]

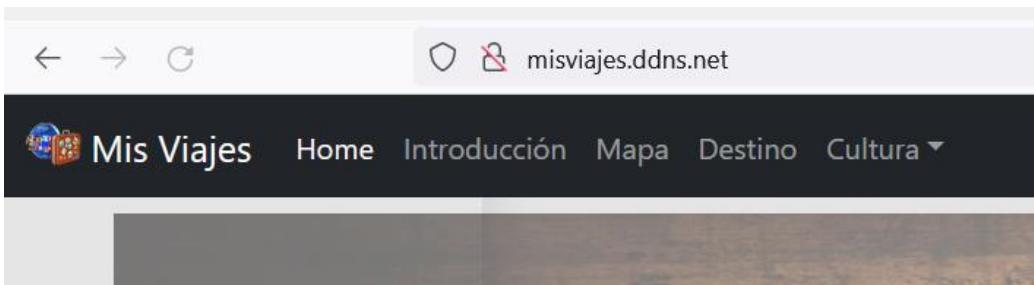
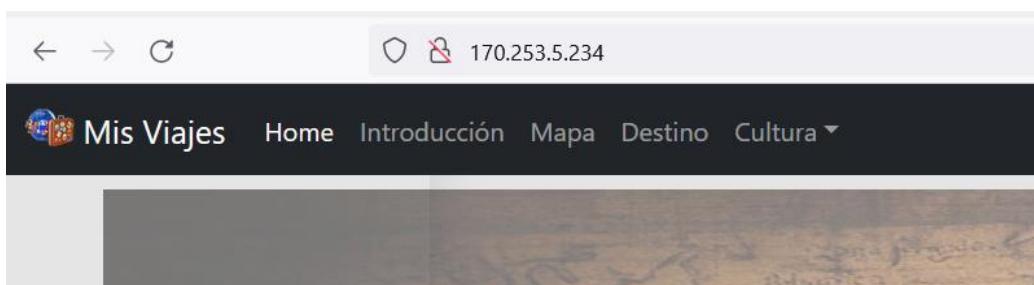
[Save](#)

Activar Windc

A partir de este momento está todo listo para que siempre esté funcionando (salvo por las condiciones de gratuidad del proveedor de servicios). Pero si entramos vemos que nos redirige al *dashboard*. Para que esto no sea así tenemos que indicarle a Apache cuál queremos que sea la ruta que ofrezca y lo haremos mediante la directiva *DocumentRoot*.

```
#  
# DocumentRoot: The directory out of which you will serve your  
# documents. By default, all requests are taken from this directory, but  
# symbolic links and aliases may be used to point to other locations.  
#  
DocumentRoot "C:/xampp/htdocs/MisViajes"  
<Directory "C:/xampp/htdocs/MisViajes">
```

Después de reiniciar Apache ya está todo listo para su funcionamiento, se muestran unas fotos posteriores (no se había iniciado el desarrollo).



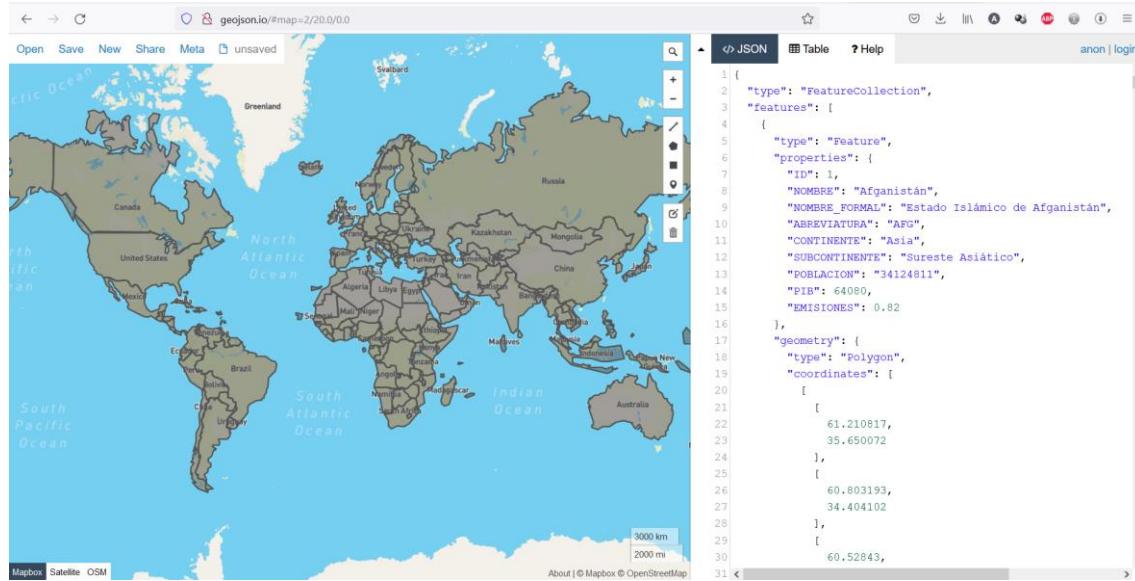
## GeoJSON

El desarrollo de los objetos de tipo GeoJSON es otro de los momentos tediosos del desarrollo puesto que hay que “dibujar” los 195 polígonos. Aunque se cogieron varias bases de internet, ninguna se adecuaba a los intereses del proyecto por lo que hubo que modificarlo continuamente hasta que se creó un archivo *js* que almacenara únicamente una variable con todo los polígonos y sus características (países) debido al gran tamaño del mismo.

Se muestra un el primer país, el resto seguirá la misma estructura.

Se observa cómo se crea la variable que será tratada por JS posteriormente con el valor de los países (*featurecollection*) que almacena cada una de las *features* (países). Todas tienen las mismas propiedades: id, nombre, nombre formal, abreviatura, continente, subcontinente, población, PIB y emisiones (se establecen las que se estimen necesarias) y, posteriormente, la representación geométrica en el mapa; es decir, un polígono con coordenadas.

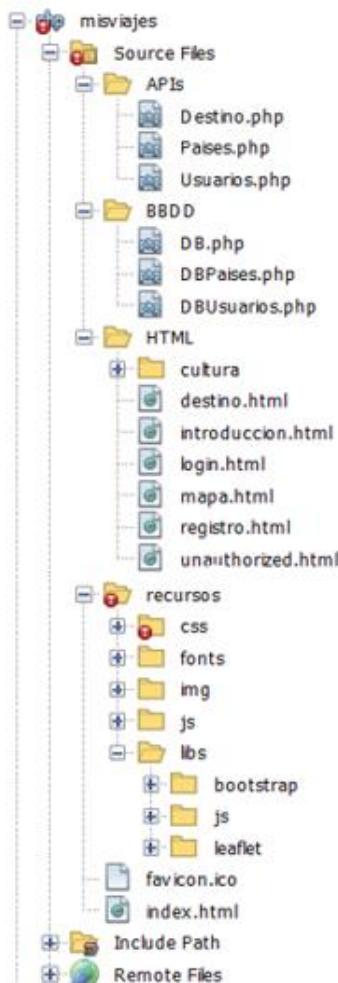
Aquí se puede observar el mapa en una web GeoJSON:



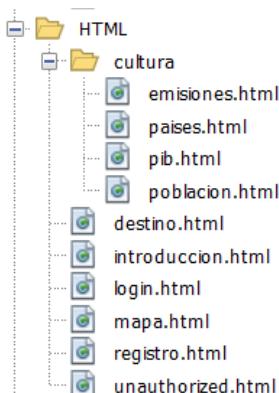
## Desarrollo

### Estructura

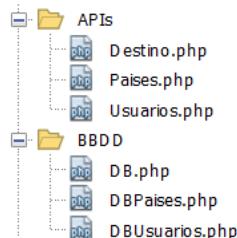
Para desarrollar un proyecto, lo primero que hay que tener en cuenta es una buena estructura planificada. Es cierto que luego pueden surgir modificaciones, pero hay que tener una idea clara de base. La estructura final de *Mis Viajes* es la siguiente:



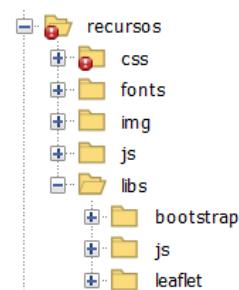
En el esquema se puede observar el *index.html* con su *favicon.ico* y la estructura de carpetas en la que se organizan el resto de archivos y recursos. Los más importantes serían APIs, BBDD y HTML. En esta última carpeta se encuentran todas las páginas accesibles por el usuario que no son el *index.html*; es decir: introducción, mapa, destino, *unauthorized*, *login* y *registro* y, en cultura, países, PIB, población y emisiones (todas HTML).



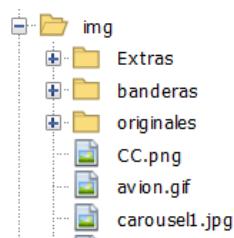
En APIs se encuentran los archivos PHP que son llamados mediante AJAX por cualquiera de estas páginas. Son: *Destino.php*, *Paises.php* y *Usuarios.php*. Cada una de estas APIs, lo único que hace (ya se explicará en detalle) es invocar a los archivos PHP que sí que tendrán contacto con la BBDD, los guardados en esta carpeta, que son: *DB.php*, *DBPaises.php* y *DBUsuarios.php*.



Por último, queda la carpeta de recursos, dónde se almacenan todos los archivos de estilo CSS (css), los scripts en JS (js), las imágenes (img), fuentes (fonts) y las librerías de los frameworks utilizados: *Leaflet*, *Bootstrap* y *Jquery*.



Dentro de la propia carpeta de imágenes se establecen otras subcarpetas para la organización puesto que se almacenarán las banderas de los 195 países junto con una gran cantidad de imágenes utilizadas y sus originales.



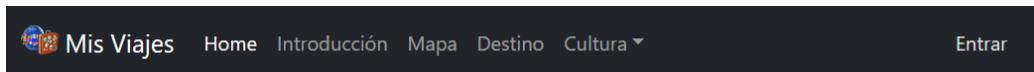
A continuación, se van a exponer las diferentes páginas el sitio. Primero se explicará la funcionalidad de cada una de ellas, luego la implementación dinámica de las que lo tengan y como se ha hecho y, por último, el estilo de las mismas. Pero antes de ello cabría mencionar la relación de los archivos. **Todos los archivos js tienen referencia al nombre del HTML** que lo enlaza; de este modo *mapa.js* y *funcionesmapa.js* son enlazados por *mapa.html* (al igual que *CONST.js*, que contiene la referencia a los mapas). Por su parte *funcionesCultura.js* es invocado por todas las páginas de cultura y *funcionesDestino.js* por *destino.html*. Por último *loginLogout.js* es referenciada por todas, puesto que es la que gestiona el comportamiento “dinámico” del *nav-bar*.

**El estilo sigue la misma regla;** de este modo *modal.css* es utilizado por todas las páginas que tienen un modal (index, introducción, mapa y destino). El resto: *mapa.css*, *destino.css*, *cultura.css* y *forbidden.css* son los utilizados por sus correspondientes HTML. El único diferente es *estilo.css* utilizado por el index e introducción. Además, *animate.css* también es utilizado por index (animación del contenido integrado en el *carousel*).

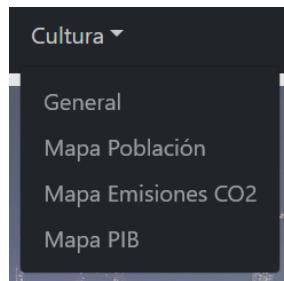
### index.html

Irónicamente no fue la primera página desarrollada ya que primero había que comprobar que *mapa.html* funcionara correctamente puesto que iba a ser la funcionalidad principal. La página es muy sencilla, aunque puede no parecerlo debido a que cuenta con mucho código, ya que se han utilizado unos ejemplos de *Bootstrap* que permiten un desarrollo rápido y estéticamente agradable. De este modo se creó la página, sirviendo muchas de las cosas aquí desarrolladas para el resto.

Lo primero que habría que explicar de esta página (y que es común para todas) es el **navbar**. La barra de navegación que se encuentra siempre en la parte superior sin ocultar (*fixed-top*) con los enlaces a todas las páginas webs accesibles por el usuario.



Como se puede ver es un estilo sencillo de blanco sobre negro para facilitar la lectura y en ella se puede observar el logo del sitio. Como ya se ha dicho es común a todas las páginas y siempre será así. Existe un *hover* que crea un color intermedio entre el blanco oscurecido de los no seleccionados y el blanco claro de la página en la que te encuentras. La sección entrar se encuentra a la derecha para diferenciarla claramente. Por su parte, la sección de cultura es un desplegable con todas las opciones existentes relacionadas.



No se comenta el código puesto que es muy sencillo. Simplemente se comenta la opción de entrar. Parece sencilla puesto que simplemente crea un enlace a la página de *login*; sin embargo, si el usuario ya se encuentra con la sesión iniciada se le muestra un mensaje personalizado y se convierte en un desplegable (como cultura) en el que el usuario puede cerrar la sesión o ir a la página de “Área personal” (que es la misma que “*login*”, pero modificada dinámicamente).



Cabría reseñar que en dispositivos con pantalla más pequeña se comprime todo el menú en un botón.



La siguiente parte del *index.html* es el *carousel* de imágenes que ha sido modificado a las necesidades de *Mis Viajes*, pero es bastante sencillo implementarlo desde *Bootstrap*. Es el eje central de la página.



En la imagen se pueden identificar las partes. Tenemos un fondo en el que se ve la imagen (1), también se aprecia la posibilidad de navegar entre las imágenes (2 – no activo y 2b – activo) hacia la imagen previa o la siguiente, en la parte inferior encontramos un recuadro (3) con fondo para facilitar la lectura que cuenta, en todas las imágenes, con un título, un mensaje y un enlace (varios en la última imagen de “cultura”). Por último, (4) existen unos botones que permiten la navegación directa entre las distintas imágenes.

Todos los elementos del recuadro (3) cuentan con una animación de entrada a la web.

El siguiente elemento del *index* son 3 zonas que engloban características comunes: una foto descriptiva, un pequeño mensaje de información y el botón que enlaza a una página importante.

## Funciones principales de Mis Viajes Web



**Mapa**

Identifica en un mapa los países en los que has estado o quieras estar

[Mapa »](#)



**Destino**

¿Eres indeciso? Mis Viajes te ayuda a elegir tu próxima aventura

[Destino »](#)



**Cultura**

¿Quieres conocer más del mundo que te rodea? Entra en esta sección

[Cultura »](#)

Al igual que con el *carousel*, es muy sencillo implementarlo con *Bootstrap*. No cuentan con la atención principal de la página (que es el *carousel*), pero tienen una posición principal puesto que enlazan a las páginas más importantes.

La última parte del *main* son dos *featurettes* creadas con *Bootstrap* que dan información sobre el sitio.

Elige la opción *Mapa* para navegar por el mundo que has conocido o deseas conocer La aplicación que cambiará tu ocio

Diviértete dibujando tus viajes. Disfruta con tus amigos compartiendo tus hazañas. ¡Que se mueran de envidia!



¿Has viajado tanto que no sabes cuál puede ser tu próximo destino? Deja que Mis Viajes lo descubra por ti

Selecciona tus preferencias como el idioma, continente, tipo de ocio, etc. Nuestra dilatada experiencia en este mundo te indicará las mejores opciones para ti

Por último, se encuentra el *footer* que simplemente tiene información del creador, un enlace a la parte superior de la página y los términos y la privacidad de la web.

© 2021 Alejandro Calvo, Inc. · [Privacy](#) · [Terms](#)

[Back to top](#)

Tanto los términos como la privacidad son dos modales que aparecen cuando se pulsa el enlace.

[X](#)

## Privacidad

Datos recogidos

Le informamos que no llevamos a cabo ningún tipo de recogida de datos a excepción de los que usted introduce para registrarse en la página web. De este modo sólo tenemos tu **nombre** de usuario (puede no ser real) y tu **contraseña**. No recopilamos información sobre tu ubicación o IP, preferencias, uso, etc.

Tratamiento de los datos

Pese a ser mínimo, tienes derecho a la revocación de los mismos, por ello puedes dar de baja a tu usuario desde login

Motivos

No eres un negocio, por lo que no queremos comerciar con tu información. Podríamos recopilarlos para conocer como usas la aplicación; pero preferimos preguntarte. Para ello te proponemos que **contactes** con el administrador del sitio

Es importante destacar que son *responsive* puesto que se adaptan al dispositivo que los invoca y que tendrán el mismo estilo que el resto de modales del sitio. El fondo es un degradado de blanco a gris y cuentan con un botón de cerrar que es una X blanca sobre un círculo rojo que tiene un *hover* en el que la X se vuelve negra.

## introduccion.html

Ha sido una de las últimas páginas a desarrollar debido a su sencillez. La funcionalidad de la página es para que el usuario conozca como se utiliza y sepa todas las funcionalidades que tiene el sitio.

La parte superior cuenta con el mismo **nav-bar** que *index* (y que el resto del sitio), simplemente tiene activo su enlace y no otro, como es evidente.



La parte superior es una pequeña introducción en la que se habla de la web de forma muy genérica, con enlaces a todas las páginas del sitio.

### Get Started

Aprende a utilizar el sitio web Mis Viajes. Este sitio web te va a permitir, siempre que estés **registrado**, identificar aquellos países en los que has residido o a los que has viajado, *¡¡incluso a los que deseas hacerlo!!* El proceso es muy sencillo, simplemente debes entrar en la página **mapa**, allí vas a ver el mundo y podrás navegar en él, identificando los países que quieras que se tengan en cuenta. En un mapa no se aprecia todo un potencial que puede tener un país, por eso existe la página **destino**, que te ayudará a elegir tu próxima aventura. Como todo buen viajero seguro que te mueres por conocer curiosidades de otros lugares del mundo, no te preocupes en la sección de **cultura** las conocerás, ya sea de forma **general**, o mediante los mapas interactivos de **población**, **emisiones** o **PIB**. Si eres de los viajeros que planifican bien su viaje, te recomendamos que sigas leyendo para entender el funcionamiento de la aplicación; si, por el contrario, eres un aventurero, **¿Por qué no descubrirlo por tí mismo?**

La parte central es idéntica a las *featurettes* de *Bootstrap* utilizadas en el *index*. Existen 4, cuentan con un título, un texto que explica cómo funciona cada una de las páginas principales (mapa, destino y cultura) y el *login* junto con el área personal. Para ayudar a la comprensión se acompañan de un pequeño gif en el que se muestran los pasos principales y le da dinamismo a la página de una forma sencilla.

Elige la opción *Mapa* para navegar por el mundo que has conocido o deseas conocer

Siempre que hayas **iniciado sesión**, se te abrirá el mundo de una forma interactiva. Podrás navegar por él, como ya estarás acostumbrado en otras aplicaciones. Nosotros te invitamos a que, cuando identifiques un país en el que has estado o deseas estar, hagas click en él porque a continuación te preguntaremos si has residido allí, si has estado de vacaciones o si, por el contrario, quieras que ese país sea un destino futuro. No te preocupes hombre, todos somos humanos y cometemos errores si has seleccionado mal el país siempre podrás volver a hacer click en él para borrar tu elección. Sencillo, ¿No? Además como sabemos que te encanta recordar los sitios a los que has ido el *mapa* te los mostrará cambiando el color de los mismos. Seguro que estás deseando comenzar para ver la cara que se le queda a tus amigos cuando vean tu mapa casi lleno, ¿No es así?

Hay opciones secundarias que deberías conocer. ¿Deseas conocer el nombre de un país? *Sitúa el ratón encima de él* y te aparecerá en la parte superior derecha ¿Deseas centrar la vista en un país? *Haz click con el botón secundario del ratón* ¿Deseas cambiar el estilo del mapa o incluso desactivarlo? *Sitúa el ratón encima del ícono situado en la esquina superior derecha y ¡¡Selecciona el que más te guste!!*

¿Has estado o deseas estar en  
Francia?

Visitado Vivido Deseado

La parte inferior es el mismo **footer** que tiene el *index*, por lo que también cuenta con dos modales ocultos que se muestran si se hace *click* en los enlaces del **footer** (en esta ocasión se muestra el otro modal, términos).



## mapa.html

Es una de las dos páginas que necesitan una sesión iniciada para poder tener acceso debido a que necesitan información del usuario para mostrársela. En el caso de que no exista dicha sesión, se le redirigirá a una página que simplemente muestra que no tiene acceso y le dice que inicie sesión, que se registre si no lo ha hecho o que vaya al *index* de la página.



En el caso de que sí exista una sesión iniciada, se muestra la funcionalidad principal de la página: el mapa dinámico para marcar los países. Recordemos que cuenta con un **nav-bar** como el resto de páginas (salvo **unauthorized.html**, la que se muestra en la imagen previa).



En la imagen se pueden apreciar las partes principales. En principio el eje principal es el mapa de fondo (se ha elegido el *layer* satélite por defecto para que destaque) sobre la que enseguida distinguimos las fronteras remarcadas con unas líneas no continuas que son los polígonos GeoJSON. Es decir, tenemos la superposición de la parte dinámica sobre la estática.

También se observa la posibilidad de hacer *zoom in* y *zoom out* en la parte superior izquierda con el + y el -. Justo debajo tenemos una leyenda estática que simplemente informa al usuario de que representa cada uno de los colores y, justo debajo, una escala para que vea a qué tamaño está representado el mapa a ese nivel de zoom.

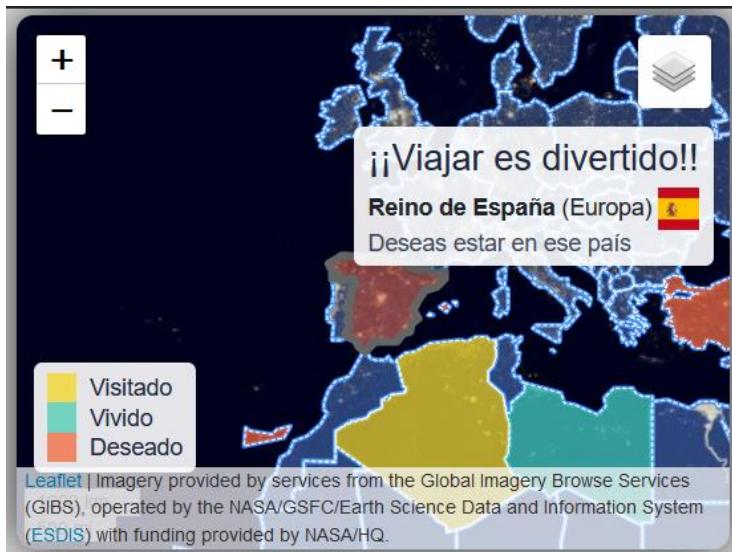
Por otra parte, a la derecha, se pueden observar 3 funcionalidades. La primera sería la elección de capa, si se hace *hover* por la imagen se abre un desplegable con todas las opciones.



Si se pulsa en alguna, se establece esa capa de fondo (*layer*), por ejemplo, podemos ver la opción histórico y noche.



Justo debajo se encuentra la leyenda dinámica. Es muy interesante porque reacciona al movimiento del ratón del usuario, si el usuario se sitúa sobre un país la leyenda se actualiza y en lugar de poner “Sítuate encima de un país” a las características del país en el que te encuentras situado.



En esta imagen se puede apreciar cuando el usuario se encuentra con el ratón situado en España (cambia el estilo del país) y la leyenda se actualiza mostrando información de él como el nombre formal y el continente junto con la bandera del país y, justo debajo, información de si se ha visitado, vivido o se ha marcado como futuro deseo.

Por último, en la parte inferior, se encuentra información referente a la capa que se encuentra cargada en ese momento debido a que es de licencia CC-by.

Además de las partes que se ven a simple vista hay una serie de acciones que se pueden realizar en esta página. La más sencilla es hacer *click* con el segundo botón del ratón, haciendo un zoom animado el mapa hacia el país en el que se ha hecho.



También se puede hacer *click* normal en cada uno de los países pudiendo pasar 2 situaciones: 1) ya se ha estado en ese país o se ha marcada como deseo (como lo está España en la imagen previa) o 2) no ha sido marcado (como lo está Portugal).

Si pulsamos en un país que no ha sido marcado (opción 2) se muestra un modal que ha permanecido oculto para que elijamos, si queremos, como marcar el país.



Por su parte, si hacemos *click* en un país que ya ha sido marcado, podremos eliminar esa estancia.



Sea cual fuere el elemento pulsado, el sitio almacena un registro en la BBDD para que cuando el usuario vuelva a entrar los siga teniendo o lo borra si así lo ha elegido él.

## destino.html

Es la otra página que necesita tener sesión iniciada, por lo que tiene la misma condición que la previa (redirige a *unauthorized.html* sin sesión de usuario activa). Al igual que todas cuenta con el **nav-bar** en la parte superior. La parte central es un formulario en el que el usuario elegirá una serie de características y la aplicación le devolverá los países concordantes (si existen). A la izquierda del formulario se puede apreciar un gif de un avión moviéndose con un pequeño mensaje sobre la funcionalidad del formulario.

The screenshot shows a web page titled "Destino" with a dark blue header featuring a small airplane icon. Below the header, a message says "¡¡Estás a 30 segundos de conocer tu futuro destino!!". The main form area has a white background. At the top right are two buttons: "Nuevo" and "Repto". Below them is the question "¿A dónde quieres ir?". The left side of the form contains several input fields with radio buttons:

- Hemisferio: Norte (radio button), Sur (radio button), Indiferente (radio button selected).
- Continente: No (radio button selected), Si (radio button).
- Idioma: No (radio button), Si (radio button selected).
- Covid: No (radio button), Si (radio button).

The right side of the form lists various characteristics with checkboxes:

- Playa, Montaña, Cultural, Safari, Exótico, ONG, Peligroso, Turístico.

Below these lists is a blue button labeled "¿Cuál será tu destino?". At the bottom center is a large blue "Buscar" button. In the bottom right corner, there is some very small text that appears to be a link or a note.

La parte central del formulario se divide en una parte superior en la que aparece el título con un botón que tiene dos posiciones (nuevo o repito). La parte central del formulario se divide en dos zonas, la de la izquierda que son unos *radio inputs* que, salvo los primeros (hemisferio), la funcionalidad es la de mostrar u ocultar un *select*; es decir continente → No (no muestra nada), continente → Si (muestra un *select* para elegir el continente por el que filtrar).

This is a zoomed-in view of the "destino.html" search form. It shows four dropdown menus with radio buttons:

- Continente: Options are "No" (radio button) and "Si" (radio button selected). The "Si" option is followed by a dropdown menu containing "Europa".
- Idioma: Options are "No" (radio button) and "Si" (radio button selected). The "Si" option is followed by a dropdown menu containing "Español".
- Covid: Options are "No" (radio button) and "Si" (radio button selected). The "Si" option is followed by a dropdown menu containing "Pasaporte Covid".

La parte de la derecha permite filtrar atendiendo a otro tipo de características (más centradas en lo que busca un viajero) son unos *checks* debido a que el país tiene o no tiene: playa, montaña, cultural, safari, exótico, ONG, peligroso o turístico.

A vertical list of checkboxes labeled "Características" on the left:

- Playa
- Montaña
- Cultural
- Safari
- Exótico
- ONG
- Peligroso
- Turístico

La parte inferior es una zona para establecer un mensaje que puede ser de 3 tipos: 1) neutral (letra blanca sobre fondo azul), 2) de error (letra roja oscura sobre fondo rojo claro) o 3) de éxito en la búsqueda (letra verde oscura sobre fondo verde claro).

¿Cuál será tu destino?

No hay ningún país que concuerde

Listado de países

Selecciona tu próximo destino

Australia

Austria

Azerbaiyán

Bahamas

Selecciona tu próximo destino



Si el usuario pulsa aceptar se marcará como deseo y si pulsa en cancelar se cerrará el modal.



## Cultura

Cultura se divide en 4 páginas, una en la que se muestra información mediante texto y 3 mediante mapa (aprovechando toda la información extra que se cargó a las *features* en el GeoJSON). La primera es [\*paises.html\*](#).

Es una página simple que muestra el logo de la web y la posibilidad de elegir un continente en un *select*.



Atendiendo al continente elegido, los países le serán mostrados al usuario en la parte inferior, por ejemplo, Europa.

Lo mostrado son unos enlaces que van a permitir elegir al usuario un país del continente seleccionado para que se le muestre información, por ejemplo, Austria. Cuando se pulsa la página se transformará totalmente viéndose sólo la información del país en cuestión.

Además de la bandera y la información del país se muestra un botón que permite “salir” y volver a la situación previa en la que se puede elegir un continente.

Las tres páginas siguientes se explican juntas puesto que son idénticas salvo la leyenda y la forma de “pintar” el mapa. Son exactamente la misma página que *mapa.html* pero no “pintan” el mapa atendiendo a la BBDD y a las elecciones del usuario, sino a sus propias características: población, emisiones CO<sub>2</sub> o PIB.



Como se puede ver en la imagen tiene la misma funcionalidad que mapa (capas, leyenda dinámica, leyenda estática que indica la representación de colores, etc.), simplemente cambia que con el *click* no se elige si se ha visitado o no, sino que muestra un *pop-up* con la información de ese país (la misma que en la leyenda dinámica). Las otras dos páginas son idénticas, pero con su información.



## login.html

Las dos páginas que restan son muy similares en diseño (compartirán archivo CSS) y son dos pequeños formularios, uno de ingreso a la página (usuario y contraseña) y otra de registro. La primera es muy simple.



La página tiene validación de datos mediante JS, aunque PHP tiene las suyas propias, controla que los campos estén rellenos (también HTML mediante *required*), que no sean mayores a 16 caracteres y que el nombre no sea sólo números o contenga caracteres especiales.

Usuario no puede estar vacío ni contener más de 16 caracteres.



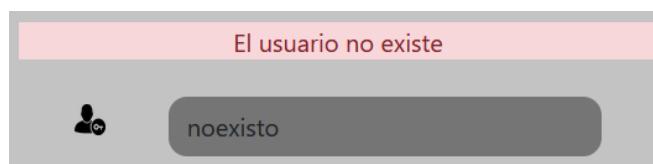
Ídem con el campo “contraseña”.



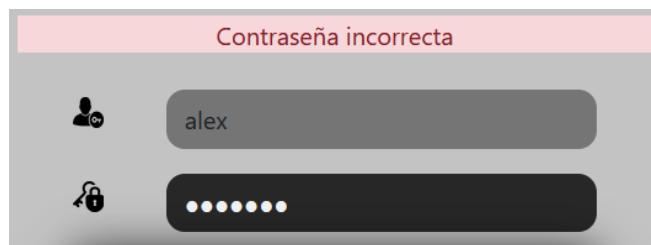
El campo nombre tiene una validación especial.



Por su parte, **PHP**, además de estas validaciones, puede mostrar mensajes como que el usuario no existe.



También puede darse la situación que la contraseña introducida no es correcta.



Si, por el contrario, el usuario introduce los datos correctamente se inicia la sesión y se redirige al usuario al *index*.



La característica especial de esta página es que si el usuario entra en esta página habiendo iniciado sesión se transforma y, en lugar de permitir el ingreso (que ya no es necesario), permite cerrar sesión, la modificación de la contraseña o borrar la cuenta.



Si se pulsa en cambiar la contraseña aparece el campo contraseña para introducirla y un nuevo campo “validar” para asegurarnos que introduce la misma. Si el usuario la cambia correctamente se le muestra un mensaje, pero, si introduce algún dato mal, se le muestra un error (idéntico a los mostrados previamente).

The screenshot shows a user interface for changing a password. At the top, a blue header bar says "Introduce la nueva contraseña". Below it are two input fields: the first has a lock icon and contains five black dots; the second also has a lock icon and contains four blue dots. A large dark blue button labeled "CAMBIAR" is at the bottom.

The screenshot shows a green header bar with the text "Pass cambiado: alex". Below it is a grey footer bar with three options: "Cerrar Sesión", "Cambiar contraseña", and "Borrar usuario".

Si se pulsa en borrar, se mostrará un botón de confirmar el borrado. Si el usuario lo pulsa (confirma el borrado del usuario), se borra ese usuario y el formulario vuelve a mostrar las opciones principales de usuario y contraseña.

The screenshot shows a confirmation dialog with the text "Hola alex" at the top. A dark blue button in the center contains the text "¿Borrar a alex?".

The screenshot shows a "Inicia sesión" (Log in) page. A green header bar says "El usuario alex ha sido eliminado". Below it are two input fields: the first has a user icon and says "Usuario"; the second has a lock icon and contains four black dots. A large dark blue button labeled "ENTRAR" is at the bottom.

Desde cualquiera de las opciones siempre podemos pulsar un enlace que lleva a la otra página relacionada con los usuarios, a *registro.html*.

registro.html

De estilo idéntico y con idéntica funcionalidad, el único cambio es que, en lugar de permitir el acceso a la web, permite el registro. De la misma forma si el usuario introduce algún parámetro mal se le mostrará un error (tanto por JS como por PHP) o, si introduce todos los datos correctamente, se inicia la sesión y se le redirige al *index*, como en *login.html*.

Habría que recordar que los errores se comprueban tanto por JS como por PHP y que son: que ningún campo esté vacío y que tengan menos de 16 caracteres, que el campo nombre no sean sólo números y que el campo nombre no tenga ningún parámetro especial. En esta página todas estas características (salvo las particulares de “nombre”) las debe cumplir también el campo “confirmar” que es el campo en el que se confirma la contraseña para cerciorarse que el usuario escribe bien su contraseña. Además, en registro, el campo “contraseña” y el campo “confirmar”, como es evidente, deben coincidir. Por último, no se puede crear un usuario que ya existe.



Campos vacíos

Nombre no puede estar vacío	
Nombre	<input type="text" value="Nombre"/>
Contraseña no puede estar vacío	
Confirma pass no puede estar vacío	

Campos con más de 16 caracteres.

Nombre tiene que tener 16 caracteres como máximo

Nombre

Contraseña tiene que tener 16 caracteres como máximo

Confirma pass tiene que tener 16 caracteres como máximo

Campo nombre particular.

Nombre no puede ser un número o contener caracteres especiales

Nombre

Nombre no puede ser un número o contener caracteres especiales

Nombre

El campo contraseña y confirmar deben coincidir.

Debe introducir la misma contraseña

No se puede crear un usuario que ya existe.

Ya existe ese usuario

Nombre

Cuando se crea correctamente se redirige al *index* con la sesión iniciada.

Bienvenido: nuevo ▾

### *Responsive*

La aplicación, gracias a *Bootstrap*, se encuentra diseñada para su correcta visualización desde cualquier tipo de dispositivo. A continuación, se muestran las diferentes páginas visualizadas en un teléfono y en una *Tablet*.

En la página de *index.html* podemos identificar las tres partes principales de la página.

**Funciones principales de Mis Viajes Web**



**Mapa**

Identifica en un mapa los países en los que has estado o quieras estar

[Mapa »](#)



© 2021 Alejandro Calvo, Inc. · [Back to top](#)

[Privacy · Terms](#)

Ahora en su versión Tablet.

**Funciones principales de Mis Viajes Web**

**¿Has viajado tanto que no sabes cuál puede ser tu próximo destino? Deja que Mis Viajes lo descubra por ti**

Selecciona tus preferencias como el idioma, continente, tipo de ocio, etc. Nuestra dilatada experiencia en este mundo te indicará las mejores opciones para tí

© 2021 Alejandro Calvo, Inc. · [Privacy · Terms](#) · [Back to top](#)

También sucede con el *navbar* (en una Tablet no se comprime).

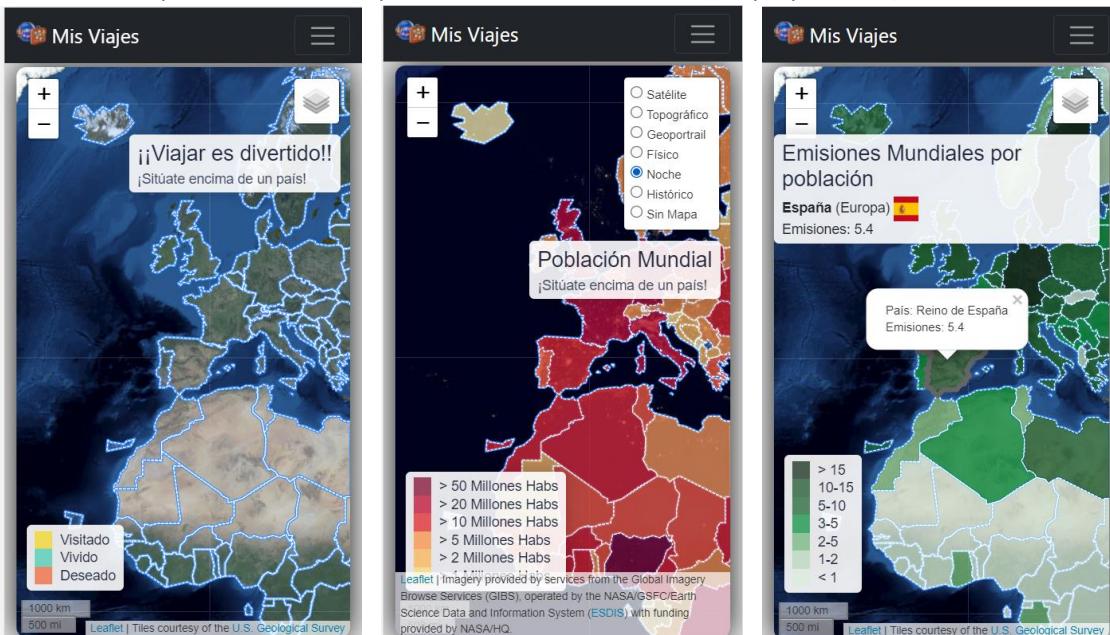


La página de introducción, para pantallas pequeñas.

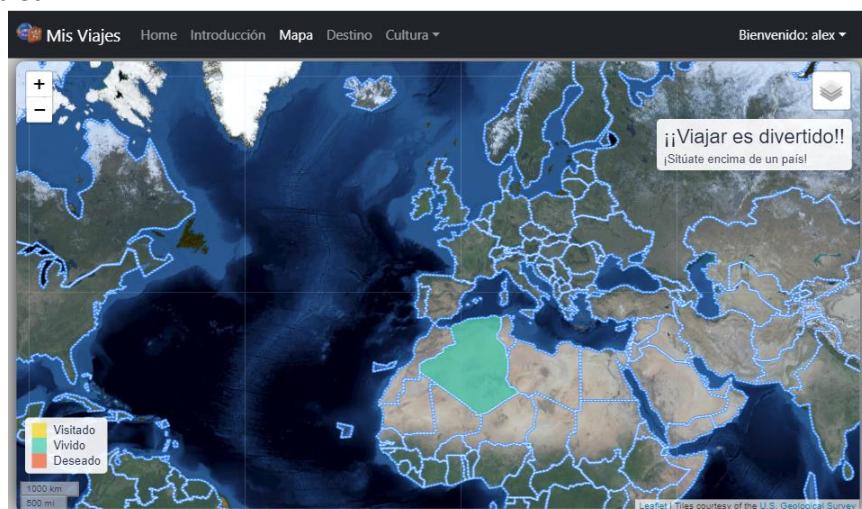
Para pantallas más grandes.

*Unauthorized.html*

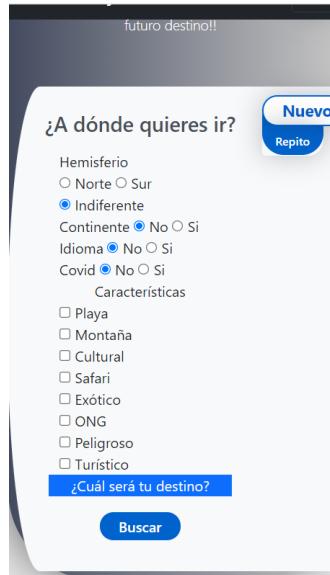
El mapa también se adapta, en todas sus versiones (mapa y cultura).



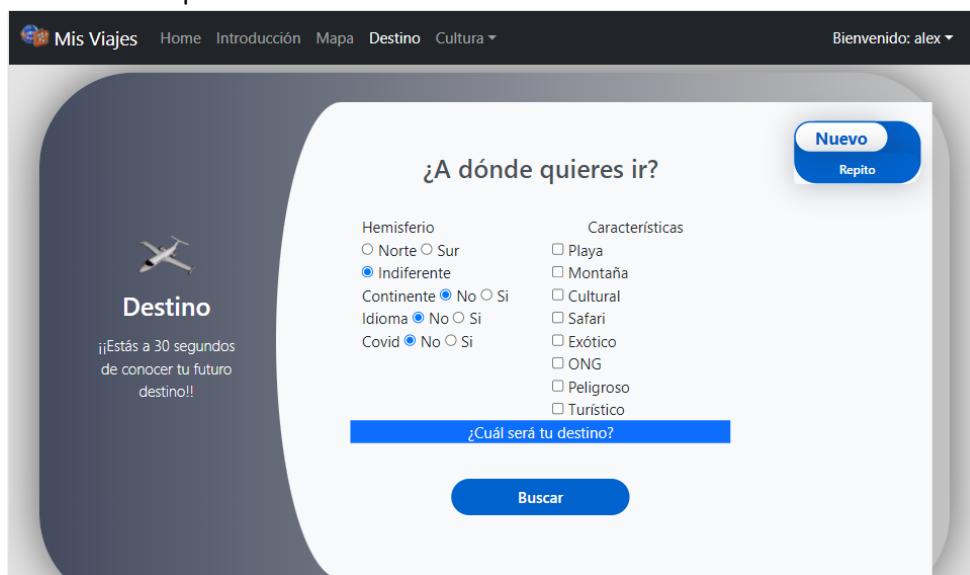
Tablet.



Destino.html en una versión para teléfonos móviles.



Destino.html para tabletas.

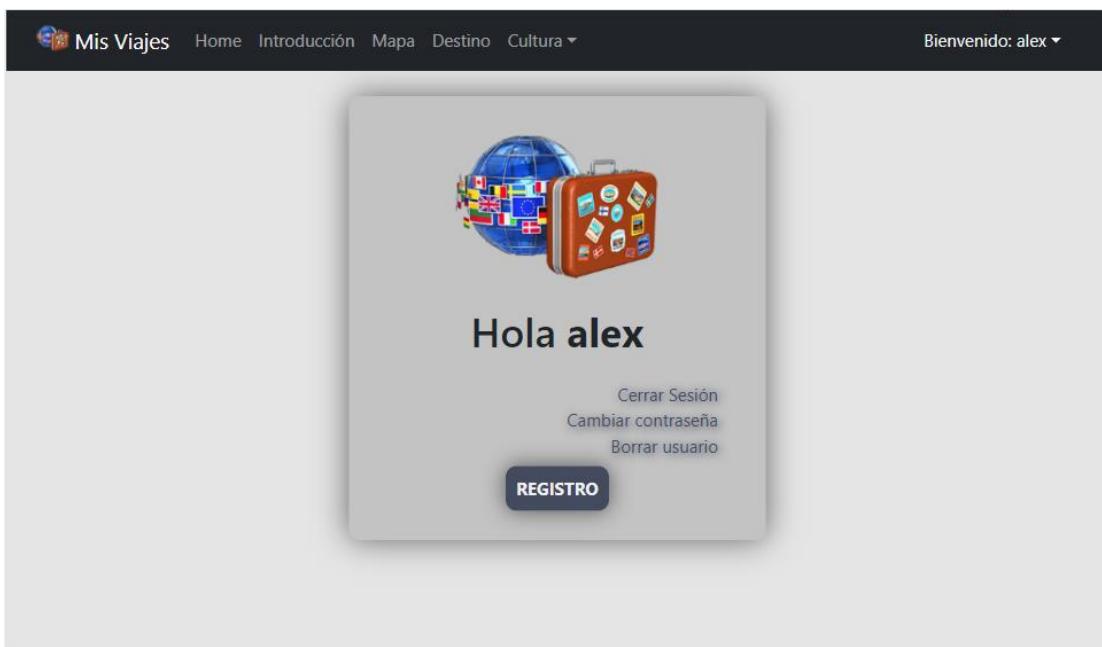
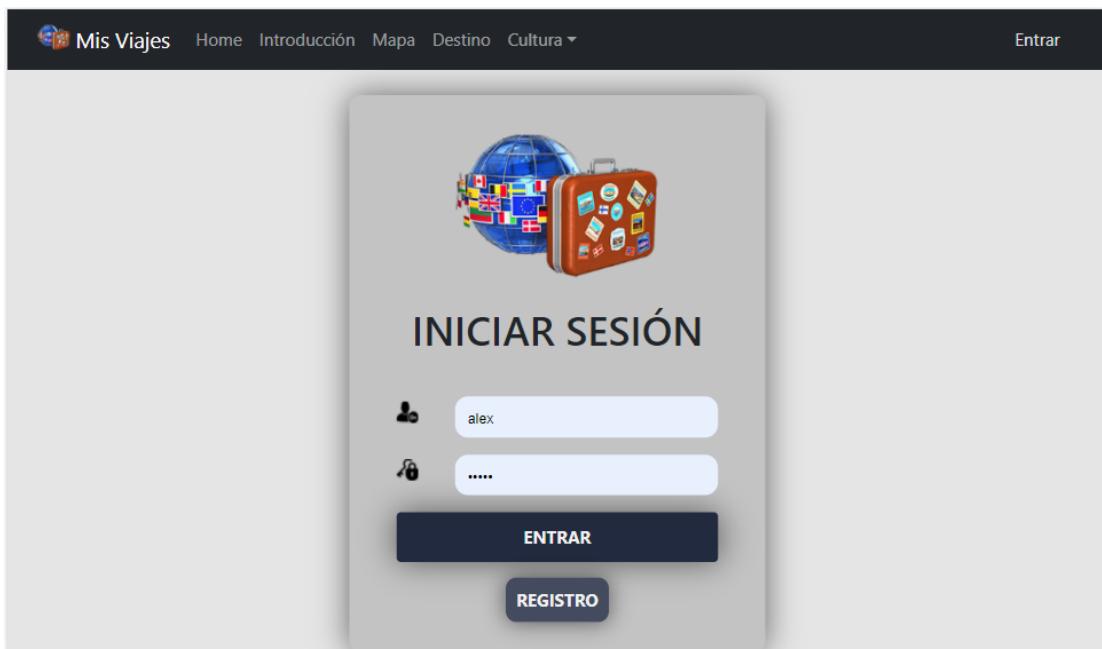


La página países.html (dentro de cultura).

Mostrada en una Tablet.

Por último, las páginas de *login.html* y de *registro.html*.

Para una Tablet.



## Registro.



# Mis Viajes



## REGISTRO

Nombre

Contraseña

Confirmar

Ya estoy registrado



# Mis Viajes

Home Introducción Mapa Destino Cultura ▾

Bienvenido: alex ▾



## REGISTRO

Nombre

Contraseña

Confirmar

## Funcionalidad

Ya se ha explicado el funcionamiento de la web a nivel de usuario, ahora se va a explicar cómo se han implementado las funcionalidades explicadas<sup>11</sup>.

### [index.html](#) e [introduccion.html](#)

En estas dos páginas sólo existen dos funcionalidades implementadas mediante JS. La más sencilla es la implementación de un *listener* de tipo *click* a los enlaces que mostrarán los modales privacidad y términos:

```
TERMS.addEventListener('click',function(){
    MODAL_TERMS.classList.add('show');
});
```

Con su correspondiente *listener* a los enlaces a cerrarlos.

```
document.querySelectorAll('.closeModal').forEach(c => {
    c.addEventListener('click',function(){
        MODAL_PRIVACY.classList.remove('show');
        MODAL_TERMS.classList.remove('show');
    });
});
```

**La otra funcionalidad es una común a todo el sitio.** Recordemos que si el usuario tiene sesión iniciada el enlace del *nav-item* “Entrar” se modifica por un *dropdown-item* con dos opciones. Para eso se invoca a una función que se encuentra en [loginLogout.js](#) denominada **setLogInOut\_index\_HTML(LOG\_IN\_OUT)**.

```
function setLogInOut_index_HTML(LOG_IN_OUT){
    if(sessionStorage.getItem('nombre')){
        setAreaPersonal(LOG_IN_OUT, sessionStorage.getItem('nombre'));
        document.getElementById('entrarCarousel').innerHTML="Área Personal";
    }
}
```

Recibe como parámetro una referencia al *nav-bar* “entrar” y se comprueba si existe una sesión iniciada. Si fuera así se invoca a una función para cambiarlo. La llamada a esta función se hace por modularidad y por no repetir código, puesto que habrá que hacerla por cada una de las páginas. La función se denomina para ser descriptiva: *settea* el enlace que hace *log-in* o *log-out*, en este caso en la página de *index*.

La verdadera función que lo modifica es **setAreaPersonal(LOG\_IN\_OUT,nombre)**.

```
function setAreaPersonal(LOG_IN_OUT, nombre){
    LOG_IN_OUT.href = "#";

    //Se usa ruta relativa para que no de problemas con la organización de carpetas
    const AREA_PERSONAL =
        "<div class='nav-item dropdown'>" +
        "  <a class='nav-link dropdown-toggle text-light p-0' href='#' id='dropdownEntrar'" +
        "    data-bs-toggle='dropdown' aria-expanded='false'>Bienvenido: "+nombre+"</a>" +
        "  <ul class='dropdown-menu bg-dark' aria-labelledby='dropdownEntrar'>" +
        "    <li><a class='dropdown-item bg-dark' href='/HTML/login.html'>Área Personal</a></li>" +
        "    <li><a class='dropdown-item bg-dark' href='#' id='logoutAP'>Log Out</a></li>" +
        "  </ul>" +
        "</div>";

    LOG_IN_OUT.innerHTML = AREA_PERSONAL;

    //Listener para el área personal creado ahora
    const LOG_OUT_AP = document.getElementById('logoutAP');

    LOG_OUT_AP.addEventListener('click',function(){
        LOG_OUT_AP.setAttribute('href','');
        sessionStorage.clear();
        location.href = "/index.html";
    });
}
```

<sup>11</sup> Nota, no se muestra todo el código puesto que se encuentra enlazado como anexo, sino sólo las partes importantes que merecen especial mención

Como se puede apreciar recibe el enlace que tiene que cambiar (la sesión del usuario, puesto que, si no, no se modificaría) y mediante una constante que tiene el nuevo código HTML se lo *settea* mediante *innerHTML*. Además, le añade un *listener* al enlace creado en esa constante *logOutAP* porque si el usuario la pulsa es que desea cerrar la sesión.

Esto va a ser igual para todas las páginas salvo para *mapa.html* y para *destino.html* puesto que como es obligatorio que tengan sesión iniciada, no se hace la comprobación mediante el *if-else* (el resto sí que es idéntico).

### mapa.htm

La funcionalidad principal es la carga del mapa en un *div*. HTML.

```
<!-- MAPA -->
<main class="container-fluid">
    <div id="map"><!-- Aquí va el mapa que se carga por JS --></div>
```

Carga del mapa por JS.

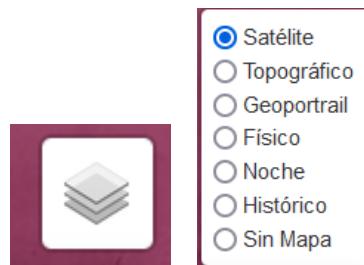
```
// 1) --> Creación del mapa

//Variable que recibe el mapa situado en el "centro" del mundo
//No se ejecuta sólo para añadirle funcionalidades extra
let map = L.map('map', {
  center: [41.66, -4.72],
  zoom: 3,
  layers: [SAT_US] // Adquiere el nombre del primer layer, pero están todos (CONST.js)
});

// 2A) --> Se le añade una escala de tamaño para guiar al usuario en el tamaño
L.control.scale().addTo(map);

// 2B) --> Se le añade el selector de capas por defecto
L.control.layers(BASEMAPS).addTo(map);
```

Se crea una variable *map* que invoca a *L.map* de la librería *Leaflet* y se le establece una posición inicial (España) con un zoom y las *layers* que se quieran (diferentes capas que va a tener el mapa). Las *layers* son la forma en la que se va a ver el mapa, son las siguientes:



En código (se almacenan en un archivo externo de JS denominado CONST.js)

```
const SAT_US = L.tileLayer('https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/{z}/{y}/{x}', {
  maxZoom: 20,
  attribution: 'Tiles courtesy of the <a href="https://usgs.gov/">U.S. Geological Survey</a>'
}),
WORLD_PHYSICAL = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Physical_Map/MapServer/tile/{z}/{y}/{x}', {
  attribution: 'Tiles &copy; Esri &mdash; Source: US National Park Service',
  maxZoom: 8
}),
TOPOGRAPHY_US = L.tileLayer('https://basemap.nationalmap.gov/arcgis/rest/services/USGSTopo/MapServer/tile/{z}/{y}/{x}', {
  maxZoom: 20,
  attribution: 'Tiles courtesy of the <a href="https://usgs.gov/">U.S. Geological Survey</a>'
```

Como se puede apreciar son unas *tileLayer* que se agregan a L y se le establece un zoom máximo y el enlace al desarrollador al ser *Creative Commons by*.

Con el mapa cargado y sus *Layers*, lo siguiente (opcional) era añadir una escala para ver el tamaño y el selector de *layers* por defecto, para no tener que desarrollarlo manualmente (fotos previas).

Ya está nuestro mapa creado y funcionando, pero en *Mis Viajes* se quiere que sea interactivo, por lo que hay que añadir la variable *countries* que se encuentra en *mapa.js* que almacena los países dibujados. Además, se le añaden dos leyendas, una estática que indicará al usuario que representa cada color en el mapa y una dinámica para que el usuario reciba información de ese país.

```
// 3) --> Se le añaden las capas de GeoJson --> A partir de este momento el mapa es interactivo
L.geoJson(countries).addTo(map);

// 3A) --> Se le da el estilo a todas las capas (el de defecto, el hover y el usuario lo pueden cambiar)
// El de por defecto incluye ya los países actualizados con la BBDD
let geojson = L.geoJson(countries, {
  style: setStyle,
  onEachFeature: onEachFeature
}).addTo(map);

// 3B) --> Se le añade una leyenda inferior izquierda para identificar el país
setColourLegend().addTo(map);

// 4) --> Variable que controla el flujo de información a la leyenda superior derecha
let info = L.control();
//--> Método que forma dicho contenedor para modularidad
setLegend();
```

Como se puede observar es un código muy limpio y sencillo y eso es debido a que no hay nada desarrollado, sino que por modularidad simplemente se invocan a unas funciones que tienen encapsuladas esos pedazos de código. Estas funciones se encuentran en *funcionesMapa.js*. Resulta interesante, antes de pasar a dicho archivo de JavaScript, como al archivo *geoJSON* se le añaden además de los países, el estilo de cada uno de ellos (*setStyle*) y qué hacer para cada una de las *features* (países).

La primera función reseñable sería la que da el estilo a cada mapa ***setStyle(feature)***.

```
function setStyle(feature) {
  return {
    weight: 1, //Tamaño de las formas
    opacity: 1, //Opacidad toda la forma
    color: '#FFFFFF', //Color de toda la forma
    dashArray: '2', //Borde continuo 1 o no
    fillColor: setColor(feature.properties.ID),
    fillOpacity: setOpacity(feature.properties.ID)
      //Relleno interior y opacidad de forma dinámica
  };
}
```

*SetStyle* settea el estilo para los mapas. En este caso, al tener que colorear atendiendo a los valores introducidos previamente se llaman a dos funciones que darán el color y la opacidad: ***setColor(ids)*** y ***setOpacity(ids)***.

```
function setColor(ids) {
  if(visitados.includes(ids)) return COLOR_VISITADO;
  else if(vividos.includes(ids)) return COLOR_VIVIDO;
  else if(deseos.includes(ids)) return COLOR_DESEO;
}

function setOpacity(ids) {
  if(visitados.includes(ids) || vividos.includes(ids) || deseos.includes(ids)) return 0.7;
  return 0.1;
}
```

El funcionamiento es sencillo. A `setOpacity` le van llegando los ids y si se ha visitado en cualquiera de sus formas devuelve 0.7 (más) o sino 0.1 (menos). `SetColor` devuelve un color para los países visitados, otro color para los países en los que se ha vivido y un tercero para los deseos futuros (para el resto, ninguno).

¿Cómo se sabe los países en los que se ha vivido o visitado? Mediante una llamada de AJAX a la API (la se explicará en las APIs); en el *front-end* lo único que interesa es que enviado el id del usuario que se ha registrado, se recibe una respuesta con todos los países que contiene “visitados”, “vividos” y “deseados”; asíque en `success` (conexión correcta con la API) se recorren esos arrays asociativos para recopilarlos en JS, que serán los que se les envían a las funciones previas.

```
success: function(response){
    //La respuesta es un array asociativo con "visitados" "vividos" "deseos"
    //Si existen resultados con ellos --> De array asociativo a simple
    if(response.visitados){
        for (var i=0; i<response.visitados.length;i++) {
            visitados[i] = response.visitados[i]["id_pais"];
        }
    }

    if(response.vividos){
        for (var i=0; i<response.vividos.length;i++) {
            vividos[i] = response.vividos[i]["id_pais"];
        }
    }

    if(response.deseos){
        for (var i=0; i<response.deseos.length;i++) {
            deseos[i] = response.deseos[i]["id_pais"];
        }
    }
},
```

En resumen, el color que ve el usuario nada más abrir mapa es debido a la llamada a `setStyle` (ya explicado) desde el mapa. Pero se le oferta al usuario a que interactúe con el mapa; eso va a ser posible con `onEachFeature` (método asignado a su correspondiente función con el mismo nombre de Leaflet).

**`onEachFeature(feature,layer)`** es una función que se asocia a cada una de las *features* (países) a las que se le pueden establecer unos *listeners*, en este caso han sido: *hover*, *mouseout*, *click* y *contextmenu*; es decir: al situarse encima del país, al salir del foco del país, al hacer *click* en el país y al hacer *click* con el botón secundario del ratón en el país.

```
function onEachFeature(feature, layer)
    //feature se ejecutaría siempre, ir
    layer.on({
        mouseover: setHover,
        mouseout: setHoverOut,
        click: insertOrDeleteCountry,
        contextmenu: zoomToFeature
    });
}
```

A cada *listener* se le asocia una función. La más sencilla es **`zoomToFeature`**. Si se hace *click* con el segundo botón del ratón, el mapa hará zoom en el país.

```
function zoomToFeature(layer) {
    map.flyToBounds(layer.target.getBounds());
}
```

Por su parte, en el *hover*, la función lo que hace es dar un estilo específico al país, para que el usuario sepa que está situado sobre él, por eso se vuelve a llamar a *setStyle* pero esta vez desde una *feature* específica. Además, hace un *update* de *info* (se explicará posteriormente).

```
function setHover(layer) {
    let seleccionado = layer.target;
    //let ids = layer.target.feature.properties.ID;
    seleccionado.setStyle({
        weight: 5,
        color: '#666',
        dashArray: '1',
        fillOpacity: 0.5
    });

    if (!L.Browser.ie && !L.Browser.opera && !L.Browser.edge)
        seleccionado.bringToFront();

    //Solo para control personalizado -> Le asigno una capa con
    info.update(seleccionado.feature.properties);
}
```

Evidentemente, cuando el usuario ya no está encima del país, no interesa que ese país tenga el color resaltado que se le había puesto, así que en el *mouseout* se quita.

```
function setHoverOut(layer) {
    geojson.resetStyle(layer.target);
    //Solo para control personalizado
    info.update();
}
```

*Info*, es una variable que recopila información de forma dinámica de la *feature* seleccionada. En *Mis Viajes* se utiliza para crear una leyenda dinámica en la parte superior derecha en la que se muestra que país es, la bandera del mismo y si se ha visitado: *setLegend()*.

```
function setLegend() {
    info.onAdd = function (map) {
        this._div = L.DomUtil.create('div', 'info');
        this.update();
        return this._div;
    };

    info.update = function (props) {
        this._div.innerHTML = '<h4>Población Mundial</h4>';
        if(props){
            this._div.innerHTML += '<span class="infoText">';
            this._div.innerHTML += '<b>' + props.NOMBRE_FORMAL + '</b> (' + props.CONTINENTE + ')';

            this._div.innerHTML += "<img src='../recursos/img/banderas/" +
                props.NOMBRE + ".png' width='25px' height='25px' /> <br />";

            if(visitados.includes(props.ID)){
                this._div.innerHTML += '<span class="infoText">Has visitado ese país</span>';
            }else if(vividos.includes(props.ID)){
                this._div.innerHTML += '<span class="infoText">Has vivido en ese país</span>';
            }else if(deseos.includes(props.ID)){
                this._div.innerHTML += '<span class="infoText">Deseas estar en ese país</span>';
            }else{
                this._div.innerHTML += '<span class="infoText">Nunca has estado en ese país</span>';
            }
        }else{
            this._div.innerHTML += '<span class="infoText">Sitúate encima de un país!</span>';
        }
    };

    info.addTo(map); //No devuelve, lo settea directamente
}
```

Lo hace en el país que se está seleccionando mediante el *hover* gracias a *setHover* y *SetHoverOut* que hacen un *update* de esta variable *info* creada aquí.

Antes de ver el último *listener*, el que tiene el desarrollo más complicado, se podría hacer referencia a la otra leyenda del mapa, a la estática que indica al usuario que significa cada país,

se hace en la función **setColourLegend()**, que simplemente crea un *div* que se añade a la parte inferior izquierda del mapa con esa información.

```
function setColourLegend(){
    let legend = L.control({position: 'bottomleft'});

    legend.onAdd = function (map) {
        let div = L.DomUtil.create('div', 'info legend');
        div.innerHTML += '<i style="background:' + COLOR_VISITADO + '"></i>' + VISITADO + '<br>';
        div.innerHTML += '<i style="background:' + COLOR_VIVIDO + '"></i>' + VIVIDO + '<br>';
        div.innerHTML += '<i style="background:' + COLOR_DESEO + '"></i>' + DESEADO + '<br>';

        return div;
    };
    return legend;
}
```

El *listener* establecido en el *click* de la *feature*; **insertOrDeleteCountry(layer)**. El parámetro recibido es el layer general, para acceder a la *feature* en la que se ha hecho *click* hay que acceder a la propiedad “*target*”, luego “*feature*” y finalmente a “*properties*” y la propiedad; en este caso interesan tanto el ID (para compararlo con la BBDD) como el nombre, para mostrárselo al usuario.

```
//Valores de la feature que necesito
let pais = layer.target.feature.properties.NOMBRE;
let id = layer.target.feature.properties.ID;
```

Se hacen las referencias a los modales y se crean los *listeners* para cerrarlos mediante la “X” existente en ellos. Se crea un *listener* general para todos que simplemente se provoca que, si se hace *click* en cualquiera, se les quite la clase “*show*” a todos.

```
//listener cerrar ambos modales (con la X roja)
document.querySelectorAll('.closeModal').forEach(c => {
    c.addEventListener('click', function(){
        MODAL_INSERT.classList.remove('show');
        MODAL_DELETE.classList.remove('show');
    });
});
```

En este momento tenemos dos opciones, que el país ya haya sido visitado o no, por lo que existe un *if-else* basado en ello.

```
if(visitados.includes(id) || vividos.includes(id) || deseos.includes(id)) {
```

Si se encuentra el ID en cualquier array quiere decir que ya se ha visitado, entonces lo que se hace es mostrar el modal para borrar la estancia junto con el nombre y la bandera del país.

```
MODAL_DELETE.classList.add('show');
MODAL_TEXT_DELETE.innerHTML = "¿Desea borrar su estancia en <b>" + pais + "</b>?";
MODAL_TEXT_DELETE.innerHTML += "<img src='../recursos/img/banderas/" + pais + ".png' width='25px' height='25px' />";
```

Y se le añade un *listener* al único botón con el que cuenta para borrar ese registro.

```
//Listener del botón del modal
document.getElementById("borrarButton").addEventListener("click", function() {
    deleteCountry(id, id_usuario);
});
```

En el *else* es la situación contraria, no se encuentra el ID en ningún array por lo tanto se muestra el modal de “visitado”, “vivido”, “deseo”; también con el nombre y la bandera. Además,

se añade un *listener* para cada uno de los 3 botones para que, si el usuario pulsa alguno, se cree el registro.

```
document.querySelectorAll('.visitadoListener').forEach(c => {
    c.addEventListener("click",function() {
        if(this.value >= 1 && this.value <= 3)
            insertCountry(id, id_usuario, this.value);
    });
});
```

Se comprueba que el valor está entre 1 y 3 (obligatorio) y si lo es, se llama a la función.

Sea cual fuere el destino *if o else*, cuando el usuario pulse un botón, ambos *listeners* llaman a una función, si es el botón desde el modal de borrar llama a **deleteCountry(id, id\_usuario)** y si es desde cualquier botón del modal de insertar llamará a **insertCountry(id, id\_usuario, visitado)**.

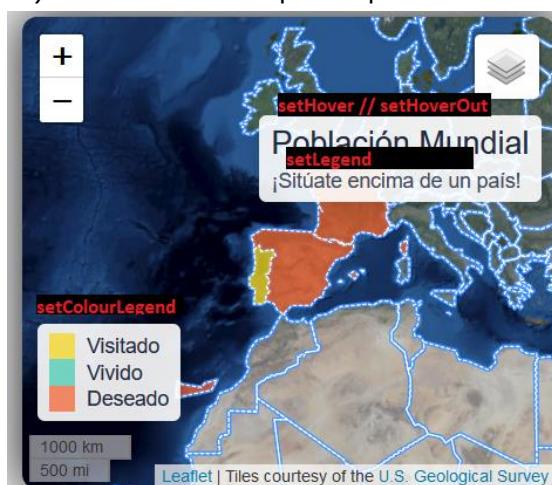
El funcionamiento de ambas funciones es la misma, es una llamada a la API, la única diferencia es que la función *deleteCountry* llama, como es evidente, a la API mediante un DELETE y la función *insertCountry* lo hará mediante un POST. En cualquiera de ambas funciones si todo ha funcionado correctamente se recarga la página para que se muestre la nueva situación y, si no, se cierra el modal que se encuentra mostrado y se muestra el modal de error con el error o “Error inesperado”. Habría que remarcar que existe una opción de colorear el polígono con una nueva llamada a *setStyle* con la referencia del país, pero, aunque sería más óptimo, generaba algún problema.

```
success: function(response) {
    if (response.success){
        //setStyle(layer.target.feature);
        location.reload();
    }
}
```

Por último, al igual que pasará con la página *destino.html*, sólo se puede acceder a esta página si el usuario tiene una sesión activa, por lo que en el inicio del *body* se comprueba y si no se le redirige a una web de acceso no autorizado.

```
<script>
    if(!sessionStorage.getItem('nombre')){
        window.location.href= "./unauthorized.html";
    }
</script>
```

En la siguiente imagen se muestra la propia página con las funciones asignadas. Aquellas asignadas al *click* como *insertOrDeleteCountry* o *zoomToFeature* no se pueden mostrar. Lo que sí se aprecia es como *setStyle* ha coloreado los países que encuentran coincidencia con el ID.



destino.html

Además de la llamada obligatoria a `setLogInOut_destino_HTML(LOG_OUT)`; destino también tiene funcionalidades propias. La más básica es la muestra de los `selects` si el usuario así lo decide.

```
RADIO_CONTINENTE_TRUE.addEventListener('click', function() {
    SELECT_CONTINENTE.classList.remove('oculto');
}) ;
```

El resto de la funcionalidad está implementada en `funcionesDestino.js`, aunque cuenta con mucho código es muy sencillo, va recopilando la información de todos los campos poniéndolos en variables que, posteriormente, serán enviados mediante la API a la BBDD para la consulta (GET). Recopilación.

```
if(document.getElementById('montaña').checked)
    montaña = 1;
else
    montaña= 0;
```

Envío mediante AJAX.

```
data: { "newOrNot": newOrNot,
        "hemisferio": hemisferio,
        "continente": continente,
        "idioma": idioma,
        "playa": playa,
        "montaña": montaña,
        "cultural": cultural,
        "safari": safari,
        "exotico": exotico,
        "ong": ong,
        "peligroso": peligroso,
        "turistico": turistico,
        "covid": covid,
        "id_usuario": id_usuario
```

En la respuesta puede suceder que no existan países, por lo que se muestra el mensaje con un tono adecuado (rojo) o que sí existan, por lo que se muestra el mensaje en verde junto con un `select` con los resultados.

```
}else{
    RESULTADO.classList.remove('alert-primary','alert-success','bg-primary','text-light');
    RESULTADO.innerHTML = "<p class='alert-danger text-center'>"+
        "No hay ningún país que concuerde</p>";
}
```

En el caso de que sí existan países.

```
textoResultado = "Listado de países ";
textoResultado += "<select id='paises' class='w-100 form-control' newDesire='newDesire'>";
textoResultado += "<option disabled selected>Selecciona tu próximo destino</option>";
//option "Selecciona tu próximo destino" --> Por si sólo hay 1 resultado
//disabled porque así le puedo poner listener sin problema

for (var i = 0; i < response.length; i++) {
    textoResultado += "<option class='paisesOption' value='"+
        response[i].id+"'>"+response[i].nombre+"

```

Se establece un *listener* al *select* de tipo *change* para que si el usuario pulsa sobre un país se muestre un mensaje modal para marcarlo como deseado o cerrar el modal.

```
//Le damos texto
MODAL_TITULO.innerHTML = "¿Deseas ir a <b>" + pais.target[pais.target.selectedIndex].text + "</b>? " +
"<img src='../recursos/img/banderas/" + pais.target[pais.target.selectedIndex].text + ".png'" +
"width='25px' height='25px' />";

//Listeners para cerrar el modal (Cancelar y X)
MODAL_BTN_CANCELAR.addEventListener("click",function(){
    MODAL.classList.remove('show');
});
MODAL_X_CERRAR.addEventListener('click',function(){
    MODAL.classList.remove('show');
});
```

Mediante AJAX si el usuario ha pulsado aceptar, se registra en la BBDD y se redirige al usuario a *mapa.html* para que pueda ver que el país ha sido marcado.

```
success: function(response) {
    if (response.success == "1")
        window.location.href="mapa.html";
```

cultura.html

Las tres opciones de cultura con mapa son idénticas a lo explicado con mapa, lo único que cambia el mensaje de las leyendas, por lo que simplemente se adapta en funciones diferentes (ubicadas en funcionesCultura.js) que serán llamadas desde cada una de las páginas, por ejemplo, en emisiones, función de la leyenda estática que devuelve un color atendiendo a las emisiones recibidas.

```
function setColorEcology(emisiones) {
    return emisiones > 15 ? '#08210F' :
    emisiones > 10 ? '#134f23' :
    emisiones > 5 ? '#155E29' :
    emisiones > 3 ? '#008F39' :
    emisiones > 2 ? '#70B578' :
    emisiones > 1 ? '#B8DABA' :
    '#DBEDDC';
}
```

Así será con todas, se denominan igual que las de mapa, pero reciben un “apellido” atendiendo a la página que las va a llamar y la funcionalidad se adapta a su necesidad.

Por su parte, **paises.html** sí que es completamente diferente. Existe un *listener* que crea un *div* con los países que tiene el continente que debe seleccionar el usuario.

```
//Creación de los enlaces a los países
for (var i = 0; i < paises.length; i++) {
    DIV_PAISES.innerHTML +=
        "<div class='col-sm-2 p-3 mb-2'>" +
        "<a href='#' class='shadow-link'>" + paises[i] + "</a>" +
        "</div>";
}

DIV_OCULTAR.classList.remove('oculto');
```

Para todos los enlaces creados, se establece un *listener* de tipo *click* que lo que hace es ocultar todo lo previo y mostrar la información del país: nombre, bandera e información recibida de la BBDD.

```
document.querySelectorAll('.shadow-link').forEach(a => {
  a.addEventListener('click',function(event) {
    let pais = event.target.text;
    DIV_OCULTAR.classList.add('oculto');
    DIV_SELECT_CONTINENTE.classList.add('oculto');

    document.getElementById("bandera")
      .setAttribute("src", "../../recursos/img/banderas/"+pais+".png");
    document.getElementById('titulo').innerHTML = pais;

    setCaracteristicas(pais);

  });
});
```

Estas características son recibidas a través de AJAX.

```
success: function(response) {
  if(response[0].nombre) {
    drawCaracteristicas(response[0]);
  }else{
    alert("Error inesperado");
  }
},
```

Desde el *success* es llamada a una función que las dibuja. Aquí se muestra las primeras opciones.

```
function drawCaracteristicas(caracts) {
  //En este momento está todo lo que no interesa oculto
  //Solo hay que desocultar donde irá el resultado
  const RESULTADO = document.getElementById('resultado');

  RESULTADO.innerHTML =
    "<p>El país <b>" + caracts.nombre + "</b>" +
    ", se encuentra en el hemisferio " + caracts.hemisferio +
    ", en el continente de " + caracts.continente + "</p>";
  RESULTADO.innerHTML +=
    "<p>El idioma oficial y mayoritario del país es el " + caracts.idioma +
    " y la capital es " + caracts.capital + "</p>";
```

Por último, se muestra el *div* que va a contener estos datos y se crea un botón que restaurará todo para que pueda elegir otro país si así lo desea el usuario.

```
//Se muestra
RESULTADO.classList.remove('oculto');

//Botón para salir
RESULTADO.innerHTML += "<input type='button' onclick='location.reload();' +
  " value='Salir' class='btn btn-lg mb-3' />";
```

## Funcionalidad de las APIs

Para no mezclar el código del *back-end* con código de *front*, se han creado 3 archivos PHP siguiendo, en la medida de lo posible, el estilo de una API. Como se puede ver en el esquema al inicio de esta documentación, la parte que se comunica entre los archivos HTML (a través de JS) con la BBDD son unas APIs que tienen unos archivos asociados PHP que se comunican con la BBDD.

Antes de comenzar con el desarrollo individual conviene dar una visión general de cómo funcionan las APIs; y es que no existe un contacto directo desde el primer PHP hasta la BBDD, sino que los datos se van manejando hasta la clase **DB.php** que es la que hace el contacto real. Desde modo quedaría así: cualquiera de las APIs invoca a un método estático de los archivos hijos de **DB.php**, quienes contactarán con su padre a través de otro método estático que puede ser: **executeQuery(\$sql)** o **modifyBBDD(\$sql)**.

Toda la lógica de las APIs sigue esta razón: ¿Qué tipo de datos se esperan recibir? → Petición al hijo de DB.php para que genere el \$sql necesario y → Contacto real con la BBDD (DB.php).

### Paises.php

Todas las APIs tienen una funcionalidad muy sencilla en PHP es una cláusula *switch* que evalúa el *REQUEST\_METHOD* enviado en la variable superglobal *\$\_SERVER*. Todas las APIs evalúan si es una petición *GET* (recibir datos), *POST* (enviar datos), *PUT* (modificar datos) o *DELETE* (borrar datos); aunque no se utilizan todas las cláusulas en las 3 APIs.

El archivo *Paises.php* relacionará el sitio web con la BBDD en su relación con los países, por lo que será llamada desde “mapa.html” principalmente, aunque también desde *cultura.html* (no se le hace una API propia puesto que sólo necesita *GET*). Por este motivo es la única que no sólo comprueba que el valor recibido esperado existe (por pura seguridad) sino que la condición evaluará que dato recibe para saber si es invocada desde *mapa.html* o desde cualquiera de las páginas de cultura.

```

case 'GET': //Recibir
    if(isset($_GET['id_usuario']) && !is_null($_GET['id_usuario'])) {...13 lines}

    //Llamada desde "Cultura"
    else if(isset($_GET['continente']) && !is_null($_GET['continente'])) {...11 lines}

    //Llamada desde "Cultura --> 2"
    else if(isset($_GET['pais']) && !is_null($_GET['pais'])){
        $response = DBPaises::getCaracteristicasPais($_GET['pais']);

        //Si está vacío --> Error
        if(empty($response)) {...3 lines}

        //Se devuelve la respuesta
        echo json_encode($response, true);
    }
}
break;

```

De este modo, como se puede observar en la imagen, si existe *id\_usuario* es que es llamada desde *mapa.html* e invocará un método de acceso a la BBDD; si no, si existe continente es llamada desde cultura e invocará otra función y, si no, si existe país llamará a la última de las funciones posibles. El desarrollo de cada uno de ellas es idéntico, por lo que sólo se explicará uno.

```

$response = DBPaises::getPaisesVisitados($_GET['id_usuario']);

//Si está vacío --> No hay países
if(empty($response)) {
    $response = array("SinPaises"=>"0");
}

//Se envía una JSON con los países o con "Sin países => 0"
echo json_encode($response, true);

```

Se invoca a un método estático de *DBPaises.php* del que se recibe una respuesta; si esa respuesta está vacía se crea un JSON con un mensaje de error “controlado” y si no se crea un mensaje de JSON con la respuesta.

Si por el contrario no se quiere recibir datos, sino insertar en la BBDD, el *request* será de tipo POST.

```

case 'POST': // Insertar
    if(isset($_POST['id_pais']) && isset($_POST['id_usuario'])
        && isset($_POST['visitado'])){
        $id_pais    = $_POST['id_pais'];
        $id_usuario = $_POST['id_usuario'];
        $visitado   = $_POST['visitado'];

        if(!empty($id_pais) && !empty($id_usuario) && !empty($visitado)
            && !is_null($id_pais) && !is_null($id_usuario) && !is_null($visitado)){

            //Llamada al método para insertar --> Debe devolver 1, sino --> error
            //Para insertar se necesita id_usuario, el país y el tipo de visita
            if(DBPaises::insertPais($id_usuario,$id_pais,$visitado) == 1)
                echo json_encode(array("success" => $id_pais),true);
            else
                echo json_encode(array("error" => "No ha sido posible"
                    . " actualizar"),true);
        }
    }
}

```

Se puede observar que si existen los datos adecuados y no son nulos lo que hace *POST* simplemente es invocar al método estático de *DBPaises.php* que se denomina *insertPais(\$id\_usuario, \$id\_pais, \$visitado)*; es decir, le envía los datos que han sido validados para que genere el \$sql adecuado y este, a su vez, se lo mande a su padre. Si la respuesta es 1 (una fila insertada => resultado esperado) se crea un JSON con “success” sino se crea uno con “error” que será lo que reciba *mapa.html* para manejar esta salida.

Como no se puede modificar la estancia en su país, sólo crearla o borrarla, “PUT” es inalcanzable.

---

```
case 'PUT': // Unreachable
```

Tanto PUT como DELETE no son “*request*” válidos, por lo que siempre que se utilicen hay que recibir los datos enviados mediante una sentencia *file\_get\_contents*. Por el resto es igual, tras validar los datos, estos, se envían a *DBPaises* para su manejo.

```

case 'DELETE': // Eliminar
    //Como no es ni post ni get necesito forzar la recepción
    parse_str(file_get_contents("php://input"), $variables);

    $id_pais = $variables['id'];
    $id_usuario = $variables['id_usuario'];

    //Si las variables no están vacías
    if(!empty($id_pais) && !is_null($id_pais)
        && !empty($id_usuario) && !is_null($id_usuario)){
        //Llamada al método de borrado de países --> Debe devolver 1
        //Para borrar se necesita el usuario y el país
        if(DBPaises::deletePais($id_pais, $id_usuario) == 1)
            echo json_encode (array("success" => $id_pais),true);
        else
            echo json_encode (array("error" => "No ha sido posible "
                . "borrarlo"),true);
    }
}

```

### DBPaises.php

Los datos validados en el archivo *Paises.php* llegan a *DBPaises.php* antes de ir a la BBDD. Esta clase, hija de **DB.php** lo que hace es recibirlos (ya validados) y crear una sentencia SQL que será la que se envíe a **DB.php**.

Sólo implementa métodos estáticos para que puedan ser invocados sencillamente desde los archivos previos sin tener que crear ninguna instancia. En el primer caso (GET) invoca a la función estática **getPaisesVisitados(id\_usuario)**; en este caso concreto se puede observar cómo se generan 3 consultas para luego crear un array asociativo con los países que se han visitado, se han vivido o se desean ir.

```

public static function getPaisesVisitados ($id_usuario) {
    $queryVisitados = "SELECT id_pais FROM paisesVisitados WHERE id_usuario = $id_usuario and visitado = 1";
    $queryVividos = "SELECT id_pais FROM paisesVisitados WHERE id_usuario = $id_usuario and visitado = 2";
    $queryDeseos = "SELECT id_pais FROM paisesVisitados WHERE id_usuario = $id_usuario and visitado = 3";

    //Realiza las 3 consultas
    $resultadoVisitados = parent::executeQuery($queryVisitados);
    $resultadoVividos = parent::executeQuery($queryVividos);
    $resultadoDeseos = parent::executeQuery($queryDeseos);

    //Crea un array vacío que será la unión de todos asociativamente
    $resultado = array();
    if(!is_numeric($resultadoVisitados)){//Si existen visitados, se añaden
        $resultado['visitados'] = $resultadoVisitados;
    }

    if(!is_numeric($resultadoVividos)){//Si existen vividos, se añaden
        $resultado['vividos'] = $resultadoVividos;
    }

    if(!is_numeric($resultadoDeseos)){//Si existen deseos, se añaden
        $resultado['deseos'] = $resultadoDeseos;
    }

    //Se devuelve el array de los 3 tipos de visitas (si existe)
    return $resultado;
}

```

El método invocado desde POST, lo único que hace es crear el SQL que enviará al padre.

```
public static function insertPaís($id_usuario, $id_pais, $visitado) {
    //Inserta la visita (1,2,3) a un país de un usuario
    $sql = "INSERT INTO paisesVisitados(id_usuario,id_pais,visitado)"
        . " VALUES($id_usuario,$id_pais,$visitado)";

    //Devuelve affected rows (1)
    return parent::modifyBBDD($sql);
}
```

Sucede exactamente lo mismo con DELETE (recordemos que los datos ya han sido validados).

```
public static function deletePaís($id_pais, $id_usuario) {
    //Borra la visita de un usuario a un país
    $sql = "DELETE FROM paisesVisitados WHERE id_usuario = $id_usuario AND"
        . " id_pais = $id_pais";

    //Devuelve affected rows (1)
    return parent::modifyBBDD($sql);
}
```

Deberían ser los únicos, pero como desde los países de cultura necesitan datos referentes a países y nunca necesitarán modificarlos recordemos que invocan también a esta API a través de GET, pero con otros parámetros. Los métodos que son llamados son los siguientes, teniendo la misma función que el primero (GET, también).

Preparo una consulta para mostrar los países filtrados por continente para *países.html*.

```
public static function getPaisesContinente ($continente) {
    $sql = "SELECT nombre FROM paises WHERE 1";

    if($continente != "Indiferente")
        $sql .= " AND continente = '$continente';";

    return parent::executeQuery($sql);
}
```

Preparo una consulta para mostrar las características de un país para la misma página.

```
public static function getCaracterísticasPaís($pais) {
    return parent::executeQuery("SELECT * FROM paises WHERE nombre = '$pais';");
}
```

O, por su parte, tenemos un método con mucho código puesto que recibe desde la API de *destino.php* puesto que también hace referencia a países. Nótese como se reciben gran cantidad de datos y sólo se muestra el inicio de la creación del SQL, habiendo mucha función que no aparece.

```
public static function getPaisesFiltrados($id_usuario, $playa, $montaña,
    $safari, $cultural, $song, $sexotico, $uristico, $peligroso, $covid,
    $hemisferio, $continente, $idioma, $newOrNot) {

    //La manera más sencilla para unir es dejar siempre un where (=1) y todo
    //lo que se anexe $unirConsultas.claúsula

    //Dame todos los nombres e ID de los países (where 1 --> Todos)
    $query = "SELECT nombre, id FROM paises WHERE 1";
    //String para unir las consultas
    $unirConsultas = " AND ";

    //Si se ha marcado playa --> Que tengan playa (1), sino, nada
    if($playa == 1)
        $query = $query.$unirConsultas."playa = 1";

    //Si se ha marcado montaña --> Que tengan montaña (1), sino, nada
    if($montaña == 1)
        $query = $query.$unirConsultas."montaña = 1";
```

### DB.php

La clase padre de *DBPaises* y *DBUsuarios* es la que invoca a la BBDD realmente. Para ello tiene 2 métodos estáticos *protected* (para que sólo los llamen los hijos) que son **executeQuery(\$sql)** y **modifyBBDD(\$sql)** que harán lo que su nombre dice: consultar o modificar la BBDD. Para realizar la conexión a la BBDD o para cerrarla hay dos métodos privados, que nadie puede invocar que son **getConnection()** y **closeConnection()**. De este modo la lógica es sencilla: recibe el SQL de *DBPaises* o *DBUsuarios*, abre la conexión, realiza lo que ese método tiene que realizar y cierra la conexión antes de devolver el resultado.

Método invocado para consulta, pide conexión, hace la consulta y devuelve el resultado, búsqueda vacío o error. Antes de devolver, siempre, cierra la conexión.

```
protected static function executeQuery($sql) {
    $bbdd = self::getConnection();

    if(isset($bbdd) && !empty($bbdd)) {
        $resultado = $bbdd->query($sql);
        $resultadoFetch = $resultado->fetchAll(PDO::FETCH_ASSOC);

        self::closeConnection();

        if(!empty($resultadoFetch)){
            return $resultadoFetch;
        }else{
            return 0;
        }
    }else{
        return 1;
    }
}
```

Para poder realizar la consulta, hay que conectar la base de datos.

```
private static function getConnection() {
    $bbdd = null;
    try {
        $opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
            PDO::ATTR_EMULATE_PREPARES => false); // Me devolvía Strings los
        $dns = "mysql:host=localhost;dbname=paisesDB";
        $dbUsuario = "viajesDB";
        $dbPass = "Viajes.2021";
        $bbdd = new PDO($dns, $dbUsuario, $dbPass, $opciones);
        $bbdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }catch (PDOException $e) {
        $error = $e->getCode();
        $mensaje = $e->getMessage();
    }
    return $bbdd;
}
```

Siempre hay que cerrar la conexión.

```
private static function closeConnection() {
    //Desconexión de la BBDD
    unset($resultado);
    unset($bbdd);
}
```

Por último, el método que sigue la misma lógica, pero para modificar la BBDD (*insert o update*).

```
protected static function modifyBBDD($sql) {
    $bbdd = self::getConnection();

    if(isset($bbdd) && !empty($bbdd)) {
        $resultado = $bbdd->exec($sql);

        self::closeConnection();

        return $resultado;
    }
}
```

### Destino.php

La lógica de las otras dos APIS es la misma. Destino puede pedir datos a *DBPaises* o crear un registro, por lo que habrá que analizar las “*request*” de tipo *GET* y *POST*.

Si existen todos los campos que se deben tener, se envían al método estático correspondiente (ya visto previamente). La respuesta será un error o los países filtrados.

```
$paises = DBPaises::getPaisesFiltrados($_GET['id_usuario'],
$_GET['playa'], $_GET['montaña'], $_GET['safari'],
$_GET['cultural'], $_GET['ong'], $_GET['exotico'],
$_GET['turistico'], $_GET['peligroso'], $_GET['covid'],
$_GET['hemisferio'], $_GET['continente'], $_GET['idioma'],
$_GET['newOrNot']);

//Si no está vacío, lo devuelves, si no se settea a 0 los países
if(!empty($paises))
    echo json_encode($paises,true);
else
    echo json_encode(array('paises' => 0),true);
```

Por su parte, si se quiere hacer un *insert* (*POST*) la función es de este modo.

```
//Siempre que el id_pais y el id_usuario no estén vacíos
if(!empty($id_pais) && !empty($id_usuario)
&& !is_null($id_pais) && !is_null($id_usuario)) {

//Llamada al método estático para insertar el registro en paisesVisitados
//Para insertar se necesita el país, el usuario y que tipo de visita
if(DBPaises::insertPais($id_usuario,$id_pais,$visitado) == 1)
    echo json_encode(array("success" => 1),true); // OK
else
    echo json_encode(array("error" => "No ha sido posible"
        . " actualizar"),true); // NO OK
}
```

Ambos métodos llaman a *DBPaises* que es quien llamará a su padre quién será el que ejecute las sentencias SQL.

### Usuarios.php

Su funcionamiento es idéntico. Sirve para consultar usuarios (*GET*), crearlos (*POST*), modificarlos (*PUT*) o borrarlos (*DELETE*).

Cuenta con una función validar, por si JS estuviera desactivado, que se llamará siempre para ver si los datos introducidos por el usuario son los correctos.

```
function validar($nombre, $pass) {
    //Misma validación que JS
    if( (empty($nombre) || empty($pass) )
        || ( is_null($nombre) || is_null($pass) )
        || ( strlen($nombre) > 16 || strlen($pass) > 16 )
        || is_numeric($nombre)//Pass si puede ser número
        || preg_match("/[^A-Za-z\d]/", $nombre) ){
            return false;
    }

    return true; // ni están vacío o son mayores de 16
}
```

En el caso de GET, se quiere conocer si el usuario y la contraseña son correctos.

```

if(isset($_GET['nombre']) && isset($_GET['pass'])){

    $nombre = $_GET['nombre'];
    $pass   = $_GET['pass'];

    if( validar($nombre, $pass) ){
        if(DBUsuarios::userExists($nombre)){ // ¿Existe? con ese nombre
            $response = DBUsuarios::checkUser($nombre, $pass);
            echo json_encode($response, true); // Si existe un JSON con nombre
        }else{
            //El usuario no existe
            echo json_encode(array("error" => "El usuario no existe"),true);
        }
    }else{
        //Ha fallado la validación, no se pormenoriza porque está en JS
        echo json_encode(array("error" => "Ningún campo puede estar vacío, "
            . "ser mayor de 16 caracteres o el nombre ser sólo "
            . "números o contener caracteres especiales"),true);
    }
}else{
    //Error inesperado
    echo json_encode(array("error" => "Error en el envío de datos"),true);
}
break;

```

Para eso tiene que invocar a dos métodos que se verán con *DBUsuarios*, primero **userExists(\$nombre)** para saber si ese usuario existe o no y **checkUser(\$nombre, \$pass)**, que es el que verdaderamente comprueba que el nombre y la contraseña son correctas.

En el siguiente pedazo de código se muestra la parte importante del POST, en la que si existen los valores necesarios y cumplen los requisitos (se validan los campos y el \$pass es idéntico que \$confirmar) siempre que el usuario no exista ya, se crea, sino se manda un mensaje de que ya existe y sino se van controlando otro tipo de posibles errores.

```

if(isset($_POST['nombre']) && isset($_POST['pass'])
    && isset($_POST['confirmar'])){

    $nombre    = $_POST['nombre'];
    $pass      = $_POST['pass'];
    $confirmar = $_POST['confirmar'];

    if(($pass == $confirmar) && validar($nombre, $pass)){
        //¿Existe ya?
        $userAlredyExists = DBUsuarios::userExists($nombre);

        //Si no existe, lo creo
        if(!$userAlredyExists){

            if(DBUsuarios::createUser($nombre,$pass) == 1){ //Creado
                $id_usuario = DBUsuarios::getId_Usuario($nombre);
                $createOrNot = array(
                    "nombre" => $nombre,
                    "id_usuario" => $id_usuario[0]['id_usuario']
                ); //Como se llama FetchAll en la clase padre
                    // y sólo se espera 1 resultado...
            }else{
                //Error en la creación
                $createOrNot = array("error" => "No ha sido posible crear el usuario");
            }
        }
    }
}

```

Los casos de PUT y DELETE son idénticos, se comprueban los valores y, si son acordes, siempre que exista el usuario, se modifica, si no, no se puede y se van controlando los errores.

```

if(validar($nombre, $newPass) && ($newPass == $confirmar) ){
    //Sólo si existe el usuario le puedo cambiar el pass
    if(DBUsuarios::userExists($nombre)) {

        //Cambiar pass --> Necesito el usuario y la nueva Pass
        if(DBUsuarios::updatePass($nombre, $newPass) == 1) //Actualizado
            $updateOrNot = array("nombre" => $nombre);
        else
            $updateOrNot = array("error" => "No introduzcas la "
                . "misma contraseña que anteriormente");

    }else{
        //No se puede cambiar si no existe el usuario
        $updateOrNot = array("error" => "El usuario no existe");
    }
}

```

DELETE.

```

//Para borrar un usuario, debe existir
if(DBUsuarios::userExists($nombre)){// Aunque se supone que no

    //Para borrar el usuario necesito el id_del usuario
    $deleteOrNot = DBUsuarios::deleteUser($id_usuario);

    if($deleteOrNot == 1) //Si se ha borrado 1 fila
        $result = array("borrado" => true); //Éxito
    else if($deleteOrNot['error'])
        $result = $deleteOrNot; // Si existe error, lo recojo
    else
        return false; // Salida inesperada
}else{
}

```

## DBUsuarios

Con la misma lógica que `DBPaises` tiene unos métodos estáticos que crean el SQL necesario. El primero `userExists($nombre)` crea un SQL para comprobar si existe el usuario.

```
public static function userExists($nombre) {
    //Select nombre del usuario que tiene ese nombre --> ¿Existe?
    $sql = "SELECT nombre FROM usuarios WHERE nombre = '$nombre'";
    $userExists = parent::executeQuery($sql);

    //Existe TRUE o no existe FALSE
    if(!empty($userExists) && $userExists[0]['nombre']==$nombre)
        return true;
    else
        return false;
}
```

También `checkUser($nombre, $pass)` que creará un SQL para comprobar la contraseña de un usuario.

```
public static function checkUser($nombre, $pass) {
    //Select nombre de usuario si existe ese usuario con ese password
    $sql = "SELECT nombre, id_usuario FROM usuarios WHERE nombre=" . "'$nombre' AND pass = md5('$pass')";

    $userPassCorrect = parent::executeQuery($sql);

    //DB.php solo tiene "FetchAll" --> Por lo que se recibe la posición
    //Sólo se espera 1 resultado
    if(isset($userPassCorrect[0]['nombre']) && $userPassCorrect[0]['nombre'] == $nombre)
        return $userPassCorrect[0]; //Devuelve el array

    elseif(is_numeric($userPassCorrect) && $userPassCorrect==0) // Error
        return array("error" => "Contraseña incorrecta");
    else
        return array("error" => "Error en la Base de Datos");// Error
}
```

Es necesario en alguna circunstancia conocer el ID de un usuario específico: `getIdUsuario($nombre)`.

```
public static function getId_Usuario($nombre) {
    //Devuelve el ID del usuario atendiendo al nombre recibido
    $sql = "SELECT id_usuario FROM usuarios WHERE nombre = '$nombre';

    //Devuelve el id
    return parent::executeQuery($sql);
}
```

Existen `updatePass($nombre, $pass)` que modificará la contraseña para ese usuario y `createUser($nombre, $pass)` que hacen prácticamente lo mismo salvando la diferencia entre que el primero el SQL cuenta con un UPDATE y el segundo con un INSERT.

```
public static function updatePass($nombre, $pass) {
    //Modificar el campo pass con el nuevo pass para el usuario que se ha recibido
    $sql = "UPDATE usuarios SET pass = md5('$pass') WHERE nombre = '$nombre';

    //Devuelve directamente la ejecución (affected rows)
    return parent::modifyBBDD($sql);
}
```

```

public static function createUsuario($nombre, $pass) {
    //Inserta en la tabla de usuarios al nuevo usuario con este nombre y pass
    $sql = "INSERT INTO usuarios (nombre,pass) VALUES ('$nombre',md5('$pass'))";

    //return affected rows
    return parent::modifyBBDD($sql);
}

```

Por último, está el más “complicado” que es **deleteUser(\$nombre)** puesto que un usuario puede tener “viajes” en *paisesViajados* (tabla de la BBDD) y antes de eliminarlo hay que borrar esos registros tanto por optimizar la BBDD como por la obligación al tener esta tabla registros con una clave FK **id\_usuario** que es PK en la tabla usuarios.

```

public static function deleteUser($id_usuario) {
    //id_usuario es FK en paisesVisitados
    $userHasFKRegistersQuery = "SELECT id FROM paisesvisitados WHERE id_usuario = $id_usuario";

    //¿Existen registros como FK?
    $userHasFKRegisters = DB::executeQuery($userHasFKRegistersQuery);

    //Si la variable es array es que si existen campos, sino sería 0 o null
    if(is_array($userHasFKRegisters)){
        $deleteCountriesFKQuery = "DELETE FROM paisesvisitados WHERE id_usuario = $id_usuario";

        //Si existen datos --> Borrarlos todos
        if(!(parent::modifyBBDD($deleteCountriesFKQuery) >= 1))
            return array("error" => "Error en el borrado de los países que has viajado");
        // Si no ha borrado una fila o más y ha entrado aquí sal con error
    }

    //En este momento es cuando de verdad se borra el usuario
    $deleteUserQuery = "DELETE FROM usuarios WHERE id_usuario = $id_usuario";

    //Devuelve directamente el affected rows (se espera 1, un usuario)
    $result = parent::modifyBBDD($deleteUserQuery);

    return $result;
}

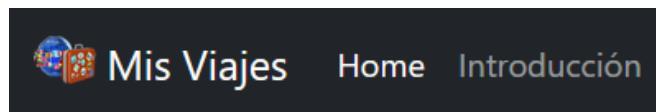
```

De este modo simplemente se comprueba si existen registros para ese usuario, existan o no (el *if* es opcional, no tiene *else*) se borra al usuario. Pero, en el caso de que hubieran existido, en el mencionado *if*, se provocaría el borrado de estos registros debido a que sino SQL mostraría un error de imposibilidad de borrado al contener la tabla *paisesVisitados* registros con un *id\_usuario* que es FK del usuario que se quiere borrar.

## Estilo

El estilo otorgado a la web es muy sencillo se utiliza predominantemente el azul en varias tonalidades y, también el blanco sobre negro. Todo ello para intentar facilitar la lectura por usabilidad y accesibilidad dentro de un intento de estética acorde a la temática.

El blanco sobre negro es elegido, principalmente, para el *nav-bar*.

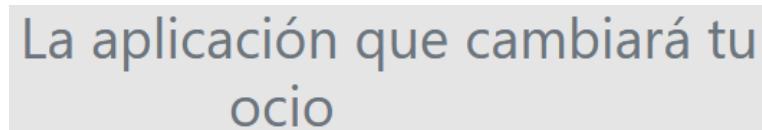


Pero el color predominante es el azul. El sitio está formado por un fondo de esta tonalidad (rgba(0, 0, 0, .1)).

Luego existen dos colores muy utilizados también, que serán un azul oscuro: #222A3F y un azul más claro #444B5F.



También es muy utilizado para textos largos el color “text-muted” propio de Bootstrap, que corresponde a #6C757D, un gris azulado muy suave.



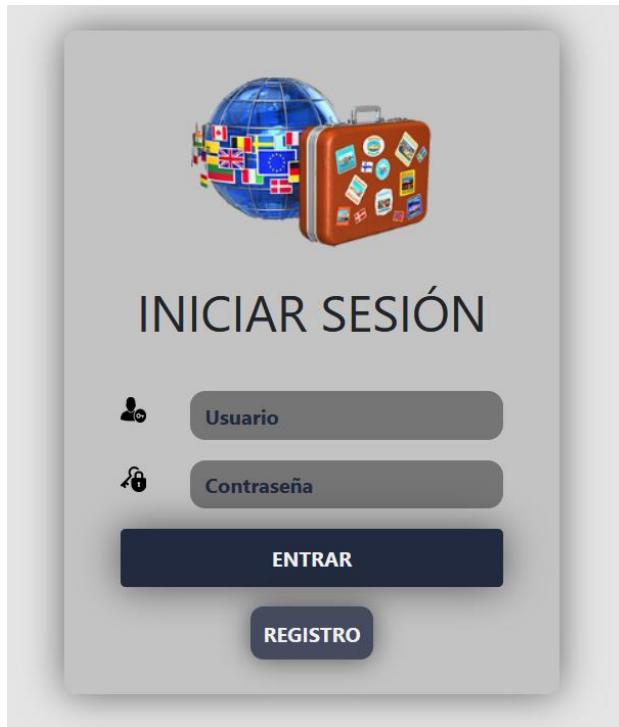
En los enlaces encontramos la combinación de estos colores. De este modo tenemos el blanco sobre azul oscuro para el botón, convirtiéndose en el azul claro con el sombreado de la misma tonalidad cuando se está en el *hover*.



Siguiendo la misma temática, en el *dropdown-button*, se utilizan estos colores.



Todas las páginas que tienen un recuadro (que son muchas) cuentan con una diferencia de tonalidades muy simple, se le otorga un poco más de opacidad para crear un tono azulado más oscuro, provocando contraste, ayudado por un ligero sombreado.



Respecto a la página destino, también sigue estos tonos de azules-grisáceos, aunque tiene un degradado hacia el gris claro para darle algo de dinamismo, contraste y diferenciarla del resto de páginas (es una funcionalidad principal). Los enlaces, botones y primera respuesta del formulario están en el azul típico de *Bootstrap "primary"*.

Por último, los modales, para darle algo de notoriedad, sin salir de esta escala de azules-grisáceos; cuentan con un degradado de blanco a gris-azulado (claro), con el mismo estilo de los botones (*hover* incluido).



## Conclusiones

Las conclusiones que se extraen de este proyecto son sumamente positivas. Mis Viajes es un proyecto real que funciona perfectamente y se encuentra accesible mediante internet de forma sencilla sin haber tenido que recurrir a ningún tipo de servicio o proveedor; habiendo sido desarrollado mediante los conocimientos adquiridos en el Ciclo Formativo de Grado Superior DAW junto con los conocimientos previos de la experiencia laboral adquirida gracias al Ciclo Formativo de Grado Superior DAM.

El desarrollo de un proyecto es algo vital para la consecución de las competencias necesarias para poder ejercer la profesión estudiada, pero, además, es el reto personal que el alumno se propone para su consecución. En este caso ha sido el intento de desarrollar una web estéticamente atractiva al mismo tiempo que funcional; conseguir una funcionalidad adecuada ha resultado sencillo, puesto que es uno de mis puntos fuertes como desarrollador. El aspecto estético ha sido un gran reto puesto que no poseo grandes conocimientos de diseño gráfico, su edición o desarrollo.

Por ello me decante por la investigación en un *framework* para su utilización y poder solventar esta debilidad. Pero, además, el interés de la utilización de *Bootstrap* era el intento personal de optar por utilizar formas de desarrollo que desconocía para completar mi formación; simultáneamente asimilaba la realidad de un *developer* en cuanto a una continua formación en nuevas tecnologías.

Todos los objetivos propuestos en el anteproyecto para el sitio web han sido cumplidos. Se ha creado el sitio de la forma que se estableció, se desplegó personalmente, como se había establecido, sin tener que recurrir a ningún tipo de *hosting* y *Mis Viajes* tiene un correcto funcionamiento.

La aplicación tiene un posible nicho de mercado que se encuentra vacío en estos momentos. Es cierto que, en la situación actual de desarrollo del proyecto, no es una web completamente viable y rentable económicamente, pero es la base a un posible negocio basado en los viajes. El desarrollo ha sido muy sencillo, ajustado a las horas que se establecen en el marco de la legislación vigente para los proyectos de fin de ciclo. No obstante, el sitio *Mis Viajes* tiene muchísimas posibilidades de mejora que pueden atraer a un mayor número de público y, sobre todo, conseguir rentabilidad económica.

**Como desarrollos potenciales y de perfeccionamiento**, se podrían crear muchas más propiedades en el mapa GeoJSON que permitirían la implementación de muchos más mapas dinámicos (o más completos), dando una posible utilidad real del sitio en términos de enseñanza; aunque no sea su funcionalidad principal. Tenemos que tener en cuenta el gran avance de las TIC en el proceso del sistema de enseñanza-aprendizaje y el sistema educativo de nuestro país (especialmente después de toda la situación provocada por la pandemia del COVID-19) y la gran cantidad de veces que los docentes, padres y alumnos recurren a estos métodos digitales para la muestra de información. El mapa enrollable encima de la pizarra, se estima, que ha pasado a mejor vida.

Pero no estamos sólo ante una posibilidad de negocio, sino que se abre una coyuntura de múltiples oportunidades. Actualmente tenemos una BBDD muy sencilla con datos básicos y relativamente estáticos, pero, ¿Dónde se informan cierto tipo de colectivos ante la posibilidad de viajes? Poniendo un caso específico: no sería idóneo que un colectivo LGTBI+ pudiera marcar una casilla que le indicara si es un país recomendable o es un país, **desgraciadamente**, hostil

para ellos. Y, sobre todo, el campo relativo a la pandemia de la COVID-19, si se mantiene actualizado y con información completamente contrastada es una posibilidad de crecimiento muy considerable para la aplicación.

De este mismo modo se tiene que tener en cuenta que el sitio puede crecer en protagonismo de varios modos. El primero sería poder compartir tu mapa con tus contactos. Todos sabemos que en nuestros días la mayoría de las personas utilizan de forma asidua alguna red social; especialmente notable en **Instagram** (en cuanto a “adicción” o muestra de “eventos sociales”). La posibilidad de compartir los viajes que se han hecho mediante este tipo de redes, como la posible interacción sobre ellos por varios usuarios, pudiera aumentar el número de visitas de nuestra web.

También sería interesante la recopilación de los principales destinos turísticos de los países para que saliera la información relativa a los usuarios. Si un usuario desea ir al Reino Unido, deberían aparecerle los posibles lugares a visitar: la *Abadía de Westminster*, el *Big Beng*, el *British Museum*, *Abbey Road*, etc. Incluso unas posibles rutas atendiendo a los días que quiere estar en el país: “Ruta para visitar Londres en 2 días”, “Ruta para visitar Londres en 1 semana”, etc.

Por último, el mayor nicho de mercado sería la posibilidad de la relación directa mediante una API con aplicaciones de gestión de viajes como **Booking** o similares. De esta manera no sólo se obtendría rendimiento económico por visitas, sino que, también, se podrían generar ingresos por referencias al ligar los enlaces desde *Mis Viajes* a otros portales web. Cuando un usuario de *Mis Viajes* marcará un país como futuro destino, saldrán automáticamente posibles ofertas de viajes, estancias, etc., provistas por estos agentes externos. En este sentido pudiera ser interesante también tener la opción de mostrar los territorios de los países de forma individual, pudiendo por ello marcar subdivisiones territoriales más pequeñas que las fronteras estatales como regiones, zonas naturales o turísticas, etc.

Con todo ello conseguiríamos un flujo de visitas apreciable y unos ingresos que harían rentable el proyecto, además de crear un sitio divertido y didáctico.

El proyecto es *CC-By-NC* aunque si se desarrollara en las vías establecidas debería ser propietario. Además, aunque se ha intentado dar una seguridad básica al sitio, es cierto que habría que invertir más en ciberseguridad, puesto que es uno de los mayores problemas en nuestros días para este tipo de desarrollos web.

En resumidas cuentas, el sitio web «***Mis Viajes***» ha sido creado de manera que pueda ser utilizado por todo tipo de usuarios de cualquier edad, tratando de dar la máxima accesibilidad posible. Cuenta con un estilo *responsive* que permite su correcta visualización desde cualquier tipo de dispositivo sin que esto genere un problema. Todo ello permite un uso agradable y divertido que incita a la utilización de la web durante un amplio periodo de tiempo.

## Bibliografía

Recopilación de información general sobre países

<https://www.un.org/en/about-us/member-states>

<https://www.saberespractico.com/geografia/paises/paises-de-africa/>

<https://www.saberespractico.com/curiosidades/cuantos-paises-tiene-america-del-sur/>

Información relativa a la gestión de la pandemia de la COVID-19 por países

[https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/exemptions-coronavirus-travel-restrictions-eu\\_es](https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/exemptions-coronavirus-travel-restrictions-eu_es)

<https://capturetheatlas.com/es/paises-de-africa-para-viajar/>

<https://www.lavozdegalicia.es/noticia/sociedad/2021/07/15/requisitos-pide-pais-vas-viajar/00031626350298334576909.htm>

<https://capturetheatlas.com/es/paises-de-sudamerica-para-viajar/>

<https://capturetheatlas.com/es/paises-abiertos-al-turismo/>

Información sobre la viabilidad del sitio

<https://mktefa.ditrendia.es/blog/todas-las-estad%C3%ADsticas-sobre-m%C3%B3viles-que-deber%C3%A1s-conocer-mwc19>

<https://datos.bancomundial.org/indicator/IT.NET.USER.ZS>

<https://es.statista.com/temas/3612/el-turismo-en-el-mundo/>

<https://www.epdata.es/datos/turismo-espana-mundo-datos-graficos/272>

<https://lamenteesmaravillosa.com/el-sindrome-wanderlust-la-obsesion-por-viajar/>

<https://www.psicologialflexible.com/es/falsa-apariencia-redes-sociales/>

<https://www.camarazamora.com/emprendimiento/vivero-de-empresas>

<https://www.ionos.es/digitalguide/dominios/consejos-sobre-dominios/cuanto-cuesta-un-dominio-web/>

<https://www.marketingguerrilla.es/lo-que-una-web-con-100-000-paginas-vistas-ingresa-con-publicidad-online-al-mes/>

Páginas con información para el desarrollo de la aplicación

GeoJSON

<http://geojson.io/>

<https://enterprise.arcgis.com/es/portal/latest/use/geojson.htm>

<https://geojson-maps.ash.ms/>

[https://datos.canarias.es/catalogos/estadisticas/dataset/dc327e7b-ddac-4fc4-afb0-445a6e191525/resource/a2ece3dc-f967-4ed7-8739-bb04e3809c3f/download/paises\\_2013\\_geom\\_60.json](https://datos.canarias.es/catalogos/estadisticas/dataset/dc327e7b-ddac-4fc4-afb0-445a6e191525/resource/a2ece3dc-f967-4ed7-8739-bb04e3809c3f/download/paises_2013_geom_60.json)

<https://gist.github.com/Kahlaoui-Ismail/00983b311620f50d598db60fc7b4c7d3>

<https://digitalki.net/es/2021/03/26/geojson-pais-lmites-datos-para-todos/>

Leaflet

<https://mappinggis.com/2013/08/como-crear-un-mapa-web-a-partir-de-un-shapefile/>

<https://leafletjs.com/examples/extending/extending-3-controls.html>

<https://leafletjs.com/examples/layers-control/>

<https://vivaelsoftwarelibre.com/anadir-y-modificar-los-controles-en-leaflet/>

<https://github.com/Leaflet/Leaflet/issues/690>

<https://www.youtube.com/watch?v=MDTIhWvIcDs>

<https://www.youtube.com/watch?v=Zy89Nj7tNNM>

<https://github.com/Leaflet/Leaflet/issues/690>

<https://gis.stackexchange.com/questions/75590/setstyle-function-for-geojson-features-leaflet>

Modal

<https://es.stackoverflow.com/questions/72918/como-crear-una-ventana-modal-con-tan-solo-html-y-css/72922>

<https://www.youtube.com/watch?app=desktop&v=XH5OW46yO8I>

<https://stackoverflow.com/questions/46646035/leaflet-modal-opens-behind-map>

<https://jsfiddle.net/askebos/Lh1y12uq/>

<https://jsfiddle.net/askebos/Lh1y12uq/>

Creación de un único *listener* para varios elementos

<https://parzibyte.me/blog/2019/02/01/escuchar-click-botón-botones-javascript/>

*LocalStorage*

<https://anexsoft.com/html-5-diferencias-y-ejemplos-entre-local-storage-y-session-storage>

<https://ed.team/blog/que-es-y-como-utilizar-localstorage-y-sessionstorage>

<https://qastack.mx/programming/9943220/how-to-delete-a-localstorage-item-when-the-browser-window-tab-is-closed>

## API

<https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/>

<https://www.youtube.com/watch?v=K-KUfg-Cuc8>

<https://www.youtube.com/watch?v=dVm8bDdGhWM>

<https://www.youtube.com/watch?v=w5akT9VS-xg>

[https://www.youtube.com/watch?v=qGEWYjVWVj8&list=PLIbWwxXce3VpvBT\\_O977da8XECEp-JTJt&index=3](https://www.youtube.com/watch?v=qGEWYjVWVj8&list=PLIbWwxXce3VpvBT_O977da8XECEp-JTJt&index=3)

<https://blog.logrocket.com/axios-or-fetch-api/>

<https://www.youtube.com/watch?v=9ZVooCNbdIY>

[https://www.youtube.com/watch?v=DhF\\_MNMcPtg](https://www.youtube.com/watch?v=DhF_MNMcPtg)

<https://www.youtube.com/watch?v=slrGgi6VHrl>

## AJAX

<https://code.tutsplus.com/es/tutorials/how-to-use-ajax-in-php-and-jquery--cms-32494>

<https://es.stackoverflow.com/questions/189000/como-exactamente-funciona-ajax-type-post-data-en-este-conecto>

<https://stackoverflow.com/questions/26191920/ajax-does-not-work-with-bootstrap-select>

<https://www.baulphp.com/crud-ajax-php-mysql-usando-bootstrap/>

<https://es.stackoverflow.com/questions/340090/ejecutar-con-jquery-y-ajax-un-modal-bootstrap-3-que-est%C3%A1-en-modal-php>

### Despliegue

<https://www.muycomputer.com/2014/08/05/montar-servidor-web/>

<https://desarrolloweb.com/faq/alojar-web-en-ordenador-casa>

<https://www.masmovil.es/ayuda-clientes/pasos-para-abrir-puertos-router-fibra-huawei-hg659>

<https://www.adslzone.net/foro/jazztel-soporte-tecnico.97/solucionado-ayuda-dmz-mi-router-jazztel-huawei-h.302983/>

<https://tutobasico.com/directivas-apache/>

### Estilo

<https://getbootstrap.com/>

<https://startbootstrap.com/>

<https://findbootstrapsnippets.com/bootstrap-templates/bootstrap-dual-design-registration-form.html>

<https://stackoverflow.com/questions/23699773/bootstrap-change-background-color-dropdown-menu>

## RECURSOS

### Imágenes

<https://pixabay.com/>

<https://www.gifanimados.org/cat-aviones-71.htm>

### Colores

<https://htmlcolorcodes.com/es/>

<https://www.significadodelcolor.com/>

<https://encycolorpedia.es/272aff>

<https://paletasdecolores.com/tag/amarillo-y-negro/>

## Anexos

Código

<https://github.com/eltitorobus/Mis-Viajes.git>

Enlace al sitio

<http://misviajes.ddns.net>