Documentación adicional

A continuación se presenta un documento donde se exponen los problemas encontrados a lo largo del desarrollo de la práctica grupal, así como las soluciones expuestas y una serie de tecnologías usadas durante la misma.

Despliegue:

Nada más comenzar el proyecto, el primer pensamiento fue acomodar el despliegue del mismo para que durante el desarrollo tuviéramos que toquetear lo mínimo posible en ese aspecto. Así pues, con el montaje que queríamos usar para tener Wordpress como front-end y laravel como back-end la opción en mente era que para acceder al proyecto se hiciera a través de la siguiente URL: **http:informatica.ieszaidinvergeles.org/9027/nombredelproyecto.**
Para esto, era necesario que la configuración inicial de uno, aplicara la del otro.

Solución:

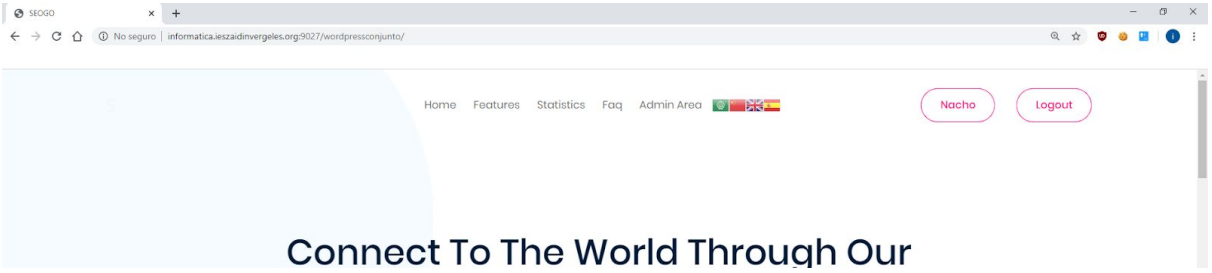
La solución inicial pasaba por instalar wordpress dentro de una de las carpetas públicas de laravel, para poder acceder a él a través de la configuración del propio laravel.

Rápido descartamos esta opción y la de instalar wordpress dentro del sistema de vistas de laravel puesto que no teníamos acceso a todas las herramientas disponibles (Blade, funciones de wordpress, funciones de laravel, clases vendor de laravel… etc).

Después de investigar por nuestra cuenta y tratar con Miguel Ángel para ver si nos podía ayudar en el tema, decidimos, a través de un artículo[*1] lanzar los archivos de configuración de ambos de manera simultánea.

De primeras, instalamos la carpeta de wordpress y en el raíz metimos nuestro laravel. A continuación ,mezclamos ambos archivos de configuración index.php y corregimos las rutas que iban en sendos archivos para que al cargar todo estuviera correcto, y por último modificamos la composición de carpetas de laravel para que todo estuviera en una carpeta private,y otra carpeta public.

(*1 El articulo era este: https://programarivm.com/instalar-laravel-5-dentro-de-wordpress/)



Geolocalización y Google Maps Api:

La propuesta hecha en clase partía de pedir la geolocalización del usuario y a partir de ella generar mapas hechos a través de la Google Api que mostrara la localización del usuario y los puntos de acceso más cercanos.

El primer problema sale cuando para poder pedir la geolocalización a través de javascript con el objeto Navigator.geolocation , es necesario tener un protocolo de comunicación seguro (HTTPS). La siguiente idea era crear un certificado autofirmado que nos generara el protocolo seguro pero google no reconoce los certificados autofirmados y tampoco podíamos acceder a la geolocalización.

Otra propuesta fue calcular la distancia entre dos puntos geográficos con un algoritmo físico en un método, pero rápidamente la descartamos por que nos parecía que no era la forma correcta de hacer las cosas.

Solución:

Realmente la solución fue prescindir de la geolocalización y trabajar con la api de google maps para generar los mapas y poner los puntos en el mapa que correspondían a los puntos de acceso.

Google Maps Api:

Cuando nos propusimos trabajar con la api de google maps tenía por seguro que no podía tratarse de algo complejo, que ingenuos. Desde la creación de la propia key que te obligan a utilizar a la propia configuración de la cuenta de google cloud te encuentras con una serie de impedimentos como pueden ser el idioma (algunas páginas las traduce, otras no, de forma nativa), la mala distribución de los menús o el simple hecho de que te pidan datos de facturación para un mapa en pruebas.

La experiencia ha sido terrible, pero con un resultado mucho más que satisfactorio, por lo que la conclusión es que las medidas impuestas por el camino son hechas adrede para evitar que cualquier persona pueda hacer uso de esas medidas.

Solución:

Después de mucho vagar, contactamos con el compañero Alejandro Romero, el cual se mostró más que dispuesto a darnos unas bases con las que trabajar. En este caso el trabajo en equipo entre proyectos nos ha permitido poder utilizar una herramienta extensa y útil, que sinceramente, de otra manera no hubiéramos podido.

Tras la información recibida, el resto fue relativamente sencillo y la personalización nos hizo que indagarámos a fondo en la nueva tecnología descubierta.

Charts.js y AJAX:

Uno de las librerías que más aspecto dan al back-end de nuestro proyecto es Charts.js, que desde un inicio nos propusimos trabajar mediante ajax para poder relacionar con algo ya usado en clase. La sorpresa fue la claridad con la que algo tan complejo funciona, es una librería bastante desarrollada pero que los únicos problemas que encuentra es el funcionamiento con Ajax.

Al principio tratábamos de poner varias gráficas en un mismo canvas que limpiabamos a través de ajax y rellenabamos con la nueva gráfica, pero pronto descubrimos que generaba un bug que superponía las vistas de gráficas una encima de otra.

Así pues el único problema y su consecuente solución fue limpiar de manera correcta el canvas y todas sus propiedades mediante jQuery y hacerle append de las gráficas de manera completa.

Textarea Editor:

La propuesta basaba en crear un textarea que tuviera funcionalidades extra y no el típico input feo y básico que proporciona HTML. Así pues investigando encontramos https://ckeditor.com/ , un vendor de laravel que utiliza una clase para configurar un objeto textarea complejo. La puesta en marcha es tan sencillo como configurar su archivo inicial e instalarlo en vendors, usar la clase en el input y ya estaría. Sencillo, rápido y útil.

Multilenguaje en la página:

Para realizar el multilenguaje en la página usamos un plugin de wordpress, puesto que todo el front-end realmente sale del motor de wordpress. Hemos usado GTranslate que recoge toda la información de la vista y hace una traducción a través del traductor de Google. También nos da la opción de anexar un diccionario de palabras para poder hacer mas especifico su uso.