

**DPTO. INFORMÁTICA – I.E.S. RIBERA  
DE CASTILLA MÓDULO PROYECTO  
C.F.G.S.**

**Desarrollo de Aplicaciones Web**

**Plainer**

Autor: Mario Gañán Fuentes

Fecha: 28/05/2018

Tutor: Jesús Sánchez Curto



# Índice

Introducción.....	4
Origen y Contextualización del Proyecto.....	6
• Material design .....	7
Objetivo General del Proyecto.....	8
• Aplicaciones web.....	8
Objetivos Específicos .....	14
Etapas del desarrollo de la aplicación .....	15
• Herramientas Hardware .....	15
• Herramientas Software.....	15
• Diseño de la interfaz.....	16
• Vista anual.....	18
• Vista mensual.....	18
• Vista de eventos .....	19
• Diseño del back-end.....	19
• Proceso de desarrollo.....	20
Ejecución del proyecto .....	23
Bibliografía .....	26

# Introducción

En un centro educativo o en un centro de trabajo, de forma habitual hay acontecimientos que influyen a un grupo amplio de personas. Existen muchas formas de comunicar éste tipo de eventos al alumnado/empleados etc. tales como el envío de correos, comunicar el acto a través de carteles o comunicación verbal.

No siempre las formas habituales de comunicación son las más directas o más efectivas para transmitir el mensaje, y a menudo se echa de menos una manera de centralizar éste tipo de sucesos.

Existen numerosas herramientas informáticas que sirven el propósito de hacer llegar el mensaje de forma directa y, a ser posible, a toda la gente implicada en ése mensaje, sin embargo no siempre son las formas más eficientes de realizarlo.

Muchas herramientas se antojan complejas, poco visuales o repletas de características y funciones que en el uso habitual del día a día no se les saca un provecho real.

Por ello mismo, en numerosas ocasiones se hace necesaria la existencia de una herramienta capaz de cubrir la necesidad de la comunicación y el almacenamiento de eventos por parte de un centro/organización hacia sus empleados, alumnos, etc.

Debido a eso, surge Plainer, una utilidad capaz de reunir distintos eventos diferenciados, con parte de administración y parte de consulta.

Sin embargo, dado que Plainer pretende otorgar una experiencia “seamless”, es decir, lo más fluida posible y sin “costuras” en su ejecución, la división administrador/cliente se trata desde una perspectiva que no se antoja tan diferenciada la una de la otra.

Ésta división “seamless” permite:

- Al administrador de la aplicación tener un control total sobre los eventos que se suben a Plainer, así como de su edición o eliminación completa, a la vez que conserva el mismo manejo, diseño y funcionalidad.
- Al cliente tener una accesibilidad rápida de todos los cambios o adiciones del administrador.

El objetivo de Plainer es ofrecer un diseño simple, atractivo e intuitivo a los usuarios, mientras ejerce la función para la que fue diseñada.

## Origen y Contextualización del Proyecto

Cuando se piensa en una aplicación para almacenar eventos generalmente se piensa en herramientas del estilo de Google Calendar o Microsoft Outlook, que permite añadir eventos, recordatorios etc. con una gran variedad de opciones extra.

Sin embargo, éste tipo de herramientas son unipersonales, es decir, solamente se centran en el usuario que las está utilizando para apuntar recordatorios, cumpleaños, citas, etc. y no tanto como una plataforma donde consultar eventos de cierta relevancia para un grupo de personas.

Al igual que nos podemos informar de los festivos para los estudiantes de E.S.O., Bachillerato, Grados Superiores y demás a través de la web de educacyl, no es una manera versátil de que el encargado de dicha plataforma pueda editar, cambiar o añadir nuevas fechas (ya que son datos estáticos).

De ésta manera, y con el pensamiento en mente de grandes centros para comunicar, anotar y/o almacenar distintos eventos comunes a un grupo grande de personas, surge Plainer.

Plainer permite que un usuario administrador pueda subir diferentes eventos diferenciables entre sí para que, las personas pertenecientes a ése grupo objetivo, puedan tener una plataforma donde consultar qué está pasando, que tareas deben seguir, etc.

Ésta cualidad organizativa se junta con un diseño sencillo pero no escueto, atractivo pero no sobrecargado e intuitivo pero no escaso en funcionalidad.

La intención de Plainer es que la propia interfaz sea indicativa de la funcionalidad, es decir, que con solo mirar la interfaz ya se sepa que información está mostrando exactamente.

En definitiva y a modo de contextualización, la intención de Plainer es salirse del marco de las herramientas habituales que cada vez reúnen más características poco accesibles y diseños toscos, y crear una aplicación que facilite las cosas al usuario tanto en funcionalidad como en manejo.

## • Material design

Material design es la base de Plainer. Se trata de un lenguaje de diseño introducida por Google a partir de la versión 5.0 Lollipop de Android, y de la interfaz de Google+ en 2014.

Se trata de dar significado a cada componente que puedas tocar. Los bordes, superficies y sombras son representaciones físicas que reaccionan con la interacción del usuario.



Es un lenguaje de diseño basado en el papel y la tinta. La predominación más clara de Material design son los colores pastel, las sombras, las superficies planas y las animaciones.

Material Design utiliza el papel como una metáfora con reglas para el diseño, localización, movimiento, transformación, transiciones y animaciones de los objetos en la pantalla.

Existen reglas para mostrar fechas, elementos de navegación y avisos de error, de la misma manera que la manera de cargar imágenes y hacer scroll o swipe.

En definitiva, Material Design supuso una revolución en el diseño de interfaces, ya que introdujo otra manera de pensar y se alejó de diseños de iconos realistas, transparencias y animaciones de bloque en favor de diseños planos, colores pastel y animaciones que unen todo movimiento interactivo.

## Objetivo General del Proyecto

El objetivo del proyecto era crear una aplicación capaz de reunir los requisitos relacionados con la unificación de eventos de un centro en una misma plataforma, todo ello con un diseño sencillo y atractivo.

Precisamente y a propósito del diseño, la intención también era crear una aplicación la cual integrara la funcionalidad con el diseño.

Esto significa que, si se visualiza la herramienta por primera vez se sepa de forma inmediata en que consiste y que se está viendo exactamente.

De ésta misma manera, si el usuario no identifica lo que está viendo, mediante la navegación a través de la página podrá saber qué está pasando.

- **Aplicaciones web.**

Debido a las últimas tendencias a la unificación de las aplicaciones en aplicaciones web, se ha escogido éste modelo para Plainer.

Ésta última tendencia facilita el desarrollo de aplicaciones dado que no hay que utilizar distintos lenguajes dependiendo de la plataforma, sino que se utiliza una misma aplicación adaptativa que se amolde a cada dispositivo.

Con ésta mentalidad asentada, se ha optado por el uso de las siguientes tecnologías:



- **HTML5:** Se trata del estándar de lenguaje de marcado de hipertexto para la elaboración de páginas web, la base de toda web.



Destacar la última versión 5 debido al nuevo uso de atributos data-\* para el manejo y almacenamiento de datos en el DOM.



- **CSS3:** El lenguaje estándar de estilos web. Utilizado en el desarrollo de Plainer a través de Sass.

Destacar la implementación en su versión 3 del sistema de flexbox.



- **JavaScript:** El lenguaje de programación de lado de cliente por excelencia. Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos y es débilmente tipado y dinámico.

Permite la interacción con los nodos HTML (DOM) de la página donde se ejecuta y destaca por su versatilidad.



- **AJAX:** La tecnología de peticiones asíncronas más utilizada. En Plainer sirve un propósito clave, debido a la necesidad de comunicación entre JavaScript (jQuery) y PHP.



- **Sass:** Una de las extensiones de CSS3 más utilizadas, gracias a su estabilidad, madurez y amplia gama de posibilidades.

Se trata de un lenguaje similar a CSS3 con variables, funciones y más características, que se compila a CSS3 mediante Ruby para su interpretación por todos los navegadores.

Como curiosidad, cabe destacar que, si utilizamos Sass para el desarrollo de nuestras páginas web, en las herramientas de desarrollador de nuestro navegador veremos reflejadas las líneas que hacen referencia a nuestro archivo Sass (.scss) en lugar de al .css compilado, a pesar de que el archivo .scss no es realmente interpretado.

Todo esto sucede gracias al archivo .map generado por el compilador:

```
Filter                                     :hov .cls +
element.style {
  transform: scale(1);
}
.month {                                  style.scss:282
  width: 317px;
  height: 317px;
  padding: 10px;
  overflow: visible;
  transition: .2s;
  cursor: pointer;
  border-radius: 3px;
  margin: 0 auto;
}
div {                                    style.scss:7
  text-align: center;
}
```



- **PHP:** Uno de los lenguajes de programación del lado del servidor más utilizados del mundo.

Permite la creación de contenido dinámico en una página web.

PHP utilizado junto con Ajax, permite la realización de consultas a bases de datos sin que hagan falta recargas de página.



- **MySQL:** El sistema gestor de bases de datos relacionales más utilizado del mundo.

Está considerada como la base de datos de código abierto más popular existente.

Sigue una estructura de tablas, claves primarias y claves externas entre tablas que permite una relación entre ellas muy accesible.



- **jQuery:** Uno de los grandes frameworks de JavaScript. Su lema es “Write less, do more”, lo que nos sugiere el aspecto de sencillez que destaca.

Brilla por ser una librería rápida, sencilla y repleta de posibilidades. El manejo de eventos, elementos del DOM y llamadas Ajax con jQuery es mucho más rápido y sencillo que mediante JavaScript plano.

En numerosas ocasiones se postula como un lenguaje de programación totalmente independiente de JavaScript, debido a su característica estructura de selectores y llamadas encadenadas de funciones. Esto puede ser una ventaja, pero también puede ser un inconveniente ya que, a diferencia de otras librerías de JavaScript, requiere una curva de aprendizaje.



**Moment.js**

- **Moment.js:** Un framework de JavaScript que introduce una nueva manera de formatear y trabajar con fechas. Permite un grado amplio de opciones, tanto genéricas como locales (de región). La sencillez de manejo es uno de sus pilares principales.



- **Materialize:** Se trata de un framework visual basado en el más que conocido Material Design de Google.

Es tanto un framework de css como un framework de JavaScript debido a su sistema de elementos instanciables.

Materialize basa sus elementos (tooltips, ventanas modales, efectos, formularios...) a objetos que se instancian tanto desde **JavaScript** como desde **jQuery**, permitiendo el uso de objetos JavaScript para configurar cada elemento.

Ya que se basa en Material Design, destaca por su interacción con el usuario, su diseño plano y sencillo, sus animaciones, etc.

En éste caso particular, se utiliza una versión de Materialize ligeramente manualmente modificada para Plainer.



- **noUiSlider:** Se trata de un framework que introduce un slider selector de rangos en JavaScript.

La versión de noUiSlider que se está utilizando en Plainer, ha sido modificada por **Materialize** para integrarlo con la filosofía Material del framework visual.



- **Normalize.css:** Es un framework de CSS encargado de unificar todos los estilos a través de los diferentes navegadores para forzar un renderizado uniforme y consistente.  
Atiende a los últimos estándares de estilos CSS.



- **JSON:** El formato de intercambio de datos por excelencia. Se trata de un tipo de archivo utilizado para traspasar datos de un sistema a otro.

Su papel en Plainer es muy importante, debido al uso de lenguajes distintos tanto para el lado del cliente como al lado del servidor, y la necesidad de transmitir datos de uno al otro.

El formato de archivo json también es clave en las peticiones asíncronas de AJAX.

## Objetivos Específicos

El objetivo principal con Plainer siempre fue la elaboración de una aplicación con una usabilidad clara, pero con un diseño simple y accesible.

El acercamiento de una plataforma directa, sencilla y con una accesibilidad tan marcada a un gran grupo de personas fue siempre el requisito más marcado.

Por esa misma razón, el desarrollo de Plainer siempre ha ido inclinado hacia ése pensamiento, construyendo la aplicación y todas sus funciones con la misma filosofía.

El desarrollo de Plainer ha tenido la particularidad de haber sido desarrollado plantando la base antes mencionada, y empezar a construir desde ahí.

En su creación, se priorizó la fidelidad a la filosofía de Plainer por encima de su funcionalidad, y ello permitió conseguir el resultado final.

## Etapas del desarrollo de la aplicación

La Elección de las herramientas a la hora de crear la aplicación es clave, por ello se han escogido las siguientes herramientas:

- **Herramientas Hardware**

El hardware utilizado en el desarrollo de Plainer ha sido el siguiente:

- PC de sobremesa personal y portátil personal.
- PC portátil proporcionado por empresa de FCT.

- **Herramientas Software**

El sistema operativo utilizado en el desarrollo de Plainer ha sido en todo momento Windows 10. En cuanto al software específico empleado ha sido el siguiente:



- **Visual Studio Code:** El entorno de desarrollo principal. Destacable por ser de Microsoft, al igual que por todas las opciones de configuración (archivos de configuración del programa en formato JSON) y el amplio catálogo de extensiones. Para poder utilizar Sass es necesario contar con un compilador que traduzca nuestras hojas de estilo en formato .scss a .css. Para ello, lo más habitual es utilizar Ruby, sin embargo, en el desarrollo de Plainer se ha utilizado Live Sass Compiler, una extensión de Visual Studio Code.



- **Xampp:** Una herramienta que engloba diferentes aplicaciones, tales como Apache, MySQL, Tomcat y más. Ideal para trabajar en local, se ha utilizado en el desarrollo de Plainer para las pruebas y despliegue en local de la aplicación.



- **MySQL Workbench:** Un gestor de bases de datos como alternativa de phpMyAdmin. Debida a su naturaleza de programa instalable, al contrario de phpMyAdmin, lo hace más versátil y rápido para hacer una administración de la base de datos de Plainer.



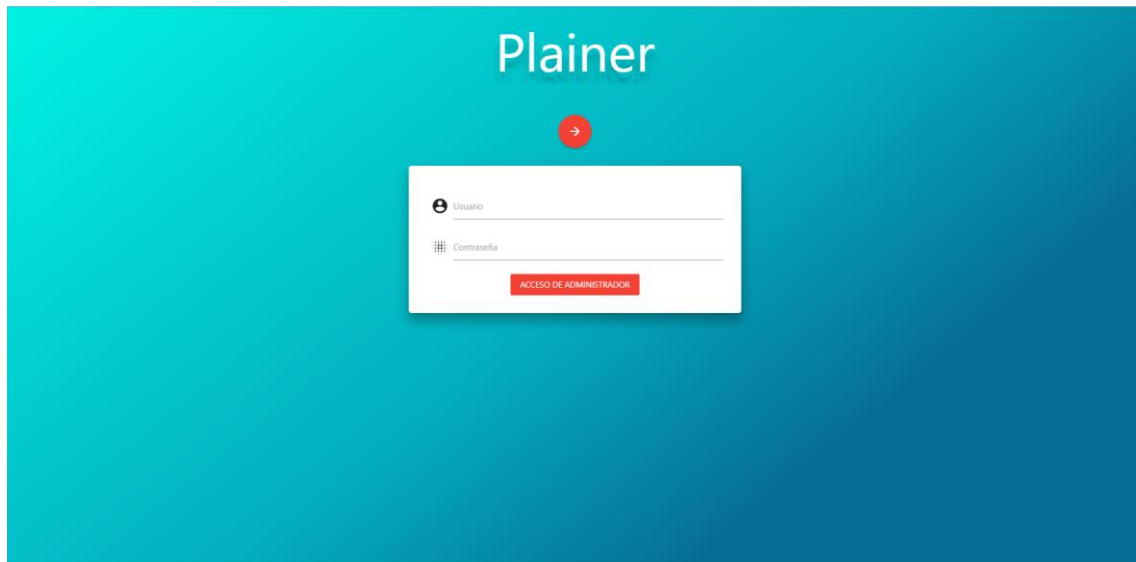
- **Google Chrome:** El navegador más utilizado en el mundo. Google Chrome se ha utilizado en el desarrollo de Plainer tanto para visualizar la aplicación como para su desarrollo. Destacar su menú de desarrollo, que permite hacer debug de una forma visual, directa y sencilla, así como introducir código temporal a través de la consola integrada.

- **Diseño de la interfaz**

El diseño de la interfaz de Plainer es el aspecto más cuidado de la aplicación, dado que es la base desde la cual se cimentó la idea del proyecto.



De ésta manera, al abrir Plainer por primera vez obtenemos una pantalla que sirve a modo de declaración de intenciones de cómo va a ser la filosofía de diseño de la aplicación. Es la siguiente:



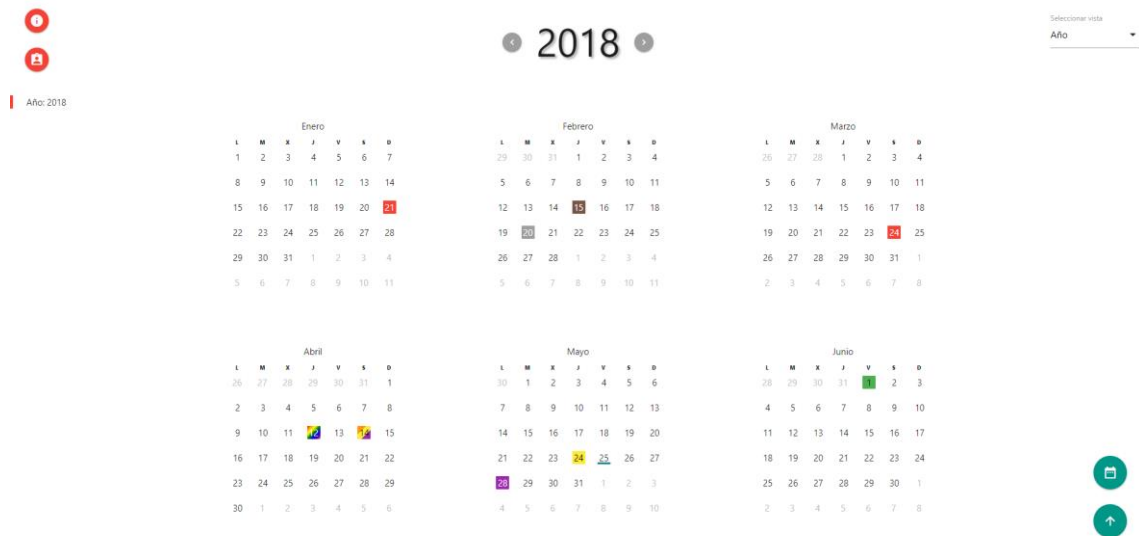
Nada más abrir Plainer por primera vez nos encontramos con un fondo de pantalla con un degradado y mucho espacio libre en pantalla, lo que intenta remarcar el diseño plano y la simplicidad.

De mismo modo, tenemos el nombre de “Plainer” en una fuente muy similar a la conocida Helvética, que es sans serif para eliminar todo rastro de decoración. También tiene un ligero sombreado para marcar la actitud “Material”.

Por ese mismo motivo, los cuadros de texto, botones y demás se amoldan a éste mismo conjunto de reglas.

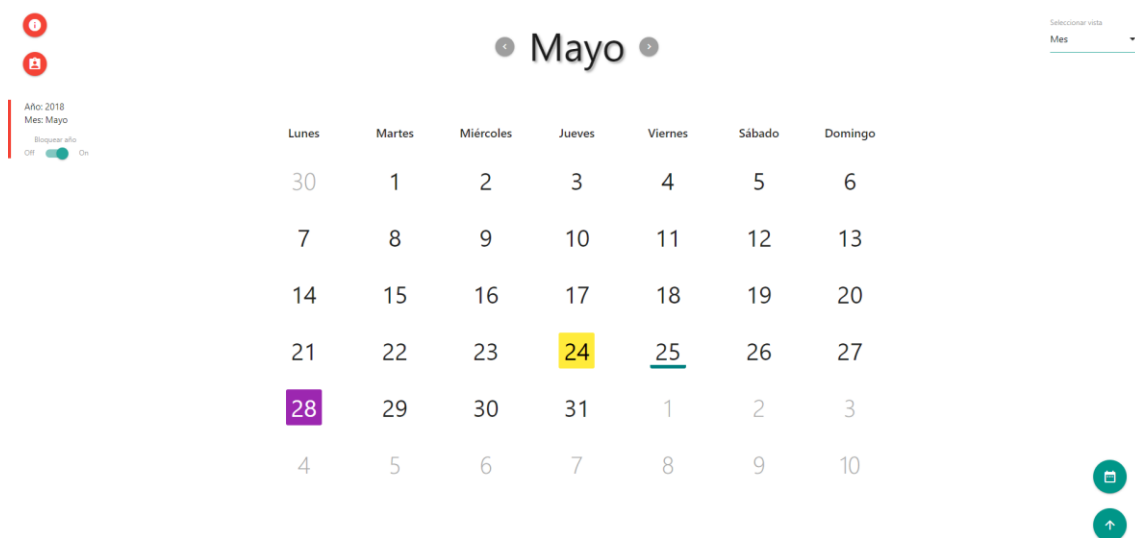
En cuanto al contenido de la aplicación como tal, observamos 3 tipos de vistas:

- Vista anual



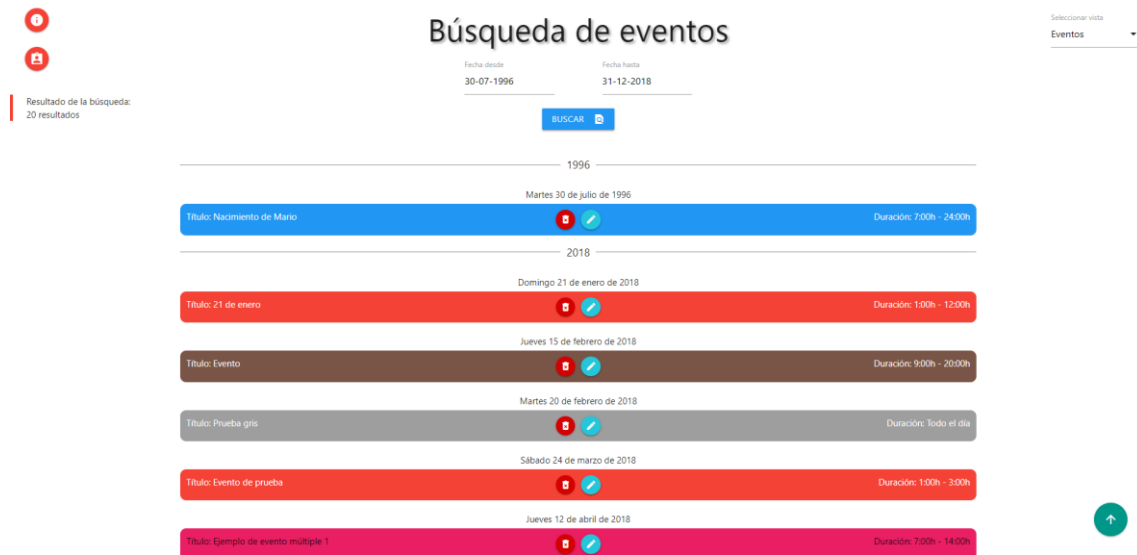
La vista anual (la vista por defecto) consta de una cuadrícula de 3x4 con los meses. Podemos seleccionar tanto los meses, como las semanas y los días por separado.

- Vista mensual



La vista mensual cuenta con una visión más ampliada de cada mes, pudiendo ver con más detalle los eventos asignados a cada día.

- Vista de eventos



En la vista de eventos podemos hacer una búsqueda a partir de dos fechas, y obtener todos los eventos comprendidos.

Los eventos se agruparán tanto por año como por día.

La transición entre cualquiera de las vistas es totalmente fluida, sin recargas de navegador.

- Diseño del back-end

La predominancia de jQuery es absoluta, estando todo programado con éste framework. La comunicación al back-end se realiza a través de peticiones AJAX.

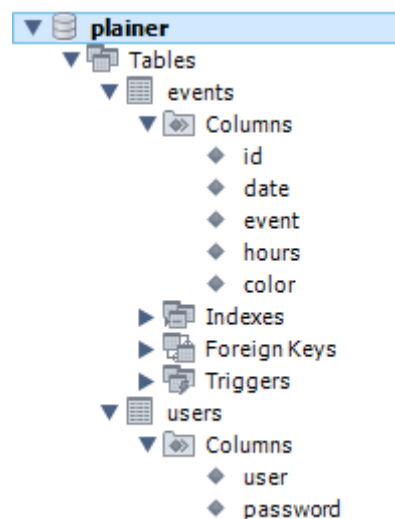
```

640 | $(function () {
641 |     $.get("./src/funcs/funcs.php?firstEntryRequest", function () {}).done(function (data) {
642 |         firstEntry = data;
643 |         //Inicializaciones de los componentes javascript de Materialize y el slider de noUiSlider materializado
644 |         M.AutoInit();
645 |         $("#fromDateInput").datepicker({
646 |             format: "dd-mm-yyyy",
647 |             firstDay: 1,
648 |             showDaysInNextAndPreviousMonths: true
649 |         });
650 |         $("#toDateInput").datepicker({
651 |             format: "dd-mm-yyyy",
652 |             firstDay: 1,
653 |             showDaysInNextAndPreviousMonths: true
654 |         });
655 |         var slider = $("#test-slider")[0];
656 |         noUiSlider.create(slider, {
657 |             start: [1, 24],
658 |             connect: true,
659 |             step: 1,
660 |             orientation: 'horizontal', // 'horizontal' o 'vertical'
661 |             range: {
662 |                 'min': 1,
663 |                 'max': 24
664 |             },
665 |             format: wNumb({
666 |                 decimals: 2
667 |             })
668 |         });
669 |         var slider2 = $("#editTest-slider")[0];

```

Debido al carácter asíncrono de AJAX, el código que precede de una petición AJAX cuelga de cada función “done”.

En cuanto a la base de datos, se compone de dos tablas: Una tabla con los usuarios administradores (con la contraseña encriptada) y otra tabla para el almacenaje de eventos. La estructura es la siguiente:



- Proceso de desarrollo

El desarrollo de Plainer comenzó con una primera reflexión sobre las tecnologías a utilizar.

Al final se optó por utilizar PHP como lenguaje base y del lado del servidor, y JavaScript en el lado del cliente, con jQuery predominando como librería.

Una vez escogidos los lenguajes, el entorno de desarrollo a utilizar no supuso una reflexión extra, dado que Visual Studio Code había sido el software con el que más había trabajado.

El inicio del desarrollo comenzó con la creación de la vista anual en su totalidad. El tratamiento de los días y las semanas supuso una dificultad añadida, ya que había que tener en cuenta casos particulares como años bisiestos, etc.

Una vez finalizada la vista anual, se comenzó el desarrollo de la visualización de las semanas. Por la misma razón anterior, el tratamiento de las semanas implicó una especial atención en los casos particulares, tales como los días correspondientes a la primera semana de enero, que contenían días de diciembre del año anterior.

La vista mensual fue la siguiente en ser desarrollada, y después fue el turno de introducir la identificación de eventos. El resultado final presenta una división de los colores de fondo (siempre correspondientes al color escogido para el evento en cuestión) de los días siendo:

- Un color único para días con solo un evento.
- Un degradado de dos colores para días con dos eventos.
- Un degradado de tres colores para días con tres eventos.
- Un degradado multicolor para días con más de tres meses.

De ésta manera, con un primer vistazo a los días se puede identificar los tipos de eventos en ése día.

La última fase del desarrollo implicó la creación de la vista de eventos, en la cual podemos buscar entre un rango de fechas determinadas todos los eventos comprendidos.

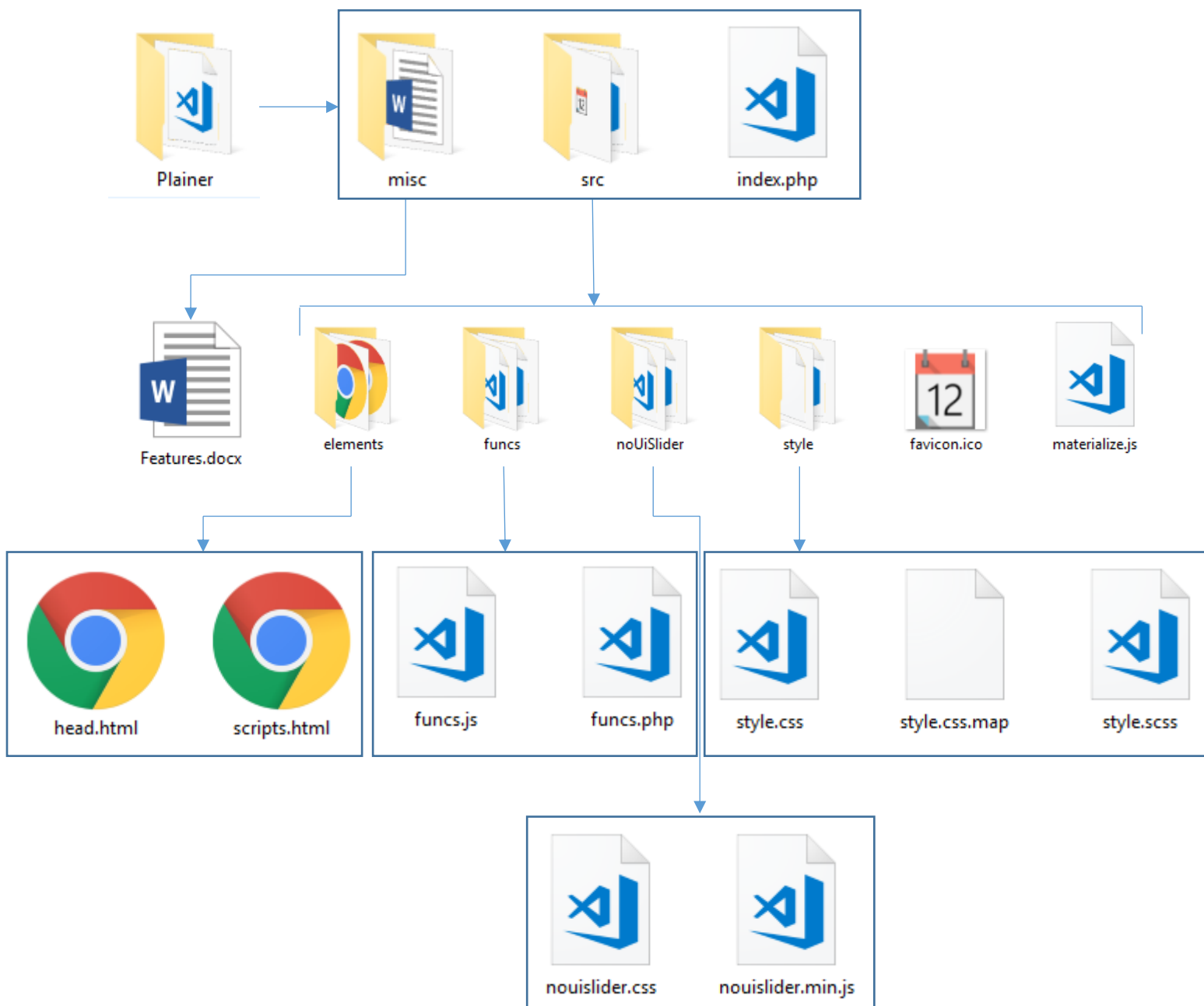
La corrección de bugs, pulido de características y atención en los pequeños detalles se fue realizando según avanzaba el desarrollo, no hubo una fase final de corrección como tal.

## Ejecución del proyecto

Si queremos desplegar Plainer en un entorno local, tendremos que seguir las siguientes instrucciones:

Necesitaremos un servidor de Apache y de MySQL. Se recomienda utilizar **XAMPP**, aunque herramientas como **WAMP** también son más que válidas.

La estructura de carpetas de Plainer es la siguiente:

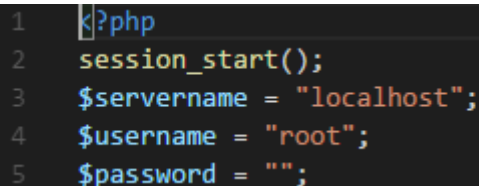


Únicamente tendremos que copiar la carpeta “Plainer” dentro de la ubicación de nuestro directorio de páginas (htdocs en el caso de XAMPP, www en el caso de WAMP, etc.).

Debemos hacer un ajuste en nuestro servidor apache para eliminar los avisos “notice” de PHP, ya que pueden interferir en las recepciones de las peticiones AJAX. Por defecto en XAMPP vienen ya deshabilitados, en caso contrario, utilizando htaccess, simplemente añadiremos las siguientes líneas a nuestro archivo .htaccess:

```
php_flag display_startup_errors off
php_flag display_errors off
php_flag html_errors off
php_flag log_errors on
php_flag ignore_repeated_errors off
php_flag ignore_repeated_source off
php_flag report_memleaks on
php_flag track_errors on
php_value docref_root 0
php_value docref_ext 0
php_value error_log /home/path/public_html/domain/PHP_errors.log
php_value error_reporting -1
php_value log_errors_max_len 0
```

La conexión a base de datos está configurada para que el servidor, usuario y contraseña sean “localhost”, “root” y “” en ése orden. En caso contrario, deberemos cambiar las siguientes tres variables en el archivo func.php:



```
1  <?php
2  session_start();
3  $servername = "localhost";
4  $username = "root";
5  $password = "";
```

Con el servidor Apache y MySQL desplegados y todas las configuraciones anteriores tenidas en cuenta, con nuestro navegador (Google Chrome y Firefox recomendados) entraremos en la dirección de nuestro servidor local, y después dentro de la carpeta de Plainer. Quedará algo parecido como:

<http://localhost/Plainer>



La base de datos y tablas de Plainer se crean de forma automática en la ejecución de la aplicación, con lo cual no hay que hacer ninguna configuración extra en éste aspecto.

Nada más abrir Plainer tendremos la opción de entrar en modo invitado o iniciar sesión como administrador. Para disfrutar de todas las características de Plainer deberemos entrar como administrador. El usuario y contraseña de administrador son, en ése orden, “admin” y “admin”.

Ya estamos listos para probar Plainer, ¡que lo disfrutes!

## Bibliografía

API de Materialize: <https://materializecss.com/>

API de moment.js: <https://momentjs.com/docs/>

API de noUiSlider: <https://refreshless.com/nouislider/>

API de jQuery: <http://api.jquery.com/>

W3Schools: <https://www.w3schools.com/>

StackOverflow: <https://stackoverflow.com/>