UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Weblectric 2018

Presentado por Francisco Saiz Güemes
en Universidad de Burgos — 31 de diciembre de 2018
Tutor: Álvar Arnaiz Gonzales - Jesús Maudes Raedo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Francisco Saiz Güemes, con DNI 71567002H, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 31 de diciembre de 2018

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Unificando y buscando complementar nuestros conocimientos y oportunidades con las necesidades manifestadas por el cliente, podríamos hablar de los siguientes objetivos:

2.1. Objetivos generales

Como objetivos técnicos propiamente del proyecto, podríamos destacar:

- Conocer el funcionamiento general del algoritmo NSGA-II [5], un algoritmo de optimización.
- Dar soporte a través de una aplicación web a los resultados obtenidos mediante la ejecución del algoritmo de optimización.
  - Una administración que permita crear, editar y eliminar cualquiera de las entidades.
  - Una pantalla de simulación del algoritmo.
- Aprender a emular en nuestra máquina, un servidor Apache [**?**] que nos permita desarrollar en nuestro entorno local.
- Familiarizarme con CakePHP. Un framework de desarrollo en PHP con arquitectura MVC[1].
- Conectar la aplicación con una base de datos MySQL que nos permita manejar los datos proporcionados por el cliente.

[1] MVC: Modelo Vista Controlador

- Alojar nuestros contenidos en un servidor al que accederemos mediante un cliente FTP.

2.2. Objetivos personales

Como objetivos quizá más personales que también me gustaría incluir para lograr alcanzar con el desarrollo de este proyecto, podríamos incluir:

- Conocer mis tiempos y calcular costes de desarrollo con la idea de optimizar el rendimiento y planificación de mi tiempo.
- Aprender a adelantarme a las necesidades del cliente, aportándole una solución concreta, real y efectiva a sus necesidades.
- Viendo que las puertas al mundo profesional se me abren por este camino, me parece interesante complementar los conocimientos obtenidos durante estos años en la universidad con otros lenguajes de programación complementarios a PHP, los cuales utilizaré tambien en el desarrollo de la aplicación.

Como objetivo final del proyecto y visión de futuro para los próximos proyectos, me gustaría aprender a detectar las necesidades del cliente con la finalidad de generarle el máximo valor de negocio posible cubriendo todas sus necesidades.

Conceptos teóricos

Aunque el proyecto está enfocado a la administración a través de una aplicación web de unos datos que nos facilita el cliente, la base del proyecto se cimienta sobre un algoritmo de optimización capaz de proporcionarnos soluciones acerca de la mejor distribución de plantas energéticas en México.

Antes de explicar cómo se ha enfocado el planteamiento de la aplicación, es necesario hablar un poco sobre la optimización.

3.1. Optimización

¿Qué es la optimización?

Podríamos referirnos a la optimización como un proceso de selección del mejor elemento (con respecto a algún criterio) dentro de un conjunto de elementos disponibles [13].

Si hablamos de problemas de optimización, el proceso consiste en minimizar o maximizar una función real escogiendo de manera sistemática valores de entrada (tomados de un conjunto permitido) y computando el valor de la función [13].

Optimización clásica vs. Metaheurística

La optimización clásica garantiza el óptimo numérico y permite un numero elevado de restricciones. Algún método clásico es:

- Programación lineal.
- Programación cuadrática.

- Programación no lineal.
- Programación dinámica.
- Teoría de grafos u optimización en redes.

La optimización metaherística[2] trata de imitar fenómenos sencillos ocurrentes en la naturaleza, mecanismos específicos para evitar óptimos locales sino tener una mirada mas "global". No garantizan la obtención de un óptimo así como tampoco permiten un gran número de restricciones.

Son soluciones aplicados generalmente a problemas combinatorios que exploran un gran número de soluciones en un tiempo muy corto.

Algún método metaheurístico es:

- Algoritmos evolutivos (genéticos).
- Recocido o simulado.
- Sistemas multiagente.

Componentes [2]

Función objetivo

Medida cuantitativa de un sistema que se desea maximizar o minimizar

Variables

Podemos hablar de ellas como las decisiones que afectan al valor de la función objetivo. Pueden ser dependientes o independientes.

Restricciones

Conjunto de relaciones que las variables están obligadas a satisfacer

Ejemplos básicos

Por ejemplo: Un problema de optimización puede ser representado de la siguiente forma:

Dada: una función $f : A \rightarrow R$

[2]Inteligencia Artificial

Buscar: un elemento x_0 en A tal que $f(x_0) \leq f(x)$ para todo x en A si hablamos de minimización o tal que $f(x_0) \geq f(x)$ para todo x en A si se trata de un problema de maximización.

En nuestro ejemplo, A sería un subconjunto del *espacio euclídeo*. Si trabajamos con una dimensión, denominaríamos *espacio euclídeo* a una recta que va desde $-\infty$ hasta $+\infty$. En cambio, si trabajamos en dos dimensiones, el *espacio euclídeo* sería un plano con infinitas rectas y puntos. La función f sería la función objetivo y se hace referencia a ella como función de costo en los problemas de minimización y función de utilidad en los problemas de maximización.

El objetivo de todos los problemas de optimización es encontrar el valor que deben tomar las variables para hacer óptima la función objetivo satisfaciendo el conjunto de restricciones.

Sin embargo, cabe destacar que dentro de los problemas de optimización tenemos dos tipos:

- Mono-objetivo.
- Multi-objetivo.

La diferencia entre ambos es que en los primeros, una solución se considera mejor que otra si con ella se obtiene una solución objetivo de menor valor si estamos minimizando, o una solución de mayor valor si estamos maximizando.

Centrándonos de nuevo en nuestro análisis, la solución proporcionada para resolver el problema viene de parte de un algoritmo multi-objetivo llamado Nondominated Sorting Genetic Algorithm II (NSGA-II).

3.2. NSGA-II

Implementando esta solución, la meta a la que se quiere llegar es a encontrar un vector de variables $x = (x1, x2, \ldots xj)$ que cumpla con todas las restricciones y condiciones, donde las funciones objetivos resultantes sean optimizadas [7].

Se denomina espacio de solución al conjunto de todas las combinaciones posibles. Es denotado mediante: $fn(x) = z = (z1, z2, \ldots zM)$.

Sin embargo, en los problemas multiobjetivo, este criterio no es correcto pues entran en juego simultáneamente funciones de minimizar y de maximizar.

Para llegar a una solución, en los problemas multiobjetivo, se introduce un nuevo operador, dominancia. Que define: una solución $x(1)$ domina otra solución $x(2)$ si se cumplen las siguientes condiciones [5]:

- La solución $x(1)$ no siempre es de menor calidad que $x(2)$ en todos los objetivos.
- Al menos en uno de los objetivos, la solución $x(1)$ es estrictamente mejor que $x(2)$.

Utilizando estas reglas de manera iterativa sobre un conjunto de soluciones de un probloema de optimización multiobjetivo, se puede llegar a establecer cuales son las alternativas dominantes. Las conocemos como Conjunto No Dominado. El resto de soluciones pasan a formar parte del Conjunto de Soluciones Dominadas.

Logrando establecer este conjunto de Soluciones Dominantes en un espacio objetivo, podemos hablar de Frente óptimo de Pareto [9]. (Figura 3.1)

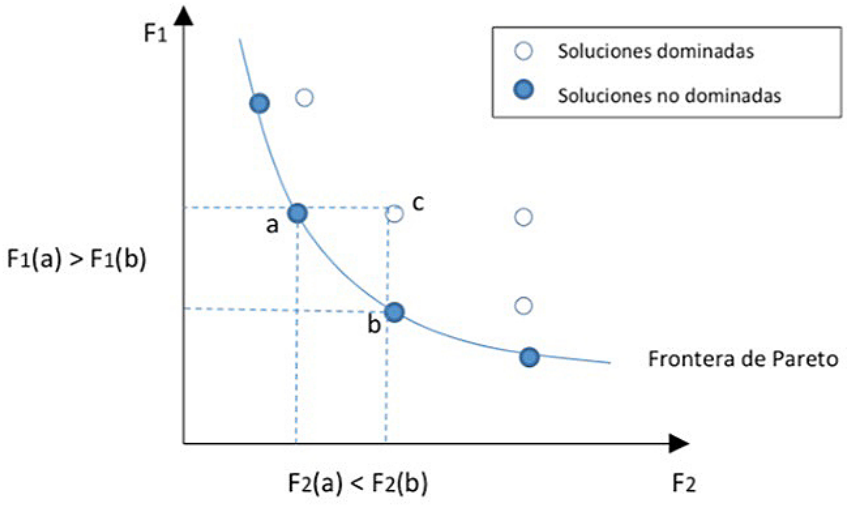

Figura 3.1: Frente de pareto en un problema de minimización. Ilustración extraida de [9]

Pseudocódigo NSGA-II [5]

1. Generar una población P de tamaño N.
2. Identificar los frentes de dominancia y evaluar las distancias de apilamiento en cada frente.
3. Usando selección (<c)[3], cruzamiento y mutación se genera una población descendiente del mismo tamaño de P.
4. Reunir Padres e hijos en un conjunto de tamaño 2N y clasificar los frentes de dominancia.
5. Determinar el conjunto descendiente final seleccionando los frentes de mejor rango. Si se supera el límite de población N, eliminar las soluciones con menor distancia de apilamiento en el último frente seleccionado.
6. Sí se cumple el criterio de convergencia, Fin del proceso. De lo contrario retornar al paso 3.

[3]Selección por torneo según operador de apilamiento. La selección retorna la solución ganadora i basándose en dos criterios fundamentales.

- Si tiene mejor rango: ri<rj.
- Si tienen el mismo rango pero i tiene mejor distancia de apilamiento: di>d.

Técnicas y herramientas

Para el desarrollo del proyecto y para lograr alcanzar tanto los objetivos académicos como personales propuestos anteriormente, he considerado de utilidad las siguientes herramientas.

4.1. Gestión del proyecto y control de versiones

Gestión del proyecto

A la hora de gestionar un proyecto, podemos clasificar el método usado para su gestión en dos grandes grupos: Metodología tradicional y metodología ágil.

Analizando las diferencias entre ellas podemos afirmar [1]:

- En la metodología tradicional, esta presente la figura de un Project Manager que basa sus conocimientos en el PMBoK[4].
- La metodología tradicional se centra en un enfoque proactivo y predictivo. Busca desde los orígenes del proyecto definir todo lo definible antes de empezar, anticiparse a cualquier cambio, buscar proyección, es decir, dar un alcance lo más completo posible y ajustar el coste al máximo.

[4]Libro donde se recogen técnicas y acciones a llevar a cabo dentro de un proyecto para obtener un resultado próspero.

- La metodología ágil surge como necesidad del cliente a proyectos no muy grandes. No existe una necesidad por parte del cliente de una planificación inicial exhaustiva, sino que necesita un producto en un espacio corto de tiempo y no hay tiempo para grandes planificaciones.

- Es probable que el producto demandado por el cliente en un principio, sea diferente del demandado a final del proyecto. Esto se debe al continuo cambio sobretodo en el mundo de las TIC. El cliente sabe que necesita pero desconoce como se va a concretar a X días vista

Estando delante de un proyecto no muy grande y aprovechando los conocimientos recibidos en el grado, hemos decidido utilizar una metodología ágil para gestionar nuestro proyecto.

SCRUM

Scrum[5] es uno de los métodos ágiles más extendidos. Se trata de un método incremental e iterativo que divide el desarrollo del producto en ciclos llamados *sprints*.

Al inicio de cada *sprint*, se realiza una reunión entre todos los integrantes del proyecto donde se definen los objetivos y requisitos de cada ciclo. Cada una de esas tareas se denomina *issues*.

Este tipo de metodología ha sido muy eficiente en el desarrollo. Una de las ventajas que nos ha proporcionado, ha sido la capacidad de reaccionar ante los cambios teniendo la oportunidad de ir creando en cada *sprint*, especificaciones y requerimientos nuevos.

ZenHub

ZenHub es una plataforma de gestión de proyectos que se integra en Git-Hub, instalándose en el navegador mediante una extensión [6].

Es una herramienta muy cómoda y visual, ya que permite administrar todos los elementos comentados anteriormente característicos de la metodología SCRUM. Destacar que ZenHub llama a los *sprints*, *milestones*. El resto de nomenclatura es igual.

[5] No son siglas. Su significado viene de la palabra melé. Jugada de rugby en la que jugadores de ambos equipos se agrupan en una formación en la cual lucharán por obtener el balón que se introduce por el centro. [4]

[6] Podemos descargarlo en https://www.zenhub.com/

En la figura 4.2 tenemos presentes las diferentes columnas del tablero. Siendo de gran utilidad para clasificar diferenciar por *milestones*, cada *issue*.

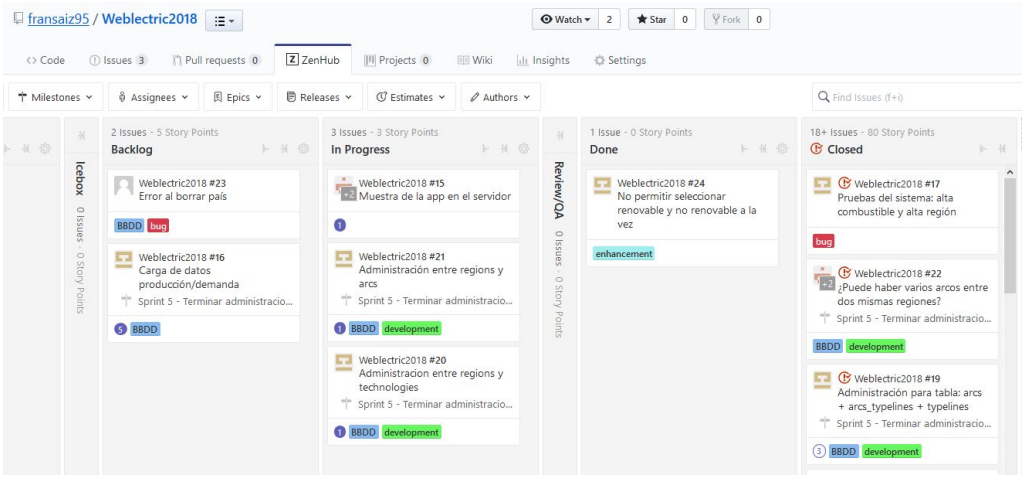

Figura 4.2: Ejemplo de nuestro tablero en *ZenHub*.

Control de versiones

Como repositorio y control de versiones, hemos elegido *Git* a través de la herramienta *GitHub*[7]

GitHub es una plataforma online basada en Git, que permite la creación de repositorios tanto públicos como privados en los que posteriormente un equipo puede alojar su trabajo.

Además, GitHub ofrece una version para escritorio (*GitHub Desktop*) con la que poder realizar subidas y bajadas (*push - pull*) directamente desde el escritorio. Nosotros, en cambio hemos utilizado *Sourcetree*. Un cliente Git que proporciona una interfaz amigable para interactuar con nuestros repositorios. Pertenece a la empresa *Atlassian*.

[7]Software de código abierto y gratuito que permite un control de versiones a través de ramas (*branchs*) en las que cada usuario puede editar y publicar cambios.

4.2. Entorno de desarrollo

Puesto que el objetivo principal del proyecto es construir una aplicación web dando soporte al cliente a que pueda cubrir sus necesidades, necesitamos un sitio donde alojar la aplicación.

Antes de contratar ningún servicio externo, empezamos a desarrollar la aplicación en nuestro entorno local.

Para ello necesitamos de la siguiente herramienta.

XAMPP

XAMPP es un paquete de software libre que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache e intérpretes para lenguajes PHP y Perl [14].

La ventaja de utilizar esta herramienta es que te ahorras el tiempo y la dificultad de configuración de cada uno de los servicios por separado.

HeidiSQL

Aunque XAMPP trae consigo el servicio phpMyAdmin para gestionar las bases de datos MySQL, nosotros hemos decidido utilizar *HeidiSQL* ya que tiene una interfaz más amigable y permite más opciones.

HeidiSQL es un software libre de código abierto que nos da la facilidad de conectarnos a servidores MySQL.

Para administrar las bases de datos con esta herramienta, el usuario tiene que iniciar sesión en un servidor MySQL local o remoto.

Filezilla

Una vez que tenemos nuestro entorno de desarrollo en local, es común necesitar un servidor externo donde alojar tu aplicación. Para transferir los archivos al servidor, utilizamos Filezilla. Un gestor FTP de código abierto y software libre.

Tras establecer la conexión con el servidor, el manejo de los archivos y la navegación ente los directorios es fácil e intuitiva. Además, permite arrastrar y soltar, lo que facilita su uso.

Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft. Al incluir control integrado de Git, hace que el resultado y el control de versiones de nuestro código sea más manejable.

No es un IDE como tal, pero gracias a numerosas extensiones hace que el trato con él sea mucho más satisfactorio. Acepta extensiones para hacer más facil la lectura de los lenguajes PHP, HTML, CSS y Javascript.

Prepros

Prepros es una herramienta completa para desarrollo front-end[8]. Se trata de un compendio de funcionalidades que abarcan desde el desarrollo con lenguajes como CSS o Javascript, a la optimización de imágenes.

Además, en mi caso, lo utilizo como compilador de archivos **.scss* a **.css* pues el desarrollo lo hago en Sass que aunque es parecido a CSS, no es igual. Posteriormente compilo mi estructura de archivos **.scss* en un *.css general que utiliza la web.

Para una correcta estructuración de la hoja de estilos, en la figura 4.3 podemos ver como se ha llevado a cabo. Tenemos un directorio con cada uno de los archivos *.scss* que se agrupan en un *estilos.scss* que a su vez, siendo compilado con el *Prepros*, genera un *estilos.css*. Este archivo es de donde se nutre la aplicación web.

[8]La parte del software que interactúa con los usuarios.

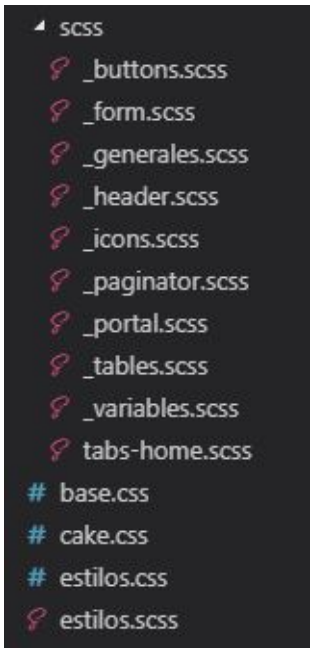

Figura 4.3: Estructura de directorios de la hoja de estilos.

4.3. Frameworks

A continuación, iremos mencionando los diferentes*frameworks* [9] que tras integrarlos entre sí, han hecho posible el desarrollo del proyecto.

CakePHP

CakePHP [6] es un *framework* que facilita el desarrollo de aplicaciones web en PHP, utilizando el patrón de diseño MVC [12].

Facilita alguna ayuda para integrar *Ajax*[10], Javascript, formularios... por lo que hace de su uso una herramienta interesante.

Además, vienen incluidos componentes de seguridad y de sesión que para una aplicación web siempre son interesantes.

En cuanto al Modelo Vista Controlador (figura 4.4), es un patrón de arquitectura de software que separa por una parte los datos y la lógica de

[9] Entorno de trabajo.

[10] Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. [11]

una aplicación y por otra, la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Separar las funciones de la aplicación en modelos, vistas y controladores hace que la aplicación sea mucho más ligera.

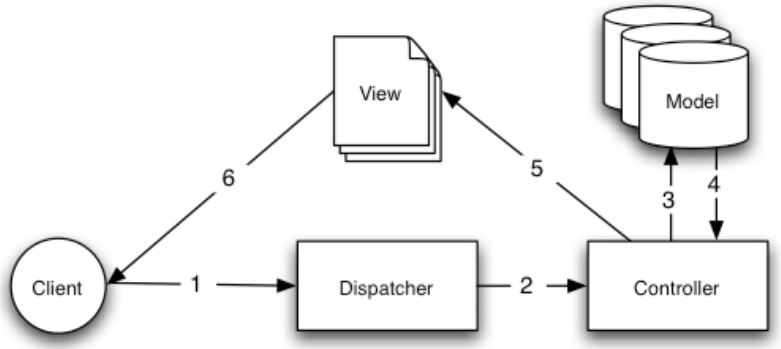

Figura 4.4: Ejemplo de arquitectura MVC.

Zurb Foundation

Foundation [10] es un *framework* de interfaz de usuario. Proporciona una cuadrícula responsive e incluye diferentes componentes:

- Interfaz de usuario HTML y CSS.
- Plantillas.
- Fragmentos de código reutilizables.
- Tipografías.
- Formularios.
- Botones, barras de navegación y otros componentes de interfaz usuario.
- Extensiones de Javascript opcionales.

Foundation está mantenida por Zurb y es un proyecto de código abierto.

La competencia más directa de *Foundation* como *framework de CSS*, es *Bootstrap*. Aunque en nuestra aplicación hemos usado *Foundation*, cabe destacar que *Bootstrap* ofrece unos servicios muy parecidos y una documentación de calidad como la de *Foundation*.

Por destacar alguna ventaja a favor de *Bootstrap*, podemos decir que es más popular. Tiene más plugins desarrollados y una comunidad de usuarios más grande, lo que facilita cualquier tipo de consulta en la web. Sin embargo, nosotros hemos decidido utilizar Foundation, ya que no considerábamos insignificante la diferencia de prestaciones entre ambos *frameworks*. Además, *Foundation* viene con un archivo de ejemplo con el que hemos podido practicar y obtener conocimientos iniciales.

Aspectos relevantes del desarrollo del proyecto

5.1. Tratamiento de datos

Lectura de datos

Teniendo en cuenta los objetivos del proyecto ya comentados en apartados anteriores y queriendo lograr la construcción de una aplicación estable para que el usuario final pueda cargar sus datos y administrarlos, nos vimos obligados en primera instancia a buscar una librería en PHP capaz de leer una serie de datos de un Excel[11] para posteriormente alojarlos en nuestra base de datos.

Phpspreadsheet

Para realizar esta operación conocíamos una librería llamada PHPExcel pero que actualmente a día de hoy está deprecada, por tanto, me vi en la obligación de investigar y buscar cuál era la solución actualmente. Así conseguimos dar con *Phpoffice/Phpspreadsheet*. Mediante esta librería, que podemos encontrar su documentación en phpspreadsheet [8], podemos realizar lecturas y escrituras sobre un Excel.

Como dejamos indicado en la figura 5.5 la librería cumple con creces nuestros requisitos además de ofrecernos alguna opción extra por si en un futuro o en próximas versiones del proyecto fuesen de interés [8].

[11] El cliente nos envió así los datos que en un futuro desearía que fuesen administrables.

Format	Reading	Writing
Open Document Format/OASIS (.ods)	✓	✓
Office Open XML (.xlsx) Excel 2007 and above	✓	✓
BIFF 8 (.xls) Excel 97 and above	✓	✓
BIFF 5 (.xls) Excel 95	✓	
SpreadsheetML (.xml) Excel 2003	✓	
Gnumeric	✓	
HTML	✓	✓
SYLK	✓	
CSV	✓	✓
PDF (using either the TCPDF, Dompdf or mPDF libraries, which need to be installed separately)		✓

Figura 5.5: Formatos de ficheros soportados por Spreadsheet.

Para instalar librerías en CakePHP[12], hay diversas opciones, pero una de las más sencillas es utilizar *Composer*[13].

Para instalar *Spreadsheet* en nuestro proyecto, nos dirigimos mediante comandos a la ruta de nuestro directorio que a tyravés de *Composer* instalaremos la librería:

```
composer require phpoffice/phpspreadsheet
```

Una vez la instalada, simplemente con la siguiente linea podemos importarlo en nuestro proyecto.

```
use PhpOffice\PhpSpreadsheet\Spreadsheet;
```

Como vemos, aunque el trato de los datos y la organización de los mismos si que ha sido un tema delicado, la instalación de las librerías en CakePHP es bastante sencillo.

[12]Es el framework que hemos utilizadop para el desarrollo del proyecto

[13]Herramienta para la gestión de dependencias en PHP. Le permite declarar y administrar las bibliotecas de las que depende su proyecto.

Escritura y almacenamiento de los datos

En cuanto al manejo de los datos, aquí si que hemos tenido más problema. Se nos ha juntado la gran cantidad de datos con que el servidor donde hemos alojado esta primera versión de la aplicación no cuenta con recursos muy elevados.

Hemos tenido problemas a la hora de subir al servidor archivos muy grandes pues se agotaba la memoria al guardar los datos [14] y se quedaba atascado. En nuestro entorno de desarrollo en local, el problema del tamaño de los archivos lo solucionamos modificando la propiedad $upload_max_filesize$ en el archivo *php.ini* pero al no tener acceso a este archivo en el servidor, no podemos modificarlo. El problema de la memoria al guardar, no lo hemos podido solucionar aun utilizando la sentencia $ini_set('memory_limit', -1)$; para indicar que puede utilizar toda la memoria que necesite y la sentencia $set_time_limit(0)$; para indicar que tiene el tiempo de ejecución ilimitado no lo hemos podido resolver en el entorno local pues se nos acababan los recursos.

Por esta razón es por lo que hemos tenido que fragmentar las cargas y administraciones de ciertos datos aunque lo ideal de cara a la usabilidad del usuario hubiera sido poder hacerlo todo en un mismo paso.

Para almacenar todos estos datos y poder dar al usuario la posibilidad de administrarlos, hemos construido una base de datos relacional. Para gestionarla, hemos utilizado el software *HeidiSQL* que ofrece la posibilidad de conectarse a servidores *MySQL*[15].

5.2. Estructura de la aplicación

La estructura de directorios de nuestro proyecto lo podemos ver en la figura 5.6. Destacamos los siguientes directorios:

- *config*: En la figura 5.7 vemos los ficheros de configuración de nuestra aplicación. Con una importancia especial podemos destacar:
  - app.php: Archivo donde se establecen los parámetros de configuración para el email, la base de datos, los logs, la sesión, el debug, etc.

[14] Rondando las 400K registros.
[15] Sistema de gestión de bases de datos relacional

  - constantes.php: Como indica el nombre del archivo, aquí se declaran clases con constantes comunes para poder usarlas desde el resto de la aplicación.
  - paths.php: Definir variables globales para rutas de directorios concretos.

- *webroot*: Dentro de la estructura cliente-servidor, en la figura 5.8 encontramos lo relacionado con el cliente. Nuestra hoja de estilos, los archivos $*.js$, las imágenes, las fuentes y el favicon.pnp[16].

- *src*: Aquí se almacenarán los archivos de tu aplicación.

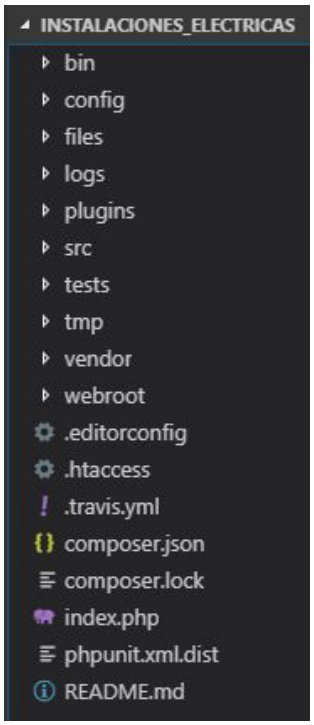

Figura 5.6: Estructura general de un proyecto desarrollado con CakePHP.

No obstante, en la documentación oficial de *CakePHP* [3] se explican con mayor detalle cada uno de los directorios del proyecto.

[16]Se conoce como favicon al icono que aparece en la pestaña del navegador junto con el nombre de la aplicación.

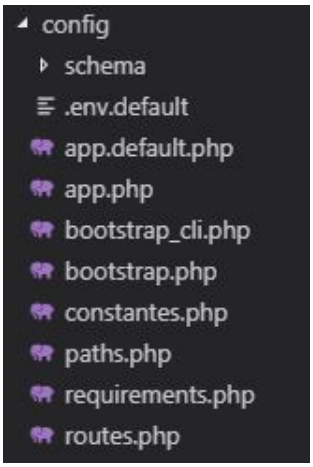

Figura 5.7: Directorio *config* de nuestro proyecto.

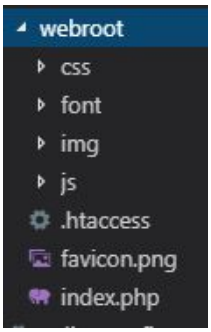

Figura 5.8: Directorio *webroot* de nuestro proyecto.

MVC

El utilizar *CakePHP* como *framework* nos ha facilitado mucho la estructura del proyecto pues trabaja con el patrón MVC. De esta manera, como podemos ver en la figura 5.9, tenemos bien separadas las tres partes fundamentales de la aplicación.

Por un lado tenemos la conexión con nuestra base de datos en lo que serían los Modelos[17]. Allí están la mayoría de consultas y operaciones contra la base de datos.

Por otro lado tenemos la parte de los controladores[18]. que hacen de intermediarios entre la base de datos y la vista. Aquí albergamos la parte

[17]Directorio *Models*.
[18]Directorio *Controller*.

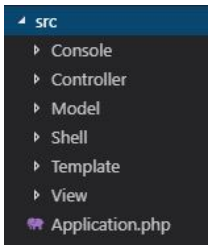

Figura 5.9: Directorio *src* de nuestro proyecto.

lógica de nuestra aplicación.

Por último las vistas[19]. Esto es lo que ve el usuario. Como ya hemos comentado, para facilitarnos el trabajo y aprovechar herramientas muy útiles ya creadas y puestas en el mercado, hemos utilizado *Foundation 6.4.2*

Su implementación dentro del proyecto es muy sencilla, simplemente tenemos que colocar la siguiente linea en el *layout*[20]:

```
echo $this->Html->css('lib/foundation-6.4.2/css/foundation.css');
```

- Explicar por que hemos utilizado esa guia de estilos.
- Select 2 para los formularios
- Sweet Alert para los mensajes de confirmación.
- Ionicons y flaticons para los iconos gratis
- Jquery

[19] Directorio *Templates*.
[20] Vista definida como plantilla común al resto de vistas.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Altran. Metodología ágil vs metodología tradicional., 2016.

[2] title = Andrés Ramos Begoña Vitoriano.

[3] CakePHP. Cakephp folder structure., 2018.

[4] Emma Blanco Muñoz Jose Antonio Dorado Cerón. ¿qué significa scrum?, 2010.

[5] Universidad Tecnológica de Pereira. Optimización multiobjetivo usando un algoritmo genético y un operador elitista basado en un ordenamiento no-dominado (nsga-ii)., 2007. [Internet; descargado 2-diciembre-2018].

[6] Cake Software Foundation. Cakephp 3.6 red velvet cookbook, 2018.

[7] Juan Andrés Ambuludi Olmedo. Aplicación de algoritmos genéticos para la optimización de problemas combinatorios, 2017-2018. [Internet; descargado 1-diciembre-2018].

[8] PhpSpreadsheet. Welcome to phpspreadsheet's documentation.

[9] Johan Alexander Aranda Pinilla. Optimización multiobjetivo en la gestión de cadenas de suministro de biocombustibles. una revisión de la literatura, 2015.

[10] Foundation Team. Foundation.the most advanced responsive front-end framework in the world., 2018.

[11] Wikipedia. Ajax, 2018.

[12] Wikipedia. Cakephp, 2018.

[13] Wikipedia. Optimización, 2018.

[14] Wikipedia. Xampp, 2018.