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 — 9 de enero de 2019
Tutor: Dr. Álvar Arnaiz González
Dr. Jesús Maudes Raedo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz González y D. Jesús Maudes Raedo, profesores del departamento de Ingeniería Civil del área de Lenguajes y Sistemas Informáticos.

Exponen:

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, 9 de enero de 2019

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 general III

Índice de figuras V

Índice de tablas VI

Introducción 1

Objetivos del proyecto 3
2.1. Objetivos generales 3
2.2. Objetivos personales 4

Conceptos teóricos 5
3.1. Optimización 5
3.2. NSGA-II 8

Técnicas y herramientas 11
4.1. Gestión del proyecto y control de versiones 11
4.2. Entorno de desarrollo 14
4.3. Frameworks 16

Aspectos relevantes del desarrollo del proyecto 19
5.1. Tratamiento de datos 19
5.2. Estructura de la aplicación 21

Trabajos relacionados 25
6.1. Energy Exemplar® 25
6.2. Bentley Systems 26

6.3. LEAP 27
6.4. OSeMOSYS 28
6.5. Artículos científicos 29

Conclusiones y Líneas de trabajo futuras **31**

Bibliografía **33**

Índice de figuras

1.1. Weblectric. 1
3.2. Frente de pareto en un problema de minimización. Ilustración extraida de [24]. 9
4.3. Ejemplo de nuestro tablero en *ZenHub*. 13
4.4. Estructura de directorios de la hoja de estilos. 16
4.5. Ejemplo de arquitectura MVC. Ilustración extraida de [19] . . . 17
5.6. Formatos de ficheros soportados por Spreadsheet. 20
5.7. Estructura general de un proyecto desarrollado con CakePHP. . 22
5.8. Directorio *config* de nuestro proyecto. 23
5.9. Directorio *webroot* de nuestro proyecto. 23
5.10. Directorio *src* de nuestro proyecto. 24

Índice de tablas

Introducción

La optimización consiste en la selección del mejor elemento (con respecto a algún criterio) dentro de un conjunto de elementos disponibles [31]. 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 calculando el valor de la función [31].

Existen diferentes métodos de optimización a los que se hará referencia en algún punto de la documentación, sin embargo, el proyecto está enfocado a la administración a través de una aplicación web de unos datos que nos facilita el cliente fruto de la ejecución de un algoritmo multi-objetivo llamado *Nondominated Sorting Genetic Algorithm II (NSGA-II)* [9].

Para la administración de todos estos datos, en la figura 1.1, presentamos *Weblectric*, una aplicación desarrollada por petición de un cliente residente en México con la idea de darle la posibilidad de ejecutar su propio algoritmo desde la web y poder administrar los resultados.

A lo largo del proyecto podremos ver su estructura, el *framework* utilizado, su diseño y algunas librerías de interés para leer y escribir datos con PHP en las hojas de cálculo. De la misma manera, reflejaremos los problemas detectados, cómo se han resuelto y qué lineas de futuro marcamos en el horizonte para continuar con el desarrollo de *Weblectric*.

Figura 1.1: Weblectric.

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:

- Alojar nuestros contenidos en un servidor al que accederemos mediante un cliente FTP.
- 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 [2] que nos permita desarrollar en nuestro entorno local.
- Familiarizarme con CakePHP. Un framework de desarrollo en PHP con arquitectura MVC[11][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

- Conocer el funcionamiento general del algoritmo NSGA-II [9], un algoritmo de optimización.

2.2. Objetivos personales

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

- Aprender a medir cómo trabajo y el tiempo que necesito para las diferentes tareas de análisis, diseño y desarrollo.
- Aprender a adelantarme a las necesidades del cliente, aportándole una solución concreta, real y efectiva.
- 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 también en el desarrollo de la aplicación.

Como objetivo final del proyecto me gustaría aprender a gestionar el desarrollo de aplicaciones web para moverme y tener buenas oportunidades laborales dentro de este sector.

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 distintas regiones.

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 [31].

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 calculando el valor de la función [31].

Optimización clásica vs. Metaheurística

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

- Programación lineal [30].

- Programación cuadrática [16].
- Programación no lineal [8].
- Programación dinámica [33].
- Teoría de grafos u optimización en redes [7].

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

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

Algún método metaheurístico es:

- Algoritmos evolutivos (genéticos) [28].
- Recocido o simulado [21].
- Sistemas multiagente [6].

En esta aplicación, al igual que en muchas otras, se utilizan algoritmos de optimización metaherísticos porque el cálculo de un óptimo global es imposible de obtener en un tiempo razonable. Los métodos metaheurísticos, ofrecen un óptimo razonablemente"bueno.

Componentes

A continuación, se van a describir los componentes de un sistema de optimización [29].

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 $f : A \to R$

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 $f(x_0) \geq f(x)$ 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 solo hay un objetivo mientras que en los segundos hay más de uno y además pueden y suelen ir en direcciones opuestas.

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)* [9].

3.2. NSGA-II

Implementando esta solución, la meta a la que se quiere llegar es a encontrar un vector de variables $x = (x_1, x_2, \dots x_j)$ que cumpla con todas las restricciones y condiciones, donde las funciones objetivos resultantes sean optimizadas [22].

Se denomina espacio de solución al conjunto de todas las combinaciones posibles. Es denotado mediante: $fn(x) = z = (z_1, z_2, \dots z_M)$.

Sin embargo, en los problemas multiobjetivo, 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 $x(1)$ predomina sobre otra solución $x(2)$ si se cumplen las siguientes condiciones [9]:

- 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 problema de optimización multiobjetivo, se puede llegar a establecer cuáles 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* [24]. (Figura 3.2)

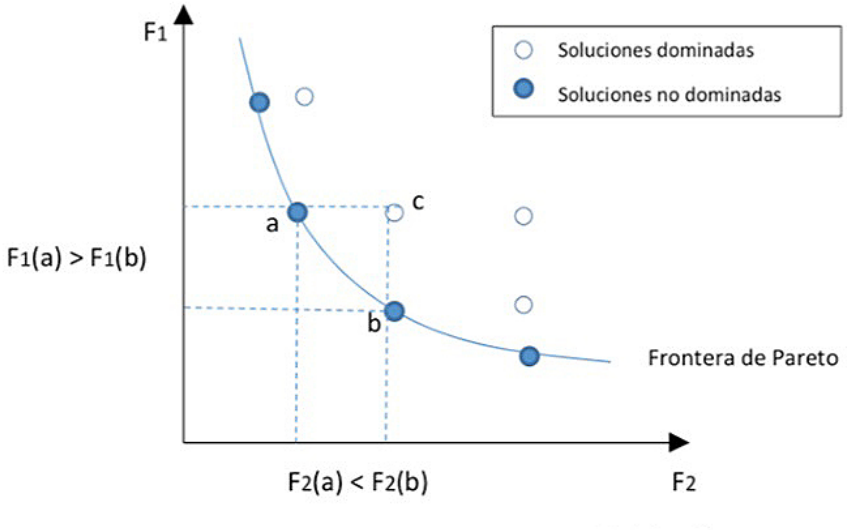

Figura 3.2: Frente de pareto en un problema de minimización. Ilustración extraida de [24].

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 técnicas y 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 *Project Management Body of Knowledge*[2].
- 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.

[2]PMBoK. 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 qué necesita pero desconoce cómo 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[3] [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 *GitHub*, instalándose en el navegador mediante una extensión [4].

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.

[3] 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. [10]

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

En la figura 4.3 tenemos presentes las diferentes columnas del *tablero*. Este elemento, es de gran utilidad para clasificar por *milestones*, cada *issue*.

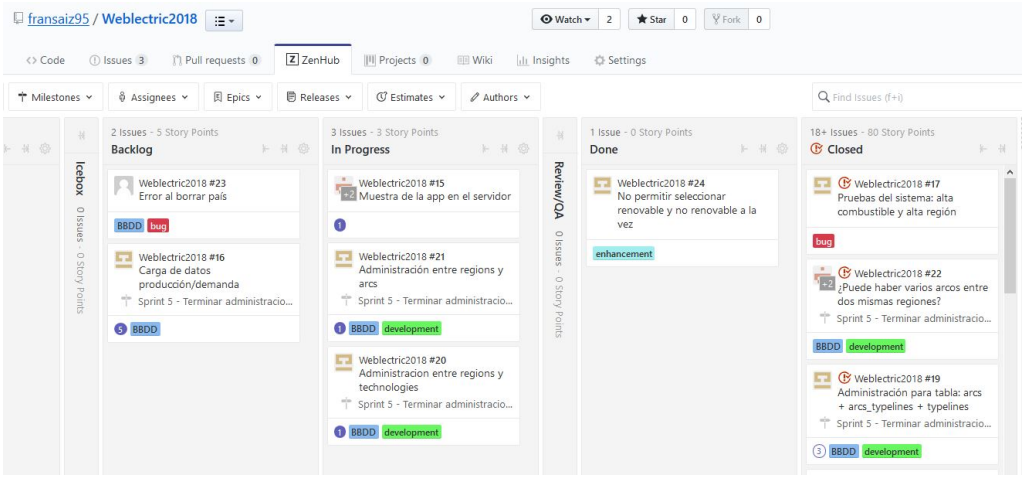

Figura 4.3: Ejemplo de nuestro tablero en *ZenHub*.

Control de versiones

Como repositorio y control de versiones, hemos elegido *Git* a través de la herramienta *GitHub*[5]

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 hemos utilizado *Sourcetree*, un cliente Git que proporciona una interfaz amigable para interactuar con nuestros repositorios. Pertenece a la empresa *Atlassian.*

A través del siguiente enlace, podemos acceder al proyecto *Weblectric* que utiliza *GitHub* como repositorio y control de versiones.

https://github.com/fransaiz95/Weblectric2018

[5]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 [15].

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 fácil la lectura de los lenguajes *PHP*, *HTML*, *CSS* y *Javascript*.

Prepros

Prepros es una herramienta completa para desarrollo front-end[6]. 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 este trabajo, se utiliza como compilador de archivos **.scss* a **.css* pues el desarrollo se hace en *Sass*, que es parecido a *CSS*. Posteriormente se compila la estructura de archivos **.scss* en un **.css* general que utiliza la web.

Para una correcta estructuración de la hoja de estilos, en la figura 4.4 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.

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

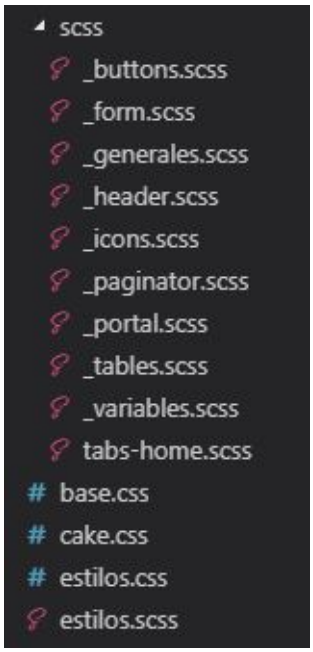

Figura 4.4: Estructura de directorios de la hoja de estilos.

4.3. Frameworks

A continuación, iremos mencionando los diferentes *frameworks (Entornos de trabajo)* que tras integrarlos entre sí, han hecho posible el desarrollo del proyecto.

CakePHP

CakePHP [14] es un *framework* que facilita el desarrollo de aplicaciones web en PHP [3], utilizando el patrón de diseño MVC.

Facilita alguna ayuda para integrar *Ajax*[7], 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 4.5), es un patrón de arquitectura de software que separa por una parte los datos y la lógica de

[7]Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas [17].

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.

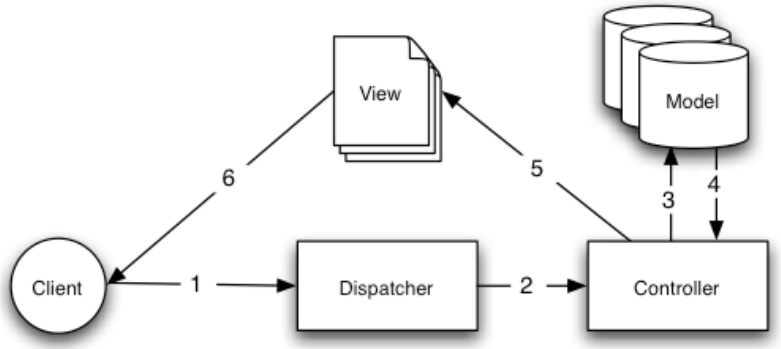

Figura 4.5: Ejemplo de arquitectura MVC. Ilustración extraida de [19]

Zurb Foundation

Foundation [27] 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, se ha decidido utilizar *Foundation*, ya que 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 desde una hoja de cálculo, pues el cliente nos envió los datos en dicho formato, para que en un futuro fuesen administrables y poder ser alojados en nuestra base de datos.

Phpspreadsheet

Para realizar esta operación, el alumno conocía una librería llamada *PHPExcel* pero que actualmente a día de hoy está deprecada, por tanto, hubo que investigar y buscar cuál era la solución actualmente. Así se consiguió dar con *Phpoffice/Phpspreadsheet*. Mediante esta librería, que podemos encontrar su documentación en phpspreadsheet [23], se puede realizar lecturas y escrituras sobre una hoja de cálculo.

En la figura 5.6 se puede ver que la librería cubre con creces los requisitos necesarios, además de ofrecer alguna opción extra por si en un futuro o en próximas versiones del proyecto fuesen de interés [23].

Para instalar librerías en CakePHP, framework utilizado para el desarrollo del proyecto, hay diversas opciones, pero una de las más sencillas es utilizar

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 5.6: Formatos de ficheros soportados por Spreadsheet.

Composer[8].

Para instalar *Spreadsheet* mediante *Composer*, desde el directorio del proyecto ejecutamos:

```
composer require phpoffice/phpspreadsheet
```

Una vez la instalada, simplemente con la siguiente linea se puede importar en nuestro proyecto.

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

Como se ve, 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 sencilla.

Escritura y almacenamiento de los datos

En cuanto al manejo de los datos, aquí si que ha habido más problemas. Se nos ha juntado la gran cantidad de datos, con que el servidor donde

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

hemos alojado esta primera versión de la aplicación no cuenta con recursos muy elevados.

Ha habido problemas a la hora de subir al servidor archivos muy grandes pues se agotaba la memoria al procesar los datos [9] y se quedaba atascado.

En nuestro entorno de desarrollo local, el problema del tamaño de los archivos lo solucionamos modificando la propiedad `upload_max_file_size` 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 procesar los datos, no se ha podido solucionar ni con la sentencia `ini_set('memory_limit', '-1');` para indicar que puede utilizar toda la memoria que necesite, ni con la sentencia `set_time_limit(0);` para indicar que tiene el tiempo de ejecución ilimitado.

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, se ha construido una base de datos relacional. Para gestionarla, se ha utilizado el software *HeidiSQL* que ofrece la posibilidad de conectarse a servidores *MySQL*.

5.2. Estructura de la aplicación

La estructura de directorios de nuestro proyecto se puede ver en la figura 5.7. Destacamos los siguientes directorios:

- `config`: En la figura 5.8 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.
  - `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.

[9] Rondando las 400K registros.

- `webroot`: Dentro de la estructura cliente-servidor, en la figura 5.9 se encuentra lo relacionado con el cliente. Nuestra hoja de estilos, los archivos $*.js$, las imágenes, las fuentes y el favicon.pnp[10].

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

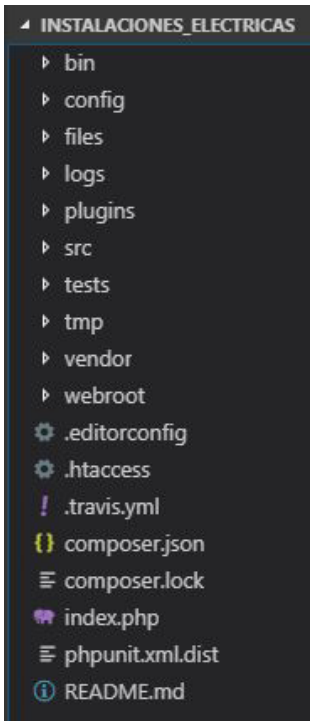

Figura 5.7: Estructura general de un proyecto desarrollado con CakePHP.

No obstante, en la documentación oficial de *CakePHP* [4] se explican con mayor detalle cada uno de los directorios del proyecto.

MVC

El utilizar *CakePHP* como *framework* ha facilitado mucho la estructura del proyecto pues trabaja con el patrón MVC. De esta manera, como podemos ver en la figura 5.10, tenemos bien separadas las tres partes fundamentales de la aplicación.

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

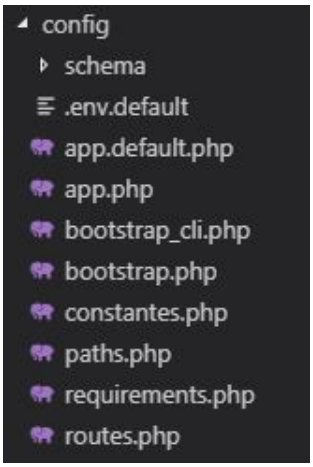

Figura 5.8: Directorio *config* de nuestro proyecto.

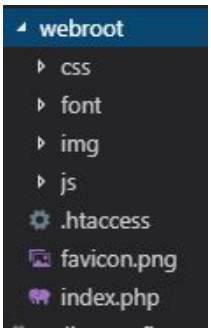

Figura 5.9: Directorio *webroot* de nuestro proyecto.

Por un lado tenemos la conexión con nuestra base de datos en lo que serían los *modelos*[11]. Allí están la mayoría de consultas y operaciones contra la base de datos.

Por otro lado tenemos la parte de los *controladores*[12]. que hacen de intermediarios entre la base de datos y la vista. Aquí albergamos la parte lógica de nuestra aplicación.

Por último las *vistas*[13]. 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*

[11] Directorio *Models*.
[12] Directorio *Controller*.
[13] Directorio *Templates*.

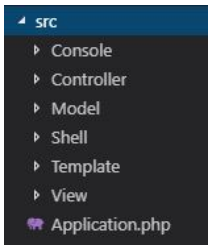

Figura 5.10: Directorio *src* de nuestro proyecto.

Su implementación dentro del proyecto es muy sencilla, simplemente tenemos que colocar la siguiente linea en el *layout*[14]:

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

[14]Vista definida como plantilla común al resto de vistas.

Trabajos relacionados

En este apartado vamos a ir presentando algunos de los recursos competidores existentes en el mercado. Actualmente, ya hay varias plataformas que ofrecen un software personalizado para la optimización de energía y redes eléctricas. Iremos desglosando las diferentes opciones y destacando los detalles más relevantes.

6.1. Energy Exemplar®

Energy Exemplar® presume de ser el líder del mercado en tecnología de simulación de energía basada en la optimización.

Su paquete de software[15], encabezado por *Plexos* y *Aurora*, se utiliza en todas las regiones del mundo para una amplia gama de aplicaciones, desde análisis a corto plazo hasta estudios de planificación a largo plazo. Permiten minimizar los costos operativos y de inversión, maximizar las ganancias y obtener pronósticos mucho mas precisos. Ofrecen sus servicios como el mejor software de simulación de energía eléctrica, gas y agua.

La empresa se distribuye por todo el mundo contando con 9 oficinas repartidas por los cinco continentes.

Plexos

Como comentábamos antes, *Plexos* es uno de los software que proporcione Energy Exemplar como solución [13]. Combina técnicas de optimización basadas en matemática para el pronóstico con una experiencia gráfica de

[15]Publican su primera versión en el año 2000.

usuario muy potente y flexible. Presume de ofrecer lo último en gestión de datos orientados a objetos

Plexos Connect mejora el software anterior al ofrecer computación distribuida[16] a través de recursos locales y en la nube. Además ofrece ejecuciones por lotes completamente automatizadas y operaciones en tiempo real. Aloja los resultados de simulación en un repositorio central.

Aurora

Como afirman sus creadores [12], *Aurora* es un software de análisis y pronóstico de modelado eléctrico confiable, fácil de aprender y centrado en el usuario.

Los cambios rápidamente personalizables, sus bases de datos integradas y la interfaz fácil de usar proporcionan resultados muy satifactorios en el tiempo. Aurora permite el nivel más alto de integración de software, control de modelos y facilidad de intercambio de datos, ahorrando al usuario tiempo y dinero.

6.2. Bentley Systems

Bentley Systems es una empresa de desarrollo de software que respalda las necesidades profesionales de los responsables de la creación y gestión de la infraestructura mundial: carreteras, aeropuertos, puentes, plantas industriales, eléctricas, etc.

Ofrece soluciones para todo el ciclo de vida del activo de infraestructura, adaptadas a las necesidades de las distintas profesiones que trabajarán con ese activo durante su ciclo de vida [25].

En la actualidad, la empresa tiene sede en: Estados Unidos, Irlanda y China.

Advancement Academy

Innovador programa que ofrece la posibilidad de coordinar equipos de ejecución complejos de arquitectura, ingeniería y construccion (AEC). Permite gestionar la complejidad de datos que se producen al incorporar a los contratistas.

[16] La computación distribuida es un modelo para resolver problemas de computación masiva utilizando un gran número de ordenadores organizados en clústeres. [32]

Garantiza que sus colaboradores entiendan los procesos y productos a entregar que se esperan para una ejecución eficaz del proyecto. La herramienta ofrecerá un currículo específico para el proyecto que será escalable y flexible para adaptarse a cualquier situación [26].

Sus soluciones abarcan todo el ciclo de vida activo, desde el diseño hasta la construcción, pasando por la optimización de las centrales.

En el siguiente *enlace*, puede conocer algunos de los proyectos llevados a cabo por *Bentley Systems*.

6.3. LEAP

LEAP, el sistema de planificación de alternativas energéticas a largo plazo, es una herramienta de software adoptada y utilizada por 190 países en todo el mundo para el análisis de políticas energéticas y la evaluación del ahorro energético desarrollada en el Instituto de Medio Ambiente de Estocolmo [20].

Entre sus usuarios, tenemos presentes a académicos, agencias gubernamentales, organizaciones no gubernamentales, empresas consultoras y empresas de energía. Su uso varía desde ciudades y estados hasta aplicaciones nacionales, regionales y globales.

En los últimos tiempos, se está convirtiendo en el estándar de facto para los países que realizan una planificación integrada de recursos, evaluaciones de mitigación de gases de efecto invernadero (GEI) y Estrategias de Desarrollo de Baja Emisión (LEDS).

Al menos 32 países utilizaron el *LEAP* para crear escenarios de energía y emisiones.

Es una herramienta de modelado integrada y basada en escenarios cuyo uso puede ser rastrear la producción, el consumo de energía y la extracción de recursos en todos los campos de una economía.

De cara al usuario, cuenta con buena reputación porque ha sido capaz de hacer transparente el uso de conceptos complejos de análisis de energía. Del mismo modo, es flexible para usuarios que si cuentan con conocimientos a fondo y experiencia.

No es un modelo de un sistema de energía en particular. Es una herramienta que se puede utilizar para cubrir unas necesidades al crear modelos de diferentes sistemas de energía, donde cada modelo requiere estructuras de

datos personales y únicas. Es compatible con una amplia gama de diferentes metodologías de modelado a medio y largo plazo [20].

Una gran parte de los estudios realizados, utilizan un período de pronóstico de entre 20 y 50 años. Sin embargo, aunque suele ser lo habitual, no siempre es así. Por ejemplo, para los cálculos del sector eléctrico, el año se suele dividir en diferentes ïntervalos de tiempo"definidos por el usuario para representar periodos de tiempo como pueden ser temporadas, días o incluso horas con un valor especial en el día.

La aplicación se distribuye en diferentes vistas/pantallas que hace que el usuario pueda interaccionar con el software.

- Vista de análisis.
- Herramientas para crear modelos.
- Informe de resultados.
- Balances de energía.
- Diagramas de Sankey.[17]
- Base de datos de tecnología y medio ambiente (TED).
- Notas y documentación.

6.4. OSeMOSYS

Alternativa de código abierto para la evaluación integrada y la planificación energética a largo plazo. El proyecto nace en 2008 durante una presentación en París.

Diseñado para aquellos que no pueden o deseen hacer una inversión financiera inicial. Cuenta con una curva de aprendizaje rápida y un compromiso de operación de poco tiempo. Gracias a su transparencia, su uso se puede encontrar como herramienta de difusión y captación.

Model Management Infrastructure (MoManI) es una interfaz de código abierto basada en navegador enfocada en el modelado de sistemas energéticos. La estructura de la aplicación ayuda a disminuir la complejidad perceptible

[17]Se utilizan para visualizar los flujos de balance de energía para cualquier área que se esté modelando en LEAP.

de *OSeMOSYS* y su estructura permite que varios equipos colabores simultáneamente desde cualquier parte del mundo. El usuario puede actualizar y editar sin dificultad cualquier parte del proceso de modelado: desde la visualización de los resultados hasta las ecuaciones matemáticas subyacentes de *OSeMOSYS*.

6.5. Artículos científicos

El 15 de Junio de 2016, Khizir Mahmud,Graham E. Town publicó un artículo [18] en el que hacía una revisión de las herramientas informáticas presentes en el mercado que sirviesen para modelar los requisitos energéticos de los vehículos eléctricos y su imparto en las redes de distribución de la energía. El artículo tuvo un gran impacto y los proyectos comentados anteriormente hacen referencia al artículo citado.

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] Peter Laurie Ben Laurie. Apache: The definitive guide, 3rd edition, 2009.

[3] Inc Cake Software Foundation. ¿qué es cakephp y por qué hay que utilizarlo?, 2012.

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

[5] Diego Calvo. Metodología scrum (metodología ágil), 2018.

[6] Fernando Sancho Caparrini. Sistemas multiagente y simulación., 2016.

[7] José María Ferrer Caja. Universidad Pontificia Comillas. Teoría de grafos y optimización en redes., 2012.

[8] Gestión de operaciones. ¿qué es la programación no lineal?., 2018.

[9] 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].

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

[11] Ralph Johnson Erich Gamma, Richard Helm and John Vlissides. Design patterns: Elements of reusable object-oriented software, 2009.

[12] Energy Exemplar. Aurora - software de análisis y pronóstico de modelado eléctrico., 2018.

[13] Energy Exemplar. Software de simulación plexos®., 2018.

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

[15] Apache Friends. Xampp apache + mariadb + php + perl. ¿qué es xampp?, 2018.

[16] Ruben Garcia. Metodo de programacion cuadratica., 2013.

[17] Jesse James Garrett. Ajax: A new approach to web applications, 2005.

[18] Graham E. Town Khizir Mahmud. A review of computer tools for modeling electric vehicle energy requirements and their impact on power distribution networks, 2016.

[19] Luiz Paulo Ladeira. Introdução ao framework cakephp.

[20] LEAP. Leap: an online community for energy analysts working for sustainability and the home of the leap software system., 2018.

[21] Blanca Rosa Pérez Salvador. Miguel Ámgel Gutiérrez Andrade, Sergio Gerardo de los Cobos Silva. Optimización con recocido simulado para el problema de conjunto independiente., 1998.

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

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

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

[25] Bentley Systems. Acerca de bentley systems, incorporated., 2018.

[26] Bentley Systems. Incorporación, mejores prácticas y gestión del cambio., 2018.

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

[28] Pablo Estevez Valencia. Optimización mediante algoritmos genéticos., 1997.

[29] Andrés Ramos Begoña Vitoriano. Modelos matemáticos de optimización.

[30] Vitutor. Programación lineal., 2017.

[31] Wikipedia. Optimización, 2018 (Accedido 05/12/2018).

[32] Wikipedia. Computación distribuida., 2018 (Accedido 27/12/2018).

[33] Víctor Manuel Quesada Ibarguen y Juan Carlos Vergara Schmalbach. ProgramaciÓn dinÁmica., 2006.