UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Weblectric
Documentación Técnica

Presentado por Francisco Saiz Güemes
en Universidad de Burgos — 3 de febrero de 2019
Tutor: Dr. Álvar Arnaiz González
Dr. Jesús Maudes Raedo

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En el siguiente apartado se presentará la planificación del proyecto y la metodología adoptada junto con las técnicas y herramientas utilizadas.

La metodología que se ha utilizado para el desarrollo del proyecto ha sido *Scrum*, una metodología de desarrollo ágil.

Para saber qué recursos se necesitan, hay que analizar todas las partes que forman el proyecto. El analisis podemos desglosarlo en:

- Planificación temporal: Tiempo que tendremos que invertir en cada una de las fases del desarrollo.
- Estudio de viabilidad: Aquí tendremos en cuenta las licencias necesarias, beneficios, costes y normativas de leyes a cumplir.

A.2. Planificación temporal

La metodología comentada anteriormente, *Scrum*, facilita la organización pues está diseñada para poder realizar entregas parciales y periódicas en pequeños y cortos espacios de tiempo, donde unos resultados y una flexibilidad considerable son fundamentales. Para concretar todo lo comentado, se necesitaba una herramienta que nos permitiera desarrollar el producto acorde a esta metodología y así es como se empezó a utilizar *GitHub* como plataforma donde alojar el proyecto.

*T*anto Scrum como la herramienta *GitHub*, contienen numerosos términos, pero a continuación explicaremos aquellos que se han utilizado en el proyecto.

- *Milestone:* También llamado *Sprint*. En nuestro caso, tiempo comprendido entre una y dos semanas. A cada *Milestone* le corresponde un número determinado de *issues*. Al inicio de cada *Sprint* hemos tenido como costumbre realizar una pequeña reunión de no más de una hora para comentar dificultades tenidas hasta ese momento y qué cosas nuevas íbamos a planificar.
- *Issue:* Así llamamos a cada una de las pequeñas tareas a realizar en este periodo de tiempo. No hay número máximo de *issues* por *milestone*. En nuestro caso, ha oscilado entre siete o quince, en función de la duración del sprint y del tiempo del que disponíamos.
- *ZenHub:* A través de *ZenHub*[1], una plataforma que se integra en *GitHub* mediante la instalación de una extensión para el navegador, se puede gestionar el tiempo de una manera más eficiente ya que permite asignar tiempos a cada una de las *issues*. De esta manera, te permite hacer cálculos para evitar crear más issues de las que ibas a ser capaz de realizar y poder organizar mejor los siguientes *sprints*.
- *Board:* Otro de los elementos más significativos de *ZenHub* es el tablero. Cuenta con numerosas *pipelines* personalizables entre las que mover *issues* en función de su estado de realización.

A continuación se mostrará el avance a lo largo de las iteracciones del proyecto:

Sprint 0: Inicio del proyecto

Reunión inicial en la que se comenta en rasgos generales la estructura y finalidad del desarrollo de este proyecto.

Issues:

- Elegir el *framework* que se va a utilizar. Ver qué versión de *CakePHP* es la más estable hoy en día.
- Diseñar la base de datos a través de un modelo Entidad-Relación.

[1]Se ha valorado la utilización de alguna alternativa como puede ser *Trello* pero nos hemos decantado por *ZenHub* al integrarse perfectamente en *GitHub*.

- Buscar un servidor gratuito donde poder alojar la aplicación.
- Crear la estructura de directorios del proyecto.

Este *sprint* ha servido para tomar un primer contacto con el proyecto, preparar el entorno de desarrollo en el que, a través de *XAMPP*, se ha tenido que simular en local un servidor *Apache* con *PHP* y *MySQL*. Además, se ha realizado una primera búsqueda de lo que sería el servidor donde se va a alojar y ha servido para refrescar conceptos de bases de datos y realizar el modelo entidad relación.

En cuanto al *framework* a utilizar, el alumno se decantó por *CakePHP 3.5* pues aunque en la empresa en la que trabaja el alumno se utiliza esta herramienta, se hace en una versión más antigua, por lo que es una buena forma de actualizarse y adquirir nuevos conocimientos.

Sprint 1: Base de datos y administración

Issues:

- Subir proyecto base al servidor.
- Contactar con el cliente: Carga inicial de datos. Necesitábamos conocer qué estructura de ficheros venía utilizando hasta ahora para poder diseñar la base de datos en función de sus necesidades.
- Crear tablas en el gestor de bases de datos MySQL.
- Documentación: Cargar la plantilla LaTeX.

Una vez que teníamos claro qué *framework* se iba a utilizar, y que habíamos creado correctamente la estructura del proyecto, el paso siguiente fue subirlo a un servidor. La opción que elegimos en este primer momento, fue la más económica, eligiendo un hosting gratuito en https://domitienda.com/hosting-ilimitado/ con las siguientes características:

- Hosting 1 Dominio.
- 250 MB de espacio en disco.
- 1 buzón de correo.
- Certificado Let's Encrypt.

- Protección permanente de aplicaciones
- Soporte 24 horas.

El cliente envió una hoja de cálculo con los datos que manejaba hasta ahora por lo que se decidió dejar para el siguiente *sprint* la carga de tablas en base de datos mientras se analizaba el documento.

Para la realización de la documentación, elegimos *LaTeX* acompañado del editor *TeXstudio* por lo que importamos la plantilla facilitada desde la universidad.

Sprint 2: Base de datos y carga inicial

Issues:

- Crear diagrama relacional y crear estructura de tablas en base de datos.
- Carga inicial de datos en la aplicación.

Los objetivos de este *sprint* son claros. Una vez analizados los datos de la hoja de cálculo facilitada por el cliente, modelar un diagrama relacional y crear la estructura de tablas en base de datos. Con esto cumplido, el siguiente paso era alimentar nuestra base de datos con datos reales. Esta carga inicial se ha hecho leyendo los datos directamente de la hoja de cálculo facilitada por el cliente.

Dado que existían pequeñas disconcordancias entre los propios datos y no teniendo la seguridad de que fuesen los datos y estructura definitiva, se decidió hacer un primer boceto de lo que sería el diagrama relacional y se empezó a preparar la estructura MVC (Modelo, Vista y Controlador) para las entidades de las que se tenía confirmación. Desde el *framework* es necesario establecer una comunicación entre el cliente y la base de datos. Esto es lo que se trató de hacer en este *sprint*.

Sprint 3: Carga de datos con una primera administración

Issues:

- Renombrar tablas de base de datos.

- Cargar en base de datos los datos facilitados en una hoja de cálculo.
- Empezar la administración de los datos.

Una vez que teníamos la estructura del proyecto, se continuó dando forma a nuestro diagrama hasta su versión definitiva. Con la base de datos ya creada, utilizando la librería *PhpSpreadsheet* se empezó a cargar los datos.

En cuanto a la administración de los datos, no fue viable por lo que lo dejamos para el siguiente *sprint*.

Sprint 4: Terminar la carga de datos y empezar la administración de las entidades

Issues:

- Terminar de cargar la información en base de datos.
- Administración de la tabla *countries*.
- Administración de la tabla *regions*.
- Administración de la tabla *fuels*.
- Administración de la tabla *technologies*.

En este *sprint* ha habido problemas a la hora de cerrar la carga inicial de todos los datos, pues había tablas cuyos datos no existían por lo que se tuvo que esperar a que el cliente los facilitara.

Sin embargo, con la mayoría de tablas con sus datos correspondientes cargados, se ha empezado a construir la administración para cada una de ellas.

El primer proceso es laborioso pues tienes que realizar los *mockups* de las pantallas que se quieren construir.

En este *sprint* se ha invertido más tiempo del planificado pues los tiempos de elaboración de *mockups*, maquetación de una cabecera y plantilla común a toda la aplicación no han sido los previstos. Esto sirvió para, a partir de ahora, ser más cuidadosos con el presupuesto de horas pues las tareas eran menos previsibles y más complejas.

Sprint 5: Terminar la administración:

Issues:

- Administración de la tabla *arcs*.
- Administración de la tabla *typelines*.
- Administración entre *arcs* y *typelines* (*arcs_typelines*).
- Administración entre *regions* y *technologies* (*regions_technologies*).
- Administración entre *regions* y *arcs* (*regions_arcs*).
- Administración de la tabla *rangerenewables y rangemeteos*.

Mientras que para las tablas en las que la administración era a través de un formulario no existía mayor problema, las tablas *rangerenewables y rangemeteos* se abastecen de un fichero con formato hoja de cálculo que el usuario sube a la aplicación. Esto ha resultado bastante problemático por el gran número de registros a insertar en la tabla dándonos problemas con la memoria disponible y el tiempo de ejecución empleado.

Otro problema detectado ha sido el tamaño máximo de fichero que podemos subir a la aplicación. Por defecto, en la configuración de *PHP* que trae *Apache*, este valor viene limitado, por lo que se ha modificado la propiedad `upload_max_filesize=100M` del fichero `php.ini`.

Al margen del problema con el tamaño del archivo, que ha quedado corregido, y tras muchas horas de pruebas, se han intentado solucionar los inconvenientes pero en este *sprint* no ha sido posible, por lo que arrastramos el desarrollo de esta funcionalidad al siguiente.

Sprint 6: Logotipo de Weblectric y documentación del proyecto:

Issues:

- Logotipo para la aplicación: *Weblectric*.
- Documentación: 2 Objetivos del proyecto.
- Documentación: 3 Conceptos teóricos.
- Documentación: 4 Técnicas y herramientas.

- Documentación: 5 Aspectos relevantes del desarrollo del proyecto.
- Documentación: 6 Trabajos relacionados.

Aprovechando el parón de navidades, se ha decidido aprovechar para diseñar un logotipo para la aplicación y formalizar por escrito en la documentación del proyecto todas aquellas cosas que estaban en el aire.

Escribir el apartado «Objetivos del proyecto»(2) y «Técnicas y herramientas»(4) lo cual no nos ha llevado demasiado tiempo porque era material ya trabajado y comentado previamente, en cambio, si que ha requerido de una labor más a fondo los apartados «Conceptos teóricos»(3), «Aspectos relevantes del desarrollo del proyecto»(5) y «Trabajos relacionados»(6); pues ha requerido labor de investigación sobre algún algoritmo de optimización para el apartado número 3 y de competencias profesionales ya existentes en el mercado para el apartado número 6.

Sprint 7: Anexos y terminar administración mediante hojas de cálculo

Issues:

- Anexos: A Plan de proyecto.
- Anexos: B Requisitos.
- Anexos: C Diseño.
- Anexos: D Manual del programador.
- Terminar la administración mediante hojas de cálculo.

Retomando el problema de tratamiento de los excels en los que no se lograba completar la transacción debido al gran volumen de datos, se decidió plantear la función de procesamiento de forma diferente y se logró optimizar el código por lo que el problema llegó a su fin. Además, quitando el *debug* de la aplicación, hacía que el funcionamiento fuese más fluido recortando varios minutos de procesamiento.

Por otro lado se empezaron a escribir los anexos, lugar donde se ha ido detallando cada uno de los grandes bloques que lo forman.

Sprint 8: Corregir Anexos y añadir extras a la aplicación

Issues:

- Añadir migas de pan.
- Funcionalidad a la pantalla de objetivos.
- Administración de usuarios.
- Funcionalidad a la pantalla de simulación.
- Anexos: Explicar y actualizar diagramas entidad-relación y relacional.
- Anexos: Añadir el diagrama de despliegue al anexo de diseño.
- Anexos: Crear diagramas de secuencia.

Viendo que con el cambio de la estructura de la aplicación producido por la petición del cliente, el flujo de las pantallas quedaba poco intuitivo y usable, se vió conveniente añadir migas de pan para facilitar la navegación entre las pantallas de la aplicación.

Además se ha introducido una administración de usuarios con dos roles diferentes. De esta manera el acceso a funcionalidades como restablecer la base de datos y crear y eliminar usuarios, ha quedado restringida a los usuarios con rol administrador.

En la pantalla de objetivos que hasta ahora estaba sin funcionalidad, se guardan los valores de los *checkbox* en sesión para que en un futuro se puedan integrar con la lógica de la aplicación.

Por último, desde la pantalla de simulación se genera un archivo *.zip* con parte de los archivos necesarios para la ejecución del algoritmo del cliente.

En cuanto a los anexos, se han ido corrigiendo detalles y explicando más en profundidad diferentes secciones como son los requisitos y casos de uso y el diseño.

A.3. Estudio de viabilidad

En esta sección se hablará del presupuesto económico equivalente al desarrollo del proyecto y de su viabilidad legal.

Viabilidad económica

A continuación, se va a proceder a hacer un estudio de viabilidad económica del proyecto. Se van a justificar todos los gastos como si el desarrollo formara parte del mundo laboral en lugar del plan de estudios del grado.

A través de *ZenHub* podemos asignar el tiempo que estimamos en la planificación del *sprint* para el desarrollo de una *issue*. En nuestro caso, salen un total de: 330 horas aproximadas. Suponiendo que la hora de trabajo del desarrollador está pagada a 8 €, tenemos un total de:

$$8€/\text{h} \times 375\text{h} = 3000€$$

Además se tendrá en cuenta el trabajo llevado a cabo por los tutores del proyecto. Sumando al valor anterior 1 hora de reunión por *sprint* y 3 horas extras para revisar documentación, plataforma y correos electrónicos con consultas procedentes del desarrollador y cobrando a 10 €/h, hacen un total de:

$$8\textit{sprints} \times 10€/\text{h} \times 4\text{h} = 320€$$

De momento sumamos un total de 3320 € que si le aplicamos alrededor del 30 % de seguridad social, se queda en un total de:

$$3320 + (3320 \times 0{,}30) = 4316€$$

Por último, destacar el coste material. En la parte software no se tienen gastos pues las aplicaciones y herramientas son todas gratuitas. En la parte hardware, podríamos valorar:

- Ordenador portátil: 600 €.
- Monitor: 150 €.
- Teclado y ratón: 25 €.

Sumando el todo nos da un total de 775 € y según la agencia tributaria [2], los equipos para procesos de información tienen una vida útil de 4 años

[2]Tablas de amortización en 2018: https://bit.ly/2ND4vCw

(35.040 horas) y como el proyecto ha tenido una duración de 3 meses y medio, deja un total de:

$$\frac{775€}{12 \text{ Meses} \times 4 \text{ Años}} \times 3{,}5 \text{ Meses} = 28,26€$$

Por último, sumar el gasto producido por el servidor donde hemos tenido alojada la web. Aunque gran parte de la duración del proyecto se ha tenido en un *hosting* gratuito, en la recta final y por causa de las limitaciones que nos ofrecía el plan, nos hemos visto obligados a alojarlo en otro plan diferente. En consecuencia, los gastos han sido:

$$\frac{36€ \text{ al año}}{12 \text{ Meses}} = 3€ \text{ al mes. Nosotros: 1 mes: } 3€ \text{ totales.}$$

Con todos los costes mencionados, el desglose e importe final se recoge en la tabla A.1:

Tabla A.1: Costes totales del proyecto

Costes	Importe
Desarrolladores	3000 €
Tutores del proyecto	320 €
Seguridad Social	996 €
Material hardware	28,26 €
Servidor	3 €
Totales	4347,26 €

Viabilidad legal

A continuación se hará un análisis de las librerías que hemos utilizado en nuestro proyecto. El análisis consistirá en buscar los tipos de licencias y después de analizar las compatibilidades, seleccionaremos la más restrictiva. En la tabla A.2 se puede ver un listado de todas las licencias.

Puesto que la licencia *GNU General Public Licence* indica que en caso de utilizar un software bajo dicha licencia, es obligatorio calificar el software distribuido de la misma manera, establecemos a *Weblectric* una licencia *GNU General Public Licence*.

Tabla A.2: Herramientas utilizadas y sus licencias

Herramienta	Licencia
CakePHP	MIT
Foundation Zurb	MIT
jQuery	MIT
Visual Studio Code	MIT
XAMPP	GNU General Public Licence
Heidi SQL	GNU General Public Licence

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se presentarán los requisitos y casos de uso que han dado lugar a la funcionalidad de la aplicación. Esta información irá acompañada del modelo relacional y de los diagramas entidad relación.

B.2. Objetivos generales

Los principales objetivos del proyecto son:

- Dar soporte a través de una aplicación web a los resultados obtenidos tras la ejecución del algoritmo de optimización que posee el cliente.
- Lograr una aplicación intuitiva sin problemas de usabilidad para el usuario.
- Crear una administración que permita crear, editar y eliminar cualquiera de las entidades mediante formularios.
- Para las entidades con registros en instantes de tiempo, dar posibilidad al usuario de subir una hoja de cálculo para actualizar los datos.
- Innovar en como el tribunal de proyectos puede hacer una batería de pruebas sobre una aplicación fuertemente basada en una interfaz de usuario. Para ello se implementará un botón de restauración de la base de datos.

- Construir una pantalla desde la que el cliente pueda ejecutar el algoritmo.

B.3. Catalogo de requisitos

Los requisitos que debe satisfacer la aplicación son:

Requisitos funcionales

- **RF-1:** Administrar mediante formularios las entidades correspondientes a:
  - Arcos entre dos regiones.
  - Tipo de conexión de ese arco.
  - Relación entre el arco y el tipo de arco.
  - Países.
  - Combustibles.
  - Tecnologías
  - Regiones.
  - Relación entre combustibles y tecnologías.
  - Relación entre regiones y tecnologías.
- **RF-2:** Administración para las entidades correspondientes a través de una hoja de cálculo en la que se guarda un valor concreto para cada una de las horas del año en una región determinada:
  - Datos acerca del clima.
  - Datos acerca de fuentes renovables.
  - Datos de demanda.
- **RF-3:** El usuario de la aplicación podrá tener acceso a la visualización de cada una de las entidades. Se realizará a través de tablas paginadas con posibilidad de acceder a un detalle más exhaustivo pulsando en cada elemento.
- **RF-4:** El usuario podrá exportar una hoja de cálculo con los datos de las entidades cuya administración funciona con la subida de un fichero. Esta misma hoja de cálculo, podrá modificarla y subirla para actualizar los datos.

- **RF-5:** El usuario tras realizar pruebas de funcionalidad sobre la aplicación, podrá restablecer la base de datos a un punto de partida pulsando un botón situado en la cabecera.
- **RF-6:** Existirá un control de usuarios en el que haya usuarios con dos posibles roles: administrador y participante.

Requisitos no funcionales

- **RNF-1:** La aplicación tiene que ser intuitiva y fácil de usar de cara al usuario.
- **RNF-2:** El rendimiento de la aplicación tiene que ser lo más óptimo posible. La navegación entre las diferentes pantallas tiene que ser fluida.
- **RNF-3:** La aplicación deberá funcionar con normalidad en todos los navegadores.

B.4. Especificación de requisitos

En este apartado se describirán los actores y casos de uso asociados a cada uno de los requisitos definidos anteriormente.

Actores

Actores presentes en la aplicación:

- Administrador de la aplicación: Realiza la primera carga de datos iniciales y prepara la consulta *sql* para poder restablecer la base de datos.
- Usuario: Interactúa con la aplicación web.
  - Usuario con rol administrador: Tiene acceso al control de usuarios y al botón para restablecer la base de datos.
  - Usuario con rol participante: Interactúa con la aplicación web sin tener acceso a ninguna funcionalidad especial.

Diagrama de casos de uso

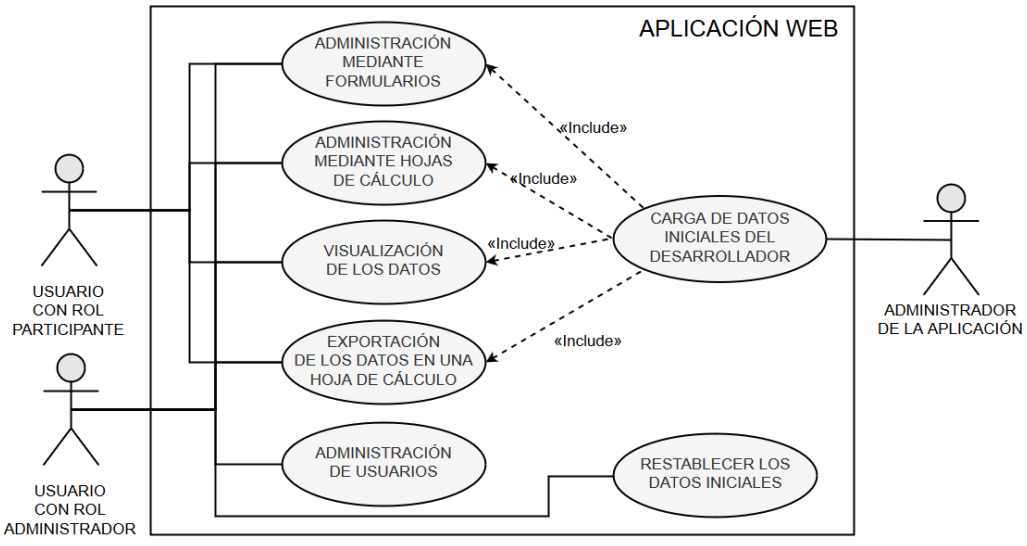

Figura B.1: Diagrama de casos de uso de *Weblectric*.

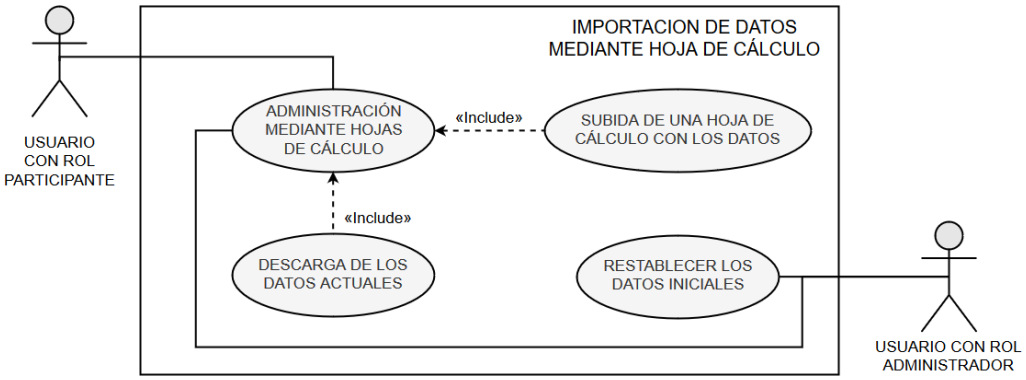

Figura B.2: Diagrama del casos de uso 2: Administración mediante hojas de cálculo. B.4.

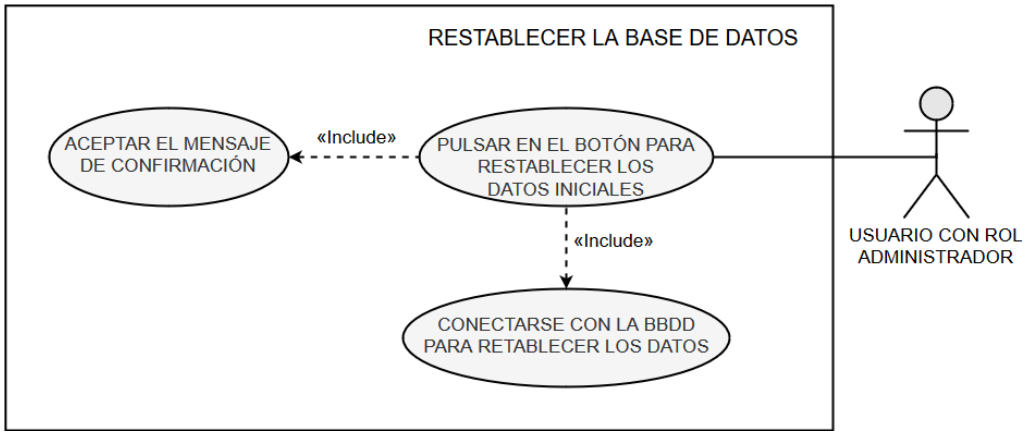

Figura B.3: Diagrama del casos de uso 5: Restablecer base de datos. **??**.

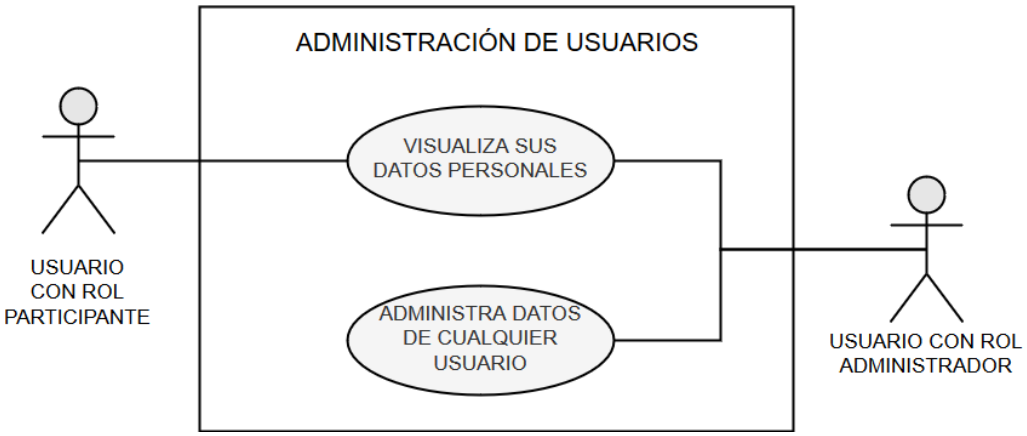

Figura B.4: Diagrama del casos de uso 6: Administración de usuarios. B.4.

Caso de uso 1	**Administración mediante formularios.**
Versión	1.0
Requisitos	RF-1
Descripción	El usuario podrá crear/editar/borrar elementos de las entidades.
Precondiciones	1. Base de datos iniciada. 2. Carga inicial de datos realizada.
Acciones	En función de la labor que desee realizar, el usuario interaccionará con los botones de crear/editar/borrar.
Postcondiciones	La aplicación emitirá un mensaje de confirmación en caso de realizarse una baja y se redireccionará al listado de todos los datos y el usuario podrá ver el resultado de la acción.
Excepciones	1. No haya datos para una entidad. 2. Eliminar algún elemento relacionado con otra entidad.
Importancia	Alta

Tabla B.1: Caso de uso 1 - Administración mediante formularios.

Caso de uso 2	**Administración mediante hojas de cálculo.**
Versión	1.0
Requisitos	RF-2
Descripción	El usuario podrá actualizar los datos subiendo a la aplicación una hoja de cálculo.
Precondiciones	1. Base de datos iniciada. 2. Carga inicial de datos realizada.
Acciones	1. En usuario deberá entrar en la pantalla correspondiente a la entidad que quiere modificar. 2. Se deberá descargar la plantilla. 3. Subirá un excel con los datos que desee.
Postcondiciones	La aplicación se redireccionará al listado de todos los datos y el usuario podrá ver la fecha de la última subida.
Excepciones	Introducir algún elemento extraño en la hoja de cálculo que agote los recursos de procesado de la máquina produciendo un error.
Importancia	Alta

Tabla B.2: Caso de uso 2 - Administración mediante hojas de cálculo.

Caso de uso 3	**Visualización de datos en tablas paginadas.**
Versión	1.0
Requisitos	RF-3
Descripción	El usuario podrá visualizar un conjunto de datos en tablas paginadas.
Precondiciones	1. Base de datos iniciada. 2. Carga inicial de datos realizada. 3. Realización de una consulta.
Acciones	1. En usuario deberá de entrar en la pantalla correspondiente a la entidad que quiere visualizar. 2. Si lo desea, podrá modificar la consulta utilizando los filtros del buscador. 3. Pulsar el botón de búsqueda para obtener los nuevos resultados. 4. Si se desea una información más exhaustiva, se podrá pulsar en el botón de visualización.
Postcondiciones	Pulsando en el botón de buscar, se actualizará la página con los nuevos resultados.
Excepciones	Introducir filtros que no representen ningún dato.
Importancia	Media

Tabla B.3: Caso de uso 3 - Visualización de datos en tablas paginadas.

Caso de uso 4	**Exportar datos en hoja de cálculo.**
Versión	1.0
Requisitos	RF-4
Descripción	El usuario podrá exportar una hoja de cálculo con los datos de las entidades cuya administración funciona con la subida de un fichero.
Precondiciones	1. Base de datos iniciada. 2. Carga inicial de datos realizada.
Acciones	1. En usuario deberá de entrar en la pantalla correspondiente a la entidad que quiere exportar. 2. Pulsar sobre el icono exportar.
Postcondiciones	Se descargará un fichero con los datos actuales en base de datos.
Excepciones	Entidad sin datos cagados. En ese caso los datos de la exportación estarán vacíos.
Importancia	Media

Tabla B.4: Caso de uso 4 - Exportar datos en hoja de cálculo.

Caso de uso 6	**Administración de usuarios.**
Versión	1.0
Requisitos	RF-6
Descripción	Existirá la posibilidad de administrar los usuarios de la aplicación. Los usuarios con rol administrador podrán visualizar y modificar los datos de cualquier usuario mientras que un usuario con rol participante solo podrá administrar los suyos propios.
Precondiciones	
Acciones	1. Usuario administrador accede a la pantalla de usuarios. 2. De los resultados listados en la tabla, elige el usuario a modificar. 3. Mediante la acción correspondiente, crea, edita y elimina un usuario.
Postcondiciones	Redirección a la pantalla principal de los usuarios.
Excepciones	Error al ejecutar la acción.
Importancia	Media

Tabla B.5: Caso de uso 6 - Administración de usuarios.

Apéndice C

Especificación de diseño

C.1. Introducción

A continuación se presentan los diseños elaborados para poder realizar los objetivos anteriores. Se ha incluido el diagrama de despliegue, el diseño de datos, el diseño procedimental y el diseño arquitectónico.

C.2. Diagrama de despliegue

Como se puede ver reflejado en la figura C.1, la aplicación cuenta con 3 bloques claramente diferenciados:

- El navegador personal del cliente donde se ejecuta la aplicación.
- Un servidor web con *Apache* instalado en el que se despliega un proyecto en *CakePHP*.
- Un servidor de base de datos donde se aloja una base de datos relacional de *MySQL*.

C.3. Diseño de datos

La estructura de datos presentada en este proyecto se recoge en una base de datos relacional *MySQL*. En un primer momento se construyó el diagrama entidad relación a través del cual se obtuvo el diagrama relacional. En las

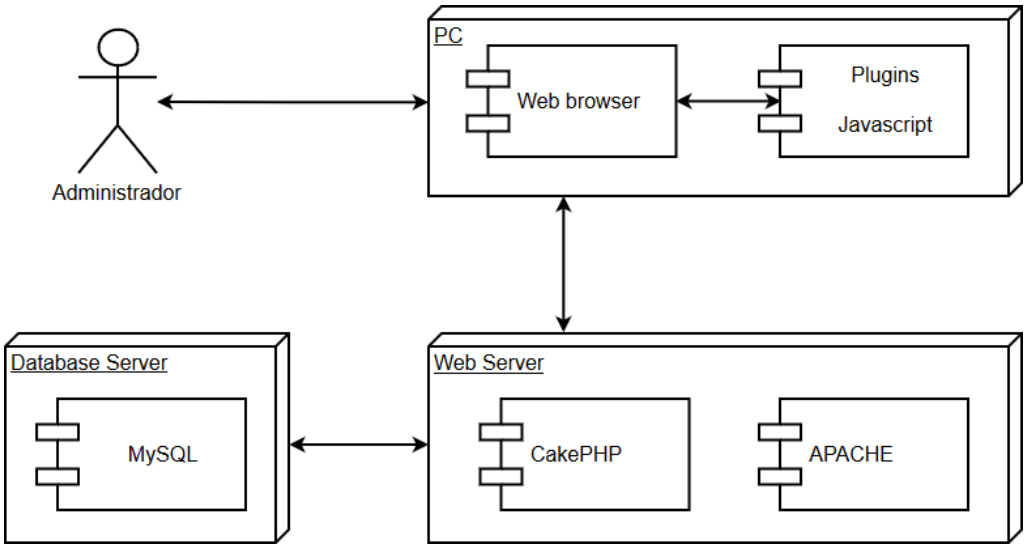

Figura C.1: Diagrama de despliegue de la aplicación.

figuras C.2 y C.3 se presentan los modelos entidad relación y relacional respectivamente.

A continuación se recogerá en un diccionario de datos cada una de las entidades con sus correspondientes campos y las unidades en función de si es relevante:

- *countries*: Los países presentes en la instalación.
  - *id*: Identificador del país.
  - *name*: Nombre del país.
- *regions*: Cada una de las regiones pertenecientes a un país.
  - *id*: Identificador de la región.
  - *id_country*: Identificador del country al que pertenece.
  - *dem_for*: Margen de reserva por error en el pronostico de demanda.
  - *ren_for*: Margen de reserva por error en el pronostico de generación renovable.
- *arcs*: Comunicación entre dos regiones.

- *id*: Identificador del arco.
- *id_region_1*: Identificador de la región de origen.
- *id_region_2*: Identificador de la región de destino.
- *distance*: Distancia entre 2 regiones (Km).

- *fuels*: Tipos de combustibles.
  - *id*: Identificador del combustible.
  - *name*: Nombre del combustible.
  - *fue_cos*: Coste por unidad de combustible importado. ($/MMBTU).
  - *production*: Reserva nacional de combustible. (MMBTU).
- *technologies*: Tipos de tecnologías renovables y no renovables.
  - *id*: Identificador de la tecnología.
  - *name*: Nombre de la tecnologia.
  - *renewable*: Si es una tecnología renovable $(0 - 1)$
  - *wat_wit*: Desperdicio de agua por generación de un *megawhatt*. (m^3/MWh).
  - *genco_pri*: Precio de mercado por *megawhatt* $(\$/MWh)$.
  - *cap*: Capacidad de planta (MW).
  - *new_cap_cos*: Costo por *megawhatt* debido a la instalación. $(\$/MW)$.
  - *man_cos*: Costo por *megawhatt* debido al mantenimiento de plantas viejas $(\$/MWh)$.
  - *man_cos_new_cap*: Costo por *megawhatt* debido al mantenimiento de plantas nuevas. $(\$/MWh)$.
  - *gen_cos*: Costo por *megawhatt-hora* generado para plantas viejas. $(\$/MWh)$.
  - *gen_cos_new_cap*: Costo por *megawhatt-hora* generado para plantas nuevas. $(\$/MWh)$.
  - *life_time*: Vida util de las plantas de generación.
  - *ghg_emi*: Toneladas de CO_2 producidas por *megawhatt-hora*. (CO_2/MWh).
  - *inv_cap_emp*: Empleo en la construcción de las plantas de generación.

- *man_cap_emp*: Empleo en el mantenimiento de las plantas de generación.
- *dec_cam_emp*: Empleo en el desmantelamiento de plantas de generación.
- *om_cap_emp*: Empleo en la operación de plantas de generación.
- *fue_cap_emp*: Empleo en la producción, procesamiento y transporte de combustibles.
- *wat_con*: Consumo de agua por generación de un *megawhatt* (m^3/MWh).

- *typelines*: Tipo de conexión de los arcos entre dos regiones.
  - *id*: Identificador del tipo de linea.
  - *lin_cap*: Capacidad de transmisión de linea.
  - *tension*: Tensión de la linea. (MW)
- *arcs_typelines*: Relación entre el arco y el tipo de arco.
  - *id_arc*: Identificador del arco.
  - *id_typeline*: Identificador del tipo de linea.
  - *num_lines*: Número de lineas de cada tipo.
- *fuels_technologies*: Relación entre el combustible y la tecnología.
  - *id_fuel*: Identificador del combustible.
  - *id_technology*: Identificador de la tecnología.
  - *perc_con*: Porcentaje de aportación de combustibles para generación. $(0 - 1)$
  - *fue_con*: Consumo de combustible necesario para generar un *megawhatt* ($MMBTU/MWh$)
- *regions_technologies*: Relación entre la región y qué tecnología hay presente.
  - *id_region*: Identificador de la región.
  - *id_technology*: Identificador de la tecnología.
  - *power*: Potencia instalada en cada región. (MW)
  - *cap_ava*: Capacidad disponible por región y por tecnología. (MW)

- *rangedemands*: Datos de demanda en un instante de tiempo concreto.
  - *id_region*: Identificador de la región
  - *start*: Hora de inicio
  - *end*: Hora de fin
  - *demand*: Demanda por registro de transmisión. (MWh).
- *rangemeteos*: Datos meteorológicos en un instante de tiempo concreto.
  - *id_region*: Identificador de la región
  - *start*: Hora de inicio
  - *end*: Hora de fin
  - *temp*: Temperatura promedio de cada región. (^oC)
- *rangerenowables*: Datos de fuentes renovables en un instante de tiempo concreto.
  - *id_region*: Identificador de la región.
  - *id_technology*: Identificador de la tecnología.
  - *start*: Hora de inicio.
  - *end*: Hora de fin.
  - *gen_ava*: Porcentaje de capacidad de generación $(0 - 1)$

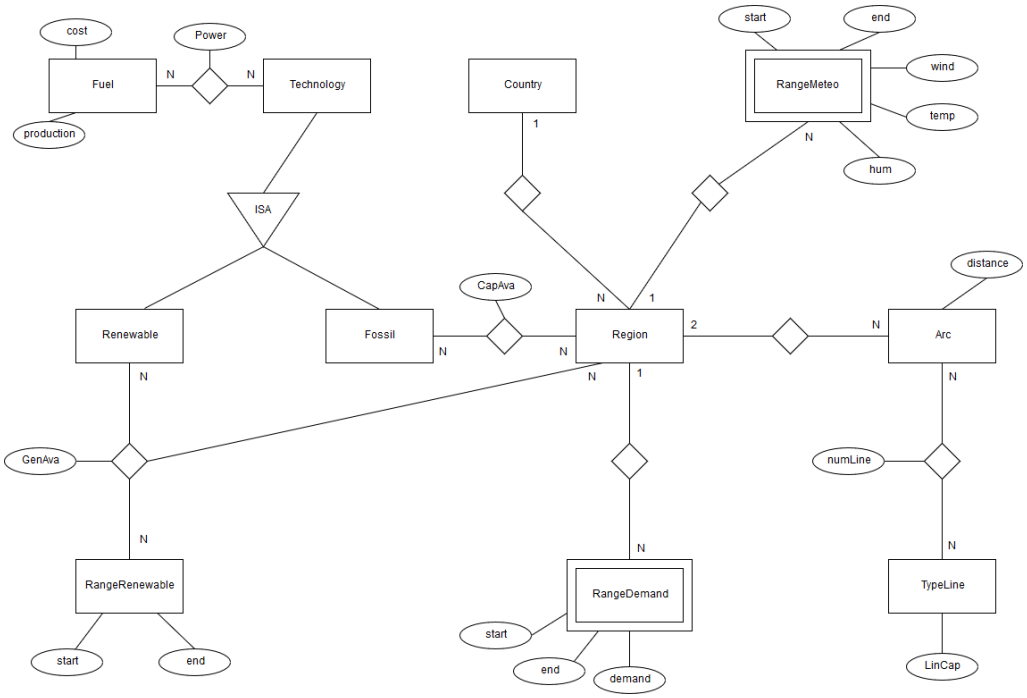

Figura C.2: Diagrama entidad relación.

C.4. Diseño procedimental

En esta sección se realizará un diagrama de secuencia del caso de uso 1 que representa la actualización de datos mediante formularios. Concretamente, se representará la edición de una región. Podemos ver el diagrama de secuencia en la figura C.4.

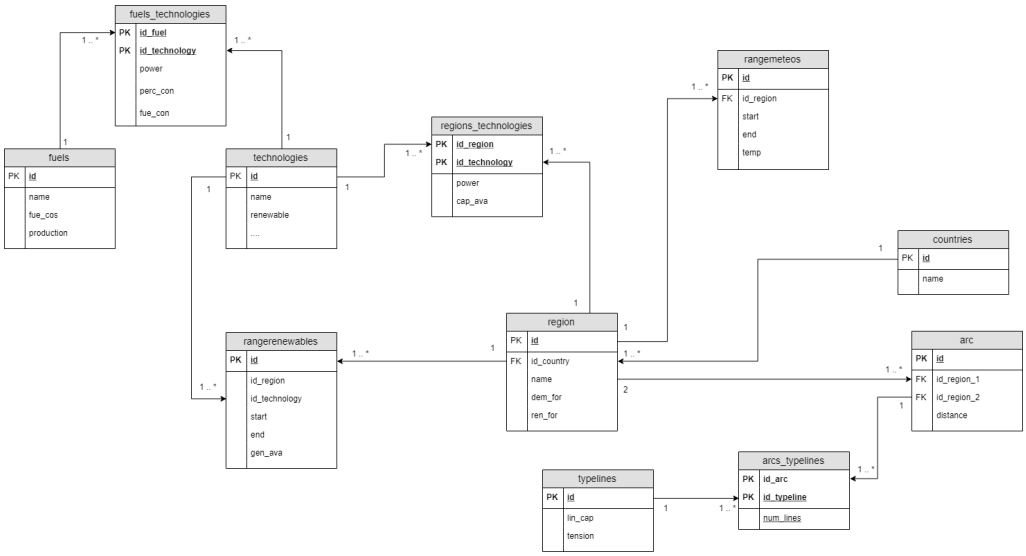

Figura C.3: Diagrama relacional.

C.5. Diseño arquitectónico

En este apartado se va a comentar como está diseñada la aplicación.

Para el diseño de *Weblectric* se ha seguido un patrón Modelo-Vista-Controlador C.5. Mediante este patrón, la aplicación queda completamente estructurada en tres secciones:

- *Modelo:* Se destinan a esta parte todas las funciones de interacción directa con la base de datos.
- *Vista:* Aquí se guardan las pantallas que interactúan con el cliente.
- *Controlador:* Aquí se encuentra la lógica de la aplicación. Hace de unión entre la vista y el modelo.

En la tabla C.1 podemos ver qué directorio corresponde en nuestro proyecto a cada una de estas tres grandes secciones.

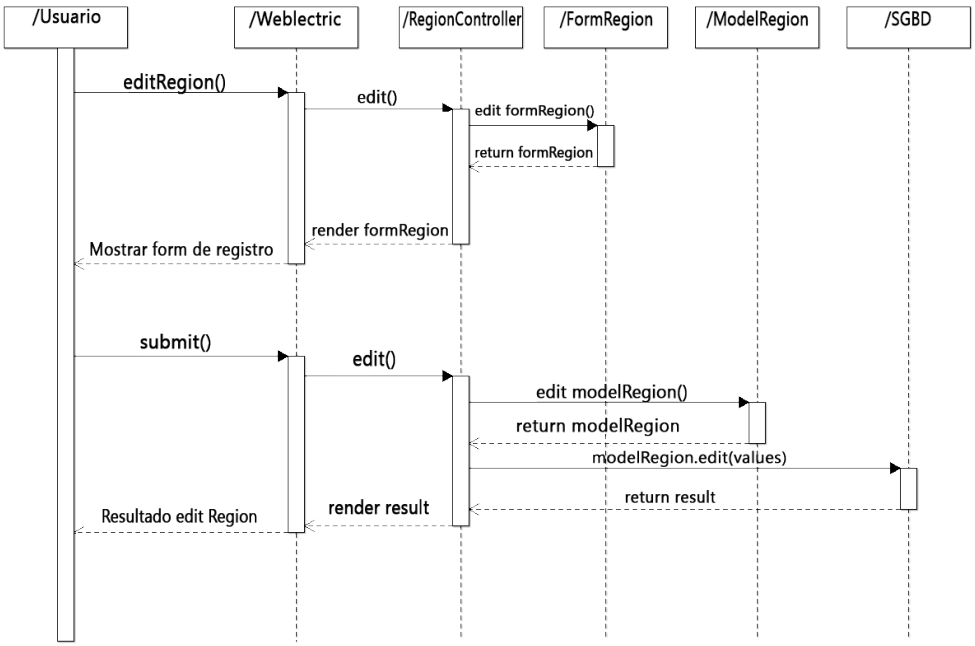

Figura C.4: Diagrama secuencial del caso de uso 1: Administración mediante formularios. B.4.

Tabla C.1: Estructura MVC en *Weblectric*.

Patrón MVC	Directorio en *Weblectric*
Modelos	*models*
Vistas	*templates*
Controladores	*controllers*

Una vez conocida la estructura de la aplicación, mostraremos los pasos seguidos para construir las vistas. Dividiremos la clasificación en dos grupos: la idea inicial elaborada a través de unos *mockups* y el resultado final.

Mockups

El primer paso fue diseñar una plantilla (*layout*). Esta plantilla, que se puede ver en la figura C.6 es común a todas las vistas.

Posteriormente, pasamos a diseñar la pantalla principal C.7. Aquí se iban a listar todas las entidades para que el usuario pudiese acceder a

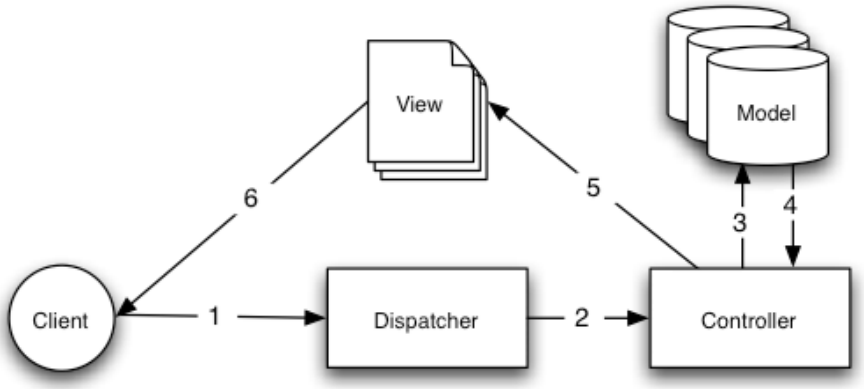

Figura C.5: Patrón Modelo-Vista-Controlador. [1]

administrarlas.

Como se puede ver en la figura C.7, al pasar el ratón por encima, aparecerá una pequeña animación diferenciando cada una de las entidades. La siguiente pantalla que se va a presentar es la administración. Se tienen dos tipos de administraciones, las que son a partir de formularios y las que son mediante la subida de una hoja de cálculo. Pulsando en una de las entidades, nos iremos a su pantalla principal, a cada entidad la que le corresponda.

Administración mediante formularios

En la figura C.8, podemos ver la pantalla de administración de las entidades que utilizan formularios. En esta vista podremos hacer las siguientes acciones:

- Crear un nuevo elemento: C.9.
- Editar un elemento existente: C.10.
- Visualizar detalles de un elemento existente: C.11.
- Eliminar un elemento: C.12.

Administración mediante hojas de cálculo

Como ya hemos comentado, hay algunas entidades que se administran a través de hojas de cálculo. En la figura C.13 encontramos un ejemplo.

Además, existe la posibilidad de poder descargar una plantilla con todos los datos actuales, podemos modificar los datos y subir el fichero.

Resultado final

Una vez presentados los mockups, comentaremos los cambios que se han ido produciendo a raíz de proposiciones del cliente y mejoras personales que hemos ido realizando durante el desarrollo.

La plantilla de la aplicación ha sufrido modificaciones pues además de añadir un logotipo personalizado para *Weblectric*, se ha tenido que incluir una nueva funcionalidad, un botón para restablecer la base de datos a su situación inicial. Estos cambios se contemplan en la figura C.14.

Puesto que las etiquetas de los campos de los formularios no tienen un nombre representativo, se ha decidido añadir un título a cada una de las etiquetas del formulario con la finalidad de aportar información acerca del significado del campo. Además, como resultaba de interés conocer qué usuario ha iniciado sesión en la aplicación, se creó en la cabecera una nueva sección. Por último, para facilitar la navegación por la aplicación, se han creado migas de pan. Todos estos resultados pueden verse en la figura C.25.

Sin embargo, el cambio más significativo viene dado en la pantalla principal. Una vez enseñado el resultado al cliente, nos mandó reestructurar la forma en que distribuimos cada una de las entidades. Actualmente, ya no se representan así, sino que existen diferentes agrupaciones, cada una con su color y animación identificativos.

La estructura de la aplicación, con los cambios aplicados, queda de la siguiente manera:

- Un primer grupo con tres pestañas C.15:
  - Countries data: C.16.
    - Country.
    - Renewable source.
    - Climate.
    - Current System.
  - Technologies:C.17.
    - Generation technologies.
    - Types of lines.
    - Fuels.
  - Simulation: C.18.
    - Objectives.

- Download.

A continuación, siguiendo con el mismo criterio que en el apartado de los *mockups*, mostraremos las imágenes clasificando los dos tipos de administraciones

Administración mediante formularios

Cogiendo como ejemplo la entidad "technologies", los resultados son:

- Pantalla principal de esa entidad C.19.
- Crear una nueva tecnología: C.20.
- Editar una tecnología: C.21.
- Visualizar detalles de una tecnología: C.22.
- Eliminar una tecnología: C.23.

Administración mediante hojas de cálculo

En la figura C.24 encontramos un ejemplo de como se administran los datos climáticos.

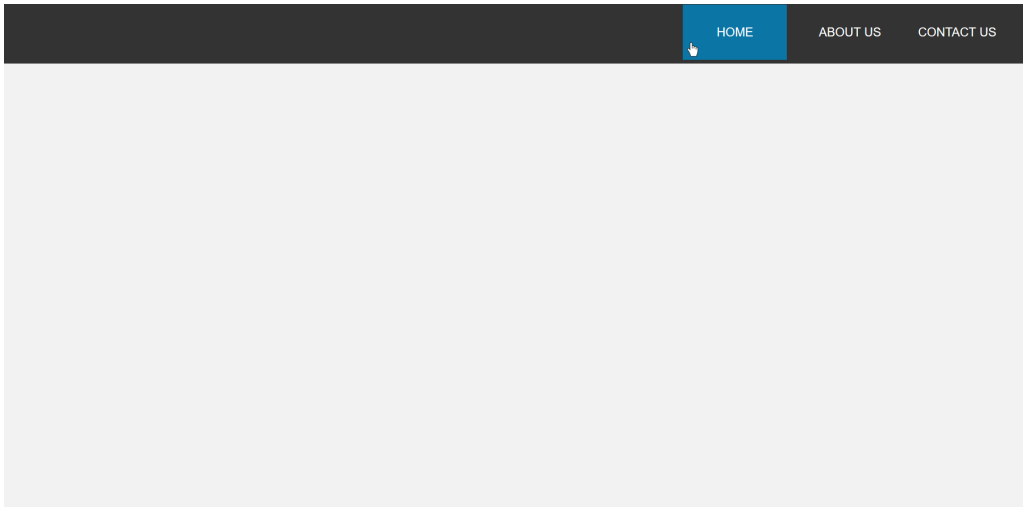

Figura C.6: Mockup: Plantilla común a todas las vistas.

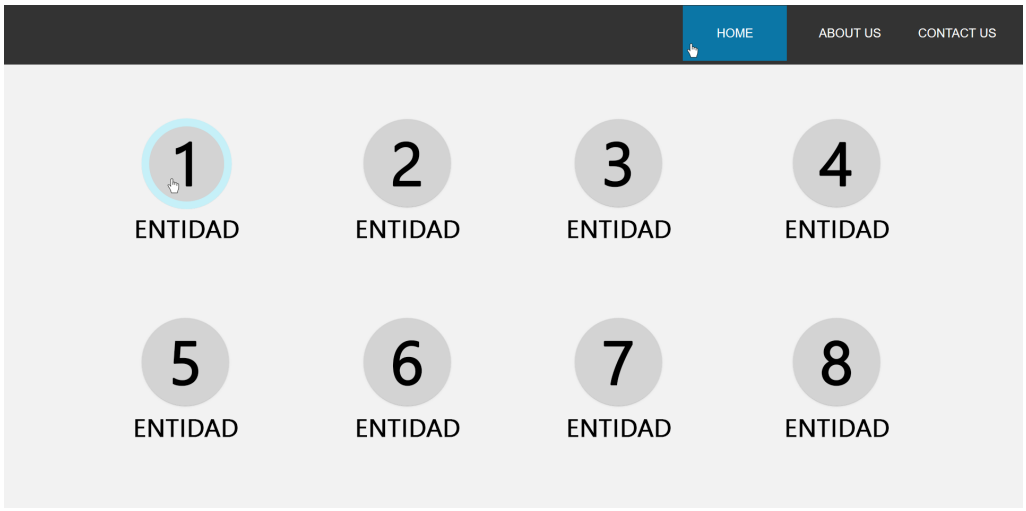

Figura C.7: Mockup: Pantalla principal.

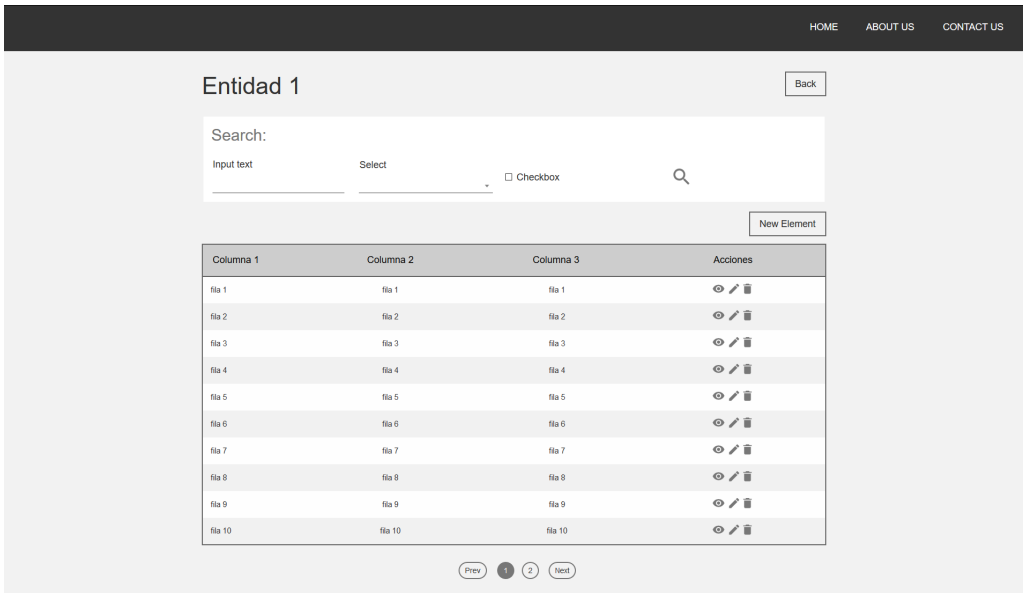

Figura C.8: Mockup: Pantalla principal de la entidad 1.

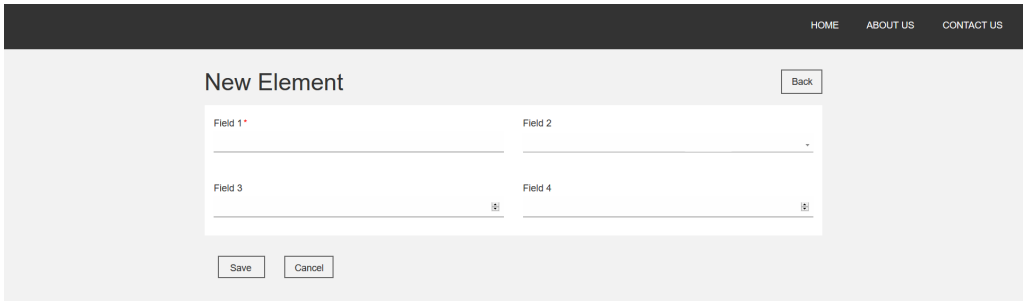

Figura C.9: Mockup: Alta de un elemento.

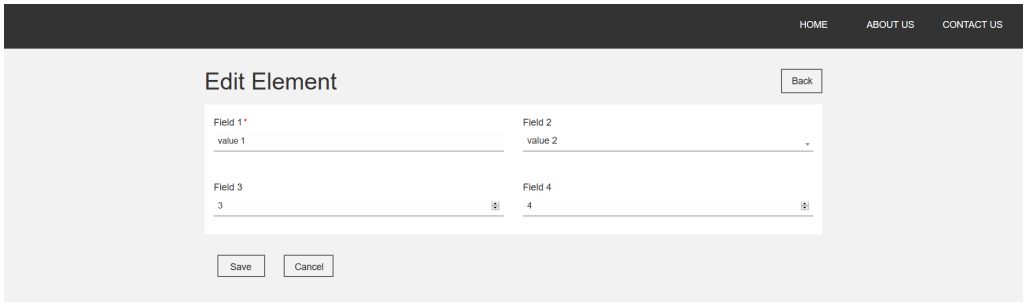

Figura C.10: Mockup: Edición de un elemento.

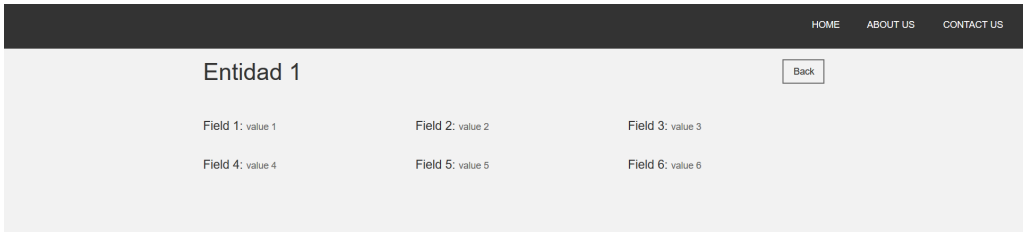

Figura C.11: Mockup: Visualización completa de un elemento.

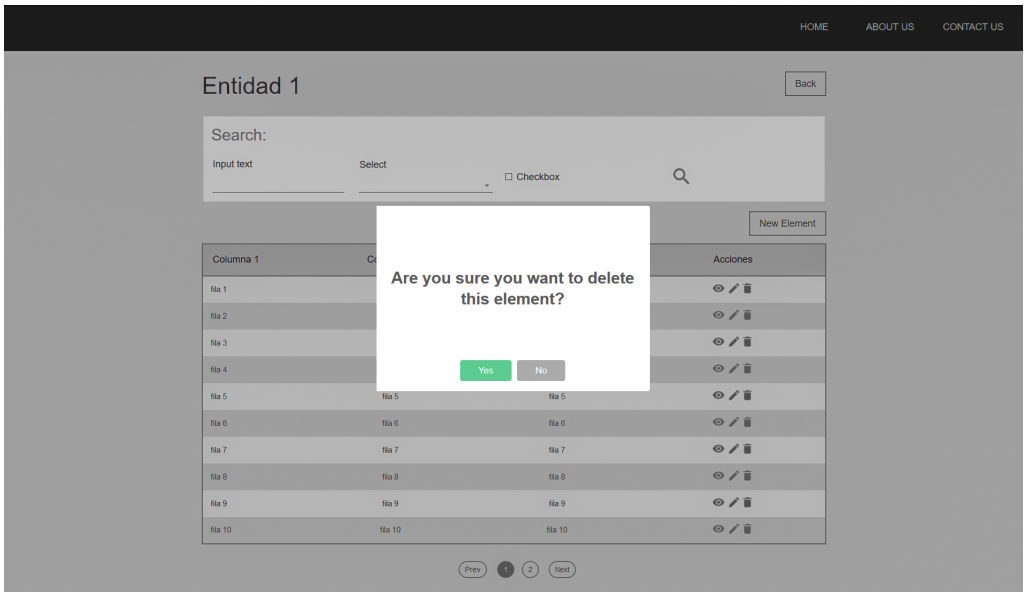

Figura C.12: Mockup: Mensaje de confirmación al eliminar un elemento.

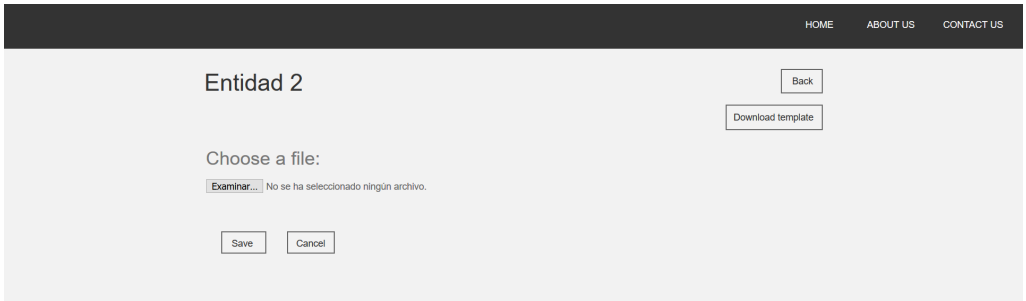

Figura C.13: Mockup: Administración de entidades mediante hoja de cálculo.

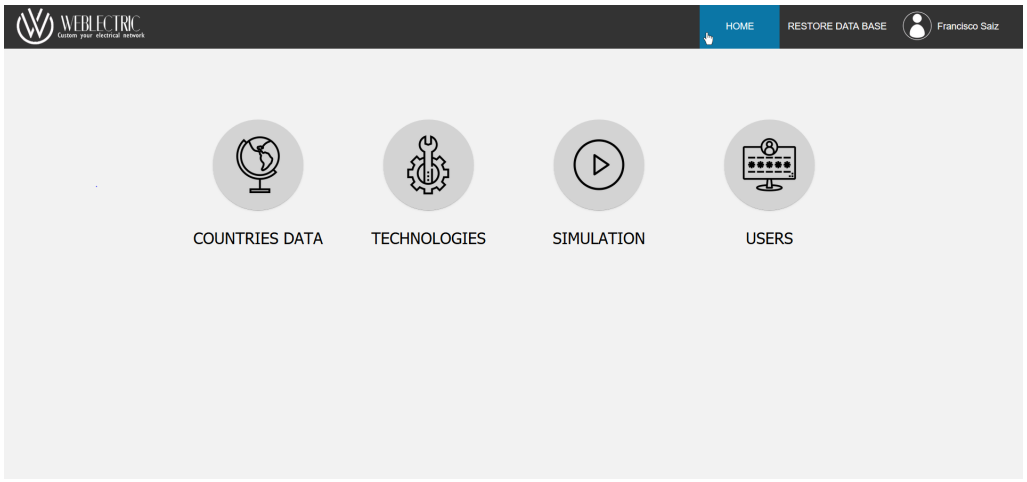

Figura C.14: Resultado final: Plantilla común a todas las vistas.

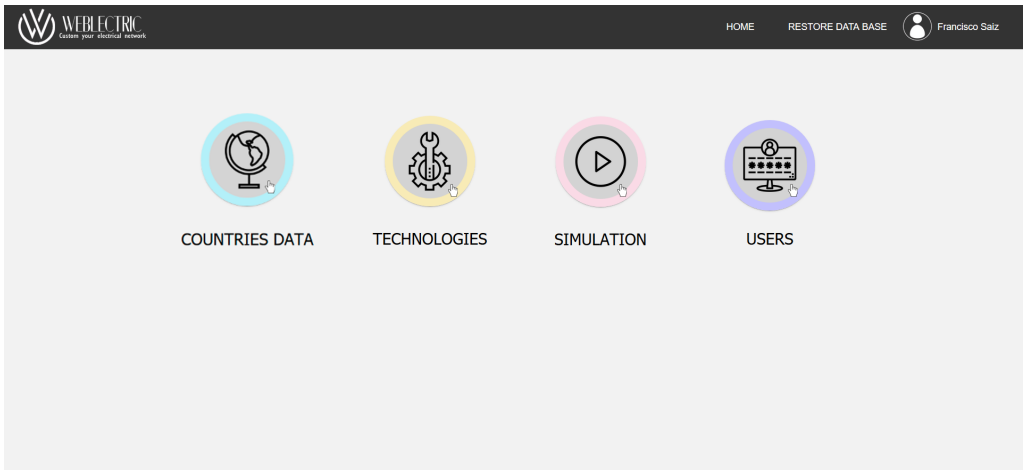

Figura C.15: Resultado final: Pantalla principal.

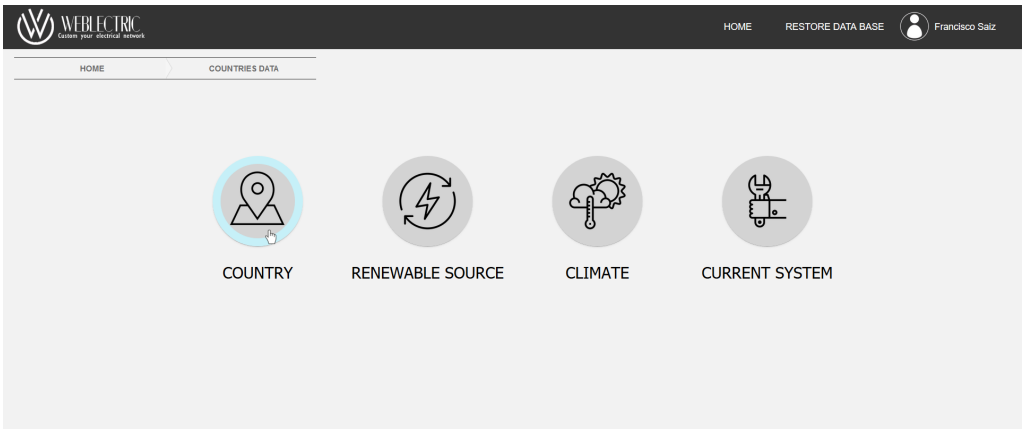

Figura C.16: Resultado final: Pantalla países.

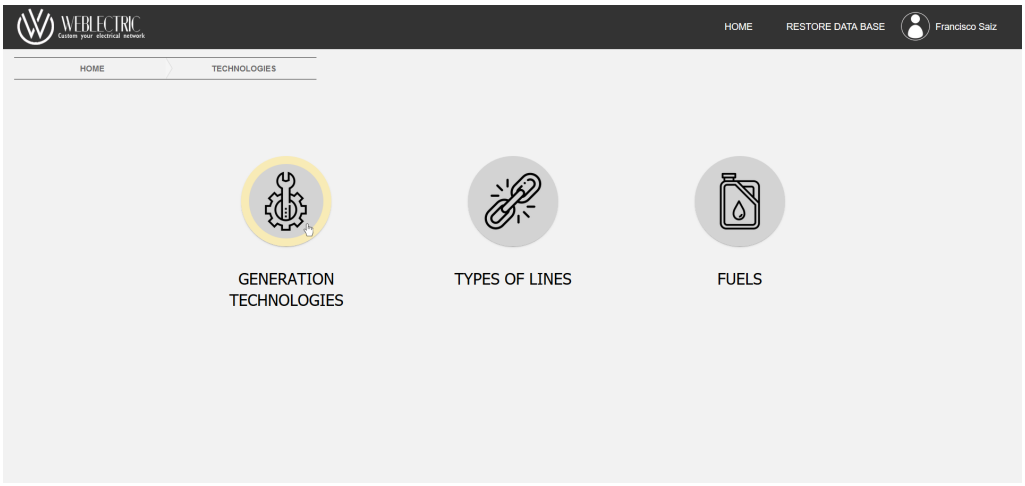

Figura C.17: Resultado final: Pantalla tecnologías.

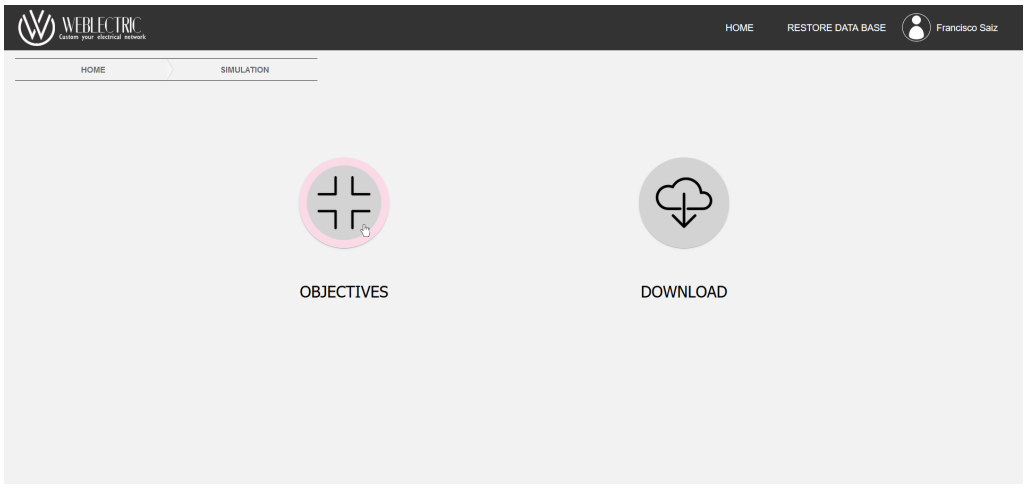

Figura C.18: Resultado final: Pantalla simulación.

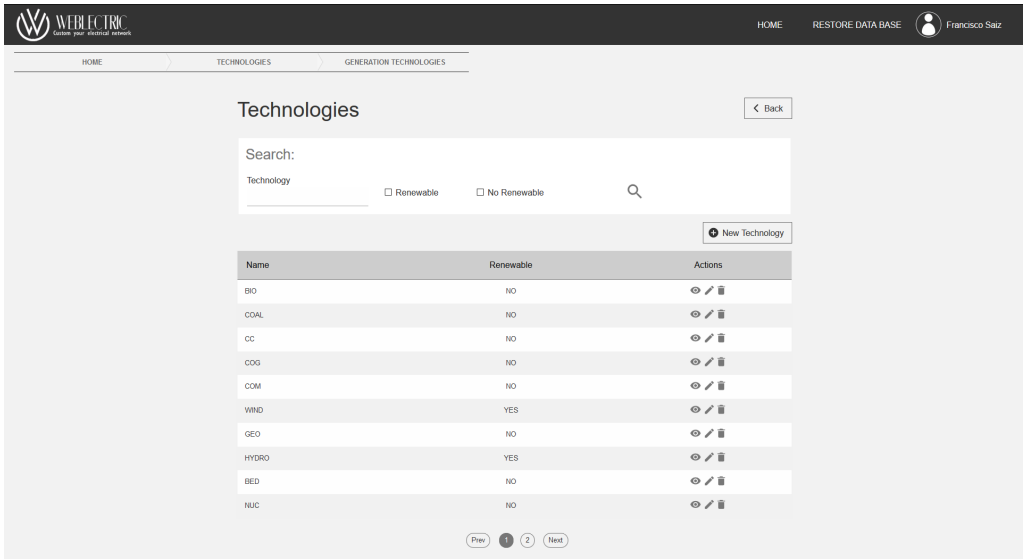

Figura C.19: Resultado final: Pantalla principal de la entidad Technology.

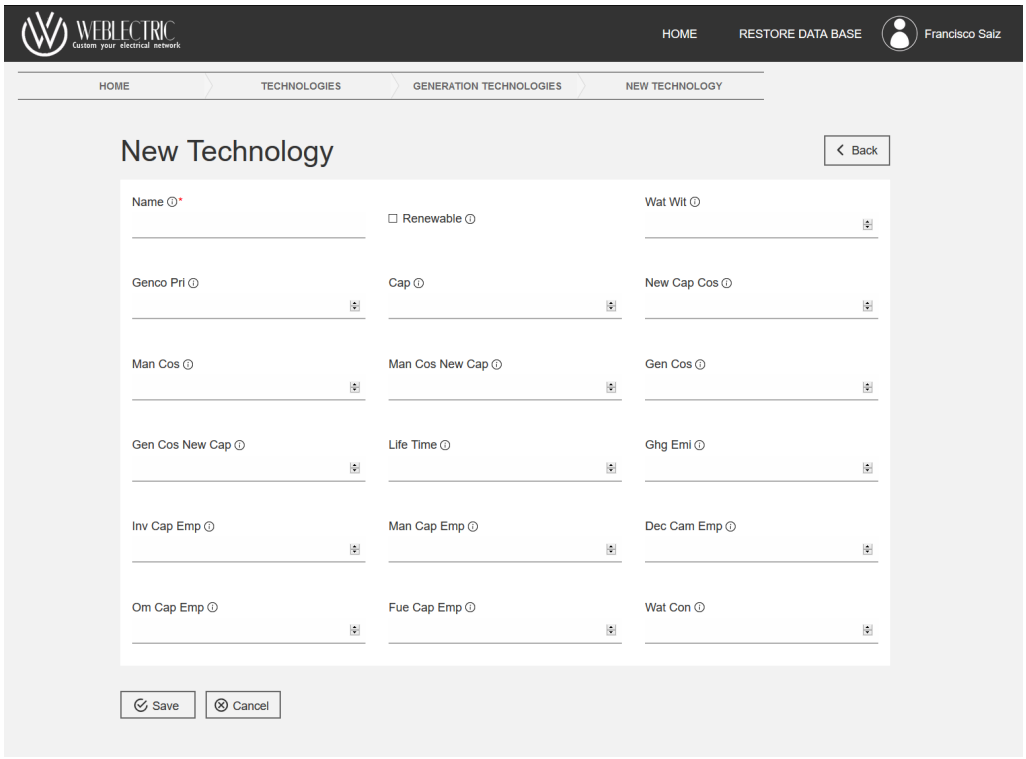

Figura C.20: Resultado final: Alta de un elemento.

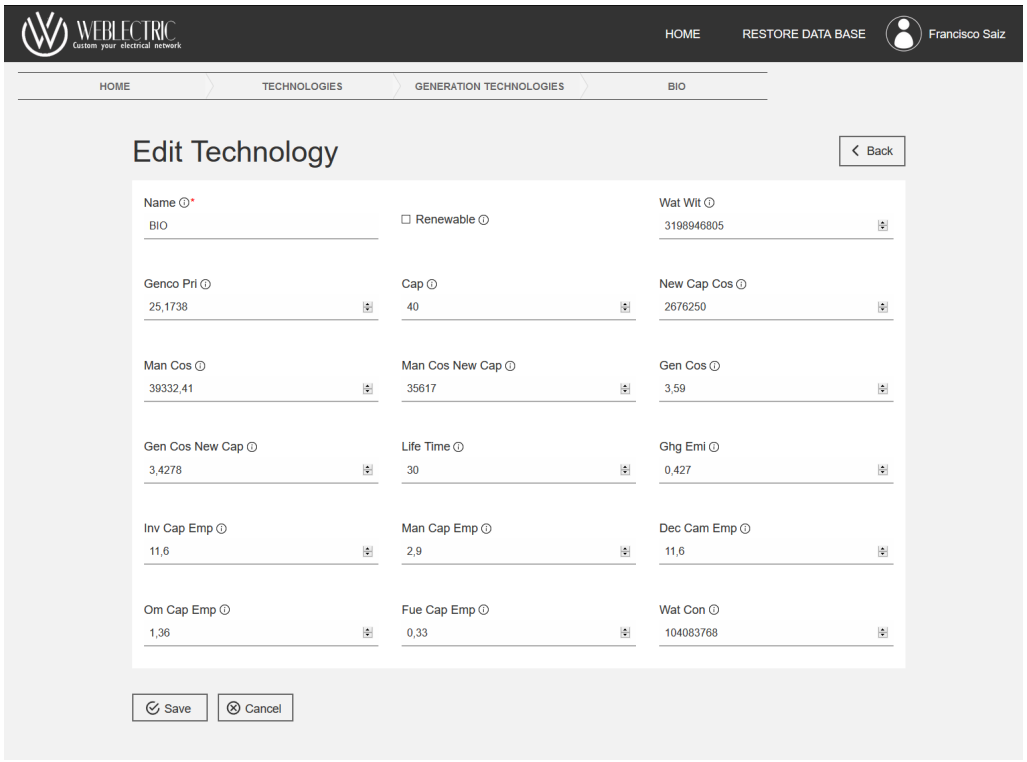

Figura C.21: Resultado final: Edición de un elemento.

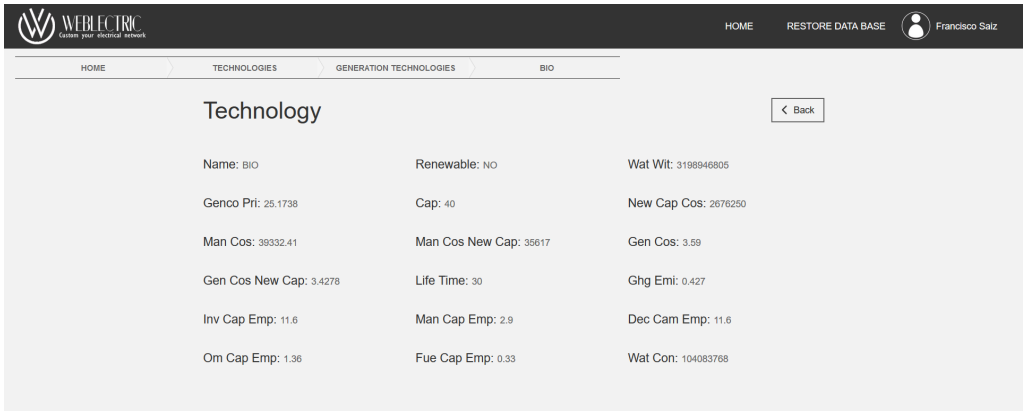

Figura C.22: Resultado final: Visualización completa de un elemento.

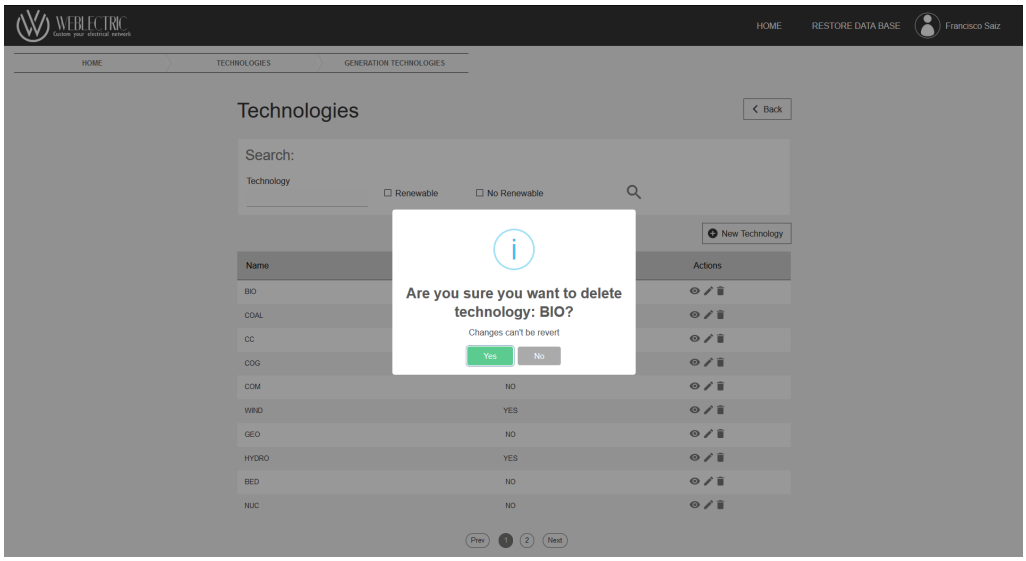

Figura C.23: Resultado final: Mensaje de confirmación al eliminar un elemento.

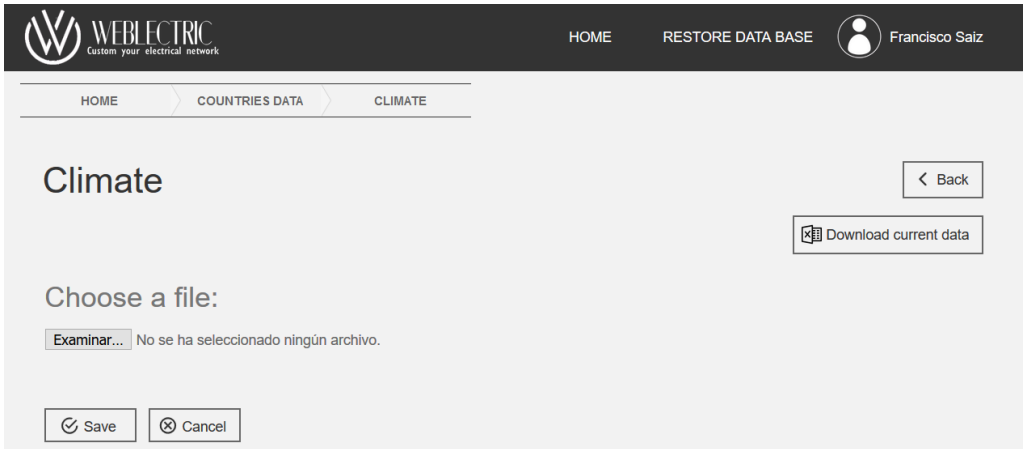

Figura C.24: Resultado final: Administración de entidades mediante hoja de cálculo.

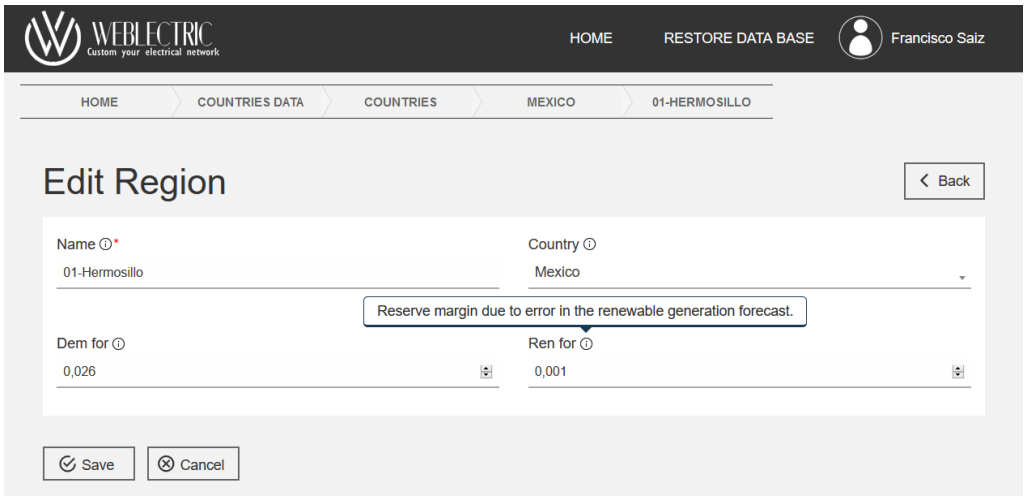

Figura C.25: Resultado final: Etiquetas con iconos y títulos, migas de pan y seccion de usuario.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En el siguiente anexo se explica todo lo que tiene que conocer el programador para instalar el entorno de trabajo y poder seguir con el desarrollo de la aplicación.

D.2. Estructura de directorios

A continuación se explicará cada uno de los directorios de la aplicación con una pequeña explicación que facilite al siguiente desarrollador su entendimiento.

```
/
├── Diseño/Diagramas  ... Aquí se encuentran los diagramas
│                         utilizados para diseñar la base de
│                         datos.
│
├── documentacion  ... Estructura de directorios de la
│   │                  plantilla LaTeX.
│   ├── img  ... carpeta donde se almacenan las
│   │            imágenes de la documentación.
│   └── tex  ... Ficheros .tex a compilar.
└── resources
    └── readme  ... Logotipos utilizados en el
                    README.md.
```

```
    |
    |__ weblectric_logo  ... Logotipo personalizado para la
    |                        aplicación.
    |
    |__ web/instalaciones_electricas  ... Estructura raíz del proyecto
    |   |                                 desarrollado en CakePHP.
    |   |
    |   |__ config  ... Ficheros de configuración.
    |   |   |__ app.php  ... Fichero de configuración más
    |   |   |                importante. Aquí se establece el
    |   |   |                valor del debug y la conexión con la
    |   |   |                base de datos entre otras cosas.
    |   |   |__ routes.php  ... Lugar donde se declaran constantes
    |   |                       globales para las rutas del
    |   |                       proyecto.
    |   |
    |   |__ files  ... Carpeta para almacenar archivos a
    |   |              utilizar en la aplicación.
    |   |__ src  ... Directorio donde se encuentra
    |   |   |        la lógica de la aplicación.
    |   |   |        Estructurada a través del patrón
    |   |   |        MVC.
    |   |   |__ Controller  ... Controladores de la aplicación.
    |   |   |
    |   |   |__ Model  ... Modelos de la aplicación.
    |   |   |__ Template  ... Vistas de la aplicación.
    |   |__ tmp  ... Lugar en el que CakePHP almacena
    |   |            temporalmente la información.
    |   |__ webroot  ... Raíz de los documentos públicos de
    |   |   |            la aplicación.
    |   |   |__ css  ... Hojas de estilos de la aplicación.
    |   |   |__ img  ... Directorio donde clasificar las
    |   |   |            imágenes que se carguen en la
    |   |   |            aplicación.
    |   |   |__ js  ... Directorio para los ficheros
    |   |                escritos en Javascript.
    |__ README.md  ... Descripción del proyecto.
```

D.3. Manual del programador

En esta sección, se hablará de los puntos más importantes a tener en cuenta y de las aplicaciones necesarias con el objetivo de que en un futuro,

un desarrollador pueda seguir trabajando en el proyecto.

XAMPP

En primer lugar, será necesario descargar *XAMPP* en su versión 7.2.10. Su función es simular un servidor en un entorno de desarrollo local. Este paquete incluye las siguientes herramientas:

- Apache 2.4.34
- MariaDB 10.1.36
- PHP 7.2.10
- phpMyAdmin 4.8.3
- OpenSSL 1.1.0g

La ventaja que proporciona la instalación de todas estas herramientas a través del paquete *XAMPP*[1] es que la configuración ya viene hecha, no obstante, si el futuro desarrollador desea realizar la instalación y configuración de cada uno de los servicios por su cuenta en su máquina local, no hay ningún problema.

Si ejecutamos el archivo descargado, como podemos ver en la figura D.1, nos dará a elegir entre qué servicios queremos instalar. El siguiente y último paso, figura D.2, es indicar en qué ruta queremos realizar la instalación.

Para ver si se ha descargado e instalado correctamente, se puede ejecutar el siguiente comando con el que se mostrará por consola la versión del *PHP* instalado:

```
php -v
```

Los resultados deben de ser como los que se muestran en la figura D.3.

Otra de las herramientas que presenta *XAMPP* es un interesante panel de control como el de la figura D.4, en el que se podrá iniciar y apagar cada uno de los servicios.

Si se inicia el servicio de *Apache* correctamente, se puede ver que accediendo a la dirección localhost[2] nos muestra información satisfactoria como la de la figura D.5.

[1] *XAMPP*: https://www.apachefriends.org/es/index.html
[2] http://localhost

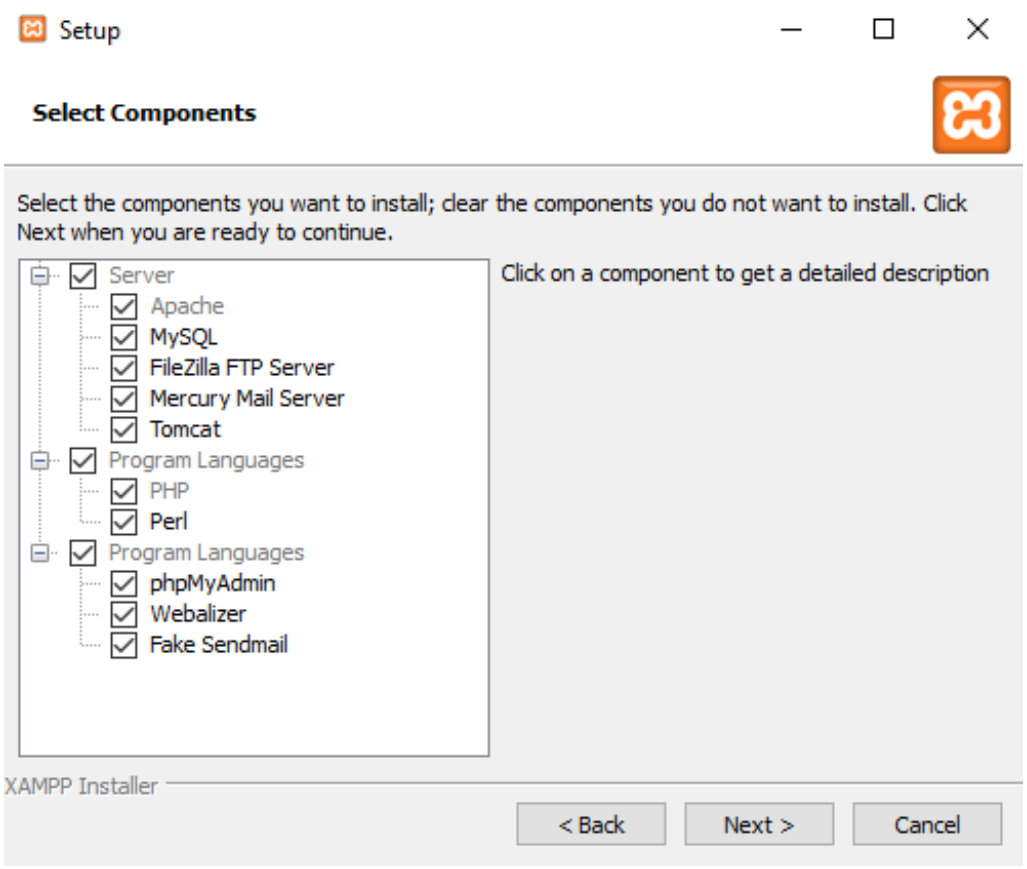

Figura D.1: *XAMPP*: Servicios que se desean instalar.

Con *XAMPP* funcionando, toca configurar las direcciones para indicar dónde tenemos nuestro proyecto. Para ello habrá que cambiar alguna linea de código en el fichero de configuración *httpd.conf* que se encuentra en la ruta *C:/xampp7/apache/conf*.

Buscamos las lineas:

```
DocumentRoot "/xampp7/htdocs"
<Directory "/xampp7/htdocs">
```

y las cambiamos por la dirección dónde hayamos colocado nuestro proyecto, en este caso:

```
DocumentRoot "C:\Users\fran_\Documents\GitHub\TFG-
    ↪ Instalaciones-Electricas\web\instalaciones_electricas"
```

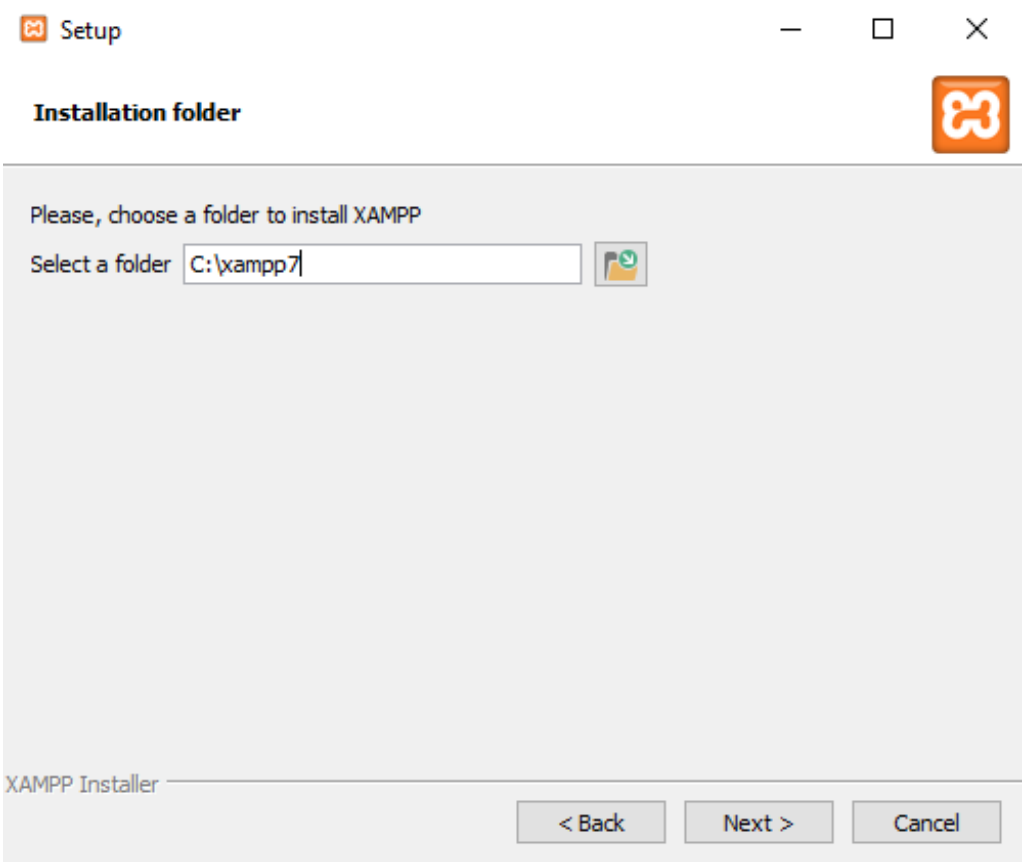

Figura D.2: *XAMPP*: Ruta donde desplegarlo.

```
<Directory "C:\Users\fran_\Documents\GitHub\TFG-Instalaciones-
    ↪ Electricas\web\instalaciones_electricas">
```

De esta manera, ya no redireccionará a la página anterior sino que cogerá la ruta especificada.

Aunque lo que se va a comentar a continuación no es obligatorio, si que es aconsejable pues si se desea tener varios proyectos, es incómodo trabajar con la dirección *localhost*, pues tendríamos que andar modificando estos archivos continuamente. Para ello existe la posibilidad de configurar todos los *virtual hosts* que se deseen. Esto significa que se podrán dar direcciones concretas a cada uno de los proyectos. Para ello, se descomentará la siguiente linea:

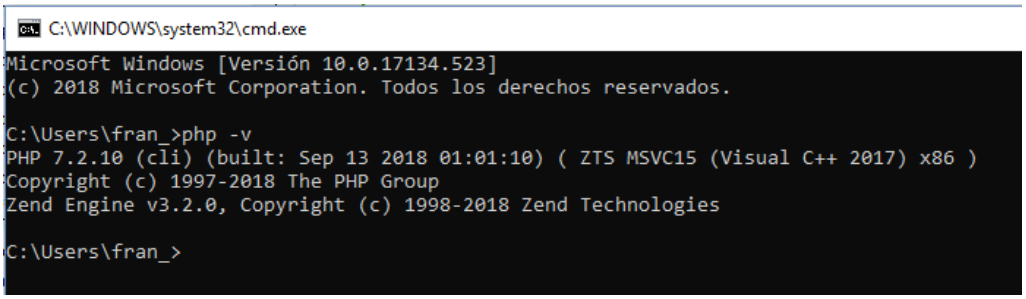

Figura D.3: *XAMPP*: Resultado de ejecutar el comando $php - v$.

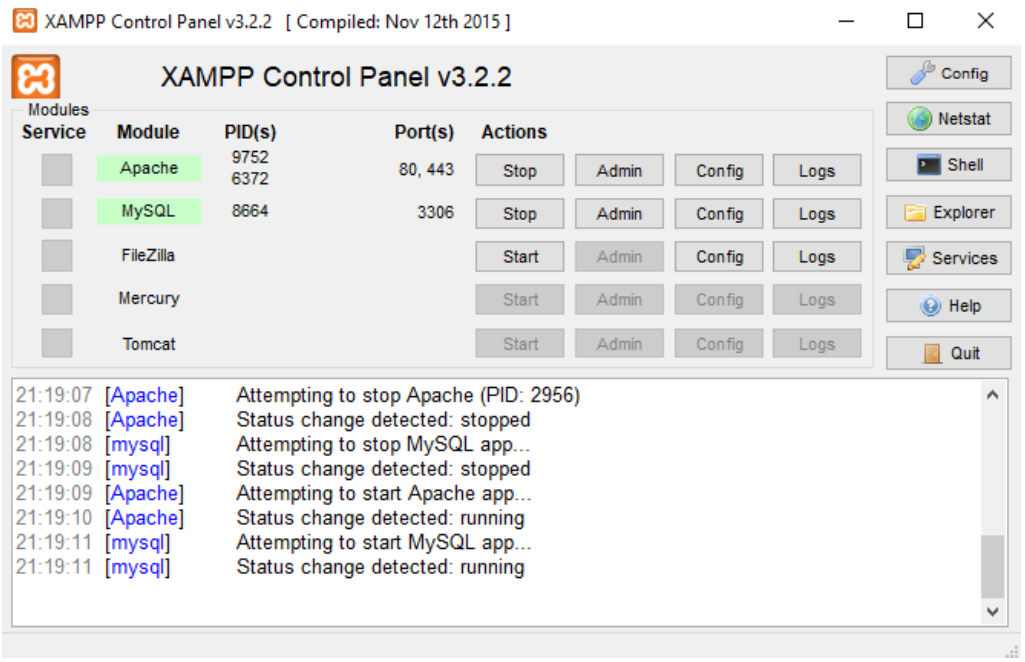

Figura D.4: *XAMPP*: Panel de control con *Apache* y *MySQL* iniciados.

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
```

El siguiente paso es dirigirse al fichero $httpd-vhosts.conf$ que se encuentra en el directorio *C:/xampp7/apache/conf/extra* para colocar las siguientes lineas al final del archivo y crear un primer *virtual hosts*.

Figura D.5: Mensaje de información acerca de que se ha instalado correctamente.

```
<VirtualHost *:80>
DocumentRoot "C:\Users\fran_\Documents\GitHub\TFG-
    ↪ Instalaciones-Electricas\web\instalaciones_electricas\
    ↪ webroot"
ServerName instalaciones_electricas.fsg
<Directory "C:\Users\fran_\Documents\GitHub\TFG-Instalaciones-
    ↪ Electricas\web\instalaciones_electricas\webroot">
Require all granted
</Directory>
</VirtualHost>
```

- *DocumentRoot*: Aquí se indica la ruta del proyecto hasta la carpeta *webroot*, que como se ha explicado antes es la parte pública a la que tiene acceso el usuario.
- *ServerName*: Dirección url para ese proyecto.

Cabe destacar que siempre que se modifique cualquier fichero de configuración se tendrán que reiniciar los servicios afectados, por lo que a continuación, el último paso será acceder de nuevo al panel de control D.4 y reiniciar el servicio.

Por último, lo que queda de configurar es la base de datos *MySQL*. Aunque con el paquete *XAMPP* viene preparado para su uso desde *phpMyAdmin*, se ha decidido descargar *HeidiSQL*[3] pues presenta una interfaz más amigable y

[3] *HeidiSQL*: https://www.heidisql.com/download.php

estable. Además, existe una versión portable que no requiere de instalación.

Si ejecutamos la aplicación, aparecerá una pantalla como la de la figura D.6, donde nos aparecerán todas las sesiones que tengamos activas:

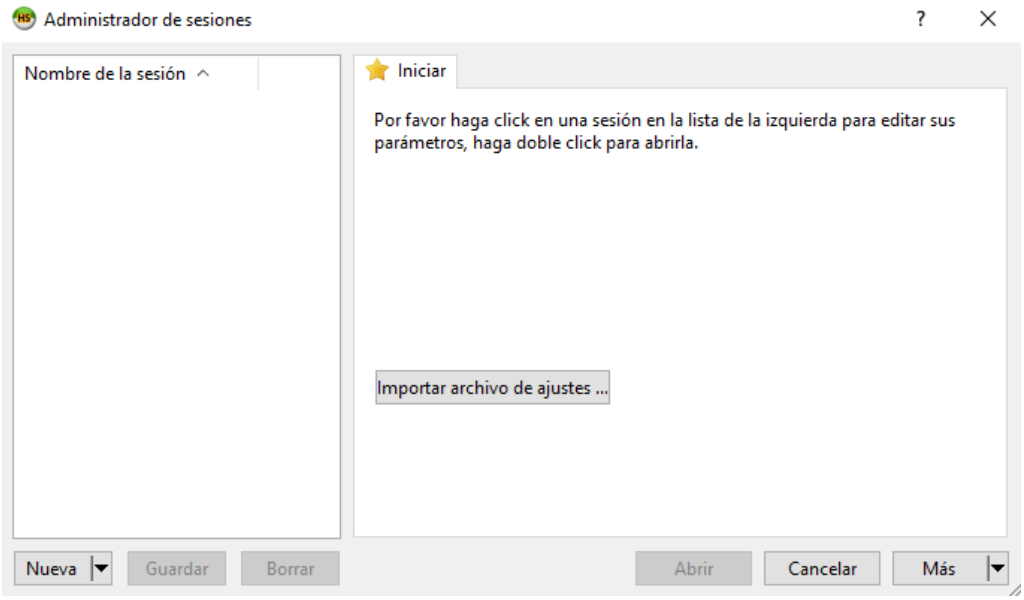

Figura D.6: Pantalla de inicio de *HeidiSQL*.

Procedemos a crear una nueva sesión con los parámetros que se ven en la figura D.7:

Con la sesión ya creada, se podrá crear una base de datos nueva con las tabas que se desee. A continuación, en la figura D.8 se va a mostrar como crear un usuario para esa base de datos para que posteriormente se pueda enlazar con el proyecto.

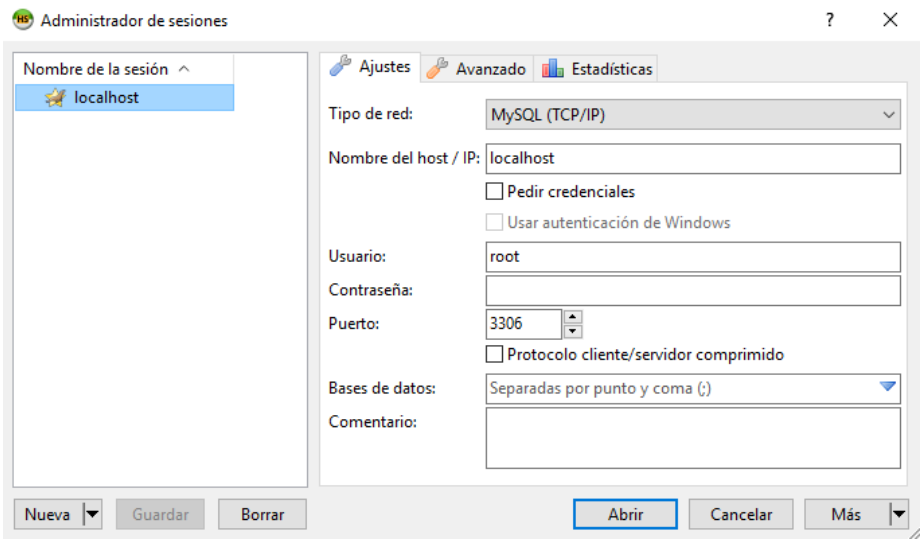

Figura D.7: Parámetros para una nueva sesión en entorno local.

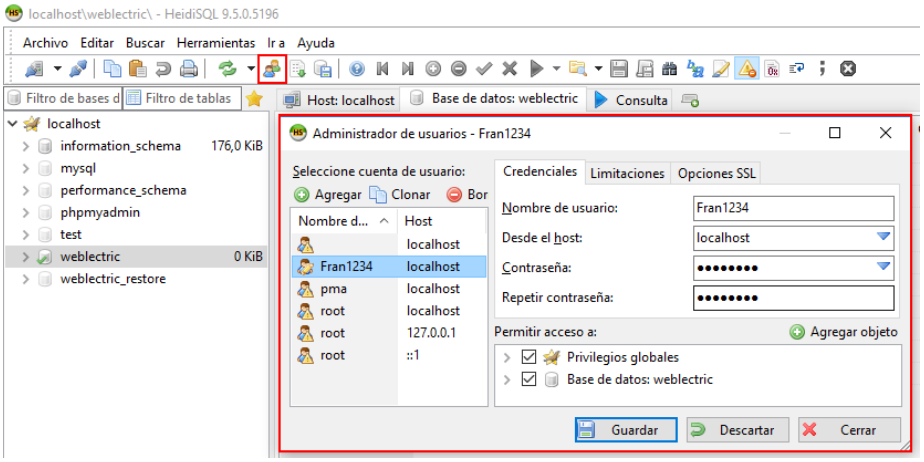

Figura D.8: Parámetros para asociar un usuario a la base de datos.

D.4. Compilación, instalación y ejecución del proyecto

En primer lugar, se procederá a descargar *Weblectric*[4] del repositorio de *GitHub*.

El lugar en el que se descomprimirá el archivo descargado, será el direc-

[4] *Weblectric*: https://github.com/fransaiz95/Weblectric2018

torio que se haya especificado anteriormente, en nuestro caso:

```
C:\Users\fran_\Documents\GitHub\TFG-Instalaciones-Electricas\
    ↪ web\"
```

Como se puede ver en la figura D.9, si ahora se introduce la url que se ha establecido en el *virtual host*, se accederá satisfactoriamente a la pantalla principal del proyecto.

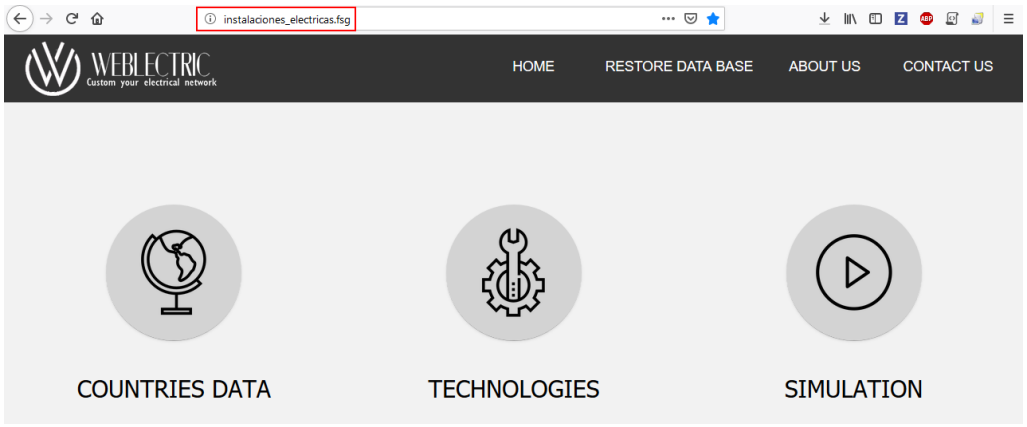

Figura D.9: *Virtual host* correctamente configurado.

Dentro del entorno de desarrollo, no hay dificultades pues no hay más que abrir la carpeta contenedora del proyecto.

En este proyecto, que se ha utilizado *Visual Studio Code* como herramienta para desarrollar, la estructura viene representada de la siguiente manera: D.10

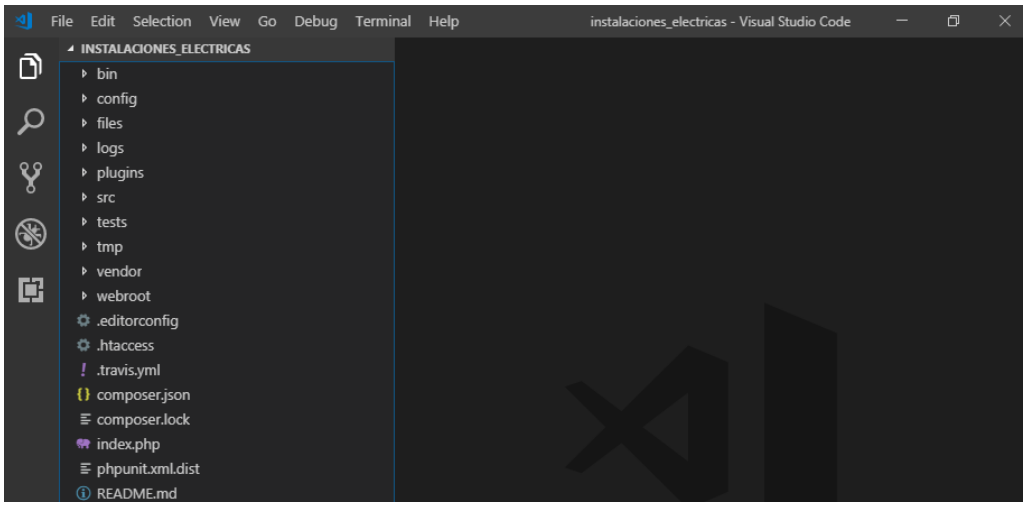

Figura D.10: Estructura de directorios con el proyecto importado.

Para acabar, en el archivo *app.conf* que se encuentra dentro de la carpeta *config*, es necesario reemplazar las lineas que a continuación se van a describir para enlazar el proyecto con la base de datos:

```
'Datasources' => [
        'default' => [
                'className' => 'Cake\Database\Connection',
                'driver' => 'Cake\Database\Driver\Mysql',
                'persistent' => false,
                'host' => 'localhost',
                'username' => 'Fran1234',
                'password' => 'Fran1234',
                'database' => 'weblectric',
                'encoding' => 'utf8',
                'timezone' => 'UTC',
                'flags' => [],
                'cacheMetadata' => true,
                'log' => false,
        ],
],
```

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

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