



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**título del TFG
Documentación Técnica**



Presentado por nombre alumno
en Universidad de Burgos — 13 de enero
de 2019

Tutor: nombre tutor

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	IV
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	7
Apéndice B Especificación de Requisitos	11
B.1. Introducción	11
B.2. Objetivos generales	11
B.3. Catalogo de requisitos	12
B.4. Especificación de requisitos	14
Apéndice C Especificación de diseño	21
C.1. Introducción	21
C.2. Diseño de datos	21
C.3. Diseño procedimental	23
C.4. Diseño arquitectónico	24
Apéndice D Documentación técnica de programación	39
D.1. Introducción	39
D.2. Estructura de directorios	39
D.3. Manual del programador	39

D.4. Compilación, instalación y ejecución del proyecto	39
D.5. Pruebas del sistema	39
Apéndice E Documentación de usuario	41
E.1. Introducción	41
E.2. Requisitos de usuarios	41
E.3. Instalación	41
E.4. Manual del usuario	41
Bibliografía	43

Índice de figuras

B.1. Diagrama de casos de uso.	14
C.1. Diagrama entidad relación.	22
C.2. Diagrama relacional.	23
C.3. Diagrama secuencial.	24
C.4. Patrón Modelo-Vista-Controlador. [1]	25
C.5. Mockup: Plantilla común a todas las vistas.	28
C.6. Mockup: Pantalla principal.	29
C.7. Mockup: Pantalla principal de la entidad 1.	29
C.8. Mockup: Alta de un elemento.	30
C.9. Mockup: Edición de un elemento.	30
C.10. Mockup: Visualización completa de un elemento.	30
C.11. Mockup: Mensaje de confirmación al eliminar un elemento.	31
C.12. Mockup: Administración de entidades mediante hoja de cálculo.	31
C.13. Resultado final: Plantilla común a todas las vistas.	32
C.14. Resultado final: Pantalla principal.	32
C.15. Resultado final: Pantalla países.	33
C.16. Resultado final: Pantalla tecnologías.	33
C.17. Resultado final: Pantalla simulación.	34
C.18. Resultado final: Pantalla principal de la entidad 1.	34
C.19. Resultado final: Alta de un elemento.	35
C.20. Resultado final: Edición de un elemento.	35
C.21. Resultado final: Visualización completa de un elemento.	36
C.22. Resultado final: Mensaje de confirmación al eliminar un elemento.	36
C.23. Resultado final: Administración de entidades mediante hoja de cálculo.	37

Índice de tablas

A.1. Costes totales del proyecto	9
A.2. Herramientas utilizadas y sus licencias	10
B.1. Caso de uso 1 - Administración mediante formularios.	15
B.2. Caso de uso 2 - Administración mediante hojas de cálculo.	16
B.3. Caso de uso 3 - Visualización de datos en tablas paginadas.	17
B.4. Caso de uso 4 - Exportar datos en hoja de cálculo.	18
B.5. Caso de uso 5 - Restablecer base de datos.	19
C.1. Estructura MVC en <i>Weblectric</i>	25

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 análisis 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 nuestra organización pues está diseñada para poder realizar entregas parciales y periódicas en pequeños y cortos espacios de tiempo, donde unos pronto resultados y una flexibilidad considerable son fundamentales. Para concretar todo lo comentado, necesitábamos 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.

Tanto 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 o 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 nos 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, podemos gestionar nuestro tiempo de una manera más eficiente ya que permite asignar tiempos a cada una de las *issues*. De esta manera, te permitía 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 iteraciones 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.

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

- Diseñar la base de datos *MySQL* a través de un modelo Entidad-Relación.
- Buscar un servidor gratuito donde poder alojar la aplicación.
- Crear la estructura de directorios del proyecto.

Este *sprint* me 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 lo vamos a alojar, me ha servido para refrescar conceptos de base de datos y realizar el modelo entidad relación.

En cuanto al framework a utilizar, me decanté por *CakePHP 3.5* pues aunque en el trabajo utilizamos una versión bastante más antigua² no deja de ser el mismo *framework* y satisfacía las necesidades presentadas.

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 nosotros diseñar la base de datos en función de sus necesidades.
- Crear tablas en Base de Datos *MySQL*.
- Documentación: Cargar la plantilla *Latex*.

Una vez que teníamos claro qué *framework* se iba a utilizar, que habíamos creado correctamente la estructura del proyecto, el paso 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.

² CakePHP 2.x

- 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 desgranaba 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 desgranados 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 la hemos hecho leyendo los datos directamente de la hoja de cálculo facilitada por el cliente.

Dado que existían pequeñas discrepancias entre los propios datos y no teniendo la seguridad de que ser los datos y estructura definitiva, nos limitamos a hacer un primer boceto de lo que sería nuestro diagrama relacional y se empezó a preparar la estructura MVC (Modelo, Vista y Controlador) para las entidades que teníamos fijas. 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 un excel. II
- Empezar la administración de los datos.

Una vez que teníamos la estructura montada en el proyecto, seguimos dando forma a nuestro diagrama hasta su versión definitiva. Con la base de datos ya bien montada, utilizando la librería *PhpSpreadsheet* empezamos 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 nos dimos cuenta de que había tablas cuyos datos no existían y es que el cliente estaba pendiente de facilitarlos.

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 nos vino bien 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 *rangedemands*.

Mientras que para las tablas en las que la administración era a través de un formulario no existía mayor problema, la tabla *rangedemands* se abastece 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. Nosotros hemos tenido que entrar al fichero `php.ini` y modificar el valor de una propiedad dejándola como: `upload_max_filesize=100M`

Al margen del problema con 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) 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.

...

...

...

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

Se va a proceder a hacer un estudio de viabilidad económica del proyecto justificando todos los gastos que se hubieran tenido si este 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: XXX horas. Suponiendo que la hora de trabajo del desarrollador está pagada a 8 €, tenemos un total de:

$$8€/h \times 000h = 280€$$

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:

$$7sprints \times 10€/h \times 4h = 0000€$$

De momento sumamos un total de XXXX € que si le aplicamos el XX % de seguridad social, se queda en un total de:

$$XXXX + (XXXX \times 0,00) = YYYY$$

Por último, destacar el coste material. En la parte software no tenemos 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 ³, los equipos para procesos de información tienen una vida útil de 8 años (70.080 horas) y como nuestro proyecto ha tenido una duración de XXX horas, deja un total de:

$$\frac{775€}{12 \text{ Meses} \times 8 \text{ Años}} \times 3,5 \text{ Meses} = 28,26€$$

Por último, sumar el gasto producido por el servidor donde hemos tenido alojada la web. Aunque gran parte de la duración del proyecto lo hemos

³Tablas de amortización en 2018: <https://bit.ly/2ND4vCw>

tenido en un *hosting* gratuito, en la recta final y por causa de las limitaciones que nos ofrecía el plan, en el último mes de proyecto, los gastos han sido:

$$\frac{18\text{€}}{3 \text{ Meses}} = 6\text{€}$$

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	XXXX €
Tutores del proyecto	280 €
Seguridad Social	XXXX €
Material hardware	28,26 €
Servidor	6 €
Totales	YYYY €

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.
- 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.
 - Aceites.
 - Tecnologías
 - Regiones.
 - Relación entre aceites 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 a 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.

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:

- Desarrollador: Realiza la primera carga de datos iniciales y prepara la consulta *sql* para poder restablecer la base de datos.
- Usuario: Interactua con la aplicación web.

Diagrama de casos de uso

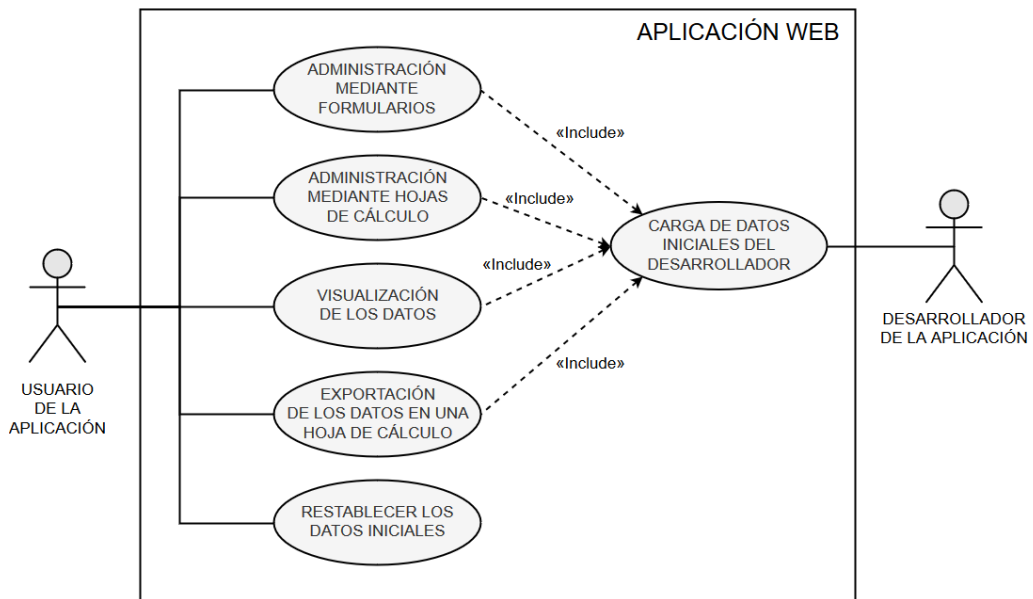


Figura B.1: Diagrama de casos de uso.

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 de entrar en la pantalla correspondiente a la entidad que quiere modificar. 2. Se deberá de 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 ultima 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	<ol style="list-style-type: none"> 1. Base de datos iniciada. 2. Carga inicial de datos realizada. 3. Realización de una consulta.
Acciones	<ol style="list-style-type: none"> 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 hacer click 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 5	Restablecer base de datos.
Versión	1.0
Requisitos	RF-5
Descripción	El usuario podrá podrá restablecer la base de datos a un punto de partida pulsando un botón situado en la cabecera.
Precondiciones	Consulta <i>sql</i> mal preparada.
Acciones	1. Pulsar sobre el botón situado en la cabecera. 2. Aceptar el mensaje de confirmación.
Postcondiciones	Mensaje de información anunciando que la base de datos ha vuelto al punto de partida.
Excepciones	Error al ejecutar la acción.
Importancia	Media

Tabla B.5: Caso de uso 5 - Restablecer base de datos.

Apéndice C

Especificación de diseño

C.1. Introducción

A continuación presentaremos los diseños que se elaborado para poder realizar los objetivos anteriores. Se ha incluido el diseño de datos, el diseño procedimental y el diseño arquitectónico.

C.2. 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 figuras C.1 y C.2 presentamos los modelos entidad relación y relacional respectivamente.

A continuación se describirán cada una de las entidades:

- *countries*: Los países presentes en nuestra instalación.
- *regions*: Cada una de las regiones pertenecientes a un país.
- *arcs*: Comunicación entre dos regiones.
- *fuels*: Tipos de aceites.
- *technologies*: Tipos de tecnologías renovables y no renovables.
- *typelines*: Tipo de conexión de los arcos entre dos regiones.

- *arcs_typelines*: Relación entre el arco y el tipo de arco.
- *fuels_technologies*: Relación entre el aceite y la tecnología.
- *regions_technologies*: Relación entre la región y qué tecnología hay presente.
- *rangedemands*: Datos de demanda en un instante de tiempo concreto.
- *rangemeteos*: Datos meteorológicos en un instante de tiempo concreto.
- *rangerenowables*: Datos de fuentes renovables en un instante de tiempo concreto.

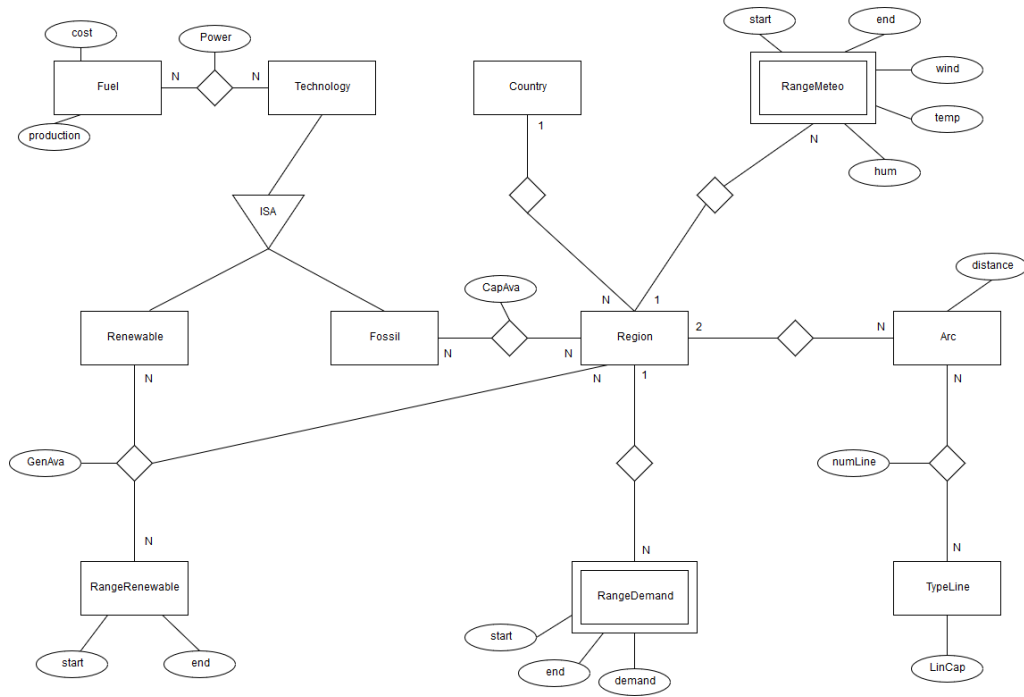


Figura C.1: Diagrama entidad relación.

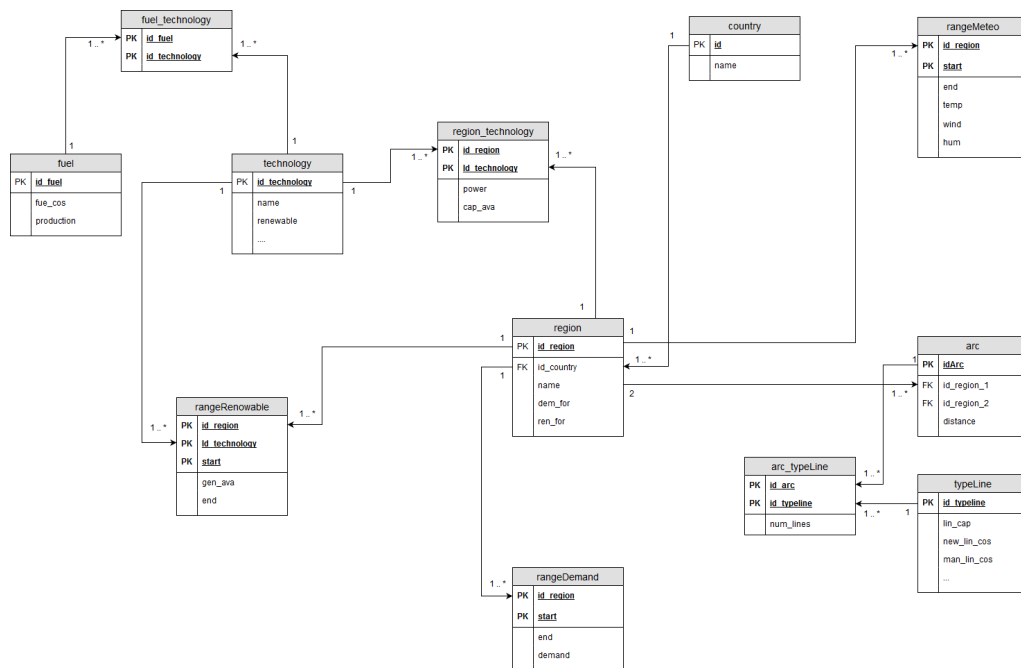


Figura C.2: Diagrama relacional.

C.3. 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.3](#).

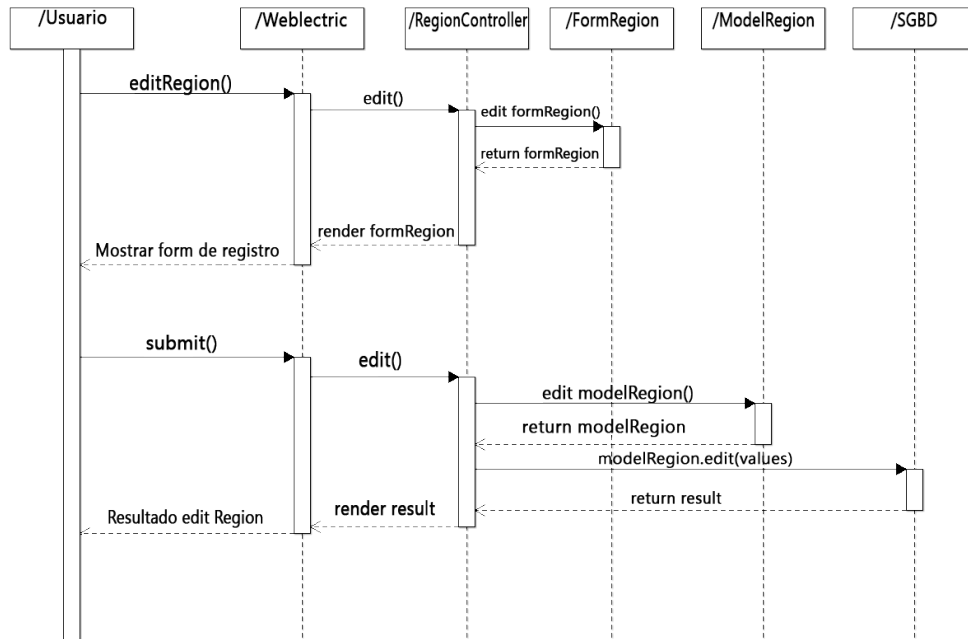


Figura C.3: Diagrama secuencial.

C.4. 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.4. 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 guardamos las pantallas que interactúan con el cliente.
- *Controlador*: Aquí encontramos lo que es la lógica de la aplicación. Hace de unión entre la vista y el modelo.

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

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

Patrón MVC	Directorio en <i>Weblectric</i>
Modelos	<i>models</i>
Vistas	<i>templates</i>
Controladores	<i>controllers</i>

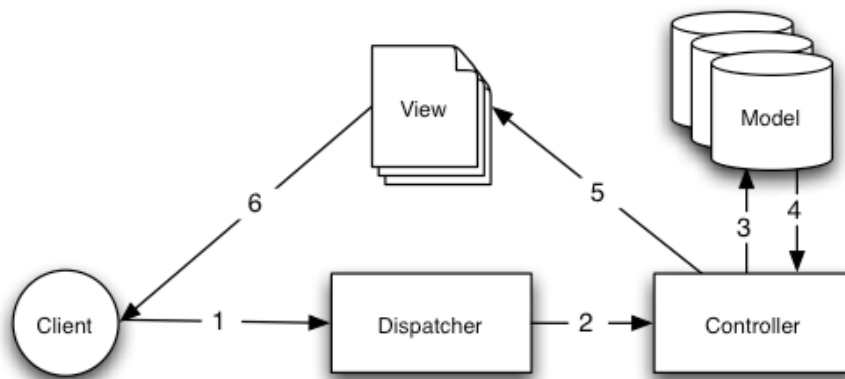


Figura C.4: Patrón Modelo-Vista-Controlador. [1]

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 podemos ver en la figura C.5 es común a todas las vistas.

Posteriormente, pasamos a diseñar la pantalla principal C.6. Aquí se iban a listar todas las entidades para que el usuario pudiese acceder a administrarlas.

Como se puede ver en la figura C.6, al pasar el ratón por encima, aparecerá una pequeña animación diferenciando cada una de las entidades. La siguiente pantalla que vamos a presentar es la administración. Tenemos 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.7, 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.8.
- Editar un elemento existente: C.9.
- Visualizar detalles de un elemento existente: C.10.
- Eliminar un elemento: C.11.

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.12 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.13.

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.14:
 - Countries data: C.15.
 - Country.
 - Renewable source.
 - Climate.
 - Current System.
 - Technologies: C.16.
 - Generation technologies.
 - Types of lines.
 - Fuels.
 - Simulation: C.17.
 - Objectives.
 - Scenarios.
 - 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.18](#).
- Crear una nueva tecnología: [C.19](#).
- Editar una tecnología: [C.20](#).
- Visualizar detalles de una tecnología: [C.21](#).
- Eliminar una tecnología: [C.22](#).

Administración mediante hojas de cálculo

En la figura [C.23](#) encontramos un ejemplo de como se administran los datos climáticos.

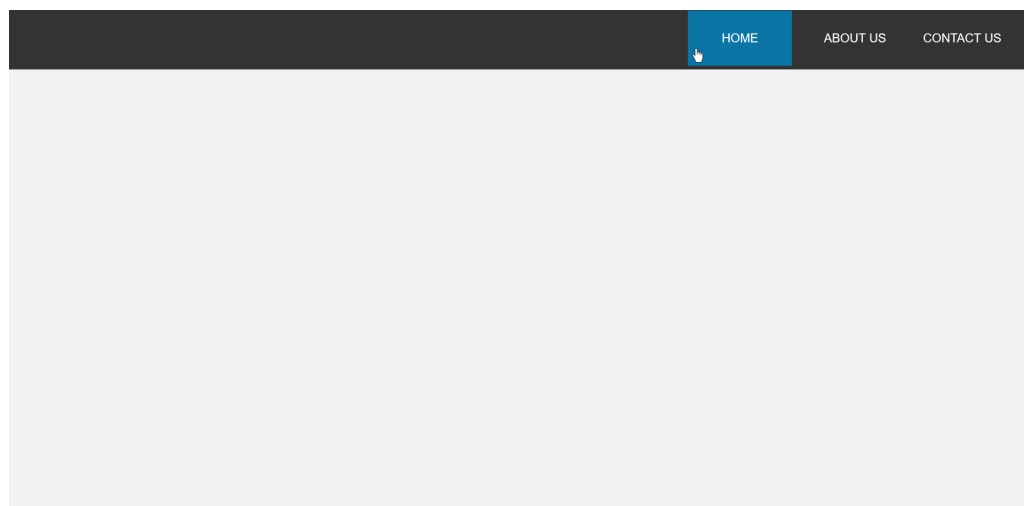


Figura C.5: Mockup: Plantilla común a todas las vistas.

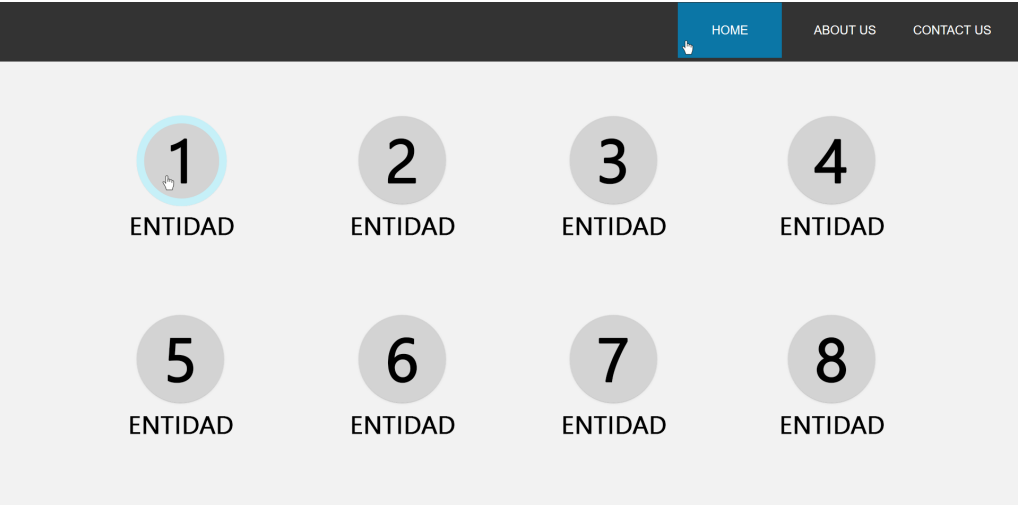


Figura C.6: Mockup: Pantalla principal.

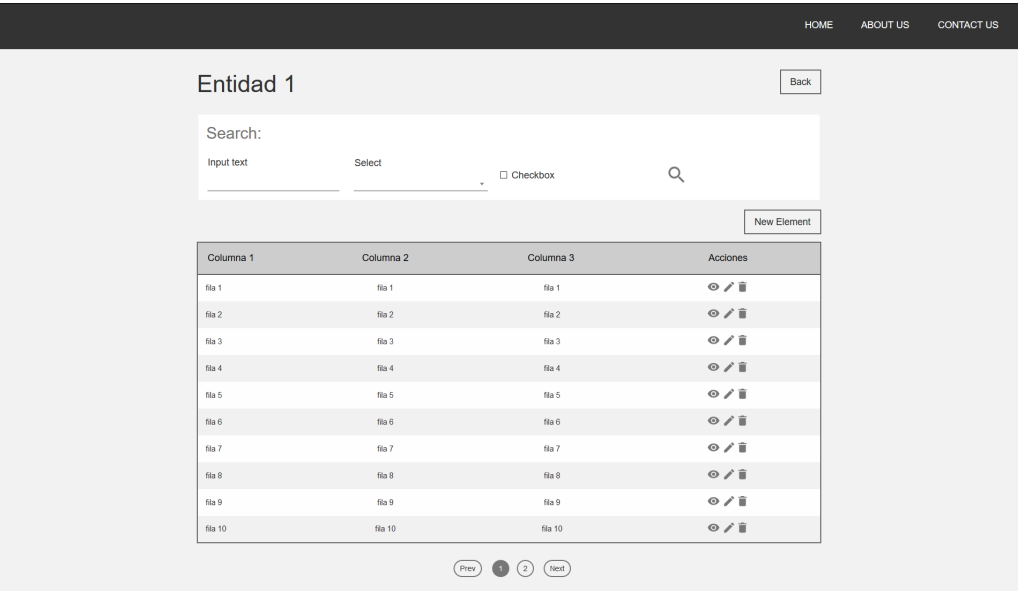


Figura C.7: Mockup: Pantalla principal de la entidad 1.

HOMEABOUT USCONTACT US

New Element

Back

Field 1 *

Field 2

Field 3

Field 4

Save

Cancel

Figura C.8: Mockup: Alta de un elemento.

HOMEABOUT USCONTACT US

Edit Element

Back

Field 1 *

value 1

Field 2

value 2

Field 3

3

Field 4

4

Save

Cancel

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

HOMEABOUT USCONTACT US

Entidad 1

Back

Field 1: value 1

Field 2: value 2

Field 3: value 3

Field 4: value 4

Field 5: value 5

Field 6: value 6

Figura C.10: Mockup: Visualización completa de un elemento.

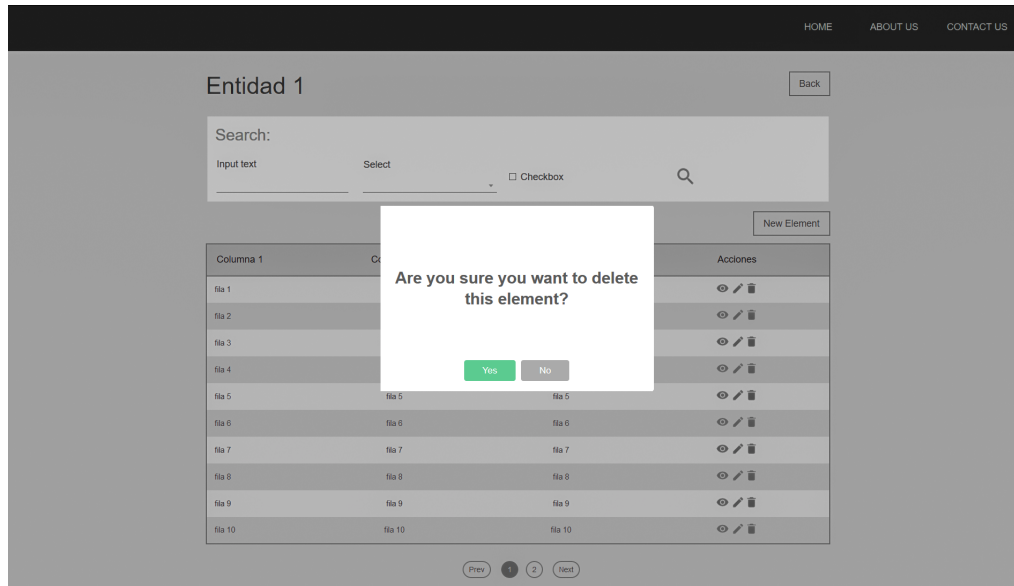


Figura C.11: Mockup: Mensaje de confirmación al eliminar un elemento.



Figura C.12: Mockup: Administración de entidades mediante hoja de cálculo.

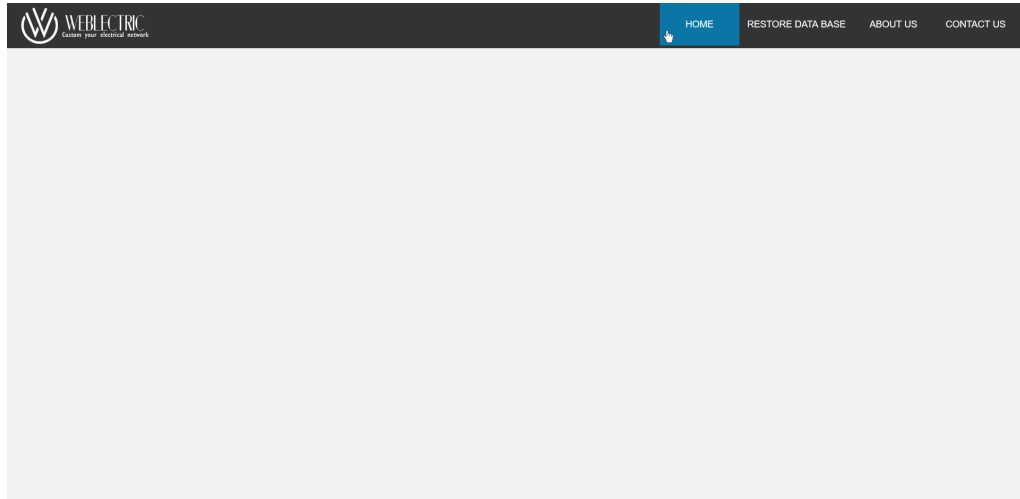


Figura C.13: Resultado final: Plantilla común a todas las vistas.

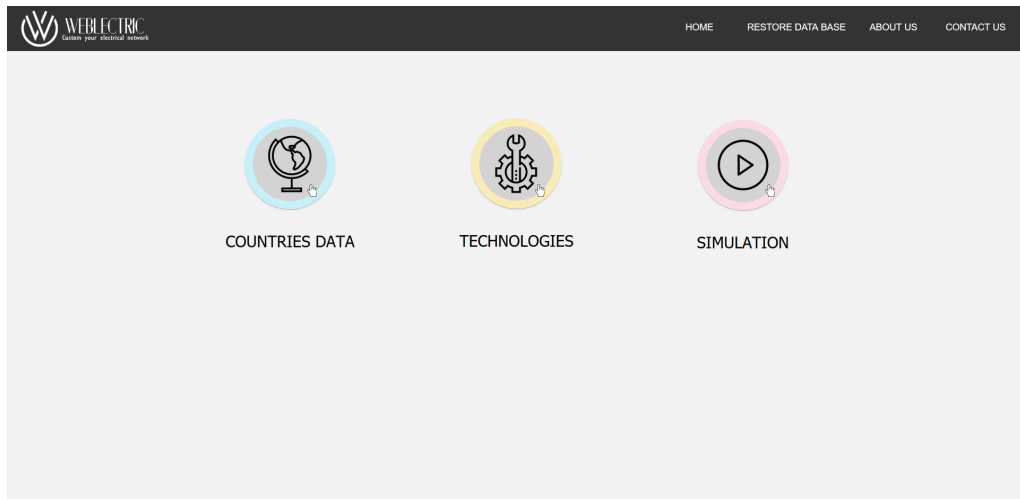


Figura C.14: Resultado final: Pantalla principal.

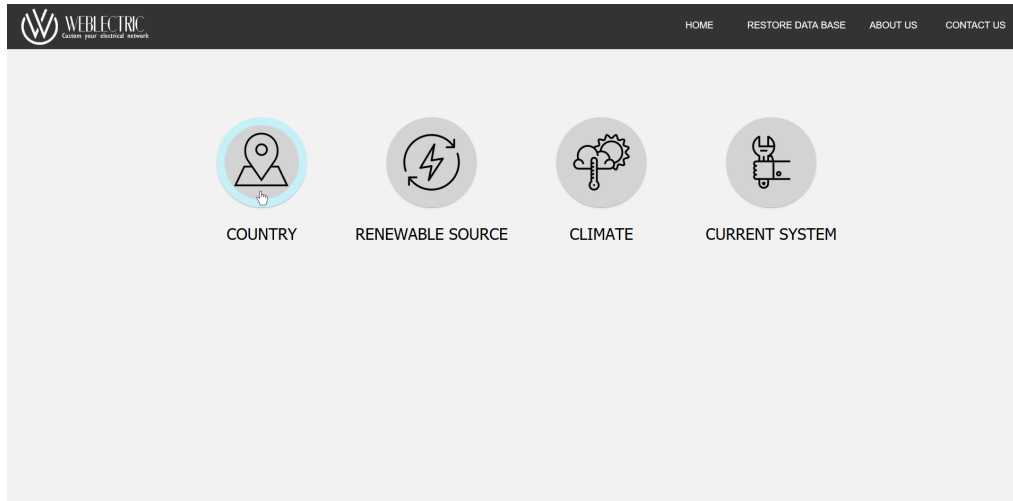


Figura C.15: Resultado final: Pantalla países.

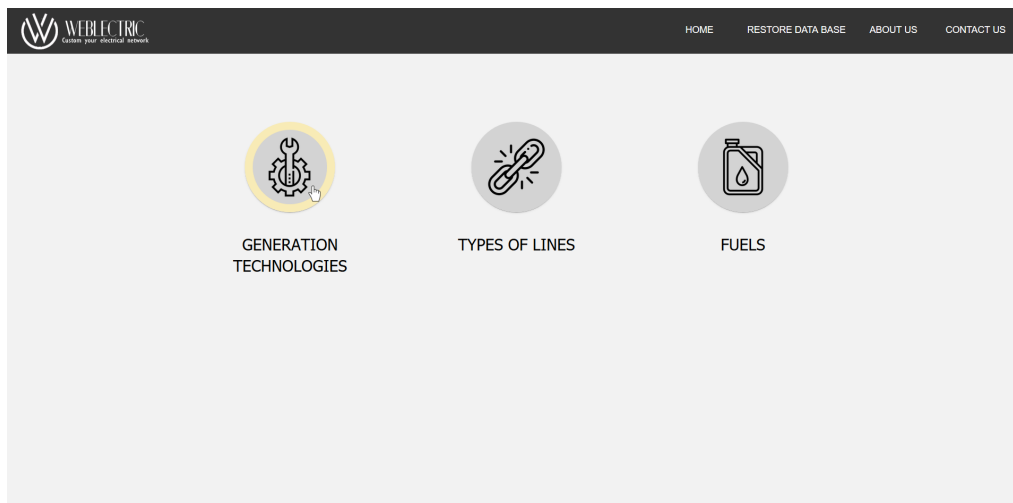


Figura C.16: Resultado final: Pantalla tecnologías.

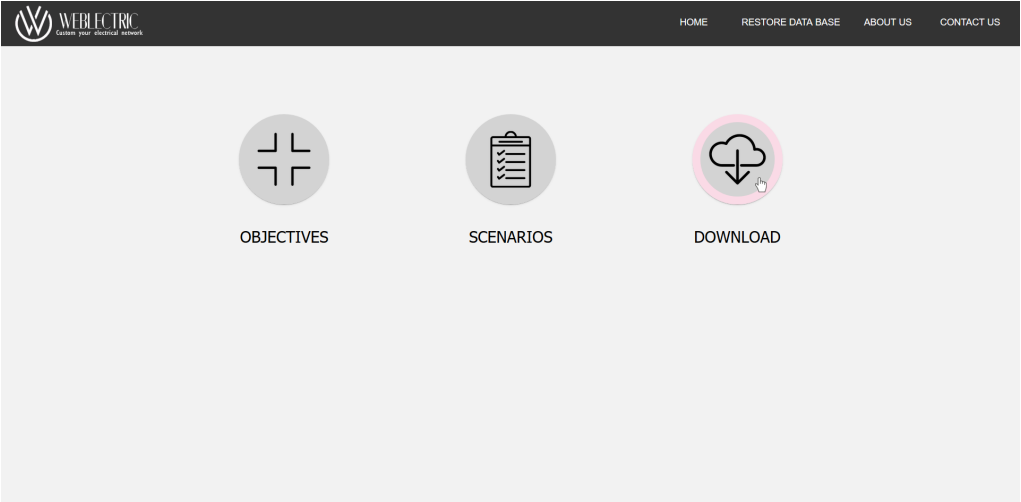


Figura C.17: Resultado final: Pantalla simulación.

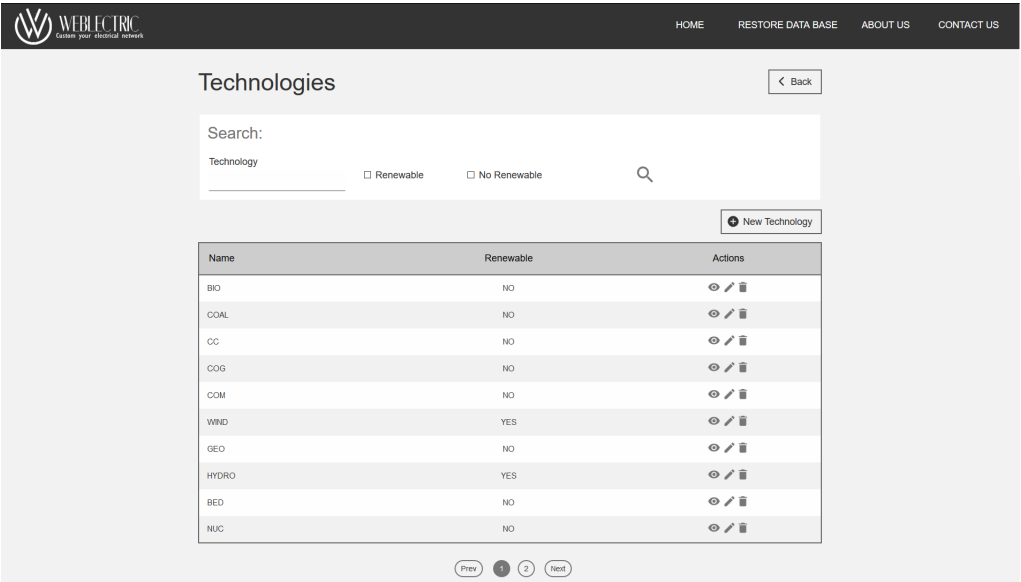



Figura C.18: Resultado final: Pantalla principal de la entidad 1.



HOME RESTORE DATA BASE ABOUT US CONTACT US

New Technology

< Back

Name *

☐ Renewable

Wat Wlt

Genco Ptl

Cap

New Cap Cos

Man Cos

Man Cos New Cap

Gen Cos

Gen Cos New Cap

Life Time

Ghg Emi

Inv Cap Emp

Man Cap Emp

Dec Cam Emp

Om Cap Emp


Fue Cap Emp

Wat Con

Save

Cancel

Figura C.19: Resultado final: Alta de un elemento.



HOME RESTORE DATA BASE ABOUT US CONTACT US

Edit Technology

< Back

Name *

BIO

☐ Renewable

Wat Wlt

3198946805

Genco Ptl

25,1738

Cap

40

New Cap Cos

2676250

Man Cos

39332,41

Man Cos New Cap

35617

Gen Cos

3,59

Gen Cos New Cap

3,4278

Life Time

30

Ghg Emi

0,427

Inv Cap Emp

11,6

Man Cap Emp

2,9

Dec Cam Emp

11,6

Om Cap Emp

1,36

Fue Cap Emp

0,33

Wat Con

104083768

Save

Cancel

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

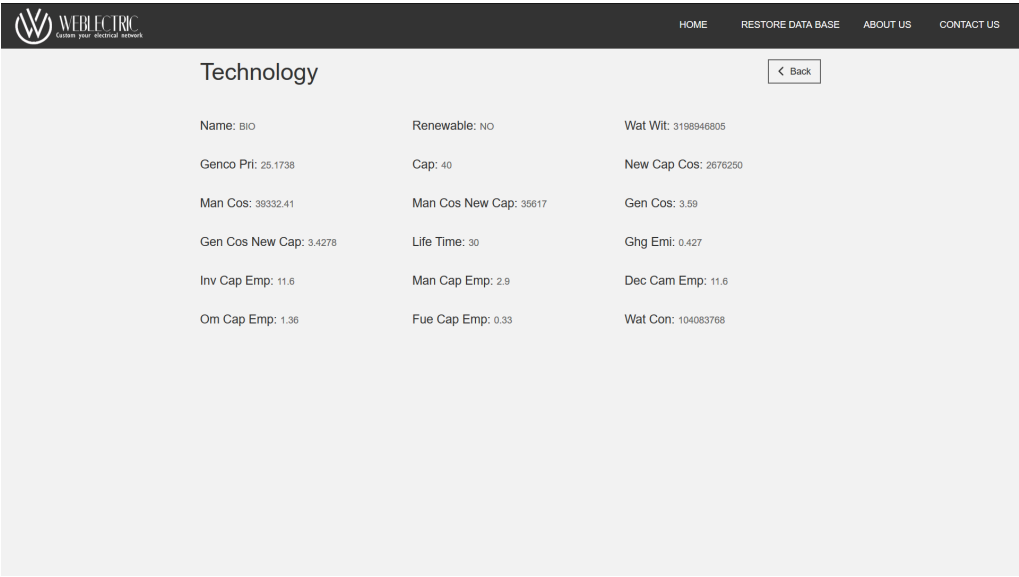


Figura C.21: Resultado final: Visualización completa de un elemento.

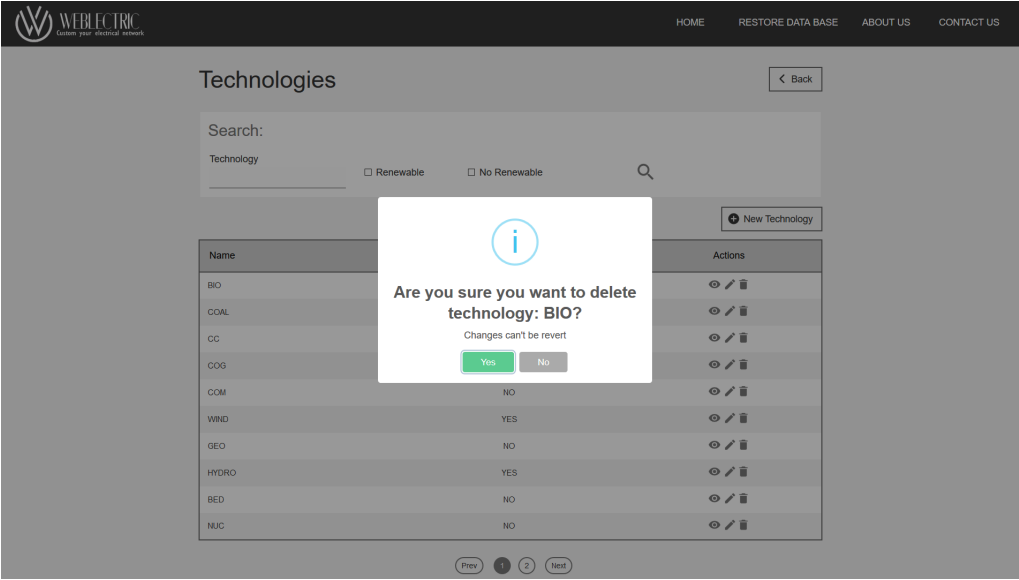


Figura C.22: Resultado final: Mensaje de confirmación al eliminar un elemento.

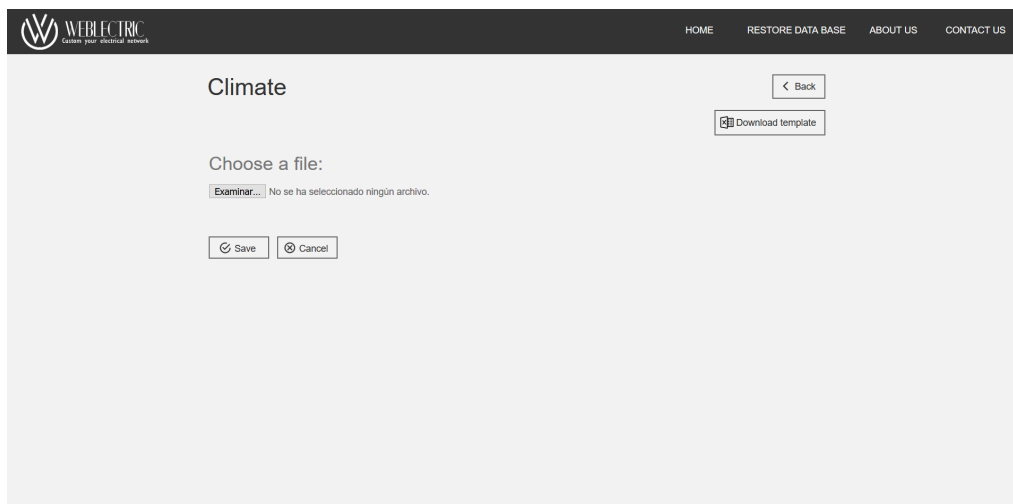


Figura C.23: Resultado final: Administración de entidades mediante hoja de cálculo.

Apéndice D

Documentación técnica de programación

- D.1. Introducción
- D.2. Estructura de directorios
- D.3. Manual del programador
- D.4. Compilación, instalación y ejecución del proyecto
- D.5. Pruebas del sistema

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.