Universidad Zaragoza

1542

Proyecto Fin de Carrera

Energy4People

Sistema de soporte para la toma de decisiones en el dominio de las energías renovables orientado al usuario no experto

Borja A. Espejo García

Director: Naveen Sidda
Ponente: Dr. Francisco Javier López Pellicer

Escuela de Ingeniería y Arquitectura

2013

Agradecimientos

A los miembros del grupo IAAA y trabajadores de GSL por su reconfortante compañía durante estos últimos meses, especialmente al profesor Francisco Javier López por su inestimable ayuda en mi formación como persona e ingeniero, y a Naveen Sidda por facilitarme la comprensión del proyecto.

A todos mis compañeros de carrera por hacer de ella una etapa más divertida y entrañable .

A mis compañeros de equipo en el concurso de la NASA, especialmente a Alberto por acompañarme en la aventura.

A toda mi familia, especialmente a mis padres por su apoyo durante toda la carrera

Energy4People

Sistema de soporte a la toma de decisiones en el dominio de las energías renovables orientado al usuario no experto

Resumen

El sector de las energías renovables está viviendo su particular crisis. Los últimos cambios en su marco regulatorio han dejado tocado a un sector que hace pocos años estaba en alza. Esto, junto al carácter fluctuante de las fuentes de energía, como la radiación solar o la velocidad de viento, hacen que la toma de decisiones en este sector sea muy complicada.

Los sistemas de toma de decisiones tienen una gran relevancia dentro de este sector. Un ejemplo de estos sistemas es la denominada Inteligencia de Negocio o Business Intelligence (BI), que guía en el mundo empresarial en la toma de decisiones. BI tiene como objetivo generar información fiable a partir fuentes de datos heterogéneas, y de esta manera facilitar la toma de decisiones.

Este proyecto propone un sistema de soporte a la toma de decisiones, en el dominio de las energías renovables y orientado a usuarios no expertos. El sistema utiliza de manera transparente para el usuario técnicas de BI como la integración de datos, que ha sido aplicada a datos procedentes de satélites de la NASA, estructuras multidimensionales como cubos OLAP y modelos de datos orientados al análisis como el esquema de estrella de un data warehouse. Con este sistema, por ejemplo, un usuario no experto puede visualizar y entender dónde es mejor realizar una inversión en paneles solares.

Un prototipo de este proyecto llamado Energy2People participó en el concurso “Space Apps Challenge” organizado por la NASA resultando ganador nacional junto a otros tres proyectos, lo que le dio derecho a participar en el concurso internacional.

Índice

1 Introducción

El presente documento tiene como objetivo recoger toda la información de este Proyecto Fin de Carrera (PFC) titulado "Energ4People: Sistema de soporte a la toma de decisiones en el dominio de las energías renovables orientado al usuario no experto".

1.1 Contexto del proyecto

1.1.1 Beca de colaboración en el IAAA

El PFC presentado en este documento surge dentro de las líneas de investigación del Grupo de Sistemas de Información Avanzados (IAAA); concretamente como parte del trabajo de investigación predoctoral del director del proyecto Naveen Sidda.

Su trabajo investiga el uso de Sistemas de Información utilizados habitualmente en los procesos de toma de decisiones en las empresas para ayudar a identificar y desarrollar nuevas oportunidades en el dominio de las energías renovables.

1.1.2 Energy2People y el Space Apps Challenge

A principios del mes de Abril de este mismo año llegó la noticia de que la NASA celebraba en Madrid junto a otras 82 ciudades de todo el mundo el concurso "Space Apps Challenge". El objetivo de este concurso era resolver retos propuestos por la NASA.

El fin de semana del 20 de Abril, presenté el proyecto Energy2People, un explorador de energías renovables, prototipo de Energy4People y fruto del trabajo de los primeros meses del proyecto.

Un grupo de profesionales y estudiantes de distintas ramas de la ingeniería se mostraron interesados en participar en este proyecto. El 21 de Abril, el proyecto Energy2People se proclamó ganador del concurso en Madrid y se clasificaba para la final internacional.

Posteriormente, los medios de comunicación mostraron gran interés en este proyecto. Entrevistas en radio y televisión me sirvieron para entender mejor el contexto de este proyecto: Los ciudadanos no conocen bien que hay detrás de las energías renovables, pero quieren un futuro sostenible que pasa por ellas. El proyecto debía continuar.

El Anexo I recoge más información del proyecto Energy2People.

1.2 Contexto tecnológico

La toma de decisiones recae cada vez de una manera mayor sobre distintas herramientas de tecnologías de la información. La razón de esto es que cada día hay más datos sobre los que decidir. Y un mal tratamiento de estos datos lleva a una información poco fiable que conduce a una pobre toma de decisiones.

Dentro del contexto de la toma de decisiones, surge el concepto de **BI** (Business Intelligence) o inteligencia de negocios. BI es un conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización. Dicho de otra manera, **BI facilita a la empresa la toma de decisiones**. BI sirve como término paraguas de otras tecnologías como **OLAP** (On-Line Analytic Processing) o **minería de datos**.

OLAP es una solución tecnológica cuyo objetivo es la consulta de grandes cantidades de datos. Para ello se utilizan estructuras multidimensionales o cubos OLAP que contienen datos resumidos de grandes bases de datos. Al contrario que los sistemas transaccionales, OLAP está optimizado para obtener una respuesta rápida al realizar consultas gracias al modelo multidimensional. Estas grandes cantidades de datos suelen estar almacenadas en un **DW** (Data Warehouse).

Un DW es una colección de datos orientada a un determinado ámbito o negocio, integrado, no volátil y variable en el tiempo que ayuda a tomar decisiones en la entidad que se utiliza. La pregunta ahora reside en cómo integrar todos los datos en un DW. La respuesta recibe el nombre de proceso **ETL** (Extract, Transform, Load).

Un proceso ETL permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos, limpiarlos y cargarlos en un DW.

Es importante mencionar que en el proyecto Energy4People **se utilizan datos georreferenciados**, y que por esto el DW pasa a ser un **SDW** (Spatial DW) y OLAP se convierte en un **SOLAP** (Spatial OLAP).

Para completar la pila de tecnologías de BI (ilustradas en la figura 1.2) falta la cima, la visualización. Esta permite al analista de negocios ser capaz de **tomar decisiones**. Existen muchas tipos de visualizaciones (diagrama de barras, información tabular,…), pero lo más acertado para visualizar datos geográficos es un mapa.

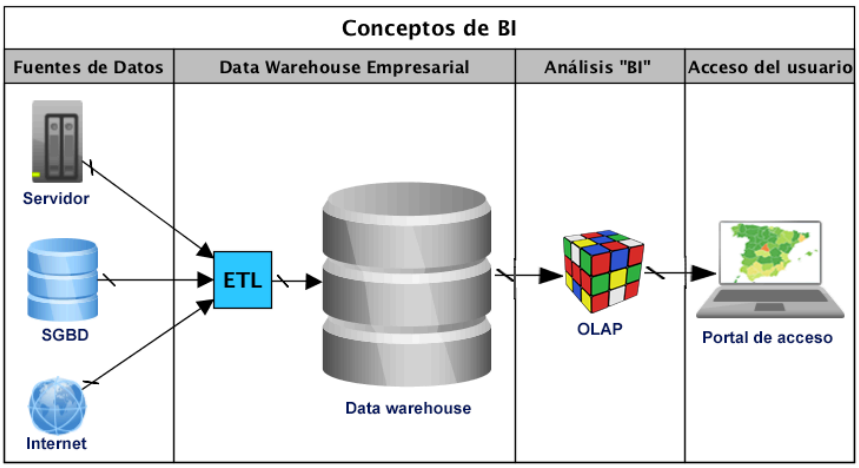

Figura 1.1: Conceptos de BI. Partiendo de diversas fuentes de datos se crea un data warehouse a través de un proceso ETL. Esta información será explotado a través de un SOLAP, que recuperará los datos para ser visualizados.

Los mapas como transmisores de información georreferenciada tienen un enorme potencial. La toma de decisiones en un mapa se asume como una **gran ventaja competitiva**, pero también presentan problemas a la hora de mostrar información. Por ejemplo, cuando hay gran cantidad de datos en una determinada escala, se puede perder el foco de la información y empobrecer el proceso de la toma de una decisión. La **generalización cartográfica** surge como una solución a este problema.

La generalización cartográfica es el proceso de abstracción de toda la información disponible para mostrar, en aquella que realmente necesaria para un propósito particular. La figura 1.2 muestra de manera gráfica esta definición.

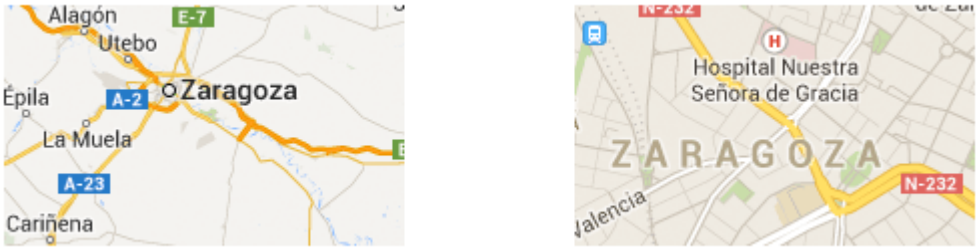

Figura 1.2: Ejemplo de generalización cartográfica en Google Maps. La imagen de la izqda. muestra Zaragoza tras realizar 6 zooms. La de la dcha. tras realizar 12 zooms.

Finalmente, mencionar el papel del **Usuario** dentro del contexto tecnológico. Su rol ha cambiado enormemente, y ahora, gracias a la tecnología, es capaz de alcanzar metas que nunca habría pensado. Algunas empresas se han dado cuenta de este cambio y lo usan para aumentar su valor corporativo (Tapscott & Williams, 2006). **El usuario guía el presente y el futuro de la tecnología**.

1.3 Objetivos y alcance del proyecto

Como se ha mencionado, el proyecto Energy4People proviene de una línea de investigación que propone el uso de Sistemas de Información orientados a la toma de decisiones en el dominio de las energías renovables.

Además, la importancia de los ciudadanos como inversores en este tipo energías, les convierte en potenciales clientes a los que hay que facilitar la toma de la decisión de invertir en un sector complicado para una persona no experta.

Por ello, este proyecto propone una solución que acerque el mundo las energías renovables a los ciudadanos, a través de un prototipo de producto que permita a un hipotético nicho de mercado, en este caso, las empresas de energías renovables, ofrecer un mejor servicio a sus potenciales clientes.

Los objetivos a alcanzar son los siguientes:

- Realizar un estudio previo y generar documentación referenciada de los conceptos teóricos relacionados con BI, así como estudiar las distintas alternativas tecnológicas que implementan dichos conceptos.
- Conceptualizar los requisitos funcionales y no funcionales que llevarían los conceptos teóricos de una investigación a un prototipo de un producto de innovación.
- Diseñar y desplegar un spatial data warehouse.
- Diseñar e implementar un proceso ETL.
- Diseñar e implementar un cubo SOLAP que permita el acceso a los datos a través de un modelo multidimensional.
- Diseñar e implementar una aplicación web para permitir la consulta y visualización de los datos en un mapa.
- Realizar un estudio del estado del arte en las aplicaciones de visualización de recursos energéticos así como la manera en la que empresas de energías renovables se sitúan en la web.

1.4 Estructura del documento

El documento del PFC se estructura en los siguientes capítulos:

I. Trabajo realizado: Análisis, diseño, implementación y tests del proyecto. Para comenzar, se realiza un **estudio del estado del arte** en el mundo de las herramientas de análisis y visualización de energías renovables. Distintas organizaciones están interesadas en el desarrollo y divulgación de este tipo de herramientas, pero lo cierto es que se aprecian ciertos problemas. Y estos problemas llevan a la sección de **análisis** del proyecto. Para guiar estos problemas a una solución basada en el trabajo predoctoral de Naveen Sidda, se conceptualiza un **producto mínimo viable (MVP) sobre una hipótesis en el mercado de las energías renovables**. Una vez definido este producto, se definen unos **escenarios de caso de uso** que permiten aclarar los **requisitos funcionales** y **no funcionales** del MVP. El **diseño** de la solución llevará al diseño de dos aplicaciones. A continuación se presenta la **implementación** con un breve resumen de las tecnologías usadas en cada una de las aplicaciones y la razón de su elección. Por último, se presentan los tests.

II. Conclusiones: **Resultados**, **trabajo futuro** y la **conclusión** pueden ser encontrados en esta sección.

III. Gestión del proyecto: **Scrum** es la **metodología** elegida para la realización del proyecto. Planificación y herramientas también se presentan en esta sección.

IV. Bibliografía: Libros, artículos y documentos utilizados en el desarrollo del proyecto.

Anexos: Profundiza en distintos aspectos del proyecto. Además, sirven como referencia técnica a Naveen Sidda, y de apoyo en nuestra comunicación durante el proyecto. Por esto, las secciones técnicas y teóricas del anexo están en inglés.

2 Trabajo realizado

El trabajo realizado durante el proyecto se estructura en un estudio del estado del arte (*sección 2.1*), continuando con el descubrimiento y definición de los requisitos del sistema (*2.2*), el diseño de la solución (*2.3*), su implementación (*2.4*) y por último la verificación de su correcto funcionamiento mediante tests (*2.5*). Todo esto se puede complementar con la documentación generada en los anexos y que sirve de referencia técnica al director de este proyecto.

2.1 Estudio del estado del arte

El mundo de las energías renovables está viviendo su particular crisis. Por esto, grandes organizaciones como IRENA o consultoras de energías renovables como ENNERA, buscan a través de Internet atraer a nuevos clientes. Una de las mayores apuestas en este sentido es "The Global Solar And Wind Atlas". Esta iniciativa promueve **el mayor proyecto realizado para evaluar el potencial de las energías renovables en una escala global**. Para esto, se ha desarrollado una completa plataforma web para **ayudar a la planificación de políticas enérgicas y atacar a posibles inversores en mercados energéticos emergentes**. (IRENA_Atlas_brochure, 2013)

Por su parte, las empresas consultoras de energías renovables están desarrollando **nuevas estrategias para acercase a clientes potenciales**. Técnicas **SEM** (Search Engine Marketing) para ser mas visibles en Internet o calculadoras verdes, en las que es posible calcular la rentabilidad de una inversión, son algunas de las herramientas usadas por las empresas para captar nuevos clientes.

Pero en la web no todo son aplicaciones y empresas buscando nuevos clientes. También se puede encontrar una gran cantidad de datos libres (Volk, 2011). Estos datos están almacenados muchas veces en ficheros que no facilitan su explotación. Los datos de energías renovables no escapan a estos problemas, y distintas fuentes proveyendo datos con diferentes estructuras son un hecho habitual.

Centrado de nuevo la atención en el estudio de aplicaciones más profesionales, NREL e IRENA surgen como organizaciones protagonistas que proveen de aplicaciones, surgidas en el año 2013, y que se estudian a continuación.

NREL (The National Renewable Energy Laboratory) es el laboratorio primario del Departamento de energía e investigación de eficiencia energética de los Estados Unidos. NREL provee de una aplicación web con un mapa donde se pueden mostrar datos de radiación solar y velocidad del viento en EEUU.

La Figura 2.1, muestra un problema observado en esta aplicación a la hora de realizar el análisis de recursos energéticos. Al hacer zoom, llega un momento en el que el área mostrada en el mapa carece de total valor a la hora de tomar una decisión. Es decir, los datos mostrados están a una sola escala. No hay generalización cartográfica.

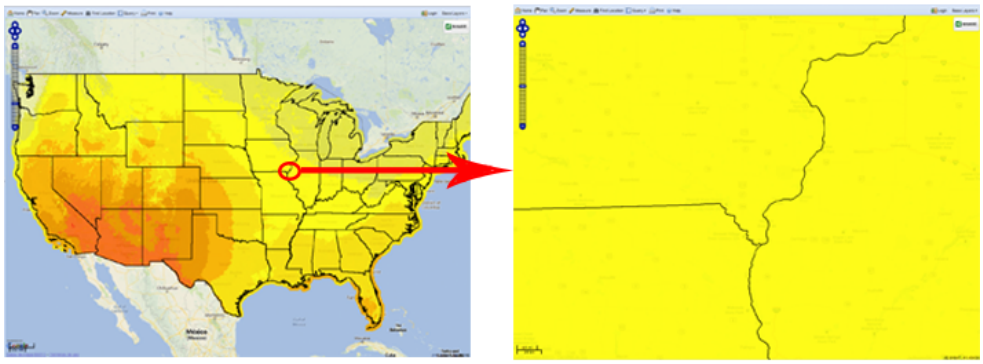

Figura 2.1: Aplicación NREL. La imagen muestra dos momentos de la interacción con la aplicación. La figura de la izquierda muestra la radiación solar en el mapa antes de ejecutar ninguna acción. La imagen de la derecha muestra la información mostrada en el mapa tras aplicar 4 zooms.

Por su parte, IRENA lanzó su aplicación, "The Global Solar And Wind Atlas", con el apoyo de países como España, Alemania o Brasil. Esta aplicación tiene como novedad, sobre la aplicación de NREL, la integración en un mismo mapa de distintas fuentes de datos de un mismo recurso energético. La Figura 2.2 muestra el resultado de mostrar las primeras cinco fuentes de datos. La coherencia entre las fuentes es inexistente. Algunas son anuales, otras mensuales, etc. También carece de generalización cartográfica.

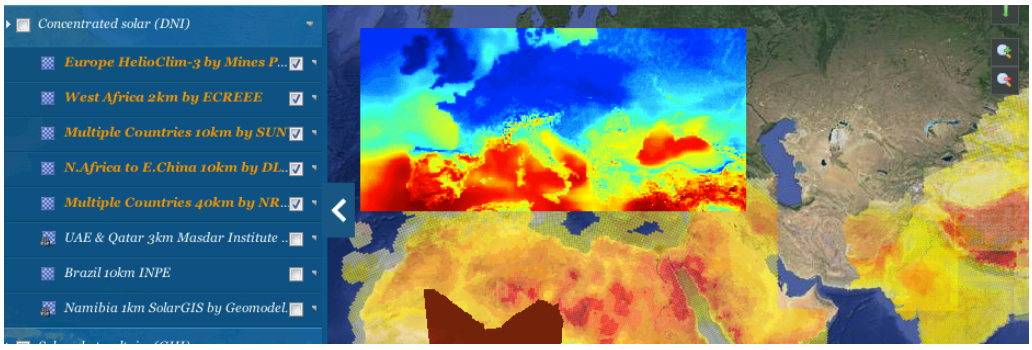

Figura 2.2: "The Global Solar And Wind Atlas" de IRENA. La imagen muestra el resultado de mostrar datos de un mismo recurso procedentes de distintas fuentes.

2.2 Análisis del problema

A continuación se presenta el análisis del problema. Requisitos del producto, escenarios de uso y requisitos del sistema guiarán a acotar el problema y mostrar cómo es realizado el análisis. Más detalles se pueden encontrar en el Anexo A (en inglés).

2.2.1 Requisitos del producto

Energy4People es un proyecto que surge a raíz del trabajo de una investigación predoctoral. El primer objetivo del proyecto es conceptualizar los requisitos de un sistema que permitan validar técnicamente conceptos de este trabajo, convirtiendo una investigación en un producto de innovación. Para guiarse en esta conceptualización se usa una herramienta de validación de modelos de negocio llamada *Lean Canvas*. Con ella, partiendo de una hipótesis en el mercado de las energías renovables, se define un posible producto. Y a partir de ahí, definir sus requisitos funcionales y no funcionales.

Hipótesis sobre la evolución del sector

"Las empresas de energías renovables van a apostar por la transformación de datos en información, para guiar a los ciudadanos en la decisión de invertir en Energías renovables."

La figura 2.3 muestra el Lean Canvas desarrollado en este proyecto. Su misión es **ofrecer una visión rápida de un modelo de negocio y agilizar posibles cambios**.

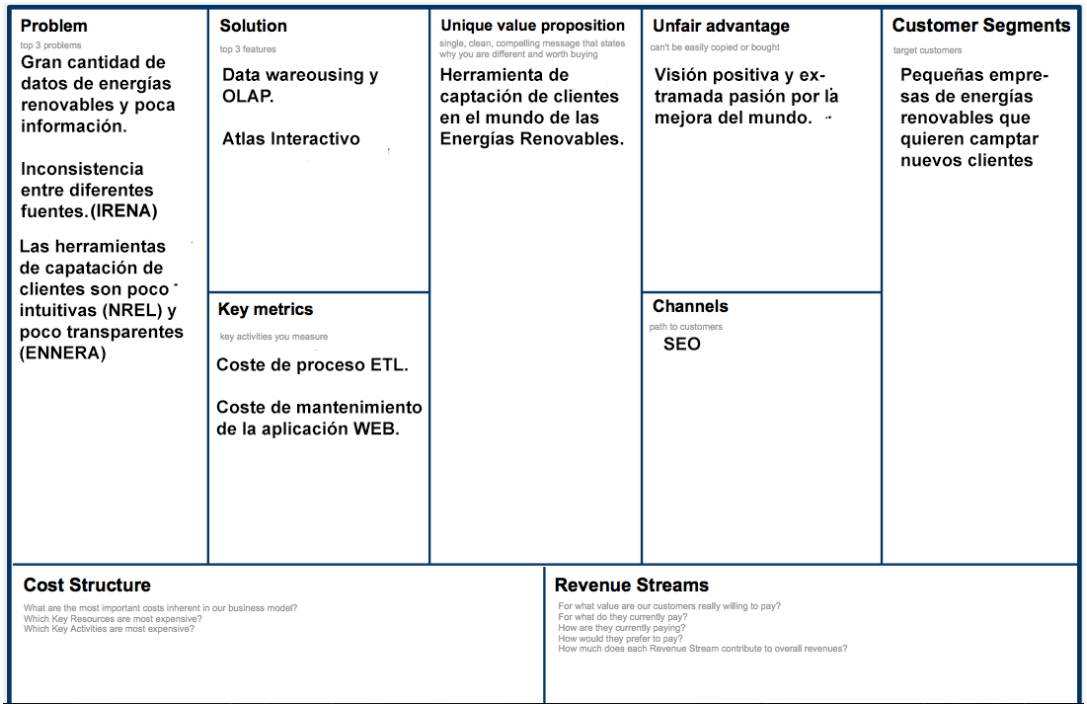

Figura 2.3: Lean Canvas final de Energy4People.

2.2.2 Escenarios de casos de uso

Para ir dando forma al producto, se presentan a continuación dos escenarios de caso uso que presentan los actores del sistema, sus metas y sus acciones para conseguirlas.

1. Servicios de consultoría en el sector de las energías renovables

El señor Jobs tiene una empresa de consultoría en energías renovables. Lleva tiempo buscando nuevas vías para conseguir clientes. Un día, visitando el sitio web de la NASA para acceder a datos de energías renovables, se le ocurre la idea de visualizar dichos datos en un mapa. Que se puedan visualizar los datos por años, por meses de un determinado año y por días de un determinado mes tal y como muestra la Figura 2.4. Es decir, profundizar en la información a través del tiempo.

Figure 2.4: Ejemplo de navegación temporal en la aplicación imaginada por el señor Jobs.

También se le ocurre que al hacer zoom en el mapa los datos vayan cambiando de escala, tal y como muestra la figura 2.5, para mejorar la comprensión de sus clientes. Es decir, aplicar técnicas de generalización cartográfica "jugando" con los zooms y la resolución de los datos. Se puede ver como la posibilidad de profundizar en la información espacial a través del mapa.

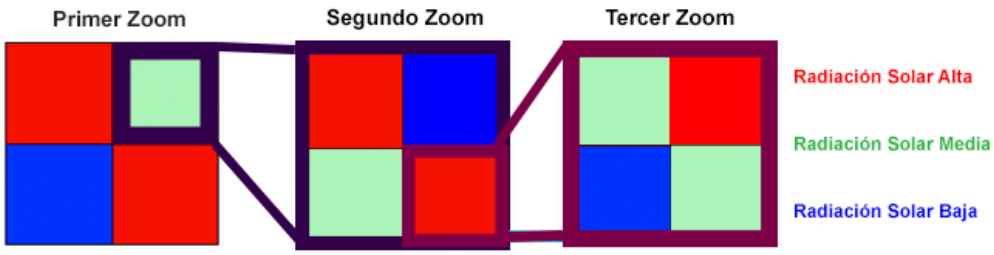

Figure 2.5: Modelo de datos multi-escalar. Cada zoom muestra nueva información.

Por último, es deseable la posibilidad de aplicar modelos sobre esos datos.

El señor Jobs se pone en contacto con la empresa del señor Espejo.

2. Business Intelligence al servicio del cliente

El señor Espejo dispone de una empresa proveedora de sistemas de BI. Para su próximo proyecto necesita un software que le permita integrar datos procedentes de un sitio web en un repositorio central orientado al análisis. Además puesto que la resolución de los datos es baja, un proceso de interpolación va a ser necesario.

Todas las técnicas que él y su equipo conocen se traducirán en un deliberarle: **una aplicación web que mostrará datos de energías renovables alimentándose de un repositorio** creado con el software descrito en el párrafo anterior. Además, gracias a este mismo software, se ofrecerá un servicio de mantenimiento y actualización del repositorio. La elección de una aplicación web permitirá acceder a muchos más clientes, tal y como el señor Jobs tiene en mente.

2.2.3 Requisitos funcionales

Los escenarios presentados en la sección anterior ofrecen el paso previo a la definición de los requisitos funcionales. En la búsqueda de un acercamiento al cliente no técnico, se han definido unas historias de usuario que pueden ser encontradas en el Anexo A. Esta manera de presentar los requisitos, extensamente utilizada en las metodologías ágiles, ha facilitado la comunicación con el director del proyecto, Naveen Sidda, cuya formación es la Ingeniería Civil. Estas historias han sido el paso previo a la definición final de los escenarios y los requisitos funcionales del sistema. Hay que especificar que los requisitos que se presentan a continuación son el resultado de cuatro iteraciones. La evolución de la investigación que realiza el director del proyecto, que actúa como un cliente, y su evaluación del producto tras finalizar cada iteración, han marcado los requisitos o cambios de la versión a entregar en la siguiente iteración. La tabla 2.1 muestra los requisitos funcionales del sistema.

Id	Nombre	Descripción
1	Repositorio central	El sistema almacenará datos de energías renovables.
2	Aplicación web	El sistema permitirá acceder a los datos de manera remota a través del navegador.
3	Visualización espacial multi-escalar	El sistema permitirá visualizar los datos en un mapa y navegar entre distintas resoluciones ante cambios de zoom.

Id	Nombre	Descripción
4	Visualización temporal jerárquica	El sistema permitirá acceder a los datos navegando por año, meses y días de manera jerárquica.
5	Aplicación de modelos	El sistema permitirá aplicar uno o más modelos.
6	Interpolación	El sistema permitirá interpolar los datos extraídos.
7	Web scraper	El sistema permitirá obtener datos mediante técnicas de web scraping.
8	Crear y almacenar procesos ETL.	El sistema permitirá implementar y almacenar un proceso de extracción, transformación y carga de datos.

Tabla 2.1: Requisitos funcionales del sistema.

2.2.4 Requisitos no funcionales

Al igual que ocurre con los requisitos funcionales, los requisitos no funcionales han sido extraídos de historias de usuario que pueden encontrarse en el Anexo A. La tabla 2.2 muestra los requisitos no funcionales del sistema.

Id	Nombre	Descripción
1	Fuentes NASA	La fuente de datos del sistema será del sitio web de la NASA.
2	Usar Google Maps	El mapa para mostrar los datos en el sistema será Google Maps.
3	Chrome y Mozilla como navegadores	El sistema deberá funcionar de manera correcta el los navegadores Google Chrome y Mozilla Firefox
4	Formato KML	El sistema exportará los datos en formato KML.

Tabla 2.2: Requisitos no funcionales del sistema.

2.2.5 Requisitos de usabilidad

Una de las metas de este proyecto es proveer una aplicación web donde usuarios no expertos visualizan información de energía renovable para obtener mayor información y poder tomar la decisión de invertir en energías renovables.

La interfaz gráfica es realmente importante en los sistemas de soporte a la decisión (Power, 2002). Mostrar sólo la información necesaria de la manera adecuada es fundamental para proporcionar una buena toma de decisiones. Por ejemplo, en el contexto de este proyecto la radiación solar y velocidad del viento son datos numéricos que se pueden mostrar en una tabla, pero esto no se considera nada intuitivo para un usuario no experto. Un mapa y el uso de una escala de colores resultan a priori más intuitivos.

Por lo explicado en el párrafo anterior, después de cada iteración en el desarrollo del proyecto, un usuario debe realizar algunas tareas en la aplicación. Estas tareas se consideran representativas del uso que lleva a cabo un usuario las primeras veces que utiliza el sistema.

Finalmente, se generara un documento que recoge los resultados del test de usabilidad. Los documentos generados se encuentran en el Anexo D.

2.2.6 Conclusión del análisis

Una vez definidos los requisitos del sistema, se empieza a intuir que para el diseño de la solución se van a tener que utilizar arquitecturas que favorezcan la lectura de grandes cantidades de datos. De esta manera, el tiempo que se tarda en visualizar los datos en el mapa también es menor.

Aunque se dan más detalles en la sección de diseño, una aproximación que permite guiar el diseño de la solución es el **uso de técnicas de BI para diseñar y posteriormente implementar el sistema**.

Por un lado, un **DW** proveería de un repositorio central de datos. Además, el acceso temporal que se define en los requerimientos, así como **la visión de los datos en múltiples escalas se relaciona con el modelo multidimensional que aporta un cubo OLAP**. En este caso al haber datos espaciales como la latitud y la longitud, se trata de un cubo **SOLAP** y de un **SDW**.

Para conseguir la solución propuesta en el párrafo anterior, **un proceso ETL tiene que ser ejecutado** para extraer datos de la web de la NASA, transformarlos (en el caso que haga falta) y por último almacenarlos en el SDW.

2.3 Diseño de la solución

Partiendo de las conclusiones del análisis, está justificado el uso de técnicas de BI para diseñar el sistema. Arquitectura, diseño del proceso ETL y del SDW son estudiados en próximas secciones. Además, el diseño de la aplicación web también es examinado. Más detalles del diseño de la solución se pueden encontrar en el anexo B. (En inglés)

2.3.1 Arquitectura

Energy4People es un sistema compuesto por dos aplicaciones. Por un lado el proceso **ETL**, configurado y ejecutado por el proveedor para implementar un repositorio de datos. Por otro, una aplicación web con la que los usuarios pueden acceder a los datos previamente almacenados en el repositorio. Para integrar ambos sistemas, se utiliza un estilo de arquitectura de integración en la que **la comunicación entre ellos se realiza a través de una base de datos** (Hohpe & Woolf, 2007). En este caso, como se muestra en la Figura 2.6, la base de datos de comunicación es el SDW.

Figura 2.6: Integración de las dos aplicaciones a través del SDW.

Como se define en los requisitos definidos dentro de la fase de análisis, la solución incluye un acceso a los datos a través del navegador. Para resolver este requerimiento se va a proceder al diseño e implementación de una aplicación web.

La aplicación web del proyecto Energy4People presenta una **arquitectura N-capas**. Esta arquitectura se considera adecuada ya que provee de un diseño totalmente modular que permite poder desarrollar cada una de las capas de manera independiente. Cuatro capas se pueden distinguir en la aplicación: capa de presentación, capa de negocio, capa de servicio y capa de datos.

Aunque estas capas se explican más detalladamente en las siguientes secciones, una visión global de la arquitectura se puede ver en la Figura 2.7.

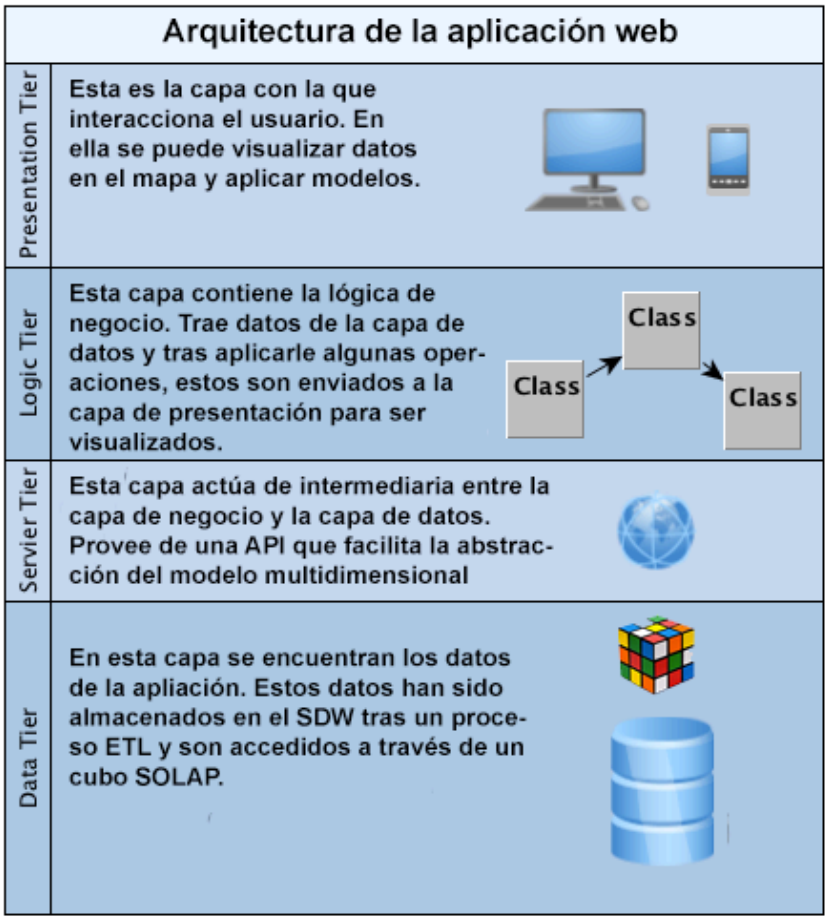

Figura 2.7: Arquitectura global de la aplicación web de Energy4People.

Cabe destacar los **conceptos de BI** que aparecen en esta arquitectura y que se desarrollan en las próximas secciones.

Por un lado la capa de presentación presenta una interfaz en la que se pueden visualizar los datos de una manera intuitiva. La **visualización** es parte fundamental de un sistema de BI. Por otra parte y no menos fundamental, se dispone de un **SDW** y un servidor **SOLAP** en la capa de datos[1].

La lógica de negocio, tal y como ocurre en muchos sistemas de BI, se utiliza para "enriquecer" los datos. Es decir, cuando los datos llegan desde la capa de datos se le puede aplicar diferentes modelos, o incluso transformaciones en formatos que puedan ser visualizados a través de una determinada herramienta. De esta manera el usuario del sistema puede tomar decisiones de una manera mucho más intuitiva.

[1] Tras evaluar diferentes alternativas, el SDW y el SOLAP se implementan por su mejor rendimiento sobre sistemas de bases de datos relacionales sin soporte espacial nativo, gracias a las características de los datasets utilizados.

2.3.2 Diseño del proceso ETL

Atendiendo a los requisitos del sistema, es necesario implementar un repositorio con datos de energía renovable. Además hay que usar como fuente de datos el sitio web de la NASA. Uno de los problemas encontrados en los datos de este sitio web es que son de baja resolución. Concretamente cada dato cubre 1º de latitud por un 1º de longitud.

La zona de staging, dentro de un proceso ETL, se utiliza para resolver problemas como el descrito en el párrafo anterior. En ella se aplican transformaciones como puede ser la interpolación de los datos una vez han sido extraídos.

Una vez los datos han sido transformados en la zona de staging, son cargados en el SDW. Este, presenta un esquema, que será descrito en las próximas secciones, y que está orientado al análisis de grandes cantidades de datos.

La Figura 2.8 muestra el diagrama de contexto utilizado para modelar el proceso ETL. Este diagrama solo muestra a grandes rasgos el proceso completo. La notación pertenece a los Diagramas de Flujo de Datos (DFD).

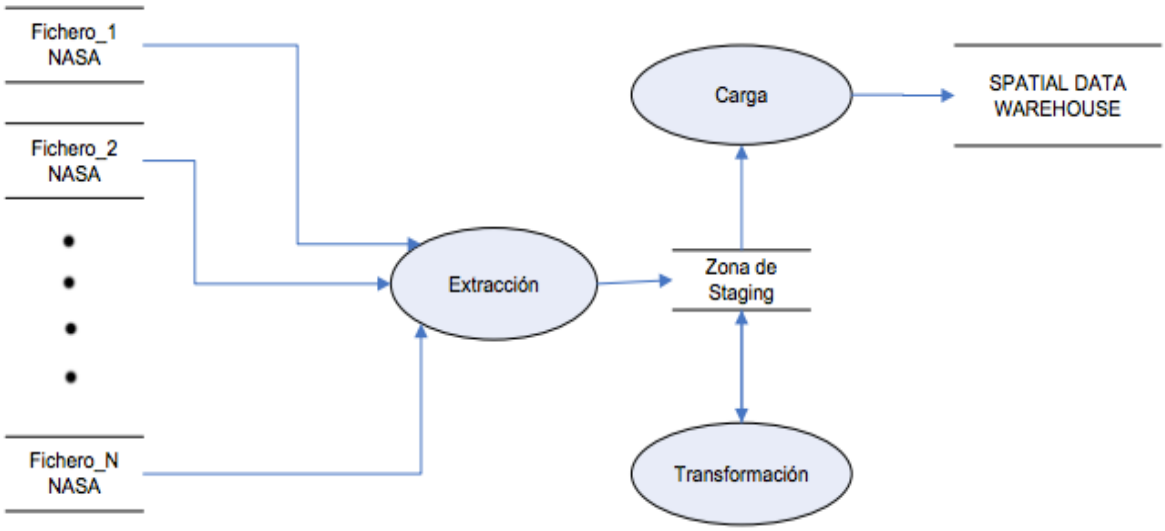

Figura 2.8: Diagrama de contexto modelado usando DFD.

2.3.3 Diseño de la capa de datos

El diseño de un SDW (Spatial Data Warehouse) es diferente del diseño de una base de datos relacional. Esto tiene sentido partiendo de que SDW es usado principalmente para análisis de datos. Por tanto, el modelo ha de estar optimizado para la lectura de grandes cantidades de datos. A continuación se presentan el diseño conceptual y lógico del SDW.

Diseño conceptual y lógico

El diseño conceptual es el primer paso a la hora de diseñar el SDW. Los datos son representados de acuerdo al modelo dimensional. Este es el modelo que implementa el cubo SOLAP. Para entenderlo mejor, es necesario explicar brevemente 4 conceptos:

- **Hecho**: Es un concepto que es relevante para la toma de una decisión. En el contexto del proyecto los hechos son **la radiación solar y la velocidad del viento**.

- **Medida**: Es la propiedad numérica de un hecho. En este caso es **la cantidad de radiación solar y la velocidad del viento**.

- **Dimensión**: Es una propiedad de un hecho descrito con respecto a un domino finito. Es decir, es una propiedad que describe un hecho. Por ejemplo, en el contexto de este proyecto, las dimensiones son el tiempo (dimensión temporal), y la localización (dimensión espacial) del recurso energético. A su vez la dimensión espacial está formada por pares latitud/longitud, así que **el recurso energético está definido realmente por 3 dimensiones, tiempo, latitud y longitud.** Por simplificar, y teniendo en cuenta que en este dominio latitud y longitud van siempre juntas, se agrupan como dimensión espacial.

- **Jerarquía**: Son grupos de atributos dentro de una dimensión que siguen un orden preestablecido. Las jerarquías definen cómo los datos son agregados desde los niveles más bajos de la jerarquía hacia los más altos. En el contexto del proyecto, **tanto la dimensión temporal como la espacial tienen jerarquías**. La primera, aplica una jerarquía entre día, mes y año. La segunda aplica una jerarquía entre latitudes y longitudes de distinta resolución.

La Figura 2.9 muestra el modelo conceptual, diseñado en el proyecto, usando una notación basada en el modelo relacional (Malinowskis, 2008). De esta manera se intenta que no sea excesivo el salto entre el modelo relacional y el modelo dimensional. Observando el modelo descrito en la imagen se pueden apreciar todos los conceptos mencionados anteriormente. También se observa que la dimensión espacial (*SpatialDimension)* está compuesta por dos dimensiones, una que guarda la latitud de los datos, y otra que guarda la longitud. La palabra *All* es típica en los modelos multidimensionales y representa un nivel que contiene a todos los datos, es decir, si por ejemplo, la dimensión temporal (*TemporalDimension)* tiene datos de 20 años, *All* representa una agregación de todos los datos de esos 20 años.

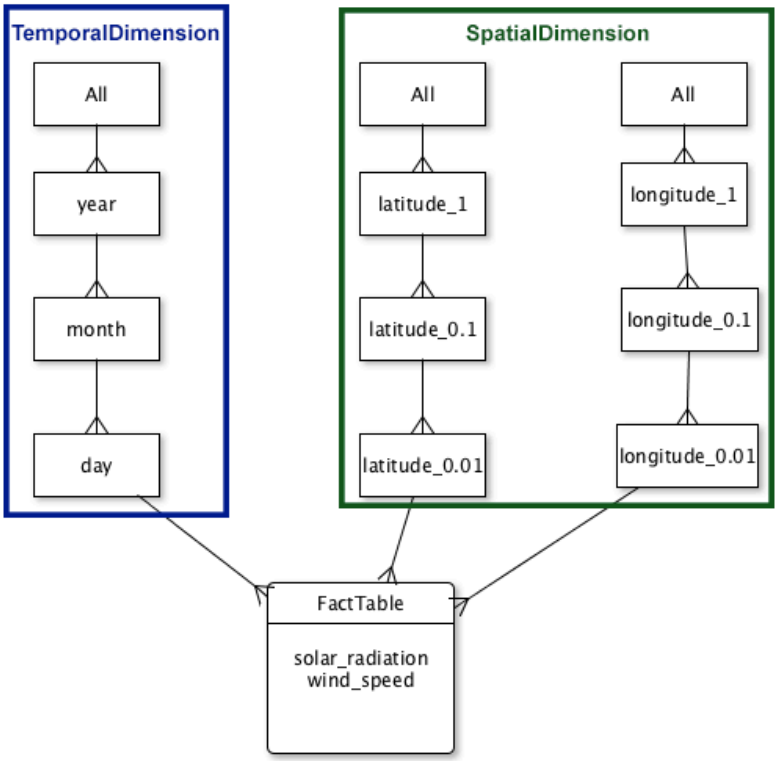

Figura 2.9: Modelo conceptual del SDW.

A continuación se presenta el modelo lógico que acerca en mayor medida a los conceptos del modelo relacional. Para diseñarlo se pueden elegir entre varios tipos de esquemas que se ajustan mejor o no según sean las necesidades de SDW. El elegido es el esquema de estrella. Este permite un acceso más rápido a los datos por la **no** aplicación de formas normales en el modelo de datos. La Figura 2.10 muestra el diseño realizado en el proyecto. Dos tablas de dimensiones, y la tabla de hechos definen este modelo. Si hubiera más dimensiones, la forma de estrella sería más apreciable.

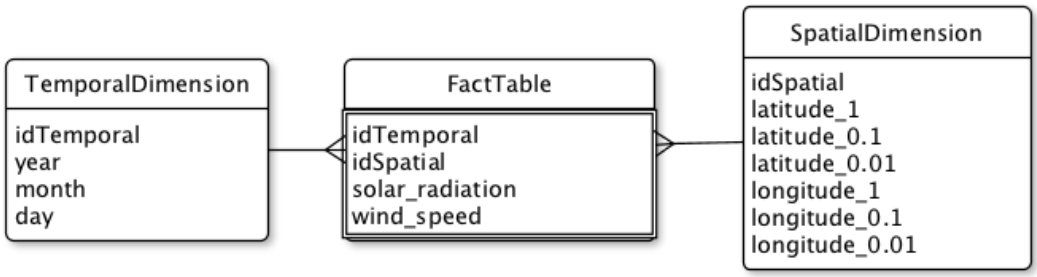

Figura 2.10: Modelo lógico de SDW

2.3.4 Diseño de la capa de negocio

A continuación se presenta el diseño de la estructura y comportamiento de la lógica de negocio de la aplicación web. Se utiliza notación UML 2.0 simplificada.

Una de las capas dentro de la aplicación web es la **capa de negocio**. En ella se ejecutan procesos que posibilitan el cumplimiento de los requisitos del sistema. Concretamente, esta capa recibe las peticiones de la capa de presentación, y trae los datos de la capa de datos para aplicarle los modelos correspondientes y devolver una respuesta a la capa de presentación.

Para el diseño de esta capa se han utilizado dos patrones de diseño: **DAO** (Data Access Object) y **Facade** (Gamma, Helm, Johnson, & Vlissides, 1998). En este caso las clase *Facade* actúa como interfaz abstrayendo de toda la lógica de negocio que hay tras ellas al controlador de la aplicación. El patrón DAO provee de una interfaz entre la lógica de negocios y la capa de datos. En este contexto, la clase *RenewableEnergyDaoImpl* está ocultando al resto del sistema el acceso a un servicio que le provee de los datos de energías renovables que le hayan sido requeridos. La Figura 2.11 muestra un diagrama de clases simplificado de la lógica de negocio.

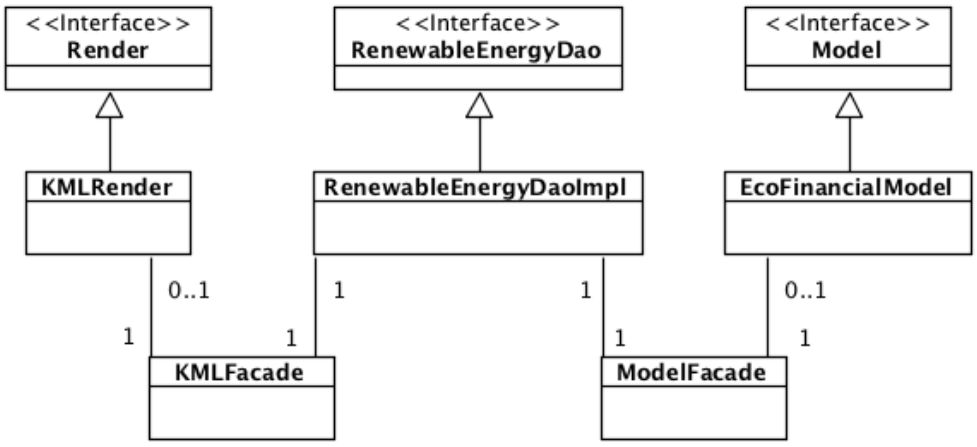

Figura 2.11: Diagrama de clases simplificado usando notación UML 2.0.

Las clase *KMLFacade* oculta la lógica que transforma los datos en ficheros KML. Esta transformación se hace en la clase *KMLRender*. Por otro lado, la clase *RenewableEnergyDaoImpl* se encarga de traer los datos de la capa de servicio. Por último la clase *ModelFacade* oculta la lógica que aplica la clase *EcoFinancialModel* para transformar los datos en resultados de aplicar un determinado modelo.

Para modelar la **interacción entre el usuario y el sistema**, se usa el diagrama de secuencia de la Figura 2.12. Ésta describe a un actor que interactúa con la aplicación web. Cada una de las clases de la lógica de negocio mostradas en la Figura 2.11 se comunican y realizan su funcionalidad de manera transparente al usuario,.

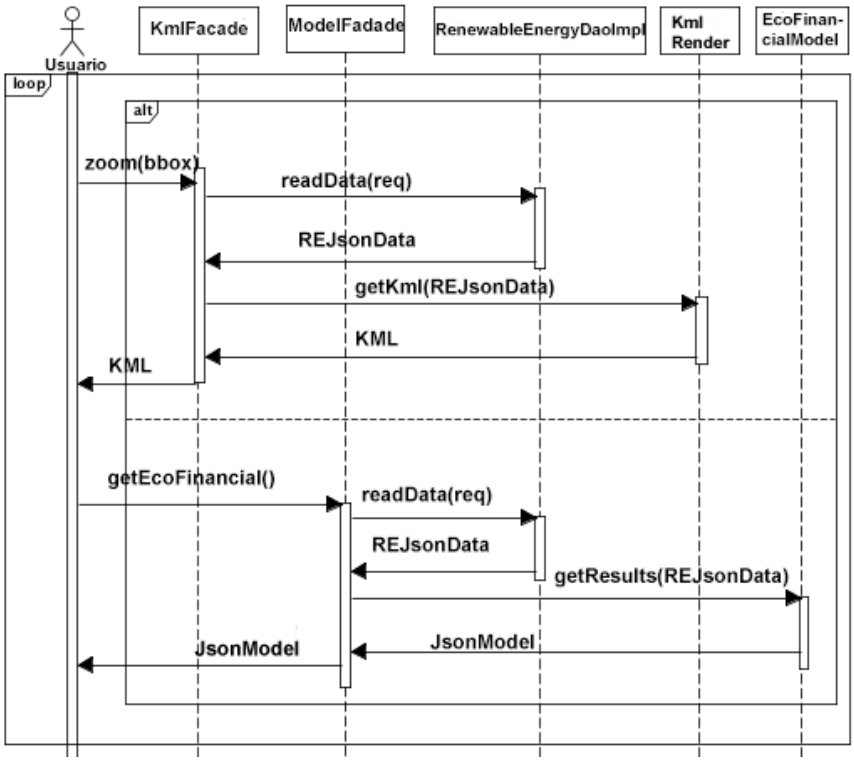

Figura 2.12: Diagrama de interacción de la aplicación web usando notación UML 2.0.

2.3.5 Conclusión del diseño

El uso de técnicas de BI contempla la posibilidad de diseñar un proceso ETL que extraiga datos de la NASA y los cargue en un repositorio centralizado (SDW). Además, gracias a la definición del modelo dimensional (cubo SOLAP) que se propone, se pueden realizar análisis multi-escalar de los datos. Esto sería una manera de implementar la generalización cartográfica. También gracias a este modelo se puede navegar por los datos a través de su dimensión temporal de manera jerárquica.

2.4 Implementación del sistema

Tras realizar el diseño de la solución, se procede a la implementación de las dos aplicaciones. Para explicar las tecnologías usadas en cada una, el criterio de su elección frente a otras alternativas y su funcionalidad dentro del sistema se utiliza la tabla 2.3. La columna izquierda de la tabla nombra las tecnologías evaluadas para la implementación. Las que aparecen en color **verde** son elegidas durante una determinada fase del proyecto. Las que aparecen en color **rojo** han sido descartadas. La columna derecha de la tabla describe brevemente todo lo referente al proceso de selección e implementación.

Cabe destacar que el uso de **Java** como lenguaje de programación guía en la elección de varias tecnologías para facilitar la integración de distintas partes del sistema. La elección de este lenguaje se debe a su potente y extensa API junto a la gran cantidad de librerías implementadas por su comunidad y que proporcionan gran cantidad de funcionalidades. Además al tratarse de un lenguaje orientado a objetos permite la aplicación de patrones de diseño, que en mi experiencia como desarrollador me ha dado un gran resultado.

Tecnologías	Descripción y razón de su selección o rechazo
Aplicación ETL	
Geokettle **Spring Batch** **PDI (Kettle)** **Talend**	Geokettle se elige como herramienta ETL por su potente soporte espacial y por su total integración con Java. Este soporte espacial, hizo que prevaleciera sobre otras alternativas como Kettle, Talend y Spring Batch. Además, Geokettle provee de un repositorio de transformaciones a la que se pueden agregar nuevas tareas definidas por el usuario. Por ejemplo, scrapear o interpolar.
JWebUnit **JAI** **HttpClient** **java.net** **ArcGIS**	Para la implementación del web scraper se hace uso de la librería JWebUnit. Otras alternativas como HttpClient de Apache, o la librería estándar de Java, fueron revisadas pero ninguna alcanzaba el nivel de abstracción de JWebUnit a la hora de scrapear una página web. Para interpolar se utiliza la librería JAI (Java Advance Imaging). Esta librería permite interpolar matrices de valores numéricos, aumentando de esta manera su resolución. El uso de funciones de interpolación geoestadística, proveídas por ArcGIS, como el Kriging, se deja para futuras versiones.

Tecnologías	Descripción y razón de su selección o rechazo
Aplicación WEB	
SDW	
Postgresql **Postgis** **LucidDB**	Postgresql es el SGBD elegido para implementar el SDW. Dos son los motivos que llevan a tomar esta decisión. El primero es el soporte espacial que provee gracias a Postgis. El uso de la extensión espacial fue utilizada durante la última fase del proyecto junto a Geomondrian, que será explicado en la parte de OLAP. Los resultados no fueron los esperados y se decidió volver a usar Postgresql sin extensión espacial aunque en un futuro se revisará la posibilidad de usarlo. Su buena documentación es el segundo motivo de su elección. También se prueba con el uso de LucidDB, un SGBD orientado a columnas en lugar de a filas. Su especial manera de indexar los datos provoca que el tiempo de agregar datos sea menor, y esto le hace ser candidato a implementar el SDW de Energy4People. Pero su mayor virtud también se convierte en su mayor desventaja, la orientación a columnas hace que el tiempo de inserción de datos sea muy costoso y caro en el uso de recursos. El hardware usado durante el proyecto, no lo soportó y está opción fue descartada y dejada como una posible mejora de futuro.
SOLAP	
Mondrian **Geomondrian**	Mondrian es un servidor OLAP que permite analizar grandes cantidades de datos. Aunque no se trata de un servidor SOLAP nativo, el modelo conceptual diseñado en el proyecto permite el uso de Mondrian como si de uno se tratara. Durante una fase del proyecto se trabajó con Geomondrian junto a Postgis, y los tiempos de respuesta en el tratamiento de geometrías quedaron lejos de los esperados. La función básica de Mondrian es la transformación de las consultas MDX a SQL para que puedan ser ejecutadas por Postgresql. Previamente se ha tenido que definir un esquema en XML que representa el modelo multidimensional. Su elección se ha basado en la buena documentación que provee, y en su integración con Java, a través de la librería Olap4j. Esta librería se puede describir a través de la analogía “el JDBC para OLAP”.
Servicio WEB	
Jersey **Spring WS**	Jersey es un framework para el desarrollo de servicios RESTful. Como se mencionaba en la fase de diseño, para abstraer a la lógica de negocio del modelo multidimensional se implementa una API REST. Esta librería, por su buena documentación e implementación del estándar JAX-RS API, se considera la mejor elección. La alternativa de Spring WS se tiene en cuenta durante el estudio de la tecnología para implementar esta capa. Es descartada porque tras revisar su documentación, su uso parece más complejo que el de Jersey.

Tecnologías	Descripción y razón de su selección o rechazo
Aplicación WEB	
Lógica de Negocio	
JAK	API de Java para el tratamiento de ficheros KML. Una vez los datos llegan desde la capa de datos a la capa lógica de negocio, estos son transformados a formato KML para su visualización en un mapa en la capa de presentación.
Java EE	Plataforma Java para la programación de arquitecturas de N-capas distribuidas. Para la ejecución de la aplicación web se utiliza un servidor de aplicaciones (**Apache Tomcat**) que convierte una petición HTTP a objetos Java para su tratamiento por la lógica de negocio.
Interfaz gráfica	
Google Maps **Google Earth** **OpenLayers** **Leaflet**	Google Maps dispone de un amplio conjunto de APIs que permiten trasladar su funcionalidad hasta un sitio web propio y superponer datos en él. Su buena documentación, así como su comunidad le hicieron candidato desde el primer momento a ser el cliente de mapas de la aplicación Energy4People. Aunque Google Earth fue elegido en primer lugar por su atractiva interfaz y la posibilidad de llamar la atención en el concurso de la NASA. Más adelante se estudió la posibilidad de utilizar librerías como OpenLayers o Leaflet, pero la falta de tiempo hizo que se dejará para trabajo futuro la profundización en dichas tecnologías.
Swing y AWT **JQuery** **Bootstrap de Twitter** **Backbone.js** **GWT**	Varias tecnologías han sido utilizadas en la interfaz de usuario. Como se verá en la sección de resultados, una aplicación de escritorio fue implementada en la primera fase del proyecto. La tecnologías utilizadas fueron las librerías Swing y AWT de Java. Una vez se empezó a realizar la aplicación web, se utilizó el 'stack' típico de tecnologías web, es decir, HTML, JavaScript y CSS. Para mejorar el desarrollo se utilizaron tecnologías que facilitaban y agilizaban el proceso de desarrollo. Estas fueron jQuery y el Bootstrap de Twitter. La primera proveía una manera sencilla e intuitiva de manejar el DOM (Document Object Model), la segunda "estandarizaba" la manera de implementar las hojas de estilo en la aplicación. Otras tecnologías como GWT y Backbone.js se estudiaron como alternativa para enriquecer la potencia del lado del cliente. El tiempo limitado del PFC, junto a que ninguno de ellos era crítico para mostrar la funcionalidad del sistema, hizo que fueran descartados pero considerados interesantes para futuras versiones del sistema.

Tecnologías	Descripción y razón de su selección o rechazo
Aplicación WEB	
Caché	
Ehcache	Para mejorar los tiempos de respuesta del servidor web a la hora de devolver los resultados, se introduce una memoria caché. Ehcache se configura a través de un fichero XML. El tamaño estimado para responder con rapidez en las pruebas de la aplicación y verificar que se produce una mejora es de 25 MB. Si la aplicación es llevada a un entorno de producción, se debe hacer un estudio más profundo de está tecnología y sus posibilidades.

Tabla 2.3: Principales tecnologías utilizadas en el proyecto.

2.5 Tests

Los tests acompañan durante todo el desarrollo del proyecto Energy4People. Para validar el desarrollo de las distintas partes del sistema (web scraper, modelos…) se sigue la metodología **TDD** (Test Driven Development). Esta metodología forma parte de la gestión del proyecto mediante una metodología ágil como Scrum. El uso de TDD conlleva la realización de los tests antes de implementar una determinada funcionalidad. La librería JUnit se utiliza en este proyecto para implementar tests en Java.

Pero en esta sección no se describen las pruebas unitarias sino las pruebas finales para probar la validez del sistema y el cumplimiento de los requisitos funcionales y no funcionales.

Para diseñar los tests se buscan escenarios que permitan que los resultados sean repetibles y observables. De esta manera se pueden usar como una guía sobre la que mejorar en futuros desarrollos y modificaciones del sistema.

Todos los tests se verifican junto al director del proyecto, Naveen Sidda, al que se le considera el experto que debe concluir si los resultados que observa se corresponden con lo que él espera.

En el Anexo D de la memoria se encuentran la documentación generada con los tests.

3 Conclusiones

A continuación se presentan los resultados y conclusiones extraídas del proyecto.

3.1 Resultados

Aunque se explica detalladamente en la sección *Gestión del proyecto*, el proyecto Energy4People se desarrolla en 4 fases de manera **incremental e iterativa**, produciéndose al final de cada una de estas fases unos entregables como ilustra la Figura 3.1. Se concluye que **los objetivos propuestos al principio del proyecto se cumplen**, y su consecución se resume en los siguientes puntos. Más detalles, Anexo E.

Primera iteración. Se implementa una **aplicación de escritorio** que tras leer un **SDW** genera KMLs de datos de radiación solar, que son abiertos posteriormente con alguna herramienta externa. Se implementa una primera versión del web scraper del sitio web la NASA. Por último, **se genera documentación para su uso como referencia técnica en el trabajo predoctoral del director del proyecto Naveen Sidda**.

Segunda iteración. Se implementa **Energy2People**. Se trata de una **aplicación web** que pide datos de energías renovables al servidor donde esta desplegado el SDW y el **SOLAP**. En el servidor, los datos se codifican en un fichero con formato KML que será mostrado en *Google Earth,* el cual está embebido en la interfaz web. Geokettle se implanta como herramienta para **crear y almacenar procesos ETL**.

Tercera iteración. Se mejora **Energy2People.** Se implementa un proceso de interpolación. Se mejora la generación de KMLs. También se implementa la visualización espacial multi-escalar, es decir, la generalización cartográfica ya es posible. También se mejora la interactividad utilizando *Google Maps* en lugar de *Google Earth*. La aplicación Energy4People ya es casi un hecho.

Cuarta iteración. Concluye el proyecto **Energy4People**. **Incremento en la documentación y posibles líneas futuras**, por el uso de nuevas tecnologías y soluciones no exploradas durante anteriores fases del proyecto. Destaca el despliegue de LucidDB como DW, Geomondrian como SOLAP nativo, Postgis como SDW con soporte geoespacial nativo, el uso de ficheros ESRI-Shapefiles, así como el estudio de cómo diseñar tablas agregadas que mejorarán el rendimiento de algunas consultas.

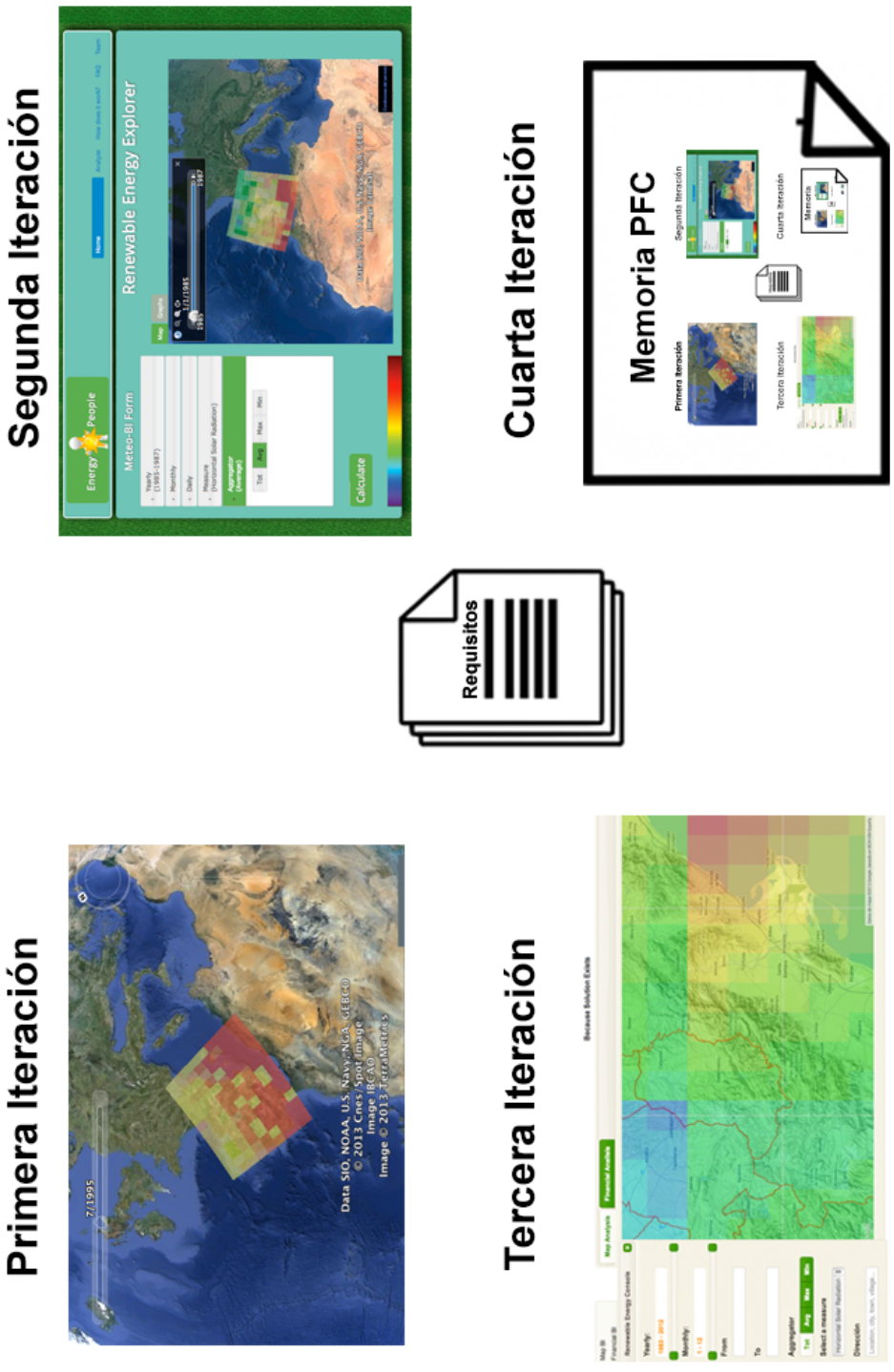

Figura 3.1: Resumen de los 4 entregables en cada una de las iteraciones del proyecto Energy4People.

3.2 Trabajo futuro

El proyecto Energy4People ha abierto **varias líneas de trabajo** sobre las que el grupo IAAA continuará. Además se anima al resto de profesionales de la ingeniería a coger ideas en ellas para posibles trabajos futuros.

Como se menciona en el documento, BI es un término paraguas que agrupa varias soluciones de ingeniería para el apoyo a la toma de decisiones. Algunas de estas soluciones recogen técnicas bien conocidas en el mundo de las ciencias de la computación. Su aplicación puede dotar de nuevas y mejoradas soluciones al dominio de las energías renovables. A continuación se presentan las líneas de trabajo con las ramas de conocimiento con las que están relacionadas.

1. **Minería de datos**

   El análisis de los datos mediante técnicas de **Inteligencia Artificial** y **Estadística** puede traducirse en el hallazgo de patrones desconocidos y dar soporte a la mejora de la toma de decisiones en el futuro de los recursos energéticos (Kalogirou, 2011).
2. **Almacenamiento de datos en estructuras no tradicionales**

   Los **sistemas de información** están avanzando por caminos que responden a las nuevas necesidades interpuestas por la enorme cantidad de datos que hay que almacenar. El uso de un SGBD como LucidDB o Apache Hive sobre Hadoop, se presenta como una alternativa de futuro próximo. (Krishman, 2013)
3. **Interacción del usuario no experto en la toma de decisiones**

   La toma de decisiones se considera un acto más emocional que racional (Vohs, 2007). El usar técnicas de **IPO** (Interacción Persona Ordenador) teniendo en cuenta esta afirmación es un gran desafío. (Christian & Russel, 2008)
4. **Web semántica**

   Siendo la **web** semántica el hilo conductor que guía el desarrollo futuro en el mundo web, tiene sentido que los datos que tengan que ver con recursos energéticos sean publicados utilizando formatos como RDF/XML que ayuden a su descubrimiento y posterior explotación. Además, surge la posibilidad de definir ontologías dentro de la toma de decisiones a través de la web. (Thuraisingham, 2005)

3.3 Conclusión personal

La realización de este proyecto me ha proporcionado una experiencia profesional y personal con la que espero poder guiar el resto de mis próximos trabajos.

La dirección de una persona de una cultura y formación diferente me ha ayudado a mejorar mis habilidades en la comunicación, lo cual considero muy importante en mi futuro como ingeniero.

La oportunidad de ganar un concurso organizado por la NASA, tratando un tema tan candente como el de las energías renovables, me ha permitido conocer a personas increíbles, llenas de conocimientos y sobre todo, de dudas y ganas de aprender cada día. Ellos han sido una fuente inagotable de inspiración.

La posibilidad de imaginar un producto, apasionarme por él y de ir definiéndolo iteración a iteración me ha hecho introducirme de una manera muy básica en la mente de un emprendedor. La pasión y la aplicación de metodologías ingenieriles han tenido más de una disputa en mi interior durante el desarrollo del proyecto, pero la sensación final siempre fue tremendamente positiva. Había aprendido un poco más.

4 Gestión del proyecto

En esta sección se presentan la metodología, la planificación y las herramientas utilizadas durante el proyecto.

4.1 Metodología

La asociación de este proyecto con una línea investigación en curso, lo une a un riesgo que hace que algunos de los parámetros definidos al principio del proyecto, pudieran tener poco o ningún sentido cuando este fuera avanzando. Por esto, y tratándose además de un proyecto de DW/BI, las metodologías ágiles son una buena elección para la gestión del proyecto (Collier, 2011).

Por otro lado, tal y como mencionaba en la sección de *Análisis,* he definido una hipótesis de mercado y un producto que la satisfaga para ayudarme en la definición de los requisitos del sistema. Cuando se quiere lanzar un producto nuevo al mercado, es conveniente utilizar un proceso iterativo en la que se pueda pivotar en torno a ese producto sin generar más coste del necesario. (Maurya, 2012)

Por lo descrito en los dos párrafos anteriores, **Scrum** y **Lean** son las metodologías usadas en este proyecto. Antes de describirlas, es importante decir que por las restricciones temporales y de recursos que impone el desarrollo de un PFC, ninguna de las dos metodologías se hace de una manera estricta y completa.

Scrum es una metodología ágil para la gestión de proyectos que define el progreso en un proyecto a través de series de iteraciones que reciben el nombre de sprints. Cada sprint es típicamente de entre 2 y 4 semana, siendo de 4 semanas los usados en este proyecto. Esta metodologías se considera correcta por lo descrito en el primer párrafo y porque **los requerimientos del sistema se han ido definiendo de una manera progresiva durante el desarrollo del proyecto**.

Lean define un flujo de trabajo para la construcción de software basado en la web aplicando y testeando el desarrollo del cliente. Esta técnica es importante en el proyecto Energy4People porque **la definición de un prototipo de producto al principio del proyecto y su aceptación va a ser la guía para desarrollar el resto del proyecto**.

4.2 Planificación

La metodología Scrum, provee de un componente para planificar el desarrollo de un proyecto que recibe el nombre de **Scrum Product Backlog**.

El *product backlog* es un documento de alto nivel para todo el proyecto. Contiene descripciones genéricas de todos los requerimientos, funcionalidades deseables, etc.

En la tabla 4.1 se presenta el product backlog desarrollado en el proyecto Energy4People. Como se puede observar consta de 4 sprints. Hay que destacar dos características. La primera es que en cada sprint la mayoría de las tareas a implementar son nuevas, es decir, que no están definidas en el primer sprint. Esto se debe al carácter "investigador" del proyecto Energy4People. La otra característica es que los esfuerzos que aparecen son los reales y no los estimados, que son los que habitualmente aparecen en el product backlog.

El total de horas invertidas en el proyecto Energy4People es de **1100 horas.**

La figura 4.1 muestra un balance de los esfuerzos realizados en cada sprint.

Antes de realizar cada uno de los sprints, se realiza una estimación de cuánto tiempo puede llevar cada una de las tareas que se añaden al product backlog. De esta manera al final de cada uno de los sprints se puede verificar si las estimaciones han sido buenas y a partir de esto guiar la planificación del siguiente sprint. Un resumen final de este trabajo de planificación se encuentra en el en el Anexo E.

ID	Descripción	1	2	3	4
	Esfuerzo realizado en cada sprint	**235**	**290**	**300**	**275**
1	Estudiar y entender las técnicas de BI y documentarlas	100	50	50	0
2	Conocer las tecnologías existentes y documentarlas	25	25	0	0
3	Implementar web scraper y extraer datos del sitio web de la NASA	25	25	0	0
4	Desplegar SDW	25	0	0	0
5	Implementar SOLAP	25	0	0	0
6	Implementar aplicación de escritorio que extraiga dato y genere KMLs	15	0	0	0
Estudio de BI/DW, y aplicación de prueba para validar conocimientos y tecnologías					
7	Estudio de la herramienta GeoKettle	-	20	0	0
8	Estudiar e implementar mejoras en el modelo relacional del SDW	-	50	0	0
9	Estudiar e implementar mejoras en el modelo multidimensional del SOLAP	-	50	0	0
10	Crear interfaz web para acceder a datos	-	30	0	0
Herramienta ETL, Mejoras en los modelos de datos y aplicación web Energy2People					
11	Interpolar los datos de la zona de Staging	-	-	70	0
12	Cargar los datos en el nuevo esquema del SDW	-	-	50	0
13	Cambiar modelo conceptual del modelo multidimensional	-	-	60	0
14	Cambiar interfaz web y utilizar Google Maps	-	-	10	0
15	Cambiar lógica cliente para acceso multi-escalar a los datos			20	0
Nuevas mejoras en cliente y servidor para permitir análisis multi-escalar en Energy4People					
16	Estudiar, probar y generar documentación de nuevas tecnologías	-	-	-	100
17	Organizar documentación generada antes	-	-	-	15
18	Escribir memoria	10	20	20	70
19	Escribir anexos	10	20	20	90
Corregir errores, estudiar nuevas tecnologías y documentación final					

Tabla 4.1: Tareas realizadas junto al tiempo dedicado a ellas en cada uno de los sprints.

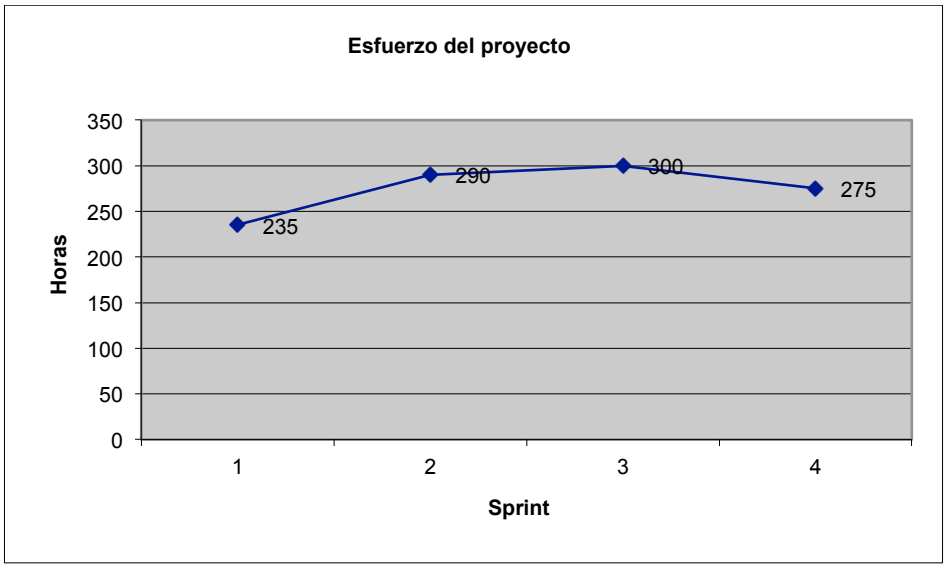

Figure 4.1: Número de horas realizadas en cada sprint del proyecto.

4.3 Herramientas

Para el desarrollo de las metodologías descritas en las secciones anteriores, así como para desarrollar el proceso, han sido utilizadas las siguientes herramientas.

Herramienta	Descripción
Gestión del proyecto	
Libre Office	Se ha utilizado para gestionar los documentos de gestión del proyecto como el product backlog.
Eclipse Mylyn	Se trata de un gestor de ciclo de vida de aplicaciones que viene con Eclipse IDE. Se ha utilizado para organizar cada una de las tareas en las que se dividía los requisitos del sistema.
Apache Maven	Se ha usado para gestionar el ciclo completo del desarrollo del proyecto. Tests unitarios, de integración y despliegues son algunas de las tareas que se han automatizado gracias a esta herramienta.
Control de Versiones	
Subversion	Cliente SVN para la gestión del control de versiones. Ha sido utilizado a través del plugin de Eclipse Subclipse. Para almacenar los datos se ha utilizad Assembla, un repositorio privado y gratuito en la web.
Git	Usado en la segunda iteración del proyecto para gestionar el proyecto atendiendo a la reglas del concurso de la NASA. Es un sistema de control de versiones con un enfoque distribuido.
Desarrollo	
Eclipse IDE	Entorno de desarrollo en el que se ha realizado el desarrollo de la aplicación. Gran soporte para el desarrollo en Java.
Google Chrome	Su consola JavaScript ha servido para depurar el código.
Desarrollo	
JUnit	Se utiliza para implementar la metodología TDD. Los tests unitarios son definidos y lanzados con JUnit.
Análisis del proyecto	
Lean Canvas	Herramienta de validación de hipótesis de negocio o productos. Su uso ha guiado en la definición del prototipo de producto.

Tabla 4.2: Herramientas utilizadas para gestionar el proyecto.

Bibliografía

Abbey, C. A. (2008). *Oracle8 Data Warehousing.* Oracle Series.

Adamson, C. (2006). *Mastering Data Warehousing Aggregates.* Wiley.

ArcGIS. (2003). *ArcGIS 9, Using ArcGIS Geostatical Analyst.*

Badard, T. (2011). *Spatialytics*. Obtenido de Spatialytics: http://www.spatialytics.org/

Ballard, C., Herreman, D., Schau, D., Bell, R., Kim, E., & Valencic, A. (1998). *Data Modeling Techniques for Data Warehousing.* IBM.

Barltzer, O. (2011). *Computational Methods for Spatial Olap.*

Berry, M., & Linoff, G. (2000). Mastering Data Mining: The Art and Science of Customer Relationship Management. *Wiley Computer Publishing, New York* .

Burstein, F., & Holsapple, C. (2008). *Handbook on Decision Support System.* Springer.

Caserta, J., & Kimball, R. (2004). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data.*

Chapman, A. D. (2005). *Principles and methids of data cleaning.*

Chapman, A. D. (2005). *Principles of data quality.*

Christian, P., & Russel, B. (2008). *Affect and Emotion in Human-Computer Interaction.* (Spring, Ed.)

Chung, H. M., & Gray, P. (1999). Special Section: Data Mining. *Journal of Management Information Systems* .

Collier, K. (2011). *Agile Analytics.*

Council, E. A. (2008). *Enterprise Architecture Business Intelligence (BI) Definition* .

ESRI. (2012). *ESRI.* Obtenido de ESRI: www.esri.com

Fayyad, U. (2001). The Digital Physics of Data Mining. *Communications of the ACM* .

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1998). *Design Patters.*

GlobalAtlasforSolarandWindEnergy-Implementationstrategy. (2013). *Irena.* Obtenido de Irena: irena.org

Gray, P., & Watson, H. (1998). Decision Support System in the Data Warehouse.

Harizopoulos , S., Abadi , D., & Boncz, P. (2009). *Column-Oriented Database Systems* .

Hohpe, G., & Woolf, B. (2007). *Enterprise Integration Patterns.* Addison-Wesley.

Hyde, J. (2009). *Mondrian 3.0.4 Technical Guide.*

Inmon, W. (2005). *Building the Data Warehouse.*

IRENA_Atlas_brochure. (2013). *Irena.* Obtenido de Irena: www.irena.org

Kalogirou, S. (2011). *Soft Computing in Geen and Renewable Energy System.* Springer.

Kamber, M., & Han, J. (2001). Concepts and techniques. *Morgan-Kaufmann Academic Press* .

Katja, H., Klan, D., & Sattler, K.-U. (2009). Online Tuning of Aggregation Tables for OLAP.

Kimball, R. (2002). *The Datawarehouse toolkit.*

Kniberg, H. (2007). *Scrum and XP from the Trenches.* InfoQ.

Kortink, M. A. (2010). *From ER Model to Dimensional Models.* Kortink&Associates.

Krishman, K. (2013). *Data Warehousing in the age of big data.*

MacEachren, A. M., & Kraak., M. (2001). *Research challenges in geovisualization. Cartography and Geographic Information Science.*

Malinowski, E., & Zimányi, E. (2009). *Advancer Data Warehouse Design.* Springer.

Malinowskis. (2008).

Maurya, A. (2012). *Running Lean.* (M. Treseler, Ed.) O'Reilly.

Microsoft Corporation. (2001). *XML for Analysis Specification.*

Mitchell, T. (2005). *Web Mapping.* O'Reilly.

Paulraj, M., & Syvaprakasam, P. (2012). *Functional behavior pattern for data mart based on attribute relativity.*

Pentaho. (s.f.). *Pentaho*. Obtenido de Pentaho: www.pentaho.com

Power, D. J. (2002). *Decisson Support Systems.* Greenwood Publishing Group.

Rodriguez, A. (2008). *IBM*. Obtenido de IBM: http://www.ibm.com/

Sauter, V. L. (2010). *Decision support systems for Business Intelligence.* Wiley.

Spencer, T., & Loukas, T. (1999). *Enerprise Systems Journal.*

Sprague, R., & Carlson, E. D. (1982). *Building Effective Decision Support Systems.*

Stephens, R. K., & Plew, R. R. (2000). *Database Design.*

Tapscott, D., & Williams, A. D. (2006). Wikinomics.

Thuraisingham, B. (2005). *Web Data Mining and Application in Business Intelligence and Counter-Terrorism.* CRC PRESS.

Vijayendra, N. (2005). *A couseware on ETL Process.*

Vohs, K. D. (2007). *Do Emotions Help ot Hurt Decision Making?*

Volk, M. (2011). Open Linked Data, Open Government Data Sets.

Zurita, A. (2010). *SEOSAR/PAZ Mission: Panel Design and Performance.*

Zurita, A. (2013). *Towards a SMOS Operational Mission: SMOSOps-Hexagonal.*

ANEXOS

A. Analysis

A.1 Product prototype requirements

In the *State of the Art* section, some problems like inconsistency between different datasets or impossibility of do multi-scale analysis are appointed. These problems need a solution, which are part of the system presented in Energy4People.

Used format to describe: **Market Problem/Requirement -> Product solution**

1. Unstructured and heterogeneous data -> A centralized renewable energy data repository.

2. No multi-scale analysis -> Provide multi-scale analysis on the map.

3. Impossibility of applying models mixing solar and wind information -> Allow users to apply models mixing solar and wind information.

4. Temporal queries cannot be applied -> Temporal queries can be applied.

5. High Latency displaying data on the map -> Minimize latency of displayed data on the map.

After defining the problems and the technical solutions. The solution must take Naveen Sidda's predoctoral research concepts. So, it is important to clarify them with a tool that allows extracting the most important points of the analysis. This tool is called Lean Canvas and it is described in the Figure A.1.

With this tool, it is tried to visualize a "product" or a "business model" that helps to turn Naveen Sidda's research into a set of functional and non-functional requirements that can be transformed into a system that can have a future industrial use.

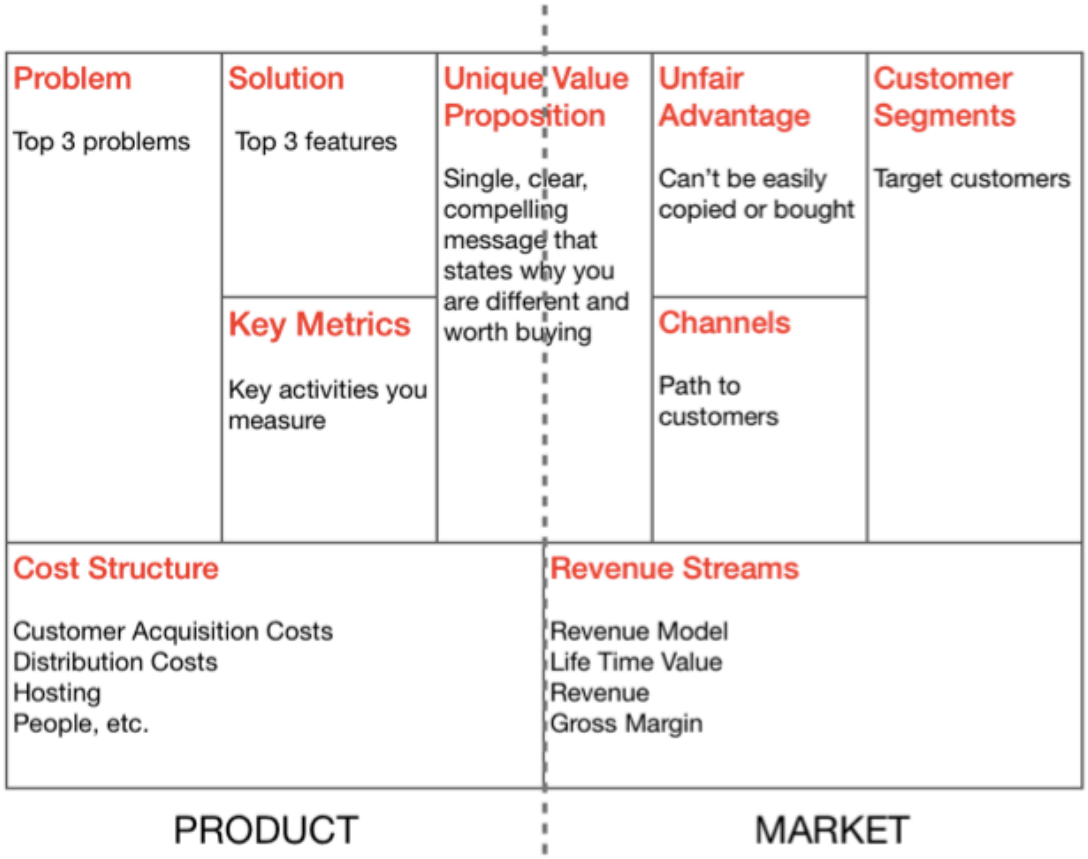

Figure A.1: Lean cavas template. (Maurya, 2012)

Next lines describe steps followed to define a good lean canvas.

Problem and customer segments

Top 3 Problems

1. Huge and heterogeneous amount of renewable energy data.
2. Inconsistency between different datasets.
3. Not enough engagement with renewable energy consultancy tools.

Existing Alternatives

- Solar Maps
- Green calculators
- Atlas

User Roles

- Citizens interested in renewable energy data.

Early Adopters

- Little Renewable energy consultancies that want to use Internet to engage new potential users.

Unique Value Proposition

- Expert analysis tools for citizens. Citizens have the next word.

Solution

- **Problem**: Huge and heterogeneous amount of renewable energy data.
- **Solution**: ETL process and data warehousing.
- **Problem**: Inconsistency between different datasets.
- **Solution**: Data warehousing.
- **Problem**: Not enough engagement with renewable energy consultancy tools.
- **Solution**: OLAP cubes and multi-scale map visualization.

Channels

- SEO

One of the problems in this project is the right definition of requirements in order to complete the product prototype. For this reason, different iterations with different deliverables are done. Besides the final Lean Canvas that is described in the main document, two other Lean Canvases have been defined and used to guide development during a determined phase. They are rejected because of some new considerations that appear.

The figure A.2 and the figure A.3 show these Lean Canvases.

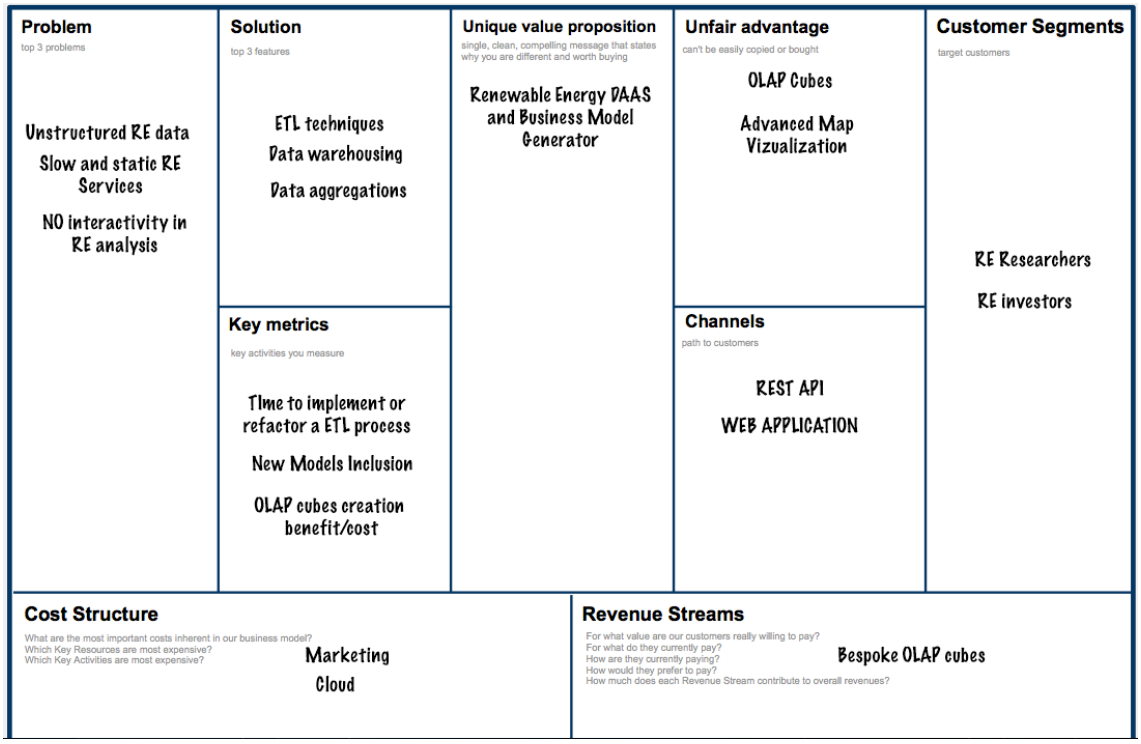

Figure A.2: First rejected Lean Canvas.

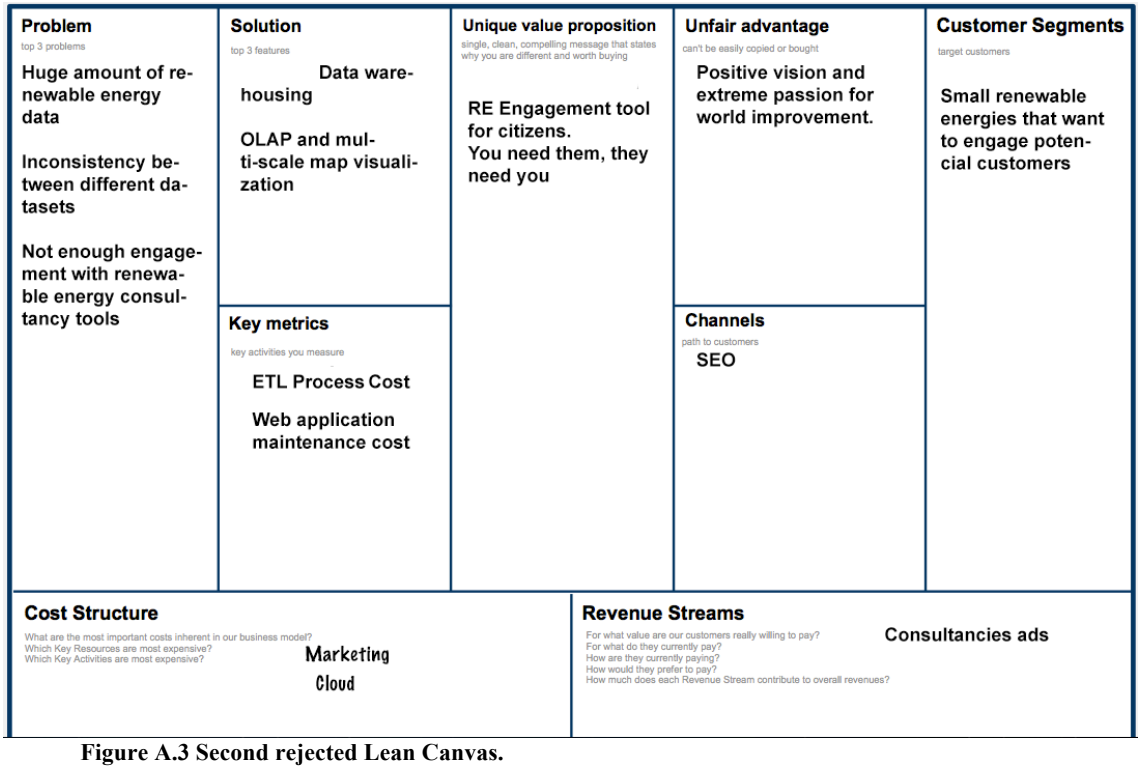

Figure A.3 Second rejected Lean Canvas.

A.2 User stories

Before the use case scenarios are defined, in order to guide tasks to be implemented, some user stories are defined. Then, they are transformed the use case scenarios, and in functional and non-functional requirements that can be found in the *Analysis* section of the main document. The table A.1 defines these user stories.

ID	Description
1	As a provider, I need to study BI techniques and documenting
2	As a provider, I need to know the available BI technologies
3	As a user, I want NASA as data source
4	As a user, I want a central repository
5	As a user, I want to use OLAP cubes to access data
6	As a user, I want an application than access data and generates KMLs
To study BI/DW and develop and application to validate knowledge and technologies.	
7	As provider, I need a tool that helps me to implement ETL processes
8	As a user, I want some improvements in central repository
9	As a user, I want some improvements in temporal dimension in OLAP
10	As a user, I want a web application to access data
ETL tool, new improvements in data model, and web application. Energy2People	
11	As a user, I want to interpolate data in order to have better resolution
12	As a user, I want to do some improvements in central repository
13	As a user, I want some improvements in spatial dimension in OLAP
14	As a user I want Google Maps to display data
15	As a user, I want to change the way to access data
New improvements in backend and frontend. Near Energy4People.	
16	As a provider, I need to test new technologies to improve my business
17	As a student, I need to organize documents generated
18	As a student, I need to write a report
19	As a student, I need to write an annex
To study new technologies and generate final documentation. Energy4People	

Table A.1: User stories

B. Solution Design

Next pages show with more details the solution design described in the main document. The ETL process, the spatial data warehouse and the web application are designed in to order to achieve the goals of this project.

B.1 ETL

Data Flow Diagram

Although the context diagram is presented in design section of the main document, ETL process must be expanded to detail deeper all the process.

The Figure B.1 shows again the context diagram of ETL process. Three different processes can be distinguished: extract, transform and load. The *Stage* storage is where data is transformed before loading it in the DW. In this case, an **interpolation** function is applied in order to enrich data resolution. Then, data is load in the Spatial Data Warehouse, the central repository of Energy4People project.

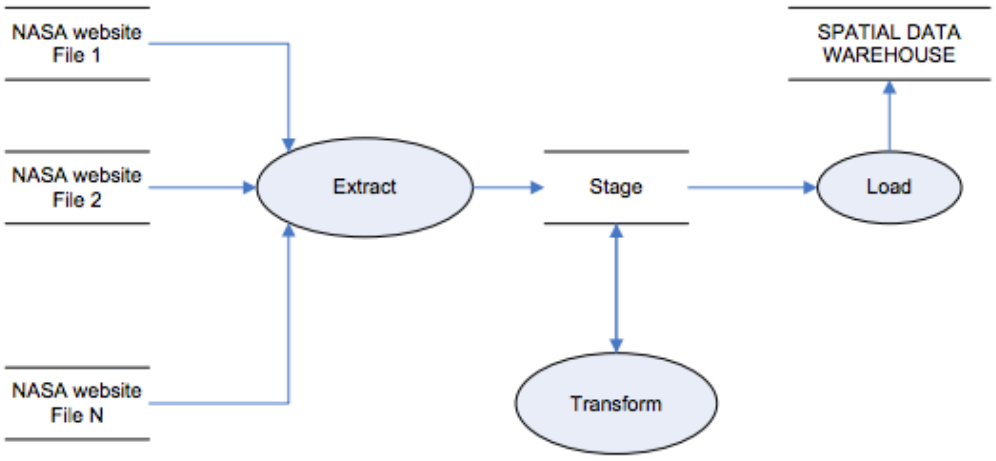

Figure B.1: Context driagram using DFD notation.

The Figure B.2 illustrates some processes that must be run in order to obtain the final snowflake schema in the spatial data warehouse. A *TemporalDimension* table must be created in order to support temporal queries in the web application. Once *TemporalDimension* table is created, it is necessary to join fact table with dimension tables, so a process is considered to be necessary to achieve this goal. Finally, at this

point of the load process, some data is stored in various tables; so considering that the spatial data warehouse size is going to be a critical point, all redundancies between tables must be removed and a final process is created to do this task.

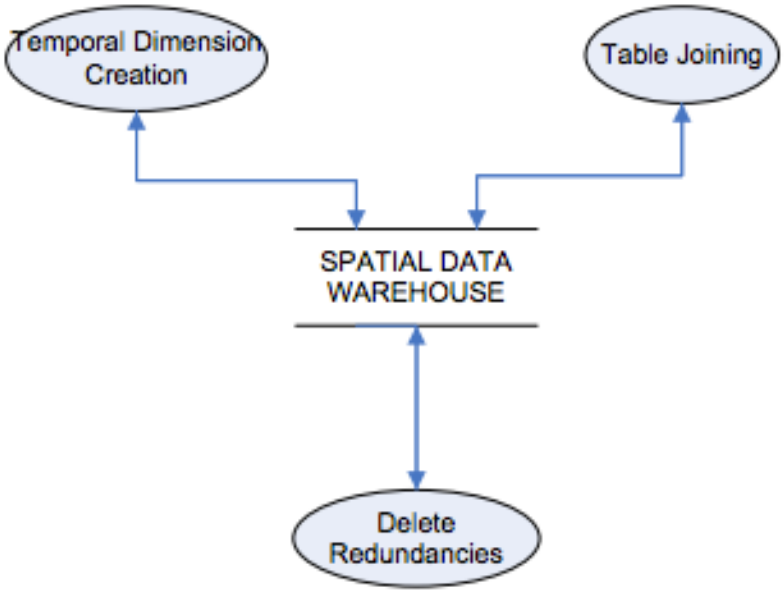

Figure B.2: Load process in detail.

So, two different processes take special importance in this ETL process. The first is the extraction process. Being NASA website the source of data, it is necessary apply some techniques that are difficult to find in a traditional ETL tool. So it is necessary to design and implement a web scraper that behaves like a person who is downloading the data using the browser. The second important process is the interpolation process. This is needed in order to give data some more resolution. The NASA original data has low resolution, so in order to achieve the multi-scalar analysis, it is necessary to interpolate data.

The next pages explain *Web Scraper* and *Interpolation process* in detail.

B.1.1 Web scraper

The Web scraper is a deferential part of this project. Although many datasets are analysed to extract data from them and being used as sources, the NASA website is thought to be the best data source despite its low data coverage. This can be resolved with interpolation.

The Figure B.3 shows the web scraper structure. It is modelled with a class diagram using UML 2.0 notation. One of the main goals in all tasks and programs developed in ETL process is to code just one time, and then, the changes in programs are done via configuration files. *ScraperConfigurator* class has this responsibility.

The use of software design patterns is one of the points taken into account when designing this program. After reviewing some design patterns**,** the **command pattern** is thought to be the most appropriate. The command pattern is a behavioural design pattern in which an object is used to represent and encapsulate all the information needed to call a method at a later time.

The commands in this application are: *NasaScraper*, *TextFileParser* and *StageLoader*. They are invoked for *Façade* class that executes firstly *NasaScraper*, then *TextFileParser* and finally *StageLoader*.

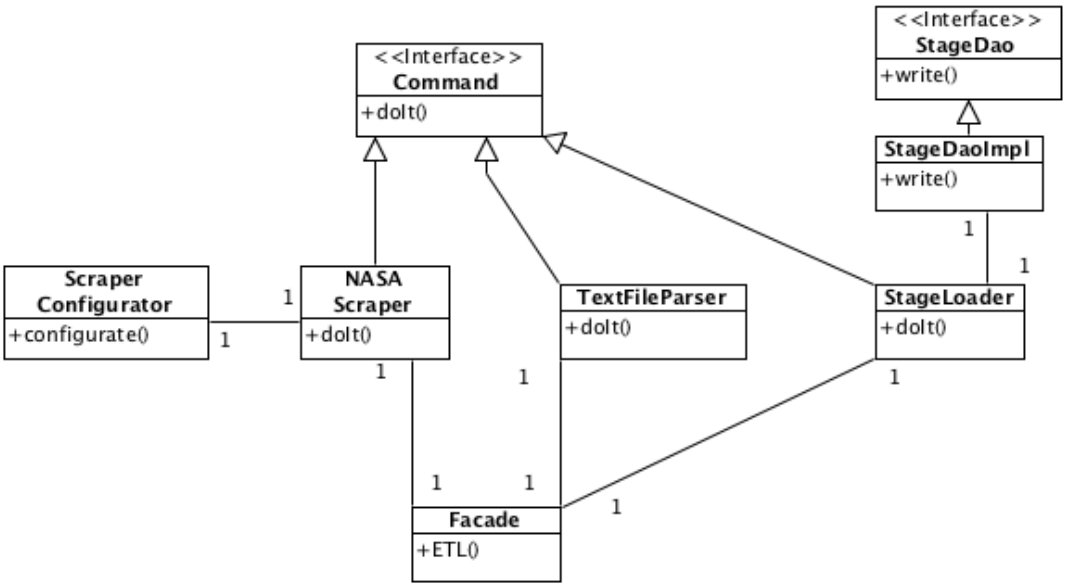

Figure B.3 Scraper class diagram using UML 2.0 notation.

B.1.2 Interpolation process

Interpolation is a necessary part in this project. The problem with NASA data resolution and the requirement of using multi-scalar analysis are taken into account when this program is designed. Summarizing, this class is designed to solve the low resolution problem. Although, some different interpolation algorithms exist, its implementation should be transparent to other classes. So, an interface is used to achieve this goal.

The Figure B.4 shows the program structure. It is modelled with a class diagram using UML 2.0 notation. One of the main goals in all tasks and programs developed in ETL process is to code just one time and then, different changes in programs are done via configuration files. *InterpolationConfigurator* class has this responsibility.

The use of software design patterns is one of the points taken into account when designing this program. After reviewing some design patterns**, facade pattern** is thought to be the most appropriate.

Facade pattern is a structural design pattern that provides a simplified interface to a larger body of code, such as a class library.

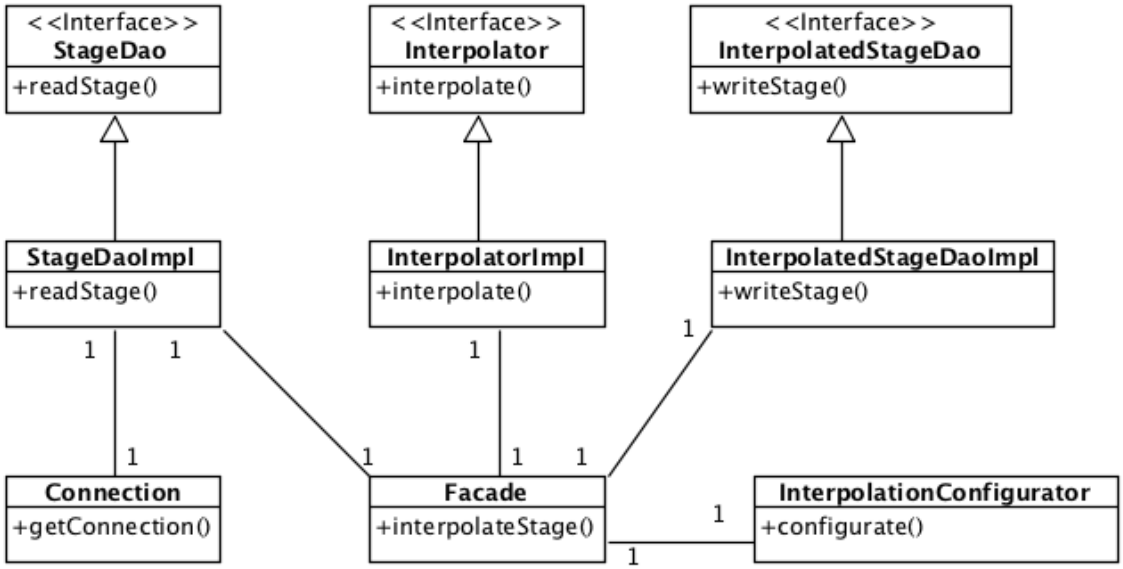

Figure B.4: Interpolation process class diagram using UML 2.0 notation.

In conclusion this design will provide the configurability achieved in the ETL processes. What it means that a person without programming knowledge would be able to change Interpolation behaviour without "touching" any code lines.

B.1.3 Future work in ETL

Next lines describe an ETL process that do not appear in final system functionality, but that is designed and implemented during the last phase of the project. But some changes must be done in order to improve the performance. So, in this phase, it is tried to use a spatial data warehouse with native spatial native support. For example, with this new approach, some ESRI-Shapefiles can be stored in the SDW.

The Figure B.5 illustrates the extraction, transformation and load of data from ESRI Shapefiles sources to Spatial Data Warehouse. The ESRI-Shapefile, or simply a shapefile, is a popular geospatial vector data format for geographic information system software. Shapefiles spatially describe vector features: points, lines, and polygons, representing, for example, water wells, rivers, and lakes. Each item usually has attributes that describe it, such as name or temperature.

The Shapefiles can contain data that is not relevant in final application, so before loading data in the SDW, these fields are removed in the extract/transform process.

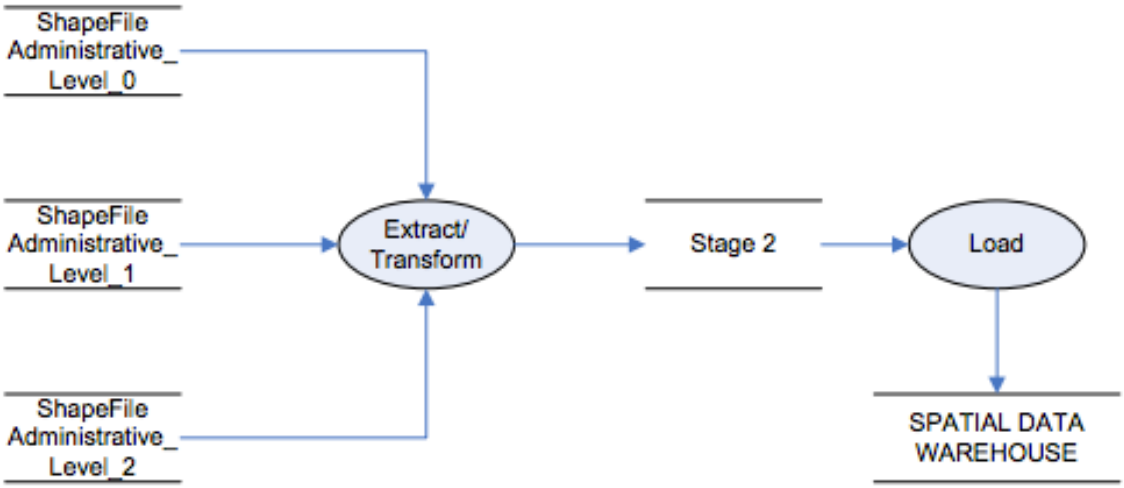

Figure B.5: ETL using ESRI-Shapefiles. This is a future work line.

Once spatial data is load in SDW, new operations can be executed. These operations, for example, can be used to separate data from different administrative units into data marts where data can be analysed faster because of its smaller size.

The Figure B.6 shows one of the most important steps in this new approach. The change between data models (spatial data warehouse model and data mart model) to achieve a

more powerful multi-scale analysis. Taking advantage of administrative names stored in spatial data warehouse, the data marts can contain only data from a particular province, region or city. This last approach maximizes possibilities to speed up analysis depending on the zone displayed on the map.

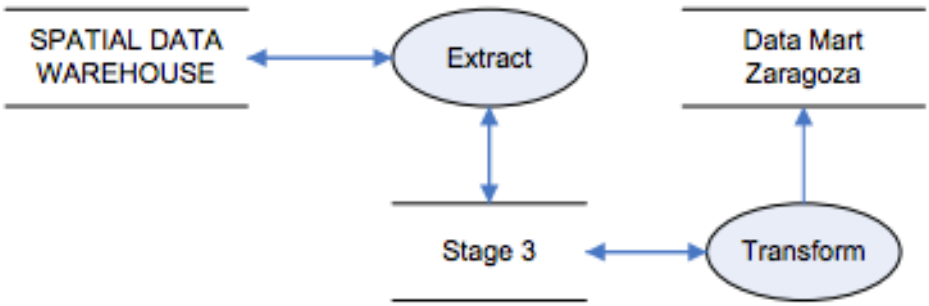

Figure B.6: Data marts building from the spatial data warehouse. This is a a future work line.

B.2 Spatial data warehouse

The design of a data warehouse is quite different from a relational database design. While in the relational databases the ER diagram is used, in data warehouse projects, dimensional modelling is the most appropriate technique. The two main features of this modelling type is the possibility to use OLAP to query the data warehouse and the goal of a de-normalized schema instead of the normalized, which is preferred in ER modelling.

B.2.1 Conceptual Model

The first step in DW modelling is the conceptual design. Data is represented according to the data cube model. This model has four main concepts:

- Fact: It is a concept that is relevant for the decisional process. For example, in this project the facts are the solar radiation and the wind speed.
- Measure: A numerical property of the fact. For example, in this project the measures are the quantity of solar radiation and the quantity of wind speed.
- Dimension: A property of a fact described which respect to a finite domain. In the context of this project, a spatial dimension composed by a Latitude Dimension and a Longitude Dimension, and a temporal dimension, are necessary to characterize the facts.
- Hierarchy: The elements of a dimension can be organized as a hierarchy, a set of parent-child relationships, typically where a parent member summarizes its children. For example, the year member is the parent of the month member, and this one is the parent of the day member.

The Figure B.7 shows the conceptual modelling of the spatial data warehouse used in Energy4People. Although only two dimensions are marked, the reality is that there are three dimensions. *LatitudeDimension* and *LongitudeDimension* can be seen as two different dimensions in the *SpatialDimension*.

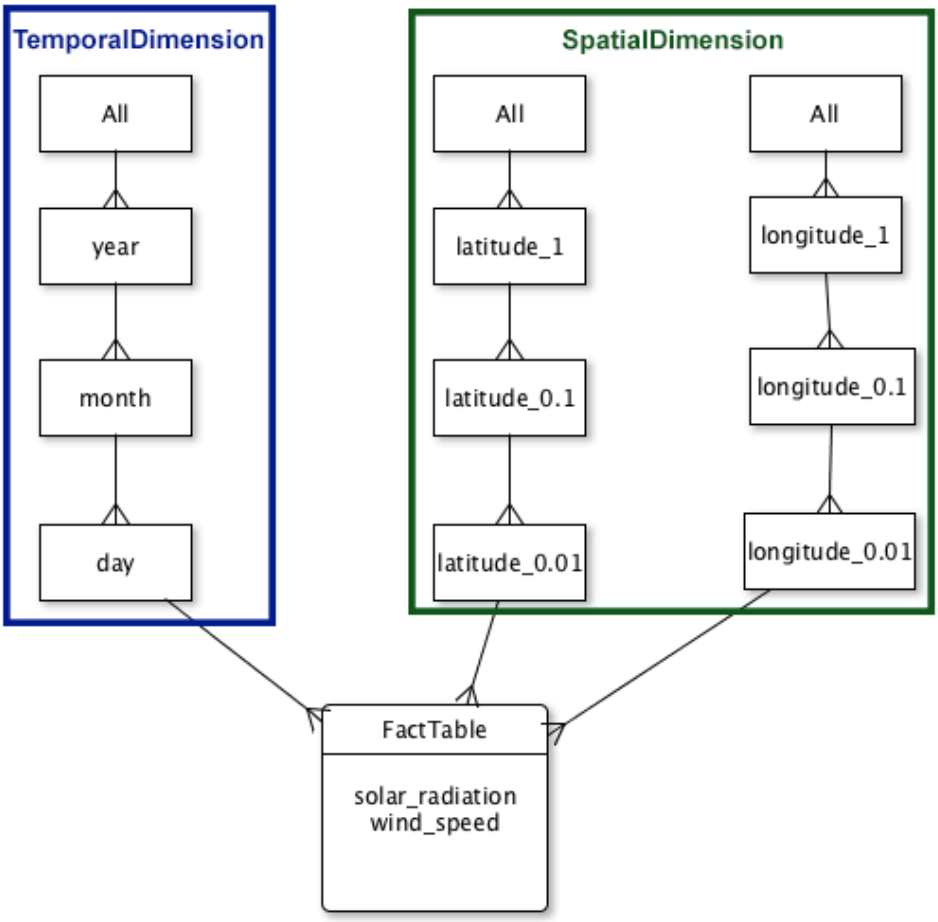

Figure B.7: SDW conceptual model.

B.2.2 Logic Model

Starting from the conceptual design, it is necessary to determine the logical schema of data. At this point, different models to represent multidimensional data can be used. These models are based on relational technologies (ROLAP), relational technologies with spatial support (SOLAP), multidimensional technologies (MOLAP) or a combination of both (HOLAP).

In this approach, **ROLAP and SOLAP model has been chosen**. ROLAP and SOLAP use the relational data model, which means that data is stored in relations. The reason of using the two approaches is because **SOLAP in final system** is used just like a logical component, what it means that a native SOLAP is not used. Summarizing, it means that a ROLAP approach is used but it behaves like a SOLAP.

Three schemas can be used, the star schema, the snowflake schema and the constellation schema. They all are more explained in the Annex G.

The Star schema is used in this project. In computing, the star schema (also called star-join schema) is the simplest style of data mart schema. The star schema consists of one or more fact tables referencing any number of dimension tables. The star schema is an important special case of the snowflake schema, and is more effective for handling simpler queries.

The star schema gets its name from the logical model's resemblance to a star with a fact table at its centre and the dimension tables surrounding it representing the star's points.

The Figure 9 shows the star schema used in the SDW. This model can be seen as a translation of the conceptual model into an ER model. And this is consequence of the use of a SOLAP/ROLAP approach. Both of them use a relational database in order to store data. One of the most important features of this model, is that *SpatialDimension* in the logical model, has latitude and longitude members while in conceptual model they can be seen as two different dimensions with their own hierarchies. Summarizing that is the difference between de cube or dimensional model and the logic model.

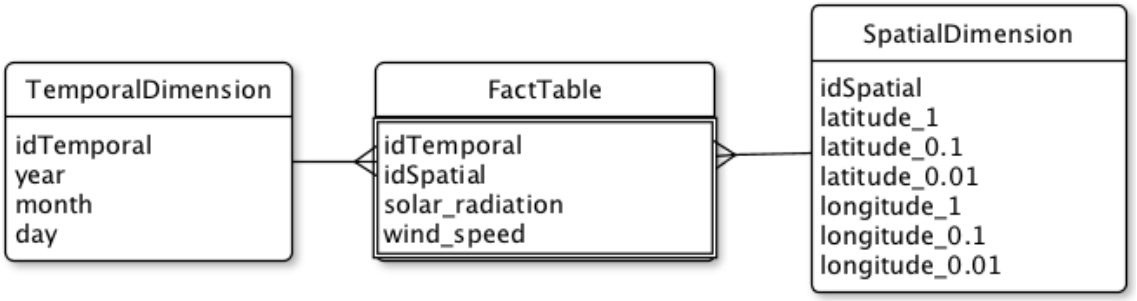

Figure B.8 SDW logic model.

B.2.3 Future Work in SDW Design

As it is mentioned, during a phase of the project it is tried to do some research in order to open new future work lines. One of the things that are tried to do is to combine complex geometries of administrative units with the pair latitude/longitude extracted from NASA website.

The Figure B.9 shows a high level abstraction of design goal. Data warehouse stores renewable energy quantity with their geometries and administrative names, for example, Spain -> Aragón -> Zaragoza. However, the data marts contain only latitude/longitude pairs and renewable energy quantity.

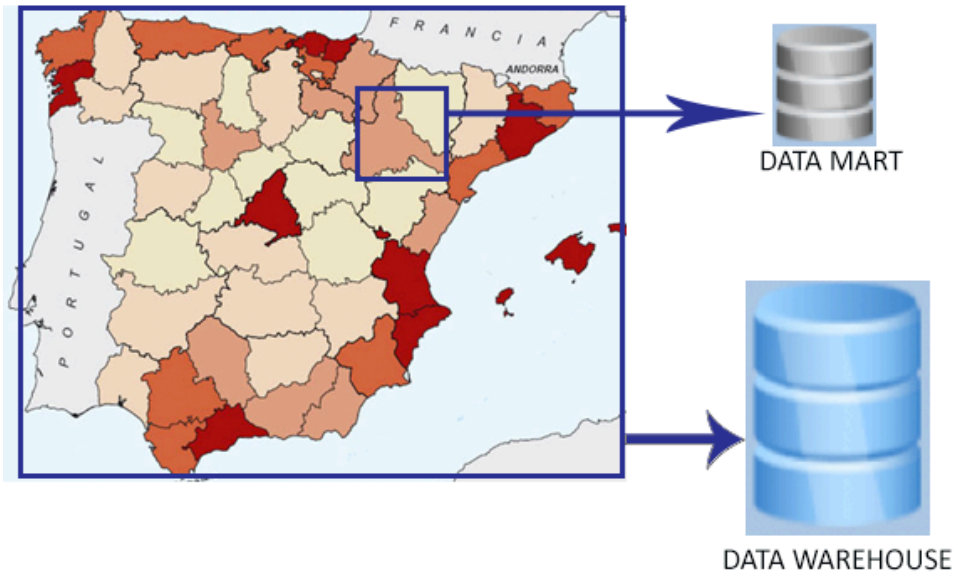

Figure B.9 Abstraction of the future goal.

The figure B.10 shows the conceptual model of this approach. It can be seen that *SpatialDimension* (the right one) has a hierarchy composed by four geometries. The smallest geometry is the point. In this case, the point is not a pair latitude/longitude but a geographical point. Then we have province, region and country (adm_unit_level_3, adm_unit_level_2 and adm_unit_level_1).

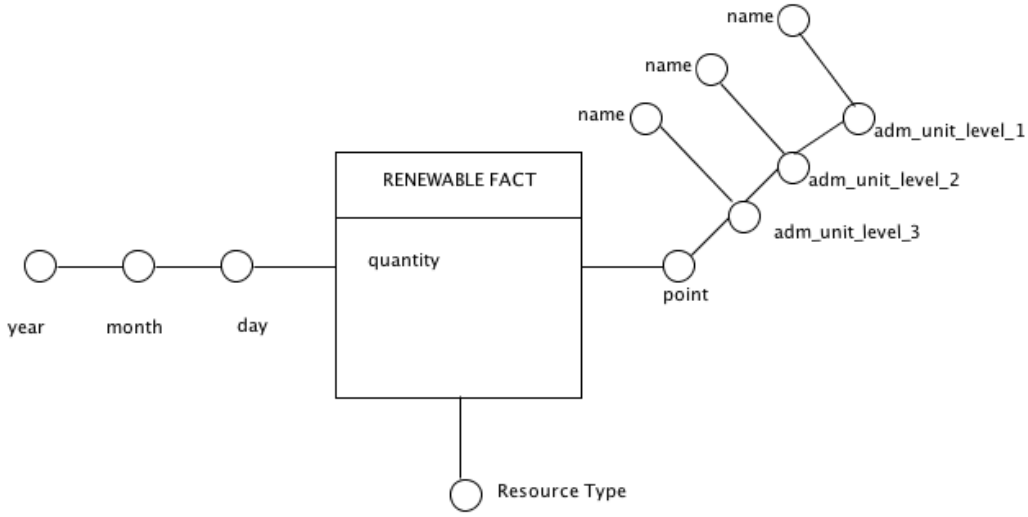

Figure B.10: Future SDW logic model.

The new spatial data warehouse need this conceptual modelling because in some decisions levels, knowing the name and the geometry of the administrative unit is really important, but when going deeper in the analysis, aggregations of huge amounts of points are the only meaningful data.

The Figure B.11 shows the logical model that is designed to be implemented in this future line. This model does not present a star schema but a snowflake schema. With this model the question, "How much solar radiation has been in **Zaragoza** for the last 20 years " can be answered. What it allows a lot of powerful new features in the system.

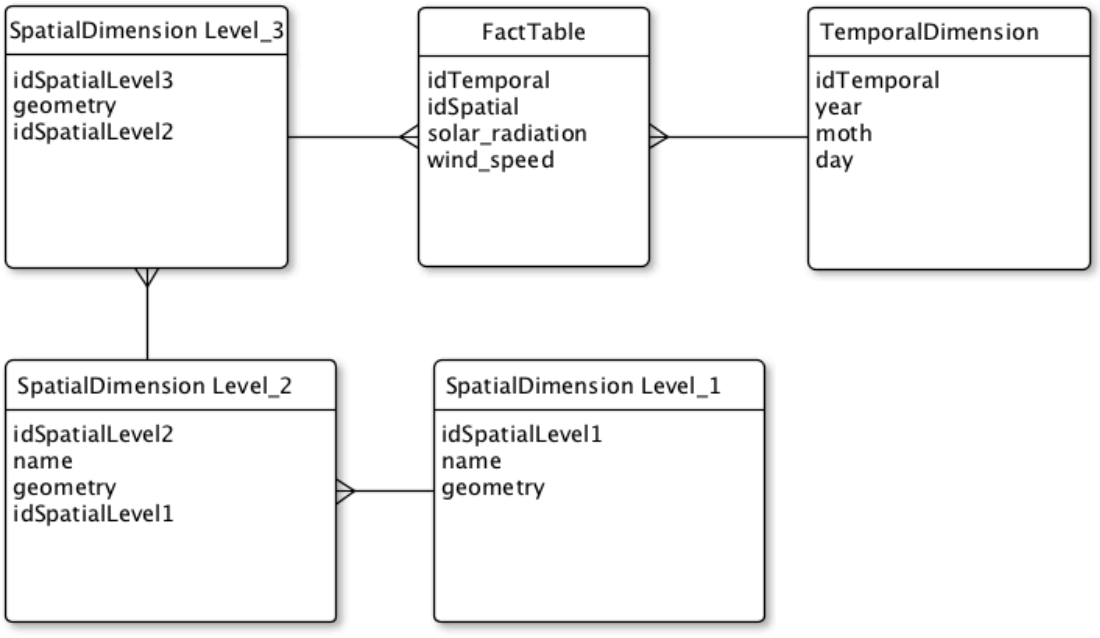

Figure B.11: Future SDW logic model.

B.2.4 Aggregate tables design

When the system requirements demand high performance and low latency it is not enough to do a good SDW design. New strategies must be taken, and one of them is the use of aggregate tables. The aggregate table is essentially a pre-computed summary of the data in the fact table. (Hyde, 2009)

Although this Energy4People version does not contain any aggregate tables, during the last phase of the project it is studied the impact of using this strategy in the system. If the number of rows in *FactTable* is about 2,737,500,000 (30 years X (365 o 366 days) X A matrix of 500*500 points), this can result in low performance and high latency. To solve this problem it is necessary to design some aggregate tables.

The main problem of using this approach is that aggregate tables are actually materialized views what it means they are using physical space in the server. Moreover, it does not exist a exact science that points how to design the aggregate tables, so the most usual solution is to research which queries are going to be executed by a the user. (Hyde, 2009)

The figure B.12 shows an example of two aggregate tables that in a future version will be implemented.

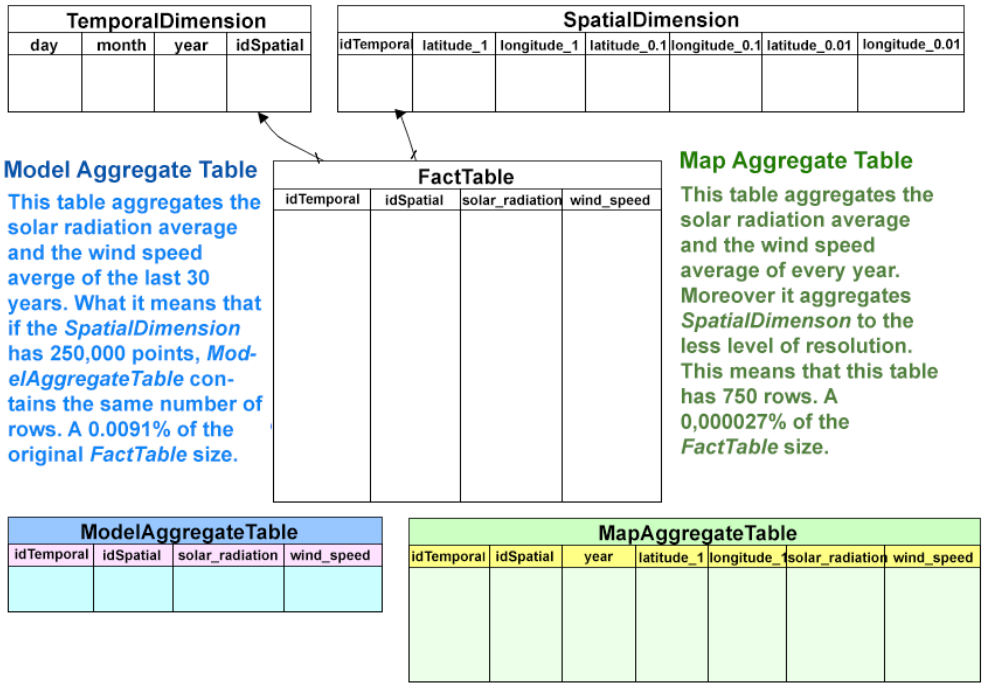

Figure B.12: Aggregate tables design.

B.3 Web Application

B.3.1 Service layer

Once SDW and OLAP cubes are designed, it is necessary to access data via web. This can be made in two ways:

1. The Business logic retrieves data from a XMLA server.
2. The Business logic requests data from a REST API that abstracts data layer with the multidimensional model.

The second option is considered better because of some reasons. But first of all it is important to describe what a REST API service is.

REST defines a set of architectural principles by which it is possible to design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP and WSDL based interface design because it's a considerably simpler style to use. (Rodriguez, 2008).

So the reason of using this approach is that queries that are available in the OLAP cube are pre-defined. So, giving some intuitive URLs, all queries can be mapped. In Energy4People system**, service layer can be seen as mapping between requests via URLs and MDX queries**. Every request triggers a MDX query that will retrieve data from data warehouse or data marts.

The goal of this REST service in the future is to provide a simple API, which any developers could use to develop new models using renewable energy data and cubes.

B.3.2 Business logic

As it is mentioned in the *design* section of the main document, a class diagram is used to design web application. It is also said that two main design patterns are used, the Façade pattern and the DAO pattern. This brings a powerful capacity of scale new functionality in the future because these patterns are used to abstract details of implementation to other system classes.

The Figure B.13 shows a more extensive class diagram than the presented in the main document. It is also a high level diagram because there are some levels of private methods that are considered to add noise to the schema. The reality is that as there is only one developer in this project the communication of designs have been always developed in high level without detailing some implementation details that are usually changed.

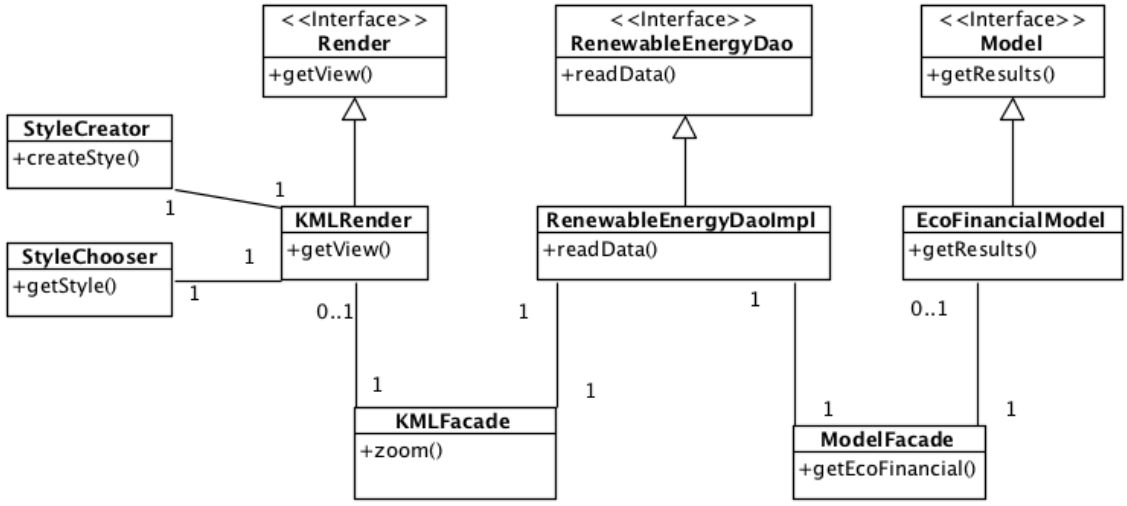

Figure B.13: A more detailed business logic class diagram using UML 2.0 notation.

Two classes that do not appear in main document are the *StyleCreator* class and *StyleChooser* class. *KMLRender* class uses these classes in order to create some specific styles in the KML file, and the *StyleChooser* selects one of these styles for each one of the data.

Finally, it must be said that in the last phase of the project some other functionalities are studied in order to improve system. One of these functionalities is to add GeoJSON support. Thanks to this design, the addition of this new class does not modify the behaviour and the structure of the rest of the components of the system.

C. Implementation

Next section describes the implementation of the solution. The implementation is divided in the implementation of ETL, SOLAP and web application. Moreover, the new future work lines thar were mentioned are presented with some implementation.

C.1 ETL

The ETL implementation is implemented using Java programming language and Geokettle tool. Other frameworks such as Spring Batch are analysed to address the implementation goals but they were rejected because they satisfy less requirements than the chosen tools.

Geokettle is a spatial ETL tool dedicated to the integration of different spatial data sources for building and updating geospatial data warehouses. In order to achieve this integration, Geokettle uses steps, transformation and jobs. Steps are simple processes, which achieve little transformations in data. Transformations are sequential steps that transform data from a source. A job is a group of transformations. This tool provides a framework where steps can be for example SQL scripts, JavaScript programs, and geospatial transformations.

Extraction, transformation and load are described in next points:

C.1.1 Extraction

One of the requirements of this project is to extract data from NASA web site. The problem is that this data was only accessible via browser. One click event is needed by every pair latitude/longitude. Doing this manually has no sense and it is necessary to automatize this task via web scraping techniques.

Moreover, Geokettle does not include anything that could be used to do web scraping. So this web scraper is implemented using Java. This web scraper is a batch program that extracts data from NASA website in text file format and loads it in a table of a **Postgresql** database.

WEB SCRAPER

Although Geokettle provides many pre-configured processes in order to implement ETL processes, specifically it doesn´t implement anything that can be considered a web scraper. For this reason one must be implemented.

Web scraping (web harvesting or web data extraction) is a computer software technique of extracting information from websites. Usually, such software programs simulate human exploration of the World Wide Web by either implementing low-level Hypertext Transfer Protocol (HTTP), or embedding a fully-fledged web browser, such as Google Chrome or Mozilla Firefox.

One of the main goals when designing and implementing scraper is flexibility and **configurability**. Different behaviours have been implemented to achieve scrapping. One thread, multiple threads, polite behaviour waiting a specific time before requests, etc. So the achieved goal is that user software administrator does not need to code any lines. Only "touching" configuration file, as the shown in Figure C.1, the extraction of data can be customized.

name	value
driver_name	org.postgresql.Driver
driver_url	jdbc:postgresql://localhost:5432/StageDB
user	madrid
password	******
minYear	1983
maxYear	2013
latMax	44
latMin	35
longMax	4
longMin	-10
polite	yes
threads	2
data	swv_dwn & WS10M

Figure C.1: Web Scraper configuration file.

Although many Java libraries aim to interact with websites i.e. **Apache Http Client,** a high level abstraction was needed to interact with NASA website like a browser. To do this task **JWebUnit** library is chosen. This library can navigate along the html document and "click" on a determinate link.

The figure C.2 describes the interaction between the scraper and the NASA site. It is important to say that this interaction is done depending on the number of points that are required.

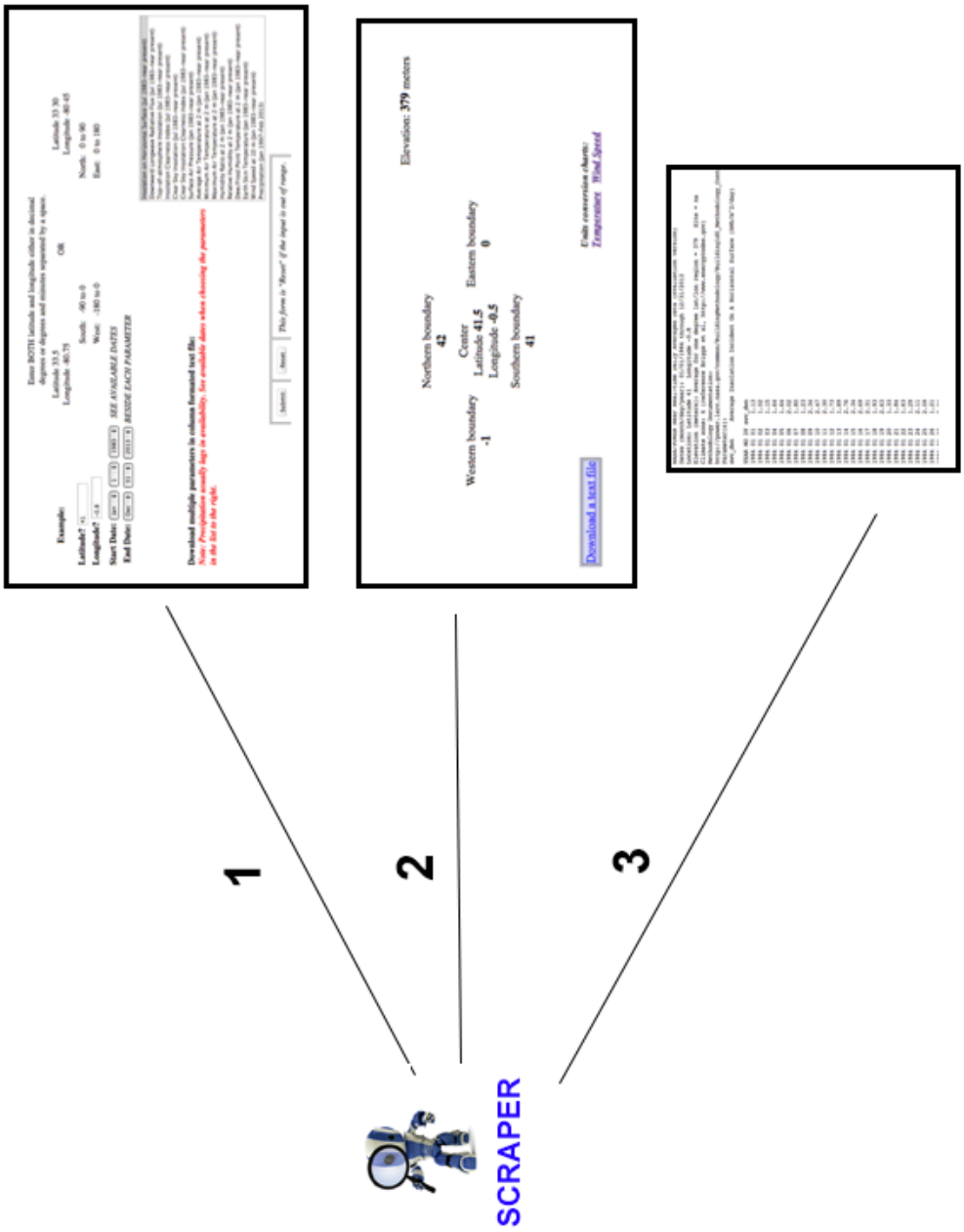

Figure C.2: How Scraper interacts with the NASA website.

1 First step when scraping NASA website, is to do an HTTP request that simulates the interaction with the form of the page that is shown in the Figure C.3. It is supposed that in this form, you can fill latitude, longitude, the data you want to retrieve, for example, solar radiation or wind speed, and the period of time of the data. Then Submit button must be clicked.

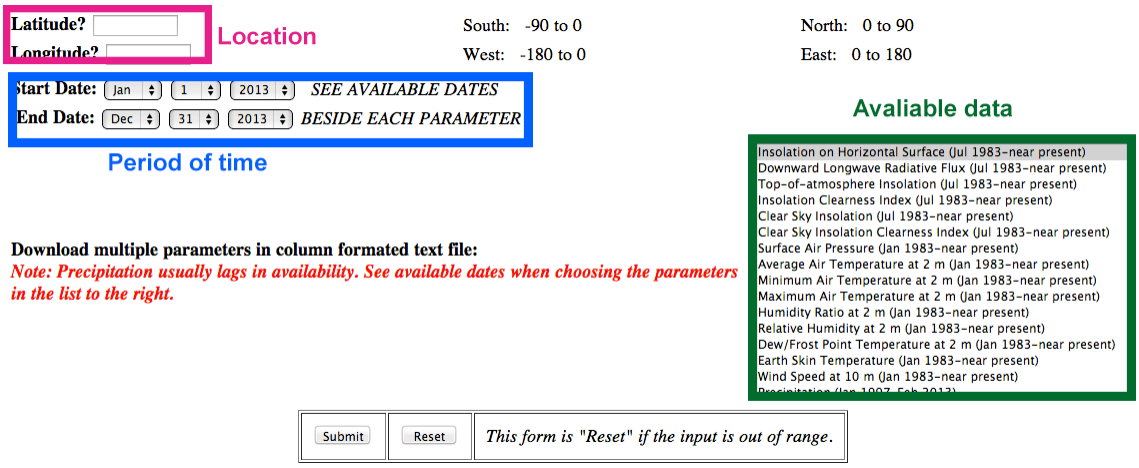

Figure C.3: First NASA website page.

2 The Figure C.4 shows the response of previous interaction. It is a page where a link to a text file appears. It is supposed that a human would click this link in order to retrieve data. So web scraper must find this link in the HTML file and click on it.

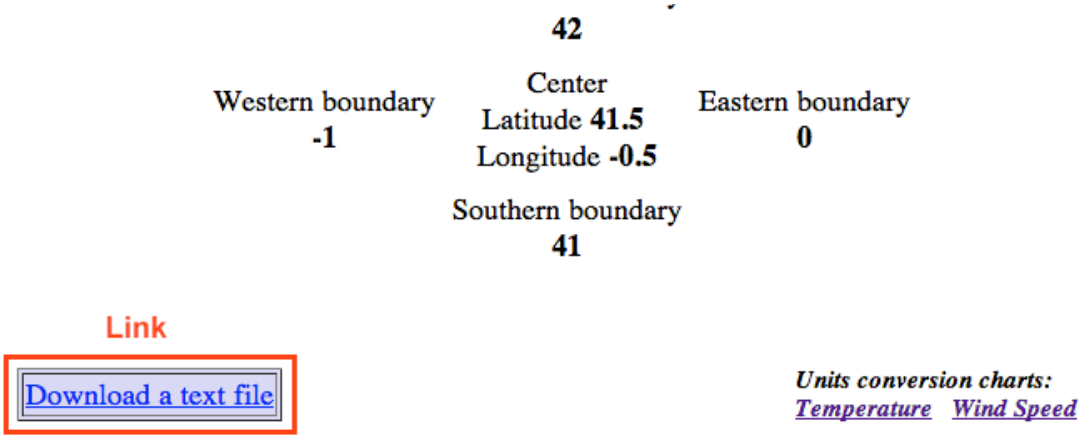

Figure C.4: Second NASA website page.

3 Finally, the temporary a text file, as the shown in the figure C.5, is returned by NASA website. Now, scraper must download it and let other processes, to store this data in a staging place where some specific transformations can be applied.

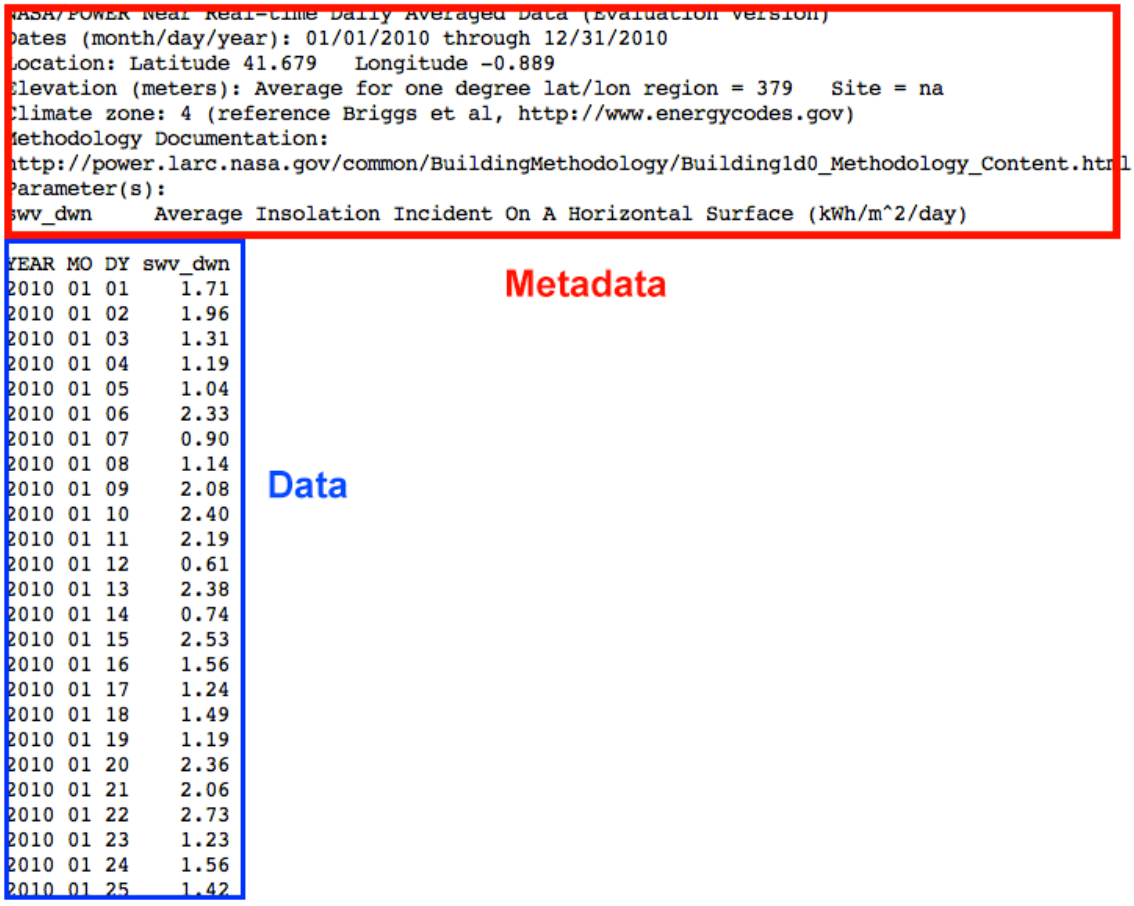

Figure C.5: The text file with the renewable energy data.

Once data and metadata are retrieved, the program implemented stores the data in the table where the transformations are going to be executed. This process uses JDBC to connect with DBMS, in this case Postgresql.

JDBC is a Java-based data access technology (Java Standard Edition platform) from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database.

C.1.2 Transformation

INTERPOLATION

The table where the web scraper loads the data is known in ETL processes as the **staging area**. Here, any kind of transformations can be done in order to clean and normalize data.

As it is mentioned in previous sections the global coverage of NASA website data is on a 1° latitude by 1° longitude grid. This data resolution is considered too low in order to build an analysis tool. So, an interpolation process must be executed.

Java Advanced Imaging (**JAI**) is a Java platform extension API that provides a set of object-oriented interfaces that support a simple, high-level programming model which allows developers to create their own image manipulation routines.

One of the main goals when designing and implementing interpolation process is flexibility and **configurability**. Different behaviours have been implemented to achieve interpolation, for example, one thread, multiple threads, different database schemas, the same schema, several years, one year, etc. So the achieved goal is that user software provider only programs this system once. Then, only "touching" configuration file, the interpolation of data can be customized.

In order to simplify execution only year 1990 is interpolated. The rest of the years would be the same but the system resources are not enough to interpolate 30 years.

The Figure C.6 illustrates the configuration file that has been developed. Many parameters can be configured but the most important are:

1. DBMS parameters: name of the driver, DBMS URL, user and password.
2. Bounding box: Maximum latitude and longitude next to their minimums.
3. Mathematical parameters: Number of decimals, type of interpolation…

name	value
driver_name	org.postgresql.Driver
driver_url	jdbc:postgresql://localhost:5432/DataMart
user	madrid
password	madridPass
minYear	1983
maxYear	2013
latMax	44
latMin	39
longMax	3
longMin	-2
granularity	0.05
dbFrom	SpatialDWH
dbTo	DataMart
measure	solarRadiation
poolSize	20
geohashLength	8
matrixSide	20
digitsNumber	2
interpolationLimit	0.99
createdDimensions	false
transformationThreads	1
insertingThreads	1
log	true

Figure C.6: Interporlation process configuration file.

So, as it is known NASA resolution is 1º, and after some study of the problem a resolution of 0.01º is considered to be correct in order to test the possibilities of multi-scale analysis. Maybe in future new interpolations of different scales must be implemented. So, in order to achieve this goal one interpolation is implemented and then the resolution of data can be changed using the configuration file.

The Figure C.7 illustrates two interpolations of data. The real interpolation results in a dataset of 10000 times more data and a resolution of 0.01 X 0.01 (latitude X longitude).

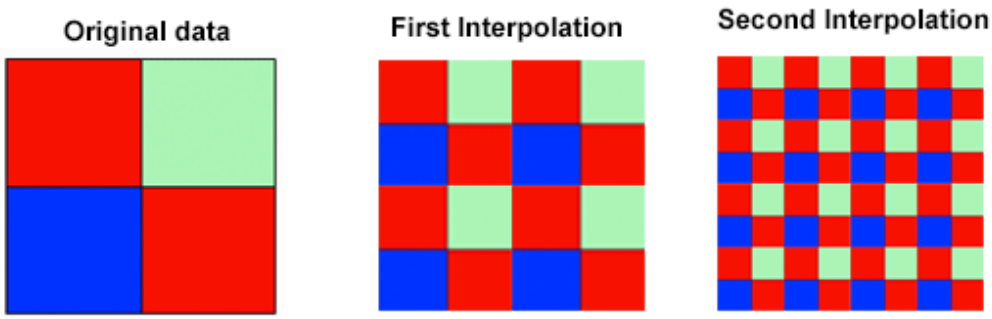

Figure C.7: Example of interpolations that can be done with the implemented program.

The Figure C.8 illustrates the transformation developed in Energy4People in a simplified schema. Before the transformation it only exists the *StagingTable.* The transformation phase executes interpolation process in order to obtain the *InterpolatedStagingTable* that is the previous step before the load phase. The *InterpolatedStagingTable* has 10,000 times more rows than the StagingTable.

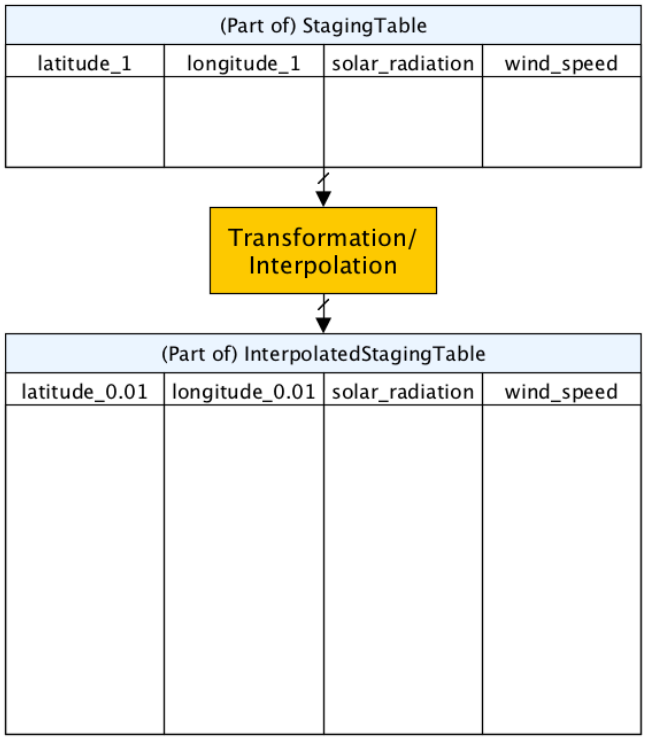

Figure C.8: The summary of the Interpolation process.

In future versions of Energy4People this interpolation process must implement a more accurate interpolation algorithm such as Kriging. This algorithm is part of the geostatistical analysis.

Geostatistical Analyst uses sample points taken at different locations in a landscape and creates (interpolates) a continuous surface. Geostatistical Analyst derives a surface using the values from the measured locations to predict values for each location in the landscape. For example, Kriging assumes that at least some of the spatial variation observed in natural phenomena can be modelled by random processes with spatial autocorrelation, and require that the spatial autocorrelation be explicitly modelled. (ArcGIS, 2003)

C.1.3 Load Implementation

The load phase loads the data into the end target, usually the data warehouse (DW). Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information; frequently, updating extracted data is done on a daily, weekly, or monthly basis. Other data warehouses (or data marts) may add new data in a historical form at regular intervals. More complex systems can maintain a history and audit trail of all changes to the data loaded in the data warehouse.

In the project Energy4People the load process is a set of SQL scripts in order to do the load that appears in the Figure C.9. This figure illustrates that after extraction and transformation phases it only exists a table called *InterpolatedStagingTable.* After the load phase the star schema is created with the three tables presented in the previous sections of documents.

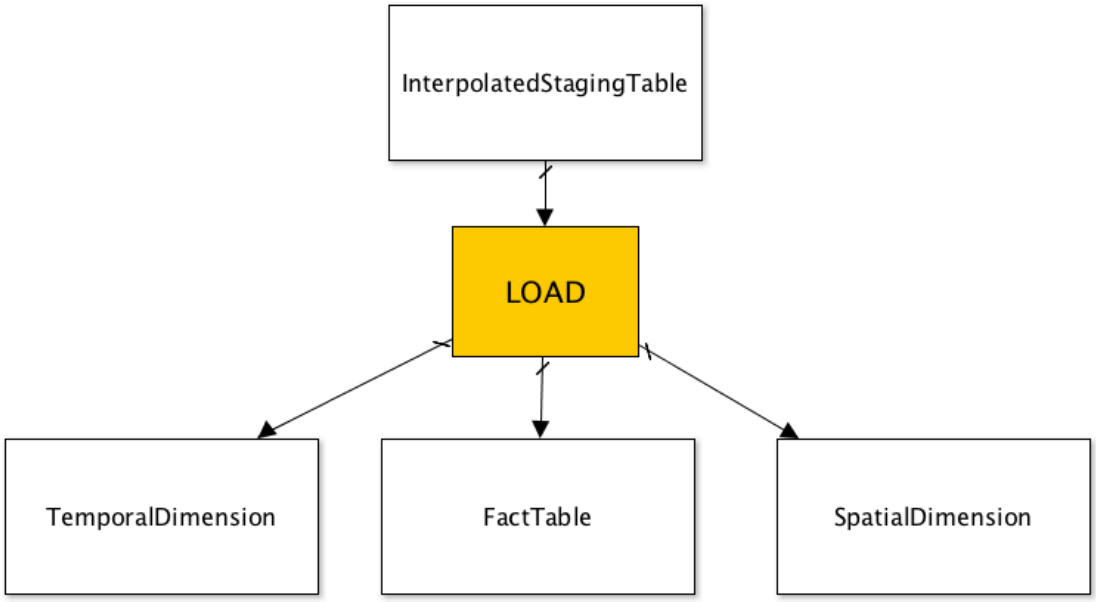

Figure C.9: Graphic summarizing Load process.

The next pages show some work that is implemented in the last phase of the project. Another technologies such as PostGIS and LucidDB are tested in order to find some improvements to the actual or if a high studied is necessary, document the given steps in order to guide future member groups.

C.1.4 Future work in ETL implementation

In the last phase of the project some spatial processes were included in the ETL process. Although the final implementation of the project does not include this spatial functionality some processes are executed with successful results. This documentation can help Naveen Sidda and IAAA group members in order to do some advances in any other research lines in the future.

Extraction

Once the data is extracted, the Staging tables are necessary in order to load spatial data from ESRI-Shapefiles. To achieve this, Geokettle provide a set of steps that solve the necessities to achieve the goal.

Figure 13 shows how this extraction is achieved using Geokettle tool.

- Shape files
    - Fields Filter: Files contain fields, which are not interesting in order to make decisions. Moreover they add unnecessary complexity to data warehouse.
    - SRS transformation: Each geospatial data has their own coordinate system and projections. In order to make accurate decisions, data extracted from NASA and geospatial data from shape files must have the same SRID (Spatial Reference System Identifier).
    - Database Load: Last step is to transform the shape file in a database table. In this case geometry support in necessary in database and Postgresql/Postgis is the SGBD chosen.

Figure C.10 ETL Flow implementation in Geokettle

Transformation

Spatial Stage Table

Some transformations could be carried out, but they are beyond the scope of the project. For example, the name of the different administrative units could be translated in different languages in order to provide a multi language analysis.

Load

After transforming data, load data in data warehouse, according to the models presented in design sections, is the last step in the ETL.

The situation at this point is the next:

- Four tables with geospatial data of different administrative levels.
- A table with renewable energy measures, and the day and location where these measures took place.

The load process has different steps. All of them are SQL scripts whose goal is to load all data with a snowflake schema in data warehouse and a star schema in the case of data marts.

Spatial Dimension Load

1. To create table ‘Spatial Dimension Level 4’ with distinct pairs latitude/longitude and “Coordinate” column extracted from ‘NASA Stage Table’.
2. To create two new columns in ‘Spatial Dimension Level 4’. First with a primary key. Second one, is going to point one of the tables obtained from shape files.
3. To do a spatial join between ‘Spatial Dimension Level 4’ and ‘Spatial Dimension Level 3’

Fact Table Load

1. Create table “Fact Table”.

2. Insert renewable energy measures, and foreign keys pointing to ‘Spatial Dimension Level 4’ and ‘’Temporal Dimension”.

At this point data warehouse is created with a **snowflake schema.**

Once the SDW is deployed, it is recommended to divide it in data marts. A **data mart** is the access layer of the data warehouse environment that is used to get data out to the users. The data mart is a subset of the data warehouse that is usually oriented to a specific business line or team. Data marts are small slices of the data warehouse. Whereas data warehouses have an enterprise-wide depth, the information in data marts pertains to a single department.

LucidDB is the DMBS that is used to implement data marts. The reason is simple, LucidDB is a column-oriented DBMS, and aggregations are computed faster than in relational DBMS. LucidDB is an open source RDBMS purpose–built entirely for data warehousing and business intelligence. It is based on some “novel” architectural cornerstones such as column-store, bitmap indexing and hash/join aggregation.

In data marts, the providing information is inside the last administrative level (local level), so geometries are not necessary and latitude/longitude pairs are enough to do an accurate analysis.

Dividing DW into some data marts, a faster access can be observed. The reason is the less number of rows in the *FactTable*. This can be a really important point to study in order to develop some politics in the future. For example, Zaragoza and Barcelona would have their own data mart with their own data.

C.2 SOLAP

C.2.1 OLAP

After the ETL process, a SDW is deployed and the next step is to build OLAP cubes. **Mondrian is considered the best option** because of its good documentation and the growing community. The problem with Mondrian is that is not a real spatial-enabled OLAP. But how this project has demonstrated, with a correct conceptual modelling, a cube OLAP implemented with Mondrian can be used as a logic SOLAP. During a phase of the project **Geomondrian** is used to implement a more powerful SOLAP. The response latency is not considered good enough and its implementation and deeper study is a future work.

In order to implement logical model, Mondrian uses **schemas**.

A schema defines a multi-dimensional database. It contains a conceptual model, consisting of cubes, hierarchies, and members, and a mapping of this model onto a physical model.

The conceptual model consists of the constructs used to write queries in MDX language: cubes, dimensions, hierarchies, levels, and members.

The physical model is the source of the data that is presented through the logical model. It is typically a star schema, which is a set of tables in a relational database.

The most important components of a schema are:

- A *cube* is a collection of dimensions and measures in a particular subject area.
- A *measure* is a quantity that you are interested in measuring.
- A *dimension* is an attribute, or set of attributes, by which you can divide measures into sub-categories.
- A *member* is a point within a dimension determined by a particular set of attribute values. For example, 1997, 2012, 5 (May) and 31 are members of the *Time Hierarchy*.
- A *hierarchy* is a set of members organized into a structure for convenient analysis.

- A *level* is a collection of members that have the same distance from the root of the hierarchy.
- A *dimension* is a collection of hierarchies that discriminate on the same fact table attribute (say, the day that a sale occurred).

The Figure C.11 shows the final schema in a graphical way. It can be seen that the main parts that are described in conceptual design phase are present in this figure.

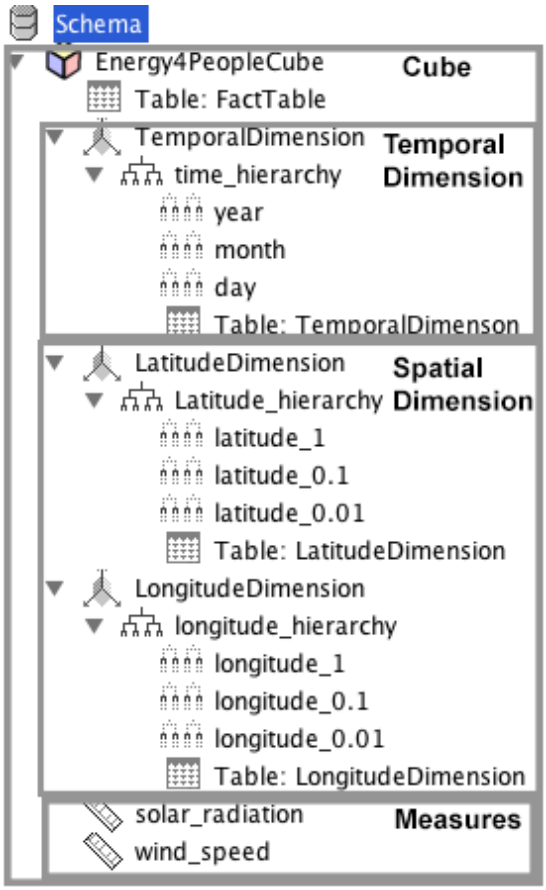

Figure C.11: A graphic visión of the implemented cube.

But what Mondrian really uses in order to retrieve data and communicate with the physical data layer is a XML file. In this file, the schema with its measures, dimensions, hierarchies and their levels are defined. The figure C.12 shows the final XML file that is executed in Energy4People project. It can be seen that the same parts of the figure C.11 are marked. It seems clear that XML file represents the multidimensional model that differences an OLAP cube between other alternative data structures.

```
<Schema name="Energy4PeopleSchema">
  <Cube name="Energy4PeopleCube" visible="true" cache="true" enabled="true">
    <Table name="FactTable" schema="energy">
    </Table>
    <Dimension name="TemporalDimension" type="TimeDimension" visible="true" foreignKey="idTemporal" highCardinality="false">
      <Hierarchy name="Time Hierarchy" visible="true" hasAll="true" primaryKey="idtime">
        <Table name="time_dimension" schema="granularity_3">
        </Table>
        <Level name="year" visible="true" column="year" type="Numeric" uniqueMembers="true" levelType="TimeYears" hideMemberIf="Never">
        </Level>
        <Level name="month" visible="true" column="month" type="Numeric" uniqueMembers="false" levelType="TimeMonths" hideMemberIf="Never">
        </Level>
        <Level name="day" visible="true" column="day" type="Numeric" uniqueMembers="false" levelType="TimeDays" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Dimension name="LatitudeDimension" type="StandardDimension" visible="true" foreignKey="idlocation" highCardinality="false">
      <Hierarchy name="Latitude hierarchy" visible="true" hasAll="true" allMemberName="Latitudes" primaryKey="idlocation">
        <Table name="location_dimension" schema="energy">
        </Table>
        <Level name="level_1" visible="true" column="latitude_1" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="level_2" visible="true" column="latitude_0.1" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="level_3" visible="true" column="latitude_0.01" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Dimension name="LongitudeDimension" type="StandardDimension" visible="true" foreignKey="idSpatial" highCardinality="false">
      <Hierarchy name="Longitude hierarchy" visible="true" hasAll="true" allMemberName="Longitudes" primaryKey="idlocation">
        <Table name="location_dimension" schema="energy">
        </Table>
        <Level name="level_1" visible="true" column="longitude_1" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="level_2" visible="true" column="longitude_0.1" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
        <Level name="level_3" visible="true" column="longitude_1" type="Numeric" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
        </Level>
      </Hierarchy>
    </Dimension>
    <Measure name="solar_radiation" column="solar_radiation" datatype="Numeric" aggregator="avg" visible="true">
    <Measure name="wind_speed" column="wind_speed" datatype="Numeric" aggregator="avg" visible="true">
    </Measure>
  </Cube>
</Schema>
```

Figure C.12: XML file of the implemented cube.

C.2.2 Future Work in SOLAP

GeoMondrian is the first implementation of a native SOLAP server. It provides a consistent integration of spatial objects into the OLAP data cube structure, instead of fetching them from a separate spatial database, web service or GIS file. GeoMondrian provides integration of spatial and non-spatial data types at a lower level. It directly interfaces with the Mondrian OLAP server and extends the capabilities of the OLAP server to handle spatial data. The OLAP server, in turn, uses a spatially enable database system, for example, PostGIS, as storage for data and to execute data queries. The OLAP server's responsibility is to translate Spatial OLAP queries issued by the user into sequences of SQL queries that are being evaluated by the underlying database engine. The extensions that GeoMondrian provides only enable support for spatial data types but do not change the strategies used by Mondrian to evaluate OLAP queries. "This prevents these systems from efficiently answering queries on complex spatial dimension hierarchies, such as non-strict dimension hierarchies." (Barltzer, 2011)

The figure C.13 shows how the *SpatialDimension* is defined when using this kind of native SOLAP. In this cases, as it is described in *design* section, a snowflake schema is modelled, and in the schema this is represented with joins between tables. It is also important to point that in the levels, it is defined a property of the type *Geometry*. This allows Geomondrian to retrieve geometry columns from the SDW.

```
<Dimension name="SpatialDimension" foreignKey="idspatial">
    <Hierarchy name= "Spatial Hierarchy" hasAll="true" primaryKey="id_2"
        primaryKeyTable="adm_unit_level_2">
        <Join leftKey="id_1" rightKey="id_1" rightAlias="adm_unit_level_1">
            <Table name="adm_unit_level_2" />
            <Join leftKey="id_0" rightKey="id_0">
                <Table name="adm_unit_level_1" />
                <Table name="adm_unit_level_0" />
            </Join>
        </Join>
        <Level name="Country" table="adm_unit_level_0" column="name_iso"
            uniqueMembers="true">
            <Property name="geom" column="the_geom" type="Geometry" />
        </Level>
        <Level name="Region" table="adm_unit_level_1" column="name_1"
            uniqueMembers="true">
            <Property name="geom" column="the_geom" type="Geometry" />
        </Level>
        <Level name="City" table="adm_unit_level_2" column="name_2"
            uniqueMembers="false">
            <Property name="geom" column="the_geom" type="Geometry" />
        </Level>
    </Hierarchy>
</Dimension>
```

Figure C.13: XML file with the future spatial dimensión. It contains geometries.

Aggregate tables implementation

Unlike many OLAP servers, Mondrian does not store data on disk: it just works on the data in the RDBMS, and once it has read a piece of data once, it stores that data in its cache. This greatly simplifies the process of installing Mondrian, but it puts limits on Mondrian's performance when Mondrian is applied to a huge dataset. (Hyde, 2009)

Aggregate tables are a way to improve Mondrian's performance when the fact table contains a huge number of rows: a million or more. An aggregate table is essentially a pre-computed summary of the data in the fact table.

An aggregate table coexists with the base fact table, and contains pre-aggregated measures built from the fact table. It is registered in Mondrian's schema, so that Mondrian can choose whether to use the aggregate table rather than the fact table, if it is applicable for a particular query.

Designing aggregate tables is a fine art. There is extensive research, empirical and theoretical, available on the Internet concerning different ways to structure aggregate tables and we will not attempt to duplicate any of it here.

A fact table can have zero or more aggregate tables. Every aggregate table is associated with just one fact table. It aggregates the fact table measures over one or more of the dimensions.

The problem of aggregate tables is that they are materialized views. A materialized view is a database object that contains the results of a query, what it means that they are using space in the DBMS and space is usually an expensive resource in BI/DW projects because of the total of information that is involved.

There are **three steps in order to implement aggregate tables**:

1. To **create** the aggregate table
2. To **populate** the aggregate table
3. To **reference** aggregate table in Mondrian Schema

C.3 Web application

The service layer, the business logic layer and the presentation layer with their corresponding technologies are presented in the next pages.

C.3.1 Service Layer

Once SDW and SOLAP are implemented, the next step us to implement data access layer. As it is mentioned in previous sections, an REST API, which abstracts the multidimensional model, provides the connection between business logic layer and data layer.

To implement this service, the framework selected is Jersey. The reason of using this framework is that it extends and simplifies the JAX-RS API, the standard java API to create RESTful Services. Moreover Jersey is open source and provides a lot of good documentation.

Resources are the key parts that compose a RESTful Web service. It is possible to manipulate resources using HTTP methods like GET, POST, PUT, and DELETE. Anything in the application can be a resource, in this case **renewable energy data**. In JAX-RX, resources are implemented by a POJO, with an @Path annotation to compose its *identifier*. A resource can also have sub resources.

There are some more annotations that help to implement this layer:

- @Context: Use this annotation to inject the contextual objects such as Request, Response, UriInfo, ServletContext, and so on.
- @Path("{v1}"): This is the @Path annotation combined with the root path "/rest" that forms part of the URI.
- @PathParam("measure"): This annotation injects the parameters into the path, measure (**wind speed and solar radiation**), in this case. Another parameters into the path are the “bbox” the “year”, the “month”, the “day” and the “resolution”.
- @Produces: Multiple MIME types are supported for responses. In Energy4People case, **application/json** will be the default MIME type.

At this point the data warehouse, the OLAP cube and the API REST are implemented. What it means that an important part of the backend is implemented. Now, it is necessary to implement a business logic layer where retrieved data can be processed and used to provide map visualization or to complete a financial or ecological model. These layers are implemented in **Java** programming language. Finally, a presentation layer, where users can interact with the application, is provided. This last layer is implemented using web technologies such as HTML, JavaScript and CSS.

C.3.2 Business Logic Layer

The business logic layer uses service layer to retrieve data required from the user when he or she interacts with presentation layer.

Once the data is retrieved, different operations can be applied to enrich the data.

As in design section is mentioned, logic in divided in different classes. They all are implemented in **Java**. To simplify the description of its implementation, they are divided in DAO, visualization and model.

DAO

The DAO pattern is chosen to design access to service layer. More details of this chosen are described in *Design* section.

The selection of **JSON** format to return data from service layer is considered to be a good chosen because of its lightness and the support in Java language.

Summarizing the functionality of this part of the system, the implementation of *RenewableEnergyDaoImpl* converts the user request via in an API REST call.

To implement connection with the API REST, **Jersey** Library is used. Once data is retrieved in JSON format, this is mapped into a **JSONArray** object from the JSON library for Java.

At this point, the data is sent to the Façade class, that is also implements in Java. This class can now send the data to the *KmlRender* or to the *EcoFinancialModel* class.

Visualization

If user is interacting with the map in the presentation layer, when data is retrieved from data layer, it is converted to KML. To implement this functionality two java libraries are essential: **JAK** (Java API for KML) and **JSON library for Java**.

The JSON library for Java is used to turn the input stream from the service layer into java objects, in this case JSONObject and JSONArray.

The main goal of the Java API for **KML** (JAK) is to provide automatically generated full reference implementation of the KML object model defined by OGC's KML standard and Google's GX extensions. It is an object orientated API that enables the convenient and easy use of KML in existing Java environments.

On one hand, KML provides a powerful way to display georeferenced data. It is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers. KML was developed for use with Google Earth but it also can be used with Google Maps. The KML format became an international standard of the Open Geospatial Consortium in 2008. It specifies a set of features (place marks, images, polygons, 3D models, textual descriptions, etc.) for display in Google Earth, Maps and Mobile, or any other geospatial software implementing the KML encoding. Each place always has a longitude and a latitude. Other data can make the view more specific, such as tilt, heading, altitude, which together define a "camera view" along with a timestamp or timespan. KML shares some of the same structural grammar as GML.

On the other hand, one of the problems of using KML as output format is the size of resulting file. This results in high latency, but Apache Software Foundation provides a library named **Commons IO**, which speeds up the file serialization. Then, it is send to presentation layer.

During the last phase of the project, it is tried to use some other kind of formats. Specifically GeoJSON. GeoJSON is a format for encoding a variety of geographic data structures. A GeoJSON object may represent a geometry, a feature, or a collection of features. GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. Features in

GeoJSON contain a geometry object and additional properties, and a feature collection that represents a list of features.

Model

The model that is presented in the next lines is developed is based on the reading of several sources, and simplicity in the implementation. Fernando Martin and Albert Zurita, engineers from European Spatial Agency documented this model during the "Space Apps Challenge" contest. They have several publications, for example, 'Towards a SMOS Operational Mission: SMOSOps-Hexagonal' and 'SEOSAR/PAZ Mission: Panel Design and Performance'.

My work in this model is to implement in Java and finally integrate with the BI system that is already defined.

The system requirements do not specify complex models, just the possibility to add models to the retrieved data. With models, the Energ4People project aims to provide a simple and fast translation between data units, for example, metres per second and time to recover the investment.

Summarizing, the models allows the estimation of energy generation at a given location based on the available renewable energy resources and the proposed facility. The user can describe the facility based on an intuitive list of parameters:

- Surface of solar panels
- Number of wind turbines
- Quality of the facility

The description of the facility quality is based on available commercial products:

Solar facility

Type	Price per m2 ($/m2)	Max Power per m2 (W/m2)	Efficiency	Installation cost ($/m2)
High	600	217	22.5	666

Type	Price per m2 ($/m2)	Max Power per m2 (W/m2)	Efficiency	Installation cost ($/m2)
Medium	460	210	19.3	666
Low	111	152.5	15.9	666

Table C.1: Solar facility table.

Wind facility

Type	Generated power (kW)	Installation cost ($/turbine)
Expensive	3000	3.300.000
Medium	1500	1.500.000
Cheap	650	460.000

Table C.2: Wind facility table.

Once the facility is described, the user selects where to place it either using a specific address or by manually entering the latitude and longitude coordinates.

The provided results are:

- The daily average generated energy
- The number of homes to be powered with that energy
- The equivalent value of the generated energy
- The facility cost
- Time to recover the investment

In the future versions of this project, these models must be improved in order to have more accuracy results. After many executions of these models, the results are not exaggerated and they correspond with usual results described on the Internet.

C.3.3 Presentation Layer

In presentation layer user can interact with the application to solve some needs registered in functional requirements and usability requirements.

The technologies that are used, and their mission are explained in next lines:

HTML Hypertext Markup Language, the mark-up language used to create documents on the World Wide Web. HTML defines the structure and layout of a Web document by using a variety of tags and attributes.

JavaScript, jQuery and Ajax: JavaScript is an interpreted computer programming language implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. Ajax is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. JQuery is a multi-browser JavaScript library designed to make it easier to navigate a document, select DOM (Document Object Model) elements, to handle events and some other features.

CSS (Cascading Style Sheets) and Twitter bootstrap: CSS is a style sheet language used for describing the presentation look and formatting of a document written in a markup language. In order to standardize development of interface components, Twitter bootstrap has been used beside jQuery-UI. Twitter Bootstrap is a free collection of tools for creating websites and web applications.

Google Maps: The Google Maps JavaScript API lets you embed Google Maps in your own web pages. This JavaScript library has great importance in Energy4People web application.

Energy4People web application uses Google maps component to send Ajax requests to business logic layer. When zooming or panning on the map, new data is retrieved. When this occurs, the business logic transforms the data in the KML files that can be displayed on the map.
Otherwise, if the user uses menu to apply a model to the data, the table displays the results. These results are sent to the presentation layer in JSON format. In this case, *EcoFinancialModel* class enriches the retrieved data.

C.4 Machine technical specification

This is the machine where the project is developed. Although it has features that are thought to be enough in this project, some operations like interpolation and the load process in a column-oriented DBMS have made that the machine has suffered important performance degradation.

Component	Technical Description
Processor	2,5 Ghz. Intel Core i5
Memory	16 GB 1600 MHz DDR3
Graphics	Intel HD Graphics 4000 512 MB
OS	OS X 10.8.2

Table C.3 Machine Technical Specifications.

D. Tests

This section presents the documents generated after passing the functional tests. This kind of documentation is thought to be a good format to document and organize tests in order to register problems and avoid future errors.

D.1 Functional Tests Documentation

The functional test document present the next structure:

Title: Document X.

Date: The date when the test is executed and documented.

Requirement name: This part contains the requirement name that is going to be tested.

Description: This part describes the steps that are followed in test case execution.

Preconditions*:* This part describes the state of the system before the test case can be performed.

Expected Results: As the system is expected to behave.

Pass/Fail: This part describes if the test has the response that is expected (pass) or not (fail).

Comments: Some comments are documented only if it is necessary.

I execute all these tests with Naveen Sidda, the director of the project, observing the results and checking if the observed results correspond with his goals.

DOCUMENT 1

Date: 29/08/2012

Requirement name: Central repository

Description:

This test must demonstrate that the data is stored in a central repository.

PgAdmin III tool (a Postgresql DBMS client) is going to be used.

1. I connect with Postgresql DBMS server.
2. I open central repository, called Energy4People_SDW
3. I open the *TemporalDimension* table, and Naveen Sidda observes the range of years.
4. I open the *SpatialDimension* table, and Naveen Sidda observes the range of coordinates.
5. I open the *FactTable table* and Naveen Sidda observes the solar radiation and wind speed data.

Preconditions*:* Any tools are not accessing central repository.

Expected Results: *TemporalDimension, SpatialDimension and FactTable* tables contain the expected data.

Pass/Fail: Pass

DOCUMENT 2

Date: 29/08/2012

Requirement name: multi-scalar visualization

Description:

This test must demonstrate that Energy4People allows multi-scalar visualization of data.

1. I load Energy4People web application with Google Chrome.
2. I zoom in three times.
3. Naveen Sidda observes the results.

Preconditions*:* Network connection must be available

Expected Results: Google Maps must display different data with each zoom.

Pass/Fail: Pass

DOCUMENT 3

Date: 29/08/2012

Requirement name: Hierarchical temporal data.

Description:

This test must demonstrate that Energy4People allows browsing through the temporal data.

1. I load Energy4People web application.
2. I request for solar radiation average in the year 1990 and Naveen Sidda sees the results.
3. I request for solar radiation average in July 1990 and Naveen Sidda compares the results with NASA website.
4. I request for solar radiation average in July 3, 1990 and Naveen Sidda compares the results with NASA website.

Preconditions*:* Network connection must be available

Expected Results: Google Maps must display different data with each request.

Pass/Fail: Pass

DOCUMENT 4

Date: 29/08/2012

Requirement name: Models applicability

Description:

This test must demonstrate that Energy4People allows apply models.

1. I load Energy4People web application with Google Chrome.
2. I open the Financial Menu.
3. I introduce 7000 in surface of solar panels.
4. I introduce 100 in number of wind turbines.
5. Naveen observes the results.

Preconditions*:* Network connection must be available

Expected Results: Energy4People table is filled of reasonable data.

Pass/Fail: Pass

Comment: Although the test passes because the application allows models integration, which is the requirement, Naveen Sidda insists that in the future, models must be improved.

DOCUMENT 5

Date: 29/08/2012

Requirement name: Interpolation

Description:

This test must demonstrate that Energy4People allows data interpolation.

PgAdmin III tool (a Postgresql DBMS client) is going to be used.

1. I connect with Postgresql DBMS server.
2. I open central repository
3. I open Staging Table with original data from NASA website.
4. I open the *FactTable* with the interpolated data.
5. Naveen observes the data.

Preconditions*:* Any tools are not accessing central repository.

Expected Results: Interpolated data is coherent with original data.

Pass/Fail: Pass

DOCUMENT 6

Date: 29/08/2012

Requirement name: Web scraper

Description:

This test must demonstrate that Energy4People allows to apply scraping to NASA website.

1. I configure scraper to request for wind speed in latitude 39 and longitude 2 in the year 2000.
2. I configure scraper so that it prints the results of scraping in a log file.
3. I launch scraper.
4. I open NASA website and I request for wind speed in latitude 39 and longitude 2 in the year 2000.
5. Naveen Sidda compares data.

Preconditions*:* Network must be available.

Expected Results: The data in logs and the website is the same

Pass/Fail: Pass

DOCUMENT 7

Date: 29/08/2012

Requirement name: NASA sources

Description:

This test must demonstrate that the data source that is used in Energy4People comes from NASA website.

PgAdmin III tool (a Postgresql DBMS client) is going to be used.

1. I connect with DBMS server.
2. I open central repository
3. I open Staging Table with the original data from NASA website displaying data from latitude 41 and longitude -1.
4. I open NASA website and request for solar radiation in latitude 41 and longitude -1.
5. Naveen Sidda compares data.

Preconditions*:* Any tools are accessing central repository.

Expected Results: The data in the central repository and the website is the same.

Pass/Fail: Pass

DOCUMENT 8

Date: 29/08/2012

Requirement name: To use Google Maps

Description:

This test must demonstrate that Energy4People uses Google Maps to display data.

1. I load Energy4People web application.
2. I zoom in.
3. Naveen Sidda sees the results.

Preconditions*:* Network connection must be available

Expected Results: Google Maps must display data when they zooming in.

Pass/Fail: Pass

DOCUMENT 9

Date: 29/08/2012

Requirement name: Google Chrome and Mozilla Firefox

Description:

This test must demonstrate that Energy4People works on Google Chrome and Mozilla Firefox.

1. I load Energy4People web application with Google Chrome.
2. I zoom in.
3. Naveen Sidda observes the results.
4. I load Energy4People web application with Mozilla Firefox.
5. I zoom in.
6. Naveen Sidda observes the results.

Preconditions*:* Network connection must be available

Expected Results: Both browsers must retrieve data and display it on the map.

Pass/Fail: Pass

DOCUMENT 10

Date: 29/08/2012

Requirement name: KML format

Description:

This test must demonstrate that Energy4People uses KML format data.

1. I load Energy4People web application.
2. I zoom in.
3. In the server side instead of sending data to the client, the server writes results in a file with a 'kml' extension.
4. I open file with Google Earth Desktop Application.
5. Naveen Sidda sees the results.

Preconditions*:* Network connection must be available

Expected Results: Google Earth must display file.

Pass/Fail: Pass

D.2 Usability Tests Documentation

USABILITY TEST 1

Date: 05/04/2012

Results table

Task	Measure	Actual value	Objective value
Load Application	Time	3 sec.	3 sec.
First query	Number of errors	9	3
Visualize Zaragoza HSR Data (1990)	Time	2.30 min.	50 sec.

Table D.1: Results table of the first usablity test.

Subjective opinion

- Really poor interaction
- Using Google Earth desktop application is not good.
- It is not interesting to show MDX queries.

Conclusion

- Google Earth Desktop Application must disappear
- User has too many doubts when interacting with the tool.
- Web application must be taken into account in the next software version.

USABILITY TEST 2

Date: 3/5/2012

Results table

Task	Measure	Actual value	Objective value
Load Application	Time	12 sec.	3 sec
First query	Number of errors	4	3
Visualize Zaragoza HSR Data (1990)	Time	30 sec.	10 sec.
Visualize Zaragoza WS Data (1990:July)	Number of errors	8	4
Apply Financial Model	Number of errors	12	2

Table D.2: Results table of the second usablity test.

Subjective opinion

- Poor interaction.
- Google Earth is cool but slow.
- Menu is not intuitive.
- Google Earth fails sometimes.
- Only Zaragoza data must be displayed, not all Spain data.

Conclusion

- Google Maps to improve map interaction.
- Less and more intuitive UI components.
- Bounding box to limit the data retrieval

USABILITY TEST 3

Date: 19/07/2012

Results table

Task	Measure	Actual value	Objective value
Load Application	Time	4 sec.	3 sec.
First query	Number of errors	2	3
Visualize Zaragoza HSR Data (1990)	Time	15 sec.	10 sec.
Visualize Zaragoza WS Data (1990:July)	Number of errors	5	4
Apply Financial Model	Number of errors	7	2

Table D.3: Results table of the third usablity test.

Subjective opinion

- A bit slow but good enough.
- Units must be specified.
- Some administrative unit shape must be shown to guide analysis.
- Colour scale must be added to the UI.

Conclusion

- Few changes in UI are must be applied.
- Improve performance.

USABILITY TEST 4

Date: 23/08/2012

Results table

Task	Measure	Actual value	Objective value
Load Application	Time	4 sec.	3 sec.
First query	Number of errors	2	3
Visualize Zaragoza HSR Data (1990)	Time	15 sec.	10 sec.
Visualize Zaragoza WS Data (1990:July)	Number of errors	6	4
Apply Financial Model	Number of errors	5	2

Table D.4: Results table of the fourth usablity test.

Subjective opinion

- The better version.
- More Intuitive.
- Although colours are intuitive, a colour scale must be included in the UI.

Conclusion

- In future versions, responses must be faster.
- More geospatial information.
- Model results must improve.
- Colour scale is really important.

E. Results

The pictures show some of the results of each iteration.

E.1 First iteration

The result of the first iteration of the project is a Desktop application that generates KML files after reading a SDW with solar radiation information. Then, as the Figure E.1 shows, the KML file can be open with Google Earth Desktop application.

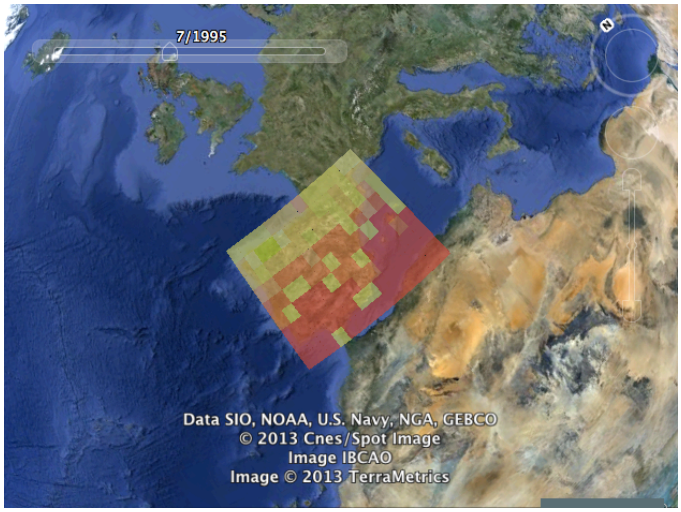

Figure E.1: Google Earth Desktop Application displaying a KML file generated by the first developed application

E.2 Second iteration

The Figure E.2 shows the web application I had developed before I went to the NASA contest. It was the Energy2People prototype.

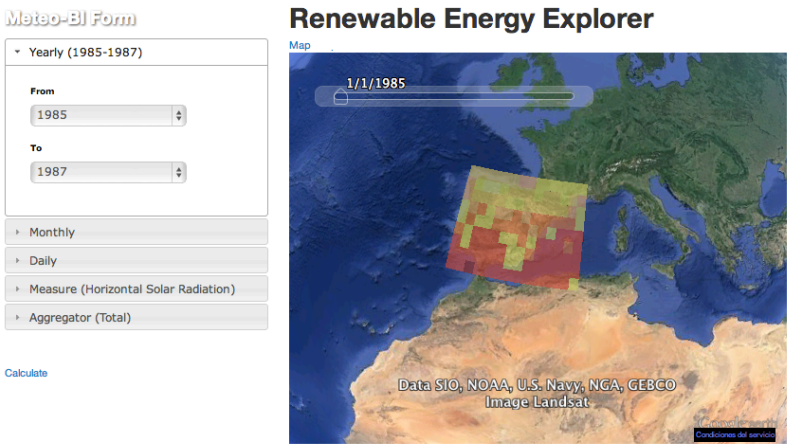

Figure E.2: Energy2People prototype. The web application before the NASA contest.

The Figures E.3 and E.4 illustrate the Energy2People final aspect. A better user interface, next to some other functionalities (graphics generation and model application) are added.

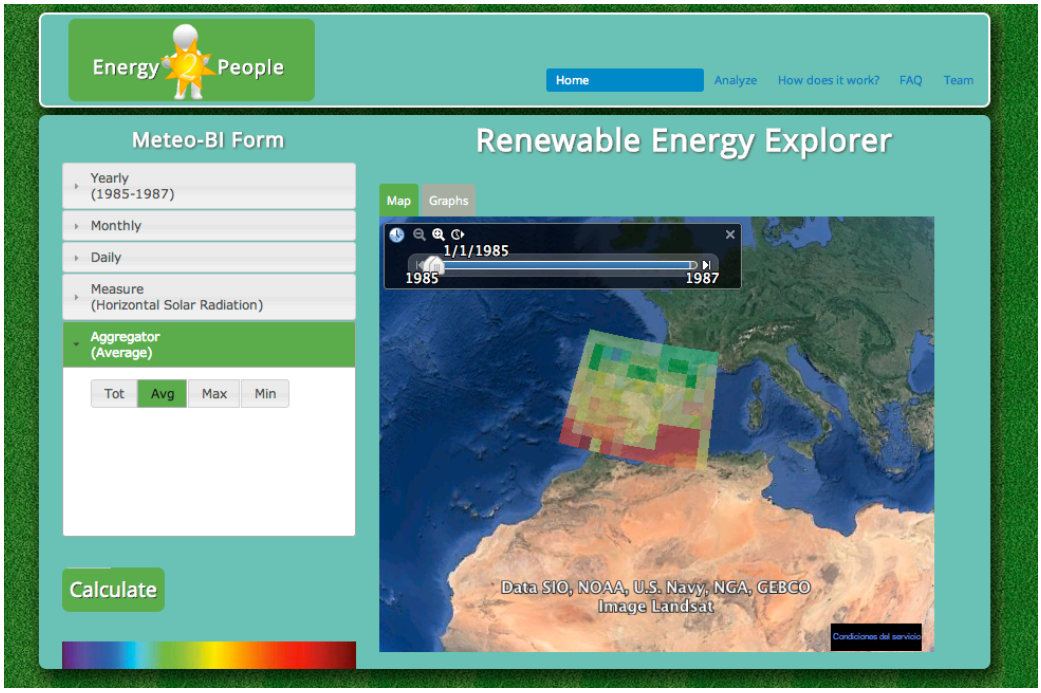

Figure E.3: Energy2People. Map Analysis

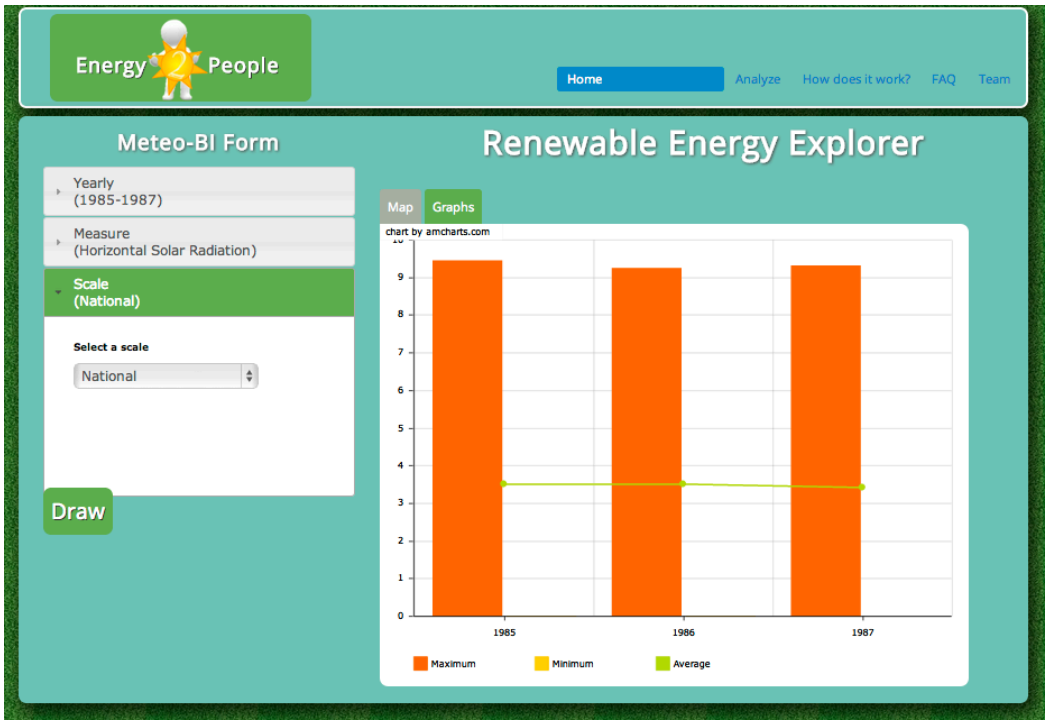

Figure E.4: Energy2People. Graphics generation.

E.3 Third iteration

After Energy2People, some improvements are needed. The most important is the possibility to apply multi-scalar analysis, a type of cartographic generalization. Moreover, KML files generation is improved to create more meaningful styles.

The next pictures show the improvement of KML files generation. These new files must be open using Google Earth Desktop Application because Google JavaScript libraries do not support so large files.

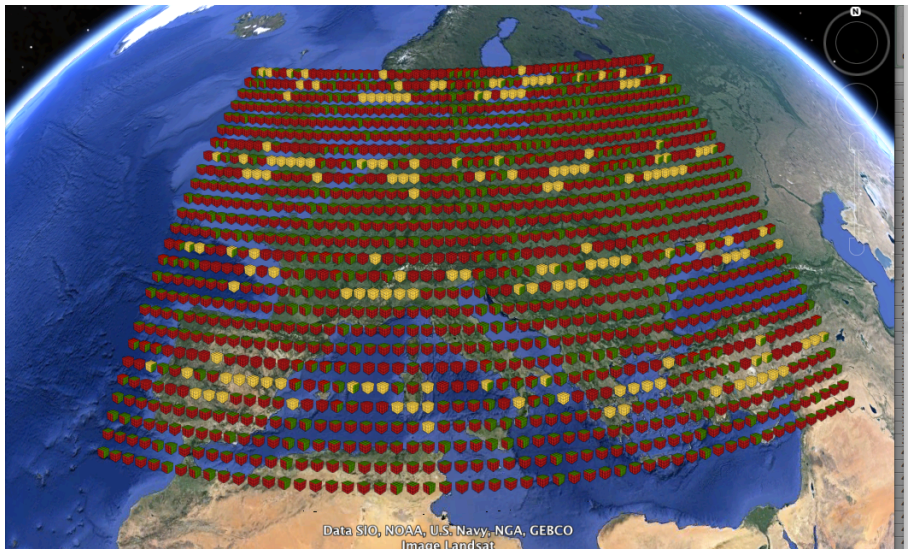

Figure E.5: Coloured cubes KML file. Depending on the colour of each cube face, a place has a high, medium or low solar radiation, wind speed and precipitation.

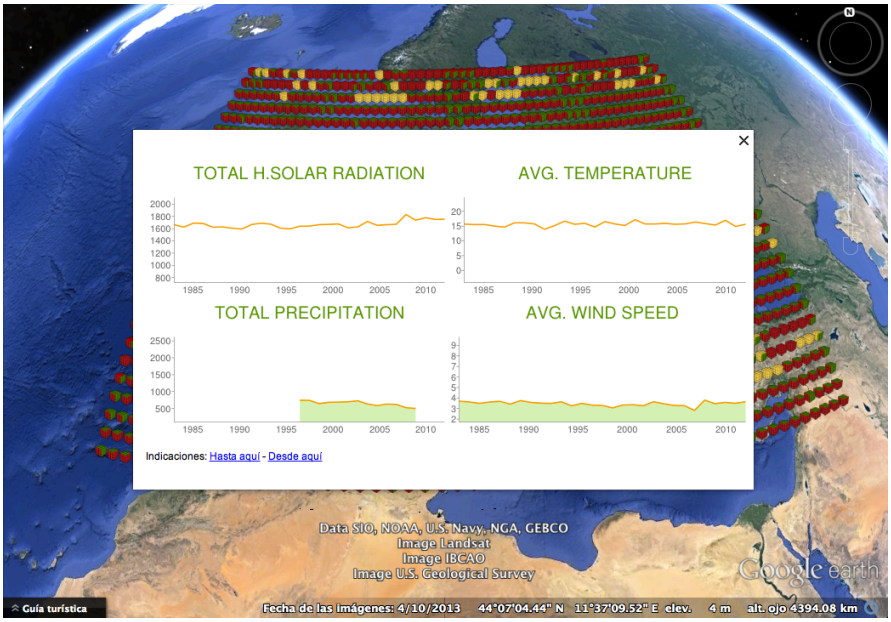

Figure E.6: Coloured cubes KML file after clicking one of the cubes.

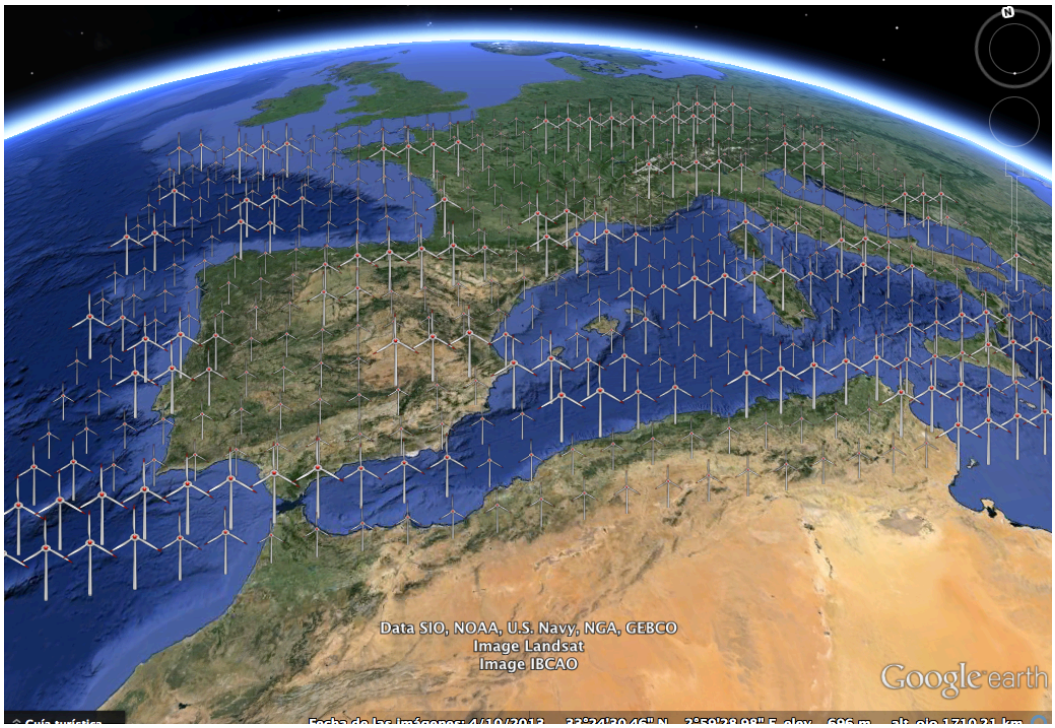

Figure E.7: Windmills KML file. Depending on the windmill size, the place has a high, medium or low wind speed average.

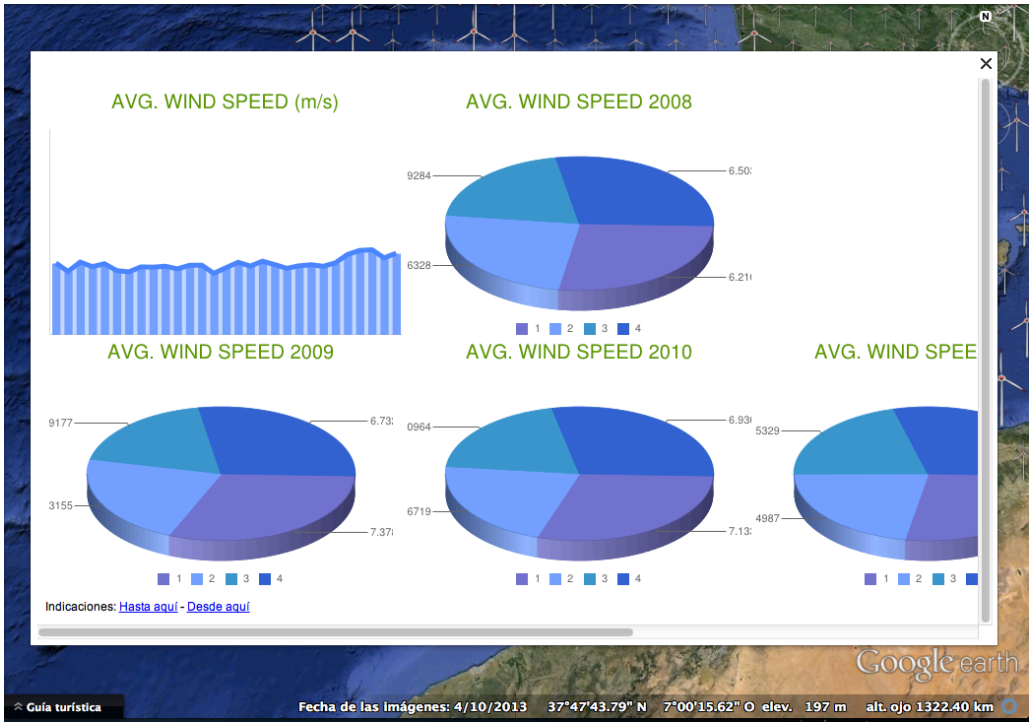

Figure E.8: Windmills KML file after clicking one of the windmills.

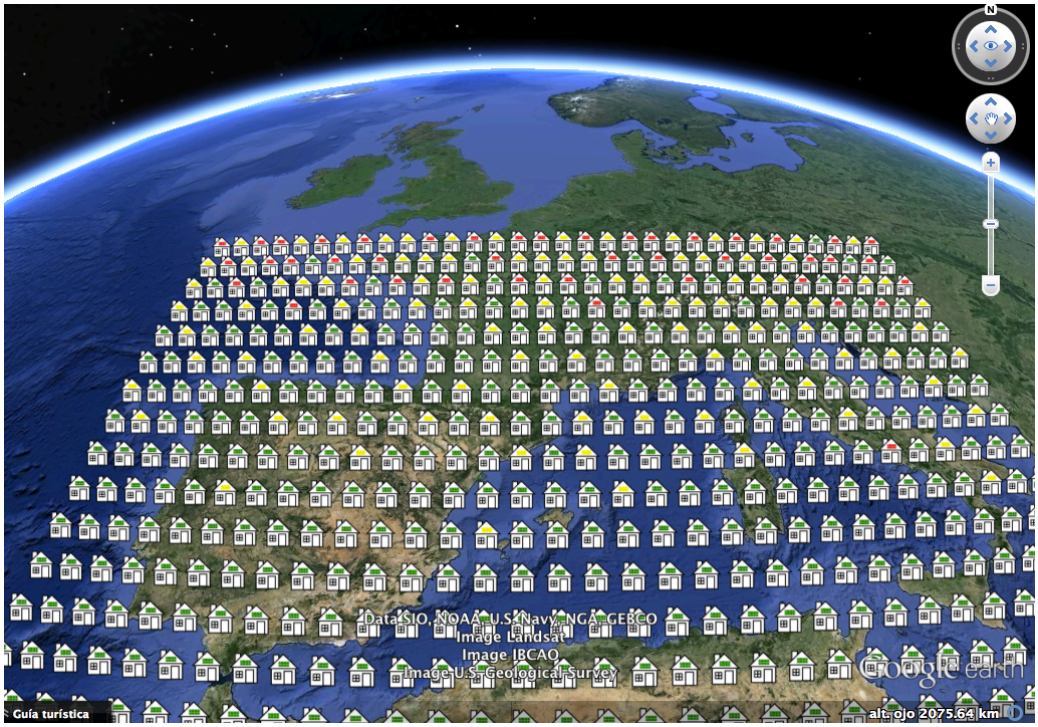

Figure E.9: Coloured houses. The colour of the house roof depends on the solar radiation of the place.

The Figure E.10 shows Energy4People before starting multi-scalar analysis.

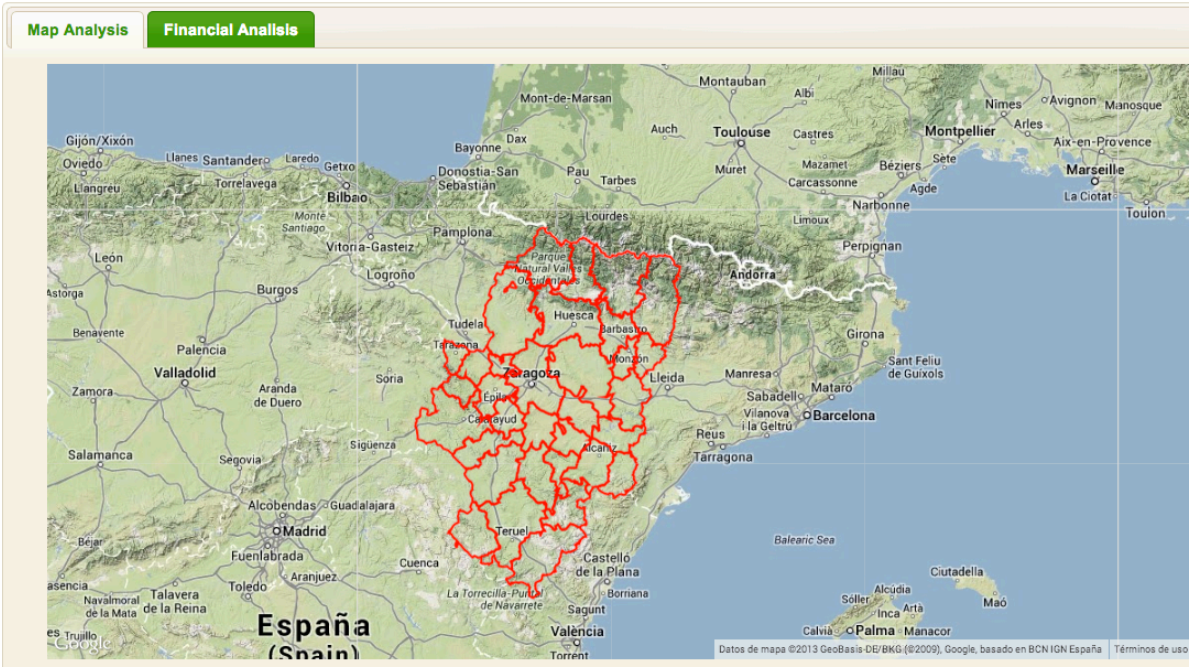

Figure E.10: Energy4People Map Analysis. Aragon is displayed on the map in order to ease analysis

Energy4People application is thought to be **the first "Renewable Energy Explorer" that provides multi-scalar analysis**. The next figures show this powerful functionality.

First zoom

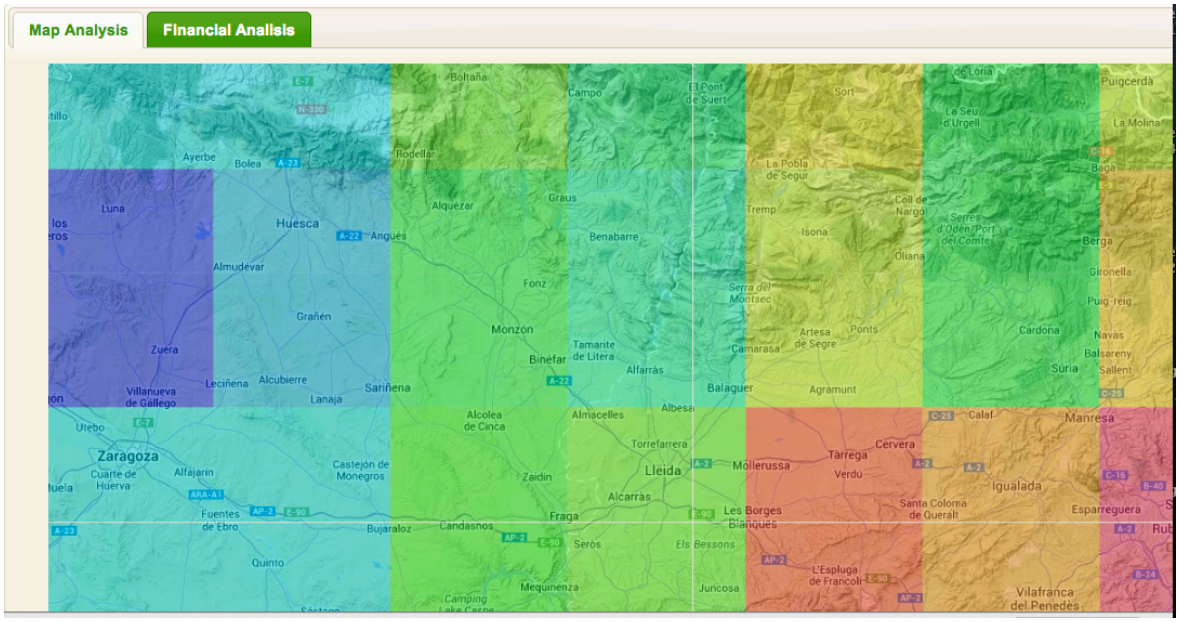

Figure E.11: Energy4People Multi-scale Analysis. First level. Data resolution = 1 X 1.

Second zoom

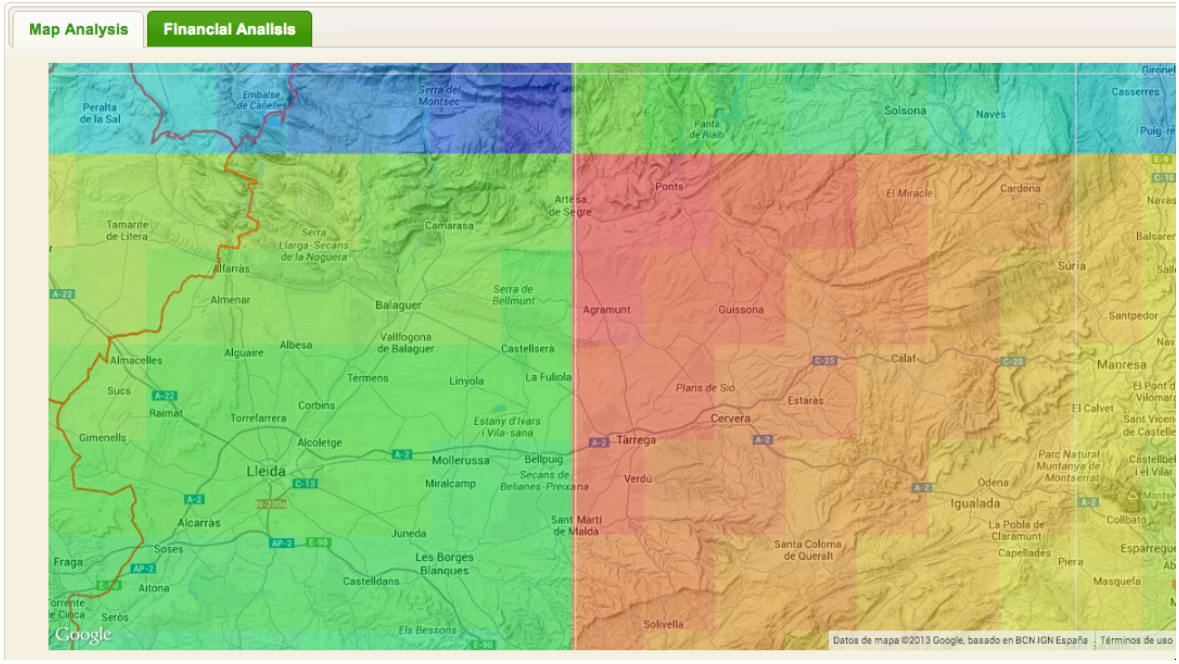

Figure E.12: Energy4People Multi-scale Analysis. Second level. Data resolution = 0.1 X 0.1.

Third zoom

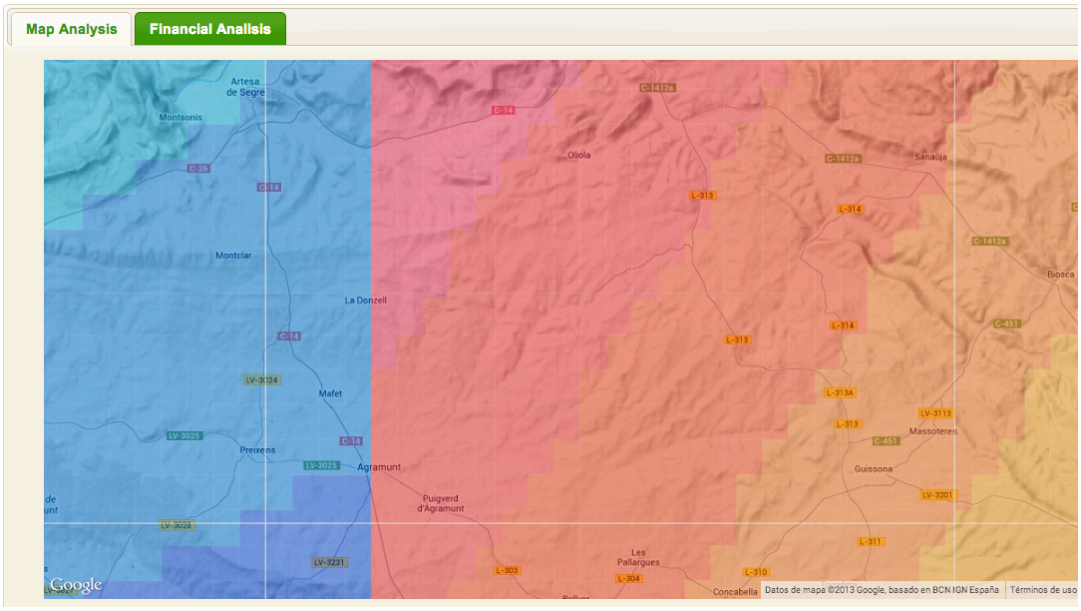

Figure E.13: Energy4People Multi-scale Analysis. Third level. Data resolution = 0.01 X 0.01.

And finally, all the KMLs of different resolutions displayed on Google Earth.

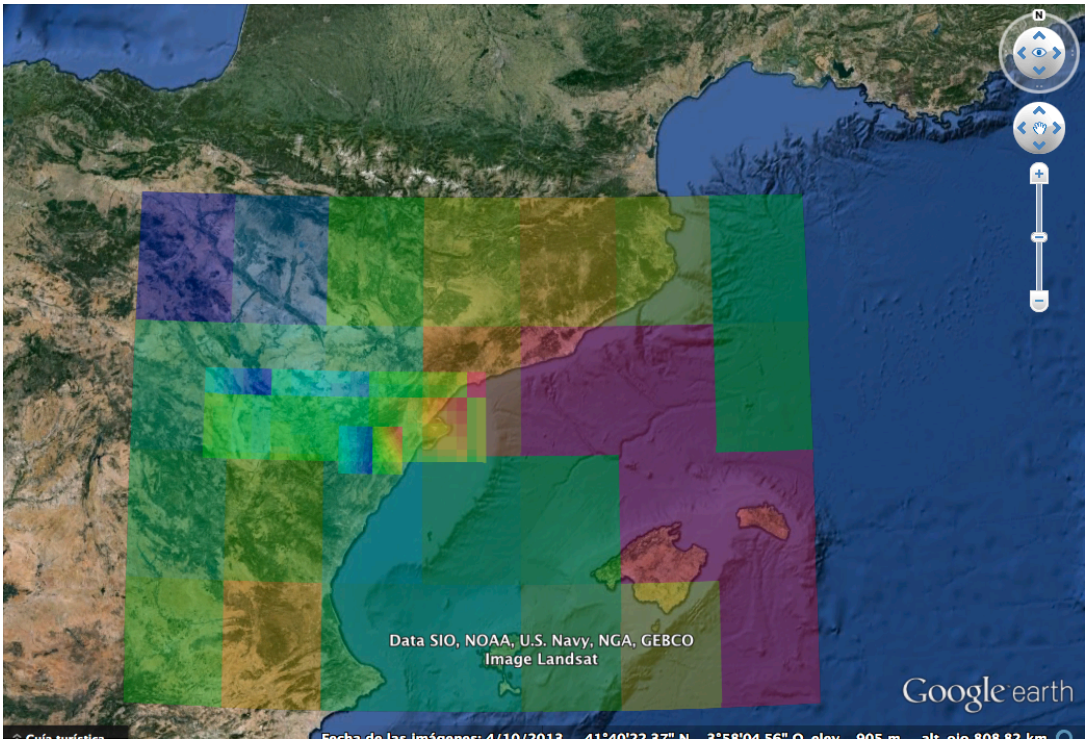

Figure E.14: All scales displayed on Google Earth Desktop Application.

E.4 Fourth iteration

To fix some bugs, new future work lines inspection and finally this document.

F. Project Management

F.1 Scrum

In order to manage Energy4People project it is used Scrum Agile Methodology and Lean. Scrum Project management is a software agile development process. Scrum models allow projects to progress via a series of iterations called agile sprints. The agile methodology and Scrum process is iterative and incremental, so the project is split into a series of consecutive sprints.

It is important to say that apply all Scrum rules is really difficult because of the PFC features. But some adjustments are done to achieve some of the Scrum characteristics that are found really interesting in this project context. For example, this is the case of roles that are shown in Table F.1.

Role	Brief Description	Person
Customer	The person who has a necessity. In this project, he needs some technical help.	Naveen Sidda
Product Owner	The 'voice' of the customer. In this project, I need to translate customer's necessities in a computer science project.	Borja Espejo
Scrum Master	Project Manager. In this project, Javier López advises me with some project management problems.	Fco. Javier López
Team	Software Developers. In this project I am the only developer.	Borja Espejo

Table F.1: Implemented Scrum rols in Energy4People Project.

F.2 Project estimations

This project is done in 4 sprints. Each sprint adds some new functionalities to the system. The effort to finish these new functionalities is estimated before each sprint. After each sprint, the real effort is calculated and then the computed value is compared with the estimated. It is important to say that some tasks such as studying BI techniques are done for more than one sprint. The table F.2 shows this comparison.

ID	Description	Estimated (hours)	Real (hours)	% Deviation
1	As a provider, I need to study BI techniques and documenting	180	200	11
2	As a provider, I need to know the available BI technologies	70	50	-29
3	As a user, I want NASA as data source	40	50	25
4	As a user, I want a central repository	15	25	66.7
5	As a user I want to use OLAP cubes to access data	20	25	25
6	As a user I want an application than access data and generates KMLs	10	15	50
To study BI/DW and develop and application to validate knowledge and technologies.		335	365	8.96
8	As provider I need a tool that helps me to implement ETL processes	15	20	33.3
9	As a user I want some improvements in central repository	40	50	25
10	As a user I want some improvements in temporal dimension in OLAP	40	50	25
11	As a user I want a web application to access data	50	30	-40
ETL Tool and new improvements in data model. Energy2People		145	150	3.45
12	As a user, I want to interpolate data in order to have better resolution	60	70	16.7
13	As a user, I want to do some improvements in central repository	60	50	-17
14	As a user, I want some improvements in spatial dimension in OLAP	55	60	9.09
15	As a user I want Google Maps to display data	15	10	-33
16	As a user, I want to change the way to access data	30	20	-33
New improvements in backend and frontend. Energy4People		220	210	-4.5
17	As a provider, I need to test new technologies to improve my business	70	100	42.9
18	As a student, I need to organize documents generated	10	15	50
19	As a student, I need to write a report	100	120	20
20	As a student, I need to write an annex	120	140	16.7
To study new technologies and generate final documentation		300	375	25
TOTAL		**1000**	**1100**	**10**

Table F.2: Energy4People tasks with its estimated and real duration. It is also added the percent error.

G. Business Intelligence

The main goal of this section is to provide a global vision of Business Intelligence. The information is extracted from different **sources** (books, papers, web resources…). They all **are referenced, and can be found in the bibliography.**

G.1 Introduction to Business Intelligence

Business intelligence (BI) "is a business management term, which refers to applications and technologies that are used to gather, provide access to, and analyse data and information about company operations and performance." (Council, 2008)

BI systems help companies to have a more comprehensive knowledge of the factors affecting their business, such as metrics on sales, production, and internal operations. Thanks to this, companies can make better business decision. BI applications support the activities of decision support, query and reporting, online analytical processing (OLAP), statistical analysis, forecasting, and data mining.

But BI is not only a system to respond to a specific business need. Rather it is a change in how people do business. This change is built upon having the information, processes, and tools needed to make decisions. Information comes from many locations both inside and outside the organization. It is important to note that common business definitions across the organization and ensuring that errors and duplicates are eliminated before being loaded in the system are critical for success.

Some companies see a great opportunity combining the features of BI systems with GIS systems. As an ESRI (2012) paper described,

> BI is a priority for organizations interested in gaining a competitive advantage. BI leverages corporate data and strategically equips knowledge workers with insights that drive sound business decisions. As BI has matured, the reach of GIS has expanded significantly as well. In addition to specialty IT groups, GIS provides agility to a multitude of departments in many industries. It allows users to visualize and intelligently analyse historically underutilized data in ways not typically seen in traditional BI implementations. Given the complementary natures of BI and GIS, the adoption of geographic analysis to enhance business intelligence is growing

rapidly. Through the fusion of these two enterprise technologies, organizations can visualize and analyse key business data through "smart" maps to discover patterns and trends that would have been easily overlooked with traditional BI tables and charts.

With the dynamic economic landscape, businesses are increasingly looking for ways to do more with less and maximize their existing assets to extract the most value. To achieve this, BI has been a significant component in many organizations' technology portfolios. Well-implemented BI “allows organizations to focus on what's important and make business decisions to drive performance.” (ESRI, 2012)

The evolution from simple reporting and online analytical processing (OLAP) through scorecards and dashboards to predictive and prescriptive analytics shows a constant expansion of value-added capabilities in the area of BI and analytics (they are not the same). Organizations have become increasingly sophisticated in the application of new and complementary technologies in this domain to drive ever-better business decisions.

Although traditional BI tools are powerful and have delivered proven results, they do not incorporate a crucial component of most business information: location, the “Spatial Dimension” of the data. The majority of business data contains some sort of location information: office locales, customer addresses, sales territories, marketing areas, facilities, and so on, it is thought that 80% of the data has spatial content. So, when this data is viewed spatially on a map, patterns and trends that were once overlooked are clearly revealed.

When combining GIS with business intelligence data, organizations can answer new questions in order to obtain a competitive advantage.

G.2 ETL

Extract, Transform and Load (ETL) is a fundamental process used to populate a data warehouse. It involves extracting data from various sources, transforming the data according to business requirements and loading them into target data structures. Inside the transform phase a well-designed ETL system should also enforce data quality, data consistency and conforms data so that data from various source systems can be integrated. Once the data is loaded into target systems in a presentation-ready format, the end users can run queries against them to generate reports which help them make better business decisions. “Even though ETL process consumes roughly 70% of computing resources they are hardly visible to the end users.” (Caserta & Kimball, 2004)

The Figure 18 shows an ETL process. It can be seen that there can be several data sources that have characteristics that differ from each other.

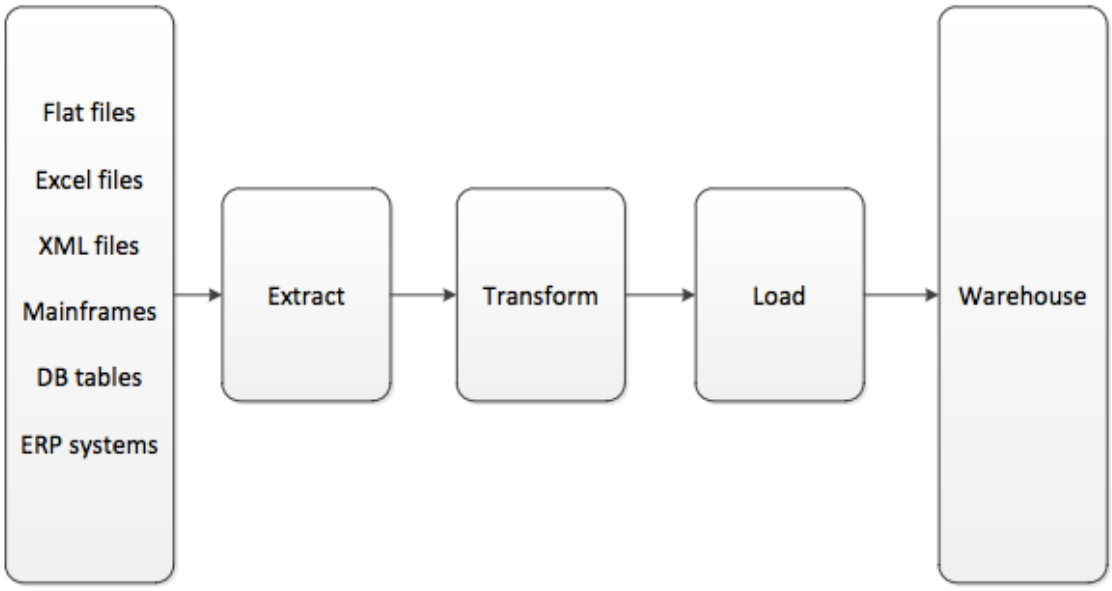

Figure G.1: ETL process. (Vijayendra, 2005)

G.3 Data warehouse

A data warehouse is, by definition, "a subject-oriented, integrated, time-variant collection of data to enable decision making across a disparate group of users. Data warehousing is the design and implementation of processes, tools, and facilities to manage and deliver complete, timely, accurate, and understandable information for decision-making." (Inmon, 2005)

The concept of data warehousing has evolved out of the need for easy access to a structured store of quality data that can be used for decision-making. It is globally accepted that information is a very powerful issue that can provide huge benefits to any organization and a competitive advantage in the business world.

As Ralph Kimball described in the book "The Data warehouse toolkit",

> Organizations have huge amounts of data but have found it increasingly difficult to access it and make use of it. This happens because data is in many different formats, exists on many different platforms, and resides in many different file and database structures developed by different vendors. Thus organizations have had to write and maintain perhaps hundreds of programs that are used to extract, prepare, and consolidate data for use by many different applications for analysis and reporting. Also, decision makers often want to dig deeper into the data once initial findings are made. This would typically require modification of the extract programs or development of new ones. This process is costly, inefficient, and very time consuming. Data warehousing offers a better approach. Data warehousing implements the process to access heterogeneous data sources; clean, filter, and transform the data; and store the data in some kind of structure that is easy to access, understand, and use. The data is then used for query, reporting, and data analysis. As such, the access, use, technology, and performance requirements are completely different from those in a transaction-oriented operational environment. The volume of data in data warehousing can be very high, particularly when considering the requirements for historical data analysis.

Spatial data warehouses aim at effective and efficient querying of spatial data. Spatial databases are suited for answering regular transactional queries where there is not a lot of historical component or aggregation. The class of queries that are needed to support the

decision making process are difficult on spatial databases. This gave a rise to the field of spatial data warehouses, which is idea of combining the traditional data warehouses with spatial databases. Spatial data warehouses "are based on the concepts of Data warehouses and additionally provide support to store, index, and aggregate and analyse spatial data." (MacEachren & Kraak., 2001)

The last years, spatial data has increasingly become part of operational and analytical systems in various areas, such as public administration, transportation networks, environmental systems, and public health, among others. Spatial data can represent either objects located on the Earth's surface, such as mountains, cities, and rivers, or geographic phenomena, such as temperature, precipitation, and altitude. The amount of spatial data available is growing considerably owing to technological advances in areas such as remote sensing and global navigation satellite systems (GNSS), namely the Global Positioning System (GPS) and the forthcoming Galileo system. All these kinds of technologies and trends are part of the Neogeography, which is though to be the future of GIS. Neogeography can be seen as the future of GIS.

The management of spatial data is carried out by spatial databases or geographic information systems (GISs). Since the latter are used for storing and manipulating geographic objects and phenomena, it is more appropriate the general term "spatial databases". Spatial databases allow one to store spatial data whose location and shape are described in a two or three-dimensional space. These systems provide a set of functions and operators that allow users to query and manipulate spatial data. Queries may refer to spatial features of individual objects, such as their area or perimeter, or may require complex operations on two or more spatial objects. Topological relationships between spatial objects, such as "intersection", "inside", and "meet", are essential in spatial applications. One technology that supports this kind of operations is Postgis. Some examples of these geospatial operations can be: two roads may intersect, a lake may be inside a state, two countries may meet because they have a common border, etc. An important characteristic of topological relationships is that they do not change when the underlying space is distorted through rotation, scaling, and similar operations.

Although spatial databases offer sophisticated capabilities for the management of spatial data, they are typically targeted toward daily operations, similarly to conventional operational databases. As we have seen, data warehouses provide OLAP capabilities for

analysing data using several different perspectives, as well as efficient access methods for the management of high volumes of data. On the other hand, spatial databases provide sophisticated management of spatial data, including spatial index structures, storage management, and dynamic query formulation. Spatial data warehouses "allow users to exploit the capabilities of both types of systems for improving data analysis, visualization, and manipulation." (Malinowski & Zimányi, 2009)

A data warehouse can be modelled with different schemas: Star Schema, Snowflake schema and Constellation schema. They all are explained deeply in next paragraphs.

Star Schema

In a dimensional model, each group of dimensions is placed in a *dimension table*; the facts are placed in a *fact table*. The result is a star schema, so called because it resembles a star when diagrammed with the fact table in the centre as in the Figure G.2.

The dimension tables in a star schema are usually wide. They can contain a large number of attributes providing rich contextual data to support a wide variety of reports and analyses. This will allow the data warehouse to track the history of changes to data elements, even if source systems do not.

Fact tables are deep. They contain a large number of rows, each of which is relatively compact. Foreign key columns associate each fact table row with the dimension tables. The level of detail represented this is referred to as *grain*. (Adamson, 2006)

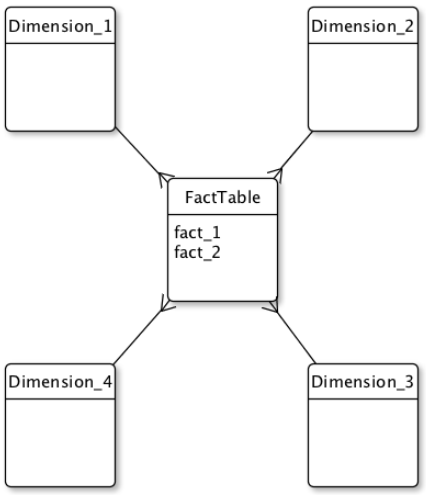

Figure G.2: Star schema.

Snowflake schema

A snowflake schema can be described as a star schema with fully normalised dimensions. It gets its name because it forms a shape similar to a snowflake. Rather than having a regular structure like a star schema, the "arms" of the snowflake can grow to arbitrary lengths in each direction. This shape can be observed in the Figure G.3. A snowflake schema "can be produced from a star schema by normalising each dimension table. Instead of collapsing classification entities into component entities to form dimension tables, they form the "arms" of the snowflake". (Kortink, 2010)

It is also argued that "snowflaking" is not desirable because it adds unnecessary complexity, reduces query performance and doesn't substantially reduce storage space. "However empirical tests show that the performance impact of "snowflaking" depends on the DBMS and/or OLAP tool used: some favour snowflakes while others favour star schemas" (Spencer & Loukas, 1999). An advantage of the snowflake representation is that it explicitly shows the hierarchical structure of each dimension, which can help in understanding how the data can be sensibly analysed. Thus, "whether a snowflake or a star schema is used at the physical level, views should be used to enable the user to see the structure of each dimension as required." (Kimball, 2002)

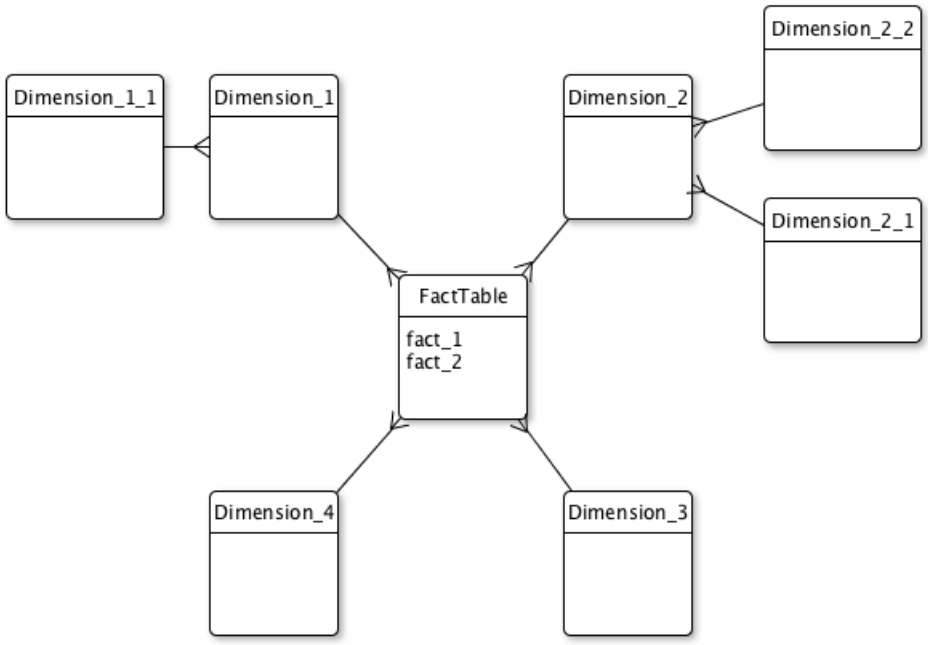

Figure G.3: Snowflake schema.

Constellation Schema

As it is illustrated in the Figure G.4, for each star schema it is possible to construct so-called fact constellation schema. The main disadvantage of the constellation schema is a more complicated design because many variants for particular kinds of aggregation must be considered and selected. Moreover, dimension tables are still large.

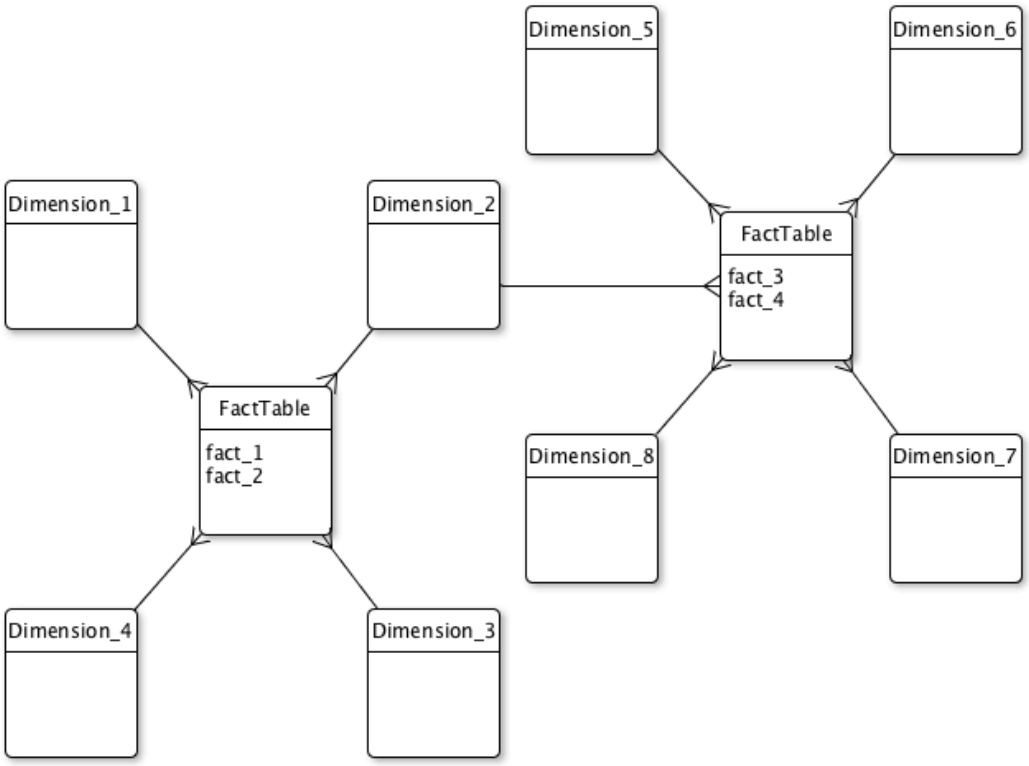

Figure G.4: Constellation schema.

Data mart

A data mart is a straightforward structure of a data warehouse that is attended on a distinct subject (or functional area), such as Finance, Sales, or Marketing.

Data marts are frequently constructed and forbidden by a particular division within an organization. “Data marts regularly illustrate data from only a small amount of sources. The sources could be inner-outfitted systems, a central data warehouse, or external data.” (Paulraj & Syvaprakasam, 2012)

G.4 OLAP

As Stephens and Plew described in “Database Design”,

> An *Online Analytical Processing* (OLAP) database is one whose main purpose is to supply end-users with data in response to queries that are submitted. Typically, the only transactional activity that occurs in an OLAP database concerns bulk data loads. OLAP data is used to make intelligent business decisions based on summarized data, company performance data, and trends. The two main types of OLAP databases are Decision Support Systems (DSS) and Data Warehouses. Both types of databases are normally fed from one or more OLTP databases, and are used to make decisions about the operations of an organization. A data warehouse differs from a DSS in that it contains massive volumes of data collected from all parts of an organization. Data warehouses must be specially designed to accommodate the large amounts of data storage required and enable acceptable performance during data retrievals.

In the book “Data Modelling techniques for Data warehousing” their authors described,

> The techniques of data analysis can impact the type of data model selected and its content. For example, if the intent is simply to provide query and reporting capability, a data model that structures the data in more of a normalized fashion would probably provide the fastest and easiest access to the data. Executing this type of capability typically might lead to the use of more direct table scans. For this type of capability, perhaps an ER model with a normalized and/or denormalized data structure would be most appropriate.

So, it is clear that ER model is the best choice when “what” question wants to be answered. Following reading the previous paragraph we find the reason of the dimensional approach,

> If the objective were to perform multidimensional data analysis, a dimensional data model would be more appropriate. This type of analysis requires that the data model support a structure that enables fast and easy access to the data on the basis of any of numerous combinations of analysis dimensions.
>
> Multidimensional analysis requires a data model that will enable the data to easily and quickly be viewed from many possible perspectives, or dimensions since a

number of dimensions are being used, the model must provide a way for fast access to the data. In this case, a dimensional data model would be most appropriate. (Ballard, Herreman, Schau, Bell, Kim, & Valencic, 1998)

In the OLAP world, there are mainly two different of types: Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP). There is another type named HOLAP that combines both technologies. "The trade-off of MOLAP versus ROLAP is performance versus storage" (Burstein & Holsapple, 2008)

ROLAP

ROLAP is "a form of OLAP where the data is stored in a relational database. Thus, like any OLAP application, a true ROLAP application must support multidimensionality, drill-down, rotation, and multiple modes of view." (Abbey, 2008)

ROLAP supports large databases while enabling good performance, platform portability, exploitation of hardware advances such as parallel processing, robust security, multi-user concurrent access (including read-write with locking), recognized standards, and openness to multiple vendor's tools. ROLAP is based on familiar, proven, and already selected technologies.

ROLAP tools take advantage of parallel RDBMSs for those parts of the application processed using SQL (SQL not being a multidimensional access or processing language). SO, although it is always possible to store multidimensional data in a number of relation tables (the star schema), SQL does not, by itself, support multidimensional manipulation of calculations.

Some advantages that can be observed in this approach are:

1. Can handle large amounts of data: The data size limitation of ROLAP technology is the limitation on data size of the underlying relational database. In other words, ROLAP itself places no limitation on data amount.
2. Can leverage functionalities inherent in the relational database: Often, relational database already comes with a host of functionalities. ROLAP technologies, since they sit on top of the relational database, can therefore leverage these functionalities.

On the other hand this approach also has some disadvantages,

1. Performance can be slow: Because each ROLAP report is essentially a SQL query in the relational database, the query time can be long if the underlying data size is large.
2. Limited by SQL functionalities: Because ROLAP technology mainly relies on generating SQL statements to query the relational database, and SQL statements do not fit all needs. For example, it is difficult to perform complex calculations using SQL.

MOLAP

The MOLAP (multi-dimensional online analytical processing) engine takes the data from the data warehouse or from the operational sources. The complexity of underlying data is hidden from the MOLAP user tool. In other words, MOLAP "perform OLAP functions without having to understand how the cubes are formed and how they differ from relational tables." (Burstein & Holsapple, 2008)

As it is done with ROLAP, next lines describe advantages and disadvantages of this approach. First, advantages are listed:

1. Excellent performance: MOLAP cubes are built for fast data retrieval, and are optimal for slicing and dicing operations.
2. Can perform complex calculations: All calculations have been pre-generated when the cube is created. Hence, complex calculations are not only doable, but they return quickly.

In the other hand, the disadvantages are:

1. Limited in the amount of data it can handle: Because all calculations are performed when the cube is built, it is not possible to include a large amount of data in the cube itself.
2. Requires additional investment: Cube technologies are often proprietary and do not already exists in the organization. Therefore, to adopt MOLAP technology, chances are additional investments in human and capital resources are needed.

G.5 Decision Support System

Decision Support Systems (DSS) are a specific class of computer-based information systems that support your decision-making activities. A decision support system analyses business data and provide interactive information support to managers and business professionals during the decision-making process, from problem recognition to implementing your decision. Analytical models, specialized databases, decision makers' own insights and judgments, and an interactive, computer-based modelling process to support semi-structured business decisions.

As mentioned by Ralph Sprague and Eric Carlson in "Building Effective Decision Support Systems",

> The art of building decision support systems is the art of building computer-based solutions to undefined problems. The secret to building these solutions is to apply a multi-layered development approach whereby generic functionalities are embedded in the lower layers and can be purchased so that an end solution can be constructed and modified rapidly just by adding the upper layers.

It is interested to mention which are the main uses of a DSS system. As it is described by Sauter in "Decision support systems for Business Intelligence",

> Decision support systems are most useful when it is *not* obvious what information needs to be provided, what models need to be used or even what criteria are most appropriate. Said differently, they are useful when it is not obvious a priori how the choice should be made. Furthermore, since DSS proceed with requests from decision makers in the order and manner selected by the user (and not necessarily linear in their application), they tend to be associated with situations where users proceed differently with each problem. However, that does not mean a DSS cannot be useful for a more structured problem.

Generally, it is accepted that DSS technology is warranted if the goal is to help decision makers, generate better alternatives, respond to situations quickly and solve complex problems.

G.6 Data Mining

The objective of data mining is "to identify valid novel, potentially useful, and understandable correlations and patterns in existing data." (Chung & Gray, 1999)

Data mining is an extension of traditional data analysis and statistical approaches in that it incorporates analytical techniques drawn from a range of disciplines including, but not limited to numerical analysis, pattern matching and areas such as artificial intelligence as machine learning, neural networks and genetic algorithms.

The construction of a data warehouse, which involves data cleaning and data integration, can be viewed as an important pre-processing step for data mining. However, a data warehouse is not a requirement for data mining.

As Gray and Watson (1998) described,

> Building a large data warehouse that consolidates data from multiple sources, resolves data integrity problems, and loads the data into a database, can be an enormous task, sometimes taking years and costing millions of dollars

The question of how data warehousing and OLAP relate to data mining is a question that often arises. The relationship can be succinctly captured as Han and Kamber (2001) described: "The capability of OLAP to provide multiple and dynamic views of summarized data in a data warehouse sets a solid foundation for successful data mining." Therefore, data mining and OLAP can be seen as tools than can be used to complement one another. The term OLAP, standing for Online Analytical Processing, is often used to describe the various types of query driven analysis that are undertaken when analysing the data in a database or a data warehouse. OLAP provides for the selective extraction and viewing of data from different points of view; these views are generally referred to as dimensions (Fayyad, 2001).

The essential distinction between OLAP and data mining is that OLAP is a data summarization/aggregation tool, while data mining thrives on detail. Data mining "allows the automated discovery of implicit patterns and interesting knowledge that's hiding in large amounts of data" (Kamber & Han, 2001). And while OLAP is used to answer "why" certain things are true in that the user forms a hypothesis about a relationship and verifies it with a series of queries against the data.

H. Study of the technologies

During the last iteration of the Project some other technologies are studied in order to open new future lines and provide more technical reference to Naveen Sidda.

H.1 LucidDB

LucidDB is an open source database purpose-build to **power data warehouses, OLAP servers and BI systems**. It is based on architectural cornerstones such as **column-store**, bitmap indexing, hash join/aggregation, and page-level multiversioning.

Due to its purpose built architecture and features designed for BI and Data Warehouse is typically perform faster than traditional row store databases, without any traditional tuning.

A column-oriented DBMS stores data tables as sections of columns of data rather than as rows of data. A column-oriented database serializes all of the values of a column together, then the values of the next column, and so on. The Figure H.1 shows graphically row-oriented DBMS approach and column-oriented DBMS approach.

As the Wikipedia article describes, some of the benefits of using column store in organizations are:

1. **Column-oriented organizations are more efficient when an aggregate needs to be computed** over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data.
2. **Column-oriented organizations are more efficient when new values of a column are supplied for all rows at once**, because that column data can be written efficiently and replace old column data without touching any other columns for the rows.

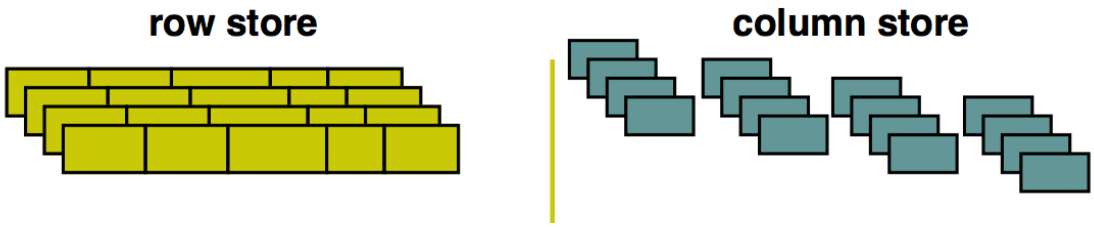

Figure H.1: Row store compared with column store. (Harizopoulos , Abadi , & Boncz, 2009)

H.2 Mondrian

Mondrian is an OLAP engine written in Java. It executes queries written in the MDX language, reading data from a relational database (RDBMS), and presents the results in a multidimensional format via a Java API, for example, Olap4j library.

OLAP (Online Analytical Processing) means analysing large quantities of data in real-time. Unlike Online Transaction Processing (OLTP), where typical operations read and modify individual and small numbers of records, OLAP deals with huge amounts of data, and operations are generally read-only.

The term 'online' implies that even though huge quantities of data are involved (typically many millions of records, occupying several gigabytes) the system must respond to queries fast enough to allow an interactive exploration of the data. As we shall see, that presents considerable technical challenges.

OLAP "employs a technique called Multidimensional Analysis. Whereas a relational database stores all data in the form of rows and columns, a multidimensional dataset consists of axes and cells." (Hyde, 2009)

As the "Mondrian Technical Guide" describes, a Mondrian OLAP system consists of four layers: the presentation layer, the dimensional layer, the star layer, and the storage layer.

The presentation layer: It determines what the end-user sees on his or her monitor, and how he or she can interact to ask new questions. There are many ways to present multidimensional datasets, including pivot tables (an interactive version of the table shown above), pie, line and bar charts, and advanced visualization tools such as clickable maps and dynamic graphics. These might be written in Swing or JavaScript, charts rendered in JPEG or GIF format, or transmitted to a remote application via XML. What all of these forms of presentation have in common is the multidimensional 'grammar' of dimensions, measures and cells in which the presentation layer asks the question is asked, and OLAP server returns the answer.

The dimensional layer: The dimensional layer parses, validates and executes MDX queries. A query is evaluated in multiple phases. The axes are computed first, then the values of the cells within the axes. For efficiency, the dimensional layer sends cell-requests to the aggregation layer in batches. A query transformer allows the application to

manipulate existing queries, rather than building an MDX statement from scratch for each request. And metadata describes the dimensional model, and how it maps onto the relational model.

Star layer: It is responsible for maintaining an aggregate cache. An aggregation is a set of measure values in memory, qualified by a set of dimension column values. The dimensional layer sends requests for sets of cells. If the requested cells are not in the cache, or derivable by rolling up an aggregation in the cache, the aggregation manager sends a request to the storage layer.

The storage layer: It is an RDBMS. It is responsible for providing aggregated cell data, and members from dimension tables.

H.3 Olap4j

Olap4j is an OLAP API. It is also a specification for database driver implementations. There are currently two known and actively supported implementations of the driver component of olap4j. The first driver implementation is a generic XMLA web service consumer. It is capable of connecting to many database engines on the market via SOAP-style web services. Mondrian OLAP engine provides the second implementation.

The figure 19 illustrates, on the left, the main components of the API. On the right, it is described with more details the hierarchical organization of an OLAP server's metadata. Olap4j replicates this hierarchy and uses it to represent all elements, which are available in the remote OLAP servers.

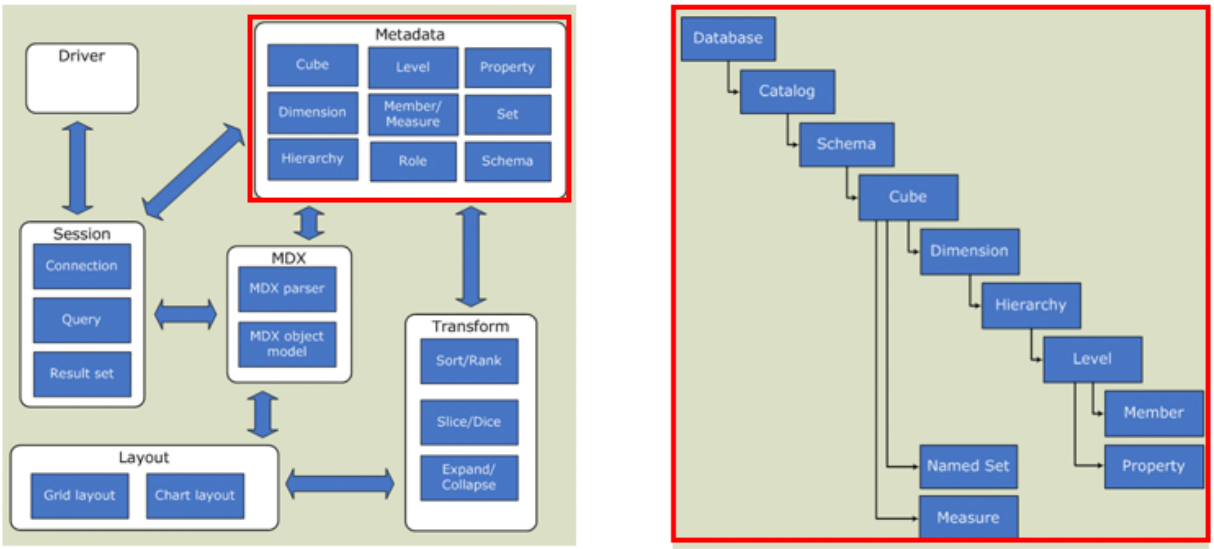

Figure H.2: On the left, Olap4j architecture. On the rigth the metadata with more details.

H.4 Geomondrian

GeoMondrian "is the first implementation of a true SOLAP server. It provides a consistent integration of spatial objects into the OLAP data cube structure, instead of fetching them from a separate spatial database, web service or GIS file. To make a simple analogy, GeoMondrian brings to the Mondrian OLAP server what Postgis brings to the PostgreSQL DBMS." It implements a native geometry data type and provides spatial extensions to the MDX query language, allowing embedding spatial analysis capabilities into analytical queries. It is important to remark that these extensions are possible thanks to the Mondrian extensible schema with new user-defined operations. These operations can be implemented in Java or JavaScript.

These geospatial extensions to the MDX query language "provide many more possibilities, such as: in-line geometry constructors, member filters based on topological predicates, spatial calculated members and measures and calculations based on scalar attributes derived from spatial features." (Badard, 2011)

At the moment of writing this report, GeoMondrian only supports Postgis based data warehouses.

H.5 MDX

Multidimensional Expressions (MDX) is to OLAP databases as SQL is to relational ones. It is an incredibly powerful query language that was created with multidimensional database operations in mind. MDX lets the user query multidimensional objects, such as cubes, and return multidimensional "cellsets" that contain the cube's data.

The Figure 22 describes the syntax of a MDX query. The main parts are the Column Axis, the Row axis and the Slicer. Thanks to its particular syntax, the measures can be retrieved with two or three dimensions.

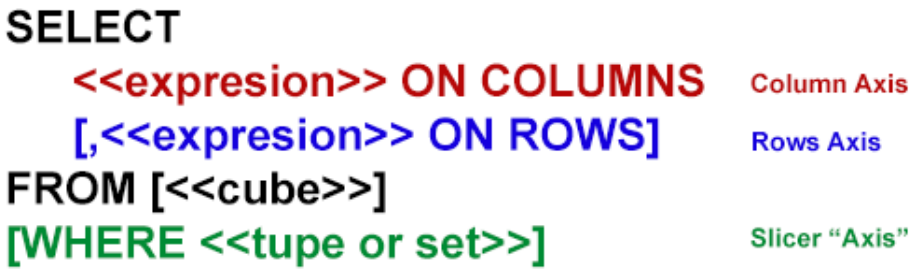

Figure H.3 MDX syntax.

H.6 XMLA

XML for Analysis specifies a SOAP-based XML communication API that supports the exchange of analytical data between clients and servers on any platform and with any language. It defines two generally accessible methods: Discover and Execute. Because XML allows for a loosely coupled client and server architecture, both methods handle incoming and outgoing information in XML format. Discover is used to obtain information and metadata from a Web Service. This information can include a list available data sources and data about the provider for a particular data source. Execute is used to execute MDX against a particular XML for Analysis data source. Moreover, Discover "allows analyst to specify generic interface and use of properties allows extensibility without rewriting existing functions." (Microsoft Corporation, 2001).

I. Enegy2People

El fin de semana del 20 de Abril asistí junto a mi amigo y compañero de carrera Alberto Alcolea al concurso "Space Apps Challenge" de Madrid.

Mi idea era seguir desarrollando mi PFC y ayudar a otras personas con sus trabajos. Llevé un prototipo de la aplicación Energy2People que se puede observar en la figura I.1.

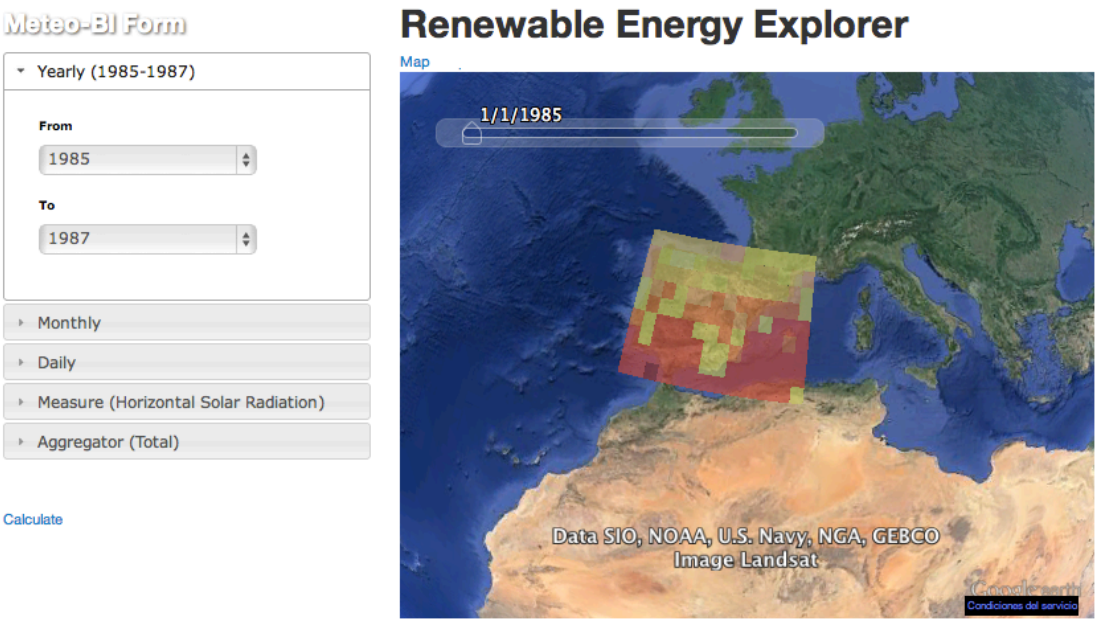

Figura I.1: Primer prototipo de Energy2People. Ya era capaz de mostrar datos de radiación solar y velocidad del viento utilizando técnicas de Business Intelligence .

Una vez allí, el facilitador de la NASA propuso que cada uno contáramos nuestras ideas para formar equipos y empezar a trabajar sobre ellas. Aprovechando esta oportunidad yo conté la idea de unir técnicas de toma de decisiones del "mundo empresarial 2.0", energías renovables y usuarios no expertos. El nombre, Energy2People.

El proyecto Energy2People, reunió a 9 personas, que abarcaban desde profesionales del mundo de la aeronáutica y las telecomunicaciones hasta estudiantes de primeros años de Ingeniería Informática. La figura I.2 muestra al equipo casi al completo.

Figura I.2 Equipo de Energ2People. De izquierda a derecha, Enrique Pérez, David Portilla, Daniel Sánchez, Albert Zurita, Fernando Martín, Pau Contreras, Alberto Alcolea y Borja Espejo.

La tarea más complicada fue organizar y priorizar las distintas ideas que tenía en mente, junto a las nuevas que surgían, entre un equipo de personas con un conocimiento tan diferente.

Tras varias de horas de trabajo, la aplicación presentaba una nueva imagen mucho más amigable tal y como se puede observar en la figura I.3 Además, nuevas funcionalidades como la generación de gráficas y la aplicación de modelos convertían a la aplicación en una herramienta mucho más potente.

Figura I.3. El nuevo aspecto de Energy2People. La imagen de la izquierda muestra el análisis en mapa. La imagen del centro muestra la generación de gráficos. La imagen de la derecha muestra la tabla de resultados tras la aplicación de un modelo financiero sobre los datos.

Tras un trabajo ininterrumpido de más de 36 horas, se dio a conocer el nombre del proyecto ganador del concurso. Y ese nombre era Energy2People.

Índice de figuras

Índice de tablas

Glosario

API: *Application Programming Interface* es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Caché: Parte de la memoria de un proceso, utilizada para almacenar objetos que ya han sido utilizados y que pueden volver a ser requeridos en un futuro.

Data warehouse: Es una colección de datos orientada a un determinado ámbito, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

DFD: *Diagrama de flujos de datos* es una representación gráfica del flujo de datos a través de un sistema de información. Un diagrama de flujo de datos también se puede utilizar para la visualización de procesamiento de datos.

DSS: *Decision Support System* o Sistema de soporte a la decisión es un sistema informático utilizado para servir de apoyo el proceso de toma de decisiones. OLAP o minería de datos son algunas de las formas que puede tomar un DSS.

Esquema: El esquema de una base de datos describe la estructura de una Base de datos, en un lenguaje formal soportado por el Sistema administrador de Base de datos (SGBD). En una base de datos relacional, el esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla, así como las restricciones.

ETL: *Extraer, transformar y cargar es* el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

Framework: Define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

GeoJSON: Es un formato abierto para codificar diferentes datos de índole espacial junto a otros no espaciales usando JSON.

JDBC: *Java Database Connectivity* es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

JSON: *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

HTTP: *HyperText Transfer Protocolo* es un protocolo de nivel de aplicación usado para la transferencia de información entre sistemas, de forma clara y rápida a través de la web. Este protocolo se usa desde 1990.

KML: *Keyhole Markup Language* es un lenguaje de marcado basado en XML para representar datos geográficos dos y tres dimensiones. Mantenido por Google.

Minería de datos: es un campo de las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos. Utiliza los métodos de la inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos.

OLAP: *On-line Analytical Processing* es una solución utilizada en el campo del *Business Intelligence* cuyo objetivo es agilizar la consulta de grandes cantidades de datos. Para ello utiliza estructuras multidimensionales o cubos OLAP.

Plugin: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Ésta es ejecutada por la aplicación principal e interactúan por medio de la API.

REST: *Representational State Transfer* es un estilo de arquitectura para desarrollar servicios haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y como se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

Scrum: Es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

Servlet: Son objetos de Java que corren dentro y fuera del contexto de un contenedor de servlets y extienden su funcionalidad. El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

SGBD: Un *sistema de gestión de bases de datos* es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.

SIG: Un *sistema de información geográfica* es una integración organizada de hardware, software y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión geográfica.

SEM: *Search Engine Marketing* es una forma de marketing en Internet que busca promover los sitios web mediante el aumento de su visibilidad en el motor de búsqueda de páginas de resultados.

SEO: *Search Engine Optimization* es el proceso de mejorar la visibilidad de un sitio web en los buscadores.

SOAP: *Simple Object Access Protocol* es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

SQL: *Structured Query Language* es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer modificaciones en ella.

Tomcat: Es un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

UML: *Unified Modelling Language* Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un modelo del sistema, incluyendo diferentes aspectos conceptuales.

Web Scraping: Es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la Web.

XML: *Extensible Markup Language* es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.