UNIVERSITATJAUME I DE CASTELLÓN

MASTER UNIVERSITARIO EN SISTEMAS INTELIGENTES

DESARROLLO DE UN GEOPORTAL WEB PARA LA MONITORIZACIÓN DE CULTIVO DE ARROZ

MEMORIA DE PRÁCTICAS

Presentado por:

IGNACIO MIRALLES TENA

Dirigido por:

JOAQUÍN HUERTA GUIJARRO

CARLOS GRANELL CANUT

Castellón, Julio, 2015

ÍNDICE

Índice de Tablas.

Índice de Figuras.

ACRÓNIMOS

- **API**: Application Programming Interface.
- **CSS**: Cascade Style Sheets.
- **DOM**: Document Object Model.
- **ERMES**: an Earth obseRvation Model based ricE information Service.
- **ESA**: European Space Agency.
- **GEOSS**: Global Earth Observation System of Systems.
- **GEOTEC**: Geospatial Technologies Research Group.
- **HTML**: HyperText MArkup Language.
- **IDE**: Integrated Development Environment.
- **INIT**: Institute of New Imaging Technologies.
- **LAI**: Leaf Area Index.
- **NDVI**: Normalized Difference Vegetation Index.
- **OF**: Objetivos Formativos.
- **OG**: Objetivos del Geoportal.
- **REST**: Representational State Transfer.
- **TFM**: Trabajo Fin de Máster.

RESUMEN

En el ámbito del sector agrícola se desea mejorar la competitividad y controlar el impacto ambiental de las prácticas llevadas a cabo. Concretamente en lo referente a los cultivos de arroz se han presentado propuestas a nivel europeo que pretenden impulsar proyectos comprometidos con esta labor.

El presente trabajo se enmarca dentro de uno de esos proyectos, ERMES, en el que se tiene como meta desarrollar aplicaciones y tecnologías que faciliten ese objetivo. Dentro de este proyecto se desea generar una aplicación Web capaz de presentar y analizar información geoespacial proveniente de diferentes fuentes como satélites, recogida en el propio campo o modelos de predicción.

El Geoportal desarrollado aprovecha la información proporcionada por estas fuentes y la presenta de forma usable, permitiendo al usuario configurar el entorno y mostrar únicamente la información que le resulta relevante y útil para realizar los análisis requeridos.

El presente documento recoge el trabajo llevado a cabo por el alumno para desarrollar la aplicación, desde la recogida de requisitos hasta la puesta en marcha y mantenimiento de la aplicación.

PALABRAS CLAVE

Aplicación Web, Geoportal, ERMES, agricultura inteligente, desarrollo web, tecnologías geoespaciales.

1 INTRODUCCIÓN

El alumno ha realizado el trabajo fin de master dentro del INIT (Institute of New Imaging Technologies) [1], concretamente en el grupo GEOTEC (Geospatial Technologies Research Group) [2]. Dicho grupo forma parte del consorcio del proyecto europeo ERMES (an Earth obseRvation Model based ricE information Service) [3]. Este proyecto está financiado por la Unión Europea, concretamente por el Séptimo Programa Marco de la Unión Europea para acciones de investigación, desarrollo tecnológico y demostración (FP7/2007-2013).

El sector agrícola europeo, especialmente el que se refiere a los diferentes cultivos de arroz, tiene como objetivo mejorar su competitividad y minimizar el impacto ambiental de algunas de sus prácticas [4]. El proyecto ERMES contribuye a ese objetivo mediante la creación de métodos capaces de monitorizar el estado de los campos durante el proceso de cultivo, permitiendo reaccionar rápidamente a los posibles problemas.

ERMES es un proyecto singular, tratándose de uno de los proyectos pioneros en poner en valor los primeros datos generados por los satélites Sentinel de la ESA (European Space Agency), recientemente puesto en órbita para la monitorización de la Tierra [5]. Los datos en bruto de Sentinel son procesados para convertirse en productos útiles (NDVI, LAI, etc) para la monitorización de la agricultura y las prácticas sostenibles. El gran reto del proyecto es ofrecer servicios a nivel regional o local, donde las imágenes satélites son de ayuda, pero deben complementarse con datos in situ recogidos por los usuarios (agricultores) a nivel local (p.ej. parcela). Es ahí donde el trabajo fin de máster tiene lugar, desarrollando herramientas y aplicaciones para la integración y visualización de productos de datos a distinta escala espacial y temporal, y procedentes de distintas fuentes (satélites, estaciones meteorológicas, observaciones de los propios usuarios) con el fin de proporcionar información útil para la detección temprana de problemas en las cosechas. Este problema ha sido también identificado como uno de los retos de futuro en el contexto de GEOSS [6], donde la integración, interpretación y visualización de datos capturados por el usuario con los datos oficiales, de gran cobertura, procedentes de GEOSS está aún por resolver.

Dentro del proyecto existen diferentes campos. El paquete de trabajo donde se enmarca este TFM se encarga del desarrollo de una aplicación web de soporte a los usuarios para que estos sean capaces de explorar y monitorizar en todo momento datos relevantes pertenecientes a los campos de cultivo de arroz. Estos datos provienen principalmente del procesamiento de imágenes ráster de distintos sensores (satélites y sensores instalados en los propios campos), y de la ejecución de modelos de simulación de cosechas. El estudiante será responsable de

diseñar y desarrollar este Geoportal, que además será capaz de comunicarse con otras aplicaciones, módulos y fuentes de datos que están siendo generados por otros miembros del consorcio del proyecto ERMES.

A continuación se presentan los objetivos del proyecto.

Objetivos

El trabajo divide sus objetivos desde dos perspectivas. Por un lado se presentan los objetivos formativos del alumno, buscando mejorar sus competencias y ampliar sus conocimientos. Por el otro se presentan brevemente los objetivos del paquete de trabajo y de la aplicación realizada.

Objetivos Formativos (OF):

- OF01: Adquisición habilidades en la obtención de requisitos de un proyecto de gran envergadura con múltiples partes implicadas.
- OF02: Adquisición de conocimientos en tecnologías Web para el desarrollo de aplicaciones Geoespaciales; utilizando Sistemas de Información Geográfica.
- OF03: Análisis y comprensión de información derivada tanto de sensores como de satélites.
- OF04: Habilidades para extraer y presentar información estadística basada en los datos obtenidos de diferentes fuentes o sensores.
- OF05: Trabajo en grupo multidisciplinar (y multilingüe) y asistencia a reuniones para definir y perfilar requisitos del proyecto.
- OF06: Análisis de grandes cantidades de datos y desarrollo de habilidades para presentar estos datos de forma usable y útil para usuarios no expertos.

Objetivos del Geoportal (OG):

- OG01: Herramienta basada en Web.
- OG02: Apoyo a los productores y autoridades regionales en el control y supervisión de las prácticas de cultivo de arroz.
- OG03: Ofrecer información pertinente, útil y actualizada al sector agrícola.
- OG04: Visualizar información relevante para monitorizar los campos de arroz a escala regional.
- OG05: Incluir datos básicos sobre las áreas geográficas involucradas.

- OG06: Incluir la información obtenida de sensores y satélites sobre las áreas geográficas involucradas.
- OG07: Incluir previsiones sobre los cultivos basándose en un modelo generado por otros miembros del consorcio.
- OG08: Ofrecer toda la información anterior en un portal accesible, usable y cómodo para usuarios sin conocimientos adicionales sobre Sistemas de Información Geográfica.

En este marco se ha desarrollado el proyecto fin de máster. El apartado **dos** presenta la planificación del trabajo, incluyendo la metodología aplicada, el apartado **tres** describe los requisitos de los usuarios, presentando los casos de uso a los que debe dar soporte la herramienta final y el alcance del proyecto. En el apartado **cuatro** se enumeran y describen las restricciones a las que se ha visto sometido el proyecto. El apartado **cinco** se centra en el diseño de la aplicación, los *mockups* que se han ido creando y el diagrama de clases en el que se basa todo el código. El apartado **seis** incluye toda la fase de implementación, describiendo algunos aspectos del código, justificando las decisiones tomadas y explicando algunos problema a los que ha habido que encontrar solución. El apartado **siete** muestra los resultados obtenidos de la aplicación y describe la experiencia en sus primeros días de uso. Finalmente, el apartado **ocho** presenta las conclusiones del trabajo.

2 PLANIFICACIÓN

La planificación el trabajo fin de máster está vinculada con la metodología seguida, en ésta se combina el esquema clásico del desarrollo en cascada con el desarrollo ágil de software.

2.1 Metodología.

Algunas de las fases iniciales se han llevado a cabo de forma tradicional, por ejemplo el inicio del proyecto, la planificación o la recopilación de requisitos. Sin embargo, cuando el proyecto ha avanzado a las fases de análisis y desarrollo se ha aplicado una metodología ágil basada en *Code Sprints* con pequeños objetivos que deben cumplirse. Los tutores y el alumno proponen las nuevas tareas para que sean realizadas en un máximo de dos o tres semanas, dependiendo del contenido y complejidad de las mismas. De estos nuevos resultados se genera *feedback* y sobre ellos se itera hasta alcanzar el hito final.

En lo referente a la obtención de requisitos se ha aprovechado parte del trabajo realizado previamente en el proyecto donde están estipulados los diferentes objetivos de alto nivel. Basándose en ellos, el alumno ha entablado conversaciones y organizado reuniones grupales para extraer las necesidades concretas del proyecto.

La memoria se ha ido redactando y revisando durante todo el desarrollo del proyecto, actualizando aquellos aspectos que se ha ido modificando. De hecho, una de las tareas principales de cada *Code Sprint* es la documentación de las tareas realizadas, que implica por ejemplo la justificación de las decisiones tomadas y las alternativas descartadas. De esta forma, la documentación generada por los *Code Sprint* ha sido una parte fundamental de la memoria final del Proyecto.

Los tutores han ido revisando el trabajo y proporcionando *feedback* sobre el mismo a lo largo de todo el proceso, permitiendo de este modo adaptar la planificación cuando han surgido problemas o modificaciones.

Finalmente, se preparará la presentación y se realizarán ensayos frente a los tutores hasta que estos decidan que tanto el contenido como la presentación se ajustan a lo esperado.

2.2 Diagrama de Gantt.

La Tabla 1 resume las tareas planificadas, las horas que se les dedicará y los objetivos de cada una de ellas.

Tabla 1. Planificación inicial del trabajo fin de máster.

Tarea	Horas planificadas	Objetivos
Definición del alcance del proyecto y objetivos	10	Se define en qué va a consistir el proyecto, cuál va a ser el alcance y se realiza una planificación previa de las tareas a realizar.
Redacción de la propuesta	4	Redacción y revisión de la propuesta de proyecto.
Planificación	10	Análisis de las tareas necesarias y el tiempo previsto para llevarlas a cabo.
Definición de requisitos	50	Leer toda la documentación del proyecto. Reuniones con coordinadores, usuarios y con otros miembros del consorcio. Lectura de diversa documentación para establecer necesidades del Geoportal.
Análisis herramientas	20	Análisis de las diferentes herramientas disponibles para llevar a cabo el desarrollo. Se realizan diferentes pruebas con ellas para comprobar su rendimiento.
Análisis aplicación	30	Análisis de las necesidades de la aplicación, las clases, objetos y métodos. La información necesaria de los sensores y como representarla.
Diseño aplicación	30	Realización de *Mockups* y propuestas de diseño de interfaz para la aplicación final. Pruebas con usuarios, coordinadores y otros miembros del consorcio para definir la forma correcta de mostrar e interactuar con la información.
Desarrollo aplicación	50	La programación del código, diferentes clases, controladores, widgets, etc para conseguir los resultados definidos en fases anteriores.
Testeo previo	20	Pruebas con los servicios utilizados y retroalimentación tanto de los tutores como de los usuarios y miembros del consorcio. Corrección de errores.

Publicación herramienta	2	Trasladar la herramienta, validada por los tutores, a un servidor real y puesta en marcha.
Testeo herramienta publicada	10	Nuevo testeo de la herramienta para comprobar que su funcionamiento se mantiene en el entorno final. Comprobación de tiempos de espera y corrección de errores.
Redacción de la memoria	50	Se redacta la memoria definitiva incluyendo toda la información del trabajo realizado.
Preparar presentación	12	Se prepara una presentación multimedia para mostrar los resultados del trabajo.
Realizar presentación	2	Presentar el trabajo final al tribunal.
HORAS TOTALES	300	

La Figura 1 presenta el diagrama de Gantt generado con respecto a la planificación anterior. En dicho diagrama no se incluyen las dos últimas tareas (las relacionadas con la presentación) dado que en el momento de redacción y entrega de este trabajo todavía no han sido fijadas sus fechas.

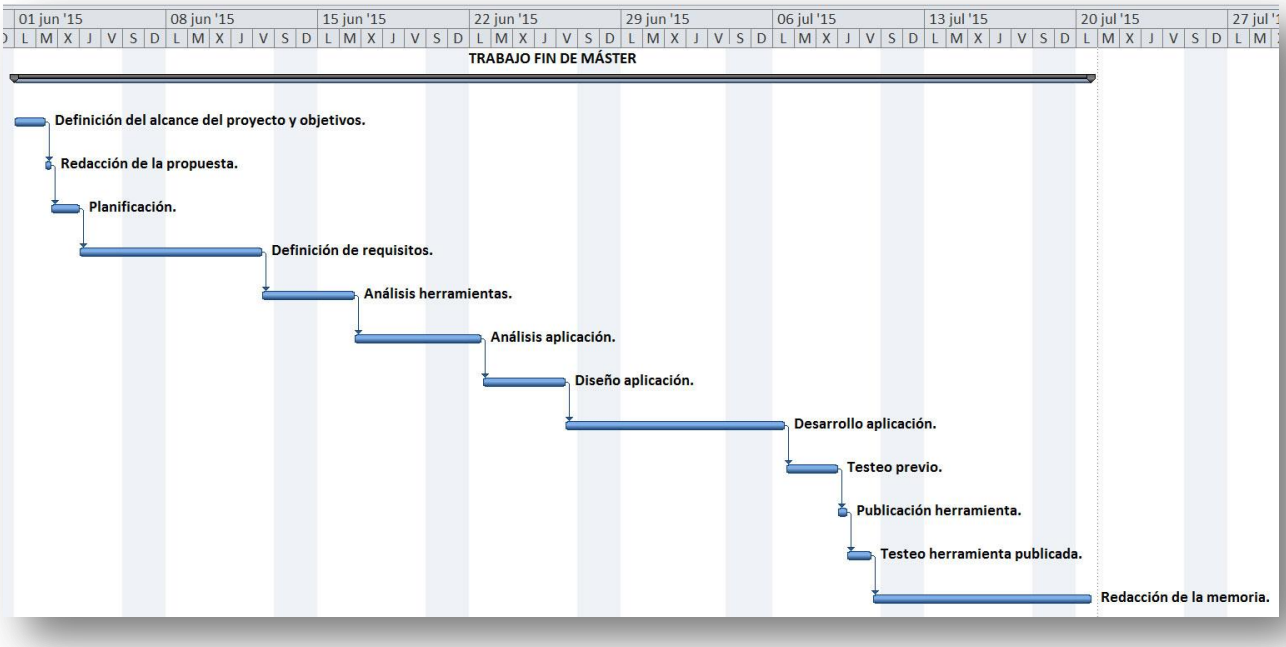

Figura 1. Diagrama de Gantt de la planificación del TFM.

3 REQUISITOS

Este apartado se encuentra dividido en tres partes diferentes, en primer lugar se presentan los requisitos intrínsecos del proyecto, incluyendo el perfil del usuario final y los casos de uso a los que debe dar soporte la aplicación. A continuación se especifican los requisitos tecnológicos necesarios. Finalmente se describe el alcance del trabajo fin de master en relación al proyecto.

3.1 Usuario y Casos de Uso.

El principal usuario del Geoportal es un experto vinculado a autoridades regionales de la agricultura de la zona. Debe estar registrado en la aplicación y, una vez identificado, debe ser capaz de recibir, visualizar y analizar la información relacionada con varios productos de ERMES. Los productos de ERMES consisten en información obtenida de diferentes fuentes (satélite, inserción manual, proporcionada por un modelo predictivo,…) que está asociada a un punto concreto en un mapa de la región indicada.

Actualmente el proyecto proporciona soporte a tres regiones:

- Zona de campos de arroz en Grecia.
- La albufera de Valencia en España.
- El norte de Italia.

Algunos ejemplos de productos son:

- Índice de Vegetación de diferencia normalizada (**NDVI**): Estima la cantidad, calidad y desarrollo de la vegetación en un lugar dado.
- Índice de Área Foliar (**LAI**): Área de la superficie de las hojas dividido por área de la tierra que cubre.
- Mapa de **Arroz**: Define las zonas donde hay arroz o no.

De este modo el usuario debe ser capaz de registrarse en la aplicación, acceder a ella (directamente se accede a la zona para la que ha sido registrado: España, Italia o Grecia) y, una vez en ella debe ser capaz de visualizar los productos deseados, en la fecha deseada. Debe ser capaz de realizar algunas operaciones de análisis como visualizar gráficos o comparar diferentes productos o fechas, y debe ser capaz de comparar la información de los productos con información como la frontera de las provincias.

Tanto los productos como las zonas visibles deben presentarse de forma dinámica, dado que se actualizan con frecuencia. Además se pide una interfaz sencilla que dé prioridad al mapa y a los servicios presentados.

Los casos de uso de forma concreta se presentan a continuación:

- Gestión de usuarios. (Tabla 2)
- Monitorizar producto. (Tabla 3)

Analizar producto. (

- Tabla 4)
- Comparar productos. (Tabla 5)
- Configurar información del mapa. (Tabla 6)

Tabla 2. Caso de uso 01

Gestión de usuarios.
Identificador: CU_ER_01 **Nombre**: Gestión de usuarios.
Descripción: Acceder a la plataforma Web y ser capaz de registrarse y de identificarse correctamente.
Secuencia: El usuario accede a través de un navegador al servidor donde se aloja la aplicación. Desde allí tiene acceso a registrarse, introduciendo algunos datos identificativos; y de identificarse, utilizando los datos identificativos previos.

Tabla 3. Caso de uso 02.

Monitorizar producto.
Identificador: CU_ER_02 **Nombre**: Monitorizar producto.
Descripción: El usuario debe ser capaz de visualizar el producto deseado sobre el mapa.
Secuencia: El usuario selecciona un producto y una fecha y este aparece sobre el mapa. El usuario puede acercarse o alejarse, haciendo zoom in o zoom out sobre las zonas deseadas y desplazándose sobre el mapa.

Tabla 4. Caso de uso 03.

Analizar producto.
Identificador: CU_ER_03 **Nombre**: Analizar producto.
Descripción: El usuario debe ser capaz de analizar líneas temporales del producto en una zona concreta.
Secuencia: El usuario debe tener claro el producto y la fecha que desea analizar. Además haber decidido sobre qué zona del mapa desea hacerlo. En ese momento el usuario debe ser capaz de obtener información concreta (numérica, si corresponde) sobre la zona deseada, esta información, además, debe ofrecer alguna opción de análisis (por ejemplo una línea temporal con la que comparar).

Tabla 5. Caso de uso 04.

Comparar productos.
Identificador: CU_ER_04 **Nombre**: Comparar productos.
Descripción: El usuario debe ser capaz de comparar el estado de dos productos o de un mismo producto en dos fechas diferentes.
Secuencia: El usuario selecciona un producto y una fecha, a continuación decide comparar y selecciona un segundo producto y una segunda fecha, ambos productos deben aparecer por pantalla ofreciendo una interfaz que permita su comparación.

Tabla 6. Caso de uso 05.

Configurar información del mapa.
Identificador: CU_ER_05 **Nombre**: Configurar información del mapa.
Descripción: El usuario debe tener algunas opciones disponibles para configurar el mapa (por ejemplo seleccionar un mapa base diferente, o añadir fronteras de provincias).
Secuencia: Accediendo a un menú de configuración el usuario puede seleccionar qué mapa base desea utilizar o que capas operacionales desea mostrar, por ejemplo una rejilla dividiendo parcelas espacialmente o un mapa de fronteras entre provincias.

3.2 Requisitos tecnológicos.

Toda la funcionalidad descrita en los casos de uso debe ser accesible desde diferentes sistemas y sin la necesidad de instalar software adicional. Esto es debido a que en muchos casos los usuarios trabajan en entornos donde la gestión y configuración de los equipos informáticos escapa a su alcance y la instalación de software supone un problema.

Por este motivo, el desarrollo se entiende desde un principio como una aplicación Web que sea capaz de conectarse con los proveedores de los productos y mostrarlos de forma sencilla y usable.

También es importante considerar que los productos y servicios ofrecidos desde el proyecto ERMES están ofrecidos a través de un servidor basado en tecnología ArcGIS Server [7]. Esto condiciona el uso de algunas de las librerías que facilitan la integración de estos servicios con el estándar HTML .

Tras analizar las principales formas capaces de dar soporte tanto a los aspectos relacionados con el portal Web, como con aquellas funcionalidades que requieren del uso de mapas, las tecnologías seleccionadas han ido las siguientes:

- **HTML5, CSS3 y JAVASCRIPT** ([8][9][10]): La combinación de estas tres tecnologías y estándares web forman actualmente el núcleo duro del desarrollo de aplicaciones Web. Adicionalmente la enorme cantidad de documentación existente en Internet facilita mucho la solución de los posibles problemas que han ido surgiendo.
- **API de ArcGIS para JavaScript** [11]: Consiste en un paquete de librerías y estilos pensados para el desarrollo y trabajo con mapas y servicios basados en ArcGIS Server. Se integra perfectamente con HTML5 dado que, además, está basado en CSS3 y JavaScript.
- **Bootstrap** [12]: Es un framework que permite desarrollar aplicaciones Web *responsive* y se integra perfectamente con el resto de tecnologías utilizadas.
- **jQuery** [13]: Es una librería de JavaScript que facilita mucho la interacción con el DOM de HTML para gestionar eventos, manipular los documentos, etc. Facilita mucho el trabajo asíncrono.
- **Dojo Toolkit** [14]: Se trata de un conjunto de librerías de JavaScript que facilita la creación de elementos dinámicos y la interacción con los mapas. Además proporciona componentes específicos que permiten reducir el tamaño de la aplicación escalándose para utilizar únicamente las librerías que necesita.

Adicionalmente a estas tecnologías utilizadas para el desarrollo del Geoportal se ha considerado el trabajo colaborativo y la seguridad. Para ello se ha trabajado, tanto con los tutores del trabajo como con los otros miembros que colaboran con otras partes del proyecto con la herramienta Git [15], concretamente con el repositorio GitHub [16] en su versión pública. Siendo la URL del proyecto:

https://github.com/ermes-fp7space

3.3 Alcance

Como se comenta en la introducción del presente documento, el desarrollo se engloba en el contexto de un proyecto mayor donde entran en contacto diferentes colaboradores, cada uno de los cuales desarrolla sus propios paquetes de trabajo. El equipo de la UJI, como miembro tecnológico, se encarga de la parte técnica, de integrar los datos provenientes de diferentes fuentes, procesarlos y ofrecerlos de forma usable. Dentro de estos trabajos se incluyen:

- Recopilar y organizar los datos del satélite.
- SmartAPP: Una aplicación móvil para recolectar datos in-situ.
- Recolectar datos del modelo estadístico capaces de ofrecer predicciones.
- Integración de todos los datos en una única plataforma que los ofrezca de forma sencilla mediante una interfaz REST (ArcGIS Server).
- Desarrollo de un servidor capaz de gestionar los datos de usuarios y controlar el acceso y perfiles por parte de los mismos.
- **Desarrollo del GEOPORTAL regional donde se ofrecen la mayor parte de los datos de forma usable.**
- Desarrollo de un GEOPORTAL local donde se proporcione acceso tanto a usuarios locales como regionales para monitorizar y gestionar los datos de las parcelas de los agricultores.

Puede observarse que la cantidad de desarrollos es amplia, el estudiante ha estado implicado y desarrollando gran parte de estas aplicaciones. Sin embargo, en el presente documento, únicamente se presenta la parte del GEOPORTAL Regional.

De este modo, dentro del alcance del mismo entra la gestión de usuarios, el acceso a los servicios, ofrecer herramientas de análisis y comparación de los mismos y desarrollar una interfaz usable para los usuarios.

Sin embargo quedan fuera del alcance de este trabajo la generación de los servicios, la integración de los mimos y la programación y mantenimiento del servidor de usuarios. Siendo estos parte de otros paquetes de trabajo.

4 RESTRICCIONES

En el desarrollo del GEOPORTAL se han presentado algunas restricciones que deben ser tenidas en cuenta.

El desarrollo de la aplicación se enmarca dentro de un proyecto de mayor envergadura en el que las fechas de entrega y la consecución de los hitos están predefinidas. Por ese motivo, los tiempos de entrega de la aplicación deben restringirse a ese mismo calendario (ya concordante con la planificación presentada): Esto presenta una **restricción temporal** que de no existir habría permitido abordar más funcionalidades, más pruebas y más iteraciones en la consecución de los requisitos.

Otra restricción impuesta por el propio proyecto son las **tecnologías** utilizadas. En el caso de la integración de datos, estos deben ser alojados y servidos a través de una plataforma ArcGIS Server, lo cual obliga a utilizar tecnologías Web compatibles con él (*ArcGIS API for JavaScript*). Del mismo modo la aplicación debe ser funcional en PCs/MAC de escritorio comunes sin necesidad de instalar nuevo software, lo que limita las posibilidades a la hora de plantear un desarrollo hecho en .NET; JAVA, etc.

Con respecto a las restricciones derivadas de los **costes** de desarrollo hay que decir que en la solicitud inicial del proyecto ERMES ya se han contemplado y evaluado, por lo que quedan asumidas desde una perspectiva diferente. Todas las tecnologías utilizadas han sido de uso libre (HTML, CSS, JAVASCRIPT, JQUERY, DOJO, API ARGIS, BOOTSTRAP,…). El equipo y puesto de trabajo donde se ha desarrollado la aplicación han sido asumidos al completo por el grupo GEOTEC, así como el tiempo invertido por parte tanto del estudiante como de los tutores.

5 DISEÑO

El presente apartado recoge el diseño del portal desde varias perspectivas. Por un lado se presentan los prototipos realizados sobre el flujo de trabajo, la interfaz gráfica y la mejor forma de presentar la información a los usuarios finales. Por otro lado se presenta el diseño desde el punto de vista de la arquitectura de la aplicación y el flujo de información. Finalmente se presenta el modelo de datos utilizado y el diagrama de clases establecido.

5.1 Interfaz Gráfica.

La interfaz de usuario es crítica para el éxito de la aplicación y durante todo el desarrollo del Geoportal ha sido un término en constante discusión.

El desarrollo de portales con información geoespacial presenta características especiales con respecto a otro tipo de aplicaciones. Se desea presentar la mayor cantidad de mapa posible, contextualizando las zonas donde le resulta relevante al usuario la información, pero a la vez es necesario que sea capaz de proporcionar información concreta y centrada en los objetivos que se desean alcanzar.

Además, se deben presentar herramientas de interacción con el mapa, adicionalmente a la escala, la posibilidad de acercar o alejar el punto de vista (Zoom In, Zoom Out), la posibilidad de moverse dentro del mapa,… También son necesarias las herramientas que ofrezcan funcionalidad para cumplir con los casos de uso de identificarse, registrarse, monitorizar, analizar, comparar y configurar.

La funcionalidad de identificación y registro es tan común y utilizada en tantas aplicaciones que no se van a presentar los *mockups* realizados sobre ella, directamente se visualizará, en la sección de resultados, el aspecto final de ambos formularios.

Con respecto a la **monitorización**, la primera propuesta realizada incluía una imagen con una serie de botones en la parte superior izquierda de la imagen (Figura 2). Al hacer clic sobre ellos debía desplegarse una ventana modal en la que el usuario configuraba las opciones necesarias, seleccionaba los productos que deseaba visualizar, etc. La Figura 3 muestra un ejemplo que representa el formato del resto de las imágenes.

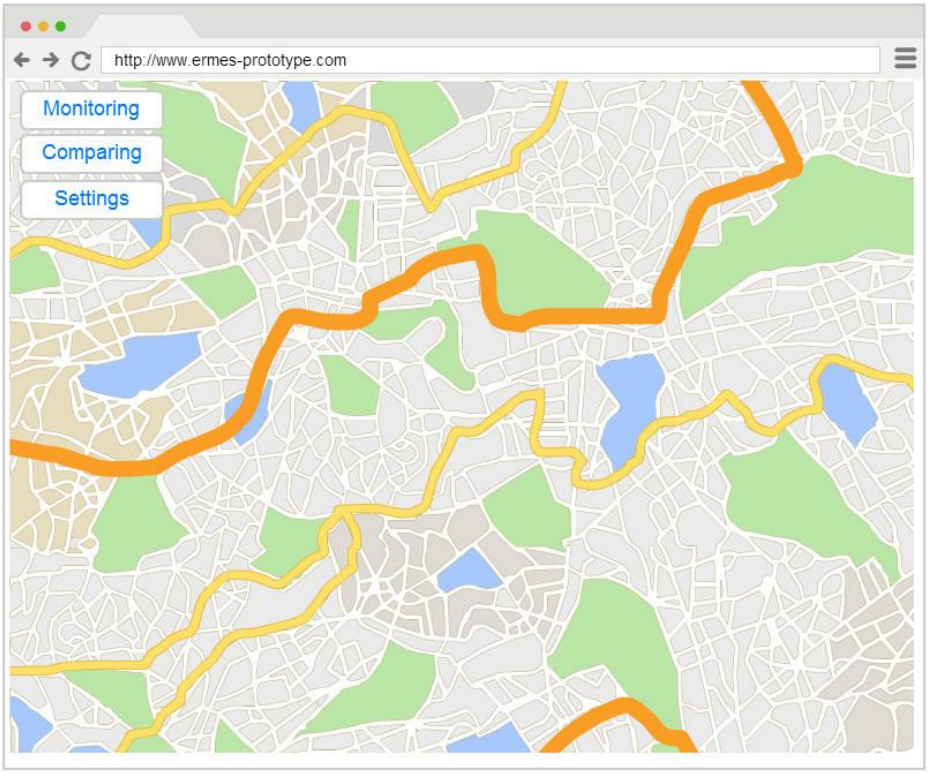

Figura 2. Primer *mockup* de la ventana inicial.

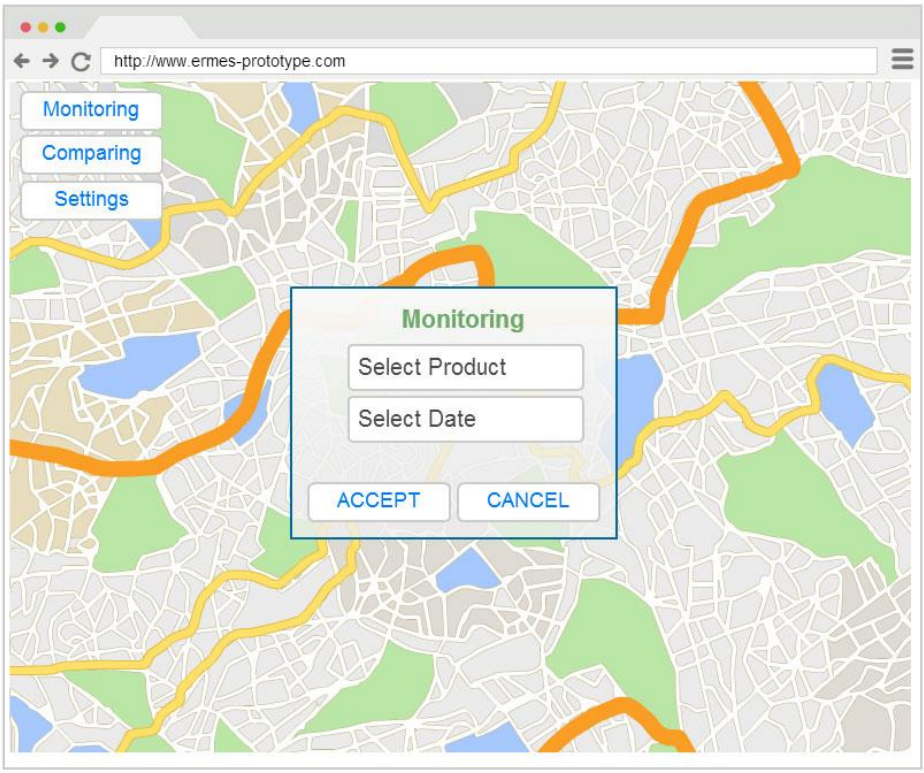

Figura 3. Primer *mockup* de la ventana de monitorización.

Sin embargo, los *mockups* fueron puestos en común con el resto de miembros del equipo, tanto con los tutores como con los participantes en el proyecto. Las conclusiones extraídas del feedback proporcionado llevaron a una segunda propuesta que se caracterizaba por necesitar un número inferior de clics para conseguir las cosas.

Por ejemplo, para mostrar un producto, con la propuesta inicial el usuario debía, una vez dentro de la aplicación, hacer 4 clics:

1. Clic sobre “monitorizar”.
2. Clic sobre el producto deseado.
3. Clic sobre la fecha.
4. Clic sobre ACEPTAR.

Además, si deseaba cambiar la fecha, debía repetir todo el proceso. En la segunda propuesta (Figura 4) se redujo el número de clics para estas acciones.

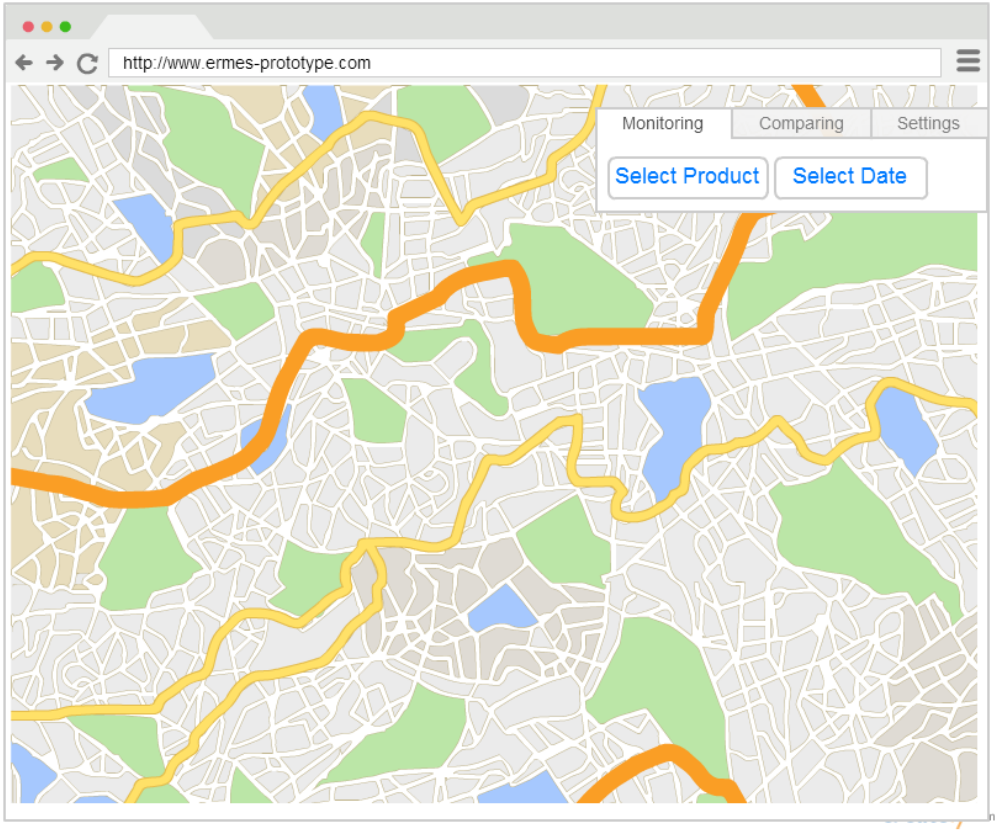

Figura 4. Segunda propuesta de interfaz gráfica.

La operación más común se trataba de la monitorización, por lo que se convirtió en la operación por defecto. De este modo para visualizar el primer producto únicamente era necesario hacer 2 clics:

1. Seleccionar producto.
2. Seleccionar fecha.

Además, si se deseaba cambiar la fecha bastaba con seleccionar una nueva fecha, y automáticamente se visualizaría el producto.

Para analizar se deseaba mostrar una gráfica que representara una sucesión de datos en una línea temporal en el punto deseado. Para ello se pensó que una interfaz en la que el usuario seleccionara un producto, hiciera clic sobre un punto y se mostrara la gráfica correspondiente. En un principio se consideró la opción de crear una nueva opción de menú para ello, pero finalmente se convirtió en parte de la misma imagen de monitorización.

La Figura 5 muestra el mockup presentado.

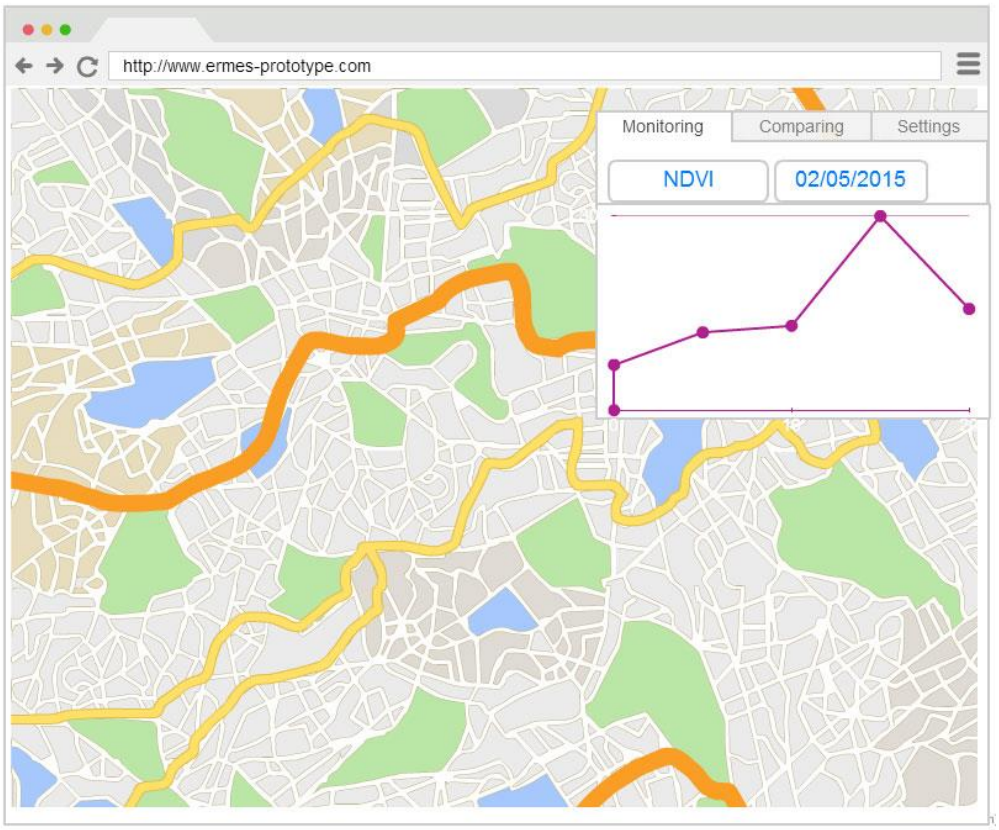

Figura 5. Mockup de la ventana que incluye la herramienta de análisis.

La herramienta de comparación debía permitir seleccionar dos productos y mostrarlos a la vez. Se plantearon varias propuestas. La primera de ellas incluía presentar dos imágenes, una para cada producto, pero quedó descartada porque había que controlar el zoom en ambas imágenes y la posición, lo cual restaba usabilidad a la herramienta.

La segunda propuesta fue crear un efecto de desvanecimiento con un slider, de modo que si el slider se encontraba completamente en un lado se mostraba uno de los productos, si se encontraba en el otro lado se mostraba en otro y en los puntos intermedios se interpolaba paulatinamente entre ambos productos. Esta opción fue testeada con miembros del consorcio para decidir si podía resultar útiles, pero finalmente se consideró que no mostraba información real dado que, al interpolar colores en ciertos productos, podía dar una sensación equívoca.

La tercera propuesta, la finalmente implementada, fue un efecto de *Swipe*, una barra en medio de la pantalla que mostraba a la izquierda uno de los productos y a la derecha el otro. Esa barra podía desplazarse y ampliar o reducir la zona de cada uno de los productos, como se muestra en la Figura 6.

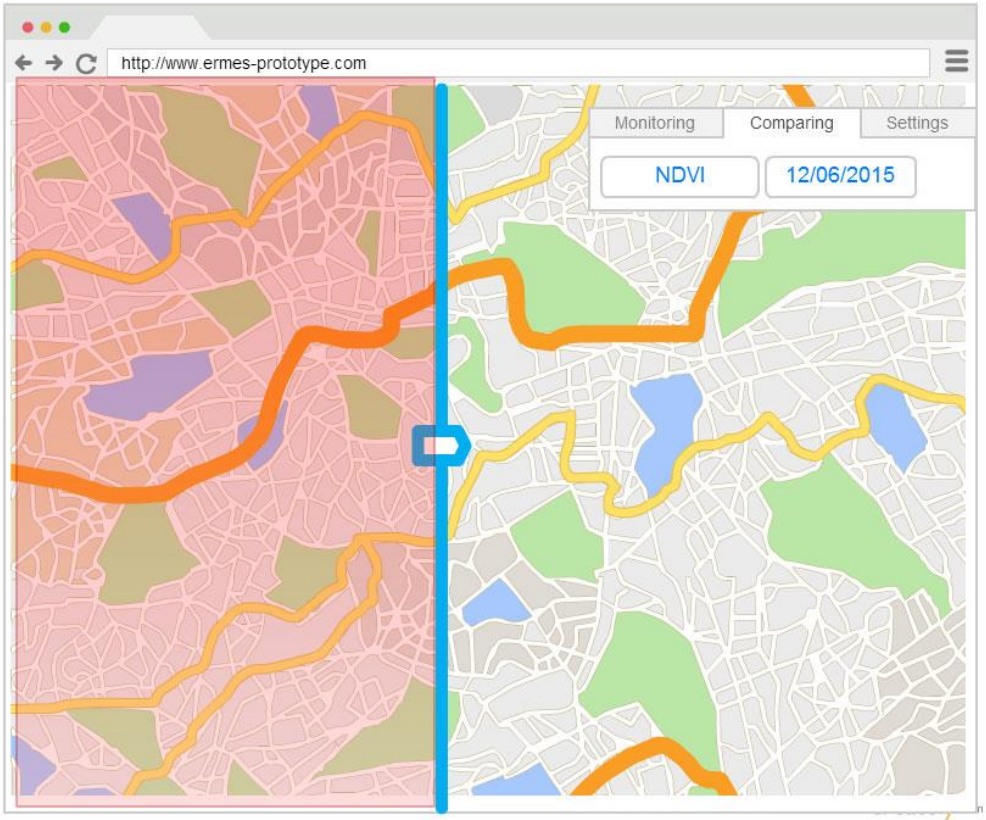

Figura 6. Mockup de la ventana que incluye la herramienta de análisis.

De esta forma lo que faltaba era permitir seleccionar los dos productos. La decisión fue de mantener en el lado derecho el producto seleccionado en la ventana de monitorización y ofrecer uno del mismo modo en la sección de comparación que se mostraría en el lado izquierdo.

Finalmente, la sección de Configuración únicamente permitía algunas funcionalidades como seleccionar o eliminar del mapa algunas capas básicas (como las fronteras entre las provincias) o la opción de imprimir, por lo que se trataba de un sencillo formulario que se mostrará directamente en su versión final en el apartado de resultados.

5.2 Arquitectura

Con respecto a la arquitectura de la aplicación, sin entrar en el diagrama de clases y modelo de datos (en el siguiente apartado) hay que tener en cuenta que los datos se generan desde diferentes fuentes y terminan integrándose en el servidor de *ArcGIS Server*; además, la gestión de usuarios se controla desde un servidor *NodeJS* [17]. La comunicación entre el cliente Web y estos servicios se hace mediante interfaces REST que proveen todo lo necesario para conectarse, actualizar y descargar la información pertinente.

De esta forma en la Figura 7 se muestra el proceso seguido. El usuario, mediante un ordenador se conecta a la página Web donde se aloja la aplicación. En primer lugar, para la identificación o registro el portal utiliza los servicios proporcionados por el servidor NodeJS, éste a su vez se conecta con una base de datos NoSQL, MongoDB [18] concretamente, que realiza las operaciones de almacenamiento o registro necesarias.

Una vez el usuario se encuentra registrado, el usuario interactúa con la aplicación, la cual a su vez se conecta con el servidor ArcGIS Server para recuperar los mapas o la información que se incluye en ellos.

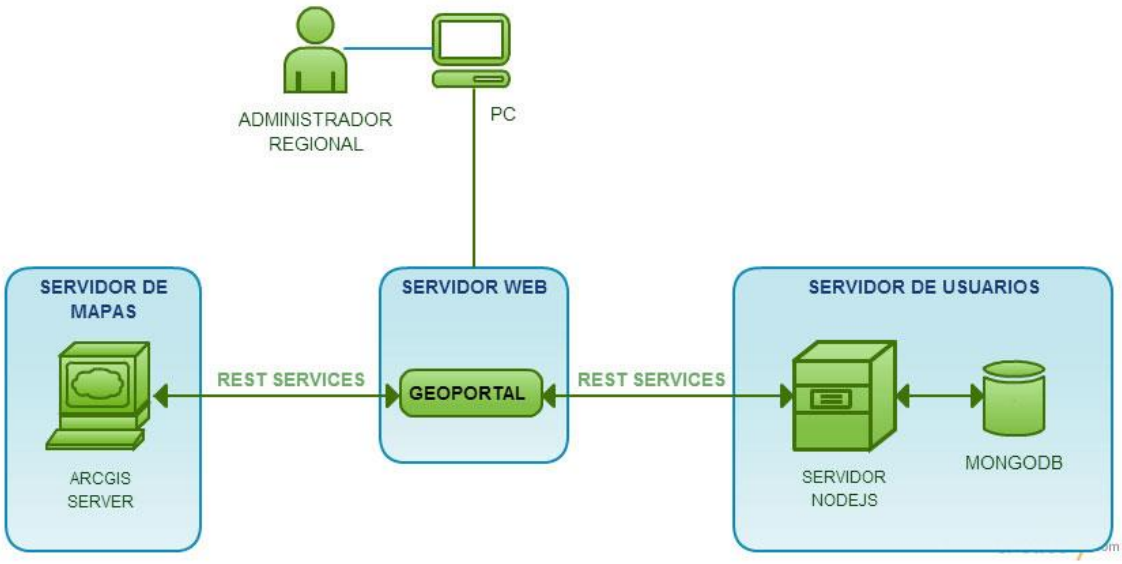

Figura 7. Arquitectura del Geoportal.

Mediante esta arquitectura los usuarios y los servicios de mapas se mantienen separados, lo cual facilita que otras aplicaciones del proyecto puedan utilizar la información necesaria sin

interferir sobre ella. Además, dado que el modelo de usuarios es independiente de la información proporcionada por los servicios de mapas, si éste cambia las aplicaciones no se interfieren entre sí.

5.3 Datos.

La fase de análisis de datos para el desarrollo de la aplicación pasó por varias fases durante el desarrollo. En las diferentes iteraciones se fueron generando diferentes modelos que, por recomendación de los tutores o por descubrir en el proceso que se requerían cambios, fueron evolucionando. A continuación se describe la estructura que tienen los datos que maneja la aplicación.

5.3.1 Usuarios.

Como se ha comentado anteriormente los usuarios son compartidos entre diferentes aplicaciones, para cada una de ellas son necesarias unos datos diferentes. En este caso únicamente se presentan los datos relevantes para el Geoportal:

- Usuario.
- Contraseña.
- Correo electrónico.
- Región: España, Italia o Grecia.
- Perfil. En el momento de desarrollar el portal únicamente se contemplaban dos posibles perfiles: Local y Regional. Aunque se contempló la necesidad de añadir más en un futuro y se dejó el código preparado para ello.

Estos datos se introducen en el formulario de registro y, mediante los servicios Web proporcionados por el servidor NodeJS se almacenan en la base de datos.

A la hora de utilizarlos en la aplicación se almacenan como *cookies* el nombre de usuario, la región y el perfil, por lo que el navegador es capaz de acceder a ellos cuando los necesita sin necesidad de reconectar con la BBDD.

5.3.2 Productos.

Los productos de ERMES han ido variando mucho durante el uso de la aplicación (de hecho continúan haciéndolo). Por ejemplo LAI, NDVI o Mapa de Arroz son productos típicos, aunque otros generados del modelo (como el Riesgo de infección) o información climática también han sido incluidos y eliminados de la aplicación en varias ocasiones.

Por ese motivo tenía sentido utilizar alguna estructura genérica que fuera capaz de actualizarse dinámicamente. En el siguiente apartado se describe con más detalle qué solución se ha tomado, sin embargo procede comentar aquí la estructura de datos del Mosaico.

Un Mosaico está compuesto de un conjunto de imágenes ráster que a su vez contienen información numérica en cada uno de sus puntos. Las diferentes imágenes de un mosaico no tienen por qué coincidir en tamaño y posición, cada una puede representar una región diferente. Sin embargo, en el caso de los productos de ERMES todas las imágenes ráster de un mosaico poseen exactamente las mismas dimensiones y la misma referencia en el mapa.

Estas imágenes se corresponden con puntos concretos del mapa, en cada una de ellas hay un valor asociado. La simbología del Mosaico define qué color se selecciona para cada valor.

Por ejemplo, en el mosaico NDVI de 2015 se muestran imágenes que representan la cantidad de vegetación en cada punto del mapa. En este caso el mosaico se compone de aproximadamente 24 imágenes, una tomada cada semana por el satélite. Para los valores más altos se asigna el verde (más vegetación) y los más bajos tienen amarillo.

A continuación se presenta un sencillo ejemplo, con dos imágenes ráster de un mismo mosaico. La Figura 8 es del 18 de Enero de 2015, mientras que la Figura 9 es del 26 de Mayo del mismo año. Puede observarse la variación en la vegetación gracias a la simbología aplicada en cada uno de los puntos.

Figura 8. Ráster del 18 de Enero del 2015, perteneciente al mosaico de NDVI de 2015.

Figura 9. Ráster del 26 de Mayo del 2015, perteneciente al mosaico de NDVI de 2015.

De modo que para cada Mosaico se tiene una colección de rasters (identificada por fecha) y cada uno de ellos posee algo similar a una matriz de valores para cada uno de sus puntos. Esta información se proporciona a través de servicios del servidor ArcGIS Server, la que se almacena en la aplicación es la siguiente:

- ID de mosaico.
- Nombre del mosaico.
- Descripción del mosaico.
- URL del servicio que lo provee.
- Cantidad de imágenes Ráster.
- Un conjunto de datos de imágenes Ráster.

Y para cada una de las imágenes Ráster se almacena:

- ID de Ráster.
- Nombre del Ráster.
- Fecha de toma de la imagen.

En principio la aplicación no almacena ni la imagen ni los valores (la cantidad de información podría ser enorme cuando se trata de productos con imágenes diarias o incluso de varios mosaicos para cada producto), cuando el usuario la solicita se realiza una petición al servidor para que la proporcione. Lo mismo ocurre con los valores de cada uno de los puntos, cuando el usuario requiere el valor concreto de un punto (aunque pueda hacerse una idea por el color) se hace una solicitud al servidor y se obtiene esa información. Además, es posible hacer peticiones que devuelvan valores medios o cadenas de valores para un punto dado.

En definitiva, toda la información necesaria sobre los productos se almacena en forma de Mosaico, por lo que el controlador principal guarda un conjunto de Mosaicos, cada uno de ellos con la información necesaria para visualizar el contenido requerido.

5.4 Clases.

Con el objetivo de que el código de la aplicación sea fácil de mantener y actualizar, el desarrollo de la aplicación trata de seguir el patrón de diseño Modelo, Vista, Controlador (MVC). Para ello la tecnología DOJO ofrece cierta facilidad ya que permite declarar objetos que se pueden importar en otros ficheros mediante el uso de JavaScript.

Con respecto al modelo, dado que la mayoría de aspectos que deberían ir vinculados al mismo se obtienen y actualizan mediante servicios REST no es necesario crear clases u objetos que almacenen las imágenes o los usuarios. El principal componente del Modelo es el Mosaico, el cual contiene toda la información necesaria para cada uno de los productos.

Las vistas están creadas en HTML y presentadas en forma de plantillas (Templates). Además de las dos iniciales para Identificación (l*ogin.html*) y *Registro* (*register.html*) se tiene una principal (*index.html)*, sobre la que se añaden o cargan nuevas plantillas. Estás se enumeran a continuación:

- Menú Principal: *mainMenu.tpl.html*
- Menú de Monitorización: *monitoringMenu.tpl.html*
- Menú de Comparación: *comparingMenu.tpl.html*
- Menú de Configuración: *settingsMenu.tpl.html*

La herramienta consta de dos accesorios para análisis, las gráficas y la tecnología para comparar dinámicamente. A estos dos accesorios se les ha denominado Widgets y, además de sus controladores, se han utilizado vistas asociadas a ellos

- Widget de Monitorización o Análisis. *monitoringWidget.tpl.html*
- Widget de Comparación. *comparingWidget.tpl.html*

Las vistas se modifican y actualizan desde los controladores, estando completamente separadas del Modelo. También hay que tener en cuenta que al tratarse de lenguaje HTML se utiliza CSS para configurar el aspecto de los diferentes componentes. Todos los ficheros de estilos se encuentran almacenados en una carpeta denominada CSS que forma parte de la vista.

Antes de pasar a explicar los controladores, la Figura 10 presenta el diagrama de clases de la aplicación, donde pueden visualizarse tanto el modelo del Mosaico como las páginas y plantillas pertenecientes a las vistas y los controladores. Incluyendo sus atributos y métodos.

La nomenclatura utilizada para los atributos y métodos incluye el símbolo guion bajo (_) para aquellos métodos o funciones que únicamente son accesibles desde la propia clase.

La zona inicial del gráfico representa el camino a seguir para acceder a la aplicación. En primer lugar el usuario carga el index.html, el cual obliga al usuario a identificarse. En caso de no tener un usuario, le permite acceder a la pantalla de registro.

Una vez el usuario llama al index autentificado, se accede al primero de los controladores: *run.js.* Este controlador únicamente configura las rutas de los diferentes scripts que van a ser necesarios, facilitando su acceso a ellos y, seguidamente, da paso al siguiente controlador, *mainLauncher.js*. Este controlador únicamente ejecuta la aplicación, lanzando el controlador principal: *MainController.js.*

El controlador principal tiene tres responsabilidades:

1. Comprobar el perfil de usuario para gestionar sus permisos, para ello utiliza ficheros JSON de configuración almacenados en la carpeta CONFIG.
2. Comunicar el resto de controladores con el modelo de datos Mosaico, controlando un vector de Mosaicos, uno para cada producto.
3. Comunicarse con el mapa y las capas que se muestran y que no. Es el responsable de mostrar u ocultar los productos seleccionados.

Una vez identifica al usuario, configura los mosaicos, crea el mapa y lo carga, lanza el siguiente controlador, el responsable de los menús: *MenúsController.js.*

Este controlador se encarga de organizar el panel principal, donde se mostrarán todas las herramientas, es el responsable de habilitar o deshabilitar las funciones de cada uno. También genera y se comunica con los tres siguientes controladores, los responsables de cada uno de los menús:

- *MonitoringController.js*: Se tata del controlador que se carga por defecto en el menú. Es el responsable de las funciones relevantes a la monitorización. Solicita y muestra los mosaicos disponibles, los rásters y actualiza la información de pantalla (siempre en contacto con el controlador principal para acceder al mosaico y actualizar el mapa).

A su vez es el responsable de gestionar un nuevo controlador, el que se encarga del Widget de análisis. Cuando el usuario solicita una gráfica o desea eliminarla, crea o destruye un controlador de gráficas que se encarga de obtener los datos y presentarlos para su análisis. Este controlador se llama *MonitoringWidget.js*.

- *CoomparingController.js*: Este controlador es responsable de las funciones de comparación, permite elegir un segundo producto, su ráster y compararlos con los elegidos anteriormente. Para la comparación hace uso de un controlador propio, llamado *ComparingSwipeWidget.js*. Este controlador se encarga de mostrar la barra divisoria en pantalla y permitir que se visualicen u oculten las partes a su izquierda o derecha.
- *SettingsController.js*: El controlador de configuración se encarga de comunicarse con el principal para decidir si mantener o eliminar algunas capas operacionales, la posibilidad de imprimir el análisis realizado o seleccionar un mapa base diferente.

En el diagrama de la Figura 10 se muestra de forma gráfica todo lo explicado anteriormente. Mediante esta división de responsabilidades ha resultado muy cómodo efectuar los diferentes cambios que se han ido pidiendo a lo largo del proceso de desarrollo. Así mismo, de cara al mantenimiento del Geoportal facilita el cambio de tecnologías en el caso de que fuera necesario. Actualmente todo está creado y mantenido con la librería DOJO, pero si alguno de los Widgets o controladores requiriera una tecnología diferente, podría actualizarse únicamente la plantilla, o controlador necesario sin afectar al resto de la aplicación.

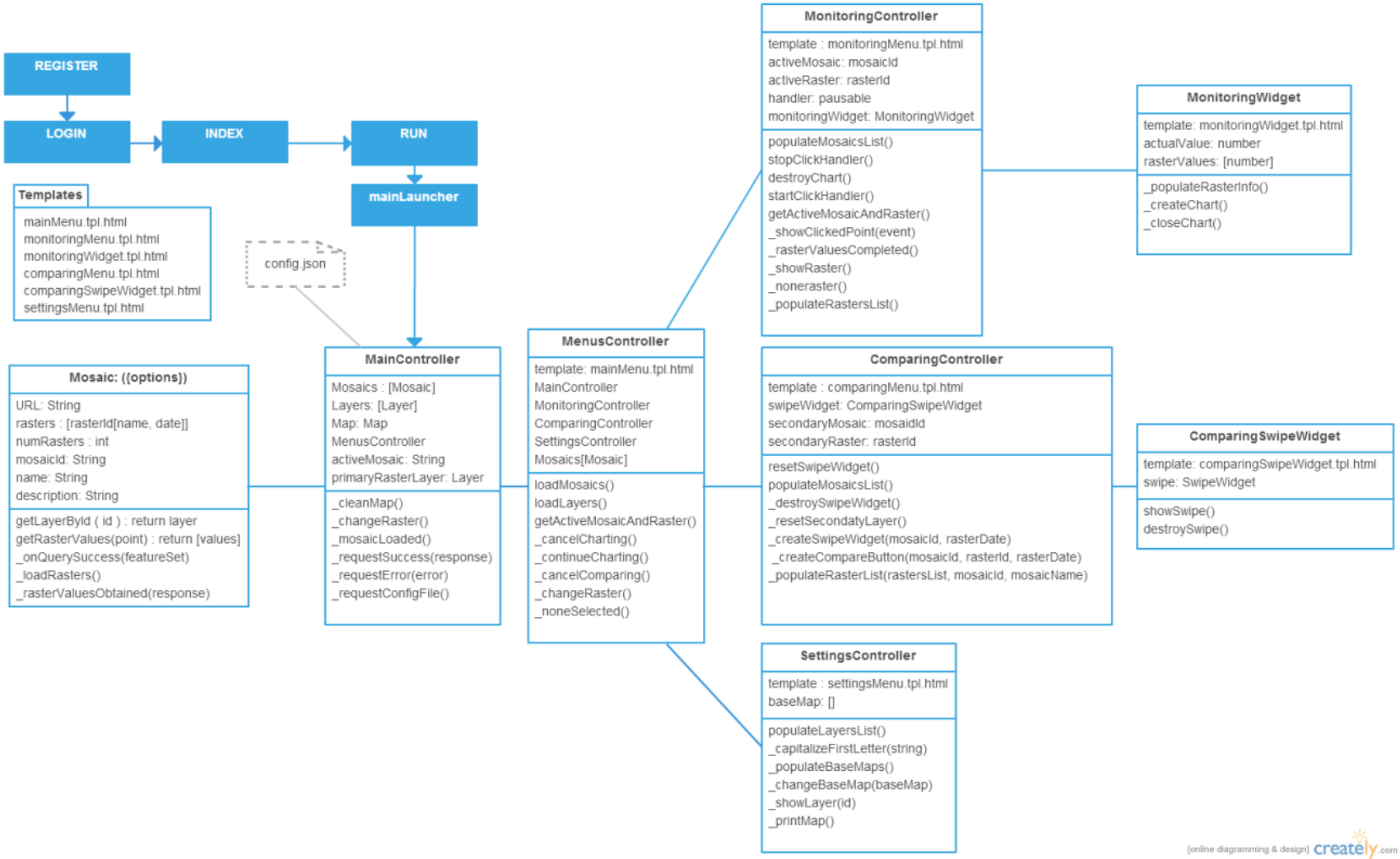

Figura 10. Diagrama de Clases del Geoportal.

6 IMPLEMENTACIÓN

Una vez queda definido el diseño de la aplicación, se desarrolla el código necesario para llevarlo a cabo. Dado que las clases están muy concretamente definidas y el flujo de la información entre el modelo, la vista y el controlador también, el trabajo ha consistido en ir desarrollando y comprobando paulatinamente los diferentes elementos.

La herramienta con la que se ha trabajado la mayor parte del tiempo ha sido el IDE WebStorm [19] en su versión 10.0.4. Este entorno de desarrollo requiere de licencias de pago, pero la Universitat Jaume I mantiene acuerdos con sus desarrolladores que permiten su uso por parte de los estudiantes.

Todo el código generado se encuentra disponible a través del repositorio público de github:

https://github.com/nachomiralles/RegionalGeoportalTFM.git

6.1 Estructura.

La estructura de ficheros se presenta a continuación:

En primer lugar aparecen los ficheros propios del repositorio y de la configuración del IDE, así como un README inicial que describe brevemente la funcionalidad de la aplicación. Estos son:

- **.idea/**
- **.gitattributes**
- **README.md**

La carpeta **doc** contiene información descriptiva relevante al desarrollo de la aplicación. En ella hay ficheros con notas, diagramas etc.

Finalmente en la carpeta **app** se encuentra el grueso de la aplicación. En ella se pueden localizar todos los ficheros descritos en el diagrama de clases implementando las funcionalidades necesarias. Además se encuentran los tres ficheros html iniciales de la aplicación: index, login y register.

- **config/**: Aquí se almacenan los ficheros json de configuración, los responsables de que el usuario reciba la aplicación personalizada con respecto a su región y perfil. En cada uno de ellos se describe la posición que debe utilizar y las capas de productos que debe cargar. Además puede especificarse cualquier detalle relativo a los propios productos.

- **css/:** Se incluyen las hojas de estilos necesarias. Algunas pertenecen a librerías externas como las de bootstrap, otras son propias, como *main.css*.
- **img/**: en esta carpeta se almacenan las imágenes relativas a los logos, fondos, etc. para la apariencia de la aplicación.
- **js/**: Esta es la carpeta principal con código. A su vez esta subdividida en varias, relativas al diagrama de clases descrito en el apartado anterior:
    - **controllers/**: Contiene los controladores básicos: el principal, el de monitorizar, el de comparar, el de configuración y el responsable de gestionar los menús.
    - **models**/: Almacena el modelo de la aplicación, en este caso únicamente el objeto Mosaic.js.
    - **templates/:** Se guardan las plantillas que se cargarán de forma asíncrona y dinámica según la aplicación las vaya necesitando. Todas ellas contienen código html básico.
    - **widgets**/: En esta se almacenan los dos controladores relativos a los widgets de comparar y monitorizar o analizar.

  Además esta carpeta contiene los ficheros run.js y mainLauncher.js, con los cuales se configuran las rutas y se lanza la carga inicial de los controladores.

6.2 Aspectos específicos.

A continuación se van a presentar algunos detalles específicos de la implementación. Se trata de mostrar únicamente los aspectos complejos o que han requerido más tiempo de desarrollo.

6.2.1 Sincronía y Ámbito.

Una de las principales problemáticas que presenta la aplicación es el funcionamiento asíncrono de muchas de sus funciones. Por ejemplo, cuando el usuario requiere un nuevo producto (una nueva imagen) el flujo de la aplicación no espera a recibirla para continuar con sus operaciones. El trabajo no es secuencial. Por ese motivo hay que considerar utilizar funciones de *callback* que se ejecutan cuando la tarea está finalizada. Un ejemplo de esto es lo siguiente:

```
getRasterValues: function(pointClicked){
    var mosaicRule = new MosaicRule();
    mosaicRule.ascending = true;
    mosaicRule.method = MosaicRule.METHOD_ATTRIBUTE;
    mosaicRule.sortField = "OBJECTID";

    var parameters = new ImageServiceIdentifyParameters();
    parameters.mosaicRule = mosaicRule;
    parameters.geometry = pointClicked;

    var identifyTask = new ImageServiceIdentifyTask(this.URL);

    if(this.historicURL)
        identifyTask.execute(parameters, lang.hitch(this, '_getAVGValues', pointClicked));
    else
        identifyTask.execute(parameters, lang.hitch(this, '_rasterValuesObtained'));

}
```

Esta función es la que se ejecuta cuando un usuario hace clic sobre un punto concreto del mapa en la sección de monitorizar. Se describe en forma de atributo porque es la forma en la que la librería DOJO es capaz de exportar los módulos y permitir trabajar en un formato similar al que se utilizaría en un lenguaje que permitiera usar clases.

El objeto `MosaicRule` permite definir la forma en la que se le realizan las consultas al mosaico de la librería de ArcGIS para JavaScript. Una vez creado se le indica como debe ser el orden, y basándose en qué método.

Esa regla se asocia a un parámetro, al que, además se le asocia una geometría, en este caso el punto concreto sobre el que se ha hecho clic. De este modo el objeto de tipo `ImageServiceIdentifyParameters` contiene toda la información necesaria para que la tarea de búsqueda sea capaz de consultar al servidor.

A continuación se crea el objeto que sabe hacer la consulta al servidor, se trata de `ImageServiceIdentifyTask` y únicamente necesita recibir por parámetro la URL a la que debe preguntar. En este caso esta está almacenada en el objeto principal en la variable `this.URL`.

Una vez se tiene la tarea con la URL correcta y los parámetros con los que se desea consultar se ejecuta el método “execute” para realizar la consulta.

La forma de ejecutarla merece ser explicada. En principio se le podría decir:

```
identifyTask.execute(parameters, MiFuncion());
```

De ese modo lanzaría la consulta y, cuando tuviera los resultados, ejecutaría la función MiFunción(). Sin embargo eso presentaría un problema con el ámbito de la nueva función, dado que no se garantiza que sea el mismo que el de la función que la está llamando. De hecho no lo es. Por ese motivo, si tratáramos de visualizar luego, por ejemplo, la URL con:

```
console.log(this.URL);
```

Encontraríamos que nos devuelve "undefined". Para solucionar esta problemática DOJO presenta la función "lang.hitch()" que permite modificar el ámbito de la función que se le pasa como parámetro. De modo que al pasarle "this" sí que es capaz de reconocer las variables propias de la función que le llama.

Se ha decidido explicar esta función porque presenta muchos de los conceptos que se han aprendido o fortalecido durante el desarrollo de este trabajo fin de master. Por un lado se ha observado la problemática de la sincronía. Cuando en las primeras fases de desarrollo se tuvieron problemas con los tiempo se plantearon algunas soluciones de espera activa, pero, tras hablar con los tutores, se buscaron métodos que fueran capaces de mejorar el rendimiento con funciones de callback. También ha sido interesante el trabajo con el ámbito de las funciones, dado que las primeras veces que se presentaban este tipo de errores se perdía mucho tiempo tratando de comprender por qué las funciones ejecutadas no eran capaces de visualizar los atributos de aquellas que las habían llamado.

6.2.2 Comunicación entre clases: Eventos.

En algunos casos, las clases deben comunicarse unas con otras. Esto es sencillo cuando se tiene referencia entre ellas, pero cuando la llamada es al revés esto no resulta tan sencillo.

Por ejemplo, la clase MainController tiene una referencia a la clase MenusController.

```
this.menusController = new MenusController({…})
```

De modo que ahora puede utilizar cuando sea necesario:

```
this.menusController.startup();
```

Que es una de sus funciones.

Sin embargo, cuando es MenusController la que debe comunicar algo a MainController la cosa es más compleja. Para ello se han utilizado eventos que permiten utilizar el patrón observador observable. En primer lugar se crean eventos, los ficheros son capaces de emitir eventos o de

suscribirse a ellos, ejecutando entonces una función de callback. Por ejemplo, MonitoringController puede decir “me han cambiado la imagen” con el siguiente código.

```
Topic.publish('imagenCambiada', parameter);
```

Cualquier objeto podría subscribirse a ese evento:

```
Topic.subscribe('imagenCambiada', MiFuncion());
```

De este modo es posible comunicarse entre objetos a pesar de no tener la referencia entre ellos.

La línea anterior presentaría, de todas formas, un problema de ámbito como el del ejercicio anterior, para solucionarlo, ahora que conocemos lang.hitch, bastaría con añadir:

```
Topic.subscribe('imagenCambiada', lang.hitch(this, 'MiFuncion'));
```

Esta forma de comunicar los eventos entre sí ha permitido que la aplicación quede más independiente, ya que cada clase únicamente es responsable de avisar de sus eventos, dejando que sean los otros objetos los que decidan cómo reaccionar a estos.

6.2.3 Generación Dinámica del Contenido

Dado que uno de los objetivos buscados en la aplicación era su capacidad de mantenerse actualizada automáticamente, mucha parte del código puramente HTML debe generarse de forma dinámica. Por ejemplo, para añadir un nuevo producto únicamente es necesario actualizarlo en el fichero de configuración y la aplicación ya lo integra en el navegador. Para ello se han utilizado muchas funciones de acceso y modificación del DOM:

```
populateLayersList: function(){
    var container = dom.byId('operational-layers');

    for(var layer in this.layers){

        var button = domConstruct.create("button");
        domAttr.set(button, "id", layer + "-button");
        domAttr.set(button, "type", "button");
        domAttr.set(button, "class", "btn btn-info");
        button.innerHTML = layer;
        var clickHandler = lang.hitch(this, "_showLayer", layer);
        this.own(on(button, "click", clickHandler));
        domConstruct.place(button, container, "after");
        var br = domConstruct.create("br");
        domConstruct.place(br, container, "after");
    }
}
```

Mediante las librerías de acceso al DOM de DOJO es sencillo crear elementos:

- `var button = domConstruct.create("button");`

Añadirles atributos:

- `domAttr.set(button, "id", layer + "-button");`

Insertarles contenido:

- `button.innerHTML = layer;`

Asignarles eventos:

- `var clickHandler = lang.hitch(this, "_showLayer", layer);`
- `this.own(on(button, "click", clickHandler));`

Y finalmente integrarlos en la página:

- `domConstruct.place(br, container, "after");`

6.2.4 Servicios Remotos.

Aunque cuando se ha explicado el tema de la sincronía ya se ha visto la forma de realizar consultas al mosaico, en este punto se desean enumerar las diferentes formas de consultar a los servidores externas que se han utilizados:

El primer servicio que se utiliza es el que se comunica con el **servidor de usuarios**, la forma de hacerlo es la siguiente:

```
_queryMongoServer: function(response){
    var serviceURL = "http://localhost:6585/api/parcelsinfo/";
    var username = this.username;
    var password = getCookie("password");
    var parcelid = response.features[0].attributes.PARCEL_ID;


    xhr(serviceURL, {
        handleAs: "json",
        method: "POST",
        data: {
            username:  username,
            password password,
            parcelid: parcelid
        },
        headers: {
            "X-Requested-With": null
        }
    }).then(lang.hitch(this, "_showInfoWindow"));

}
```

En primer lugar se asigna la URL del servicio, el nombre de usuario (lo tiene el objeto), la contraseña (se almacena en una cookie) y en este caso el identificador de la parcela a localizar (se le ha pasado como parámetro). Una vez se tiene la información necesaria se crea el objeto xhr responsable de hacer la petición. Se le indica cómo debe tratar la respuesta, que tipo de método es y los parámetros que se le pasan en el cuerpo del mensaje. Además, para asegurar que trabaje bien asíncronamente se utiliza el método "then" que se encarga de que la siguiente función se ejecute cuando haya terminado.

La URL del servicio actualmente se encuentra en el servidor de ERMES de la Universitat Jaume I y se mantiene escuchando en el puerto 6585.

Otro servicio remoto es el que proporciona las capas del mapa, denominadas "Feature Layers". Son capas que contienen información gráfica, para ser visualizada en el mapa, pero además almacenan información sobre las características que almacenan. Por ejemplo, en el caso de la capa de parcelas se almacenan las formas geométricas de cada una de ellas (*shapes*) e información relevante como el identificador de la parcela o su área. La forma de obtenerlas es la siguiente:

```
this.parcelsLayer = new FeatureLayer(response.parcelsLayer.url,
                    {
                        outFields: ["*"]
                    });
this.map.addLayer(this.parcelsLayer);
```

Se utiliza un objeto del tipo FeatureLayer, proporcionado por el API de ArcGIS para JavaScript al que, únicamente pasándole la URL de la capa y los parámetros que se desean obtener (en nuestro caso todos) nos devuelve la capa correspondiente. Después simplemente se le añade al mapa utilizando el método *addLayer*.

La capa, así como el resto de mosaicos e imágenes ráster consultadas, son servidas a través de un servidor ArcGIS Server instalado en el mismo servidor que el anterior servicio, pero escuchando en otro puerto.

La última forma que va a presentarse funciona de forma similar a la anterior, es la responsable de devolver las imágenes ráster seleccionadas de un mosaico en concreto.

```
getLayerByID: function(rasterId){
    var layer = new ArcGISImageServiceLayer(this.URL);
    var rule = new MosaicRule();
    rule.ascending = true;
    rule.method = MosaicRule.METHOD_LOCKRASTER;
    rule.lockRasterIds = [rasterId];
    layer.setMosaicRule(rule);
    return layer;
}
```

En primer lugar se obtiene la capa del mosaico mediante el objeto *ArGISImageServiceLayer.* Posteriormente se crea una regla de mosaico a la que se le asignan algunos parámetros como el orden, el tipo de petición y, en este caso, el ID concreto del ráster que se está buscando. Finalmente se le aplica esa regla a la capa se devuelve el resultado.

Como se puede comprobar en los ejemplos presentados gran parte de la aplicación se basa en el API de ArcGIS para JavaScript. La documentación existente en su página Web ha resultado de gran ayuda para todo el desarrollo del geoportal. En ella, además de la descripción de cada uno de los objetos, sus atributos y métodos, se presentan ejemplos y demos de muchas funcionalidades que facilitan mucho su comprensión e integración.

Para finalizar esta sección simplemente indicar que en el repositorio mencionado anteriormente pueden analizarse con detenimiento todos los aspectos relacionados con la implementación, sin embargo aquí han tratado de presentarse aquellos que han permitido agilizar el desarrollo o los que han supuesto una toma de decisiones relevante o han ayudado a proporcionar un conocimiento más profundo de algunos aspectos de la programación.

7 RESULTADOS

La aplicación final sirve como muestra de los resultados obtenidos. Actualmente se encuentra desplegada en el servidor de la Universitat Jaume I dedicado al proyecto ERMES:

http://ermes.dlsi.uji.es/

Dentro de este servidor existen más aplicaciones del mismo proyecto, así como algunas versiones preliminares o prototipos futuros. La aplicación aquí presentada es accesible, concretamente, desde la URL:

http://ermes.dlsi.uji.es/prototype/geoportalv2

Al acceder a la página principal (index.html) se comprueba si el usuario está registrado, de no estarlo se redirecciona a login.html (Figura 11).

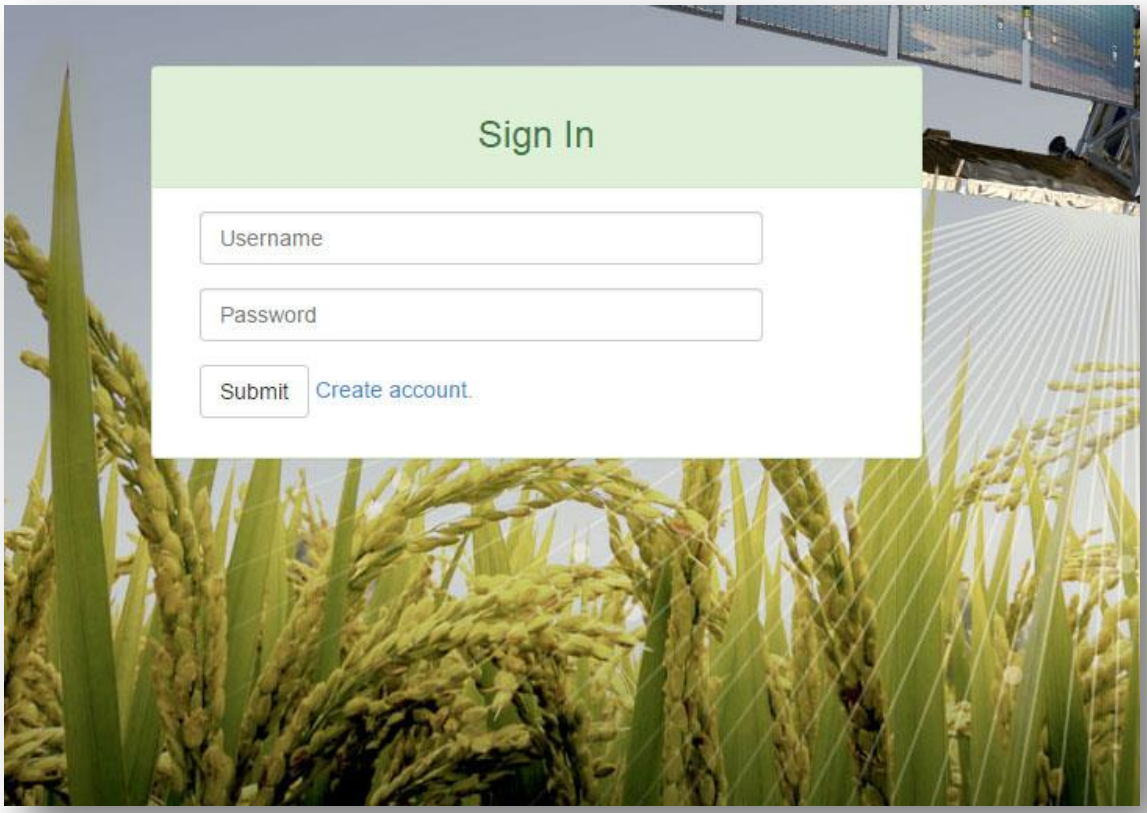

Figura 11. Pantalla de login del Geoportal.

Desde ahí es posible identificarse o acceder a la pantalla de registro (Figura 12). Actualmente el registro es libre y cualquiera puede acceder creándose un usuario en cualquier región o perfil. Esto es así porque desde la coordinación del proyecto ERMES se ha decidido mantener

de este modo para facilitar el acceso y pruebas por parte de los diferentes usuarios regionales. Para el futuro está previsto limitar privilegios a los usuarios para que solo algunos de ellos sean capaces de crear o modificar nuevos usuarios y perfiles. Sin embargo esta gestión queda fuera del alcance de este TFM.

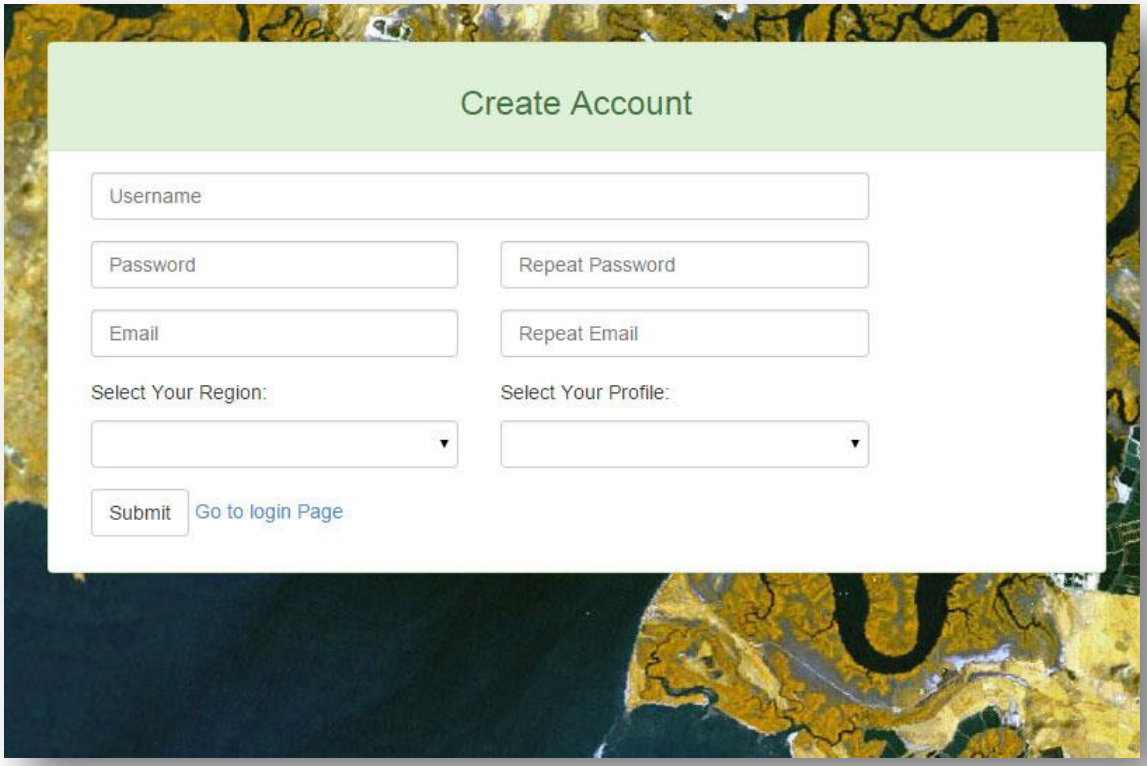

Figura 12. Pantalla de registro del Geoportal.

Una vez el usuario se ha registrado como usuario regional accede a la pantalla de inicio (Figura 13). Esta depende de su región y perfil. Dado que la mayor parte de las funcionalidades están generadas para el perfil regional (el local tiene menos privilegios en estos momentos) y que la mayoría de productos proporcionados pertenecen al área del norte de Italia, las capturas de esta sección se van a centrar en este tipo de usuario: Perfil: Regional. Región: Italia.

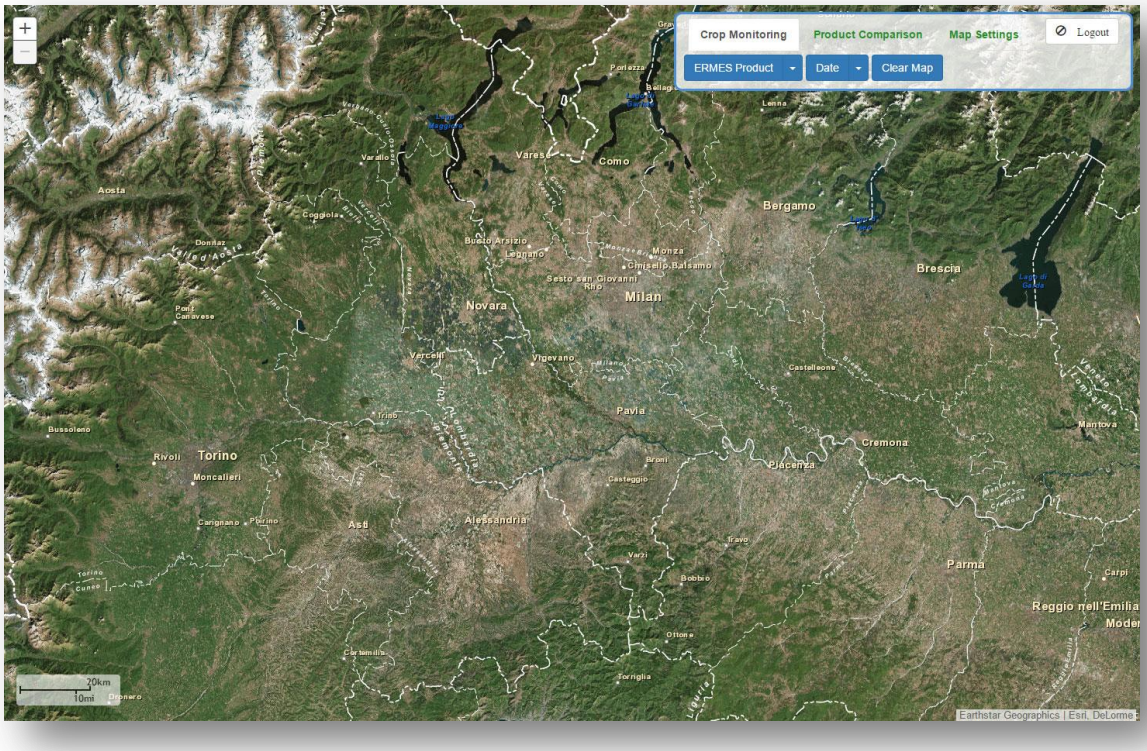

Figura 13. Pantalla de inicio del Geoportal para un usuario de Italia con un perfil Regional.

La interfaz trata de ser minimalista, tal y como se había definido en los *mockups*. Se presentan las principales herramientas del mapa: escala y posibilidad de hacer zoom in y zoom out. Además, el panel desde el que se monitorizarán, compararán y configurarán los productos deseados.

Por ejemplo, en la pantalla de configuración (Figura 14) es posible seleccionar o deseleccionar las capas operacionales a visualizar, además de cambiar el mapa base. Si por ejemplo un usuario desea ver las fronteras provinciales en un mapa de calles debería configurarlo como se muestra en la Figura 15.

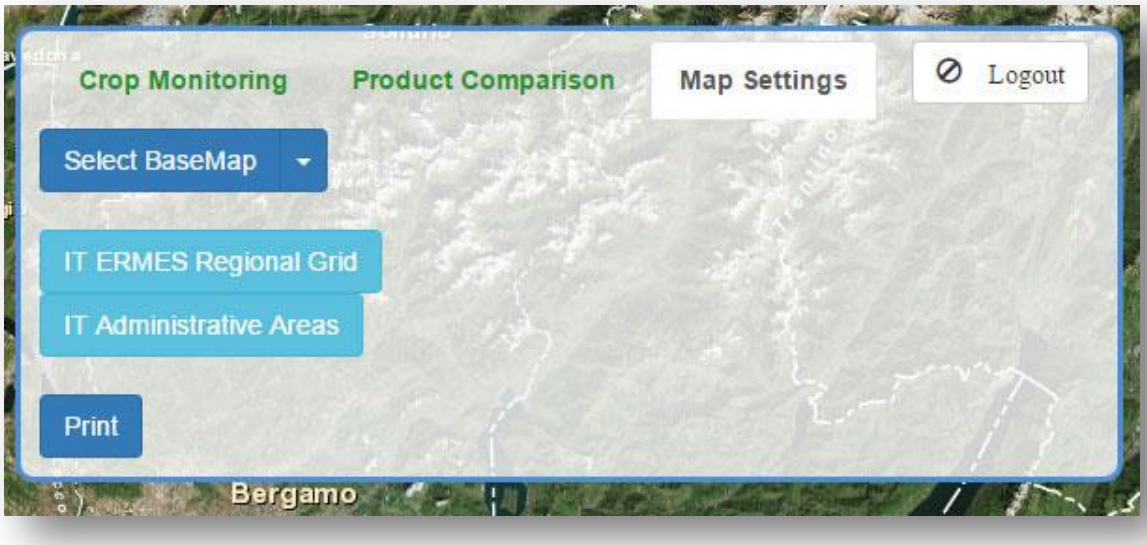

Figura 14. Menú de configuración del Geoportal.

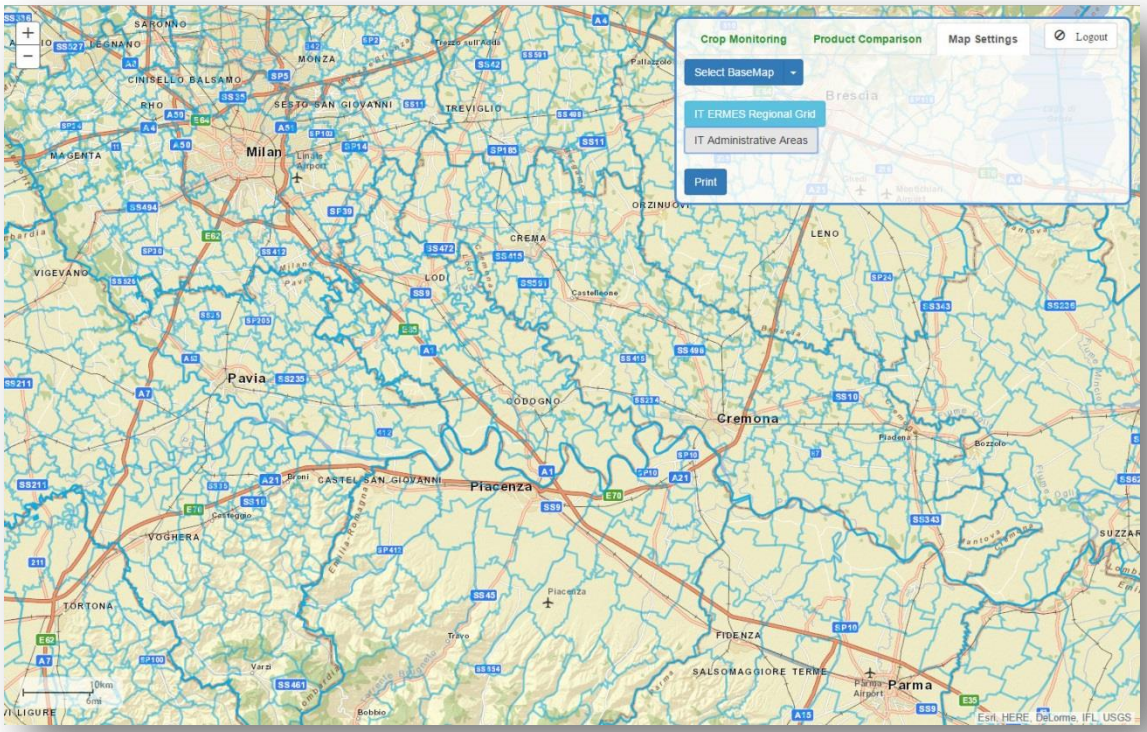

Figura 15. Ejemplo de configuración con mapa de calles y fronteras provinciales.

Para ejecutar un análisis el usuario debe acceder a la pestaña de monitorización (la primera) y seleccionar uno de los productos (Figura 16), a continuación se rellena el campo de fechas (Figura 17) y se le permite seleccionar cuál de ellas desea visualizar sobre el mapa.

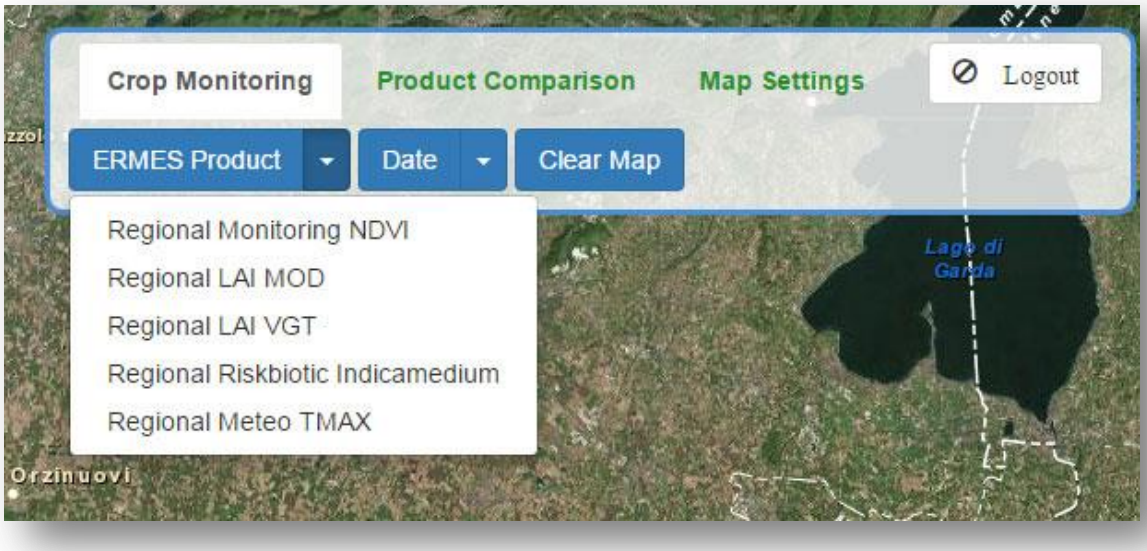

Figura 16. Selección de producto.

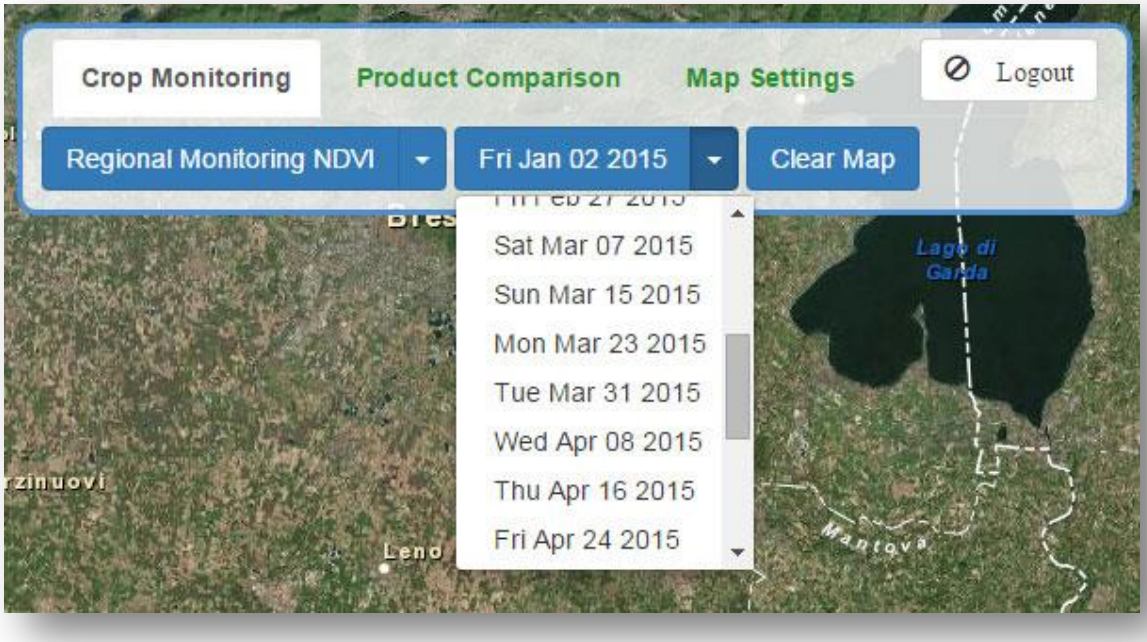

Figura 17. Selección de fecha.

Una vez seleccionado el producto se carga sobre el mapa y es posible visualizarlo. Por ejemplo para el índice de vegetación (NDVI) del 7 de Marzo se obtiene la imagen visualizada en la Figura 18.

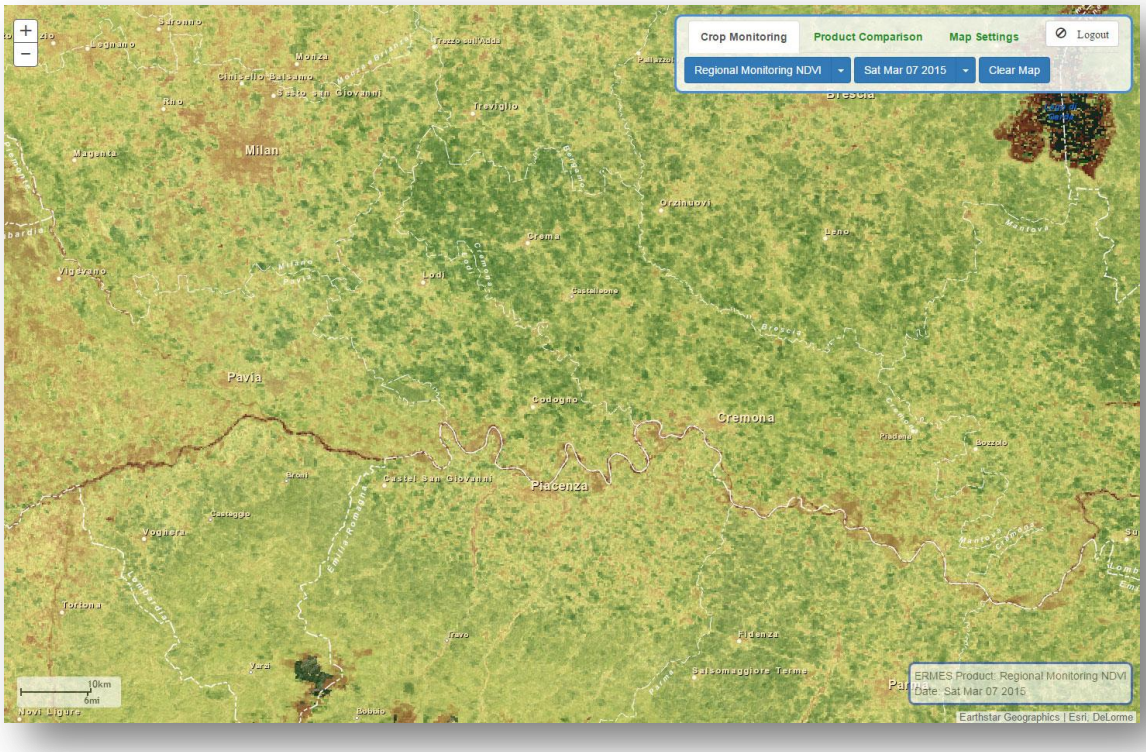

Figura 18. Ejemplo de NDVI para el 7 de marzo.

En este momento, haciendo clic sobre cualquier parte de la capa cargada es posible analizar en profundidad los valores que posee ese punto en concreto. Además, para aquellos productos para los que se cuenta con la media de los últimos años, también es posible visualizarla en la gráfica, el producto NDVI es uno de ellos.

Por ejemplo, el producto LAI (Leaf Area Index) no posee históricos todavía. Por lo tanto, si analizamos un punto del mapa del 18 de mayo nos encontramos con la gráfica de la Figura 19. En ella puede visualizarse sobre el mapa un símbolo con el punto donde se ha hecho clic y que está siendo analizado; un valor indicando lo que vale ese producto en la fecha y punto seleccionados; y debajo una gráfica que muestra la evolución de ese producto en ese punto, marcando sobre la gráfica el valor actual de forma que sea fácilmente visible. Además, si el usuario pasa el puntero del ratón por los diferentes puntos le aparece una ventana emergente (tip) que le muestra el valor exacto para ese punto. En este caso se ha marcado el de una fecha posterior.

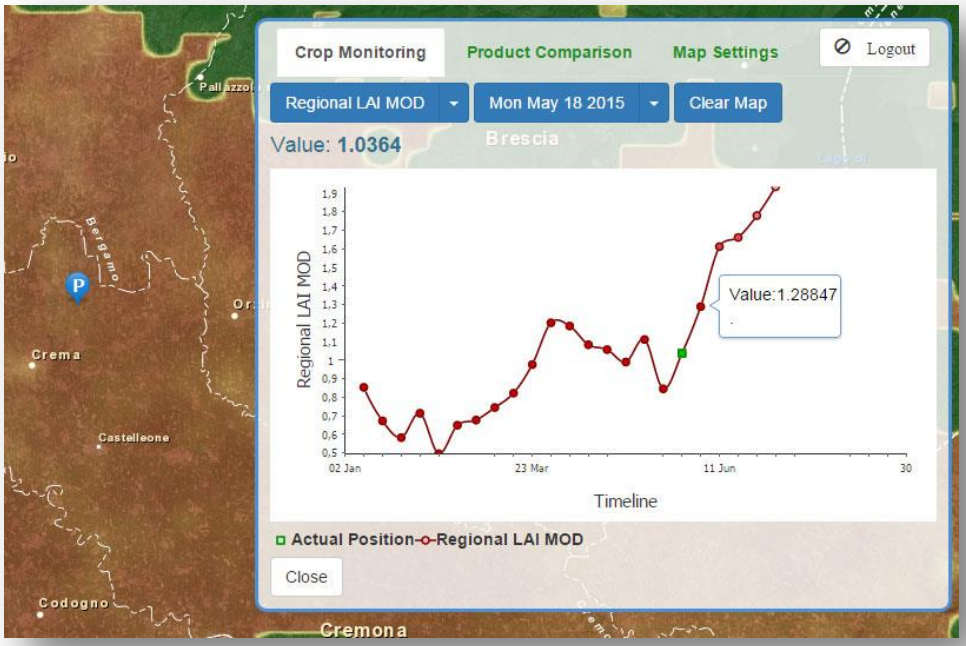

Figura 19. Ejemplo de gráfica sin histórico.

Sin embargo, si realizamos la misma operación sobre un producto con histórico, como NDVI, el resultado añade la media de los últimos años a la gráfica (Figura 20).

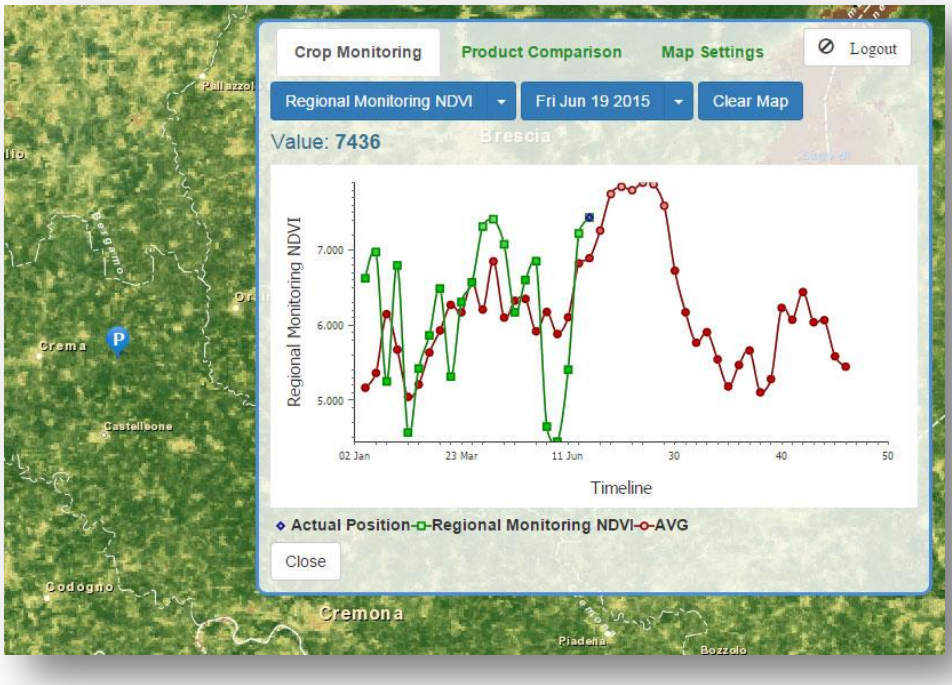

Figura 20. Ejemplo de gráfica con histórico.

La última funcionalidad incluida era la relativa a la comparación. El usuario selecciona el primero de los productos a comparar en la parte de monitorización, supongamos que ha seleccionado NDVI para el 3 de junio. A continuación se va a la pestaña de comparación, donde puede seleccionar del mismo modo que ha hecho antes, un nuevo producto y una nueva fecha. Imaginemos que quiere compararlo con las temperaturas máximas de ese mismo día Figura 21.

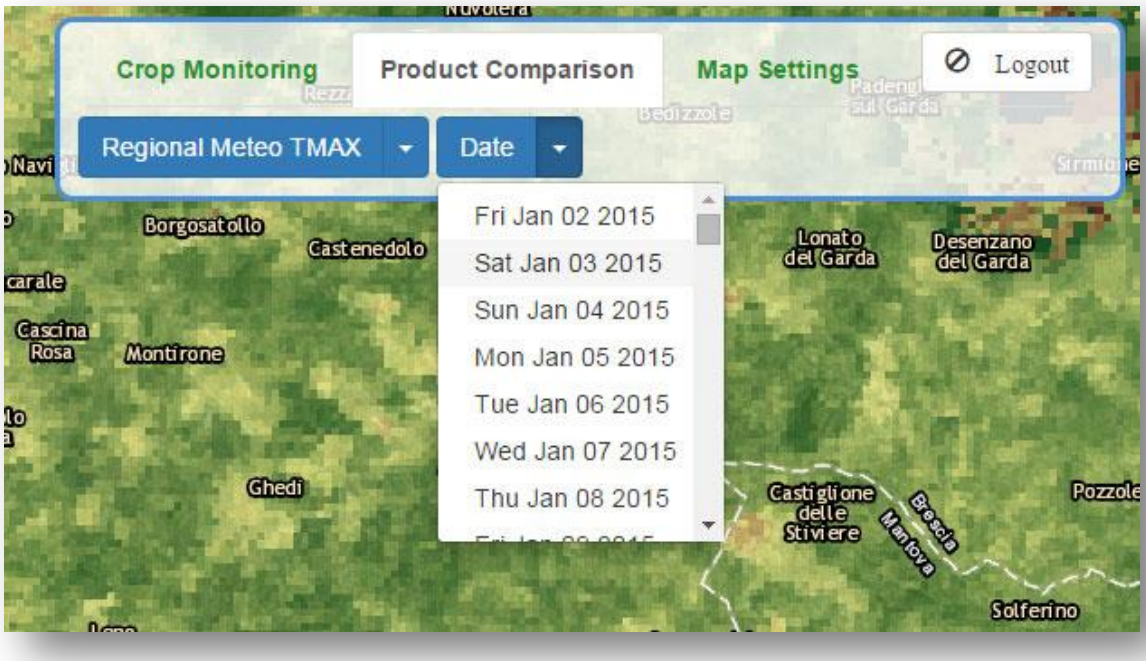

Figura 21. Selección de producto y fecha para el menú de comparación.

Una vez esta seleccionado el segundo producto aparece un botón para Comparar, al ser pulsado se genera el widget consistente en una barra vertical que permite que el usuario desplace a la izquierda o la derecha mostrando u ocultando el producto comparado.

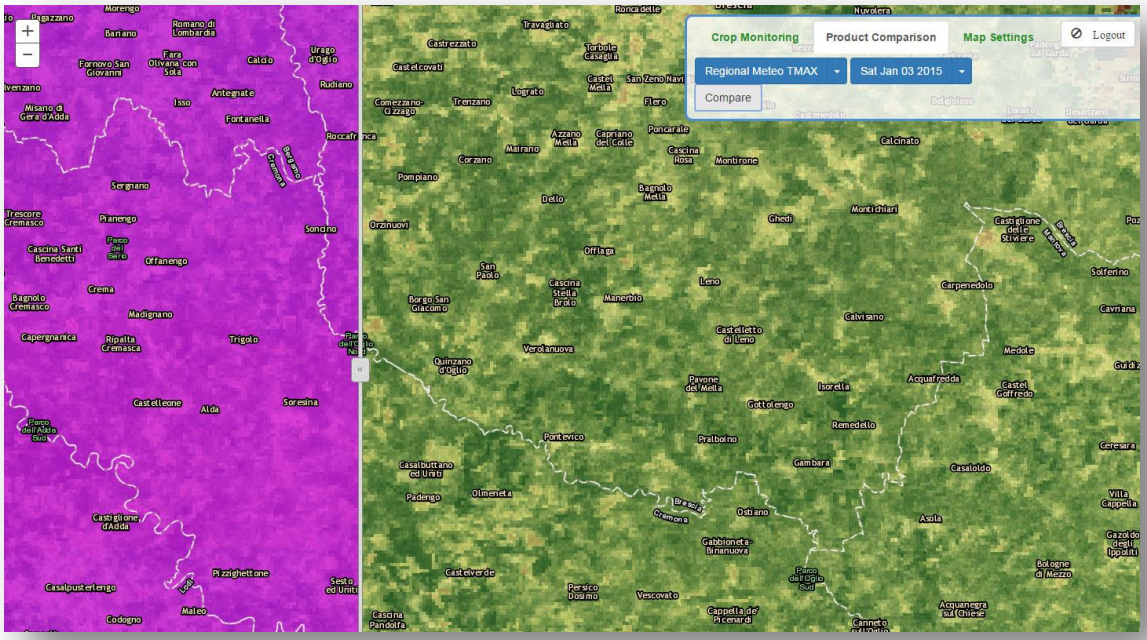

Figura 22. Ejemplo de la herramienta de comparación.

Para consultar más ejemplos es posible acceder a la página Web y realizar las pruebas deseadas.

Con esta funcionalidad la herramienta cumple con las necesidades planteadas por parte de los coordinadores y usuarios del proyecto. Además, con respecto a los objetivos formativos planteados, el estudiante ha obtenido la formación necesaria en los diferentes aspectos destacados en el apartado de objetivos. En la Tabla 7 se validan los objetivos planteados al comienzo del trabajo con los resultados obtenidos.

Puede comprobarse que todos los objetivos se han cumplido excepto “OG07: Incluir previsiones sobre los cultivos basándose en un modelo generado por otros miembros del consorcio.”. Esto realmente escapaba a las posibilidades del estudiante dado que el modelo predictivo es generado por otros miembros del consorcio que, con la fecha de redacción de este trabajo, no tenían disponibles los datos. Sin embargo la aplicación está perfectamente preparada y modularizada para que en el momento de recibir los datos su inclusión en el portal sea prácticamente inmediata (bastará con añadir unas líneas en el fichero de configuración).

Objetivos Formativos		Objetivos Geoportal	
OF01	✓	OG01	✓
OF02	✓	OG02	✓
OF03	✓	OG03	✓
OF04	✓	OG04	✓
OF05	✓	OG05	✓
OF06	✓	OG06	✓
		OG07	✗
		OG08	✓

Tabla 7. Validación de objetivos planteados y resultados obtenidos.

8 CONCLUSIONES

La necesidad de un enfoque práctico y productivo en las investigaciones y proyectos relacionados con nuevas tecnologías es algo incuestionable. Dentro del proyecto ERMES se trata de facilitar y mejorar las tareas del sector agrícola mediante el uso de las TIC. En concreto, en el desarrollo enmarcado en este TFM se ha proporcionado una aplicación que da la posibilidad de monitorizar y analizar la información de forma sencilla y obteniendo en pocos pasos datos relevantes.

Todo el proceso de desarrollo ha sido de gran utilidad en la formación del alumno. La integración en un proyecto de gran envergadura, la implicación de los tutores, ambos con amplia experiencia en el campo de las tecnologías geoespaciales y las herramientas de monitorización, reuniones con expertos de diferentes materias como *Remote Sensing*, GIS, Desarrollo Web, etc. han resultado muy fructíferas, proporcionando al estudiante técnicas, metodología y conocimientos de gran valor. Toda la parte del trabajo en equipo, el compromiso con el resto de compañeros implicados, los *Code Sprints* supervisados y el *feedback* proporcionado por los tutores ha supuesto un marco perfecto para el desarrollo del TFM.

Por contrapartida, esos mismos factores han supuesto alguna limitación temporal que ha obligado al estudiante, en algunos momentos, a terminar algunas de las funcionalidades sin profundizar todo lo deseable en la tecnología en concreto. Las reuniones con los coordinadores del proyecto priorizaban tener fases preliminares del proyecto finalizadas por lo que algunas veces el código no podía analizarse o refactorizarse todo lo que hubiera sido deseable.

Con respecto a los resultados obtenidos, por parte de los coordinadores el *feedback* recibido es muy positivo, la herramienta cumple con lo deseado y está siendo utilizada para monitorizar la temporada de arroz actual. Se han presentado algunas sugerencias de modificación de la interfaz y, fundamentalmente, posibilidad de añadir nuevos servicios para la monitorización de diferentes productos.

Por supuesto el trabajo con el proyecto ERMES no termina aquí por parte del estudiante. Además de mejorar y continuar ajustando la herramienta aquí presentada, son necesarios nuevos desarrollos sobre los que investigar las mejores prácticas y analizar cómo llevarlos a cabo. Por ejemplo todavía es necesario perfeccionar la herramienta de recolecta de datos en el campo, que debe ser capaz de funcionar y proporcionar la información relevante incluso

estando desconectado de Internet y debe funcionar al menos en teléfonos móviles con el sistema operativo Android.

También queda mucho trabajo por hacer con el servidor de usuarios e información de parcelas. Actualmente el servicio proporciona la funcionalidad básica para acceder y almacenar información, sin embargo hay que realizar un estado del arte de cuáles son las mejores prácticas para cada una de las opciones necesarias y refactorizar todo el código tratando de garantizar la seguridad, estabilidad y rendimiento lo máximo posible para cuando el número de usuarios que deba soportar se incremente.

Por último, es relevante agradecer a los tutores todo el trabajo que han realizado desde la primera toma de contacto con el estudiante hasta el momento de la revisión de este documento. Han mostrado desde un primer momento absoluta facilidad de acceso, sobrados conocimiento y han permitido al estudiante desarrollarse con libertad pero cumpliendo con los objetivos requeridos por el proyecto.

BIBLIOGRAFÍA

[1]. Página Web del INIT: http://www.init.uji.es/

[2]. Página Web GEOTEC: http://www.geotec.uji.es/

[3]. Página Web del proyecto ERMES: http://www.ermes-fp7space.eu/

[4]. Ospanov, S. S., Kaliyeva, A. Y., Dulambaeva, R. T., Aubakirova, Z. Y., y Tabeev, T. P. (2015). Competitiveness of the Agricultural Sector as a Factor in Improving Food Security in the Conditions of Globalization. Review of European Studies, 7(7), p307. Página Web de la ESA: http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview4

[5]. Página Web de la ESA: http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview4

[6]. Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M., & Ochiai, O. (2015). Big Data challenges in building the Global Earth Observation System of Systems. Environmental Modelling & Software, 68, 1-26.

[7]. ArcGIS for Server. http://www.esri.com/software/arcgis/arcgisserver

[8]. HTML. http://www.w3schools.com/html/

[9]. CSS. http://www.w3schools.com/css/

[10]. JavaScript. http://www.w3schools.com/js/

[11]. ArcGIS API for JavaScript. https://developers.arcgis.com/javascript/

[12]. Bootstrao. http://getbootstrap.com/

[13]. jQuery. https://jquery.com/

[14]. Dojo Toolkit. https://dojotoolkit.org/

[15]. Git. https://git-scm.com/

[16]. GitHub. https://github.com/

[17]. Node.JS. https://nodejs.org/

[18]. MongoDB. https://www.mongodb.org/

[19]. WebStorm. https://www.jetbrains.com/webstorm/