Universidad Pública de Navarra

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS AGRONOMOS

Nafarroako Unibertsitate Publikoa

NEKAZARITZAKO INGENIARIEN

GOI MAILAKO ESKOLA TEKNIKOA

TRABAJO FIN DE MASTER

DESARROLLO DE UNA BASE DE DATOS Y VISOR WEB PARA UN SISTEMA DE IoT DE MEDICIÓN DE HUMEDAD DE SUELO

Presentado por:

UNAI GÓMEZ IBÁÑEZ *(e)k*

aurkeztua

Dirigido por:

CÉSAR ARRIGA EGÜÉS *(e)k*

MIGUEL ÁNGEL CAMPO BESCOS *(e)k*

zuzendua

Septiembre, 2018 / *2018, iraila*

MÁSTER UNIVERSITARIO EN SISTEMAS DE INFORMACIÓN GEOGRÁFICA Y TELEDETECCIÓN

AGRADECIMIENTOS

En primer lugar, quiero dar las gracias a mis Directores Académicos, Miguel Ángel Campo Bescos y César Arriaga Egüés, por haber sido mis guías y facilitarme toda la ayuda necesaria en la elaboración de este proyecto. Gracias por el apoyo y dedicación ofrecida.

A mis compañeros de estudios, principalmente aquellos con los que he estado en estos últimos tiempos, gracias por amenizar las muchas horas de trabajo compartidas.

Y por supuesto a toda mi familia, *aita, ama eta arreba.*

A todos vosotros, mila esker.

RESUMEN

Una de las tecnologías que más auge está experimentando en los últimos años es la denominada como Internet de las cosas (IoT), debido principalmente, al desarrollo de las opciones de conectividad de aparatos y la disminución de costes de estos sistemas. Este concepto hace referencia al creciente número de dispositivos capaces de capturar y gestionar todo tipo de datos, con la finalidad de incrementar la productividad y la eficiencia en diferentes ámbitos o sectores.

Uno de esos sectores es la agricultura, donde la medición de variables agroclimáticas permite conocer las condiciones reales de desarrollo de los cultivos. Entre dichas variables, el contenido de humedad en el suelo es de especial interés, a causa de la importancia del consumo de recursos hídricos en cualquier explotación agrícola.

Este trabajo desarrolla una aplicación IoT compuesta por un sistema de procesado de datos, una base de datos espacial y un visor web, que a partir de mediciones de contenido de humedad del suelo de parcelas agrícolas ayuda en la toma de decisiones para la óptima gestión de sistemas de riego, reduciendo las visitas a campo. La aplicación se ha llevado a cabo mediante herramientas de software libre o sin coste como son la base de datos espacial PostgreSQL y PostGIS y la API SITNA. Esto permitiría una fácil adaptación e implementación en otros ámbitos similares.

Palabras clave: IoT, Agricultura de precisión, Base de datos espacial, Software libre, Web mapping.

ABSTRACT

One of the technologies that is booming in recent years is the so-called Internet of Things (IoT), mainly due to the development of device connectivity options and the reduction of costs of these systems. IoT refers to the growing number of devices capable of capturing and managing all types of data, in order to increase productivity and efficiency in different fields or sectors.

In the agricultural sector, the measurement of agroclimatic variables helps improve the knowledge of the real conditions of crop development. Among these variables, the moisture content in the soil is of special interest, due to the high importance that the consumption of water resources has in any farming business.

This work develops an IoT application composed of a data processing system, a spatial database and a web viewer, which, based on soil moisture content measurements from agricultural plots, helps in making decissions for the optimal management of irrigation systems, reducing field trips. The application has been carried out using free or open software tools such as the PostgreSQL and PostGIS spatial database and the SITNA API. This would allow easy adaptation and implementation in other similar areas.

Keywords: IoT, Precision agriculture, Spatial database, Free software, Web mapping.

ÍNDICE MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La búsqueda de sistemas, tecnologías y métodos que optimicen la producción agrícola con el objetivo de obtener mayores rendimientos a un menor costo y con un menor uso de recursos ha sido una constante en la historia de la agricultura. En los últimos años, esta búsqueda se ha acentuado con la denominada agricultura de precisión (AP), cuya idea esencial radica en analizar las necesidades específicas de los cultivos en una explotación determinada, con el fin de generar soluciones que se adapten lo mejor posible a cada caso concreto (Bongiovanni, Mantovani, Best, y Roel, 2006; Emmen, 2004).

La disponibilidad de nuevas tecnologías para el monitoreo, análisis y control de todas las variables que afectan al proceso de producción agrícola ha supuesto un gran avance para la AP, en la medida que ha permitido mejorar sustancialmente el conocimiento de las explotaciones y de la idoneidad de las soluciones aplicadas. Con la utilización de nuevas tecnologías, se posibilita una toma de datos más precisa y completa, así como un mejor análisis de la información recogida, facilitando una toma de decisiones más fundamentada (Emmen, 2004).

Concretamente, los equipos de procesamiento, servidores, nuevos sensores agrícolas remotos, sistemas de información geográfica (SIG) y de visionado de datos son algunos de los elementos más utilizados en la AP (Flores Medina, Velasco M., y González C., 2015).

El monitoreo de variables agroclimáticas es una herramienta esencial de la AP, dado que permite conocer las condiciones reales de desarrollo de los cultivos a lo largo de todo su ciclo. Entre dichas variables, el contenido de humedad en el suelo cobra una especial significación, debido a la importancia del consumo de recursos hídricos en cualquier explotación agrícola (Chávez Ramírez, 2007; Flores Medina *et al*., 2015). La agricultura es uno de los sectores con mayor consumo de agua y el uso eficiente de ésta es un imperativo de la máxima vigencia en el contexto de una agricultura sostenible (Pfister, Bayer, Koehler, y Hellweg, 2011).

Entre los sistemas más avanzados de medición del grado de humedad del suelo se encuentran los denominados sensores capacitivos (Martin y Munoz, 2017). Se trata de unos dispositivos compuestos por sondas, normalmente enterradas a diferentes alturas, que determina el contenido volumétrico de agua (VWC) a partir de estimar la constante dieléctrica del suelo (Evett, Schwartz, Casanova, y Heng, 2012). Adicionalmente, es posible conectar estos sistemas a un equipo de almacenamiento de los datos, que su vez esté en comunicación inalámbrica con plataformas alojadas en la nube.

Una red inalámbrica de sensores o RIS por sus siglas en inglés, consiste en un sistema que es capaz de medir, almacenar, filtrar y transferir de manera inalámbrica los datos captados por un sensor (Flores Medina *et al*., 2015). Este tipo de dispositivos, por lo demás, se inscribe en el conjunto de tecnologías conocidas como Internet de las Cosas (IoT). Este concepto se refiere al creciente número de dispositivos capaces de capturar y gestionar todo tipo de datos, con la finalidad de incrementar la productividad y la eficiencia en diferentes ámbitos o sectores, resultado del desarrollo de las opciones de conectividad de aparatos y la disminución de costes de estos sitemas (García-Hernández, Ibarguengoytia-Gonzalez, García-Hernández, y Pérez-Díaz, 2007; Zheng y Jamalipour, 2009).

En línea con lo anterior, la combinación de tecnologías GIS con bases de datos espaciales constituye una herramienta muy importante y extendida para el análisis y representación de elevadas cantidades de datos (Goswami, Matin, Aruna, y Bairagi, 2012). En el caso de la AP, la recolección y visualización (mapas temáticos georreferenciados) de los datos en un GIS es una de las aplicaciones más comunes, ya que ofrecen la posibilidad de realizar análisis geoestadísticos e historiales ocurridos en los cultivos (Kubicek, Kozel, Stampach, y Lukas, 2013; Wilson, Mitasova, y Wright, 2000).

Sin embargo, obtener los datos de campo en un formato manejable por los GIS requiere equipos especializados (Godwin y Miller, 2003). Es en este contexto donde el desarrollo de sistemas IoT en colaboración con otras tecnologías, como pueden ser el cálculo en la nube o las tecnologías GIS web, ha permitido expandir esta aplicación en la AP (Anaya-Isaza Diego H Peluffo-Ordoñez, Ivan-Rios Juan Castro-Silva, y Carvajal Ruiz Luis H Espinosa Llanos, 2017; Guerrero-Ibañez *et al*., 2017).

Existen diferentes líneas de investigación ligadas al desarrollo de sistemas IoT aplicados a la AP. Algunos trabajos se centran en el monitoreo, normalmente en tiempo real, de diversas variables agroclimáticas como la temperatura o la humedad del suelo (Ryu *et al.*, 2015; Wenting *et al.*, 2014). Otras investigaciones están más enfocadas a la creación de plataformas IoT que cuentan con instrumentos y utilidades que sirven de apoyo para la gestión de cultivos agrícolas, por ejemplo, controlando el consumo de agua y programando el riego. Estas plataformas normalmente se alojan en la nube y pueden ser utilizadas desde dispositivos móviles (Hari Ram, Vishal, Dhanalakshmi, y Vidya, 2015; Navarro-Hellín *et al.*, 2015; sigAGROasesor, 2018)

La mayoría de estas aplicaciones tienen en común que generan una gran cantidad de datos, por lo que el visionado de los mismos es un aspecto fundamental para su utilización. Por ello, cada vez son más relevantes las denominadas tecnologías GIS web, ya que permiten la representación de cartografía como un elemento más de una página Web. Olaya (2011) define estas tecnologías como Web Mapping, cuyo principal objetivo es trasladar las funcionalidades de un GIS a la Web, especialmente aquellas que se encuentran del lado del cliente, para así compartir el potencial de ambos componentes.

Entre las ventajas de emplear una tecnología Web Mapping, en lugar de un GIS de escritorio tradicional destacan las siguientes:

- No es necesario un software GIS específico.
- Perfil menos técnico.
- Potenciamiento del trabajo colaborativo.
- Información más actualizada.
- Independencia del sistema operativo.
- Personalización de aplicaciones.
- Combinación de cartografía y otros elementos.

La cartografía que se incluye en la web se puede clasificar en mapas estáticos o dinámicos en función de si se trata de una imagen fija o si varía y se adapta en función de los requerimientos del usuario. En la mayoría de las situaciones, son los mapas dinámicos los más indicados para la cartografía en la web (Olaya, 2011).

De esta manera, el ámbito en el que se centra el presente TFM es en el desarrollo de una aplicación de IoT compuesta por un sistema de captura y procesado de datos, incluyendo el diseño y desarrollo de una base de datos espacial y un visor web, que mediante mediciones de contenido de humedad del suelo de parcelas agrícolas ayude en la toma de decisiones para la óptima gestión de sistemas de riego, reduciendo las visitas a campo.

1.1 CONTEXTO DEL TFM

La Universidad Pública de Navarra (UPNA) ha iniciado un proceso de colaboración con la empresa navarra Embeblue S.L. orientado al diseño de dispositivos de gran autonomía y bajo coste, que permitan la monitorización del nivel de humedad existente en el suelo agrícola (Embeblue S.L, 2018).

Figura 1. Logotipo empresa Embeblue S.L. (Fuente: Embeblue S.L, 2018)

La actividad de Embeblue se centra en la ingeniería electrónica, especializada en dispositivos electrónicos para el IoT y la Industria 4.0. Entre los servicios que oferta, destacan los siguientes:

- Consultoría tecnológica.
- Diseño de electrónica.
- Programación firmware.
- Fabricación de prototipos y pequeñas series.
- Proyectos de I+d+i.

Hasta el momento, fruto de dicha colaboración, se han desarrollado e instalado en campo de maíz tres dispositivos de medición de humedad del suelo, uno en el municipio de Miranda de Arga (Navarra) y los otros dos en Enériz (Navarra).

Los sistemas instalados están compuestos por sondas 10HS, que determinan el contenido de agua volumétrica midiendo la constante dieléctrica mediante tecnología de capacitancia. Las sondas están conectadas, a su vez, a un micro-controlador SAMD21 que es capaz de medir y registrar en memoria datos cada media hora y enviar dicha información, a través de una conexión GPRS, a la plataforma de IoT de código abierto ThingSpeak. Una descripción más amplia sobre los dispositivos desarrollados puede consultarse en el apartado *3.1 Material*.

El desarrollo de los dispositivos descritos no es objeto de estudio del presente TFM, si bien constituye el punto a partir del cual alcanzar los objetivos planteados por este trabajo y que se describen en el apartado *2. Objetivos.*

Hasta el momento, los datos de los dispositivos instalados únicamente se almacenan en la base de datos de la plataforma IoT ThingSpeak, plataforma de almacenaje gratuido con restricciones. A partir de aquí, surge la idea de crear un sistema que facilite la visualización, comprensión y utilización de los datos almacenados, y ofrecer así un servicio de valor añadido a los agricultores para la mejora de su actividad.

2. OBJETIVOS

En línea con lo señalado hasta ahora, el presente trabajo tiene como objetivo principal el aportar valor añadido a los datos de humedad del suelo que actualmente miden y almacenan los dispositivos desarrollados por la empresa Embeblue en colaboración con la UPNA.

La aportación de valor consistirá en el desarrollo de una página web en la que diariamente y de manera automática serán agregados los datos procesados a partir de la información aportada por los dispositivos referidos. Ello permitirá consultar desde cualquier punto con conexión a internet los principales parámetros de humedad registrados y que son de ayuda de cara a la toma de decisiones sobre el riego de las parcelas.

Los objetivos específicos que se derivan del objetivo principal son:

- Desarrollar, mediante programación en Python, una rutina de descarga, procesado y posterior almacenamiento en una base de datos espacial de los datos de humedad del suelo capturados por los tres dispositivos mencionados.

- Crear una base de datos espacial donde se guarde toda la información relacionada con los dispositivos de medición. Además de para la gestión de los datos, se empleará para realizar análisis geoestadísticos.

- Desarrollar una página web intuitiva y visual que incluya:

    - Gráficos de agua total disponible en el suelo y su desarrollo histórico.
    - Gráficos de la variación de humedad a diferentes alturas del suelo. (en función de cada sonda instalada).
    - Estadísticas básicas, como el promedio, el mínimo, el máximo o la cantidad de registros de humedad del suelo.

- Añadir a la página web un visor donde se puedan consultar los datos más relevantes ubicados en el territorio.

- Emplear herramientas de software libre, como son el lenguaje de programación Python, la base de datos espacial PostgreSQL y PostGIS y la herramienta GIS de API SITNA, en la creación de la aplicación de IoT, que permita una fácil adaptación e implementación en otros ámbitos similares.

Además, este TFM pretende servir como base para el desarrollo de una futura plataforma de IoT que englobe las diferentes partes de este tipo de sistemas (dispositivos de medición, procesado y almacenamiento de datos) y cuente con un visor con funcionalidades GIS, e integre información de imágenes satelitales.

3. MATERIAL Y MÉTODOS

En este apartado se presentan y describen los materiales empleados en el diseño y desarrollo de la aplicación IoT, así como el proceso llevado a cabo para su selección.

3.1 DESCRIPCIÓN DISPOSITIVOS DE MEDICIÓN DE HUMEDAD

Los dispositivos desarrollados por Embeblue en colaboración con la UPNA (*Figura* 2) son equipos compuestos por los siguientes elementos:

- Sondas de medición de humedad.
- Microcontrolador SAMD21.
- Caja estanca (IP65) con cuatro salidas de conexión.

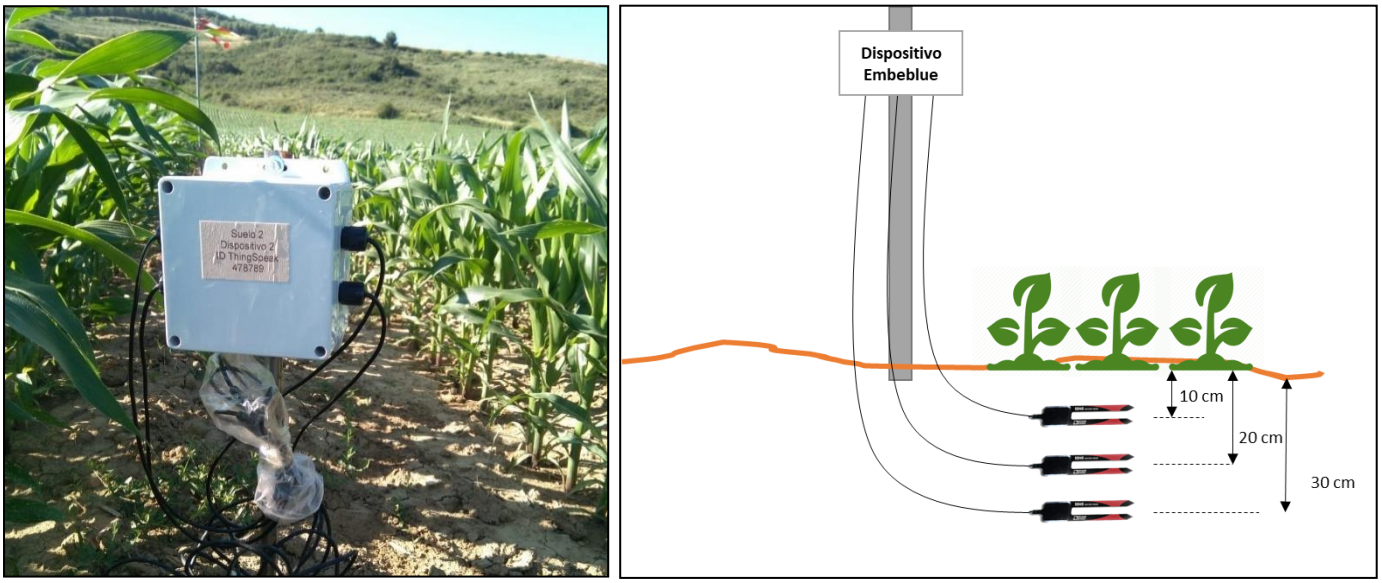

Figura 2. Dispositivo IoT de medición de humedad del suelo y esquema de instalación en campo.

Las sondas se conectan al microcontrolador y se entierran en la parcela en paralelo a diferentes alturas. A modo orientativo, en el siguiente enlace puede consultarse un vídeo explicativo sobre la instalación en campo de este tipo de equipos[1].

Cabe destacar que dos dispositivos están instalados en una parcela experimental del Instituto Navarro de Tecnología e Infraestructuras Agroalimentarias (INTIA) y la configuración de las sondas, dos sondas por un lado y otras dos por otro enterradas la misma altura, hacen que en cuanto al tratamiento de datos se les considera dispositivos independientes.

Un esquema de la configuración de estos dispositivos se recoge en la *Fgura 3*:

[1] Eemplo de instalación de dispositivos en campo: https://www.metergroup.com/environment/articles/video-install-soil-moisture-sensors/?utm_source=redirect#Soil%20Moisture%20Sensor%20Installation%20Video

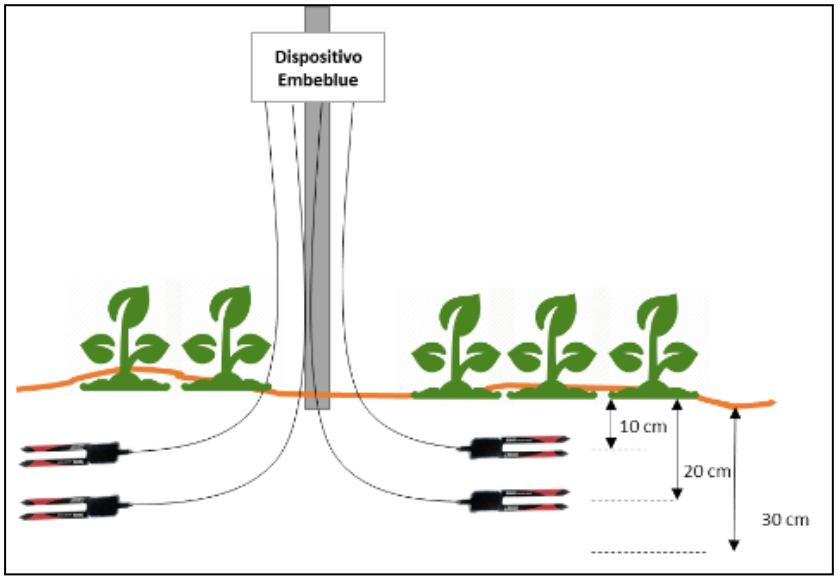

Figura 3. Esquema de configuración de sondas dispositivos de INTIA.

3.1.1 Sondas de medición de humedad

3.1.1.1 Funcionamiento

Las sondas que incorpora el dispositivo son las 10 HS de la marca comercial Decagon. Se trata de sensores capacitivos FDR (Frequency Domain Reflectrometry, Reflectometría en el dominio de la frecuencia) que miden la constante dieléctrica o permitividad del suelo.

El funcionamiento de estos instrumentos está basado en que el agua tiene una constante dieléctrica (80 ds/m) mucho mayor que el resto de compuestos del suelo (materia orgánica 4 ds/m o el aire 1 ds/m). De esta manera, las sondas detectan y miden la variación y la relacionan con cambios de humedad del suelo (LabFerrer, 2017).

Para la conversión de las mediciones de voltaje captadas por las sondas a contenido volumétrico de agua, el fabricante facilita una ecuación de calibración genérica. Esta función calibración es válida para cualquier excitación del sensor entre 3 y 15 DC (METER Group, 2017):

$$VWC = (2{,}97x10^{-9})(mV^3) - (7{,}37x10^{-6})(mV^2) + (6{,}69x10^{-3})(mV) - 1{,}92$$

Aun así, como es bien conocido, indica que, debido a la complejidad de algunos tipos de suelos, como, por ejemplo, suelos muy compactados, suelos de muy baja densidad aparente o suelos con un contenido de materia orgánica anormalmente alto, puede ser necesaria una calibración específica.

3.1.1.2 Conexión

Para facilitar la conexión con los registradores de datos o *Data Loggers*, los sensores 10HS vienen con cuatro conexiones estéreo de 3.5 mm (*Figura 4*).

Figura 4. Conector estéreo. (Fuente: METER Group, 2017)

3.1.1.3 Características técnicas

Las características técnicas de estas sondas de humedad se resumen a continuación en la *Tabla 1*:

Tabla 1. Características técnicas sondas de humedad Decagon HS 10.

Características técnicas	
Precisión:	± 3% VWC y ± 2% VWC con calibración específica
Voltaje	3 – 15 VDC @ 12- 15 mA
Frecuencia del oscilador	70 MHz
Volumen de influencia	1l. intervalo de VWC 0-57%
Señal de salida	Voltaje
Tamaño	14,5 x 3,3 x 0,7 cm

3.1.2 Características generales

El hardware empleado para la gestión y envío de datos de humedad es el microcontrolador SAMD21G18A (*Figura 5*) al que se le ha añadido una tarjeta GPRS.

Figura 5. Hardware SparkFun con SAMD21. (Fuente: SparkFun, 2018).

En relación a la alimentación, el dispositivo soporta entre 9 y 26V y cuenta con un corte de alimentación en todos los periféricos del sistema para conseguir un consumo en reposo menor de 15 uA.

Aunque actualmente no se utiliza tiene integrada una comunicación radio Lora a 868MHz y 433Mhz, además de la mencionada comunicación GPRS.

Respecto a las entradas habilitadas, dispone de:

- 4 entradas analógicas de voltaje.
- 1 entrada Analógica de 4-20 mA con voltaje de alimentación entre 9-30V.
- 1 entrada RS485 halfduplex.
- 1 entrada digital de pulsos.

Las mediciones y envíos de datos son configurables y ahora miden cada 30 minutos y mandan la información cada 4 horas.

Finalmente, la caja es estanca (IP65) y tiene unas medidas aproximadas de 12x12x9cm. No se incluye información de las conexiones entre los distintos elementos por tratarse de un tema confidencial bajo secreto tecnológico.

3.1.3 Formato de datos

El formato de envío de los datos a la plataforma se resume en la *Tabla 2*:

Tabla 2. Formato de envío de datos de humedad.

created_at	entry_id	field1	field2	field3	field4	field5	field6	field7	field8
2018-06-13 17:25:06 UTC	1	880	888	892	888	13902			
2018-06-13 17:54:07 UTC	2	880	892	896	888	13842			
2018-06-13 18:24:07 UTC	3	880	892	892	888	13818			

Los datos se envían divididos en 8 columnas, donde la primera es la fecha de medición, la segunda el identificador y de la tercera a la octava variables de medición (máximas variables permitidas por ThingSpeak).

3.1.4 Ubicación de los dispositivos

La ubicación de los dispositivos se resume en la siguiente *Tabla 3:*

Tabla 3. *Ubicación de los dispositivos de IoT disponibles*

Municipio	*Polígono*	*Parcela*	*Coordenada X*	*Coordenada Y*
Miranda de Arga	1	322	601447,1	4702917,8
Enériz	2	830	602145,1	4724938,7
Enériz	2	830	602205,2	4724778,2

La parcela de Miranda de Arga ha sido cedida a la UPNA para poder hacer diferentes ensayos y probar los dispositivos. Por su parte, la parcela de Enériz es una explotación experimental propiedad del INTIA y también ha permitido instalar dos dispositivos para comprobar su validez en casos reales.

La situación geográfica de los dispositivos que conforman el sistema de medición de humedad del suelo se presenta en la *Figura 6*:

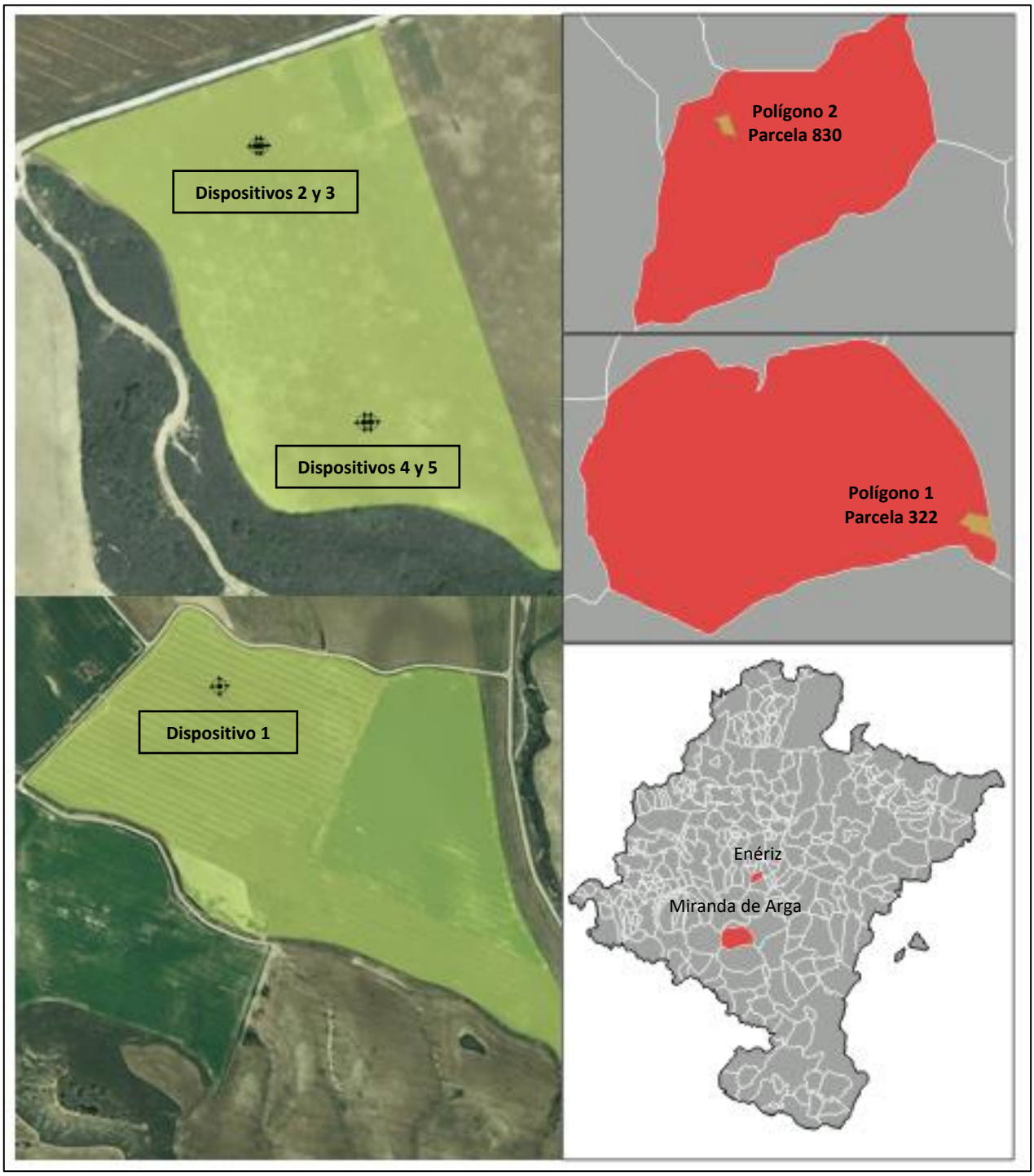

Figura 6. Emplazamiento geográfico dispositivos de medición de humedad.

3.2 ANÁLISIS DE ALTERNATIVAS

Se ha elaborado un estudio de alternativas con la finalidad de escoger los diferentes elementos que componen la aplicación de IoT de medición de humedad del suelo, entre las opciones disponibles. Tomando como referencia los objetivos descritos en el apartado 2, las tecnologías analizados son:

- Plataformas IoT.
- Bases de datos.
- Tecnologías GIS Web.

El estudio de alternativas ha estado fuertemente influenciado por la situación de partida, ya que, al partir de unos elementos ya diseñados, no todos los componentes han podido ser analizados para su elección. Este es el caso por ejemplo los micro-procesadores de los dispositivos de medición y envío de datos o el servidor para subir la información que se dispone.

En la siguiente *Figura 7* se resumen las opciones a ser valoradas para cada uno de los elementos de la aplicación:

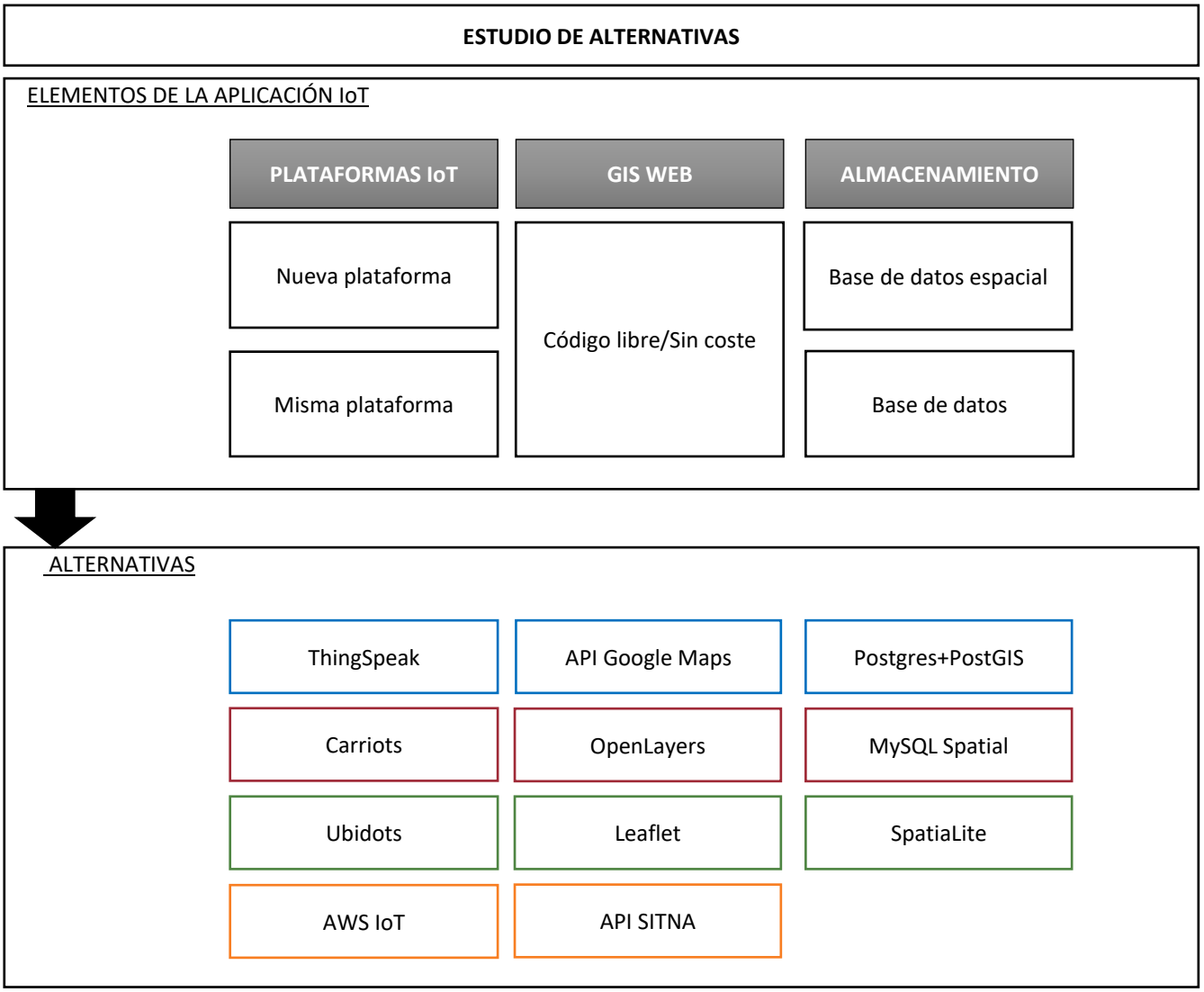

Figura 7. Elementos aplicación valorados en el análisis de alternativas.

Para determinar qué herramienta se integra mejor en la aplicación, se ha realizado un estado del arte de los productos existentes que se desarrollan en los siguientes apartados. Posteriormente se han valorado y puntuado en función de tres criterios: económico, documentación existente y calidad del trabajo.

- **Documentación existente**

Para evaluar este criterio de ponderación se ha tenido en cuenta la documentación o SDK (del inglés, Software Development Kit) disponible, a fin de garantizar la mejora y continuidad en el tiempo de la herramienta. A este criterio se le ha asignado un peso del 30%.

- **Económico**

Respecto del criterio económico, se ha valorado el coste de implementar la herramienta a la aplicación. A este criterio se le ha asignado un peso del 40%.

- **Tiempo**

Con el criterio de ponderación de tiempo se pretenden valorar el tiempo que requiere la instalación de la herramienta a la aplicación, tanto las funciones básicas como funcionalidades más avanzadas. A este criterio se le ha asignado un peso del 30%.

3.3 PLATAFORMAS DE IOT

A pesar de que los dispositivos IoT de Embeblue están diseñados para conectarse y transferir los datos a la plataforma IoT ThingSpeak, en la fase actual de prototipado, se ha creído conveniente realizar un breve estudio de mercado sobre las diferentes opciones disponibles en la actualidad, con el objetivo de comprobar y valorar la existencia de plataformas más adecuadas para el caso objeto de estudio y, además, ampliar conocimientos sobre este tipo de plataformas.

Las plataformas evaluadas son:

- Carriots.
- Ubidots.
- Amazon Web Services IoT.
- ThingSpeak.

3.3.1 Carriots

Carriots (*Figura 8*) es una empresa española orientada a proyectos IoT y máquina a máquina (M2M). Se basa fundamentalmente en ofrecer servicios en la nube, PaaS (Platform as a Service) y, a pesar de que no se trata de una plataforma de código abierto, es posible registrar hasta un máximo de 2 dispositivos de forma gratuita (Carriots, 2018).

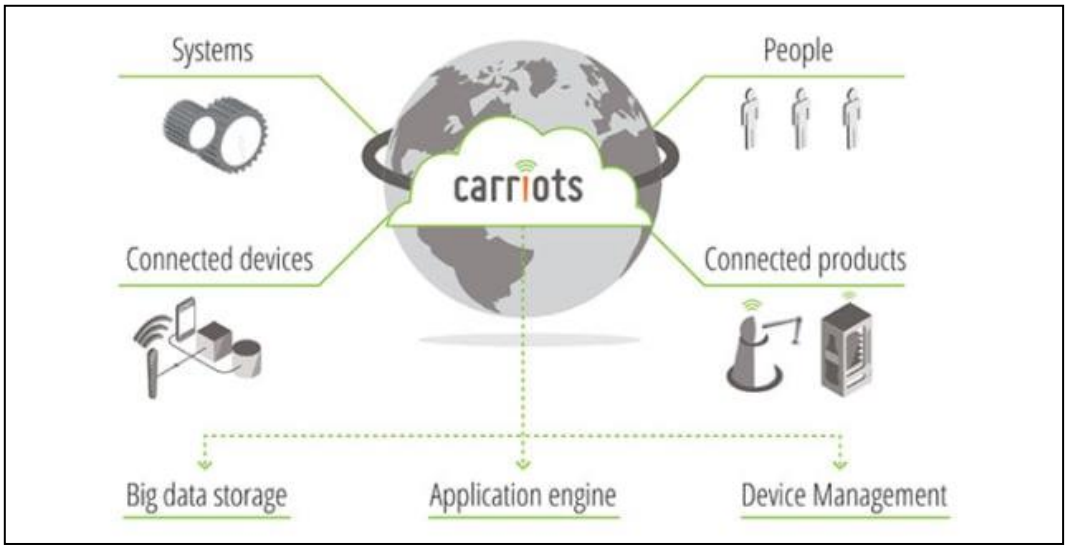

Figura 8. Esquema plataforma Carriots. (Fuente: Carriots, 2018).

3.3.1.1 Características principales

API y documentación

Carriots ofrece una API bien documentada, pero con pocos ejemplos de usos, que puede consultarse en su propia página web (https://www.carriots.com/documentation/api) o descargarse desde Github.

Se trata de una API basada en el protocolo de intercambio y manipulación de datos en los servicios de internet REST (Representational State Transfer) que es ampliamente utilizado en la actualidad. La información se puede enviar en fomato XML o JSON y se almacena en una base de datos NoSQL (Garrido, 2017).

Hardware

Los hardware compatibles con la plataforma son (Garrido, 2017):

- Arduino
- Raspberry Pi
- Beagle Bone
- Fez Cerbuino
- Cubie Board
- TST Gate
- TST Mote
- CloudGate
- Nanode
- Electric IMP
- TST Light
- Mobile Devices C4Max
- Mobile Devices C4EVO
- Mobile Device OBD Ongle
- Electric IMP April Development Board
- We500
- Satel Dataloggers DL170,DL171 y OWA31IETH

Precios

La plataforma ofrece un plan gratuito que incluye:

- Dispositivos: máximo de 2 dispositivos.
- API Keys: 4 claves para conectar con su API.
- Número de envíos: 500 envíos de datos al día y 10 por minuto.
- Tamaño de los mensajes: máximo de 5KB por mensaje y 5000KB al día.
- Almacenamiento de datos: 3 meses.

Por otro lado, Carriots dispone de otros dos planes más:

- Corporate: tiene un coste de 2€ por dispositivo (con un mínimo de 11 dispositivos).
- Lite: tiene un coste de 0,50€ por dispositivo (con un mínimo de 100 dispositivos).

Conclusiones

Se trata de una plataforma que está pensada para dar soluciones en la nube y orientada a aplicaciones de sectores muy diversos.

En principio no parece que esté enfocada a prototipos, como es el cado objeto de estudio, ya que en su versión gratuita únicamente es posible dar de alta dos dispositivos.

3.3.2 Ubidots

Es una plataforma IoT (*Figura 9*) para la gestión de datos en la nube que ofrece la posibilidad de almacenar e interpretar información captada por sensores en tiempo real. Tampoco es de código abierto pero ofrece una licencia de estudiante en el que puedes dar de alta hasta 20 dispositivos (Ubidots, 2018).

Figura 9. Logotipo Ubidots.(Fuente: Ubidots, 2018).

3.3.2.1 Características principales

API y documentación

Dispone de una API que permite enviar, leer, editar y borrar la información almacenada mediante los métodos estándares de HTTP (GET, POST, PUT y DELETE). Asimismo, cuenta con numerosa documentación accesible desde su página web (https://help.ubidots.com/),con ejemplos de conexión de distintos hardware, como Arduino o Raspberry Pi.

También tiene una completa colección de librerías para realizar llamadas a la API. Los lenguajes disponibles para estas librerías son: Python, Java, C, PHP, Node, Ruby.

La plataforma admite envíos de información en formatos XML y JSON, que es enviada mediante peticiones REST utilizando el protocolo HTTP (Rodrigo Martínez, 2017).

Hardware

Los hardware compatibles con la plataforma son:

- Arduino.
- Raspberry Pi.
- Android.
- Spark.io.
- Tessel.
- Otros dispositivos que envíen peticiones HTTP.

Precios

La plataforma ofrece una licencia estudiantil que incluye:

- Dispositivos: máximo de 20 dispositivos
- Número de envíos: 60 por minuto
- Almacenamiento de datos: 3 meses.

Por otro lado, Carriots dispone de otros cuatro planes más:

- Developer: tiene un coste de 20$ al mes que incluye 10 dispositivos.
- IoT Lab: tiene un coste de 99$ al mes que incluye 60 dispositivos.
- Developer: tiene un coste de 499$ al mes que incluye 400 dispositivos.
- Developer: tiene un coste de 2.499$ al mes que incluye 5000 dispositivos.

Conclusiones

Se trata de una plataforma que parece más enfocada que Carriots a prototipos ya que cuenta con una licencia estudiantil bastante interesante y abundante documentación con ejemplos y tutoriales muy útiles.

3.3.3 Amazon Web Services IoT

Es una plataforma que permite conectar diferentes dispositivos a la nube de Amazon Web Services y recopilar datos de origen, ejecutar análisis sofisticados y/o ejecutar acciones en tiempo real. Dispone de un número elevado de servicios diferentes (AWS Lambda, Amazon Kinesis, Amazon S3, etc.) que se adaptan a las necesidades de cada caso (Amazon, 2018).

En la siguiente *Figura 10* se muestra el esquema de esta plataforma:

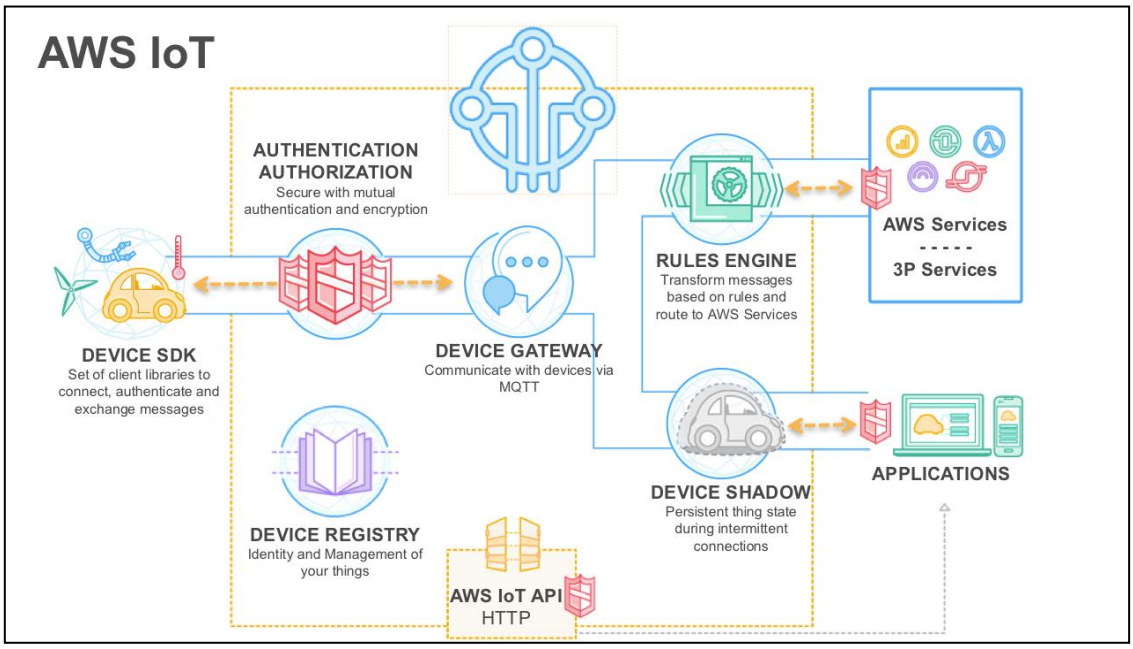

Figura 10. Esquema plataforma AWS IoT. (Fuente Amazon, 2018).

3.3.3.1 Características principales

API y documentación

Esta plataforma dispone de una documentación muy extensa para cada uno de los servicios que oferta, ejemplos incluidos, en su página web: (https://aws.amazon.com/es/documentation/?nc2=h_ql_d&awsm=ql-5).

Hardware

Los hardware compatibles con la plataforma son:

- Arduino .
- Raspberry Pi.
- Android.
- Spark.io.
- Tessel.
- Otros dispositivo que envíen peticiones HTTP.

Precios

La plataforma de Amazon cobra en función del número de mensajes recibidos y proporciona gratuitamente otros servicios de AWS, tales como: Amazon S3, Amazon DynamoDB, AWS Lambda, Amazon Kinesis, Amazon SNS o Amazon SQS.

Conclusiones

Se trata de una plataforma que está pensada para dar soluciones en la nube y orientada a aplicaciones de sectores muy diversos. Ofrece múltiples servicios y es necesario un amplio estudio de esta plataforma para poder sacar todo el rendimiento.

En principio no parece que esté enfocada a prototipos, como es el cado objeto de estudio, ya que no dispone de versión gratuita o de prueba.

3.3.4 ThingSpeak

Plataforma de código abierto que permite la recolección y almacenamiento de datos registrados por dispositivos IoT y el desarrollo de *pluggins* o aplicaciones orientadas a aprovechar esos datos. El código está disponible en GitHub y cuenta con una gran comunidad de desarrolladores.

Otro aspecto destacable es que el análisis de datos y la visualización de los mismos es mediante MATLAB, desarrollado por la misma empresa matriz que la plataforma, The MathWorks (ThingSpeak, 2018).

En la siguiente *Figura 11* se muestra el esquema de esta plataforma:

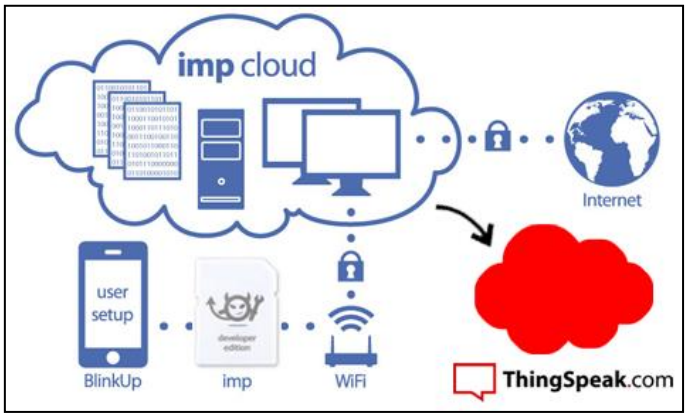

Figura 11. Esquema plataforma ThingSpeak.

3.3.4.1 Características principales

API y documentación

ThigSpeak dispone de una API disponible en GitHub para su descarga en un servidor propio. Como se ha mencionado anteriormente, es de código abierto, por lo que es posible modificar su código fuente original. Cuenta con una comunidad de usuarios y desarrolladores muy amplia y existe abundante información, con ejemplos, en su página web (https://community.thingspeak.com/).

En relación a los protocolos de comunicación, la API emplea los protocolos HTTP y MQTT.

Hardware

Los hardware compatibles con la plataforma son:

- Arduino.
- Raspberry Pi.
- IoBridge / RealTime.io.
- Electric Imp o Móbiles / Aplicaciones web.
- Otros dispositivos que envíen peticiones HTTP.

Precios

Los precios de esta plataforma se resumen en la siguiente *Tabla 4*:

Tabla 4. Precios plataforma ThingSpeak.

Plan	Gratuito	Estudiante	Hogar	Académico	Estándar
Precio		79$ anuales	95$ anuales	250$ anuales	650$ anuales
Escalable	No	Si	Si	Si	Si
Num. Mensajes	3M / Año	33M / Año	33M / Año	33M / Año	33M / Año
Tiempo entre mensajes	1 cada 15 segundos	1 cada segundo	1 cada segundo	1 cada segundo	1 cada segundo
Canales	3	Ilimitados	Ilimitados	Ilimitados	Ilimitados

Conclusiones

ThingSpeak es una plataforma muy interesante para iniciarse y realizar prototipos en IoT. Tiene disponible una API muy bien documentada y una comunidad de desarrolladores activa. Asimismo, cuenta con amplia información referente a su integración con el hardware Arduino.

3.3.5 Tabla comparativa IoT

En la *Tabla 5* se resumen las principales características de las plataformas IoT estudiadas:

Tabla 5. Comparativa platafomas IoT analizadas.

Plataformas	Hardware	Protocolos	Lenguajes	Ventajas	Desventajas
Carriots	Arduino Raspberry Electronic Imp Beagle TST Industrial	MQTT HTTP	Groovy	Múltiples ámbitos de aplicación	Poca documentación No es posible elegir como se almacena la información dentro de la plataforma
Ubidots	Arduino Raspberry Pi. Android. Spark.io. Tessel. Otros dispositivos HTTP	MQTT HTTP	Python, Java, C, PHP, Node.js, Ruby	Numerosas librerías. Abundante documentación	Sistema de precios por créditos confuso
AWS IoT	Arduino Raspberry Pi Spark.io Otros dispositivos HTTP.	MQTT HTTP	C, JavaScript, Java, Python, iOS, Android, C++	Abundantes servicios	Requiere mucha investigación para su correcto uso. No dispone de licencia gratuita
ThingSpeak	Arduino, Spark, Raspberry Pi, Electronic Imp	MQTT HTTP	MATLAB	Código abierto Abundante información	No es posible gestionar el modo de almacenamiento de los datos

3.4 BASES DE DATOS

Los sistemas gestores de bases de datos son la herramienta más adecuada para almacenar los datos en un sistema de información debido a sus características de seguridad, recuperación ante fallos, gestión centralizada, estandarización del lenguaje de consulta y funcionalidad avanzada (Víctor Olaya, 2011).

Siguiendo con los elementos de la aplicación IoT, se ha recopilado información sobre las bases de datos relaciones espaciales más utilizadas en este ámbito, donde han destacado PostgreSQL con su extensión espacial PostGIS y MySQL (*Figura 12*).

Cabe destacar que se ha descartado directamente la opción de usar SpatiaLite debido a que funciona bien en entornos de usuario único, pero no es tan robusta a la hora de manejar concurrentes múltiples conexiones, como ocurre a menudo en un entorno web (Swain *et al.*, 2015).

Figura 12. Logotipos PostgreSQL y MySQL.

3.4.1 PostgreSQL y PostGIS

Es un sistema de base de datos relacional orientada a objetos, que permite el almacenamiento de información mediante tablas. Es un software de código libre y multiplataforma, ya que trabaja y almacena información en varios formatos, entre ellos imágenes y vídeo (Flores Medina *et al.*, 2015).

PostGIS es una extensión que ofrece soporte para el almacenamiento y procesamiento de objetos geográficos. También es de código libre, multiplataforma e implementa datos espaciales que se encuentran definidos en el open geoespacial consortium (OGC).

Es la opción más utilizada de debido, principalmente, a las múltiples herramientas de análisis y geoprocesos y formatos (vectorial y raster) que soporta (Boluwade y Ferdinand, 2011).

3.4.2 MySQL Spatial

MySQL es un sistema de gestión de base de datos relacional basado en el lenguaje de consulta estructurado (SQL) y de código abierto (mySQL) de la compañía Oracle.

MySQL spatial soporta tipos de datos espaciales sin necesidad de agregar algún modulo adicional como es en el caso de PostGIS. Esta base de datos es la más popular y a menudo se ha utilizado en aplicaciones web. Sin embargo, dispone de menos funcionales que PostGIS para datos espaciales (Swain *et al.*, 2015).

3.4.3 Tabla comparativa de las principales características

En la *Tabla 6* se resumen las principales características de las bases de datos estudiadas:

Tabla 6. Comparativa bases de datos analizadas.

Característica	**PostgreSQL+PostGIS**	**MySQL Spatial**
Licencia	Código abierto, licencia PostgreSQL License, similar a BSD o MIT.	Código abierto, pero propiedad de Oracle y ofrece versiones comerciales. Licencia GNU.
ACID (atomicidad, consistencia, aislamiento, durabilidad)	Cumple completamente con ACID	Cumple parcialmente con ACID
Cumplimiento SQL	Cumple casi completamente	Cumple parcialmente (no soporta restricciones Check).
Concurrencia	Implementación de MVCC (control de concurrencia multiversión) soporta múltiples peticiones sin bloquear lecturas	Implementación de MVCC solo en algunas versiones (InnoDB)
Soporte geoespacial	Funciones geospaciales complejas y abundantes formatos espaciales	Funciones geoespaciales más simples y menos formatos espaciales
Funciones analíticas	Sí	No
Lenguajes de programación	Ruby, Perl, Python, TCL, PL/pgSQL, SQL, JavaScript, etc.	SQL:2003

3.5 TECNOLOGÍAS GIS WEB

Tal y como se ha definido en el apartado de objetivos, está previsto que la aplicación IoT incluya un visor de datos de los dispositivos de medición de humedad al que, en un futuro, se le puedan ir añadiendo funcionalidades GIS.

De las múltiples opciones para la creación de aplicaciones web que emplean información georreferenciada existentes, las opciones estudiadas han sido (*Figura 13*):

- Google Maps API.
- Leaflet.
- OpenLayers.
- API SINTA.

Figura 13. Logotipos tecnologías GIS Web analizadas.

Se ha optado por el análisis de estas herramientas por contar con abundante documentación, tener una comunidad de desarrolladores activa y, salvo la API SITNA, ser las opciones más habituales para este tipo de aplicaciones (Swain *et al.*, 2015).

3.5.1 API de Google Maps

Esta API, permite a los desarrolladores utilizar los mapas Google Maps en páginas web empleando el código de programación JavaScript.

Aunque su uso es gratuito sus productos están sometidos a normas y precios. Es decir, a pesar de que la librería es gratuita y la mayoría de los sitios web y las aplicaciones pueden usar la API para JavaScript estándar sin cargo, si se superan cierto número de visitas o se requieren algunos servicios adicionales es preciso pagar por ellos. Google ofrece 200$ en créditos de uso al mes lo que, según la página web, son suficientes para cubrir las necesidades de la mayoría de usuarios(Google, 2018).

Es una de las herramientas web más utilizadas, pero el que no sea de libre, dificulta cualquier tipo de cambio o modificación.

Otras características de importancia son:

- Formatos de datos compatibles: KML/KMZ, GeoRSS y GeoJSON.
- Última versión estable: 3.34.
- Documentación: cuenta con una galería de ejemplos, guías para el aprendizaje y una descripción de la librería muy completa. Asimismo, existe un foro en 'Stack Overflow' y otro en 'Google Groups'.

3.5.2 Leaflet

Leaflet es la librería de código abierto de JavaScript muy ligera (38 KB) para mapas interactivos y optimizados para dispositivos móviles. Al contrario que la API de Google, no tiene limitaciones o restricciones en su uso (Leaflet, 2018).

Dispone de una API sencilla y de rápido aprendizaje que se centra en un conjunto básico de funciones. Aún así, esta diseñada para aumentar dichas funcionalidades mediante la implementación plugin de terceros (más de 100).

Otras características de importancia son:

- Formatos de datos compatibles: KML, GeoJSON, CSV, WKT, TopoJSON, GPX.
- Última versión: 1.34 (agosto 2018).

- Documentación: cuenta con tutoriales, guías para el aprendizaje y una descripción de la API muy completa. Asimismo, en la propia página web índica que puedes consultar y apoyar el proyecto a través de sus foros de Stack Overflow, GIS Stack Exchange y GitHub issues.

3.5.3 OpenLayers

OpenLayers es una librería JavaScript de alto rendimiento que cuenta con abundantes funcionalidades para crear mapas interactivos en la web. Es un proyecto del OSGeo, de código abierto y que se publicó bajo la licencia BSD de 2 cláusulas (FreeBSD) (OpenLayers, 2018).

Es una librería que trabaja cerca de las funciones lo que, en principio, dificulta el aprendizaje, pero permite controlar y trabajar con la mayor parte de funciones espaciales existentes. De hecho, otras librerías, como Leaflet, se apoyan en OpenLayers.

Otras características de importancia son:

- Formatos de datos compatibles: KML, GeoJSON, CSV, WKT, TopoJSON, GPX, GML, XML, SHP.
- Última versión: 5.2.0 (agosto 2018).
- Documentación: cuenta con una galería de ejemplos, guías para el aprendizaje y una descripción de la librería muy completa en su página. Asimismo, existe un SDK que se instala con OpenGeo Suite y un foro activo en la página de Stack Overflow.

3.5.4 API SITNA

Es una API JavaScript que permite incluir en páginas y aplicaciones web un visor de mapas interactivo y así representar información georreferenciada. A pesar de que se ha desarrollado para su uso en aplicaciones web de Gobierno de Navarra, según se indica en su página web, puede ser utilizado por cualquier usuario y organización en sus páginas web. La API está basada en diversas bibliotecas JavaScript de terceros, pero está completamente autocontenida y simplemente cargando en la página el script de la API se cargan dinámicamente los recursos que necesita (SITNA, 2018b).

Se trata de una API que con muy pocas líneas de código es posible implementar un gran número de funcionalidades y de aspecto muy profesional.

Actualmente está orientada a aplicaciones ubicadas en Navarra, como es el caso de este TFM, pero está previsto la extensión de sus funcionalidades. Además, esta carencia puede ser suplida mediante la carga de servicios WMS/WFS de otras comunidades.

Otras características de importancia son:

- Formatos de datos compatibles: KML, GeoJSON, WKT, GML, XML, SHP.
- Última versión: 1.5.0.
- Documentación: cuenta con una guía de usuario y otra de técnico (SITNA, 2018a) con abundantes ejemplos de uso. Asimismo, en su página web está disponible documentación propia de la API.

3.5.5 Tabla comparativa

En la *Tabla 7* se resumen las principales características de las tecnologías GIS Web estudiadas:

Tabla 7. Comparativa tecnologías GIS Web analizadas.

Herramienta GIS Web	Documentación	Servicios web OGC	Formatos datos espaciales	Precios	Ventajas	Desventajas
API Google Maps	Completa	WMS, WMTS, WFS	KML/KMZ, GeoJSON y GeoRSS	Coste sujeto al uso.	Herramientas propias de Google (Geocoding, Places, Elevation, Distance, Roads, Time Zone, etc.)	No es de código libre. Coste en función al uso.
Leaflet	Completa	WMS, WFS, WFS-T, WMTS, TMS	GeoJSON, KML, CSV, WKT, TopoJSON, GPX	Sin coste	Apariencia atractiva Abundantes pluggins Código libre	Necesidad de conocer los plugins para aplicar funcionalidades
OpenLayers	Completa	WMS, WFS, WFS-T, WCS, WMTS, TMS, WPS, SOS, CSW	EsriJSON, GML, XML, GPX, GeoJSON, KML, MVT, TopoJSON, ESRI Shapefile	Sin coste	Muchas funcionalidades Código libre	Aprendizaje más costoso
API SITNA	Completa, pero no cuenta con foros activos	WMS, WMTS, WFS	KML, GeoJSON, WKT, GML, XML, SHP	Sin coste	Posibilidad de incorporar un mapa muy completo de manera sencilla.	Principalmente orientada a Navarra.

3.6 SERVIDOR

El servidor disponible consta de un domino que tiene el hosting activado en *hydrotecna.com*. Es posible añadir contenido estático en diferentes formatos (imágenes, HTML, CSV, etc.) para su visualización y descarga.

Sin embargo, no permite la ejecución de programas como Python o la base de datos, lo que limita a la hora de diseñar una aplicación dinámica y, también, al seleccionar los elementos de la aplicación.

La dirección principal de la aplicación es http://hydrotecna.com/TFM/.

Está prevista la instalación de un nuevo servidor en el que se puedan ejecutar los principales programas de procesado y almacenamiento de datos, que permitan incluir funciones dinámicas del lado del cliente, pero no es objeto de estudio de este TFM.

3.7 PÁGINA WEB

Para el diseño de la página web, se ha decidido emplear la plantilla *Prism* (*Figura 14*) disponible en https://templated.co/.

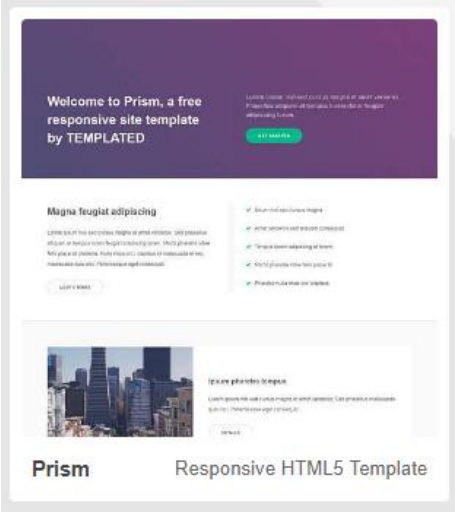

Figura 14. Plantilla web Prism. (Fuente: Template, 2018)

Template es un repositorio web que incluye colecciones de CSS, HTML5 y plantillas distribuidas todas ellas bajo la licencia Creative Commons Attribution 3.0, que permite su uso para proyectos personales y comercial siempre y cuando agregues su autoría en un lugar visible de la web (Template, 2018).

3.8 CÓDIGOS DE PROGRAMACIÓN

3.8.1 Python

Se ha utilizado el código de programación Python para realizar la rutina de descarga, procesado de los datos, almacenamiento de los mismos en la base de datos y actualizar la página web.

Python es un lenguaje de programación de código libre, orientado a objetos y creciente expansión derivada de su simplicidad, versatilidad y rapidez de desarrollo (Python, 2018).

3.8.2 JavaScript

JavaScript es un lenguaje ligero e interpretado, orientado a objetos y de código privativo (marca registrada de Oracle). Suele utilizarse en páginas web pero también se usa en muchos entornos sin navegador (Mozilla.org, 2018).

4. RESULTADOS Y DISCUSIÓN

El presente capítulo incluye los resultados obtenidos del análisis de alternativas realizado para la elección de las herramientas del sistema, así como como la descripción detallada de la solución final adoptada.

4.1 ANÁLISIS DE ALTERNATIVAS

4.1.1 Resumen de las alternativas y solución adoptada

En la siguiente *Figura 15* se resumen las alternativas valoradas y la solución final escogida:

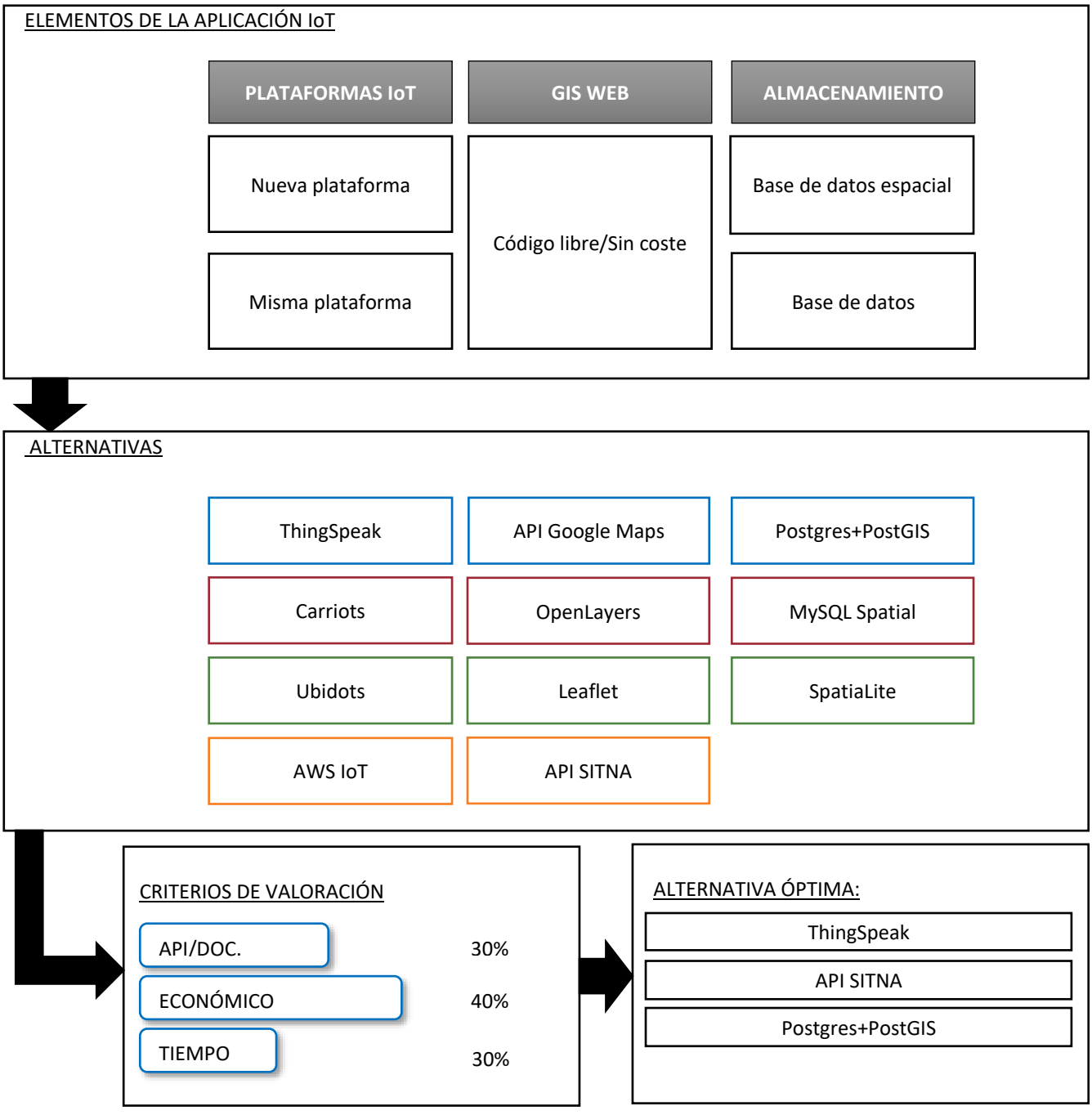

Figura 15. Resumen análisis de alternativas y solución final adoptada.

4.1.2 Valoración de las alternativas

Se han valorado del 1 al 5 las distintas alternativas en función de los criterios de valoración, siendo 1 la peor puntuación posible y 5 la mejor. En la *Tabla 8* se recogen las puntuaciones:

Tabla 8. Valoración de alternativas del proyecto.

Elementos	Herramientas	API/DOCUMENTACIÓN 30%	ECONÓMICO 40%	TIEMPO 30%	PUNTUACIÓN
PLATAFORMAS IoT	Carriots	3,5	2	3	2,75
	AWS IoT	4	2	3	2,9
	Ubidots	4,5	4	4	4,15
	ThingSpeak	4,5	5	5	**4,85**
GIS WEB	API Google	4,5	2	3	3,05
	OpenLayers	4,5	5	3,5	4,4
	Leaflet	4,5	5	3,5	4,4
	API SITNA	3,5	5	5	**4,55**
ALMACENAMIENTO	PostGIS	4,5	5	5	**4,85**
	MySQL Spatial	4,5	5	4	4,55
	SpatiaLite	3,5	5	3	3,95

Como se observa en la *Tabla 8*, relativa a las plataformas web, la opción que obtiene la puntuación más alta con 4,85 puntos es ThingSpeak, es decir, mantener la que se utiliza en la actualidad, mientras que la de Carriots es la peor valorada con un 2,75 sobre 5.

Por otro lado, respecto a las tecnologías GIS Web la ganadora es la API SITNA, con una puntuación total de 4,55 puntos. Le siguen de cerca, LeafLet y OpenLayers con 4,4 puntos y finalmente, en última posición se encuentra la API de Google Maps.

La base de datos PostgreSQL con su extensión espacial PostGIS es la opción para e l almacenamiento que ha obtenido la mayor puntuación con 4,85. Seguidamente se encuentra MySQL Spatial con 4,55 y, por último, SpatiaLite con 3,95.

4.1.3 Análisis detallado sobre los elementos de la aplicación seleccionados

4.1.3.1 Plataformas IoT

De las cuatro plataformas analizadas, dos son de código libre (Ubidots y ThingSpeak) y dos son de código privativo (Carriots y AWS IoT). Este ha sido uno de los factores más determinantes a la hora de valorar las plataformas. La dos últimas disponen de mayores funcionalidades y servicios, pero se observa claramente que están orientadas a proyectos de mayor envergadura, limitando los dispositivos que se pueden conectar a las mismas. Como el objetivo principal no es el de utilizar las herramientas de las plataformas, si no el de almacenar y gestionar los datos que envían los *dataloggers*, se ha descartado el uso de estas dos plataformas.

ThingSpeak y Ubidots son muy similares en cuanto a características y funcionamiento. Ambas funcionan correctamente con prototipos, cuentan con una versión gratuita bastante completa, abundante documentación con ejemplos y tutoriales y no es posible gestionar la manera que tienen de almacenar los datos.

Al ser tan similares y no encontrar ninguna clara ventaja de Ubidots sobre ThingSpeak, no se ha creído conveniente hacer el cambio de plataformas, ya que eso supondría un incremento económico en la reprogramación del firmware de los dispositivos y también de tiempo de migración.

4.1.3.2 GIS Web

De nuevo, en este caso la licencia de uso ha jugado un papel importante a la hora de decantarse por una tecnología u otra. En principio, las condiciones gratuitas del uso de la API de Google Maps serían suficientes para cubrir las necesidades del proyecto, pero limitaría una futura expansión del mismo.

De esta manera, se ha comprobado que, salvo en determinadas circunstancias muy concretas, las librerías Open Source son una alternativa igual o mejor para publicar los mapas en la web, ya que se pueden obtener resultados igual de profesionales sin restricciones de uso.

Entre este tipo de librerías, Leaflet y OpenLayers son dos de las más populares. Según la bibliografía consultada la primera es más sencilla, se base en el uso *pluggins* para ampliar sus funcionalidades y tiene muy desarrollada la parte estética. Por otro lado, la segunda es más completa y está más enfocada a aplicaciones con funcionalidades GIS más complejas.

La tercera opción y que finalmente ha sido la herramienta seleccionada es la API SITNA. La principal razón por la que ha obtenido una mayor puntuación que Leaflet y OpenLayers se debe a que, en muy pocas líneas de programación (menor tiempo) se consigue un visor muy completo y de aspecto muy elaborado. A pesar de que no es tan modificable como las anteriores librerías, cumple con los requisitos planteados para el visor.

4.1.3.3 Base de datos

La mayoría de bibliografía consultada coincide en que, respectos a capacidades GIS, PostGIS es una base de datos más completa y estable que otras alternativas. Si el proyecto a realizar está orientado a geoprocesos, complejos, se recomienda el uso de PostGIS. MySQL hasta ahora ha sido la más utilizada para proyectos basados en la web, pero de complejidad o aplicaciones limitadas.

A pesar de que probablemente se puedan emplear ambas bases de datos para el desarrollo del proyecto, en el caso de querer expandir las aplicaciones de la web (sobre todo relacionadas con GIS), es ampliamente recomendable el uso de PostGIS.

Teniendo en cuenta que, en un futuro se pretende implementar el uso de imágenes satelitales (raster), se considera más apropiado el uso de PostGIS, debido a que es capaz de procesarlas y almacenarlas. Según bibliografía consultada, MySQL no tiene tan desarrollada esta capacidad.

Finalmente, hay que destacar que PostGIS tiene una comunidad muy activa y abundante información, además, el hecho de que forme parte de OpenGeoSuite, facilita la unificación de sistemas y la integración de los mismos.

4.2 DESCRIPCIÓN DE LA SOLUCIÓN ADOPTADA

4.2.1 Resumen

La solución propuesta para el desarrollo de la aplicación IoT consiste en la integración de la plataforma IoT ThingSpeak, una base de datos PostgreSQL y un visor web desarrollado con API SITNA, que se conectan e interactúan a través de una tarea programado, corriendo en un servidor, programada en Python. Consta de cuatro fases principales:

1. Descarga de los datos de humedad de la plataforma IoT.
2. Procesado de los mismos.
3. Almacenamiento en la base de datos.
4. Actualizar los datos de la página web.

El esquema siguiente (*Figura 16*) muestra de forma gráfica las fases y herramientas o aplicaciones correspondientes a la metodología planteada como solución propuesta.

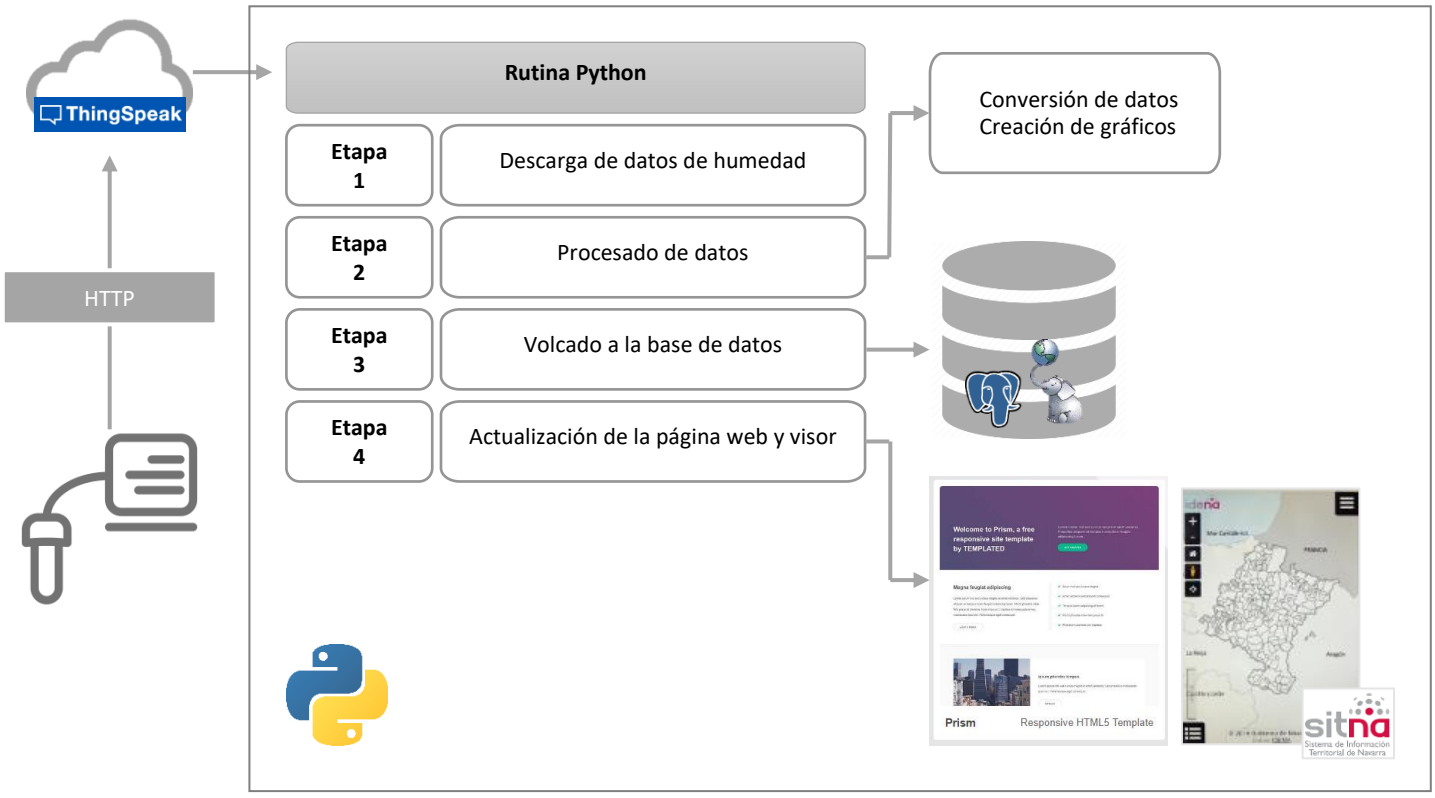

Figura 16. Esquema de la metodología propuesta.

En los siguientes apartados se describen más ampliamente cada una de las etapas principales de la metodología y sus respectivas herramientas desarrolladas.

4.2.2 Rutina de Python

La rutina programada en Python está formada por una serie de scripts y un archivo de configuración inicial, manejando de esta manera todo el proceso desde la descarga de datos de la plataforma IoT hasta la actualización de los datos de la página web (gráficos, estadísticas y visor).

Se ejecuta automáticamente (proceso en segundo plano) dos veces al día mediante las herramientas *Cron* y *Crontab* de Linux.

En la *Figura 17* se muestra un resumen de los scripts y su contenido:

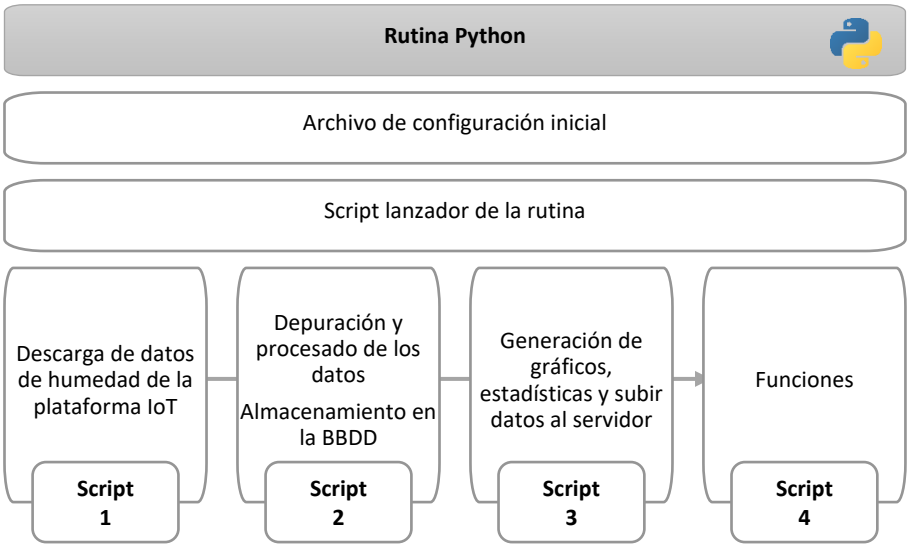

Figura 17. Resumen de los script de python desarrollados.

Tal y como se observa en la figura anterior, la aplicación de procesado se ha dividido en cuatro scripts principales y un lanzador que, a partir de los datos del fichero de configuración inicial, ejecuta secuencialmente cada script.

El código de programación de cada uno de los scipts, así como el fichero de configuración inicial puede consultarse en el *Anexo I. Códigos de programación*.

En los siguientes apartados se describen con mayor detalle cada uno de los archivos.

4.2.2.1 Archivo de configuración inicial

Se trata de un fichero de texto plano cuya extensión es *.ini*, extensión habitual de los archivos de configuración utilizados por aplicaciones de los sistemas operativos Windows.

Sigue la siguiente estructura:

```
[SECCION]
CLAVE=VALOR
CLAVE=VALOR
[OTRA_SECCION]
CLAVE=VALOR
```

Seguidamente, en la *Figura 18* se muestran las secciones y relaciones de Clave-Valor que constituyen el archivo de configuración inicial de la aplicación:

```
[configuracion]
bbdd = postgresql
labels = receptoresAgrotech
indexColumn = fecha
dropColumn =ID
ubicacionArchivos = directorios
servidor = servidor
correoErrores = correo

[postgresql]
host=localhost
database=agrotech
user=postgres
password=postgres

[receptoresAgrotech]
labels=fecha,ID,sonda1,sonda2,sonda3,sonda4,bateria

[receptoresOperativos]
id=1,2,3

[directorios]
mainPath=C:/Users/Unai/Dropbox/TFM_MUSIGT/Programacion
graphPath=/Graficos/
subGimg=SubG_idReceptor_'+str(idreceptor)+'_'+str(fechaI)+'.png
Gimg=GAcu_idReceptor_'+str(idreceptor)+'_'+str(fechaI)+'.png
htmlPath=/Web/
htmlindex=index.html

[servidor]
servidor =
usuario = h
password =  18

[correo]
from = agrotech.upna@gmail.com
to =  z@gmail.com
```

Figura 18. Archivo de configuración inicial.

Como se observa, consta de un total de 7 secciones diferentes:

- **Configuración:** índice general que indica donde se encuentran las otras secciones.
- **Postgresql:** información de la base de datos.
- **ReceptoresAgrotech:** etiquetas de cabecera para dar formato a los datos.
- **ReceptoresOperativos:** de todos los dispositivos almacenados en la base de datos aquellos que están operativos.

- **Directorios:** rutas específicas de trabajo.
- **Servidor:** datos del servidor donde se aloja la página web.
- **Correo:** datos del correo electrónico donde se mandan las incidencias ocurridas.

4.2.2.2 Script 0: lanzador de la rutina

Es el programa base a partir del cual se ejecutan los diferentes scripts y consta de las siguientes etapas (*Figura 19*):

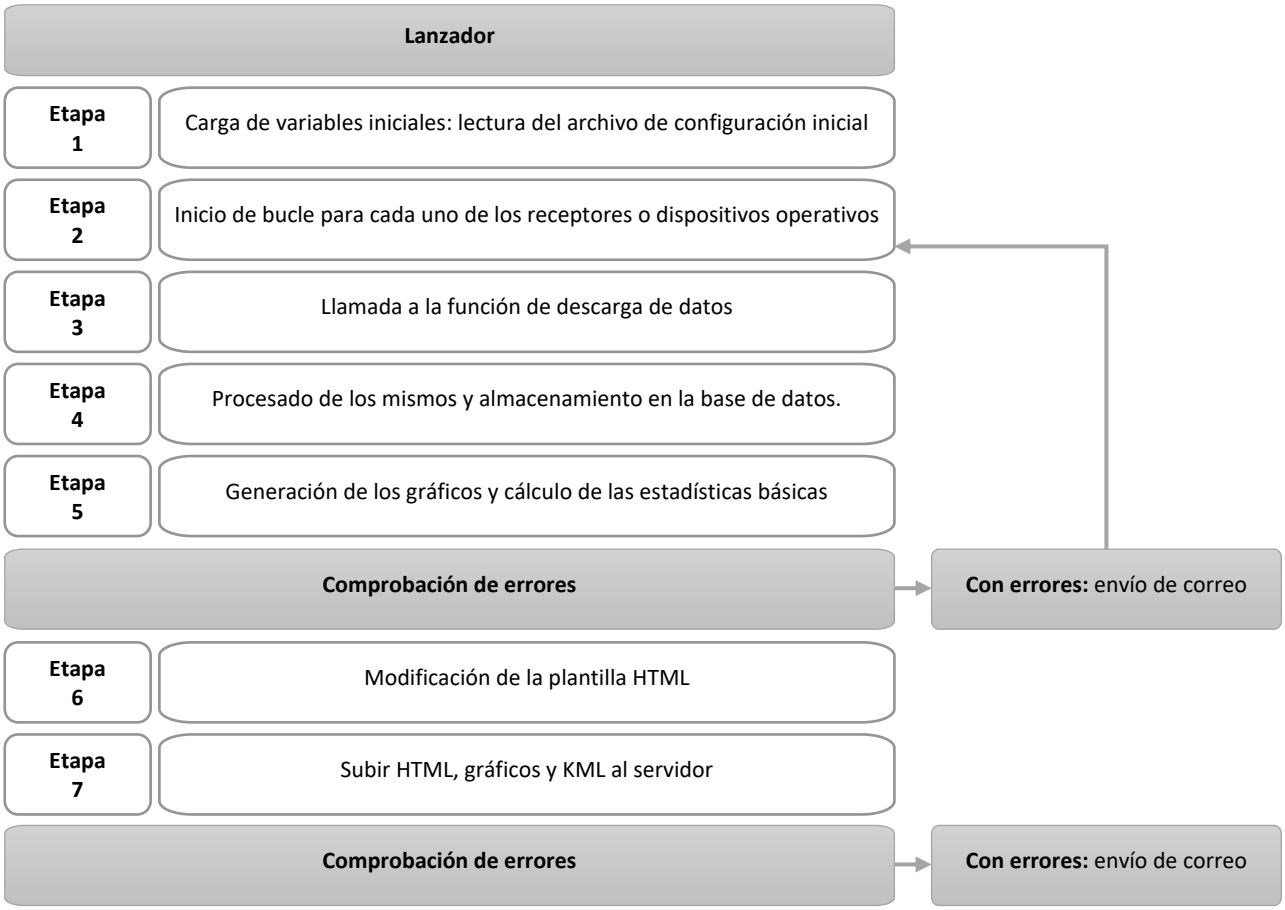

Figura 19. Etapas del script lanzador.

Como se puede observar, cuenta con siete etapas distintas que se ejecutan de manera secuencial. Se inicia con la lectura del archivo de configuración inicial, del cual adquiere la información necesaria para funcionar.

A continuación, para cada dispositivo operativo va realizando el mismo proceso que se inicia con la descarga de datos y finaliza con la actualización de la página web.

Asimismo, dispone de mecanismos de control y seguridad que envían un correo identificando el error que se ha podido producir. Un ejemplo de correo de aviso de incidentes se recoge en la *Figura 20*:

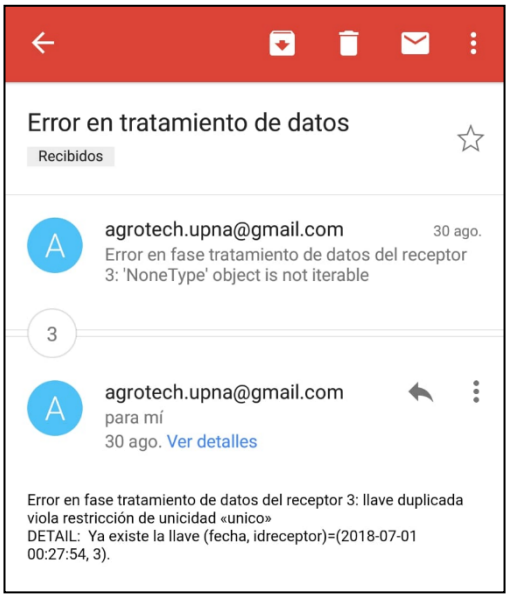

Figura 20. Correo de aviso de incidencias.

4.2.2.3 Script 1: descarga de datos

Es el script encargado de realizar la llamada de conexión a la plataforma IoT, descarga de los datos de humedad y finalmente la conversión de los mismos a un formato más manejable como es *Pandas Data Frame*. La secuencia se resume en la siguiente *Figura 21*.

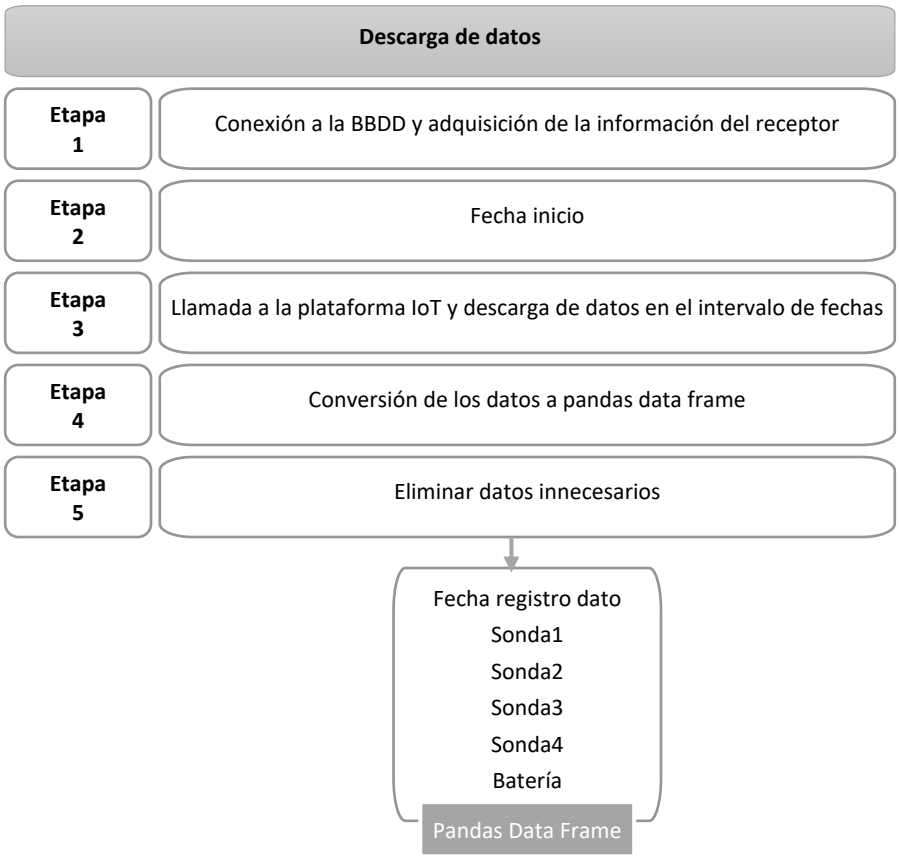

Figura 21. Etapas del script de descarga de datos.

El primer paso consiste en consultar en la propia base de datos la información referida al dispositivo que se está procesando. Esta información incluye entre otros datos, el Canal o la API key necesaria para la conexión con la plataforma ThingSpeak.

Entre los parámetros que son necesarios para la descarga de datos, se encuentra el intervalo de fechas, es decir desde cuándo y hasta cuándo se quieren descargar los datos. La fecha final (hasta cuando) es el día de ejecución del script. La fecha de inicio se adquiere de la BBDD, de una tabla que registra cual ha sido la última fecha de procesado de datos y que se actualiza cuando el volcado de la información a la BBDD ha sido satisfactorio.

Finalmente se descargan los datos, se depuran y se convierten al formato *Pandas Data Frame*.

4.2.2.4 Script 2: depuración y procesado de los datos

Una vez se disponen de los datos de humedad en el formato adecuado, este script transforma cada lectura de mV a contenido volumétrico (WVC), calcula el contenido de agua total del suelo en función de la profundidad de las sondas y vuelca los datos de humedad a la BBDD.

En la siguiente *Figura 22* se muestran las etapas de este script:

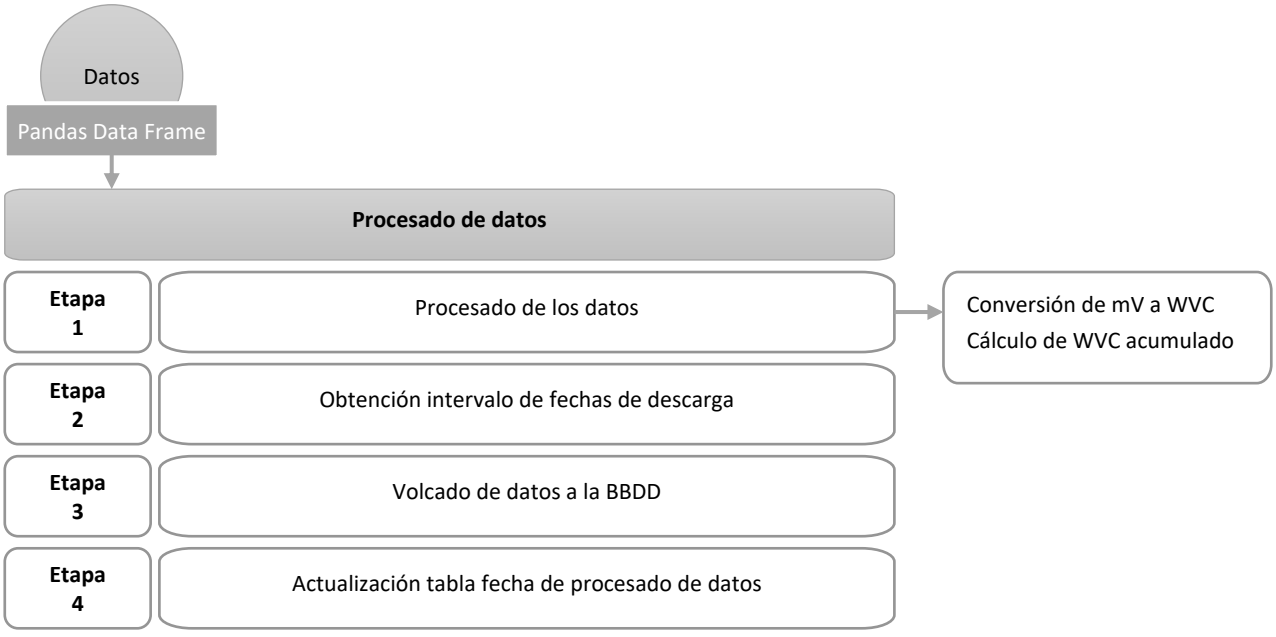

Figura 22. Etapas del script de procesado de datos.

4.2.2.5 Script 3: generación de productos

Se trata del script que, a través de consultas a la base de datos genera los gráficos de humedad acumulada en el suelo, de registro de cada sonda y las estadísticas básicas.

En la *Figura 23* se muestra, a modo de ejemplo, un gráfico de humedad del suelo acumulada:

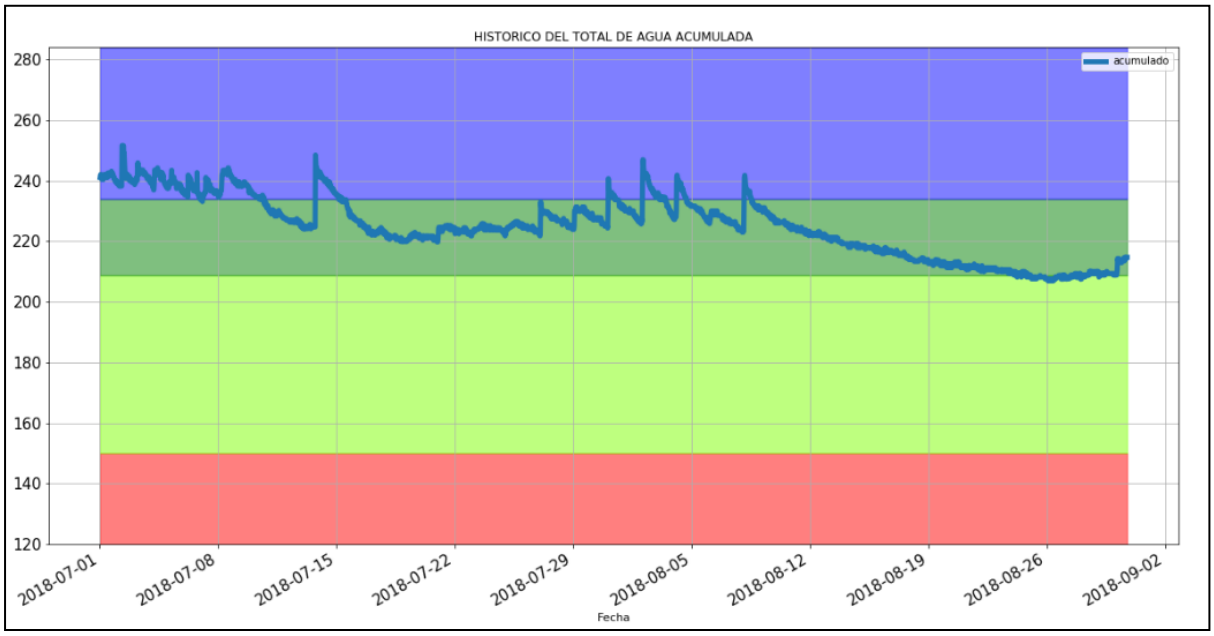

Figura 23. Gráfico de acumulación de agua en el suelo.

Como se observa, es un gráfico que en eje de las ordenadas se muestra en mm el contenido volumétrico de agua y en el de las abscisas, la fecha. Asimismo, se representa con una serie de códigos de colores que indican los límites de:

- Rojo: punto de marchitez.
- Verde claro: zona de estrés hídrico.
- Verde oscuro: zona de recarga.
- Azul: capacidad de campo.

Estos limites se obtienen de una tabla de la USDA (Departamento de Agricultura de los Estados Unidos) almacenada en la BBDD que en función del tipo de suelo de la parcela determina valores orientativos de capacidad de campo y punto de marchitez. El nivel de recarga lo fija cada cliente.

Seguidamente se presenta, a modo de ejemplo, un gráfico de medidas de humedad por cada sonda del dispositivo (*Figura 24*):

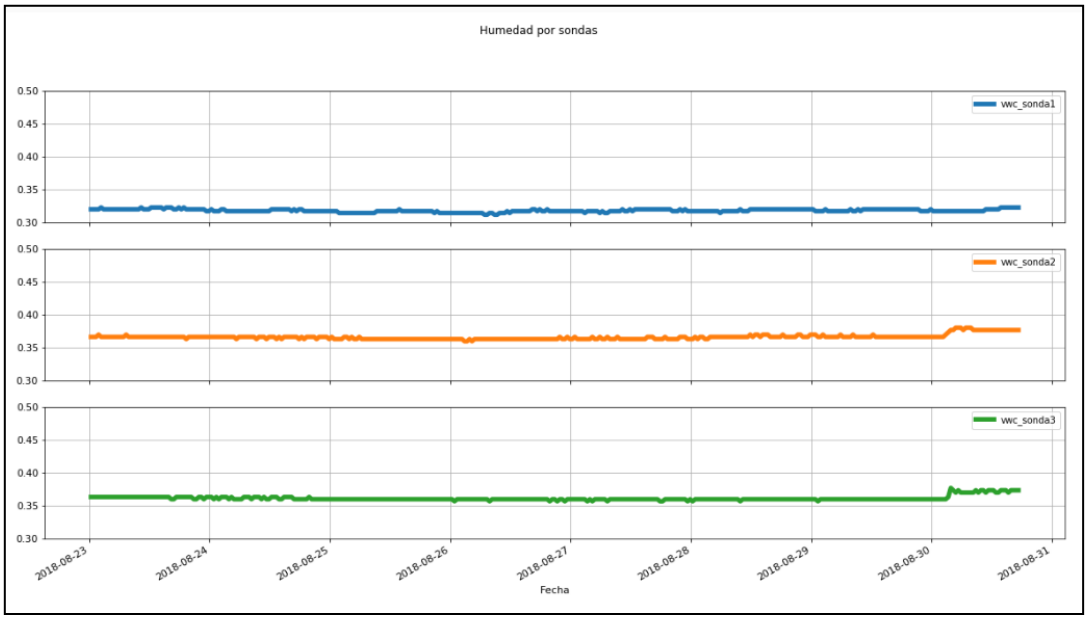

Figura 24. Gráfico de medición de humedad por sondas.

En eje de las ordenadas se muestra en mm el contenido volumétrico de agua y en el de las abscisas la fecha. Para realizar este gráfico se consideran los datos de la última semana.

Para finalizar, en relación a las estadísticas que se calculan éstas son:

- Número de registros
- Media
- Desviación estándar
- Mínimo
- Máximo
- Percentil 25%
- Percentil 50%
- Percentil 75%

4.2.2.6 Script 4: funciones

Se trata del dichero que contiene las funciones en las que se apoyan el resto de scripts de la aplicación. En total hay 36 funciones y se pueden consultar en el *Apartado 2.6 Script funciones* del *Anexo I. Códigos de programación.*

4.2.3 Base de datos

Las principales características de la base de datos se indican a continuación:

- Se han aplicado técnicas de normalización formales de bases de datos. Así, además de ofrecer un producto más profesional, se consigue disminuir el espacio de almacenamiento y se eliminan errores lógicos.

- Alta seguridad. Se han programado funciones SQL y añadido restricciones de tipo *unique* (no puede haber dos filas con la misma combinación de los campos que forman la restricción) que evitan duplicidades y/o datos incorrectos o corruptos. Asimismo, se han programado la realización periódica de copias de seguridad o *backup* para no perder la información.

- Alto rendimiento. Se han añadido índices espaciales a las tablas y programado periódicamente *vacuum* (limpia y optimiza la base de datos) para aumentar el rendimiento.

El código completo de la BBDD se recoge en el *Anexo II. Base de datos.*

4.2.3.1 Modelo conceptual: diagrama entidad relación

En este apartado se muestran las relaciones definidas entre las distintas tablas de la base de datos con el objetivo de normalizarla. Los diagramas de entidad relación se han realizado conforme se indica en el capítulo 3 del documento de (Elmasri, R., Navathe, 2000).

En la *Figura 25* se presenta el diagrama entidad- relación (E-R) de las tablas. Los campos subrayados con una línea continua y discontinua representan las llaves primarias y foráneas respectivamente.

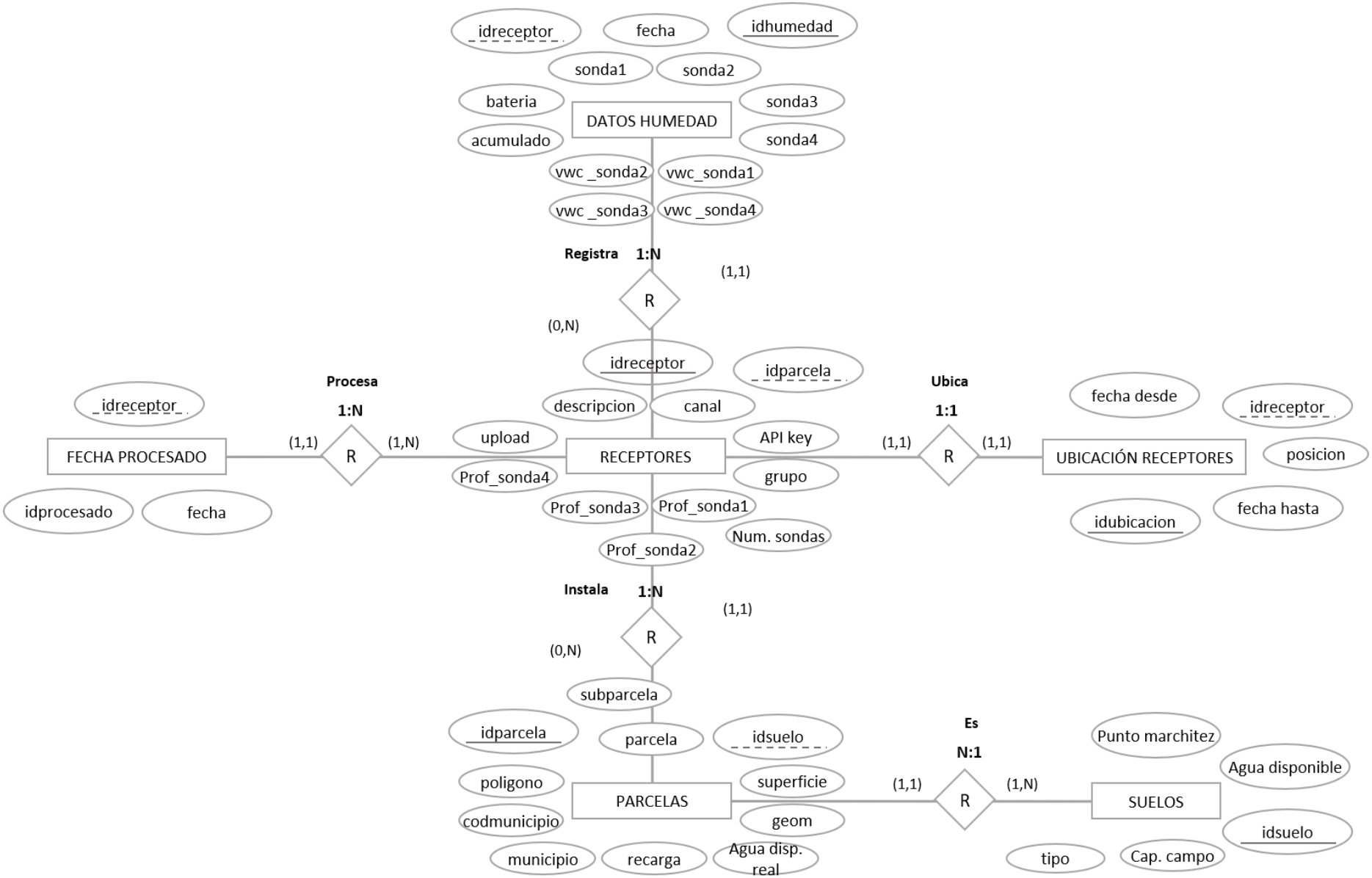

Figura 25. Modelo Entidad-Relación de la base de datos.

4.2.3.2 Modelo físico

El modelo físico de la BBDD se presenta en la *Figura 26*:

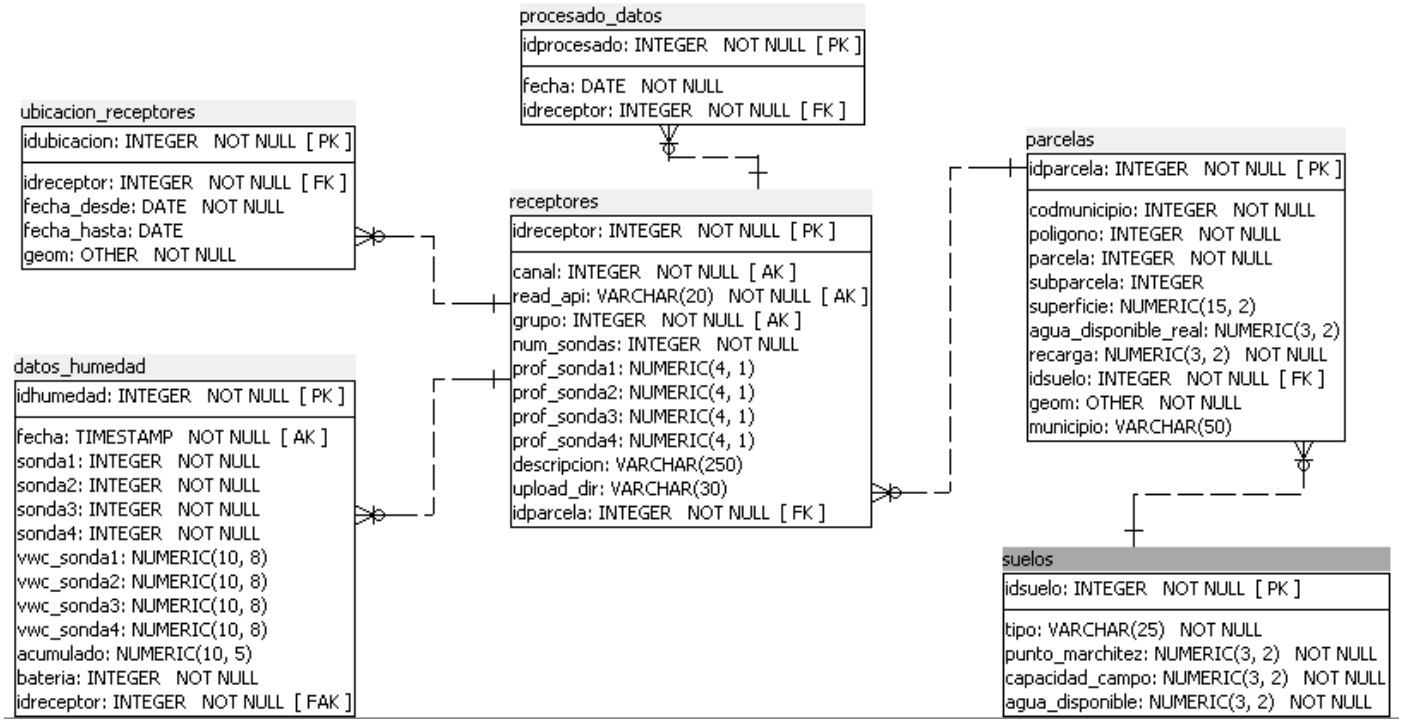

Figura 26. Modelo físico de la base de datos.

Como se puede observar, está formado por un total de 6 tablas:

- **Receptores:** información relativa a los dispositivos de medición de humedad.
- **Ubicación_receptores:** histórico de posiciones de los receptores o dispositivos.
- **Datos_humedad:** información relativa a los datos de humedad.
- **Procesado_datos:** registro que almacena la última fecha de procesado de cada receptor.
- **Parcelas:** información relativa a la parcela donde se ubica el receptor.
- **Suelos:** características de hídricas en función del tipo de suelo.

Asimismo, la base de datos consta de dos esquemas:

1. **Control**: se almacena la tabla de procesado de datos.
2. **Iot**: se almacenan el resto de tablas.

4.2.4 Página web

La estructura de la página web desarrollada se resume en la siguiente *Figura 27*:

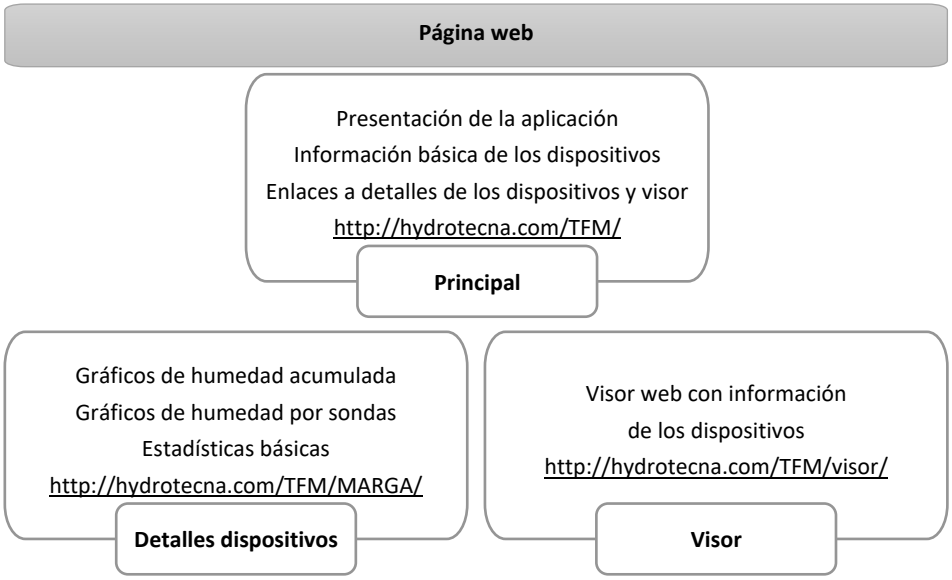

Figura 27. Esquema estructura página web.

Como se observa, se divide en una página principal que sirve a modo de presentación y de enlace a otras secciones, páginas de información de detalle de los datos de cada dispositivo y el visor web.

Cada uno de las páginas se actualiza diariamente, generando un nuevo HTML que sustituye al anterior y subiendo al servidor todo el contenido estático.

El código completo de la plantilla de la plantilla de la página principal, así como las de detalles y el visor se recogen en el *Anexo I. Códigos de programación.*

En los siguientes apartados se procede a la descripción más detallada de cada página.

4.2.4.1 Principal

Se trata de la página web principal cuya función es la de describir someramente la aplicación, los colaboradores del proyecto y los dispositivos en funcionamiento. A modo de ejemplo en la *Figura 28*, se puede observar la *interface* de esta página o se puede consultar a 20 de septiembre de 2018 en: http://hydrotecna.com/TFM/.

Asimismo, tiene enlaces a la información detallada de los registros de humedad de cada dispositivo y al visor web (*Figura 29*).

Finalmente tiene un apartado que permite contactar con los gestores de la aplicación. Esta sección está actualmente en desarrollo y, aunque está prevista su implementación, no es objeto de estudio del presente proyecto (*Figura 30*).

Figura 28. Cabecera página web principal.

Figura 29. Descripción de los dispositivos en la página web principal.

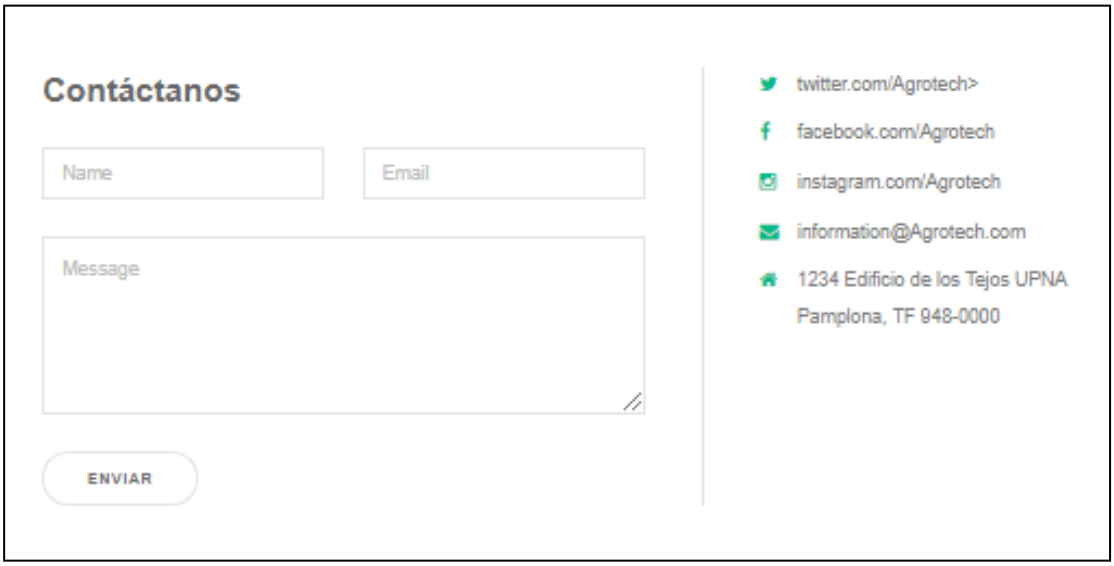

Figura 30. Apartado de contacto de la página web principal.

4.2.4.2 Detalles dispositivos

En la cabecera de esta página se muestra información básica de la ubicación del dispositivo y enlaces a las diferentes secciones (*Figura 31*). Permite la visualización de los datos de humedad medidos por los dispositivos en forma de gráficos (*Figura 32*) y sus estadísticas básicas (*Figura 33*).

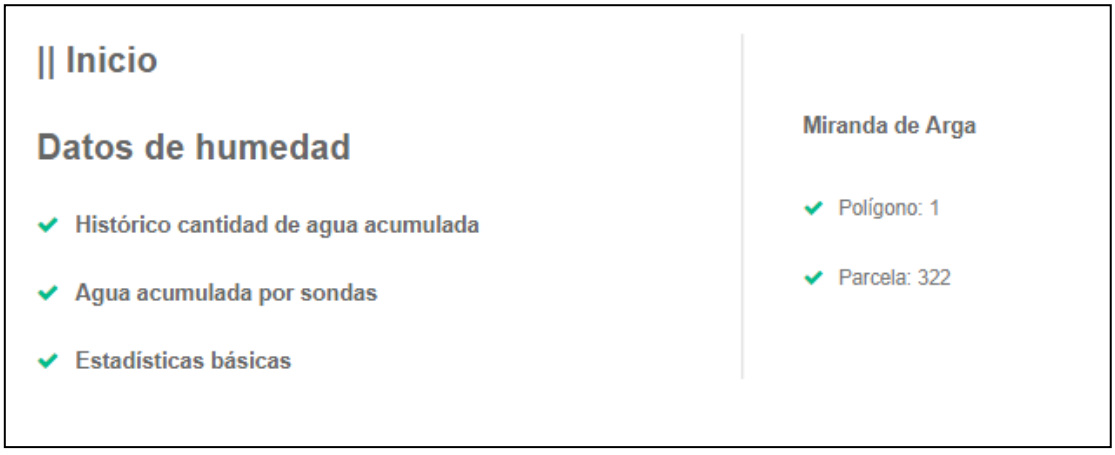

Figura 31. Cabecera de la página web de detalles de los dispositivos.

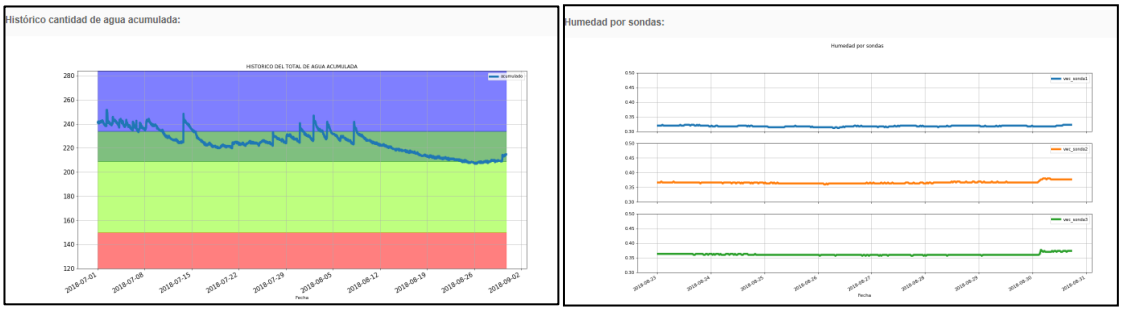

Figura 32. Gráficos de la página web de detalles de los dispositivos.

Figura 33. Estadísticas de la página web de detalles de los dispositivos.

4.2.4.3 Visor

Se trata de un visor web en el que se pueden ver georreferenciados, los datos más relevantes captados por los dispositivos. El visor se describe con mayor detalle en la siguiente sección.

4.2.5 Visor

Se ha empleado la API SITNA para crear un visualizador de mapa (*Figura 34*) que incluye distintos mapas de fondo, así como funcionalidades habituales de navegación (zoom, mapa de situación, medición, búsquedas, Google Street View, etc.)

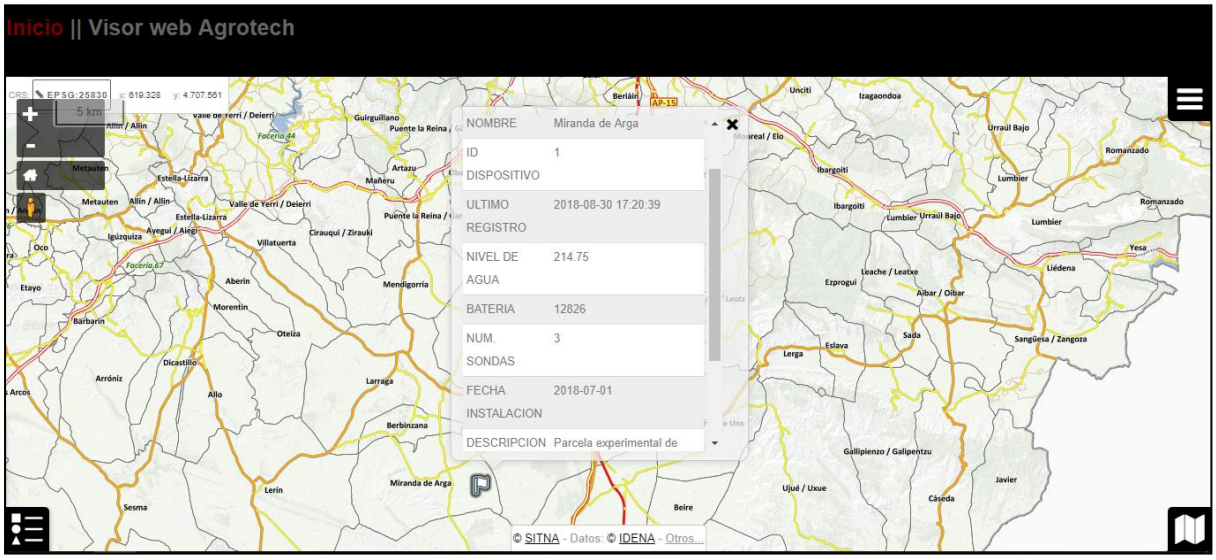

Figura 34. Visor web Agrotech.

Desde el visor se visualiza un fichero KML que incluye la siguiente información:

- Nombre de dispositivo
- ID de dispositivo
- Fecha del último registro
- Nivel de agua en campo (mm)
- Nivel de batería (mV)
- Número de sondas
- Fecha de instalación del equipo
- Descripción
- Enlace a los datos detallados

En relación a las funcionalidades del visor, éstas se resumen en la siguiente *Figura 35*:

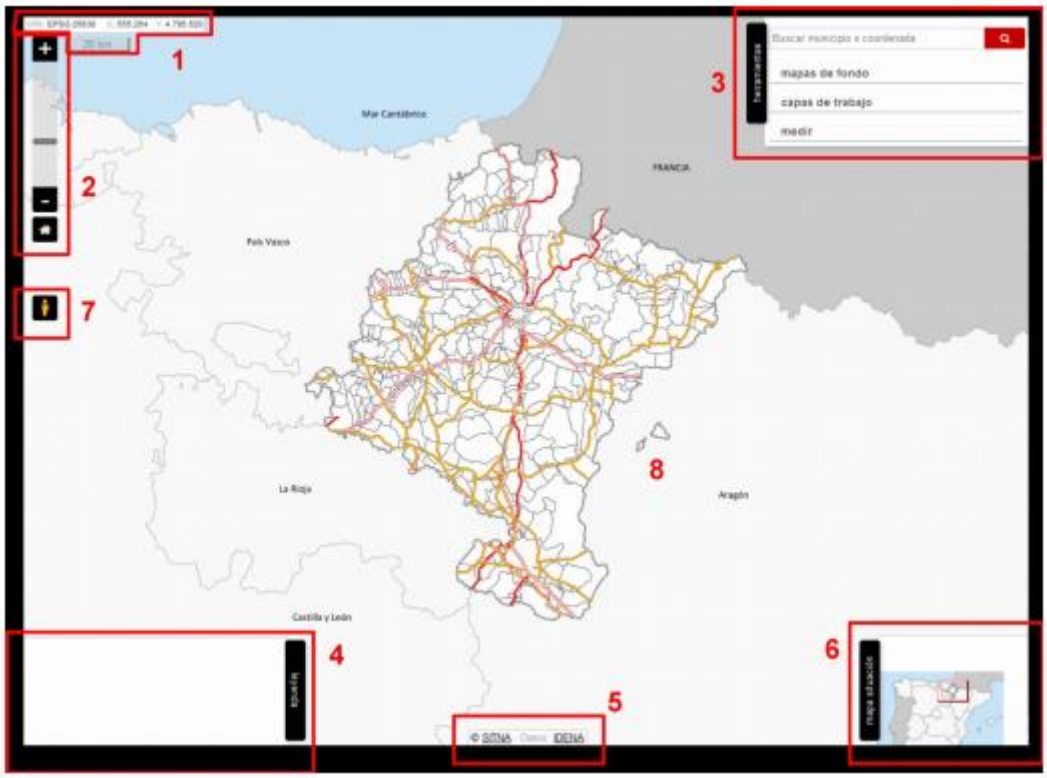

Figura 35. Funcionalidades del visor. (Fuente SITNA, 2018a)

1. Información del sistema de referencia espacial del mapa y las coordenadas del punto en que está situado el cursor. El sistema de referencia espacial por defecto es el EPSG:25830 ETRS89 y coordenadas UTM, pero clicando sobre el panel es posible modificarlo (*Figura 36*). Asimismo, se incluye la barra de escala.

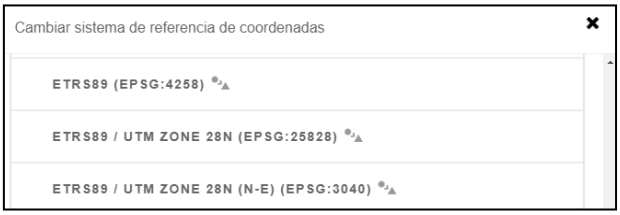

Figura 36. Funcionalidad de cambio de sistema de coordenadas del visor.

2. Barra de zoom para acercar o alejar el mapa y botón de inicio que ajusta el mapa a su extensión inicial.

3. Pestaña herramientas con las funciones de búsqueda, mapas de fondo, capas de trabajo y medición.

- Búsqueda espacial de municipios, cascos urbanos, direcciones, referencias catastrales y coordenadas (*Figura 37*).
- Cuatro mapas de fondo: la ortofoto más reciente, catastro, cartografía topográfica y el mapa base con información como límites administrativos, toponimia, hidrografía, etc. (*Figura 38*).
- La función de medir permite la medición tanto de la longitud como de áreas y perímetros (*Figura 39*).

Figura 37. Funcionalidad de búsqueda espacial del visor. (Fuente SITNA, 2018a).

Figura 38. Mapas base del visor.

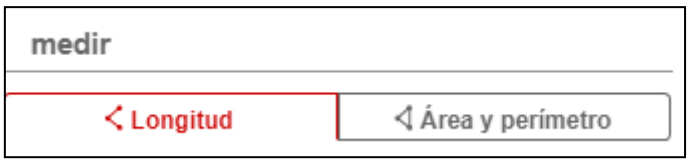

Figura 39. Funcionalidad de medición del visor.

4. Leyenda del mapa.

5. Procedencia de la información geográfica.

6. Mini mapa que muestra la ubicación del área geográfica actual del visor.

7. Control de Google Street View, arrastrando el icono a un punto del mapa, abre una ventana de Google Street View.

5. CONCLUSIONES

En este trabajo se ha dearrollado un sistema IoT compuesto por una rutina programada en Python una base de datos espacial PostgreSQL y PostGIS y un visor web con la API SITNA.

- El desarrollo de un prototipo de sistema de medición del nivel de humedad de un suelo agrícola que constituye el objeto de este TFM se ha demostrado posible, mediante la utilización de herramientas de bajo coste y software libre. Ello facilita su implementación y aplicación en el ámbito de la agricultura de precisión, y su utilización de forma accesible y comprensible por personas no especializadas en estas tecnologías.

- Este tipo de soluciones se revelan particularmente idóneas en entornos rurales, donde no se suede disponer de las conexiones inalámbricas exitentes en ámbitos urbanos, como por ejemplo wifi, y donde la capacidad de inversión económica en este tipo de sistemas es limitada.

- La aplicación de soluciones como la descrita en este TFM no queda restringida a la medición de la humedad del suelo, sino a otras variables agroclimáticas de interés en la producción agrícola.

- El planteamiento de este trabajo pone de manifiesto la idoneidad de integrar diversas tecnologías relacionadas con GIS e IoT, para aplicaciones de utilidad real, lo que se corresponde con las materias incluidas en el Máster Universitario en Teledetección y Sistemas de Información Geográfica de la UPNA.

- El análisis de alternativas realizado evidencia que cada vez existen más opciones para desarrollar soluciones profesionales en este campo utilizando herramientas de software libre o sin coste para el usuario.

- En lo que respecta concretamente a las plataformas IoT, existen en el mercado numerosas soluciones para diversas aplicaciones, siendo las de código libre ThingSpeak y Ubidots las más adecuadas para el desarrollo de prototipos como el planteado en este TFM.

- Ha sido posible gestionar y tratar información georreferenciada sin necesidad de recurrir a programas GIS de escritorio, debido al uso de bases de datos espaciales como PostgreSQL y PostGIS y el visor web de la API SITNA.

6. LÍNEAS FUTURAS

El principal aspecto de mejora de la aplicación está relacionado con el desarrollo de una web dinámica que permita ejecutar los programas en la web. De esta manera, se podría conectar el visor a la base de datos, por ejemplo, de manera asíncrona con con Ajax u otros protocolos, para que muestre los datos en tiempo real.

Asimismo, se podrían utilizar herramientas para la creación de gráficos dinámicos como es Python Dash y cuyo primer análisis se incluye en el *Anexo III. Python Dash.*

En cuanto a las funcionalidades del visor, una línea futura es incorporar información de imágenes satelitales, para entre otos aspectos, controlar el estado fenológico de los cultivos.

Una mejora que debe analizarse es la de añadir en la aplicación la posibilidad de manejar o programar los sistemas de riego y que ofrezcan recomendaciones personalizadas en función del contenido de humedad.

Otra vía de investigación interesante es incluir la monitorización de más variables agroclimáticas, como puede ser la temperatura, de forma que se pueda controlar más ampliamente el desarrollo del cultivo y se optimice la gestión de la explotación agrícola.

Por último, queda pendiente el desarrollo de perfiles de usuario, tanto en la base de datos como en el visor, que restrinja el acceso de información en función de los dispositivos dados de alta.

7. BIBLIOGRAFÍA

Amazon. (2018). Internet de las cosas | Plataforma como servicio | AWS IoT. Obtenido de: *https://aws.amazon.com/es/iot/*

Anaya-Isaza Diego H Peluffo-Ordoñez, A. J., Ivan-Rios Juan Castro-Silva, J. A., y Carvajal Ruiz Luis H Espinosa Llanos, D. A. (2017). Internet of Things for Irrigation System (IoT). *Research Gate*

Boluwade, A., y Ferdinand, A. (2011). The Design and Development of Spatial Database Management Systems (SDMS) for Hydrographic Studies using Coupled Open-Source GIS and Relational Database. *The Pacific Journal of Science and Technology*, *12*(1), 286–291.

Bongiovanni, R., Mantovani, E., Best, S., y Roel, Á. (2006). Agricultura de precisión: integrando conocimientos para una agricultura moderna y sustentable. *Procisur/IICA.*

Carriots. (2018). Carriots - IoT Platform. Obtenido de: *https://www.carriots.com/*

Chávez Ramírez, E. (2007). Aproximación del Riego en Tiempo Real; a Partir de Variables Agroclimáticas (Caso de Estudio Nogal Pecanero). *Universidad Autónoma Agraria Antonio Narro.*

Elmasri, R., Navathe, S. B. (2000). Fundamentals of database systems (3rd edtition). *Addison Wesley Longman, Inc.*

Embeblue S.L. (2018). embeblue - Embeblue. Obtenido de: *http://www.embeblue.com/*

Emmen, D. (2004). La agricultura de precisión: una alternativa para optimizar los sistemas de producción. *Investig. Pensam. Crít* , *2*, 68–74.

Evett, S. R., Schwartz, R. C., Casanova, J. J., y Heng, L. K. (2012). Soil water sensing for water balance, ET and WUE. *Agricultural Water Management*. Obtenido de: *https://doi.org/10.1016/j.agwat.2011.12.002*

Flores Medina, M. de J., Velasco M., V. D., y González C., G. (2015). Sistema de monitoreo automático de la humedad del suelo. Redes inalámbricas de sensores y sistemas de información geográfica: nuevas tecnologías para la agricultura de precisión. *Editorial Académica Española.*

García-Hernández, C. F., Ibarguengoytia-Gonzalez, P. H., García-Hernández, J., y Pérez-Díaz, J. A. (2007). Wireless sensor networks and applications: a survey. *International Journal of Computer Science and Network Security*, *7*(3), 264–273.

Garrido, R. L. (2017). Estudio Plataformas IoT. Obtenido de *http://openaccess.uoc.edu/webapps/o2/bitstream/10609/42812/6/rloureiroTFC0615memoria.pdf*

Godwin, R. J., y Miller, P. C. H. (2003). A review of the technologies for mapping within-field variability. *Biosystems Engineering*, *84*(4), 393–407.

Google. (2018). Google Maps Platform - APIs de geolocalización | Google Maps Platform | Google Cloud. Obtenido de: *https://cloud.google.com/maps-platform/?hl=es*

Goswami, S. B., Matin, S., Aruna, S., y Bairagi, G. D. (2012). A Review: The application of remote sensing, GIS and GPS in precision agriculture. *Journal of Advanced Technology and Engineering Research*, *2*, 50–54.

Guerrero-Ibañez, J. A., Estrada-Gonzalez, F. P., Medina-Tejeda, M. A., Rivera-Gutierrez, M. G., Alcaraz-Aguirre, J. M., Maldonado-Mendoza, C. A., Lopez-Gonzalez, V. I. (2017). SGreenH-IoT: Plataforma IoT para Agricultura de Precisión. *Sistemas, Cibernética e Informática*

Hari Ram, V. V., Vishal, H., Dhanalakshmi, S., y Vidya, P. M. (2015). Regulation of water in agriculture field using Internet Of Things. *Technological Innovation in ICT for Agriculture and Rural Development (TIAR), 2015 IEEE* (pp. 112–115).

Kubicek, P., Kozel, J., Stampach, R., y Lukas, V. (2013). Prototyping the visualization of geographic and sensor data for agriculture. *Computers and Electronics in Agriculture*, *97*, 83–91.

LabFerrer. (2017). *Las sondas ECH2 O de Decagon Devices*. Obtenido de: *http://www.decagon.com/education/soil-moistu-*

Leaflet. (2018). Leaflet - a JavaScript library for interactive maps. Obtenido de: *https://leafletjs.com/index.html*

Martin, E. C., y Munoz, C. (2017). Métodos para Medir la Humedad del Suelo para la Programación del Riego ¿Cuándo?. *College of Agriculture, University of Arizona (Tucson, AZ)*

METER Group. (2017). ECH2 O 10HS Soil Water Content. Obtenido de: *http://manuals.decagon.com/Manuals/13508_10HS_Web.pdf*

Mozilla.org. (2018). JavaScript | MDN. Obtenido de: *https://developer.mozilla.org/es/docs/Web/JavaScript*

Navarro-Hellín, H., Torres-Sánchez, R., Soto-Valles, F., Albaladejo-Pérez, C., López-Riquelme, J. A., y Domingo-Miguel, R. (2015). A wireless sensors architecture for efficient irrigation water management. *Agricultural Water Management*, *151*, 64–74.

OpenLayers. (2018). OpenLayers. Obtenido de: *http://openlayers.org/*

Pfister, S., Bayer, P., Koehler, A., y Hellweg, S. (2011). Projected water consumption in future global agriculture: Scenarios and related impacts. *Science of the Total Environment*, *409*(20), 4206–4216.

Python. (2018). About Python™ | Python.org. Obtenido de: *https://www.python.org/about/*

Rodrigo Martínez, J. (2017). Comparativa y estudio de plataformas IoT. *Universitat Politécnica de Catalunya.*

Ryu, M., Yun, J., Miao, T., Ahn, I.-Y., Choi, S.-C., y Kim, J. (2015). Design and implementation of a connected farm for smart farming system. *SENSORS, 2015 IEEE* (pp. 1–4).

sigAGROasesor. (2018). sigAGROasesor - Herramienta para la gestión sostenible de cultivos ecológicos. Obtenido de: *http://www.agroasesor.es/es/*

SITNA. (2018a). API SITNA v 1.5 Manual técnico de uso. Obtenido de: *http://sitna.navarra.es/geoportal/recursos/Manual%20tecnico%20uso%20API%20SITNA.pdf*

SITNA. (2018b). SITNA: Sistema de Información Territorial de Navarra. Obtenido de: *http://sitna.navarra.es/geoportal/recursos/api.aspx*

SparkFun. (2018). SparkFun SAMD21 Dev Breakout - DEV-13672 - SparkFun Electronics. Obtenido de: *https://www.sparkfun.com/products/13672*

Swain, N. R., Latu, K., Christensen, S. D., Jones, N. L., Nelson, E. J., Ames, D. P., y Williams, G. P. (2015). A review of open source software solutions for developing water resources web applications. *Environmental Modelling and Software*. https://doi.org/10.1016/j.envsoft.2015.01.014

Template. (2018). TEMPLATED - Free CSS, HTML5 and Responsive Site Templates. Obtenido de: *https://templated.co/*

ThingSpeak. (2018). IoT Analytics - ThingSpeak Internet of Things. Obtenido de: *https://thingspeak.com/*

Ubidots. (2018). IoT platform | Internet of Things | Ubidots. Obtenido de: *https://ubidots.com/*

Víctor Olaya. (2011). Sistemas de Información Geográfica. *Libro SIG*

Wenting, H., Zhiping, X., Yang, Z., Pei, C., Xiangwei, C., y Ooi, S. K. (2014). Real-time remote monitoring system for crop water requirement information. *International Journal of Agricultural and Biological Engineering*, *7*(6), 37–46.

Wilson, J. P., Mitasova, H., y Wright, D. J. (2000). Water resource applications of geographic information systems. *Urisa Journal*, *12*(2), 61–79.

Zheng, J., y Jamalipour, A. (2009). Wireless sensor networks: a networking perspective. *John Wiley y Sons.*

ANEXOS

ANEXOS

ANEXO I. CÓDIGOS DE PROGRAMACIÓN

ÍNDICE ANEXO 1

1. INTRODUCCIÓN

En este anexo se recogen los diferentes códigos de programación realizados para el desarrollo del presente TFM.

De esta manera, se incluyen todos los scripts de Python que ejecutan la aplicación y los diferentes HTML empleados en la página web.

2. SCRIPTS DE PYTHON

2.1 ARCHIVO DE CONFIGURACIÓN INICIAL

El archivo de configuración inicial es el archivo base del que el resto de los scripts de la aplicación adquieren las variables.

```
[configuracion]
bbdd = postgresql
labels = receptoresAgrotech
indexColumn = fecha
dropColumn =ID
ubicacionArchivos = directorios
servidor = servidor
correoErrores = correo

[postgresql]
host=localhost
database=agrotech
user=postgres
password=postgres

[receptoresAgrotech]
labels=fecha,ID,sonda1,sonda2,sonda3,sonda4,bateria

[receptoresOperativos]
id=1,2,3,4,5

[directorios]
mainPath=C:/Users/Unai/Dropbox/TFM_MUSIGT/Programacion
graphPath=/Graficos/
subGimg=SubG_idReceptor_'+str(idreceptor)+'_'+str(fechaI)+'.png
Gimg=GAcu_idReceptor_'+str(idreceptor)+'_'+str(fechaI)+'.png
htmlPath=/Web/
htmlindex=index.html
kmlPath=C:/Users/Unai/Dropbox/TFM_MUSIGT/Visor/dispositivos.kml

[servidor]
servidor = 82.98.134.159
usuario = hydrotecna
password = XXXX

[correo]
from = agrotech.upna@gmail.com
password = XXXXXX
to = u.gomez.ibanez@gmail.com
subject = Error en tratamiento de datos
```

2.2 SCRIPT LANZADOR DE LA APLICACIÓN

```
#-----LIBRERIAS
mainPath = '/Programacion/Script'
import sys
sys.path.append(mainPath)
import funciones_agrotech as fa
import descarga_datos_BBDD as dd
import procesado_datos_BBDD as prod
import productos_Agrotech as pa
import gestionFicheros
from datetime import datetime, date, time, timedelta
import pandas as pd

#-----VARIABLES INICIALES

#Fichero de configuracion
ficheroConfiguracion = 'BBDD.ini'
#Contenido general del fichero de configuracion
listaConfiguracion =
fa.getDicfromSection(ficheroConfiguracion,'configuracion')
#Parametros de conexion a la base de datos
db = fa.getDicfromSection(ficheroConfiguracion,listaConfiguracion['bbdd'])
#Receptores operativos
receptoresOperativos =
fa.gruposReceptores(db,ficheroConfiguracion,'receptoresOperativos')
#Directorios de trabajo
pathArchivos=
fa.getDicfromSection(ficheroConfiguracion,listaConfiguracion['ubicacionarch
ivos'])
#Datos del servidor
server=
fa.getDicfromSection(ficheroConfiguracion,listaConfiguracion['servidor'])
#Correo de aviso de errores
correoErrores=
fa.getDicfromSection(ficheroConfiguracion,listaConfiguracion['correoerrores
'])
#Diccionario para imagenes graficos del html
htmlImg = {}

try:
    #-----INICIO PROGRAMA

    #Primer bucle: por grupos de receptores
    for i in range(len(receptoresOperativos)):
        #Determinar el numero de receptores en el grupo
        lenGrupo = len(receptoresOperativos[i])
        #Segundo bucle: inicio descarga, procesado productos del receptor
del grupo[i]
        for idReceptor in receptoresOperativos[i]:
            try:
                #---DESCARGA DE DATOS
                df, infoReceptor, fechaI, fechaF =
dd.descargaDatosBBDD(ficheroConfiguracion, listaConfiguracion, idReceptor,
db)
                #---PROCESADO DATOS
                prod.procesado_datos_BBDD(infoReceptor, idReceptor, df,
fechaI, fechaF, db)
                #---PRODUCTOS (graficos, html)
                #Extraer las rutas de trabajo
```

```
                outputPath = fa.agrotechpathGenerator(pathArchivos,
idReceptor, fechaI, lenGrupo)
                #Graficos y estadísticas
                htmlImg = pa.graficosAgrotech(df, idReceptor, db,
listaConfiguracion, outputPath, infoReceptor, receptoresOperativos, i,
htmlImg)
                pathArchivos.update(htmlImg)
                #Error
                error = 'Sin errores'
            except Exception as e:
                error = 'Error en fase tratamiento de datos del receptor
%s: %s' %(idReceptor,e)
                fa.enviarCorreo(error, correoErrores)

        #---WEB
        if error == 'Sin errores':
            try:
            #HTML

fa.generarHTML(outputPath['htmltemplate'],outputPath['htmlindex'],htmlImg,i
dReceptor,db)
                #Subir datos al servidor

pa.subirDatosServidor(server,infoReceptor[0][10],db,lenGrupo,pathArchivos)
            except Exception as e:
                errorWeb = 'Error en fase web del receptor %s: %s'
%(idReceptor,e)
                fa.enviarCorreo(errorWeb, correoErrores)


        subirKml(server,parametrosConexion,pathArchivos['kmlpath'])
        #---BORRAR GRAFICOS
        eliminarGraficos =
gestionFicheros.ficherosEnCarpeta(pathArchivos['mainpath']+pathArchivos['gr
aphpath'])
        for archivo in eliminarGraficos:
            gestionFicheros.eliminaArchivo(archivo)

except Exception as e:
    errorGeneral = 'Error en receptor %s: %s' %(idReceptor,e)
    print(errorGeneral)
```

2.3 SCRIPT DESCARGA DE DATOS

```
#-----LIBRERIAS
mainPath = '/Programacion/BBDD'
import sys
sys.path.append(mainPath)
import funciones_agrotech as fa
from datetime import datetime, date, time, timedelta

def descargaDatosBBDD(ficheroConfiguracion,listaConfiguracion, idReceptor,
parametrosConexion):
    try:
        #-----VARIABLES

        #Datos de los receptores
        sqlReceptores= "SELECT * FROM iot.receptores WHERE idreceptor = "+
str(idReceptor)+" ORDER BY idreceptor"
```

```
        infoReceptor=fa.getDataFromTable(parametrosConexion, sqlReceptores)

        #Datos fecha de procesado
        sqlFechaProcesado = "SELECT fecha FROM control.procesado_datos
WHERE idreceptor = "+ str(idReceptor)+" ORDER BY fecha DESC LIMIT 1"
        fechaI  = fa.getDateFromTable(parametrosConexion,sqlFechaProcesado)
#Fecha inicio intervalo descarga
        fechaF = date.today() + timedelta(days=1) #Fecha final intervalo
descarga, se indica el día siguiente para que descargue también los de hoy

        #-----DESCARGA DE DATOS DE Think Speak
        # Lista de ID necesarias para conectarse con ThinkSpeak: #channel =
(infoReceptor[1]) y apiKey (infoReceptor[2]) de la lista de infoReceptor

        #Peticion de los datos a la API de TS en formato csv. Requiere el
canal, Api, fecha inicio y fecha final
        dataList =
fa.descargaDatosThinkSpeak(infoReceptor[0][1],infoReceptor[0][2], fechaI,
fechaF) #entre 1 y 2 devuelve los datos del día 1

        #-----CONVERSION DE LOS DATOS A TIPO DATAFRAME
        #Definir columnas
        labels =
fa.dataFromSection(ficheroConfiguracion,listaConfiguracion['labels'])

        #Pandas dataframe de una lista de datos
        df = fa.dfFromList(dataList, labels[0],
listaConfiguracion['indexcolumn'])

        #Borrar columnas
        dropCol = listaConfiguracion['dropcolumn']
        if dropCol != '':
            df = df.drop(dropCol, 1)

        return df, infoReceptor, fechaI, fechaF
    except Exception as e:
        print('Error en descarga receptor %s: %s' %(idReceptor,e))
        print(type(e))

    else:
        print("Descarga receptor %s terminada" %idReceptor)
    finally:
        print("fin funcion descarga")
```

2.4 SCRIPT DEPURACIÓN Y PROCESADO DE DATOS

```
#-----LIBRERIAS
mainPath = '/Programacion/BBDD'
import sys
sys.path.append(mainPath)
import funciones_agrotech as fa

#-----PROCESADO DATOS
def procesado_datos_BBDD(infoReceptor, idReceptor, df, fechaI, fechaF,
parametrosConexion):
    #try:
        #Listado de sondas operativas y profundidades
    sondas = infoReceptor[0][5:9] #Información de las sondas de la tabla
del receptor
    dfProcesado = fa.calculosProcesado(sondas,df)
```

```
    #Añadir datos de IdReceptor en el df para evitar la restriccion not
null de estos campos en la insercion
    df['idreceptor'] = infoReceptor[0][0]

    #--INSERTAR A BBDD
    #try:
        #Variable conexion general
    conexion = fa.getDbConn(parametrosConexion)
    #Insertar nuevos datos
    fa.insertDFBBDD(dfProcesado, conexion, 'iot.datos_humedad', idReceptor)
    #Fecha procesado
    fa.insertFechaProcesado(conexion, idReceptor)
    #Guarda los cambios realizados en la BBDD
    conexion.commit()
    #Cierra la conexión a la BBDD
    conexion.close()
```

2.5 SCRIPT GENERACIÓN DE PRODUCTOS

```
#-----LIBRERIAS
mainPath = '/Programacion/BBDD'
import sys
sys.path.append(mainPath)
import funciones_agrotech as fa
import os
from ftplib import FTP

 #-----GENERACION DE GRAFICOS
def graficosAgrotech(df, idReceptor, parametrosConexion,
listaConfiguracion, outputPath, infoReceptor, receptoresOperativos, i,
htmlImg):
#Subgraficos (representación de las mediciones de las sondas en la ultima
semana)
    fa.generarSubgraficos(df,idReceptor, parametrosConexion,
listaConfiguracion['indexcolumn'], outputPath['SubGimg'])
#Grafico Acumulado (representación de las mediciones de la acumulación
historica de agua)
    fa.generarGraficoAcumulado(idReceptor, parametrosConexion,
listaConfiguracion['indexcolumn'], infoReceptor, outputPath['Gimg'])
#Guardar nombre imagen en una lista para el html

htmlImg['ACUMULADO'+str(receptoresOperativos[i].index(idReceptor))+'']=os.p
ath.basename(outputPath['Gimg'])

htmlImg['SUBGRAFICO'+str(receptoresOperativos[i].index(idReceptor))+'']=os.
path.basename(outputPath['SubGimg'])

    estadisticas = estadisticasAcumulado(idReceptor, parametrosConexion,
listaConfiguracion, i)
    htmlImg.update(estadisticas)

    return htmlImg

#-----ESTADISTICAS
#Estadisticas basicas del acumulado de agua
def estadisticasAcumulado(idReceptor, parametrosConexion,
listaConfiguracion, i):
    dfAcu = fa.dfgraficoAcumulado(idReceptor, parametrosConexion,
listaConfiguracion['indexcolumn'])
```

```
    a = dfAcu.describe().round(2)
    b = a.T.to_dict('list')
    c= {}

    for key in b:
        b[key] = str(b[key][0])
        c[key+str(i)] = b[key]


    return c
#-----SERVIDOR
def
subirDatosServidor(server,uploadDir,parametrosConexion,lenGrupo,pathArchivo
s):

    #-----ESTABLECER CONEXION
    try:
        ftp = FTP(server['servidor'])
        ftp.login(server['usuario'], server['password'])
        print('[+] Conexion establecida correctamente')
    except Exception as e:
        print('[-] No se pudo estabpecer la conexión: '+str(e))

    #uploadDir = fa.getUploadDir(idReceptor,parametrosConexion)

    #-----ELIMINAR LOS GRAFICOS DEL DIRECTORIO
    ftp.cwd(uploadDir) #change into a specific directory
    files = ftp.nlst() #Return a list of file names
    #print(files)
    for name in files:
        if name.endswith(".png"):
            ftp.delete(name)

    #-----SUBIR ARCHIVOS
    if lenGrupo == 1:
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['ACUMULADO0'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['ACUMULADO0'],
uploadFile)
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['SUBGRAFICO0'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['SUBGRAFICO0'],
uploadFile)
        #html
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['htmlpath']+
pathArchivos['htmlindex'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['htmlindex'],
uploadFile)

        ftp.quit()

    elif lenGrupo == 2:
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['ACUMULADO0'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['ACUMULADO0'],
uploadFile)
```

```
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['SUBGRAFICO0'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['SUBGRAFICO0'],
uploadFile)
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['ACUMULADO1'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['ACUMULADO1'],
uploadFile)
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['graphpath']+
pathArchivos['SUBGRAFICO1'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['SUBGRAFICO1'],
uploadFile)
        #html
        uploadFile =
open(pathArchivos['mainpath']+pathArchivos['htmlpath']+
pathArchivos['htmlindex'],'rb')
        ftp.storbinary("STOR "+uploadDir+pathArchivos['htmlindex'],
uploadFile)

        ftp.quit()
```

2.6 SCRIPT FUNCIONES

```
#-----LIBRERIAS
import psycopg2
from configparser import ConfigParser
from datetime import datetime, date, time, timedelta
import urllib
import pandas as pd
import os
from io import StringIO
import pandas.io.sql as psql
import matplotlib.pyplot as plt
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from ftplib import FTP
import smtplib

#----INICIO
#Devuelve una lista con listas de grupos de receptores
def gruposReceptores(parametrosConexion, ficheroConfiguracion, seccion):

    receptoresOperativos = dataFromSection(ficheroConfiguracion,seccion)

    sqlGruposReceptores= "SELECT idreceptor, read_api, grupo FROM
iot.receptores  ORDER BY idreceptor"
    gruposReceptores=
getDataFromTable(parametrosConexion,sqlGruposReceptores)

    apiList = []
    apiList2 = []
    groupList = []

    for i in range(len(gruposReceptores)):
        apiList.append(gruposReceptores[i][1])
```

```
    apiSet = set(apiList)

    for apis in apiSet:
        for i in range(len(gruposReceptores)):
            if  gruposReceptores[i][1] == apis and
str(gruposReceptores[i][0]) in receptoresOperativos[0]:
                apiList2.append(gruposReceptores[i][0])

        groupList.append(apiList2)
        apiList2 = []

    groupList = list(filter(None, groupList))

    return groupList

#----DESCARGA DATOS

#Lee un archivo y devuelve en forma de diccionario los datos de la seccion
def getDicfromSection(filename, section):
    # create a parser
    parser = ConfigParser()
    # read config file
    parser.read(filename)

    # get section, default to postgresql
    dic = {}
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            dic[param[0]] = param[1]
    else:
        raise Exception('Section {0} not found in the {1}
file'.format(section, filename))

    return dic

#Conexión a la BBDD postgreSQL y descarga datos todo el contenido de una
tabla en función de una QUERY en un lista de tuplas
def getDataFromTable(parametrosConexion, sql):
    conn = None
    try:
        params = parametrosConexion
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        cur.execute(sql)
        rows = cur.fetchall()
        cur.close()
        return rows
    finally:
        if conn is not None:
            conn.close()

#Se conecta a la BBDD y devuelva el parámetro de conexión
def getDbConn(parametrosConexion):

    params = parametrosConexion
    conn = psycopg2.connect(**params)

    return conn

#Conexión a la BBDD postgreSQL y descarga de la ultima fecha de procesado
```

```
def getDateFromTable(parametrosConexion, sql):
    conn = None
    try:
        params = parametrosConexion
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        cur.execute(sql)
        fechaProcesado = cur.rowcount
        if fechaProcesado >0:
            for fecha, in cur:
                fechaProcesado = fecha
        else:
            fechaProcesado = date.today()
        cur.close()
        return fechaProcesado
    finally:
        if conn is not None:
            conn.close()

#Descarga los datos en un intervalo de fechas de la plataforma TS, los
codifica en utf-8 y devuelve una lista
def descargaDatosThinkSpeak(canal, apiKey, fechaI, fechaF):
    contents = urllib.request.urlopen(
        "http://api.thingspeak.com/channels/" + str(canal) +
"/feeds.csv?api_key=" + apiKey +
"&start="+str(fechaI)+"&end="+str(fechaF)).read() # Fecha: coge hasta el
día anterior, añadir un día más

    content_utf = contents.decode("utf-8") #Codifica en utf-8 el contenido
de la descarga
    content_utf = content_utf.replace(",", ";")
    lineas = content_utf.split("\n")
    lineas = lineas[1:len(lineas) - 1] #Para quitar el cabecero
    lineasLista = []
    for e in lineas:
        l = e.split(";")
        l = list(filter(None, l)) #Elimina los elementos en blanco de la
lista
        lineasLista.append(l)

    return lineasLista

#Lee un archivo .ini y devuelve los datos de los receptores en una lista
def dataFromSection(filename, section):
    # create a parser
    parser = ConfigParser()
    # read config file
    parser.read(filename)
    # get section, default to postgresql
    data = []
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            e = param[1].split(",")
            data.append(e)

    else:
        raise Exception('Section {0} not found in the {1}
file'.format(section, filename))

    return data
```

```
# Crea un df a partir de una lista, transfoma la columna de fecha en tipo
date y la establece como índice
def dfFromList(dataList, labels, indexLabel):
    df = pd.DataFrame.from_records(dataList, columns=labels)
    df[indexLabel] = pd.to_datetime(df[indexLabel]) #Convertir la columna
fecha en DateTime
    #df.set_index(indexLabel, inplace=True) #Se asigna el indice

    return df

# Devuelve el numero de receptores
def countReceptores(parametrosConexion):
    conn = None
    sql= "SELECT * FROM iot.receptores ORDER BY idreceptor"
    try:
        params = parametrosConexion
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        cur.execute(sql)
        numReceptores = cur.rowcount
        cur.close()
        return numReceptores
    finally:
        if conn is not None:
            conn.close()

#----PROCESADO DATOS

#Devuelve una lista con las sondas que están operativas (registran datos)
def sondasOperativas(sondas):
    sondasOperativas=[]
    #Devuelve la posicion de las sondas que estan operativas
    indices = [i+1 for i, s in enumerate(sondas) if s != None]

    for i in range(len(indices)):
        sondasOperativas.append('sonda'+str(indices[i]))

    return sondasOperativas

#Calcula (transforma) VWC a partir de un df con valores de mW de las sondas
operativas
def calculoVWC(df, sondasOperativas):
    df[sondasOperativas] = df[sondasOperativas].apply(pd.to_numeric)
#transformación a enteros los datos de las sondas operativas

    for sonda in sondasOperativas:
        df['vwc_'+sonda]=(2.97*10**(-9))*df[sonda]**3-(7.37*10**(-
6))*df[sonda]**2+(6.69*10**(-3))*df[sonda]-1.92

    return df

#Devuelve una lista con las profundidades de las sondas que están
operativas
def profundidadSondas(sondas):
    listaProfundidadSondas = []
    for e in sondas:
        if e != None:
            listaProfundidadSondas.append(e)

    return listaProfundidadSondas
```

```
#Devuelve una lista el area de inlfuencia de de las sondas que están
operativas
def areaInfluencia(profundidadSondas):
    areaInfluencia = []
    for i in range(len(profundidadSondas)-1):
        influencia = (profundidadSondas[i+1]-profundidadSondas[i])/2 +
profundidadSondas[0]
        areaInfluencia.append(influencia)

    areaInfluencia.append(influencia)

    return areaInfluencia

#Calcula el acumulado de agua de un df
def calculoAguaAcumulada(df,sondasOperativas, areaInfluenciaSondas):
    df['acumulado'] = 0
    for i in range(len(sondasOperativas)):
        df['acumulado'] +=
df['vwc_'+sondasOperativas[i]]*float(areaInfluenciaSondas[i])

    return df

#Calcula todos los datos de procesado directamente
def calculosProcesado(sondas,df):
    #Sondas operativas
    ListasondasOperativas = sondasOperativas(sondas)
    #Calculo de WMC (de mW a mm de agua)
    dfProcesado1 = calculoVWC(df, ListasondasOperativas)
    #Calculo de acumulación de agua
    profSondas = profundidadSondas(sondas)
    areaInfluenciaSondas = areaInfluencia(profSondas)
    dfProcesado2 =
calculoAguaAcumulada(dfProcesado1,ListasondasOperativas,areaInfluenciaSonda
s)

    return dfProcesado2

#Devuelve en una lista las columnas de un df
def NombreColumnasDF(df):
    nombreTodasColumnas = []
    #nombreTodasColumnas.append(df.index.name)
    nombreColumnas = list(df.columns.values)

    for nombres in nombreColumnas:
        nombreTodasColumnas.append(nombres)

    return nombreTodasColumnas

#Añade al df los valores de idReceptor y geometría para insertar en la
tabla
def addIdreceptorGeom(infoReceptor, df):

    df['idreceptor'] = infoReceptor[0][0]#posicion relativa al orden que
tiene en la tabla receptores

    return df

def insertDFIntoBBDD(df, parametrosConexion, schemaTable, idReceptores):
    conn = None
    try:
```

```
        # Get a database connection
        params = parametrosConexion
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)

        # Initialize a string buffer
        sio = StringIO()
        sio.write(df.to_csv(index=None, header=None))  # Write the Pandas
DataFrame as a csv to the buffer
        sio.seek(0)  # Be sure to reset the position to the start of the
stream

        # Copy the string buffer to the database, as if it were an actual
file
        with conn.cursor() as c:
            c.copy_from(sio, schemaTable, columns=df.columns, sep=',')
            conn.commit()
    except Exception as e:
        print('Error en INSERT INTO receptor %s: %s' %(idReceptores,e))

    else:
        print("Transacción receptor %s terminada" %idReceptores)
    finally:
        if conn is not None:
            conn.close()
        print("fin Insercion de datos")

def insertDFBBDD(df, conexion, schemaTable, idReceptores):

    # Initialize a string buffer
    sio = StringIO()
    sio.write(df.to_csv(index=None, header=None))  # Write the Pandas
DataFrame as a csv to the buffer
    sio.seek(0)  # Be sure to reset the position to the start of the stream

    # Copy the string buffer to the database, as if it were an actual file
    with conexion.cursor() as c:
        c.copy_from(sio, schemaTable, columns=df.columns, sep=',')


#Devuelve un df a partir de una consula SQL a la BBDD
def dfFromDB(parametrosConexion, sql):
    conn = None
    try:
        params = parametrosConexion
        conn = psycopg2.connect(**params)
        df = psql.read_sql_query(sql, conn)

        return df

    finally:
        if conn is not None:
            conn.close()

#Elimina los duplicados entre dos df y devuelve un df con los valores
unicos
def differenceBtwn2df(df1,df2,subset):
    df3 = pd.concat([df1,df2]).drop_duplicates(subset= subset, keep= False)

    return df3
```

```
#Realiza todos los pasos de deteccion de duplicados directamente
def detectarDuplicados(infoReceptor, df, fechaI, fechaF,
parametrosConexion):
    #Añadir columnas de idreceptor y geom
    df = addIdreceptorGeom(infoReceptor, df)

    #df de los valores de la BBDD en las fechas de procesado
    columnsList = NombreColumnasDF(df)
    columnsStr = ','.join(columnsList)
    sqlDfBBDD = "SELECT "+columnsStr+" FROM iot.datos_humedad \
            WHERE fecha BETWEEN '"+str(fechaI)+"' AND '"+str(fechaF)+"'\
            AND idreceptor = "+str(infoReceptor[0][0])

    dfBBDD = dfFromDB(parametrosConexion, sqlDfBBDD)

    #Eliminar duplicados
    dfProcesado = differenceBtwn2df(df,dfBBDD,('fecha','idreceptor'))

    return dfProcesado


#Añade la fecha del ultimo procesado a la tabla de control
def insertFechaProcesado(conexion, idreceptor):

    sql = "INSERT INTO control.procesado_datos (idreceptor,fecha) VALUES
(%s, current_timestamp)"
    # create a new cursor
    cur = conexion.cursor()
    # execute the INSERT statement
    cur.execute(sql,str(idreceptor))


#----GENERACION DE GRAFICOS

#Devuelve una lista con los sondas que intervienen para hacer el subgrafico
def vwc_sondas(df):
    dfHeaderList = list(df)
    matching = [s for s in dfHeaderList if "vwc" in s]

    return matching


#Devuelve un df con los datos de vwc de la ultima semana
def dfSubgrafico(df, idReceptor, parametrosConexion, indexColumn):

    vwcList = vwc_sondas(df)
    vwcList.append(indexColumn)
    vwcStr = ','.join(vwcList)

    fechaF = date.today() + timedelta(days=1)
    fechaI = date.today() - timedelta(days = 7)

    sqlDfBBDD = "SELECT "+vwcStr+" FROM iot.datos_humedad \
        WHERE fecha BETWEEN '"+str(fechaI)+"' AND '"+str(fechaF)+"'\
        AND idreceptor = "+str(idReceptor)

    dfSubgrafico = dfFromDB(parametrosConexion, sqlDfBBDD)
    dfSubgrafico.set_index(indexColumn, inplace=True) #Se asigna el indice

    return dfSubgrafico
```

```
#Devuelve un diccionario con las rutas de salida de los outputs (graficos,
html)
def agrotechpathGenerator(g,idReceptor,fechaI, htmlModel):

    #subgrafico
    subGpath = g['subgimg'].split("'")
    subGpath[1]=str(idReceptor)
    subGpath[3]=str(fechaI)
    subGpath="".join(subGpath)
    subGpath=str(subGpath)

    #grafico
    Gpath = g['gimg'].split("'")
    Gpath[1]=str(idReceptor)
    Gpath[3]=str(fechaI)
    Gpath="".join(Gpath)
    Gpath=str(Gpath)

    agroPath = {'SubGimg':g['mainpath']+g['graphpath']+str(subGpath), \
                'Gimg':g['mainpath']+g['graphpath']+str(Gpath), \

'htmltemplate':g['mainpath']+g['htmlpath']+'index'+str(htmlModel)+'.txt', \
                'htmlindex':g['mainpath']+g['htmlpath']+'index.html'}

    return agroPath

#Crea el subgrafico y guarda la imagen en path establecido
def subgraficos(dataFrame,titulo,imgPath):
    ax = dataFrame.plot(figsize=(20,10), grid = 1, title =
titulo,linewidth=5, fontsize=11, subplots=True, ylim = (0.3,0.5))
    plt.xlabel('Fecha', fontsize=11)
    axx = ax[0]
    sub = axx.get_figure()
    sub.savefig(imgPath)

#Genera el subgrafico directamente
def generarSubgraficos(df,idReceptor, parametrosConexion, indexColumn,
imgPath):

    #df de la ultima semana
    dfSub= dfSubgrafico(df, idReceptor, parametrosConexion, indexColumn)

    #Crea el subgrafico
    subgraficos(dfSub,'Humedad por sondas',imgPath)

#Devuelve los limites del suelo (cc, pm, recarga) en funcion del tipo de
suelo de la parcela
def limiteSuelo(infoReceptor, idReceptor, parametrosConexion):
    sondas = infoReceptor[0][5:9]
    a = profundidadSondas(sondas)
    areaI = areaInfluencia(a)
    suelo = sum(areaI) # para ajustar, consultar

    sqlSuelo= "SET SEARCH_PATH = iot, public;\
               SELECT s.punto_marchitez, s.capacidad_campo,
p.agua_disponible_real, p.recarga, r.idreceptor\
               FROM suelos s, parcelas p, receptores r \
               WHERE r.idreceptor = "+str(idReceptor)+" AND p.idsuelo =
s.idsuelo AND p.idparcela = r.idparcela"
```

```
    datoSuelo= getDataFromTable(parametrosConexion,sqlSuelo)
    pm = float(datoSuelo[0][0])*float(suelo)
    cc = float(datoSuelo[0][1])*float(suelo)
    aguaDisp = float(datoSuelo[0][2])*float(suelo)
    recarga = cc-aguaDisp*float(datoSuelo[0][3])

    return pm, cc, recarga

#Crea el grafico y guarda la imagen en path establecido
def grafico(dataFrame, titulo, lim1, lim2, lim3,imgPath):
    ax= dataFrame.plot(figsize=(20,10), grid = 1, title =
titulo,linewidth=5, fontsize=15, ylim = (lim1-30,lim3+50))
    plt.xlabel('Fecha', fontsize=11)
    plt.fill_between(dataFrame.index, 0., lim1, color='red', alpha='0.5',
label = 'Punto marchitez')
    plt.fill_between(dataFrame.index, lim1, lim2, color='chartreuse',
alpha='0.5', label = 'Estrés hídrico')
    plt.fill_between(dataFrame.index, lim2, lim3, color='green',
alpha='0.5', label = 'Recarga')
    plt.fill_between(dataFrame.index, lim3, lim3+50, color='blue',
alpha='0.5', label = 'Capacidad campo')
    G = ax.get_figure()
    G.savefig(imgPath)

#Devuelve un df con los datos de acumulados historicos
def dfgraficoAcumulado(idReceptor, parametrosConexion, indexColumn):

    sqlDfBBDD = "SELECT acumulado,"+indexColumn+" FROM iot.datos_humedad \
                WHERE idreceptor = "+str(idReceptor)

    dfAcumulado = dfFromDB(parametrosConexion, sqlDfBBDD)
    dfAcumulado.set_index(indexColumn, inplace=True) #Se asigna el indice

    return dfAcumulado

#Genera el grafico de acumulacion de agua directamente
def generarGraficoAcumulado(idReceptor, parametrosConexion, indexColumn,
infoReceptor, imgPath):

    #df historico del acumulado
    dfAcumulado = dfgraficoAcumulado(idReceptor, parametrosConexion,
indexColumn)

    #limites del suelo
    pm, cc, recarga = limiteSuelo(infoReceptor, idReceptor,
parametrosConexion)

    #Crea el grafico
    grafico(dfAcumulado, 'HISTORICO DEL TOTAL DE AGUA ACUMULADA', pm,
recarga, cc, imgPath)



#-----GENERACION HTML

#Añade el nombre de las imagenes a insertar en el html
def htmlImg(gruposReceptores, i, idReceptor, outputPath):

   outputPath = {}
```

```
outputPath['ACUMULADO'+str(gruposReceptores[i].index(idReceptor))+'']=os.pa
th.basename(outputPath['Gimg'])

outputPath['SUBGRAFICO'+str(gruposReceptores[i].index(idReceptor))+'']=os.p
ath.basename(outputPath['SubGimg'])

   return outputPath

#Genera el html
def
generarHTML(htmlTemplatePath,htmlIndex,htmlImg,idReceptor,parametrosConexio
n):

    sqlUbicacion= "SELECT p.municipio, p.poligono, p.parcela\
                   FROM iot.receptores r, iot.parcelas p \
                   WHERE r.idparcela = p.idparcela AND idreceptor =
"+str(idReceptor)


    ubicacion = getDataFromTable(parametrosConexion,sqlUbicacion)
    htmlImg['MUNICIPIO']= str(ubicacion[0][0])
    htmlImg['POLIGONO']= str(ubicacion[0][1])
    htmlImg['PARCELA']= str(ubicacion[0][2])

    f = open(htmlTemplatePath, "r")
    html = f.read()

    for key in htmlImg:
        html=html.replace(key,htmlImg[key])

    ficheroSalida = open(htmlIndex,'w')
    ficheroSalida.write(html)

    f.close()
    ficheroSalida.close()


#-----SERVIDOR

#Devuelve el directorio donde subir los datos
def getUploadDir(idReceptor, parametrosConexion):

    sqlUploadDir= "SET SEARCH_PATH = iot, public;\
              SELECT upload_dir\
              FROM servidor \
              WHERE idreceptor = "+str(idReceptor)


    uploadDir= getDataFromTable(parametrosConexion,sqlUploadDir)


    return uploadDir

#-----CORREO
def enviarCorreo(error, correoErrores):
    # create message object instance
    msg = MIMEMultipart()
    message = error
    # setup the parameters of the message
    password = correoErrores['password']
```

```
    msg['From'] = correoErrores['from']
    msg['To'] = correoErrores['to']
    msg['Subject'] = correoErrores['subject']
    # add in the message body
    msg.attach(MIMEText(message, 'plain'))
    # create server
    server = smtplib.SMTP('smtp.gmail.com: 587')
    server.starttls()
    # Login Credentials for sending the mail
    server.login(msg['From'], password)
    # send the message via the server.
    server.sendmail(msg['From'], msg['To'], msg.as_string())
    server.quit()

    print ("successfully sent email to %s:" % (msg['To']))

#-----KML
def subirKml(server,parametrosConexion,kmlPath):
    #Consulta a la BBDD para obtener los datos del kml que se mostraran en
el visor
    sqlVisor= "SET search_path = iot, control, public;\
            SELECT r.nombre, d.idreceptor as Receptor, d.fecha,
round(d.acumulado,2) as niveldeagua, d.bateria, r.num_sondas as
numerodesondas,fecha_desde as fechadeinstalacion,
r.descripcion,r.upload_dir as graficos, ST_X(ST_Transform(u.geom,4326)) as
x, ST_Y(ST_Transform(u.geom,4326)) as y\
            FROM (\
                    SELECT idreceptor, MAX(fecha) fecha\
                     FROM iot.datos_humedad\
                     GROUP BY idreceptor) rf \
            INNER JOIN datos_humedad d \
            ON rf.idreceptor = d.idreceptor AND rf.fecha = d.fecha\
            INNER JOIN receptores r ON d.idreceptor = r.idreceptor\
            INNER JOIN ubicacion_receptores u ON u.idreceptor =
r.idreceptor\
            INNER JOIN parcelas p ON p.idparcela = r.idparcela;"

    datoVisor= getDataFromTable(parametrosConexion,sqlVisor)
    #Plantilla KML
    kmlCabecera = "<?xml version=\"1.0\" encoding=\"utf-8\" ?><kml
xmlns=\"http://www.opengis.net/kml/2.2\">\n\
        <Document>\n\
            <Style id=\"MyBalloonStyle\">\n\
              <BalloonStyle>\n\
                <text><![CDATA[ <h3> $[nombre]</h3> <table style='font-
size:small;text-align:left;border-collapse:collapse;cellpadding=5;'> <tr
bgcolor='#D4E4F3' align='left' valign='top'>  <td align='left'
valign='top'> NOMBRE </td>  <td align='left' valign='top'> $[nombre] </td>
</tr>  <tr align='left' valign='top'>  <td align='left' valign='top'> ID
DISPOSITIVO </td>  <td align='left' valign='top'> $[idreceptor] </td>
</tr>  <tr bgcolor='#D4E4F3' align='left' valign='top'>  <td align='left'
valign='top'> ULTIMO REGISTRO </td>  <td align='left' valign='top'>
$[ultimoregistro] </td>  </tr>  <tr align='left' valign='top'>  <td
align='left' valign='top'> NIVEL DE AGUA </td>  <td align='left'
valign='top'> $[nivelagua] </td>  </tr>  <tr bgcolor='#D4E4F3' align='left'
valign='top'>  <td align='left' valign='top'> BATERIA </td>  <td
align='left' valign='top'> $[bateria] </td>  </tr>  <tr align='left'
valign='top'>  <td align='left' valign='top'> NUM. SONDAS </td>  <td
align='left' valign='top'> $[numsondas] </td>  </tr>  <tr bgcolor='#D4E4F3'
align='left' valign='top'>  <td align='left' valign='top'> FECHA
INSTALACION </td>  <td align='left' valign='top'> $[fechainstalacion] </td>
```

```
</tr>  <tr align='left' valign='top'>  <td align='left' valign='top'>
DESCRIPCION </td>  <td align='left' valign='top'> $[descripcion] </td>
</tr>  <tr bgcolor='#D4E4F3' align='left' valign='top'>  <td align='left'
valign='top'> DATOS DETALLADOS </td>  <td align='left' valign='top'>
$[datosdetallados] </td>  </tr> </table> ]]></text>\n\
              </BalloonStyle>\n\
              <IconStyle>\n\
                <Icon>\n\

<href>http://maps.google.com/mapfiles/kml/pal5/icon13.png</href>\n\
                </Icon>\n\
                <hotSpot x=\"32\" y=\"1\" xunits=\"pixels\"
yunits=\"pixels\" />\n\
              </IconStyle>\n\
            </Style>\n\
            <Folder>\n\
             <name>Datos</name>\n\
             <open>0</open>\n"
    #Agregamos los datos
    for x in datoVisor:
        url = x[8].split("/")
        kmlDisp = "<Placemark>\n\
                  <name>"+str(x[0])+"</name>\n\
                  <styleUrl>#MyBalloonStyle</styleUrl>\n\

<Point><coordinates>"+str(x[9])+","+str(x[10])+"</coordinates></Point>\n\
                  <ExtendedData>\n\
                    <Data
name=\"nombre\"><value>"+str(x[0])+"</value></Data>\n\
                    <Data
name=\"idreceptor\"><value>"+str(x[1])+"</value></Data>\n\
                    <Data
name=\"ultimoregistro\"><value>"+str(x[2])+"</value></Data>\n\
                    <Data
name=\"nivelagua\"><value>"+str(x[3])+"</value></Data>\n\
                    <Data
name=\"bateria\"><value>"+str(x[4])+"</value></Data>\n\
                    <Data
name=\"numsondas\"><value>"+str(x[5])+"</value></Data>\n\
                    <Data
name=\"fechainstalacion\"><value>"+str(x[6])+"</value></Data>\n\
                    <Data
name=\"descripcion\"><value>"+str(x[7])+"</value></Data>\n\
                    <Data name=\"datosdetallados\"><value>&lt;a
href='http://hydrotecna.com/TFM/"+url[3]+"/'&gt;Abrir&lt;/a&gt;</value></Da
ta>\n\
                  </ExtendedData>\n\
                </Placemark>\n"
        kmlCabecera+=kmlDisp

    kmlCabecera +=      "</Folder>\n\
                        </Document>\n\
                </kml>\n"
    #Generamos el KML
    f = open(kmlPath, 'w')
    f.write(kmlCabecera)
    f.close()
    #Subir KML al servidor
    uploadDir = '/www/TFM/visor/'
    kmlFile = os.path.basename(kmlPath)
    #-----ESTABLECER CONEXION
```

```
    try:
        ftp = FTP(server['servidor'])
        ftp.login(server['usuario'], server['password'])
        print('[+] Conexion establecida correctamente')
    except Exception as e:
        print('[-] No se pudo estabpecer la conexión: '+str(e))
    ftp.cwd(uploadDir) #change into a specific directory
    files = ftp.nlst() #Return a list of file names
    for name in files:
        if name.endswith(".kml"):
            ftp.delete(name)
    uploadFile = open(kmlPath,'rb')
    ftp.storbinary("STOR "+uploadDir+kmlFile, uploadFile)
    ftp.quit()
```

3. HTML

3.1 PÁGINA PRINCIPAL

```
<!DOCTYPE HTML>
<!--
    Prism by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license
(templated.co/license)
-->

<html>
    <head>
        <title>Agrotech Principal</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1"
/>
        <!--[if lte IE 8]><script
src="../plantillaWeb/assets/js/ie/html5shiv.js"></script><![endif]-->
        <link rel="stylesheet" href="../plantillaWeb/assets/css/main.css"
/>
        <!--[if lte IE 9]><link rel="stylesheet"
href="../plantillaWeb/assets/css/ie9.css" /><![endif]-->
    </head>
    <body>

        <!-- Banner -->
            <section id="banner">
                <div class="inner split">
                    <section>
                        <h2>Bienvenido a Agrotech, plataforma de
visualización de datos de humedad del suelo <a
href="http://www.unavarra.es/">UPNA</a></h2>
                    </section>
                    <section>
                        <p>Proyecto realizado en colaboración con el
departamento de Ingeniería de la UPNA, FSinfrestructuras y la Cooperativa
de Artajona.</p>
                        <ul class="actions">
                            <li><a href="#one" class="button
special">Empecemos</a></li>
```

```
                        </ul>
                    </section>
                </div>
            </section>

        <!-- One -->
            <section id="one" class="wrapper">
                <div class="inner split">
                    <section>
                        <h2>Aplicación IoT</h2>
                        <p>Aplicación de IoT para la óptima gestión de
sistemas de riego, que mediante mediciones de contenido de humedad del
suelo de parcelas agrícolas ayuda en la toma de decisiones necesidades
hídricas.</p>
                        <ul class="actions">
                            <li><a href="http://hydrotecna.com/TFM/visor/"
class="button special">Visor Web</a></li>
                        </ul>
                    </section>
                    <section>
                        <ul class="checklist">
                            <li>Dispositivos IoT de gran autonomía</li>
                            <li>Sondas de medición de humedad a diferentes
profundidades</li>
                            <li>Adaptable a todo tipo de cultivos</li>
                            <li>Visor web con funcionalidades GIS</li>
                        </ul>
                    </section>
                </div>
            </section>

        <!-- Two -->
            <section id="two" class="wrapper style2 alt">
                <div class="inner">
                    <h2>Dispositivos IoT</h2>
                    <div class="spotlight">
                        <div class="image">
                            <img src="../plantillaWeb/images/MARGA.jpeg"
alt="" />
                        </div>
                        <div class="content">
                            <h3>Miranda de Arga</h3>
                            <p>Situado en la parcela 322 del polígono 1 de
Miranda de Arga (Navarra).</p>
                            <p>Parcela con cultivo de Maíz.</p>
                            <p>3 sondas a 10 cm, 30 cm y 50 cm.</p>
                            <ul class="actions">
                                <li><a
href="http://hydrotecna.com/TFM/MARGA/" class="button
alt">Detalles</a></li>
                            </ul>
                        </div>
                    </div>
                    <div class="spotlight">
                        <div class="image">
                            <img src="../plantillaWeb/images/INTIA1.jpeg"
alt="" />
                        </div>
                        <div class="content">
                            <h3>INTIA UNO</h3>
```

```
                            <p>Situado en la parcela 500 del polígono 1 de
Enériz (Navarra).</p>
                            <p>Parcela con cultivo de Maíz.</p>
                            <p>2 sondas a 10 cm y 30 cm.</p>
                            <ul class="actions">
                                <li><a
href="http://hydrotecna.com/TFM/INTIA1" class="button
alt">Detalles</a></li>
                            </ul>
                        </div>
                    </div>
                    <div class="spotlight">
                        <div class="image">
                            <img src="../plantillaWeb/images/INTIA2.jpeg"
alt="" />
                        </div>
                        <div class="content">
                            <h3>INTIA DOS</h3>
                            <p>Situado en la parcela 500 del polígono 1 de
Enériz (Navarra).</p>
                            <p>Parcela con cultivo de Maíz.</p>
                            <p>2 sondas a 10 cm y 30 cm.</p>
                            <ul class="actions">
                                <li><a
href="http://hydrotecna.com/TFM/INTIA2/" class="button
alt">Detalles</a></li>
                            </ul>
                        </div>
                    </div>
                    <ul class="actions special">
                        <li><a href="#" class="button alt">Ir a
arriba</a></li>
                    </ul>
                </div>
            </section>

        <!-- Contact -->
            <section id="contact" class="wrapper">
                <div class="inner split">
                    <section>
                        <h2>Contáctanos</h2>
                        <p¡Mande un correo con sus dudas y le contestaremos
lo antes posible!.</p>
                        <form action="#" class="alt" method="POST">
                            <div class="row uniform">
                                <div class="6u 12u$(xsmall)">
                                    <input name="name" placeholder="Name"
type="text">
                                </div>
                                <div class="6u$ 12u$(xsmall)">
                                    <input name="email" placeholder="Email"
type="email">
                                </div>
                                <div class="12u$">
                                    <textarea name="message"
placeholder="Message" rows="4"></textarea>
                                </div>
                            </div>
                            <ul class="actions">
                                <li><input class="alt" value="Enviar"
type="submit"></li>
```

```
                            </ul>
                        </form>
                    </section>
                    <section>
                        <ul class="contact">
                            <li class="fa-twitter"><a
href="#">twitter.com/Agrotech></li>
                            <li class="fa-facebook"><a
href="#">facebook.com/Agrotech</a></li>
                            <li class="fa-instagram"><a
href="#">instagram.com/Agrotech</a></li>
                            <li class="fa-envelope"><a
href="#">information@Agrotech.com</a></li>
                            <li class="fa-home">1234 Edificio de los Tejos
UPNA<br/>Pamplona, TF 948-0000</li>
                        </ul>
                    </section>
                </div>
            </section>

        <!-- Footer -->
            <footer id="footer">
                <div class="copyright">
                    ycopy; Untitled. All rights reserved. Images: <a
href="http://unsplash.com">Unsplash</a>. Design: <a
href="http://templated.co">TEMPLATED</a>.
                </div>
            </footer>

        <!-- Scripts -->
            <script src="../plantillaWeb/assets/js/jquery.min.js"></script>
            <script src="../plantillaWeb/assets/js/skel.min.js"></script>
            <script src="../plantillaWeb/assets/js/util.js"></script>
            <!--[if lte IE 8]><script
src="../plantillaWeb/assets/js/ie/respond.min.js"></script><![endif]-->
            <script src="../plantillaWeb/assets/js/main.js"></script>

    </body>
</html>
```

3.2 DETALLES DISPOSITIVOS

```
<!DOCTYPE HTML>
<!--
    Prism by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license
(templated.co/license)
-->

<html>
    <head>
        <title>Datos Humedad</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1"
/>
        <!--[if lte IE 8]><script
src="../../plantillaWeb/assets/js/ie/html5shiv.js"></script><![endif]-->
```

```
        <link rel="stylesheet"
href="../../plantillaWeb/assets/css/main.css" />
        <!--[if lte IE 9]><link rel="stylesheet"
href="../../plantillaWeb/assets/css/ie9.css" /><![endif]-->
    </head>
    <body>

        <!-- One -->
            <section id="one" class="wrapper">
                <div class="inner split">
                    <section>
                        <h2>||<a href="http://hydrotecna.com/TFM/">
Inicio</a></h2>
                        <h2>Datos de humedad  </h2>
                        <ul class="checklist">
                            <h4><li><a href="#graf1">Histórico cantidad de
agua acumulada</a></li></h4>
                            <h4><li><a href="#graf2">Agua acumulada por
sondas</a></li></h4>
                            <h4><li><a href="#statistics">Estadísticas
básicas</a></li></h4>
                        </ul>
                    </section>
                    <section>
                    <h4><p> </p></h4>
                    <h4><p>MUNICIPIO </p></h4>
                        <ul class="checklist">
                            <li>Polígono: POLIGONO</li>
                            <li>Parcela: PARCELA</li>
                        </ul>
                    </section>
                </div>

                <!--<form style="text-align:left-side;" action=CSV>
                        <input type="submit" value="Descargar datos" />
                </form>-->
            </section>

        <!-- Two -->
            <section id="two" class="wrapper style2 alt">
                <div id="graf1">
                    <h3>Histórico cantidad de agua acumulada sondas 1 y
2:</h3>
                        <div class="image">
                            <img src=ACUMULADO0 alt="HumedadAcumulada"
width="100%"/>
                        </div>
                </div>
                    <h3>Histórico cantidad de agua acumulada sondas 3 y
4:</h3>
                        <div class="image">
                            <img src=ACUMULADO1 alt="HumedadAcumulada"
width="100%"/>
                        </div>
                <div id="graf2">
                    <h3>Humedad por sondas:</h3>
                        <div class="image">
                            <img src=SUBGRAFICO0 alt="Sondas"
width="100%"/>
                        </div>
                        <div class="image">
```

```
                            <img src=SUBGRAFICO1 alt="Sondas"
width="100%"/>
                        </div>
                </div>


    <!-- three -->
        <section id="three" class="wrapper">
                <div class="inner split" id="statistics">
                    <section>
                        <h3><p>Estadísticas básicas 1.</p></h3>
                            <ul class="checklist">
                                <li><strong>Registros:</strong> count0
ynbsp   ||    <strong>Media:</strong>
mean0</li>
                                <li><strong>Desviación estándar:</strong>
std0 ynbsp   ||    <strong>Mínimo:</strong>
min0</li>
                                <li><strong>Máximo:</strong> max0
ynbsp   ||    <strong>Percentil 25%:</strong>
25%0</li>
                                <li><strong>Percentil 50%:</strong> 50%0
ynbsp   ||    <strong>Percentil 75%:</strong>
75%0</li>
                            </ul>
                    </section>
                    <section>
                        <h3><p>Estadísticas básicas 2.</p></h3>
                            <ul class="checklist">
                                <li><strong>Registros:</strong> count1
ynbsp   ||    <strong>Media:</strong>
mean1</li>
                                <li><strong>Desviación estándar:</strong>
std1 ynbsp   ||    <strong>Mínimo:</strong>
min1</li>
                                <li><strong>Máximo:</strong> max1
ynbsp   ||    <strong>Percentil 25%:</strong>
25%1</li>
                                <li><strong>Percentil 50%:</strong> 50%1
ynbsp   ||    <strong>Percentil 75%:</strong>
75%1</li>
                            </ul>
                    </section>
                </div>
            </section>
                    <ul class="actions special">
                        <li><a href="#" class="button alt">Ir a
arriba</a></li>
                    </ul>
            </section>

        <!-- Scripts -->
            <script
src="../../plantillaWeb/assets/js/jquery.min.js"></script>
            <script
src="../../plantillaWeb/assets/js/skel.min.js"></script>
            <script src="../../plantillaWeb/assets/js/util.js"></script>
            <!--[if lte IE 8]><script
src="../plantillaWeb/assets/js/ie/respond.min.js"></script><![endif]-->
            <script src="../../plantillaWeb/assets/js/main.js"></script>
```

```
    </body>
</html>
```

3.3 VISOR WEB

```
<!DOCTYPE HTML>
<!--
    Prism by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license
(templated.co/license) ynbsp
-->

<html>
    <head>
        <title>Visor Web</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1"
/>
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
         <link rel="stylesheet" href="examples.css" />
        <!--[if lte IE 8]><script
src="../../plantillaWeb/assets/js/ie/html5shiv.js"></script><![endif]-->
        <link rel="stylesheet"
href="../../plantillaWeb/assets/css/main.css" />
        <!--[if lte IE 9]><link rel="stylesheet"
href="../../plantillaWeb/assets/css/ie9.css" /><![endif]-->
    </head>
    <body>

        <!-- One -->
        <section id="one" style="height:15%; width:100%;vertical-
align:middle; line-height: 50%; background:#000000">
                <div class="inner split" style="height:15%;
width:100%;vertical-align:middle; line-height: 50%; background:#000000">
                    <section>
                        <h2><a href="http://hydrotecna.com/TFM/"> Inicio
</a>|| Visor web Agrotech</h2>
                    </section>
            </div>
        </section>
        <!-- Two -->
        <div style="height:85%; width:100%; background:#000000">
                 <script src="//sitna.tracasa.es/api/"
type="text/javascript"></script>
                <div id="mapa"></div>
                <script>
             // Crear un mapa con las opciones por defecto.
                    var map = new SITNA.Map("mapa");

                    // Cuando esté todo cargado proceder a trabajar con el
mapa.
                    map.loaded(function () {
                        // Añadir al mapa un documento KML
                        map.addLayer({
                            id: "capa_kml",
                            title: "Dispositivos",
                            type: SITNA.Consts.layerType.VECTOR,
```

```
                                url: "dispositivos.kml"

                            });
                        });
                        //map.zoomToLayer();
                    </script>
                </div>


            <!-- Scripts -->
                <script
src="../../plantillaWeb/assets/js/jquery.min.js"></script>
                <script
src="../../plantillaWeb/assets/js/skel.min.js"></script>
                <script src="../../plantillaWeb/assets/js/util.js"></script>
                <!--[if lte IE 8]><script
src="../../plantillaWeb/assets/js/ie/respond.min.js"></script><![endif]-->
                <script src="../../plantillaWeb/assets/js/main.js"></script>

        </body>
</html>
```

ANEXOS

ANEXO II. BASES DE DATOS

ÍNDICE ANEXO 2

1. INTRODUCCIÓN

Seguidamente se muestra en el código SQL de la estructura de la base de datos, es decir, los esquemas, las tablas con sus atributos, los tipos de atributos, así como las relaciones entre las tablas y sus restricciones. No se incluye información sobre los datos almacenados.

2. CÓDIGO SQL DE LA ESTRUCTURA DE LA BASE DE DATOS

```
CREATE SCHEMA control


ALTER SCHEMA control OWNER TO postgres;

--
-- TOC entry 9 (class 2615 OID 18188)
-- Name: iot; Type: SCHEMA; Schema: -; Owner: postgres
--

CREATE SCHEMA iot;


ALTER SCHEMA iot OWNER TO postgres;

SET search_path = control, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- TOC entry 201 (class 1259 OID 18265)
-- Name: procesado_datos; Type: TABLE; Schema: control; Owner: postgres;
Tablespace:
--

CREATE TABLE procesado_datos (
    idprocesado integer NOT NULL,
    fecha date NOT NULL,
    idreceptor integer NOT NULL
);


ALTER TABLE procesado_datos OWNER TO postgres;

--
-- TOC entry 3277 (class 0 OID 0)
-- Dependencies: 201
-- Name: TABLE procesado_datos; Type: COMMENT; Schema: control; Owner:
postgres
--

COMMENT ON TABLE procesado_datos IS 'Control de fecha de procesado de datos
de humedad.';


--
```

```
-- TOC entry 3278 (class 0 OID 0)
-- Dependencies: 201
-- Name: COLUMN procesado_datos.idprocesado; Type: COMMENT; Schema:
control; Owner: postgres
--

COMMENT ON COLUMN procesado_datos.idprocesado IS 'Identificador del
procedado de datos (autonumérico).';


--
-- TOC entry 3279 (class 0 OID 0)
-- Dependencies: 201
-- Name: COLUMN procesado_datos.fecha; Type: COMMENT; Schema: control;
Owner: postgres
--

COMMENT ON COLUMN procesado_datos.fecha IS 'Fecha de procesado.';


--
-- TOC entry 3280 (class 0 OID 0)
-- Dependencies: 201
-- Name: COLUMN procesado_datos.idreceptor; Type: COMMENT; Schema: control;
Owner: postgres
--

COMMENT ON COLUMN procesado_datos.idreceptor IS 'FK a la tabla de datos de
los receptores.';


--
-- TOC entry 200 (class 1259 OID 18263)
-- Name: procesado_datos_idprocesado_seq; Type: SEQUENCE; Schema: control;
Owner: postgres
--

CREATE SEQUENCE procesado_datos_idprocesado_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE procesado_datos_idprocesado_seq OWNER TO postgres;

--
-- TOC entry 3281 (class 0 OID 0)
-- Dependencies: 200
-- Name: procesado_datos_idprocesado_seq; Type: SEQUENCE OWNED BY; Schema:
control; Owner: postgres
--

ALTER SEQUENCE procesado_datos_idprocesado_seq OWNED BY
procesado_datos.idprocesado;


SET search_path = iot, pg_catalog;

--
```

```
-- TOC entry 199 (class 1259 OID 18246)
-- Name: datos_humedad; Type: TABLE; Schema: iot; Owner: postgres;
Tablespace:
--

CREATE TABLE datos_humedad (
    idhumedad integer NOT NULL,
    fecha timestamp without time zone NOT NULL,
    sonda1 integer NOT NULL,
    sonda2 integer NOT NULL,
    sonda3 integer NOT NULL,
    sonda4 integer NOT NULL,
    vwc_sonda1 numeric(10,8),
    vwc_sonda2 numeric(10,8),
    vwc_sonda3 numeric(10,8),
    vwc_sonda4 numeric(10,8),
    acumulado numeric(10,5),
    bateria integer NOT NULL,
    idreceptor integer NOT NULL,
    CONSTRAINT if_sonda1_then_vwc1_is_not_null CHECK (((sonda1 IS NOT NULL)
OR (vwc_sonda1 IS NOT NULL))),
    CONSTRAINT if_sonda2_then_vwc2_is_not_null CHECK (((sonda2 IS NOT NULL)
OR (vwc_sonda2 IS NOT NULL))),
    CONSTRAINT if_sonda3_then_vwc3_is_not_null CHECK (((sonda3 IS NOT NULL)
OR (vwc_sonda3 IS NOT NULL))),
    CONSTRAINT if_sonda4_then_vwc4_is_not_null CHECK (((sonda4 IS NOT NULL)
OR (vwc_sonda4 IS NOT NULL)))
);


ALTER TABLE datos_humedad OWNER TO postgres;

--
-- TOC entry 3282 (class 0 OID 0)
-- Dependencies: 199
-- Name: TABLE datos_humedad; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON TABLE datos_humedad IS 'Lecturas de humedad de los datologger y
resultados derivados.';


--
-- TOC entry 3283 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.idhumedad; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.idhumedad IS 'Identificador único de
lectura (autonumérico).';


--
-- TOC entry 3284 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.fecha; Type: COMMENT; Schema: iot; Owner:
postgres
--
```

```
COMMENT ON COLUMN datos_humedad.fecha IS 'Fecha de toma y envío de
lectura.';


--
-- TOC entry 3285 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.sonda1; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.sonda1 IS 'Lectura de humedad en mW de la
sonda 1.';


--
-- TOC entry 3286 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.sonda2; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.sonda2 IS 'Lectura de humedad en mW de la
sonda 2.';


--
-- TOC entry 3287 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.sonda3; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.sonda3 IS 'Lectura de humedad en mW de la
sonda 3.';


--
-- TOC entry 3288 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.sonda4; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.sonda4 IS 'Lectura de humedad en mW de la
sonda 4.';


--
-- TOC entry 3289 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.vwc_sonda1; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN datos_humedad.vwc_sonda1 IS 'Lectura de humedad en mm de
la sonda 1.';


--
-- TOC entry 3290 (class 0 OID 0)
```

```
-- Dependencies: 199
-- Name: COLUMN datos_humedad.vwc_sonda2; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN datos_humedad.vwc_sonda2 IS 'Lectura de humedad en mm de
la sonda 2.';


--
-- TOC entry 3291 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.vwc_sonda3; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN datos_humedad.vwc_sonda3 IS 'Lectura de humedad en mm de
la sonda 3.';


--
-- TOC entry 3292 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.vwc_sonda4; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN datos_humedad.vwc_sonda4 IS 'Lectura de humedad en mm de
la sonda 4.';


--
-- TOC entry 3293 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.acumulado; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.acumulado IS 'Acumulación de humedad en mm
en el suelo.';


--
-- TOC entry 3294 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.bateria; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN datos_humedad.bateria IS 'Nivel de bateria del
receptor.';


--
-- TOC entry 3295 (class 0 OID 0)
-- Dependencies: 199
-- Name: COLUMN datos_humedad.idreceptor; Type: COMMENT; Schema: iot;
Owner: postgres
--
```

```
COMMENT ON COLUMN datos_humedad.idreceptor IS 'FK a la tabla de datos de
los receptores.';


--
-- TOC entry 198 (class 1259 OID 18244)
-- Name: datos_humedad_idhumedad_seq; Type: SEQUENCE; Schema: iot; Owner:
postgres
--

CREATE SEQUENCE datos_humedad_idhumedad_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE datos_humedad_idhumedad_seq OWNER TO postgres;

--
-- TOC entry 3296 (class 0 OID 0)
-- Dependencies: 198
-- Name: datos_humedad_idhumedad_seq; Type: SEQUENCE OWNED BY; Schema: iot;
Owner: postgres
--

ALTER SEQUENCE datos_humedad_idhumedad_seq OWNED BY
datos_humedad.idhumedad;


--
-- TOC entry 193 (class 1259 OID 18197)
-- Name: parcelas; Type: TABLE; Schema: iot; Owner: postgres; Tablespace:
--

CREATE TABLE parcelas (
    idparcela integer NOT NULL,
    codmunicipio integer NOT NULL,
    poligono integer NOT NULL,
    parcela integer NOT NULL,
    subparcela integer,
    superficie numeric(15,2),
    agua_disponible_real numeric(3,2),
    recarga numeric(3,2) DEFAULT 0.30 NOT NULL,
    idsuelo integer NOT NULL,
    geom public.geometry(MultiPolygon,25830) NOT NULL,
    municipio character varying(50)
);


ALTER TABLE parcelas OWNER TO postgres;

--
-- TOC entry 3297 (class 0 OID 0)
-- Dependencies: 193
-- Name: TABLE parcelas; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON TABLE parcelas IS 'Parcelas catastrales.';
```

```
--
-- TOC entry 3298 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.idparcela; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.idparcela IS 'Identificador único de parcela
(autonumérico).';


--
-- TOC entry 3299 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.codmunicipio; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.codmunicipio IS 'Código del municipio de la
parcela.';


--
-- TOC entry 3300 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.poligono; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.poligono IS 'Número de polígono catastral de la
parcela en el municipio.';


--
-- TOC entry 3301 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.parcela; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.parcela IS 'Numero de parcela dentro del
polígono.';


--
-- TOC entry 3302 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.subparcela; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.subparcela IS 'Numero de subparcela dentro del
polígono.';


--
-- TOC entry 3303 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.superficie; Type: COMMENT; Schema: iot; Owner:
postgres
```

```
--

COMMENT ON COLUMN parcelas.superficie IS 'Superficie de la parcela.';


--
-- TOC entry 3304 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.agua_disponible_real; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN parcelas.agua_disponible_real IS 'CapacidadCampo-
PuntoMarchitez de la parcela, por defecto la de la tabla de suelos.';


--
-- TOC entry 3305 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.recarga; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN parcelas.recarga IS 'Porcentaje de estrés hídrico
permitido.';


--
-- TOC entry 3306 (class 0 OID 0)
-- Dependencies: 193
-- Name: COLUMN parcelas.geom; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON COLUMN parcelas.geom IS 'Geomentría de la parcela.';


--
-- TOC entry 192 (class 1259 OID 18195)
-- Name: parcelas_idparcela_seq; Type: SEQUENCE; Schema: iot; Owner:
postgres
--

CREATE SEQUENCE parcelas_idparcela_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE parcelas_idparcela_seq OWNER TO postgres;

--
-- TOC entry 3307 (class 0 OID 0)
-- Dependencies: 192
-- Name: parcelas_idparcela_seq; Type: SEQUENCE OWNED BY; Schema: iot;
Owner: postgres
--

ALTER SEQUENCE parcelas_idparcela_seq OWNED BY parcelas.idparcela;
```

```
--
-- TOC entry 195 (class 1259 OID 18214)
-- Name: receptores; Type: TABLE; Schema: iot; Owner: postgres; Tablespace:
--

CREATE TABLE receptores (
    idreceptor integer NOT NULL,
    canal integer NOT NULL,
    read_api character varying(20) NOT NULL,
    grupo integer DEFAULT 0 NOT NULL,
    num_sondas integer NOT NULL,
    prof_sonda1 numeric(4,1),
    prof_sonda2 numeric(4,1),
    prof_sonda3 numeric(4,1),
    prof_sonda4 numeric(4,1),
    descripcion character varying(250),
    upload_dir character varying(30),
    idparcela integer NOT NULL,
    nombre character varying(30)
);


ALTER TABLE receptores OWNER TO postgres;

--
-- TOC entry 3308 (class 0 OID 0)
-- Dependencies: 195
-- Name: TABLE receptores; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON TABLE receptores IS 'receptores de humedad.';


--
-- TOC entry 3309 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.idreceptor; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.idreceptor IS 'Identificador único de los
receptores (autonumérico).';


--
-- TOC entry 3310 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.canal; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.canal IS 'Canal del dispositivo en plataforma
Think Speak.';


--
-- TOC entry 3311 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.read_api; Type: COMMENT; Schema: iot; Owner:
postgres
```

```
--

COMMENT ON COLUMN receptores.read_api IS 'Read API key del dispositivo en
plataforma Think Speak.';


--
-- TOC entry 3312 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.grupo; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.grupo IS 'Permite descriminar entre sondas de
un mismo receptor 0 (todas las sondas agrupadas).';


--
-- TOC entry 3313 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.descripcion; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.descripcion IS 'Observaciones de los
receptores.';


--
-- TOC entry 3314 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.upload_dir; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.upload_dir IS 'Dirección de subida de datos
del servidor.';


--
-- TOC entry 3315 (class 0 OID 0)
-- Dependencies: 195
-- Name: COLUMN receptores.idparcela; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN receptores.idparcela IS 'FK a la parcela en la que se
ubica el receptor.';


--
-- TOC entry 194 (class 1259 OID 18212)
-- Name: receptores_idreceptor_seq; Type: SEQUENCE; Schema: iot; Owner:
postgres
--

CREATE SEQUENCE receptores_idreceptor_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
```

```
    CACHE 1;


ALTER TABLE receptores_idreceptor_seq OWNER TO postgres;

--
-- TOC entry 3316 (class 0 OID 0)
-- Dependencies: 194
-- Name: receptores_idreceptor_seq; Type: SEQUENCE OWNED BY; Schema: iot;
Owner: postgres
--

ALTER SEQUENCE receptores_idreceptor_seq OWNED BY receptores.idreceptor;


--
-- TOC entry 191 (class 1259 OID 18190)
-- Name: suelos; Type: TABLE; Schema: iot; Owner: postgres; Tablespace:
--

CREATE TABLE suelos (
    idsuelo integer NOT NULL,
    tipo character varying(25) NOT NULL,
    punto_marchitez numeric(3,2) NOT NULL,
    capacidad_campo numeric(3,2) NOT NULL,
    agua_disponible numeric(3,2) NOT NULL
);


ALTER TABLE suelos OWNER TO postgres;

--
-- TOC entry 3317 (class 0 OID 0)
-- Dependencies: 191
-- Name: TABLE suelos; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON TABLE suelos IS 'Diferentes tipos de estructura del suelo de las
parcelas.';


--
-- TOC entry 3318 (class 0 OID 0)
-- Dependencies: 191
-- Name: COLUMN suelos.tipo; Type: COMMENT; Schema: iot; Owner: postgres
--

COMMENT ON COLUMN suelos.tipo IS 'tipos de estructura de suelo.';


--
-- TOC entry 3319 (class 0 OID 0)
-- Dependencies: 191
-- Name: COLUMN suelos.punto_marchitez; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN suelos.punto_marchitez IS 'Valor a partir del cual la
planta no puede utilizar el agua del suelo.';
```

```
--
-- TOC entry 3320 (class 0 OID 0)
-- Dependencies: 191
-- Name: COLUMN suelos.capacidad_campo; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN suelos.capacidad_campo IS 'Capacidad máxima de retención
de agua del suelo.';


--
-- TOC entry 3321 (class 0 OID 0)
-- Dependencies: 191
-- Name: COLUMN suelos.agua_disponible; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON COLUMN suelos.agua_disponible IS 'Capacidad de campo - punto de
marchitez.';


--
-- TOC entry 197 (class 1259 OID 18230)
-- Name: ubicacion_receptores; Type: TABLE; Schema: iot; Owner: postgres;
Tablespace:
--

CREATE TABLE ubicacion_receptores (
    idubicacion integer NOT NULL,
    idreceptor integer NOT NULL,
    fecha_desde date NOT NULL,
    fecha_hasta date,
    geom public.geometry(Point,25830) NOT NULL
);


ALTER TABLE ubicacion_receptores OWNER TO postgres;

--
-- TOC entry 3322 (class 0 OID 0)
-- Dependencies: 197
-- Name: TABLE ubicacion_receptores; Type: COMMENT; Schema: iot; Owner:
postgres
--

COMMENT ON TABLE ubicacion_receptores IS 'Ubicación histórica de
receptores.';


--
-- TOC entry 3323 (class 0 OID 0)
-- Dependencies: 197
-- Name: COLUMN ubicacion_receptores.idreceptor; Type: COMMENT; Schema:
iot; Owner: postgres
--

COMMENT ON COLUMN ubicacion_receptores.idreceptor IS 'Identificador único
de los receptores.';
```

```
--
-- TOC entry 3324 (class 0 OID 0)
-- Dependencies: 197
-- Name: COLUMN ubicacion_receptores.fecha_desde; Type: COMMENT; Schema:
iot; Owner: postgres
--

COMMENT ON COLUMN ubicacion_receptores.fecha_desde IS 'Fecha de instalación
en parcela.';


--
-- TOC entry 3325 (class 0 OID 0)
-- Dependencies: 197
-- Name: COLUMN ubicacion_receptores.fecha_hasta; Type: COMMENT; Schema:
iot; Owner: postgres
--

COMMENT ON COLUMN ubicacion_receptores.fecha_hasta IS 'Fecha de retirada de
parcela.';


--
-- TOC entry 3326 (class 0 OID 0)
-- Dependencies: 197
-- Name: COLUMN ubicacion_receptores.geom; Type: COMMENT; Schema: iot;
Owner: postgres
--

COMMENT ON COLUMN ubicacion_receptores.geom IS 'Posición de receptor en UTM
/ ETRS 89.';


--
-- TOC entry 196 (class 1259 OID 18228)
-- Name: ubicacion_receptores_idubicacion_seq; Type: SEQUENCE; Schema: iot;
Owner: postgres
--

CREATE SEQUENCE ubicacion_receptores_idubicacion_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE ubicacion_receptores_idubicacion_seq OWNER TO postgres;

--
-- TOC entry 3327 (class 0 OID 0)
-- Dependencies: 196
-- Name: ubicacion_receptores_idubicacion_seq; Type: SEQUENCE OWNED BY;
Schema: iot; Owner: postgres
--

ALTER SEQUENCE ubicacion_receptores_idubicacion_seq OWNED BY
ubicacion_receptores.idubicacion;


SET search_path = control, pg_catalog;
```

```
--
-- TOC entry 3135 (class 2604 OID 18268)
-- Name: idprocesado; Type: DEFAULT; Schema: control; Owner: postgres
--

ALTER TABLE ONLY procesado_datos ALTER COLUMN idprocesado SET DEFAULT
nextval('procesado_datos_idprocesado_seq'::regclass);


SET search_path = iot, pg_catalog;

--
-- TOC entry 3130 (class 2604 OID 18249)
-- Name: idhumedad; Type: DEFAULT; Schema: iot; Owner: postgres
--

ALTER TABLE ONLY datos_humedad ALTER COLUMN idhumedad SET DEFAULT
nextval('datos_humedad_idhumedad_seq'::regclass);


--
-- TOC entry 3125 (class 2604 OID 18200)
-- Name: idparcela; Type: DEFAULT; Schema: iot; Owner: postgres
--

ALTER TABLE ONLY parcelas ALTER COLUMN idparcela SET DEFAULT
nextval('parcelas_idparcela_seq'::regclass);


--
-- TOC entry 3127 (class 2604 OID 18217)
-- Name: idreceptor; Type: DEFAULT; Schema: iot; Owner: postgres
--

ALTER TABLE ONLY receptores ALTER COLUMN idreceptor SET DEFAULT
nextval('receptores_idreceptor_seq'::regclass);


--
-- TOC entry 3129 (class 2604 OID 18233)
-- Name: idubicacion; Type: DEFAULT; Schema: iot; Owner: postgres
--

ALTER TABLE ONLY ubicacion_receptores ALTER COLUMN idubicacion SET DEFAULT
nextval('ubicacion_receptores_idubicacion_seq'::regclass);


SET search_path = control, pg_catalog;

--
-- TOC entry 3151 (class 2606 OID 18270)
-- Name: pk_procesado; Type: CONSTRAINT; Schema: control; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY procesado_datos
    ADD CONSTRAINT pk_procesado PRIMARY KEY (idprocesado);


SET search_path = iot, pg_catalog;
```

```
--
-- TOC entry 3147 (class 2606 OID 18255)
-- Name: pk_humedad; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY datos_humedad
    ADD CONSTRAINT pk_humedad PRIMARY KEY (idhumedad);


--
-- TOC entry 3141 (class 2606 OID 18220)
-- Name: pk_idreceptor; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY receptores
    ADD CONSTRAINT pk_idreceptor PRIMARY KEY (idreceptor);


--
-- TOC entry 3137 (class 2606 OID 18194)
-- Name: pk_idsuelo; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY suelos
    ADD CONSTRAINT pk_idsuelo PRIMARY KEY (idsuelo);


--
-- TOC entry 3145 (class 2606 OID 18238)
-- Name: pk_idubicacion; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY ubicacion_receptores
    ADD CONSTRAINT pk_idubicacion PRIMARY KEY (idubicacion);


--
-- TOC entry 3139 (class 2606 OID 18206)
-- Name: pk_parcelas; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY parcelas
    ADD CONSTRAINT pk_parcelas PRIMARY KEY (idparcela);


--
-- TOC entry 3149 (class 2606 OID 18257)
-- Name: unico; Type: CONSTRAINT; Schema: iot; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY datos_humedad
    ADD CONSTRAINT unico UNIQUE (fecha, idreceptor);


--
```

```
-- TOC entry 3143 (class 2606 OID 18222)
-- Name: unico_receptores; Type: CONSTRAINT; Schema: iot; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY receptores
    ADD CONSTRAINT unico_receptores UNIQUE (canal, read_api, grupo);


SET search_path = control, pg_catalog;

--
-- TOC entry 3156 (class 2606 OID 18271)
-- Name: fk_procesado_receptores; Type: FK CONSTRAINT; Schema: control;
Owner: postgres
--

ALTER TABLE ONLY procesado_datos
    ADD CONSTRAINT fk_procesado_receptores FOREIGN KEY (idreceptor)
REFERENCES iot.receptores(idreceptor);


SET search_path = iot, pg_catalog;

--
-- TOC entry 3155 (class 2606 OID 18258)
-- Name: fk_humedad_receptores; Type: FK CONSTRAINT; Schema: iot; Owner:
postgres
--

ALTER TABLE ONLY datos_humedad
    ADD CONSTRAINT fk_humedad_receptores FOREIGN KEY (idreceptor)
REFERENCES receptores(idreceptor);
--
-- TOC entry 3152 (class 2606 OID 18207)
-- Name: fk_parcelas_suelos; Type: FK CONSTRAINT; Schema: iot; Owner:
postgres
--
ALTER TABLE ONLY parcelas
    ADD CONSTRAINT fk_parcelas_suelos FOREIGN KEY (idsuelo) REFERENCES
suelos(idsuelo);
--
-- TOC entry 3153 (class 2606 OID 18223)
-- Name: fk_receptores_parcelas; Type: FK CONSTRAINT; Schema: iot; Owner:
postgres
--
ALTER TABLE ONLY receptores
    ADD CONSTRAINT fk_receptores_parcelas FOREIGN KEY (idparcela)
REFERENCES parcelas(idparcela);
--
-- TOC entry 3154 (class 2606 OID 18239)
-- Name: fk_receptores_receptores; Type: FK CONSTRAINT; Schema: iot; Owner:
postgres
--

ALTER TABLE ONLY ubicacion_receptores
    ADD CONSTRAINT fk_receptores_receptores FOREIGN KEY (idreceptor)
REFERENCES receptores(idreceptor);
```

ANEXOS

ANEXO III. ANÁLISIS PYTHON DASH

ÍNDICE ANEXO 3

1. INTRODUCCIÓN

En este anexo se incluye el primer análisis realizado con Python Dash, para la creación de gráficos dinámicos de medición de humedad del suelo.

Dash es una herramientas programada en Python que sirve para construir aplicaciones web analíticas en las que no se requiere programación en JavaScript.

Así, se presenta el código de progrmación llevado a cabo y un ejemplo de su funcionamiento.

2. CÓDIGO PYTHON DASH

```
import dash
from dash.dependencies import Input, Output
import dash_core_components as dcc
import dash_html_components as html
from pandas_datareader import data as web
from datetime import datetime as dt
import pandas as pd

filePath = '/Resultados/Datos_Sondas_BBDD.csv'

#----Definir características de las serie temporal
df = pd.read_csv(filePath) #leer el csv
df.columns = ['Fecha','VWC_Sonda1',
'VWC_Sonda2','VWC_Sonda3','VWC_Sonda4','acumulado'] #Definir cabecera
df['Fecha'] = pd.to_datetime(df['Fecha']) #Convertir la columna fecha en
DateTime
df.set_index('Fecha', inplace=True)
#print(df.head())

#-----INICIO DE DASH
app = dash.Dash()

#Estilo del gráfico
colors = {
    'background': '#111111',
    'text': '#7FDBFF',
    'otro1':'#111111'
}


app.layout = html.Div(style={'backgroundColor': colors['background']},
children=[
    html.H1(
        children='Sondas humedad',
        style={
            'textAlign': 'center',
            'color': colors['text']
                }
        ),
    dcc.Checklist(
        style={
            'color': colors['text']
                },
```

```
        id='my-checklist',
        options=[
            {'label': 'Sonda 1', 'value': 'VWC_Sonda1'},
            {'label': 'Sonda 2', 'value': 'VWC_Sonda2'},
            {'label': 'Sonda 3', 'value': 'VWC_Sonda3'},
            {'label': 'Acumulado', 'value': 'acumulado'},
            {'label': 'Bateria', 'value': 'Bateria'}
        ],
        #multi=True,
        values=['VWC_Sonda1', 'VWC_Sonda2', 'VWC_Sonda3','acumulado'],
        labelStyle={'display': 'inline-block'}
    ),
    dcc.Graph(id='output-graph'),
])


@app.callback(
    Output(component_id='output-graph', component_property='figure'),
    [Input(component_id='my-checklist', component_property='values')]
)
def update_value(selected_checklist_value):

    traces = []
    for sondas in selected_checklist_value:
        trace = {'x': df.index, 'y': df[sondas],'name': sondas}
        traces.append(trace)

    return {
        'data': traces,

        'layout': {
            'title': selected_checklist_value,
            'yaxis': dict(range=[0,0.5]),
            'plot_bgcolor': colors['otro1'],
            'paper_bgcolor': colors['background'],
            'font': {
                'color': colors['text']
                }
            }
        }

if __name__ == '__main__':
    app.run_server()
```

3. EJEMPLO USO PYTHON DASH

Figura. 1. Ejemplo visualización de datos de humedad con Python Dash.