Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo de una API para datos abiertos

Development of an API for Open Data

La Laguna, 3 de marzo de 2018

D. José Luis Roda García, con N.I.F. 43.356.123-L profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Desarrollo de una Api para datos abiertos"

ha sido realizada bajo su dirección por Dña. **Núria Gonzalo Soto**,
con N.I.F. 40.360.804-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 de marzo de 2018

Agradecimientos

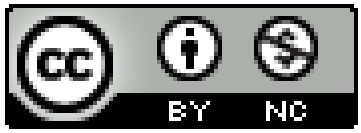
Primero agradecerles a mis padres y a mi hermana la oportunidad de permitirme el lujo de estudiar donde yo que querido y apoyarme siempre.

A mis compañeros y amigos de la carrera que actuasen de padres cuando los míos no estaban cerca.

A mi pareja por apoyarme en momentos de estrés y agobio y no rendirse conmigo y darme la energía necesaria.

A mi tutor, por haberme ayudado en la finalización de este proyecto y de mis estudios y por la paciencia que ha tenido conmigo y el proyecto.

Licencia



Resumen

El objetivo de este trabajo ha sido profundizar en los términos que engloban a los datos abiertos proporcionados por las administraciones públicas y de su situación actual y futura, así como entender el concepto e influencias de la creación de APIs.

En los primeros puntos se mostrarán conceptos que se deben entender para poder llegar a realizar el desarrollo de la API, nociones como la definición de Mashups, API y Open Data; una vez se han adquirido estos conocimientos existen otros conceptos interesantes que forman parte de los tres primeros como: API REST, API Economy, fintech, los principios de programación de una API o como se puede entender el Open Data como negocio. En los siguientes puntos se hará referencia al ámbito de actuación que se va a desarrollar este proyecto; se realizará un análisis de los portales del Gobierno de Canarias (calidad del aire) y de la AEMET lo cuales nos proporcionaran los datos. En este análisis se detallarán las dificultades encontradas para poder obtener los datos y las soluciones realizadas a cada problema.

Como punto final se especificará todo el proceso de desarrollo de la API; es decir, se documentará desde los métodos que son necesarios para hacer la captación de los datos previos a la realización de la propia API hasta la visualización de su estructura.

***Palabras clave**: Mashups, API Economy, Calidad del aire, contaminantes, AEMET, Python.*

Abstract

The objective of this work has been to deepen the terms that they include to open data provided by public administrations and their current and future situation, as well as understanding the concept and influences of the creation of APIs.

First points will show the concepts that must be understood to be able to get to the development of the API; conceptions such as the definition of Mashups, API and Open Data; after knowledge has been acquired there exist other interesting concepts that form respect to the first three like: REST API, API Economy, fintech, the principles of programming an API or how you can understand Open Data as a business. At the following points, one will refer to the area of performance that is going to develop this project; An analysis of the portals of the Government of the Canary Islands (air quality) and the AEMET will be carried out, which will provide us with the data. In this analysis, the difficulties encountered in obtaining the data and the solutions made to each problem will be detailed.

Finally, the entire API development process will be specified: I mean, it will be documented from the methods that are necessary to capture the data prior to the realization of the API itself until the visualization of its structure.

***Palabras clave**: Mashups, API Economy, air quality, contamination, AEMET, Python.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

En este apartado se verán, de una forma breve, los conceptos previos más importantes que se deben conocer para la realización de este proyecto.

1.1 Mashups

El concepto Mashup de software, proviene de un término musical en inglés que significa la creación de una canción a partir de la mezcla o trozos de otras canciones.

Los mashups son considerados como un producto de la Web 2.0. Consisten en una aplicación web que utiliza varios recursos, mezclándolos entre sí, para crear un servicio completo[4]. Normalmente se interactúa con el contenido a través de una interfaz pública o API.

Además, existen otros tipos de métodos para la obtención de contenido como son las Web Feeds (RSS o Atom) o screen scraping.

Los mashups se pueden clasificar en tres tipos:

- Mashups de consumidores.
- Mashups de datos.
- Mashups empresariales.

Un ejemplo de tipo empresarial, es Magic [27], esta empresa proporciona servicios como: acceso rápido y fácil de datos de misión crítica o visualización de datos Rich(aquellos datos que destacan frente a los demás. Un ejemplo de ellos, serían los datos destacados de una búsqueda o el tema más tuiteado).

1.1.1 Mashups de datos o Data Mashups

Los Data Mashups se caracterizan por unir los datos de dos o más fuentes y presentarlos de una forma única.

Su importancia destaca en la amplia disposición de datos que se encuentra en Internet. Se podría considerar que Internet es un gran repositorio de información. Este almacén de datos es muy valorado por las empresas, ya que aporta un mejor conocimiento de los clientes, productos, competencia, etc. Sin embargo, la mayor parte de toda esta información no está estructurada por lo que es una información difícil de manejar, de una forma automatizada y útil.

A consecuencia del valor de estos datos en las empresas apareció el concepto Enterprise Data Mashups[5]. Este tipo de data mashup permite integrar en los propios sistemas de información de la empresa formatos como: no estructurado (texto libre, foros, emails, etc.), estructurado (bases de datos, JSON, CSV, etc.) o semiestructurado (páginas web dinámicas, documentos con estructura no explícita, etc.).

1.2 ¿Qué es una API?

En términos más generales, una API es una Interfaz de Programación de Aplicaciones (Application Programming Interface), es decir, es un conjunto de funciones, procedimientos o clases que un sistema operativo, librería o servicio proporciona para soportar peticiones realizadas por un software. Permite que las funcionalidades de un servicio inicial puedan ser ampliadas a otras, pudiendo integrar otras APIs haciendo que el servicio/software sea extensible.

En conclusión, las APIs facilitan considerablemente la construcción de aplicaciones más complejas; un aspecto que estimula a una nueva generación de soluciones de tipo “quick fix” y aplicaciones cliente que operan en sistemas de mayor extensión [6].

1.2.1 API REST

Una API REST permite la utilización de servicios web disponibles a través de Internet utilizando el protocolo estándar HTTP; en otras palabras, una API REST es un servicio que al igual que el término inicial de una API provee de funciones que dan capacidad de usar un servicio web de terceros dentro de nuestro propio servicio o aplicación propia de un modo seguro.

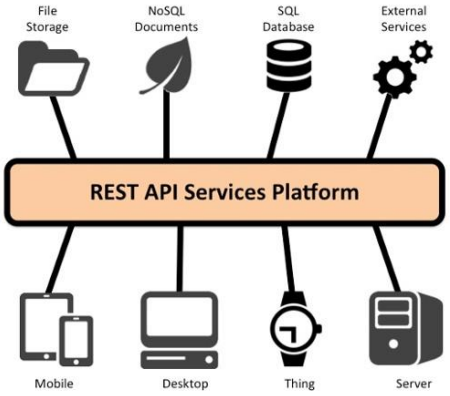


Figura 1: REST API Service[31]

Mediante un método GET se obtiene una respuesta a una consulta en una API REST. El contenido de las respuestas son datos en diferentes formatos como pueden ser: texto plano, ficheros XML, JSON, CSV, etc.

Los servicios de una REST API son los intermediarios entre el contenedor de los datos con la parte.

1.2.2 ¿Por qué son importantes las APIs?

Uno de los primeros puntos que hacen referencia al porqué las APIs son importantes es la relevancia para el desarrollo de aplicaciones web y de las aplicaciones móviles de hoy en día. Gran parte del éxito que han cosechado las aplicaciones móviles ya sea para iPhone como para Android, es gracias a una potente API en el *backend* que realiza el manejo de los datos que a posteriori muestran estas aplicaciones.

Actualmente, se está hablando de una tercera revolución industrial y, por lo tanto, de un nuevo ecosistema económico. Esta nueva era está enfocada a la digitalización de los datos, cada persona genera una gran cantidad de información provocando transferencias intensivas de datos digitales. Los ejemplos más claros de la existencia de una necesidad del desarrollo de recursos informáticos de un modo rápido y ágil son las citadas APIs. A un nivel primario una API permite que exista una comunicación bidireccional entre los productos y los servicios, dependiendo de las características del desarrollo van a satisfacer diferentes necesidades como: el comercio electrónico, las redes sociales o los pagos en línea.

Por lo tanto, en grandes términos, el punto más relevante que encontramos del porqué de la importancia de las APIs es la rapidez de creación de contenido; es decir, del gran volumen de información que se genera cada día para el uso y aprovechamiento de las empresas.

1.3 API Economy

La evolución de las nuevas tecnologías y concretamente hablando de los servicios web y los teléfonos inteligentes (incluyendo otros dispositivos móviles) ha hecho que aparezcan nuevos conceptos relacionados con el mundo de las APIs como: IoT (Internet of Things), API Economy u Open Business, entre otros.

La API Economy redefine el concepto de la API que concluye que ésta puede ser un producto en sí misma con el que comercializar y obtener beneficios. Este cambio de enfoque obliga a gestionar las APIs de un modo que ya no es exclusivo del técnico que lo implementa sino, a todo el conjunto de la empresa y/u organización. Ahora la API debe planificarse y desarrollarse según un modelo de negocio igual que cualquier otro producto o servicio, debe responder de forma clara a unos criterios de diseño, uso, calidad y retorno de la inversión; es decir, han dejado de ser vistas como una simple interfaz pragmática para convertirse en un habilitador de nuevos modelos de negocio y en un factor clave de la transformación digital de las organizaciones.

Las APIs actúan como el pegamento digital que une servicios, aplicaciones y sistemas, esto permite a las empresas sacar el máximo provecho de sus datos para crear experiencias de usuario atractivas y abrir nuevos canales de ingresos. Las necesidades de negocio de las empresas y/u organizaciones determinarán el camino y, probablemente, uno de los siguientes puntos de entrada:

- La creación de valor al ofrecer APIs que los demás quieren/necesitan.
- Utilización de APIs para ayudar a los desarrolladores a innovar libremente.
- Dar soporte al equipo de desarrollo móvil con APIs.
- Hacer APIs con el lenguaje más común e híbrido.
- Integrar dispositivos con datos de IoT.

A través de la adopción de una estrategia de integración cualquier empresa puede complementar las capacidades externas en sus procesos, esto incrementa la innovación de forma dinámica, sencilla y asequible, consiguiendo acelerar el time-to-market para una mejora en la experiencia del cliente. Un ejemplo sencillo es la integración de Google Maps (https://developers.google.com/maps/?hl=es-419), Google permite que cualquier empresa pueda integrar su servicio gratuitamente. Esto refuerza el reconocimiento de su marca e incremento del tráfico de su servicio y, para las empresas, proporcionar una mejor experiencia a sus clientes.

Figura 2: Api Economy experience[8]

Otro ejemplo es el periódico norteamericano The New York Times (https://developer.nytimes.com/) que permite la integración de los contenidos de webs o blogs de terceros. El beneficio es mutuo ya que por parte del diario incrementa su negocio y la otra parte ofrece contenidos de calidad.

Hay empresas que el mayor porcentaje de su negocio proviene principalmente de sus APIs, como por ejemplo eBay, Salesforce o Experian. Pero sin embargo, la mayoría de las organizaciones aún les queda un largo camino para llegar a este nuevo modelo de negocio entorno a las APIs, que cada vez coge más terreno en todos los sectores. Por ello, empresas como IBM han lanzado una serie de propuestas en las que se asesora a los desarrolladores en el tema de importar la Economía de las APIs. En este caso, IBM (http://www-05.ibm.com/es/api-management/discover-api.html) ofrece un conjunto de soluciones para obtener ventajas del uso de datos al igual que herramientas para fomentar la participación y confianza de los clientes.

1.3.1 Fintech

A grosso modo, el término de fintech se podría explicar como el desarrollo de servicios financieros basados en innovación tecnológica.

Prácticamente el monopolio de éste nuevo término lo abarcan las startups, esto se debe a que estas empresas ofrecen servicios financieros al margen de las grandes compañías tradicionales. Y, ¿qué tipo de servicios necesita el mercado financiero? Pagos y transacciones, banca online, asesoramiento online y monederos digitales entre otros.

Desde 2015 tanto los bancos como empresas privadas y públicas han realizado fuertes inyecciones económicas económicas a startups y a los propios bancos para el desarrollo de estos servicios para su propia reestructuración y evolución. Un ejemplo de inyección en startups es la startup española Kantox que recibió unos 6,4 millones de euros en una ronda de financiación. Por otro lado un banco que ha apostado totalmente por la evolución de las tecnologías y concretamente del fintech es el

banco BBVA, ha conseguido ser una referencia no solo a nivel de fintech sino que también a nivel de APIs. Esto ha sido gracias a su centro de innovación y una gran actividad en temas de API, ya sean artículos(y documentaicón) o algo más específico para desarrolladores.

El incremento de estas inversiones ha sido provocado por el crecimiento exponencial de la tecnología blockchain y de las criptomonedas que han revolucionado todo el sector financiero.

Figura 3: Kantox

Figura 4: BBVA API

Existe una "Asociación Fintech" (https://asociacionfintech.es/) creada por Kantox, junto a otras empresas como: Finanzarel, Arboribus, LoanBook, Comunitae y Deudaeque. La misión de la asociación es la de impulsar el sector financiero tecnológico en España viendo el crecimiento del número de empresas y de la cifra de intermediación que existe en EE. UU. y UK.

1.3.2 Blockchain

En los últimos 5 años este término ha ido cogiendo más renombre debido al

aumento de valor de la criptomoneda Bitcoin. Este aumento de popularidad en parte ha sido gracias a las startups que han encaminado el futuro del fintech hacia el desarrollo de APIs de esta tecnología.

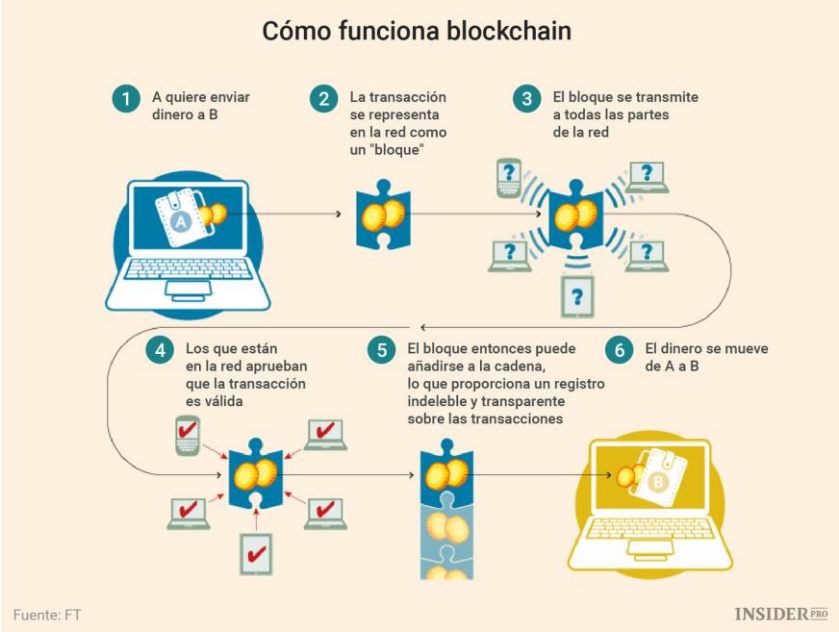


Figura 5: Funcionamiento blockhain[32]

El blockchain ha conseguido revolucionar el fintech gracias a las denominadas cadenas laterales. Estas cadenas registran las transacciones de mercados específicos ajenos a los bitcoins ya que los intercambios de los activos no tienen que ser exclusivamente criptomonedas. La principal funcionalidad de esta tecnología es crear un sistema de contabilidad público mediante cadenas de bloques mientras mantiene sus usuarios anónimos; estos bloques están enlazados y cifrados para proteger la seguridad y privacidad de las transacciones. En las cadenas de bloques deben existir varios nodos que son los encargados de la verificación de las transacciones para su posterior validación y así el bloque correspondiente de esa transacción se registre.

Un ejemplo de API de blockchain es la API de BBVA Payments que permite a desarrolladores de aplicaciones de terceros puedan ofrecer un servicio de transferencias personalizadas de intercambios rápidos y seguros de dinero a sus usuarios.

```
import urllib
import httplib

qparams = {}
parqparams = urllib.urlencode(qparams)
```

```
conn = httplib.HTTPSConnection('apis.bbva.com')

headers = {}
headers["Content-Type"] = "application/json"
headers["Accept"] = "application/json"
headers["Authorization"] = "jwt eJxt1MeOq1gUBdBfKXm...KqshJctUTsUwwYBPNpHWBS7AxmBy"

conn.request('GET', '/payments-sbx/v1/me/transfers?%s'%parqparams, None, headers)

r = conn.getresponse()

print r.status, r.read()
```

1.3.3 Insurtech

Podemos definirlo como la rama de las fintech dedicada al mundo del seguro. Se ha observado un cambio de tendencia en las aseguradoras tradicionales las cuáles están optando por este tipo de estrategia.

El insurtech aún padece cierta incertidumbre legal ya que es un sector intensamente regulado y la aparición de nuevos competidores y startups crean una situación de recelo, esto es debido a que las múltiples regulaciones que hay que tener en consideración van relacionadas con aspectos de actividad online incluyendo la protección de datos y la protección de los consumidores. Además las aseguradoras que sean totalmente digitales deben cumplir con la normativa del e-commerce

Un ejemplo del avance de esta nueva rama son los seguros de automóvil que según los datos de 2016 de la patronal de ICEA son el 16% del total del sector de los seguros. Hasta ahora los factores que determinaban la cuantía de la póliza del cliente eran el sexo, edad e incluso el color del vehículo. Actualmente integrando unos sensores en el vehículo del asegurado pueden saber realmente cuándo y cómo conduce esa persona[16].

1.3.4 Healthtech

La OMS (Organización Mundial de la Salud) define healthtech como la aplicación de conocimientos y habilidades organizadas en forma de dispositivos, medicina, vacunas, procedimientos y sistemas desarrollados para resolver un problema de salud y mejorar la calidad de vida.

El crecimiento de este sector ha sido abismal, en los últimos cinco años, la inversión de fondos de riesgo ha crecido cerca de un 200%, llegando a los 11.700 millones de dólares por parte de más de 30.000 inversores de la industria. El humano nunca va a escatimar en salud y eso se ve reflejado en la cuantía de las inversiones del *healthtech*. Además parte de este éxito es que los usuarios cada vez más están habituados a ver estas tecnologías disruptivas en su entorno.

Actualmente existen varios concursos para potenciar y promocionar el healthtech, como la Healthtech Start-Up Competition de la Generalitat de Catalunya (http://www.biocat.cat/es/agenda-del-sector/fin-de-convocatoria-healthtech-start-competition) que consiste en presentar soluciones innovadoras y los planes de negocios asociados al sector sanitario.

La tendencia de esta nueva rama es seguir en aumento, pero al ser relativamente joven el marco regulatorio actual necesitará evolucionar junto a la sanidad digital de forma que ambos se beneficien.

1.4 Principios de programación de una API

En una API no existen unos principios de programación que los expertos hayan definido como unos estándares, sino que se recogen pinceladas de otros principios ya existentes. Por ejemplo, podemos decir que una API tiene principios de programación orientada a objetos, ya que muchas de ellas están implementadas en lenguajes de ese carácter. En este caso podemos decir que algunos de los principios de programación de una API son:

- **Encapsulación**: en una API "encapsulamos" datos lejos del usuario del objeto. Esto significa que cualquier método, clase o la API en sí, no debe estar accesible para el usuario sin una modificación del programador.
- **Abstracción**: retiramos al usuario del proceso de saber cómo funcionan las API. Este principio, significa que el usuario no necesita saber cómo el objeto que éste utiliza completa una tarea.
- **Herencia**: permite implementar una API más compleja a partir de partes más simples. Es decir, una clase puede heredar funcionalidades en sus objetos de otra clase.
- **Polimorfismo**: este principio está ligado, directamente, con el principio de programación de la herencia. El polimorfismo es la forma en la que un objeto heredado interactúa con el objeto de la clase padre.

Por otro lado como en cualquier otro tipo de lenguajes, antes de la implementación está la fase de diseño para que cuando se pase a la fase de implementación resulte una tarea mucho más rápida y optimizada entre otras características. Así que podemos considerar que una API puede ser utilizada por la comunidad de desarrolladores global o incluso puede estar destinada a la red de delegaciones de la propia empresa (o por *partners*) .

Otros tipos de principios que no estarían estrictamente ligados a la programación ya que como se ha comentado anteriormente, una API puede implementarse en distintos tipos de lenguajes de programación; en este caso a nivel

técnico una API pública precisa de una plataforma capaz de gestionar las siguientes áreas:

- Autenticación, autorización de recursos, confidencialidad
- Análisis y monitorización de su uso
- Gestión de la prioridad de acceso, en función del tipo de contrato, etc.
- Gestión de ciclo de vida y control de versiones
- Documentación y ejemplos
- Plataforma de test
- Gestión de cuentas
- Conclusión

1.4.1 Arquitectura de una API

La elección de un tipo de API es una de las decisiones más relevantes que puede tomar un desarrollador de APIs. Para tomar una decisión correcta u óptima se deberán tener en cuenta los aspectos técnicos, la naturaleza de los recursos back-end mostrados y las limitaciones. Sin embargo también hay que tener en cuenta aspectos como los objetivos de negocio de la API o las necesidades y preferencias de los desarrolladores de Tecnología de la Información. Los tipos de diseños y/o arquitecturas de API más comunes son:

- Servicio web o tunelización: está relacionada con SOA. Dispone de múltiples herramientas y no es adecuada para móviles.
- REST pragmática o URI: es la más idónea para aplicaciones web y móviles. Es la más popular actualmente y familiar para la gran mayoría de dispositivos de aplicaciones, sin embargo, no resulta adaptable a lo largo del tiempo.
- Hipermedia o true REST: está muy centrada en la Web. Es escalable y con capacidad de evolución, a diferencia de la REST pragmática. Actualmente es una arquitectura desconocida para muchos dispositivos.
- Basado en eventos o IoT: la más adecuada para IoT, es ligera y dinámica pero no es adecuada para escenarios estándar.

Figura 6: Arquitectura API[11]

Hay que procurar no escoger el tipo de arquitectura "que esté de moda" sino el tipo que se adecue más a nuestras necesidades específicas, al mismo tiempo habrá que tener en cuenta el tipo de diseño que nos permita demostrar la escalabilidad y adaptabilidad a largo plazo, ya que los recursos cambian, la audiencia de usuarios aumenta y la propia naturaleza de las conexiones de red en línea evolucionan rápidamente.

Independientemente de la arquitectura que se escoja existen componentes arquitectónicos comunes en todas ellas, estos componentes no se deberán integrar en la implementación sino que se deberán implementar en una estructura de API central. Al individualizar estos componentes resultará más rápido y sencillo diseñar APIs adicionales para actualizar una determinada gama de ellas a la vez así como para garantizar una ejecución fluida en los sistemas *back-end* y las aplicaciones de cliente. Para lograrlo los componentes estarán divididos en capas, de forma que el tráfico de datos debe pasar por cada una de ella:

- Capa de seguridad.
- Capa caché.
- Capa de representación.
- Capa de articulación.

Figura 7: Resumen capas [11]

1.5 Open Data

El Open Data se considera que es una filosofía con el objetivo de proporcionar al público general los datos que la administración pública gestiona en formatos fáciles de manipular. Actualmente esta filosofía no se restringe sólo a las AAPP, se ha observado un aumento considerable de la publicación de datos en formatos abiertos por parte de las empresas privadas.

El objetivo final es dar la facilidad a cualquier persona física o jurídica para reutilizar los datos para generar un valor económico Pudiendo generar nuevos datos, conocimientos o incluso la creación de servicios que reporten beneficios económicos y/o sociales (entendido como el beneficio económico obtenido por el cobro de la transformación de los datos; es decir, se puede crear un servicio que sea de pago y explote estos datos).

Entonces, se puede definir Open Data como la utilización, reutilización y la redistribución libre de los datos por cualquier persona previniendo el uso comercial, los datos no deberían monetizarse.

1.5.1 Principios del Open Data

Existen unos principios estipulados que nos confirman el uso real del Open Data en los datos que se han obtenido o que se van a distribuir. Estos principios son:

- **Accesibles:** deben ser accesibles al mayor número de personas posible. Es decir, no debe de haber ningún tipo de restricción para su uso.
- **Actualizados:** la disposición de los datos a los usuarios debe ser con la mayor frecuencia posible para que la información seá más precisa.
- **Libres:** no tienen que estar sujetos a derechos de privacidad o privilegios. No deben tener patentes ni copyright.
- **Abiertos:** no pueden depender de una entidad o de una herramienta. El *.doc* (Word) o *.xlsx* (Excel) son ejemplo de formatos no abiertos. Ejemplos de formatos abiertos

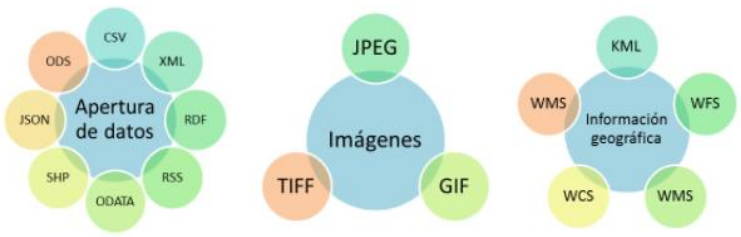


Figura 8: Formatos Open Data

- **Sin registro:** no debe ser necesario la identificación del usuario en ningún momento para el uso y disposición de los datos.
- **Detallados:** la publicación de los datos debe ser tal y como se encuentran en su origen, manteniendo el mayor nivel de detalle posible.

1.5.2 Portales de Datos Abiertos

Actualmente las webs de los gobiernos e instituciones públicas son los mayores exportadores de datos abiertos. No obstante, hay un importante auge ámbitos de la ciencia y la empresa.

Portales de instituciones públicas:

- Open Data - Gobierno de Canarias (http://opendata.gobiernodecanarias.org/)
- Open Data Barcelona - Portal de datos abiertos multiayuntamiento (http://opendata.cloudbcn.cat)
- Portal de datos abiertos de la Unión Europea (http://data.europa.eu/euodp/es/data)

Portales científicos:

- UK Data Archive (http://data-archive.ac.uk/find): proporciona los datos abiertos de las investigaciones de las ciencias sociales y humanidades del Reino Unido.
- La biblioteca universitaria de Huelva (http://guiasbuh.uhu.es/c.php?g=498100&p=3907565) mantiene un repositorio actualizado de portales web relacionados con los datos de investigación.

1.5.3 Open Data como negocio

En 2015 la reutilización y transformación de datos abiertos procedentes de fuentes públicas y/o privadas provocó un volumen de negocio medio estimado de unos 1.650 millones de euros en España. Los datos proporcionados por el sector público generaron entre 600 y 750 millones de euros, según un estudio de la ONTSI(Observatorio Nacional de las Telecomunicaciones y la Sociedad de la Información)[28].

Si hablamos de un aumento económico en el sector de los datos abiertos, no podemos ignorar la información relacionada al empleo. Es decir, en los trabajos vinculados a esta actividad un 53% de las empresas afirman haber contratado personal específicamente para trabajar en este sector, lo que supondría llegar cerca de 5.200 puestos de empleo de este tipo de servicio. Para los desarrolladores de

esta actividad el principal inconveniente es que hay muy poca homogeneidad en la información por lo que limita la actualización y el mantenimiento de lo desarrollado. Este tipo de problema es muy usual en el ámbito público.

De cara al futuro el sector privado y el público miran hacia el mismo lado, ambos sectores quieren aumentar la disponibilidad de datos en tiempo real. Esto va enfocado a dar mejor servicio a proyectos de *Big Data*, de *Smartcities* y de *Social Data*.

Capítulo 2
Requisitos

2.1 Objetivo del proyecto

El objetivo de este proyecto es la creación de una API pública cuyo resultado se muestre en una plataforma web.

El desarrollo de esta API pública será el resultado de un mashup de datos de dos fuentes distintas, donde la principal fuente de información será el Sistema de Información Medioambiental de Canarias. Este tipo de APIs tienen mucho que ofrecer en el ámbito de la información gubernamental y, como el título del proyecto indica, podrá proporcionar datos con la filosofía del Open Data que en un futura pueda generar en un servicio Big Data y de Smart Cities.

2.2 Requisitos y casos de uso

En este apartado encontraremos los requisitos funcionales y no funcionales del proyecto, concretamente, de la aplicación que se va a desarrollar.

2.2.1 Requisitos funcionales

Los requisitos funcionales son las declaraciones de los servicios que debe proporcionar el sistema; es decir, expresan cómo debe ser el funcionamiento y/o comportamiento frente a las interacciones del sistema con su entorno, por lo que a veces es interesante indicar lo que no hará el sistema o la aplicación. Estos requerimientos van a depender de los posibles usuarios de la aplicación y del enfoque general tomado al redactar los requisitos. En este proyecto el usuario de la API será un usuario general que podrá darle el uso tanto a nivel de desarrollador a un nivel más básico.

- Requisitos funcionales generales:

RF-1	Cruce de datos
Descripción	La plataforma web, por medio de las API's proporcionarán un cruce de datos de distintas fuentes de datos abiertos de distintos formatos.

RF-2	Generación de informes
Descripción	Se proporcionará al usuario la posibilidad de generar informes en formato PDF con los datos de la información que haya seleccionado

RF-3	Generación de estadísticas y gráficas
Descripción	Se podrá visualizar una serie de estadísticas y gráficas donde se reflejen el cruce de datos que el usuario haya seleccionado.

RF-4	Histórico
Descripción	En la plataforma web habrá un histórico de resultados de las consultas más frecuentes y de los datos de las fuentes.

RF-6	Visualización de los datos en tablas
Descripción	El usuario tendrá acceso a los datos en una visualización de tabla.

RF-7	APIs abiertas
Descripción	El usuario tendrá acceso a los códigos de las APIs para su utilización.

RF-8	Generación de datos abiertos
Descripción	El usuario tendrá acceso a los datos en formato JSON y CSV de un modo accesible.

RF-9	Visualización de documentación
Descripción	El usuario tendrá acceso a la documentación tanto de los datos seleccionados como de la propia API.

RF-10	Combinar datos de AEMET con medioambiente
Descripción	

RF-11	Crear API a partir de datos de cada fuente: AEMET, Medioambiente.
Descripción	

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de las funciones o servicios ofrecidos por el sistema; en otras palabras, cómo deben ser estas funcionalidades. Como indica su propio nombre, los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. En general estos requisitos se aplican al sistema en su totalidad y no de un modo individual al sistema

RNF-1	Diseño responsive
Descripción	Las páginas diseñadas y publicadas en el dominio deberán disponer de un diseño adaptable también para dispositivos móviles.

RNF-2	Open Data
Descripción	Se debe asegurar que la aplicación/plataforma siga los principios de Open Data.

RNF-3	Ágil generación de informes
Descripción	En los módulos donde sea posible la generación de informes, estos deben generar rápido. Es decir, el cliente/usuario no debe

	esperar por el resultado.

RNF-4	Usabilidad
Descripción	El diseño debe ser intuitivo, atractivo para el usuario y que permita hacer efectivo las funciones correspondientes.

2.2.3 Casos de uso

El actor principal de los diferentes casos de uso es el Usuario. Este usuario es de ámbito general, es decir, es esta fase del desarrollo del proyecto no se quiere la diferenciación de un usuario registrado a uno que no se registre. Simplemente se quiere dar acceso libre tanto a los datos de resultantes como a la API.

- CU-1:

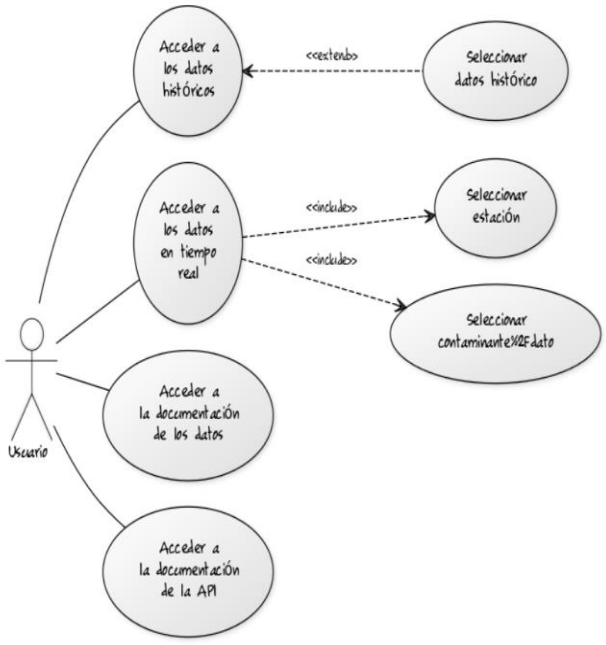


Figura 9: CU-1

Dependencias	• RF-1 • RF-4 • RF-6 • RF-7

<table>
<tr><td></td><td>• RF-8</td></tr>
<tr><td>Precondición</td><td></td></tr>
<tr><td>Descripción</td><td>El usuario puede acceder a la sección de la web que desee, sea: Histórico, datos reales o documentación.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Accede a la sección de datos Históricos.</td></tr>
<tr><td>2</td><td>Selecciona entre: Estación o Contaminante y año.</td></tr>
<tr><td>3</td><td>Accede a la sección de datos en tiempo real.</td></tr>
<tr><td>4</td><td>Selecciona entre: Estación o Contaminante.</td></tr>
<tr><td>5</td><td>Accede a la sección de documentación de los datos.</td></tr>
<tr><td>6</td><td>Leer la información sobre los datos que se proporcionan y la interpretación que reciben</td></tr>
<tr><td>7</td><td>Accede a la sección de documentación de la API</td></tr>
<tr><td>8</td><td>Leer la información sobre la API.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td></td></tr>
</table>

Tabla **1:** CU1

- CU-2:

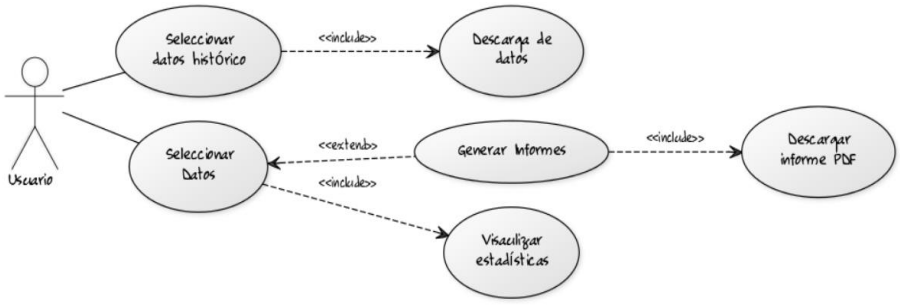


Figura 10: CU-2

<table>
<tr><td>Dependencias</td><td>RF-1
RF-2
RF-3
RF-4
RF-5
RF-7
RF-10
RNF-2
RNF-3</td></tr>
<tr><td>Precondición</td><td>El usuario deberá seleccionar el tipo de información que quiere.</td></tr>
<tr><td>Descripción</td><td>El usuario puede acceder a dos secciones; por un lado, estará la sección donde puede descargar información sobre los datos históricos. Y por el otro lado, puede acceder a la sección de visualización estadística y en tabla de los datos en tiempo real. Estos últimos datos se podrán generar informes que se podrán descargar.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Accede a la sección de datos Históricos.</td></tr>
<tr><td>2</td><td>Selecciona entre: Estación o Contaminante y año.</td></tr>
<tr><td>3</td><td>Accede a la sección de datos en tiempo real.</td></tr>
<tr><td>4</td><td>Selecciona entre: Estación o Contaminante.</td></tr>
<tr><td>5</td><td>Descargar los datos con la información deseada.</td></tr>
<tr><td>6</td><td>Accede a la sección datos en tiempo real</td></tr>
<tr><td>7</td><td>Selecciona entre: Estación o Contaminante.</td></tr>
<tr><td>8</td><td>Selecciona el tipo de gráfica que desea ver.</td></tr>
<tr><td>9</td><td>Ve los datos en una tabla descriptiva.</td></tr>
<tr><td>10</td><td>Genera un informe a partir de los datos deseados</td></tr>
</table>
</td></tr>
</table>

	11	Descarga el informe en PDF..
Postcondición		

Tabla **2:** CU2

- CU-3:

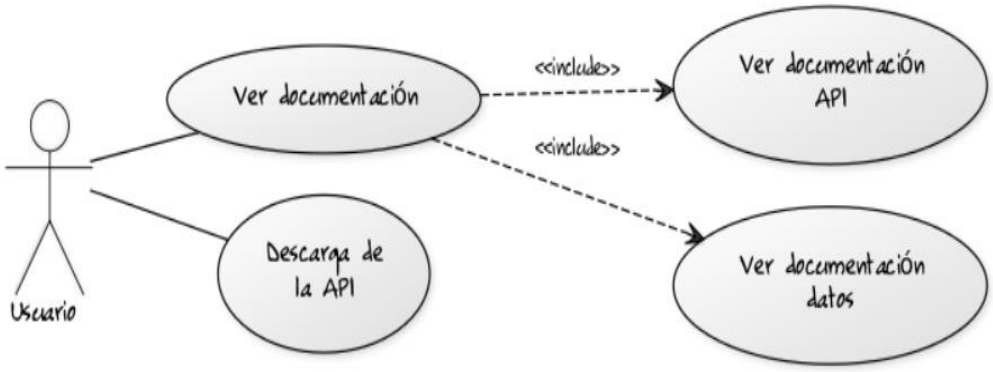


Figura 11: CU-3

Dependencias	• RF-7 • RF-8 • RF-9 • RFN-2 • RF-11
Precondición	
Descripción	El usuario puede acceder a la sección de documentación/información. Aquí podrá acceder a la documentación de las APIs o a la documentación/información sobre los datos resultantes de las APIs. El usuario también puede acceder a la visualización y/o descarga del código de las APIs.
Secuencia normal	Paso: 1 — Acción: Accede a la sección de documentación.

	2	Accede a la sección de documentación de las APIs
	3	Lee/Visualiza la información/documentación
	4	Accede a la sección de código de las APIs
	5	Lee/Visualiza el código de las APIs
	6	Descarga el código de las APIs
	7	Accede a la sección de documentación de los datos.
	8	Lee/Visualiza la información/documentación
Postcondición		

Tabla **3:** CU3

Capítulo 3
Análisis de los datos

En el capítulo 1.Introducción se han descrito conceptos como el de API y Open Data. En este apartado se procederá a realizar un análisis sobre las dos fuentes de datos que se van a utilizar para la realización del desarrollo de la API.

Primero se hará el análisis de las plataformas de las fuentes de los datos; es decir, en este primer estudio se plasmará el porqué y el cómo de la elección y obtención de estos datos.

Como punto final del capítulo se describirán los problemas y soluciones encontrados en la obtención de los datos.

3.1 Plataforma Gobierno de Canarias

La información de la calidad del aire de la isla de Tenerife son los datos que se cogen como fuente principal o fuente primaria. Estos datos se obtienen de la página web del Gobierno de Canarias destinada al medioambiente:

- http://www.gobiernodecanarias.org/medioambiente/calidaddelaire .

Figura 12: Web gobiernodecanarias/calidaddelaire

Esta web está implementada en un entorno java, a efectos técnicos, y descriptivos sobre una aplicación web desarrollada en este lenguaje podemos decir que es una colección de *servlets*, páginas dinámicas con JSP o estáticas con HTML, XHTML, clases Java, etc. o de un modo más actual utilizando frameworks y otros recursos que pueden ser empaquetados y ejecutados en distintos servidores de diferentes proveedores o de los suyos propios

Un ejemplo que indica que el desarrollo está realizado en java y JPA es la visualización de la URL; es decir, en la página de inicio se encuentra una redirección a un *inicio.do*.

3.1.1 Dificultades y/o problemas con los datos

En este apartado se procederá a enumerar y describir las dificultades y problemáticas que se han encontrado en la plataforma para la extracción e identificación de la información a utilizar en la API.

1. **Validez de los datos:** Los datos publicados no han sido validados por la Consejería de Educación, Universidades y Sostenibilidad del Gobierno de Canarias, por lo que son susceptibles de sufrir variaciones respecto a los datos definitivos (datos validados).
2. **URL no variable**: este es uno de los principales problemas de la web como se ha comentado con anterioridad. El URL no varía: es decir, se queda fijo en "*.../medioambiente/calidaddelaire/*" por lo que complica la sencillez de un método POST.
3. **Acceder a la pantalla o vista de los datos por estación**: este problema o dificultad viene originado por el problema explicado anteriormente y sumando una pantalla previa al acceso a los datos que nos muestra un combo donde aparecen todas las estaciones del archipiélago canario; es decir, para poder llegar a la vista de la cual extraer los datos hay que pasar, previamente, por varias vistas con el mismo URL.
4. **Acceder a la pantalla o vista de los datos por contaminante**: al igual que en el punto anterior para poder acceder a estos datos previamente hay que hacer una selección del tipo de acceso a los contaminantes; en este caso, se debe elegir entre: por estación o contaminante. Después, se debe seleccionar el contaminante y la zona para poder acceder a la vista que nos mostrará los datos.
5. **Obtención de datos en tiempo real:** la web nos da la opción de realizar una descarga de los datos directamente en los formatos: CSV, Excel y PDF. Al realizar esta descarga mediante un script el resultado podrá ser: un documento vacío, el esqueleto HTML de la propia web, ninguna descarga o,

con suerte, realiza la descarga correcta. Este problema existe tanto para el caso de los datos por estación como para los datos por contaminante lo que provoca realizar una obtención de los datos mucho más costosa y engorrosa de lo que se esperaba en un principio.

6. **Datos del fichero .zip del Histórico por años**: la web también nos da la posibilidad de obtener unos ficheros CSV con datos de años pasados. En este caso, la descarga con y sin script se realiza bien. El problema aparece en el contenido de estos ficheros, el contenido no es homogéneo. Tampoco existe un patrón estándar para los nombres de los ficheros, por ejemplo, en la carpeta del año 2014 nombres de ficheros o ficheros que en la carpeta del año 2015 no existen o son diferentes. El mayor problema o dificultad que se encuentra en este punto es que casi en su totalidad la disposición de los datos en formato Excel salvo algún fichero que si se encuentra en formato ODT. Dentro de estos ficheros encontramos que las hojas son los nombres de las estaciones que en no siempre coinciden en todos los ficheros y el número de contaminantes por estación y por año es muy variable.
   En conclusión, hay una falta de estandarización en los datos almacenados en los históricos por años.

3.1.2 Soluciones

Siguiendo el esquema del punto anterior se procederá a explicar las posibles soluciones pensadas o realizadas a las dificultades y problemáticas enumeradas con anterioridad. En los apartados siguientes del capítulo 3.Desarrollo de la API se describirán con más detalle algunas de estas soluciones.

1. Se indicará de una forma adecuada y visible que los datos no han sido validados por la Consejería de Educación, Universidades y Sostenibilidad del Gobierno de Canarias del mismo modo que está indicado en la página de medioambiente del gobierno de Canarias.
2. La solución hallada se realizará mediante la utilización de herramientas de desarrollo de los navegadores, en este caso se ha utilizado Google Development Tools y Firefox Development Edition.

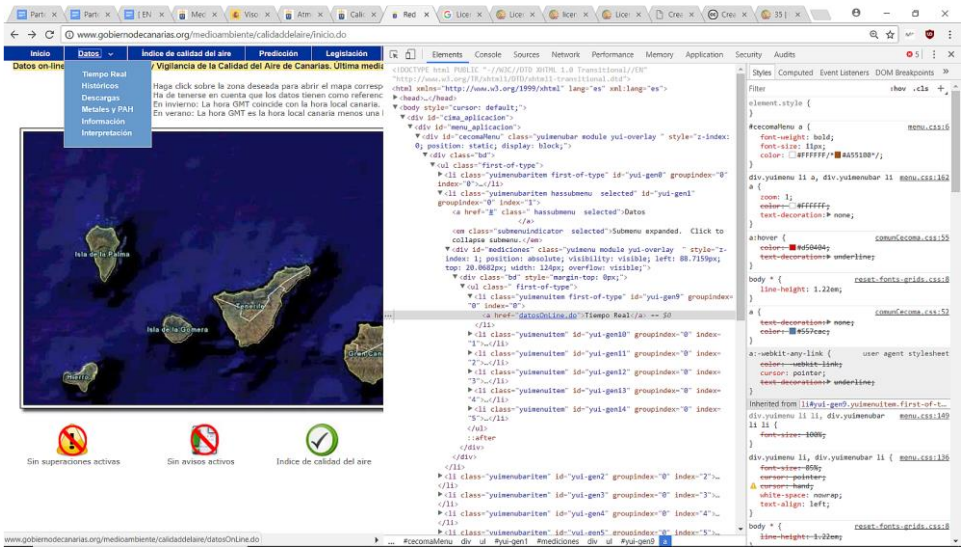


Figura 13: Solución dos

3. En los puntos 3 y 4 se sigue el mismo tipo de solución. Para ello se usan las mismas herramientas de desarrollo web mencionadas para identificar el valor de los parámetros que hacen referencia al contaminante o a la estación. En este caso al realizar el método se les pasará como parámetros el valor del *value* de la opción deseada. En la imagen siguiente se puede observar los valores de cada estación.

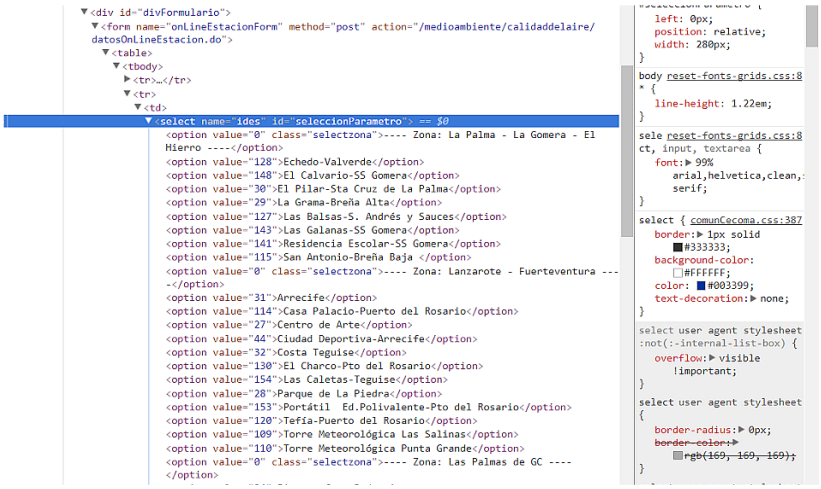


Figura 14: Solución tres

4. En el caso del problema 5 la solución realizada es realizar un *scraping* (es una técnica utilizada mediante programas de software para extraer información de sitios web[29]) utilizando la librería lxml de Python para realizar el proceso de selección de datos de las tablas y poder almacenarlos

en formatos abiertos como CSV y JSON. Esta solución se especificará en el capítulo 3.Desarrollo de la API.

5. Para el problema 6 se debe realizar un estudio del contenido de los ficheros buscando la mayor coherencia posible y patrones en los nombres para agilizar el método de transformación de los datos a un modo más homogéneo y estandarizado.

3.2 Plataforma AEMET

La segunda fuente de datos que será utilizada para realizar el mashup con la primera fuente proviene de la web de la Agencia Estatal de Metereología (AEMET).

En este último año la AEMET ha hecho una importante renovación de su web y calidad de transparencia haciéndola más usable y más accesible, al igual que permitiendo un mejor acceso a desarrolladores y usuarios generales a los datos abiertos. El acceso al portal de Open Data de la AEMET es:

- http://www.aemet.es/es/datos_abiertos/AEMET_OpenData

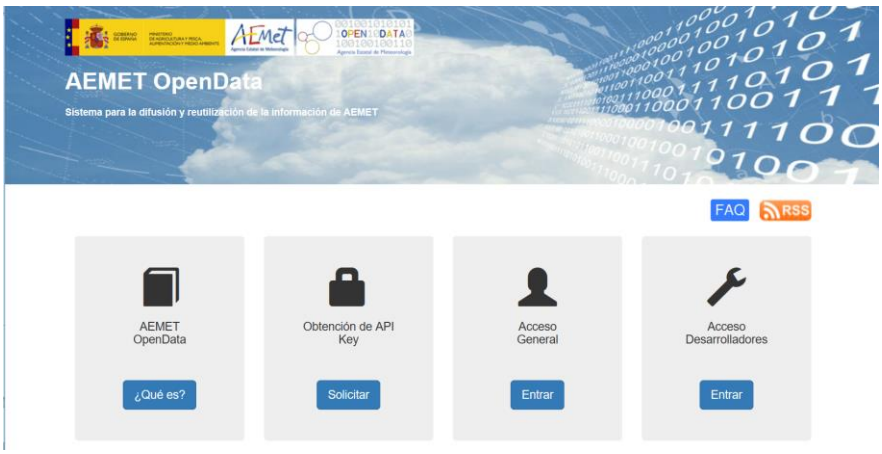


Figura 15: Web AEMET OpenData

A diferencia del primer bloque de datos la AEMET ofrece una *API Key* para acceder a su API para extraer datos, esta clave no es más que un identificador que te da ciertos privilegios o servicios de la API.

Una vez se obtiene la *API Key* se puede escoger con que tipo de perfil de usuario se quiere acceder a los datos: acceso general o acceso desarrolladores.

- **Acceso desarrolladores**: *"AEMET OpenData permite otro tipo de interacción con los datos: esta interacción se caracteriza por la posibilidad de ser periódica e incluso programada, realizada a través de un API*

destinada a un sistema informático, no se realiza a través de interfaces amigables y permite a los reutilizadores de información el incluir los datos de AEMET en sus propios sistemas de información."[24]

En este perfil nos da acceso a diferentes ejemplos del acceso a su API en lenguajes de programación como Python, Java o Ruby.

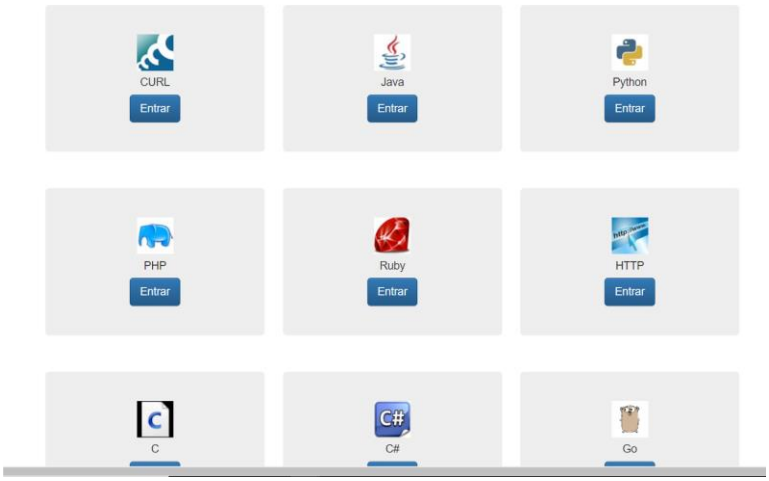


Figura 16: Más lenguajes

El ejemplo en Python sería el siguiente:

- Python http.client

```
      import http.client
      conn = http.client.HTTPSConnection("opendata.aemet.es")
      headers = {
       'cache-control': "no-cache"
      }
      conn.request("GET",
"/opendata/api/valores/climatologicos/inventarioestaciones/todasestaciones
      /?api_key=${your api key}", headers=headers)
      res = conn.getresponse()
      data = res.read()



      print(data.decode("utf-8")) import http.client
      conn = http.client.HTTPSConnection("opendata.aemet.es")
      headers = {
       'cache-control': "no-cache"
      }
      conn.request("GET",
"/opendata/api/valores/climatologicos/inventarioestaciones/todasestaciones/
      ?api_key=${your api key}", headers=headers)
      res = conn.getresponse()
      data = res.read()



      print(data.decode("utf-8"))
```

- Python Requests

```
import requests

url = "https://opendata.aemet.es/opendata/api/valores/climatologicos/
inventarioestaciones/todasestaciones/"

querystring = {"api_key":"${your api key}"}
headers = {
'cache-control': "no-cache"
 }

response = requests.request("GET", url, headers=headers, params=querystring)

print(response.text)
```

- **Acceso general**: *"El acceso para el público en general tiene como finalidad el permitir el acceso a los datos para usuarios de una manera amigable. La interacción con los datos se caracteriza por ser puntual, realizada a través de interfaces amigables destinados a un humano, dirigida paso a paso y mediante la elección de distintas opciones."*[24]

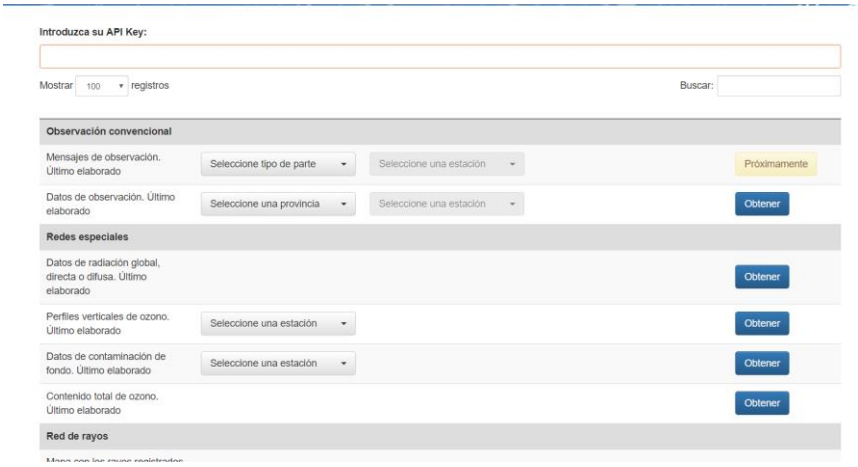


Figura 17: Web AEMET OpenData. Usuario Genral

3.2.1 Dificultades y/o problemas con los datos

1. **Se debe conocer el *idema***: hay métodos donde se necesita que se inserte el *idema*(nombre del identificador de las comunidades autónomas o de los municipios) para poder obtener los datos indistintamente de si se accede como usuario desarrollador o general a la API.

2. **Resultado temporal**: al ejecutar un método de la API no proporciona un resultado que son dos URLs con redirecciones a una página temporal con los datos y otra con los metadatos.

```
{
  " escripción" : "éxito",
  "estado" : 200,
  "datos" : "https://opendata.aemet.es/opendata/sh/4764bc0b",
  "metadatos" : "https://opendata.aemet.es/opendata/sh/8268d443"
}
```

3. **Formato variable**: como se ha indicado en el punto anterior la API en una primera instancia nos da dos URLs. Por lo general la dirección que contiene los datos está en un formato TXT o en formato JSON.
4. **Pocos datos útiles**: la AEMET te ofrece una gran cantidad de datos, pero para la Isla de Tenerife se reduce considerablemente la cantidad y la utilización de esos datos.
5. **No dan acceso a los datos de situación de alertas**: para el desarrollo de este proyecto y de la API se requiere de los datos de la situación de alertas de la isla de Tenerife pero esta información no te la da la API de la AEMET.

Figura 18: Alertas

6. **Error SSL**: al realizar ejecutar el método para acceder a la API la consola muestra un error de SSL(protocolo criptográfico que proporciona privacidad e integridad en la comunicación entre dos puntos en una red de comunicación [30]).

```
ssl.SSLError: ("bad handshake: Error([('SSL routines',
'ssl3_get_server_certificate', 'certificate verify failed')],)",)
```

3.2.2 Soluciones

1. Se ha hecho una búsqueda del *idema* (http://datos.gob.es/es/catalogo/a16003011-codigos-territoriales-de-espana-municipios-provincias-y-comunidades-autonomas o http://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177031&menu=ultiDatos&idp=1254734710990)que identifica el municipio de Santa Cruz de Tenerife y de los *idemas* de los municipios que coinciden con las localizaciones de las estaciones que obtenemos por la parte de los datos del Gobierno de Canarias.
2. La solución para este problema es el acceso programado al método de la API; se crea un método indicándole que ejecute la petición a los datos.
3. En los puntos 3 y 4 se ha llevado a cabo un estudio de los métodos de la API de la AEMET y de su información. Al observar su contenido y los datos que proporcionan se ha optado por la utilidad de dos métodos. Notificar que existen métodos en https://opendata.aemet.es/dist/index.html?#/ que no tienen contenido o que no son adecuados al contexto del proyecto.
4. Para la solución del problema 5 se ha realizado un *scraping* de la web (http://www.aemet.es/es/eltiempo/prediccion/provincias?p=381&w=11&o=ccaa) del mismo modo que para la web del Gobierno de Canarias. También se han utilizado las mismas herramientas de desarrollador para observar el esqueleto HTML.

Figura 19: Scrap

5. Para el problema 6 la solución ha sido añadir al código las librerías de Python urllib, urllib2, ssl y en la propiedad *verify=True.*

Capítulo 4
Diseño de la API

En este apartado se procederá a la descripción más detalladamente de las diferentes partes del diseño de la API. Se procederá a:

- Definición de los diferentes métodos que se han desarrollado para la obtención de los datos incluyendo ejemplos de código.
- Definición e interpretación de los datos de salida.
- Explicación y documentación de todo el proceso de desarrollo.
- Definición de los métodos que proporciona la API.

4.1 Definición de los de obtención de los datos

En este apartado se procederá a la descripción de las funcionalidades de los diferentes métodos más representativos junto a parte de su código.

- **scrapHtml()**: la funcionalidad de este método es la base principal para poder obtener los datos en tiempo real de la web del Gobierno de Canarias. Como se ha comentado en el punto 3.1.1 Dificultades y/o problemas de los datos la solución por la que se ha optado ha sido la creación de este método para la extracción directa de los datos deseados del fichero HTML. Es necesario realizar un estudio y análisis previo de la estructura de las páginas y contenidos que proporciona la web de calidad del aire ya que como se comentó en el mismo punto mencionado anteriormente no se pueden extraer los datos de un modo sencillo; esto implica que habrá un método específico según los primeros parámetros de entrada.

datosContaminanteXHoras.py

```
    def scrapHtml():
     for vZona in zona:
       for vCont in contaminante:
```

```
          tree = html.fromstring(postUrl(vCont,vZona).content)
          try:
            for xx in mapTam:
              cabecera =
tree.xpath('//table[@id="mapaFila'+str(xx)+'"]/thead/tr/th/text()')
              print(cabecera)
              for section in tree.xpath('//table[@id="mapaFila'+str(xx)+'"]'):
                content = []
                for row in section.xpath('tbody/tr[@class="odd" or
@class="even"]'):
                  tdd=row.xpath('td//text()')
                  tdd=datoEmpty(tdd)
                  content.append(tdd)
                  toJson(cabecera,content,vCont,vZona)
          except NameError:
            print('No existe esta tabla')
```

datosPorHoras.py

```
      def parseHtml(tabla,y):
       for vIdes in estaciones:
         paramsI = {"ides": vIdes}
         page = requests.post(URL,data=paramsI)
         tree = html.fromstring(page.content)
         try:
           if(y==0):
             cabecera = tree.xpath('//table[@id="'+tabla+'"]/thead/tr/th/text()')
##// muestra texto, / no muestra nada, solo \n
             for section in tree.xpath('//table[@id="'+tabla+'"]'):
               content = []
               for row in section.xpath('tbody/tr[@class="odd" or
@class="even"]'):
                 tdd=row.xpath('td//text()')
                 tdd=datoEmpty(tdd)
                 content.append(tdd)

                 toJson(cabecera,content,vIdes,tabla)
           else:
             for xx in mapTam:
               cabecera =
tree.xpath('//table[@id="'+tabla+str(xx)+'"]/thead/tr/th/text()') ##// muestra
texto, / no muestra nada, solo \n
               for section in tree.xpath('//table[@id="'+tabla+str(xx)+'"]'):
                 content = []
                 for row in section.xpath('tbody/tr[@class="odd" or
@class="even"]'):
                   tdd=row.xpath('td//text()')
                   tdd=datoEmpty(tdd)
                   content.append(tdd)


                   toJson(cabecera,content,vIdes,tabla)
```

```
        except NameError:
          print('No existe esta tabla')
```

- **toJson():** Este método consta de dos funcionalidades principales:
    - Dar formato a los datos obtenidos del primer método para su posterior manipulación o gestión de la información proporcionada. Esta función será común para ambos casos pero los parámetros se diferenciarán según el tipo de tabla encontrada o la zona (estación).

datosContaminanteXHoras.py

```
      def toJson(cabecera,content,vCont,vZona):
       c=4
       s=6
       di=10
       d=12



       x = datetime.datetime.now()

       try:
        to_unicode=unicode
       except NameError:
        to_unicode = str

       max = len(cabecera)
       data={}
       data['datosContaminante_'+str(vCont)]=[]
       if c==max:
        for j in range(len(content)):
          data['datosEstacion_'+str(vCont)].append({"detalle":[{cabecera[0]:content[
j][0],
      cabecera[1]:content[j][1],cabecera[2]:content[j][2],cabecera[3]:content[j][3]
}]})
        with
open(path+'C_'+str(vCont)+'_'+str(vZona)+'_'+str(x.day)+str(x.month)+str(x.year)+
      '.json','w+', encoding='utf8') as data_file:
          str_=json.dumps(data,indent=4, sort_keys=True,
separators=(',',':'),ensure_ascii=False)
          data_file.write(to_unicode(str_))
        data_file.close()
```

    - Almacenar los datos en formato JSON.
- **toCsv()**: al igual que en el método *toJson()* también consta de dos funcionalidades principales:
    - Almacenar los datos en formato y extensión CSV para ponerlo a disposición del usuario como se estipula en Open Data.

- Homogeneizar los datos históricos que el Gobierno de Canarias mantiene en la web para que cualquier usuario uso de ellos. Para esta funcionalidad también es necesaria realizar un estudio y análisis del contenido de las carpetas.

datosContaminanteXHoras.py

```
      def toCsv(estacion,wb,date):
       #las hojas que no existe se crea el csv
       sheet_names=wb.sheet_names() #todos los nombres de las hojas del excel
       for x in estacion:
         for sheet in sheet_names:
           sheet_=wb.sheet_by_name(sheet)
           cont=0
           if sheet_!=x:
            sheet_=sheet_names[sheet_names.index(sheet)+cont]
            cont=cont+1
            if sheet==x:
              sh = wb.sheet_by_name(x)
              your_csv_file =
open(os.path.realpath('table_'+x+'_'+str(date)+'.csv'), 'w')
              wr =
csv.writer(your_csv_file,quoting=csv.QUOTE_MINIMAL,dialect='excel')
              i = sh.nrows #numero de filas de la hoja
              j= sh.ncols # numero de columnas de la hoja
              row = 0

              while row < i:
               cell = 0
               #row=row+1
               while cell<j:
                row=row+1  # aqui hace que no se dupliquen las filas
                print('Es la fila ',row)
                print( ' del fichero ', date)
                try:
                 if int(date)==2007 or int(date)==2015:
                   cell_values= sh.row_values(row)
                   cell_values[1]=xlrd.xldate.xldate_as_datetime(cell_values[1],
wb.datemode).strftime('%d/%m/%Y')
                   cell_values[2]=str(round(cell_values[2]))+':00'
                   wr.writerow(cell_values)
                   cell=cell+1
                 else:
                   cell_values= sh.row_values(row)
                   cell_values[0]=xlrd.xldate.xldate_as_datetime(cell_values[0],
wb.datemode).strftime('%d/%m/%Y')
                   cell_values[1]=str(round(cell_values[1]))+':00'
                   wr.writerow(cell_values)
                   cell=cell+1

                except Exception as e:

                  wr.writerow(cell_values)
                  # print ('Error escribiendo en fichero ERROR-> ', e)
```

```
                break
            your_csv_file.close()
        else:
            cont=cont+1
```

- **scrapAlerta()**: la funcionalidad de este método es la solución hallada al problema de No dan acceso a los datos de situación de alertas. Se realiza un scraping de la web http://www.aemet.es/es/eltiempo/prediccion/provincias?p=381&w=11&o=ccaa, del mismo modo que se realiza para la web del Gobierno de Canarias.

```
def scrapHtml():
    page = requests.post(URL)
    print('url ->' + URL)

    tree = html.fromstring(page.content)
    try:

        dias = tree.xpath('//table/tbody/tr[not(@class="fila_impar
align_center")'
                          ']/th[@class="cabecera_oculta_niv2"]')
        for dia in dias:
            alerta = dia.getnext()
            alerta_tag = alerta.tag
            dia_ = str(dia.text)
            date = dia_.replace(" ", "_")
            print(date)
            contentDiaAlerta.append(date)
            while((alerta_tag == 'td') & (alerta != None)):
                tree_v = html.fromstring(etree.tostring(alerta,
pretty_print=True))
                nivelAlerta= tree_v.xpath('//a/div')
                for n in nivelAlerta:
                    var = str(etree.tostring(n, pretty_print=True))
                    for tf in tipoFenomeno:
                        for nf in nivelFenomeno:
                            className= tf+" "+nf
                            if (var.find('class="'+className+'"')) != -1:
```

- **toJsonAEMET()**: la funcionalidad de este método es realizar un almacenado y creación de los datos que se han seleccionado previamente en la API de la AEMET y se pasa a un formato legible.

4.2 Métodos de la API

En este apartado se mostrarán que métodos proporciona la API desarrollada, son cuatro métodos: alertas, datosContaminante, datosEstacion y datosAemet. Todos devolverán un fichero JSON con la información.

Los métodos principales de la API son:

- **/avisos_alerta/format?=api**: proporcionará los datos de las alertas por riesgo meteorológico.
- **/calidad_aire_contaminante/format?=api**: dará los datos según el contaminante seleccionado de las estaciones de la isla de Tenerife proporcionados por el Gobierno de Canarias.
- **/calidad_aire_estacion/format?=api**: este método dará los datos según la estación escogida y mostrará la información relacionada; es decir la medición de los contaminantes. Su ámbito también se restringe a la isla de Tenerife y proporcionado por el mismo organismo.
- **/datos_aemet/format?=api**: en este caso el método dará los datos que la AEMET nos proporciona y según el municipio; es decir, también se restringe a la isla de Tenerife.

Los métodos secundarios son:

- **/municipios/format?=api**: mostrará los datos referentes a los municipios de la isla de Tenerife por parte de la AEMET.
- **/estacion/format?=api**: mostrará los datos referentes a las estaciones de la isla de Tenerife por parte del Gobierno de Canarias.
- **/estacion_municipios/format?=api**: proporcionará los datos de relación entre los *municipios* y las *estaciones.*

Estas cuatro peticiones seguirán el mismo estilo de obtención de los datos; es decir, el mecanismo por el cual el usuario puede acceder a ellas y recoger la información. Por un lado, la creación de la API mediante Django framework te da la oportunidad de incorporar un botón GET que automáticamente (al "*clicar*") obtendrás los datos. Asimismo el usuario también lo podrá obtener los datos mediante *http.client* o *request*.

4.3 Definición de los datos de salida

En este apartado se definirán tanto los datos de salida de la API como los datos de salida de la web. Los datos de salida son el objetivo final de la API desarrollada y de los puntos claves para el término denominado *Smart Cities*.

La mezcla y transformación de unos datos primarios, o de fuentes primarias, su resultante es la creación de información útil. Es decir, una reinterpretación del nuevo conjunto de datos que a posteriori, este nuevo bloque de datos, podrán ser utilizados como fuentes primarias de otro resultante.

En este proyecto se diferencias dos tipos de salida de datos: texto plano y gráficas.

- **Texto plano**: llamamos texto plano a los datos de salida en formato JSON y CSV. Estos datos de salida son los que van a ser proporcionados por los métodos de la API. Su principal información serán los parámetros obtenidos de las fuentes primarias.
    - Datos directos del Gobierno de Canarias. Estos datos nos muestran los datos sin mezclar con otra fuente. Muestran los valores por ejemplo, de: Fecha, Hora, SO2(µg/m³), NO2(µg/m³), NOX(µg/m³).
    - Datos directos AEMET. Como en el caso anterior son datos directos de la fuente principal, no están mezclados.
- **Gráficas**: las gráficas son un dato de salida más visual. En este caso las resultantes serán los datos anteriores visualizados de una forma más interpretativa. Los datos de las gráficas podrán ser: datos directos o datos cruzados.
    - Datos directos: como en los datos de salida de texto plano son los datos de una única fuente, el significado de estas gráficas es poder observar la evolución de un elemento. Por ejemplo, el índice mensual de los rayos ultra violetas en la provincia de Santa Cruz de Tenerife.

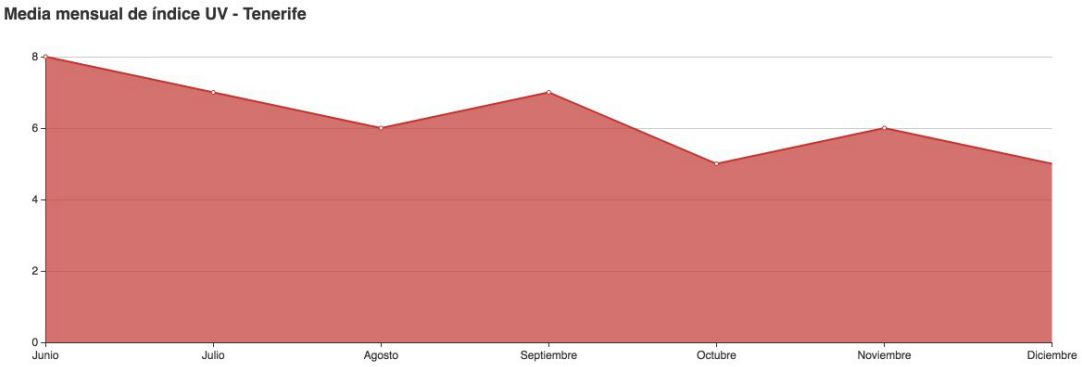


Figura 20: Ejemplo gráfica

    - Datos cruzados: los datos de salida serán una mezcla de las dos fuentes de datos. Una interpretación principal es la observación de posibles coincidencias en los valores de los elementos. Por ejemplo, podemos visualizar si un contaminante sube su nivel o no cuando hay alerta por algún tipo de fenómeno meteorológico.

4.4 Desarrollo

En este apartado se describirá parte del desarrollo práctico del proyecto. El desglose de la información se hará dividiéndolo en cuatro secciones o apartados que

son:

- Obtención de datos.
- Almacenamiento de datos.
- Back-end.
- Front-end.

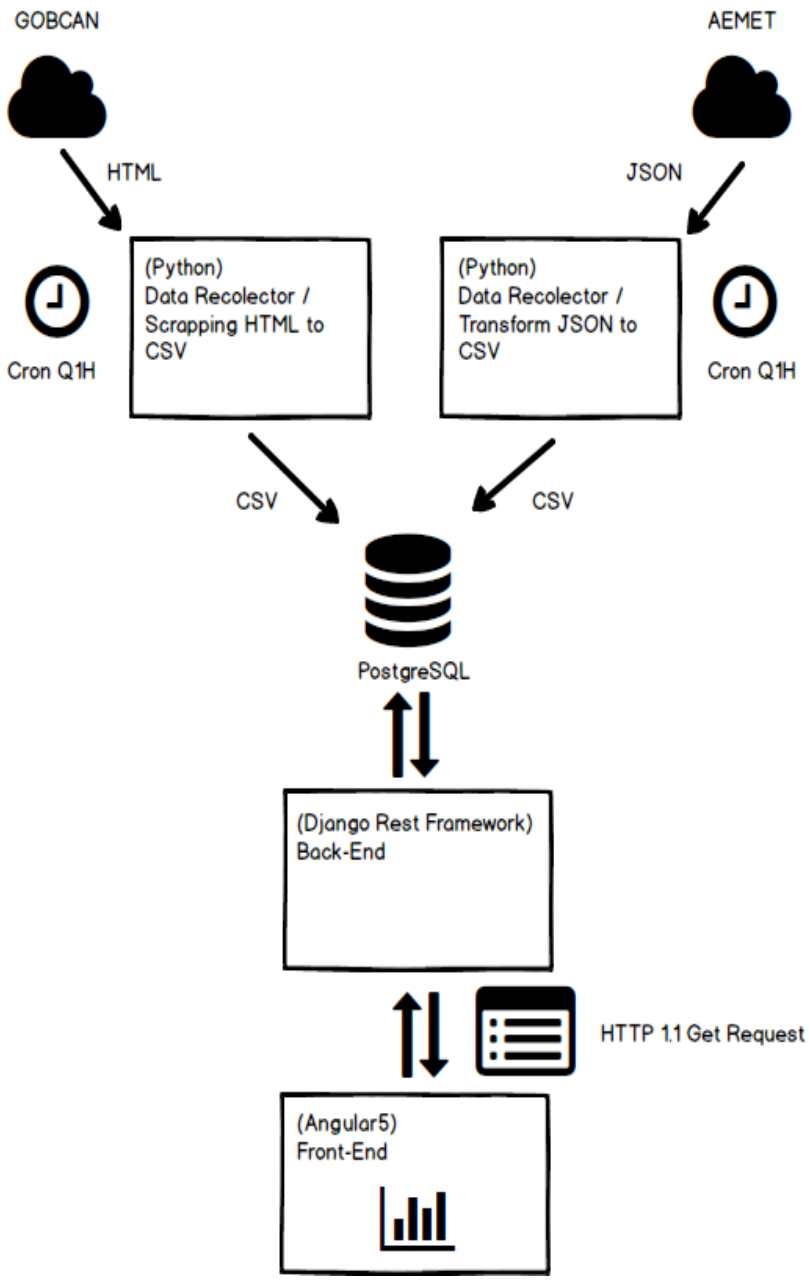


Figura 21: Esquema desarrollo

4.4.1 Obtención de datos

Como se ha ido mencionado a lo largo del documento la obtención de datos es el objetivo principal y en lo que se ha centrado el desarrollo.

En una primera fase ha sido primordial hacer un análisis de las plataformas de obtención de los datos; en una segunda fase se ha realizado la implementación de clases y métodos, entre ellos algunos de los descritos en el apartado 4.1Definición de los métodos de las APIs.

Una vez se han obtenido los datos se procede a homogeneizarlos en ficheros CSV, esto se realiza para que el almacenamiento de los datos en la base de datos sea mucho más ágil. Asimismo, se creará un método cuya funcionalidad será el acceso programado para ejecutar la recogida de datos y almacenarlo en la base de datos.

- **IDE**: como IDE principal se ha utilizado PyCharm Community Edition ya que para la programación en Python es una buena opción, este IDE te permite crear proyectos Python con estructuras similares a las que podríamos encontrar para lenguajes como Java. Además te permite una rápida creación de proyectos con frameworks como Django. PyCharm permite incorpora librerías difíciles de instalar o integrar como NumPy, en el sistema operativo Windows.
- **Lenguaje**: la utilización de Python es debido a que es un lenguaje muy potente, es totalmente dinámico por lo que te permite una mayor flexibilidad. Es uno de los lenguajes que te permite una creación rápida de APIs.

4.4.2 Almacenamiento de datos

Como se ha comentado en el apartado anterior el *parsear* los datos a CSV da paso al almacenamiento a la base de datos, en PostgreSQL.

La elección de este sistema de gestión de base de datos es porque es código abierto y cumple con las especificaciones necesarias para el desarrollo del proyecto.

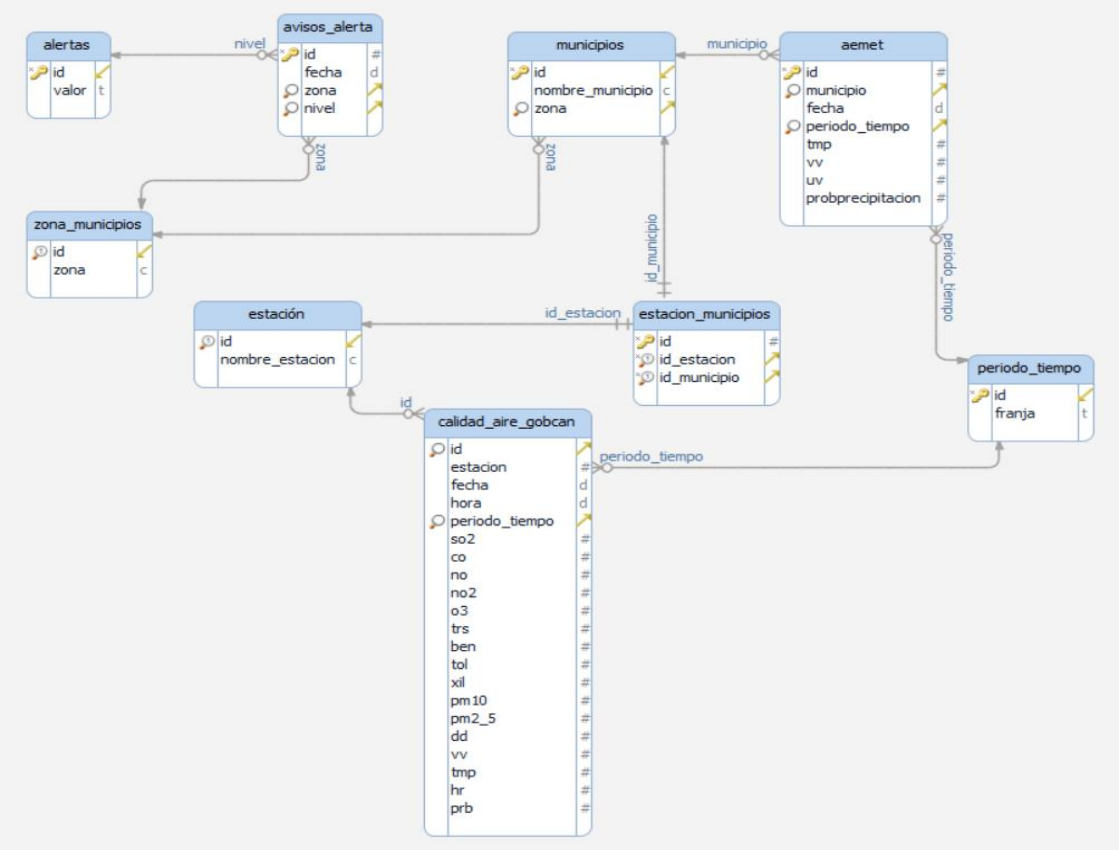


Figura 22: Esquema Base de datos

La base de datos estará *dockerizada* en el servidor.

4.4.3 Back-End

La información que se va a detallar en este apartado es la parte final de lo que engloba el back-end.

- **Servidor**: se ha escogido un servidor Nginx (https://nginx.org/en/) ya que es código abierto y software libre. También, da mucha facilidad para el uso en Windows ya que, con solo ejecutar el *exe* ya queda preparado el servidor para ser usado sin mayor necesidad de añadir las webs que se quieran visualizar en la carpeta */http*. Otra ventaja es que el consumo de memoria es menor y tiene mejor manejo de concurrencia comparado con Apache. Si lo que se pretende es tener mayor concurrencia en la web Nginx será mejor opción que Apache.
- **Django REST framework**: como se ha definido en el capítulo 1.Introducción, se va a seguir un enfoque REST. Djando permite una mayor interacción con APIs REST creadas en Python. También da mayor importancia al re-uso, la extensión de los componentes y la conectividad, eso nos va a permitir construir API REST de forma sencilla y en poco tiempo.

El nexo entre el front-end y el back-end será la petición *http GET request*. Cuando

se solicita una página Django crea un objeto HttpRequest en los métodos de dicha solicitud, entonces la acción que realizará será cargar la vista correspondiente la solicitud.

4.4.4 Front-End

Para el desarrollo del front-end se ha optado por realizarlo con Angular; su elección al igual que en las anteriores elecciones técnicas ha sido porque es un proyecto de código abierto.

- **IDE:** para la implementación del front-end se utilizará Visual Studio Code. Actualmente este IDE es código abierto, pero esto no sucedió hasta 2015 que Windows lo hizo libre. VS Code es uno de los mejores IDE para trabajar con Angular, tiene una gran potencia para TypeScript y contiene múltiples plugins imprescindibles para trabajar en Angular.

Los puntos favorables a la elección de Angular para la implementación del front-end son:

- La estructura MVVM(Model Vista – Vista Model) de Angular proporciona una rápida integración con Django framework, por lo que nos permite la creación de controladores funcionales de un modo más ágil.
- Angular utiliza como lenguaje de programación principal TypeScript, una mezcla o conjunto de JavaScript/ECMAScript que nos permite una programación reactiva.
- Los datos mostrados en los controladores son manejados en ficheros JSON y dada la naturaleza de los datos que se recogen del back-end agiliza más el proceso de no tener que seguir transformando los datos.
- El Data binding permite sincronizar automáticamente nuestro modelo con la vista de un modo bidireccional, es decir, la información será sincronizada tanto si el valor cambia en el modelo como en la vista.

Capítulo 5
Leyes y licencias

Las regulaciones, tanto licencias como leyes, del Open Data son relativamente jóvenes, es decir, en los últimos años se ha visto la necesidad de regular el qué, el cómo y para qué fin se utilizan los datos y por lo tanto la información que puede proporcionar. El ”boom” de la liberación de datos públicos fue con las olimpiadas de Londres de 2012, el gobierno inglés puso a disposición de los ciudadanos una gran cantidad de datos y siguiendo.

A nivel nacional, las regulaciones de los datos abiertos aparecieron con las aplicaciones de regulaciones y leyes a las administraciones públicas. Según la edición del informe del 2016 de Open Data Maturity in Europe, éste elaborado por Capgemini Consulting para el Portal Europeo de Datos Públicos que analiza el ecosistema de datos abiertos en más de 30 países, España se sitúa como primer país europeo mejor preparado para el “Open Data”.

En este capítulo se hace referencia y hay partes textuales de las leyes y/o licencias. Su referencia se puede hallar en la Bibliografía.

5.1 Leyes

Las leyes relacionadas con el Open Data están completamente ligadas a las nuevas leyes y directrices que deben seguir las administraciones públicas. Poco a poco se han ido realizando modificaciones en las leyes para añadir disposiciones referentes a la información que pueden proporcionar las administraciones.

Dos ejemplos de donde se establecen características y directrices sobre la utilización de los datos públicos son en:

- Ley 19/2013, transparencia, acceso a la información pública y buen gobierno: Esta Ley tiene por objeto “*ampliar y reforzar la transparencia de la actividad pública, regular y garantizar el derecho de acceso a la información relativa a aquella actividad y establecer las obligaciones de buen gobierno que deben cumplir los responsables públicos, así como las consecuencias derivadas de su incumplimiento.*”

- Real decreto 1495/2011 en el que se desarrolla la Ley 37/2007, reutilización de la información del sector público.

En cambio, no existen leyes específicas para el sector privado por lo que la información privada puede ser manipulada y tratada según las normas que ponga el propietario de dicha información, aunque también existen dos ejemplos del sector público que han empezado a comercializar sus datos que son: AEMET o el Gobierno de Navarra.

Para la regulación de los precios de los datos que ofrece AEMET se ha creado una disposición en l Anexo II de la resolución de 30 de diciembre de 2015 de AEMET. Establece los precios públicos que han de regir la prestación de servicios meteorológicos y climatológicos. Concretamente en referencia a los datos abiertos estipula que:

- "[...] *las tendencias de iniciativas como el Open Data o la reutilización de los datos del sector público, es preciso ampliar el conjunto de datos y productos meteorológicos que AEMET pone a disposición de sus usuarios de forma libre y gratuita. En consecuencia, se hace necesario adaptar a la demanda de prestaciones de AEMET a sus usuarios, diferenciando las que están sujetas al régimen de precios públicos de las que se suministran de forma gratuita y sin restricciones.* "

5.2 Licencias

Las licencias de uso asociadas a las iniciativas de datos abiertos sirven para preservar la propiedad intelectual y regular una estandarización.

El tipo de licencias más utilizadas son las Creative Commons (http://es.creativecommons.org/blog/). Esta corporación sin ánimo de lucro siguiendo la filosofía de software libre creó una serie de licencias para normalizar de una forma más flexible los términos en que como se comparten los trabajos realizados. Creative Commons dispone de seis licencias, de las cuales cuatro de ellas se aplican a los datos abiertos.

- Creative Commons Zero (CC0): ningún tipo de restricción, renunciando el creador a cualquier derecho sobre la obra, según permita la ley.
- Creative Commons Reconocimiento (CC BY): solo requiere que el reutilizador haga referencia a la fuente o autor original.
- Creative Commons Reconomiento-NoDerivadas (CC BY-ND): permite la reutilización y difusión siempre y cuando se haga referencia al autor de la misma y la obra no sufra cambio o alteración alguna.

- Creative Commons Reconocimiento-NoComercial (CC BY-NC): es posible alterar o difundir la obra original siempre y cuando se haga referencia al autor de la misma, careciendo de fines comerciales. La obra derivada no está obligada a mantener la misma licencia que la obra original.

Hasta la versión 4.0 las licencias anteriormente comentadas se centraban en los contenidos y no en los datos; es decir, suelen aplicar a los proyectos de esta índole. Por ello, la organización OKFN (Open Knowledge Foundation Network), en 2009, desarrollaron las licencias Open Data Commons (https://okfn.org/about/our-impact/legaltools/). Estas licencias son específicas para los datos abiertos.

- Open Data Commons Public Domain Dedication and License (PDDL): difundir, reutilizar o adaptar los datos sin restricción alguna.
- Open Data Commons Attribution License: exige la referencia a la autoría o fuente de los datos para la reutilización de la información.
- Open Data Commons Open Database License (ODbL): reutilización de los datos siempre que se reconozca la autoría de la información original; se mantenga la misma licencia en las obras derivadas las cuales puede restringir su uso si se distribuye una versión sin dichas restricciones de uso.

Como se ha comentado en el apartado anterior, la Administración General del Estado mediante el Real Decreto 1495/2011 ha puesto unas condiciones para homogeneizar las condiciones de reutilización de la información pública.

- No se podrá desnaturalizar el sentido de la información.
- Se debe citar la fuente.
- Se debe mencionar la última fecha de actualización de los datos, en caso de que esté disponible.
- Se deben conservar sin alterar de los metadatos.
- No se podrá sugerir o indicar que el titular de la información patrocina o apoya la reutilización que se realice.

Pero no obstante, este Real Decreto permite que los organismos públicos puedan incluir, en caso necesario y de un modo debidamente justificado, condiciones específicas para la reutilización de la información (ejemplo: AEMET).

Capítulo 6
Conclusiones y líneas futuras

6.1 Líneas futuras

En líneas futuras del proyecto a medio plazo se realizará el desarrollo de dos funcionalidades básicas y necesarias para la finalización completa del tratamiento de los datos seleccionados. Estas funcionalidades serán:

- **Generación de informes personalizables con texto descriptivo y gráficas**: los informes constarán de un texto descriptivo de las propias gráficas. El usuario podrá seleccionar diferentes aspectos para personalizarlo y de ese modo crear un informe más adecuado a sus necesidades.
- **Predicciones**: crear predicciones de los niveles de contaminación del aire según la climatología. Para ello se deberá crear una base de conocimiento y unos patrones para detectar pautas entre estos dos tipos de datos, este método a medida que se vaya teniendo más base de conocimiento y más datos se irá regulando para dar los datos de una forma más exacta.
- **Más islas:** es decir, la API no se limitarán a la isla de Tenerife, se pretende adaptarla a todo el archipiélago.
- **Comparación de datos entre islas**: al desarrollar las funcionalidades a las demás islas se podrá ver una comparación de la calidad del aire entre ellas.

A largo plazo la funcionalidad principal sería traspasar la API desarrollada para los datos de la calidad del aire a la calidad del agua de las costas canarias. Consistirá en el mismo mecanismo que se ha utilizado para los datos utilizados en el desarrollo del proyecto.

6.2 Conclusiones

Como experiencia personal el desarrollo de este proyecto me ha proporcionado aprender un nuevo lenguaje de programación como es Python que no había utilizado nunca; creo que es un lenguaje muy potente y que te permite mucha versatilidad (sobre todo porque es multiplataforma).

Asimismo, decir que al principio de este trabajo tenía muy poca idea de qué era y en qué consistía concretamente una API y a medida que he ido indagando en este mundo he podido adquirir conocimientos muy interesantes como los de API Economy. Haciendo un especial hincapié en los conceptos sobre las ramificaciones que existen en las APIs (fintech, insurtech, healthtech) ya que me han parecido muy instructivos y atrayentes, considero que son temas que en los últimos cinco años han aumentado su relevancia en las empresas y que se van a seguir hablando de ellos. En mi opinión van a seguir cambiando los modelos de negocio que entendemos como tradicionales. Con el desarrollo de este proyecto no sólo he adquirido conocimientos a nivel teórico, sino que también a nivel práctico; es decir, al empezar con la implementación no conocía ni el lenguaje ni las herramientas que he utilizado para llevarlo a cabo, por lo que me ha permitido aprender Django y concretamente Django REST Framework, siendo este último un potente framework de creación de API REST basado en Python 3.

A pesar de que en este proyecto el concepto principal es el Open Data, las Smart Cities están muy presente ya que la información de salida del tratamiento de las principales fuentes son datos sostenibles que son capaces de responder adecuadamente a las necesidades básicas de instituciones, empresas, y de los propios habitantes tanto en el plano económico, como en los aspectos operativos, sociales y ambientales.

Para concluir me gustaría añadir una pequeña crítica sobre la transparencia de los datos y el nivel de grado de éstos sobre Open Data; decir que a las administraciones públicas aún les queda un largo camino en cuanto a la aplicación a la filosofía Open Data, se debe hacer un esfuerzo mayor para procurar estandarizar los datos y, sobre todo, dar mucha más transparencia y disposición de ellos. Tampoco ayuda que se creen nuevas disposiciones en la ley de transparencia que permita a organismos públicos o AAPP (argumentándolo debidamente) poder tener su propio ajuste de ésta. Me refiero concretamente al caso de la AEMET y a la nueva disposición de 2016 que permite monetizar los datos y eso dista de la filosofía Open Data.

Capítulo 7
Summary and Conclusions

As personal experience, the development of this project has provided me to learn a new language of programming as Python, which I had never used before; I think it's a very powerful language and that it allows you a lot of versatility (especially since it's multiplatform).

Also, to say that at the beginning of this work I had very little idea of what it was and what an API consisted and as I have been investigating in this world I have been able to acquire very interesting knowledge like API Economy. I have a special emphasis on the concepts on the ramifications that exist in the APIs (fintech, insurtech, healthtech) since I have found them very instructive and attractive, I consider that they are subjects that in the last five years have increased their relevance in the companies and that they are going to continue talking about them. In my opinion they will continue to change the business models that we understand as traditional. The development of this project I have not only acquired knowledge at the theoretical level, but also on a practical level; I mean, when I started with the implementation I did not know the language or the tools I used to carry it out, so it allowed me to learn Django and specifically Django REST Framework, the latter being a powerful framework for creating REST API based on Python 3.

In spite of the fact that in this project the principal concept is the Open Data, the Smart Cities it is very present, since the information of exit of the treatment of the principal sources it is sustainable information that it is capable of answering adequately to the basic needs of institutions, companies, and of the own inhabitants, so much in the economic plane, like operational aspects, social and environmental aspects.

In conclusión, I would like to add a little criticism about the transparency of the data and the level of degree of these on Open Data; I mean, public administrations still have a long way to go in applying Open Data philosophy, a greater effort must be made to try to standardize the data and, above all, to give much more transparency and disposition of them. Neither help that new regulations

believe themselves in the law of transparency that allows public organizations or public administrations (arguing it due) to be able to have their own adjustment of this one. I refer specifically to the case of the AEMET and the new provision of 2016 that allows monetizing the data and that is far from the Open Data philosophy.

Capítulo 8
Presupuesto

8.1 Presupuesto

COMPONENTE				
	HORAS POR MES	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
MODA DE OBRA				
Analista desarrollador de Software	160 H	5	25€	20.000€
HARDWARE				
Ordenador portátil (12 GB RAM, 1T HDD, win10, i7, 3.5GHz)		1	789€	789€
SOFTWARE				
PyCharm Community		1	0€	0€
Visual Studio Code		1	0€	0€
Postgresql DBeaver		1	0€	0€
Nginx		1	0€	0€
SERVICIOS				
Energía eléctrica		5	28€	140€

Internet		5	39.95€	196,75
PRESUPUESTO				
			Sub Total	21.125,75
			15% Imprevistos	3.168.86
			20% ganancias	4.225,15
			7% IGIC	1.996,38
			Total	30.513,14

Tabla **4:** Presupuesto

Bibligrafía

[1] https://www.es.capgemini.com/sites/default/files/pdf_Mashups_de_aplicaciones_y_datos_para_incrementar_la_interacci__n_con_el_cliente.pdf

[2] SERRANO COBOS, Jorge. Pasado, presente y futuro de la Web 2.0 en servicios de información digital. 2006.

[3] VAN DER VLIST, Eric, et al. Programación Web 2.0. 2007.

[4] Los Mashups, uno de los pilares de la Web 2.0. https://www.vix.com/es/btg/tech/2007/03/26/los-mashups-uno-de-los-pilares-de-la-web-20

[5] Enterprise Mashup. https://www.inetsoft.com/company/enterprise_mashup/

[6] La importancia de la gestión optima de las APIs. http://www.networkworld.es/aplicaciones-seguras-y-optimizadas/la-importancia-de-una-gestion-optima-de-las-apis

[7] PARRA, Ezequiel. Bienvenidos a la API Economy. Byte España, 2016, no 236, p. 56-58.

[8] IBM. API-Economy. http://www.ibm.com/middleware/us-en/knowledge/hybrid-integration/api-economy.html

[9] Dreamfactory. Enterprise applications. http://blog.dreamfactory.com/topic/enterprise-applications

[10] Wikipedia. Principios de programación POO. https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

[11] CA. Arquitectura y estrategia de API Un enfoque coordinado. https://www.ca.com/content/dam/ca/es/files/ebook/api-strategy-and-architecture-a-coordinated-approach.pdf

[12] What is open data?. http://opendata.cloudbcn.cat/MULTI/es/what-is-open-data

[13] Open data. Inversion&Finanzas. http://www.finanzas.com/noticias/empresas/20170324/datos-abiertos-genero-espana-3589502.html

[14] API Economy:la posibilidad de ‘viralizar’ modelos de negocio. http://www.expansion.com/economiadigital/protagonistas/2017/04/30/58db7c8c268e3e2f3c8b45bd.html

[15] Qué es el fintech y cómo las startups quieren innovar en los servicios http://www.centrodeinnovacionbbva.com/noticias/que-es-el-fintech-y-como-las-startups-quieren-innovar-en-los-servicios-financieros

[16] Insurtech y el ecosistema asegurador disruptivo https://www.bbvaresearch.com/wpcontent/uploads/2016/10/Situacion_ED_oct16_Cap4.pdf

[17] eHealth y Health Tech, la tecnología al servicio de la salud https://www.openfuture.org/es/new/ehealth-y-health-tech-la-tecnologia-al-servic

[18] Documentación https://www.bbvaapimarket.com/documentation/bbva/payments

[19] Ejemplos Programa Cliente https://opendata.aemet.es/centrodedescargas/ejemProgramas?

[20] ¿Qué es AEMET Open Data? https://opendata.aemet.es/centrodedescargas/inicio

[21] Nota legal http://www.aemet.es/es/nota_legal

[22] Ley 37/2007 https://www.boe.es/diario_boe/txt.php?id=BOE-A-2007-19814

[23] BOE -AEMET https://www.boe.es/diario_boe/txt.php?id=BOE-A-2016-111

[24] AEMET Open Data https://opendata.aemet.es/dist/index.html?#!/valores-climatologicos/Climatolog%C3%ADas_normales_1981_2010

[25] Creative Commons Attribution License http://opendefinition.org/licenses/cc-by/

[26] Dusty Phillips, Python 3 Object Oriented Programming, PACKT Publishing,2010

[27] Magic, https://www.magicsoftware.com/es/

[28] Uso de datos abiertos generó en España unos 1.650 millones euros, un 10 % más. https://www.invertia.com/es/-/uso-de-datos-abiertos-genero-en-espana-unos-1-650-millones-euros-un-10-mas

[29] Web scraping. https://es.wikipedia.org/wiki/Web_scraping

[30] SSL. https://www.redalia.es/ssl/protocolo-ssl/

[31] Building Reusable REST API Services (Part 3 of 4) http://blog.dreamfactory.com/building-reusable-rest-api-services-part-3-of-4

[32] ¿Qué es el blockchain?. https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda