UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación Web para la recopilación, tratamiento y visualización de datos públicos

Presentado por Iván Arjona Alonso
en Universidad de Burgos — 31 de mayo de 2018
Tutores: Dr. José Francisco Díez Pastor
y Dr. Jesús Manuel Maudes Raedo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Francisco Díez Pastor y D. Jesús Manuel Maudes Raedo, profesores del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Iván Arjona Alonso, con DNI 71352655P, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "Aplicación Web para la recopilación, tratamiento y visualización de datos públicos".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 31 de mayo de 2018

Vº. Bº. del Tutor:	Vº. Bº. del Tutor:
D. José Francisco Díez Pastor	D. Jesús Manuel Maudes Raedo

Resumen

En este proyecto se va a desarrollar una aplicación web que permita la consulta de datos demográficos y sociológicos de fuentes de datos públicas en el territorio español, permitiendo combinar estas fuentes y realizar consultas más avanzadas mediante columnas calculadas.

Se trata de representar estos datos, visualizarlos en mapas temáticos sobre el mapa de España a nivel de provincia y municipio, permitiendo su exportación a *CSV* y *JSON* para su uso en procesos de minería de datos.

Los datos procesados se almacenará de forma local en una base de datos *NoSQL* para acceder más rápido y no hacer llamadas constantemente a las fuentes de datos.

La aplicación web está disponible a través de la página del proyecto:

https://tfg-datos-publicos.nanoapp.io/

Descriptores

Aplicación web, bases de datos NoSQL, datos públicos, estudio sociológico, mapas temáticos, minería de datos, visualización.

Abstract

This projects consist in the development of a web application to consult demographic and sociological data from public data sources in Spanish territory, allowing the combination of this sources and making more advanced queries through calculated columns.

The aim is to represent these data, visualize it in thematic maps on the map of Spain at the province and municipality level, allowing their exportation to *CSV* and *JSON* for data mining uses.

The processed data will be stored locally in a *NoSQL* database to access faster and not make constantly calls to the data sources.

The web application is available through the project page:

https://tfg-datos-publicos.nanoapp.io/

Keywords

Data mining, NoSQL databases, public data, sociological study, thematic maps, visualization, web application.

Índice general

Índice de figuras

Índice de tablas

Introducción

La *minería de datos* es una metodología para descubrir relaciones ocultas en grandes cantidades de datos utilizando técnicas de inteligencia artificial. Actualmente se está extendiendo bastante el uso de técnicas de minería de datos para realizar análisis sociológicos. Un ejemplo lo podemos ver en la película 'Minority Report' [**?**], en la que se predicen crímenes antes de que se cometan, en la actualidad se está intentando llegar a una aproximación similar intentando predecir cuántos, dónde y de qué tipos pueden ser los crímenes. Otro ejemplo es el reciente uso de *big data* en China como un sistema disciplinario [**?**]. En ambos ejemplos destaca la importancia de los datos demográficos y sociológicos que hasta ahora apenas se han tratado en el ámbito español.

Un tema que está siendo investigado actualmente es comprender las variables relacionadas con la comisión de un crimen. Por ello, se han realizado estudios para investigar las tasas de criminalidad utilizando técnicas de minería de datos [**?**] [**?**] a partir de datos demográficos y sociológicos con el objetivo de determinar los lugares donde asignar más recursos para reducir estos índices de criminalidad.

Se dice que los datos se están convirtiendo en el nuevo petróleo, en el sentido de que con unos datos de buena calidad pueden descubrirse relaciones insospechadas hasta el momento. Una fuente importante de datos son los datos públicos, pero tiene inconvenientes, estos datos se obtienen de múltiples fuentes, por lo que cada una de ellas nos proporciona la información organizada de forma diferente y en distintos formatos, también podrían ser accesibles de modo diferente, lo que nos complica la labor al analizarlos.

El objetivo de este trabajo es llevar estos estudios al ámbito español y unificar estas fuentes de datos de forma que el acceso sea uniforme, mucho

más sencillo y permitiendo combinar fuentes de datos de forma directa. Se ha realizado integrando varias fuentes de datos públicas utilizando técnicas de web scraping (3.3). Para después visualizar estos datos de manera sencilla, permitir crear mapas temáticos (3.4), permitir calcular columnas calculadas combinando estos datos y facilitar la exportación de los datos para después ser utilizados en procesos de minería de datos de forma directa.

Por último, agradecimiento a la idea original de Roberto Cuesta, capitán de la guardia civil y estudiante del doctorado de la Universidad de Burgos, cuyos trabajos de investigación conducirán a una futura tesis enfocada a la aplicación de técnicas de minería de datos en el ámbito criminológico.

Fuentes de datos

A continuación se van a listar las fuentes de datos de organismos públicos que se han integrado.

- Servicio Público de Empleo Estatal (SEPE)
  - Paro registrado por municipio[1].
  - Contratos registrados por municipio[2].
  - Demandantes de empleo por municipio[3].
- Estadísticas de la renta por municipio (Agencia tributaria)[4].
- Instituto Nacional de Estadística (INE)
  - Estadísticas de población por sexo, edad y procedencia[5].
  - Relación de municipios y códigos por provincias[6].
- Ministerio del interior (MIR)
  - Resultados de elecciones municipales por año[7].

Material entregado

Material adjunto a la memoria:

[1] https://datos.gob.es/catalogo/e00142804-paro-registrado-por-municipios
[2] http://datos.gob.es/es/catalogo/e00142804-contratos-por-municipios
[3] http://datos.gob.es/es/catalogo/e00142804-demandantes-de-empleo-por-municipios
[4] https://www.agenciatributaria.es/AEAT.internet/datosabiertos/catalogo/hacienda/Estadistica_de_los_declarantes_del_IRPF_por_municipios.shtml
[5] http://www.ine.es/jaxi/Tabla.htm?path=/t20/e245/p05/a2011/l0/&file=00000001.px&L=0
[6] http://www.ine.es/daco/daco42/codmun/codmunmapa.htm
[7] http://www.infoelectoral.mir.es/infoelectoral/min/areaDescarga.html?method=inicio

- Aplicación web en Flask.
- Scripts para la integración de fuentes de datos.
- Scripts para la ejecución de la aplicación web.
- Aplicación de escritorio para juntar ficheros CSV.
- Pruebas unitarias y de interfaz.
- Anexos.

Recursos disponibles en internet:

- Página web del proyecto[8] .
- Repositorio del proyecto[9] .

[8]Página web del proyecto: https://tfg-datos-publicos.nanoapp.io/
[9]Repositorio del proyecto: https://github.com/IvanArjona/TFG-Datos-publicos

Objetivos del proyecto

Este trabajo pretender colocar la primera piedra en un proyecto mucho mayor y más ambicioso.

El objetivo es desarrollar un sistema que permita integrar múltiples fuentes de datos para trabajar con ellas de manera unificada y sentar las bases para que, en el futuro, se puedan integrar más datos y formas de visualizarlos y procesarlos.

El usuario objetivo de esta aplicación es el científico de datos que quiera encontrar patrones entre datos públicos a nivel municipal. Los usuarios podrán ser académicos o también periodístas, políticos o servicios públicos.

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir, y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

2.1. Objetivos funcionales

- Integrar varias fuentes de datos públicas en una única base de datos. Estos datos son datos de carácter sociológicos, económicos y demográficos a nivel municipal en España.
- Permitir añadir nuevos conjuntos de datos de forma sencilla.
- Desarrollar un algoritmo para la carga de varias fuentes de datos en una base de datos de manera automatizada, de forma que el administrador pueda añadir nuevos conjuntos de datos de forma sencilla.

- Desarrollo de una aplicación web que permita la consulta de los datos de manera sencilla y visual.
- Facilitar la interpretación de los datos utilizando mapas coropléticos interactivos.
- Desplegar la aplicación web en un servidor de forma que sea fácil de actualizar cada vez que se realice un cambio. Además de funcionar en un entorno local.

2.2. Objetivos técnicos

- Seguimiento de principios de desarrollo ágiles durante el desarrollo del proyecto.
- Utilizar *Flask* como framework para desarrollar la aplicación web utilizando la arquitectura *Modelo Vista Controlador*.
- Utilizar *git* como sistema de control de versiones, junto con *Github* y *Zenhub* para la organización del proyecto.
- Utilizar herramientas para mejorar la calidad del código como *Codacy*.
- Manejar bases de datos no relacionales *NoSQL*.
- Elegir herramientas de *software libre* siempre que sea posible.
- Proporcionar una instalación sencilla para el desarrollador y una interfaz visual y fácil de utilizar para el usuario.

Conceptos teóricos

A continuación se van a explicar los conceptos teóricos más importantes del proyecto, de modo que se pueda entender con exactitud el proyecto en su totalidad.

3.1. Datos públicos

Se entiende como datos públicos o datos abiertos aquellos que deben estar disponibles de manera libre, para acceder, utilizar, modificar y publicar sin restricciones de copyright [**?**].

Los gobiernos tienen la capacidad de obtener grandes cantidad de información sobre la población a través de varios organismos (como podría ser el *Instituto Nacional de Estadística*). Cuando estos gobiernos liberan los datos para que cualquiera pueda utilizarlos libremente se conocen como 'Open Government Data'.

En este proyecto nos centraremos en los datos públicos españoles, aprovechando la 'Iniciativa Aporta' que promueve la apertura de información en el sector público en España. Esta iniciativa tiene el objetivo de favorecer el desarrollo de la reutilización de la información del sector público y ayudar a las administraciones para que publiquen sus datos de acuerdo al marco legislativo vigente [**?**].

Aunque nos vamos a centrar en datos proporcionados por organismos estatales, también podrían emplearse datos públicos proporcionados por empresas privadas (Por ejemplo *Google Trends* o datos meteorológicos), aunque esto último se dejará pendiente para trabajos futuros.

3.2. Bases de datos no relacionales

Las bases de datos no relacionales [**?**], también llamadas NoSQL ('Not Only SQL') son bases de datos optimizadas para ser utilizadas con modelos de datos sin esquema y potencialmente escalables.

A diferencia de las bases de datos relacionales, aquí generalmente no hay tablas, esquemas, ni relaciones, sino que los datos pueden almacenarse con cualquier esquema sin tener que seguir todos la misma estructura.

En concreto para este proyecto se han optado por bases de datos no relacionales de documentos. Este tipo de bases de datos permiten almacenar documentos en formatos como XML y JSON. Tenemos colecciones en lugar de tablas y dentro de estas colecciones tenemos documentos. Estos documentos pueden tener distintos esquemas entre ellos.

Una de las razones importantes por las que usar bases de datos NoSQL, en especial de documentos, es la especialización del trabajo con agregaciones, la distribución en la nube y la disponibilidad.

3.3. Web scraping

Web Scraping [**?**] es una técnica para extraer información de sitios web directamente de su código fuente sin utilizar APIs[10] proporcionadas por el propio sitio. La razón de utilizar estas técnicas es que no todos los sitios web nos proporcionan APIs públicas que podamos utilizar.

Lo que hacemos con un *web scraper* es buscar información dentro de un documento web siguiendo ciertos patrones en la estructura de su código fuente y extraer esta información a nuestro entorno local.

Las técnicas de web scraping se centran en transformar datos sin estructura de una página web en datos estructurados para poder ser almacenados y analizados posteriormente, por ejemplo, en una base de datos, hojas de cálculo o en dataframes de Pandas.

[10] Interfaz de programación de aplicaciones

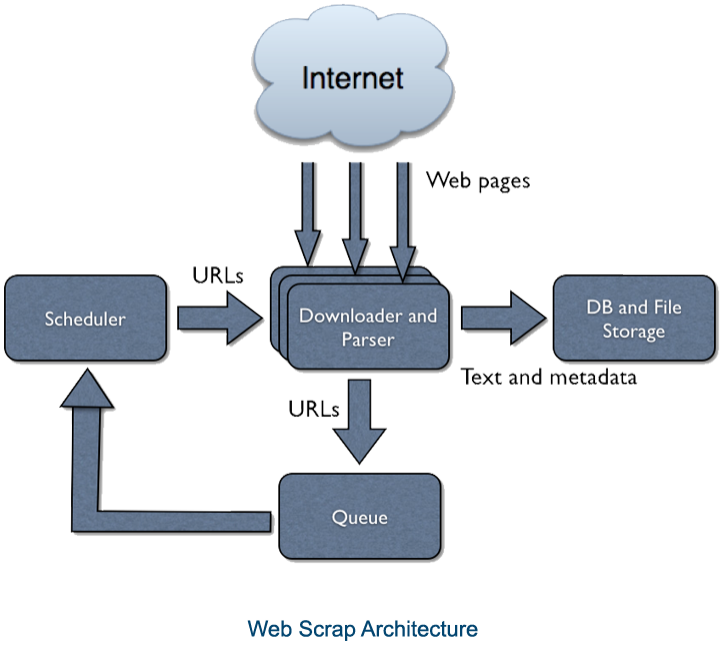

Figura 3.1: Esquema del funcionamiento de web scraping [**?**]

3.4. Mapas coropléticos

Un mapa coroplético [**?**] es un mapa topológico dividido en regiones en el que cada una de estas regiones se pinta de un color de acuerdo a una medida estadística.

Para ilustrarlo mejor pondremos el ejemplo del siguiente mapa en la figura 3.2. En él se compara el paro de todas las provincias de España pintando con colores más cálidos las provincias con mayor porcentaje de paro y con colores más fríos las provincias con menor porcentaje de paro.

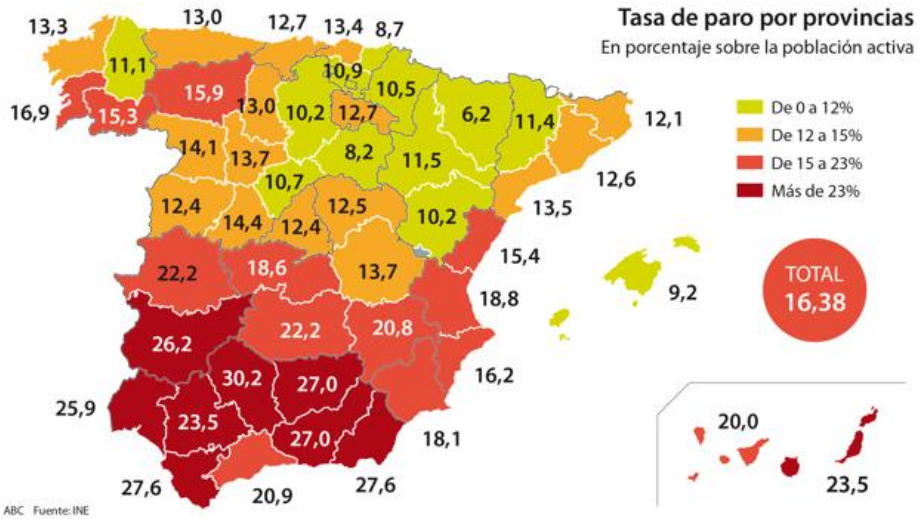

Figura 3.2: Mapa coroplético del paro en las provincias españolas [**?**]

Técnicas y herramientas

En esta sección se van a explicar, por un lado, las técnicas que se han segueoshaperido durante el desarrollo del proyecto y por otro, las herramientas utilizadas, que se han dividido en herramientas de desarrollo, gestión y documentación.

4.1. Técnicas

Metodología ágil

Este proyecto se ha realizado siguiendo los principios del manifiesto ágil [**?**]. En concreto se ha seguido la metodología *scrum* [**?**] con sprints normalmente de una semana y reuniones semanales de planificación y revisión.

Al utilizar esta metodología se persigue utilizar una estrategia de desarrollo incremental, en la que al final de cada sprint se tiene como resultado un incremento del software entregable. También se permite que los requisitos cambien a lo largo del proyecto, en lugar de fijarse al principio, consiguiendo un software de mejor calidad.

Un software creado con unos requisitos claros y fijos desde el principio puede ser un software de muy alta calidad y podría realizarse con una metodología tradicional, pero para proyectos exploratorios como este, en los que los requisitos pueden cambiar, es mejor utilizar una metodología ágil.

Para seguir esta metodología se han utilizado las issues de Github[11] (4.2) y los tableros Kanban de ZenHub[12] (4.2) para organizar la pila de tareas a realizar en cada sprint.

[11] issues de Github: https://github.com/IvanArjona/TFG-Datos-publicos/issues
[12] ZenHub: https://www.zenhub.com/

DevOps

DevOps [**?**] es una metodología para la creación de software en la que se integra el desarrollo de software y la administración de sistemas.

Se ha elegido esta metodología para conseguir desplegar rápidamente todos los cambios en el servidor web. Apoyándonos en herramientas como Nanobox (4.2) para facilitar este proceso.

4.2. Herramientas

Herramientas de desarrollo

A continuación se van a explicar las herramientas utilizadas para desarrollar el proyecto.

MongoDB

- Herramientas consideradas: Riak, Cassandra, MongoDB, LevelDB.
- Herramienta elegida: MongoDB.

MongoDB[13] [**?**] es un sistema de bases de datos NoSQL. En esta herramienta los datos se guardan en forma de documentos con un esquema similar a JSON. Con este sistema se consigue una consulta de datos más rápida. Además se ha usado PyMongo [**?**] como herramienta para integrar MongoDB en Python.

Tanto Riak como Cassandra son también bases de datos NoSQL, se descartaron porque no ofrecen soporte para equipos con sistema operativo Windows y no son bases de datos de documentos, que se cree es lo más conveniente para este proyecto.

LevelDB es una base de datos NoSQL de pares clave-valor, al igual que Riak, esta herramienta se descartó porque no se cree conveniente utilizar pares clave-valor para un proyecto como este y no hay tantos ejemplos en la documentación como en las otras herramientas.

Podría haberse utilizado bases de datos Relacionales, pero la razón por la que se eligió MongoDB es por la facilidad de almacenar los dataframes de Pandas y, sobre todo, la motivación principal fue la inquietud por formarme en esta tecnología por lo extendida que está y porque no se ve nada en la carrera.

[13] MongoDB: https://www.mongodb.com/

DigitalOcean

- Herramientas consideradas: Heroku, PythonAnywhere, DigitalOcean, Amazon Web Services.
- Herramienta elegida: DigitalOcean.

DigitalOcean[14] es un proveedor de servidores privados, por ello podemos hacer lo que queramos con el servidor sin limitaciones más allá de la capacidad de procesamiento y memoria RAM. Se ha elegido este servicio porque nos da total libertad y se puede probar gratuitamente con GitHub Education[15] [**?**].

Una alternativa que se consideró y de hecho, se probó es Heroku, en este caso se instala el entorno necesario de forma automática. El problema es que la base de datos en la capa gratuita sólo puede pesar 500MB como máximo y no es suficiente para este proyecto.

PythonAnywhere es un hosting para aplicaciones web en Python, el problema con este proveedor es que no ofrece bases de datos locales, por lo que habría que utilizar una remota. La única gratuita que se ha encontrado es mLab[16] , la misma que usa Heroku, por lo que volvemos al mismo problema del límite de tamaño.

Por último, se consideró utilizar una instancia de Amazon AWS EC2[17] . Es muy similar a DigitalOcean, se eligió el primero porque es más sencillo de utilizar.

Nanobox

- Herramientas consideradas: Heroku, Nanobox.
- Herramienta elegida: Nanobox.

Nanobox[18] es una herramienta que nos permite desplegar nuestra aplicación sin centrarnos en la infraestructura del servidor [**?**].

Para ello enlazamos nuestra cuenta de un proveedor en la nube (en este caso DigitalOcean) y nanobox se encargará de instalar el sistema operativo, configurarlo, instalar nuestra aplicación y sus dependencias y ejecutarla.

[14] DigitalOcean: https://www.digitalocean.com/
[15] GitHub Education: https://education.github.com/
[16] mLab: https://mlab.com/
[17] Amazon AWS EC2: https://aws.amazon.com/es/ec2/
[18] Nanobox: https://nanobox.io/

Heroku es un servicio muy similar, con la diferencia de que no podemos utilizar servidores en la nube externos, se descartó por lo ya explicado en el punto anterior.

Flask

- Herramientas consideradas: Flask, Django.
- Herramienta elegida: Flask.

Como uno de los objetivos del proyecto es el de crear una página web, se han considerado varios frameworks web para Python. Entre ellos se ha seleccionado Flask.

Flask[19] es un framework fácil de utilizar y muy flexible. No nos fuerza a utilizar una metodología específica y podemos organizar la aplicación con la estructura que queramos (a diferencia de *Django*) [**?**].

Además se incluyen herramientas para desplegar el servidor de desarrollo, para realizar pruebas de la aplicación y para hacer *APIs REST*.

Folium

- Herramientas consideradas: Plotly maps, Folium, Leaflet.
- Herramienta elegida: Folium.

Un aspecto importante de este proyecto es la representación de datos en el mapa.

Para ello se han considerado varias herramientas, de las cuales se ha elegido Folium[20] [**?**]. Este paquete nos permite visualizar datos manipulados con Python y visualizarlos como mapas utilizando para ello *LeafletJS* [**?**].

Se ha elegido esta herramienta porque nos permite representar mapas coropléticos (3.4) y mapas con clusters de datos [**?**] utilizando estructuras de datos geográficas personalizadas (*GeoJSON*) [**?**].

Dynatable

- Herramientas consideradas: Dynatable, Datatables.

[19] Flask: http://flask.pocoo.org/
[20] Folium: http://python-visualization.github.io/folium/

- Herramienta elegida: Dynatable.

Dynatable[21] es un framework de Javascript para visualizar tablas de una manera más clara y ordenada [**?**].

Este framework nos permite paginar los resultados, ordenar por alguna columna y buscar por cualquier campo dentro de la tabla. Todo esto sin tener que recargar la página.

Bootstrap

- Herramientas consideradas: Bootstrap, Foundation.
- Herramienta elegida: Bootstrap.

Bootstrap[22] es un framework para desarrollar páginas web con HTML, CSS y Javascript para facilitar el diseño de páginas responsive, que se adapten al tamaño de la pantalla del dispositivo.

Bootstrap nos ofrece una serie de clases de CSS predefinidas por lo que podemos crear la web de proyecto prácticamente sin tocar hojas de estilos CSS. Otra característica es que dispone de funciones de Javascript para llamar a eventos de ciertos elementos, como puede ser al pulsar un item de un menú desplegable.

PyCharm

- Herramientas consideradas: Visual Studio Code, PyCharm.
- Herramienta elegida: PyCharm.

PyCharm[23] es un IDE (*Entorno de Desarrollo Integrado*) para Python basado en IntelliJ. Posee herramientas para ayudarnos con el desarrollo como el autocompletado y herramientas para refactorizar de forma automática.

Se ha considerado porque, a diferencia de otros IDEs similares, tiene soporte para Flask y MongoDB, lo que nos facilita programar más rápidamente.

En este proyecto se ha utilizado la versión *Professional*, pero con la versión *Community* gratuita sería suficiente.

[21] Dynatable: https://www.dynatable.com/
[22] Bootstrap: https://getbootstrap.com/
[23] PyCharm: https://www.jetbrains.com/pycharm/

Herramientas de gestión

A continuación se van a describir las herramientas que se han utilizada gestionar el proyecto.

Github

- Herramientas consideradas: Github, BitBucket, GitLab.
- Herramienta elegida: Github.

GitHub[24] es un servicio web para alojar proyectos utilizando un sistema de control de versiones *Git*.

Se ha utilizado esta herramienta para alojar el repositorio del proyecto llevando un registro de todos los cambios realizados desde el inicio.

Además se han utilizado las *Issues* y *Milestones* de GitHub para planificar las tareas a realizar en cada sprint.

ZenHub

- Herramientas consideradas: Github Projects, ZenHub, GitKraken Glo.
- Herramienta elegida: ZenHub.

ZenHub[25] es una herramienta de gestión de proyectos integrada en Github mediante una extensión para el navegador. Nos proporciona un tablero cambas sobre el que mover las *issues* entre varios estados.

Las tareas se pueden estimar estableciendo un número '*Story points*' basados en la serie de Fibonacci en función del trabajo que se cree que se va a necesitar. Esto ha sido especialmente útil para planificar el tiempo empleado en cada tarea [**?**].

También se ha utilizado esta herramienta para extraer los *gráficos burn-down* de cada sprint presentes en el anexo.

GitKraken

- Herramientas consideradas: GitKraken, Github Desktop, Sourcetree.
- Herramienta elegida: GitKraken.

[24] GitHub: https://github.com/
[25] ZenHub: https://www.zenhub.com/

GitKraken[26] es un cliente con interfaz gráfica para gestión de proyecto *Git* para Windows, Mac y Linux. Nos permite realizar todas las acciones posibles con git, pero de manera mucho más sencilla y visual.

Permite sincronizarse con Github y otros servicios de alojamiento de repositorios de forma automática, de forma que el contenido del repositorio local y el remoto sea consistente.

Codacy

- Herramientas consideradas: SonarQube, Code Climate, Codacy.
- Herramienta elegida: Codacy.

Codacy[27] es una herramienta de análisis de calidad de código que ayuda a los desarrolladores a realizar código de más rápido y de mayor calidad.

Tiene métricas para medir la complegidad ciclomática, código duplicado, cubrimiento por tests y estadísticas para cada commit o pull request [**?**].

Durante el desarrollo de este proyecto se ha intentado mantener siempre una Certificación A en Codacy con todas las métricas al 100 % para asegurar la calidad del código.

[26] GitKraken: https://www.gitkraken.com/
[27] Codacy: https://www.codacy.com/

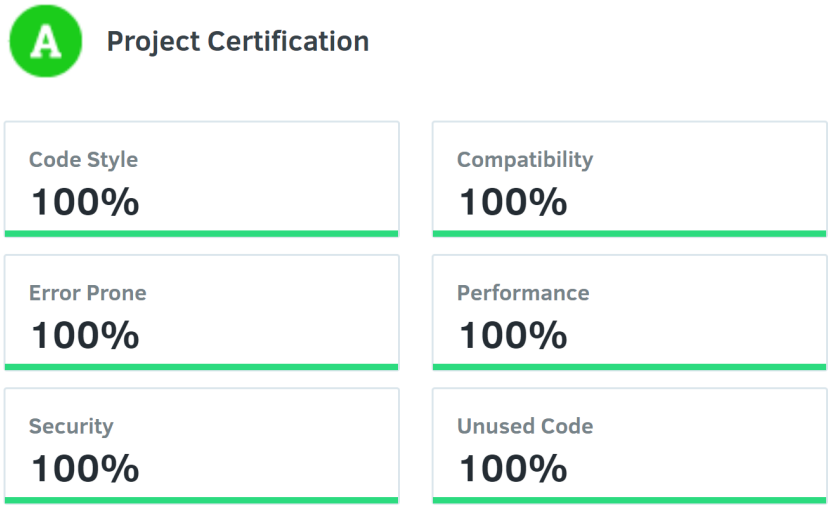

Figura 4.3: Certificado de Codacy del proyecto [?]

Herramientas de documentación

LaTeX

- Herramientas consideradas: Microsoft Word, LaTeX, LibreOffice, OpenOffice.
- Herramienta elegida: LaTeX[28] .

Esta memoria está escrita en LaTeX. Se trata de un sistema de preparación de documentos para la creación de documentos con una alta calidad tipográfica. Se usa frecuentemente en la escritura de libros y artículos científicos, entre otra cosas por la facilidad de incluir expresiones matemáticas [?] [?].

Para utilizar esta herramienta se ha utilizado TeXstudio[29] . Es un editor gratuito y multiplataforma para editar documentos escritos en Latex. Incluye iconos para localizar los comandos más comunes, corrector ortográfico, visor de PDF y descarga los paquetes de forma automática.

Este documento se ha escrito utilizando la siguiente plantilla de Latex [?].

[28]LaTeX: https://www.latex-project.org/
[29]TeXstudio: https://www.texstudio.org/

StarUML

- Herramientas consideradas: StarUML, ArgoUML, Astah.
- Herramienta elegida: StarUML.

StarUML[30] es una herramienta para modelar diagramas UML que nos permite crear diagramas de clases, objetos, casos de uso, componente, estructurales, de comunicación, de estado y de actividad y entidad-relación.

Funciona de manera gráfica, arrastrando objetos y conectándolos entre ellos permitiendo añadir atributos tanto a los objetos como a las relaciones.

[30] StarUML: http://staruml.io/

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto. Se comenta el ciclo de vida utilizado, los problemas encontrados durante el desarrollo y los detalles que se consideran más relevantes.

5.1. Formación

Para realizar estre proyecto se necesitaban conocimientos de los que no se disponía. Se han utilizado los siguientes recursos para aprender antes o durante la realización del proyecto.

Libros:

- Flask Web Development: Developing Web Applications with Python (Miguel Grinberg) [**?**].
- Scum Manager (Alexander Menzinsky, Gertrudis López, Juan Palacio) [**?**].
- Python Data Science Essentials (Alberto Boschetti, Luca Massaron) [**?**].
- UML Distilled (Martin Fowler) [**?**].
- Mastering Python (Rick van Hattem) [**?**].

Documentación oficial y cursos:

- Documentación de DigitalOcean [**?**].
- Documentación de Dynatable [**?**].
- Documentación de Folium [**?**].
- Documentación de Nanobox [**?**].
- Documentación de Pymongo [**?**].
- Information Service Engineering (Harald Sack, Maria Koutraki) [**?**].

5.2. Servidor en la nube

Como se trata de una aplicación web, un aspecto relevante es el despliegue de la aplicación en un servidor. Por lo que ya se comenzó a desplegar la aplicación desde un principio.

Inicialmente se empezó a utilizar un servidor gratuito de Heroku[31] , que al principio funcionaba bien, pero conforme se fueron añadiendo fuentes de datos, no se mostraban todos los datos.

Esto se debe a que heroku no instala la base de datos en su propio servidor, si no que utiliza un servicio para almacenar la base de datos de forma externa, mLab[32] . El problema con mLab es que tiene un límite de 500MB por base de datos y nuestra base de datos con todas las fuentes de datos ocupa alrededor de 3GB.

Para solventar este problema no quedaba más opción que cambiar de servidor de despliegue. Primero se consideró utilizar PythonAnywhere[33] , pero el problema es el mismo, utiliza también mLab para la base de datos.

Como no se encontraron más servicios gratuitos que soporten bases de datos MongoDB que no utilicen mLab, tuvimos que pasarnos a una plataforma de pago.

En este caso se consideraron las plataformas DigitalOcean[34] y Amazon Web Services ec2[35] . Se escogió DigitalOcean por su simplicidad, aunque AWS debería funcionar también perfectamente.

[31] Heroku: https://www.heroku.com/
[32] mLab: https://mlab.com/
[33] PythonAnywhere: https://www.pythonanywhere.com/
[34] DigitalOcean: https://www.digitalocean.com/
[35] Amazon Web Services ec2: https://aws.amazon.com/es/ec2/

El servidor que se eligió fue el más básico de DigitalOcean (4.2), un *Droplet*[36] con 1GB de memoria RAM, 1 CPU, 25GB de almacenamiento SSD Y 1TB de transferencia, suficiente para este proyecto. Con un coste de 5€ mensual.

Aunque el servidor, con el paquete de Github Education[37] tenemos un cupón de 50€ para gastar, con lo que podríamos mantener el servidor durante 10 meses. Se puede pedir un cupón por persona, no por proyecto, por lo que si se sigue con este proyecto en un TFG en el futuro, podría volver a pedirse un cupón para el despliegue.

5.3. Despliegue

Heroku ya nos facilitaba herramientas para desplegar el código directamente al servidor utilizando git, pero como tuvimos que abandonarlo, en un droplet no tenemos esta facilidad.

Se decidió optar por utilizar Nanobox[38] (4.2). Esta herramienta nos permite centrarlos en desarrollar código y no en el despliegue.

Desde nanobox creamos el droplet de DigitalOcean que hemos mencionado antes, de este modo ya no tenemos que preocuparnos por instalar el sistema operativo ni las dependecias o paquetes para hacer funcionar la aplicación.

Para configurar todo esto, en la raíz del proyecto hay un fichero *baxfile.yml* en el que se especifica el lenguaje de programación, la base de datos que vamos a utilizar, los paquetes adicionales a instalar y el ejecutable de la aplicación web.

Las dependencias de la aplicación deberán estar contenidas en el archivo *dependencies.txt*. De modo que en el momento del despliegue, estas dependencias se instalan automáticamente mediante pip.

Nanobox además nos permite escalar la aplicación si en algún momento se cree que no es suficiente con el droplet más básico de DigitalOcean, agregar más servidores para añadir redundancia a la aplicación y nos muestra estadísticas del uso de los recursos del servidor.

[36] Droplet: un servidor en la nube de DigitalOcean.
[37] Github Education: https://education.github.com/
[38] Nanobox: https://nanobox.io/

Otros aspecto relevante es que desde aquí se ha creado un certificado SSL para cifrar la conexión desde y hacia el servidor mediante https para mejorar la seguridad.

En los anexos, en el manual del programador, se explica con más detalle el proceso a seguir para desplegar la aplicación.

5.4. Mapas coropléticos

Uno de los requisitos del proyecto es el de mostrar datos en el mapa mediante mapas coropléticos (3.4). Para ello se optó por utilizar el paquete de python Folium (4.2).

Para representar los límites geográficos se utilizaron dos archivos geojson, uno con los límites de las provincias y otro con los límites municipales de toda España.

El primer problema encontrado fue que estos arquivos geojson eran demasiado grandes. La solución a este problema fue utilizar un herramienta, mapshaper[39] . Con esta herramienta se consiguió suavizar los bordes de las regiones minimizando el tamaño de los archivos geojson.

Otro problema encontrado es que al mostrar el mapa a nivel de municipios, no se consigue visualizar en Chrome. Esto se debe a que aunque los límites de municipio se han minimizado, se llega al límite establecido por el navegador para poder renderizar el mapa. No se puede minimizar más los límites de municipios porque se perderían algunos municipios, juntándose con los más grandes.

No se ha encontrado una solución para este problema, puede que para solucionarlo haya que cambiar de librería.

Por último, una funcionalidad que podría añadirse a los mapas es mostrar un dato al pasar el cursor sobre una región. Por ejemplo, al mostrar el mapa del paro y posicionarnos sobre Burgos, que se muestre el valor del paro en Burgos.

No se puede mostrar valores en mapas coropléticos con folium, es una limitación de la librería. Podría considerarse cambiar la librería a Plotly maps[40] , que si puede hacer esto, pero el trabajo necesario para cambiar de librería en este punto es demasiado grande, puede considerarse en trabajos futuros.

[39]mapshaper: http://mapshaper.org/
[40]Plotly maps: https://plot.ly/python/maps/

5.5. Dimensionalidad de los datos

Anteriormente comentábamos que tenemos una base de datos de 3GB. Concretamente las fuentes de datos provenientes de SEPE (Servicio Público de Empleo Estatal) son especialmente grandes, porque tienen información de todos los municipios durante todos los meses del año a lo largo de 15 años.

El problema viene al intentar mostrar información de una de estas fuentes grandes sin filtrar lo suficiente. Por ejemplo, elegir los datos de empleo del SEPE sin filtrar por alguna columna.

Como resultado devolvíamos una tabla con todos estos datos, lo que tarda demasiado en responder desde el servidor y en representarse. Incluso, en el servidor se quedaba sin memoria por la cantidad de datos a transferir.

La solución fue añadir un campo 'Filas a mostrar', por defecto 1000, en el que se especifica las filas que vamos a mostrar, por lo que ya no hay que transferir y representar todos los datos y la respuesta es mucho más rápida.

Al exportar la consulta a csv o json si que se transfieren todos los datos, pero como no hace falta representarlo, el proceso es mucho más rápido y ya no supone un problema.

5.6. Almacenamiento de datos

En un primer momento se pensó en almacenar los datos en una única colección en MongoDB, diferenciando las fuentes de datos con un atributo. Como es una base de datos no relacional nos permite hacer esto sin ningún problema.

El problema está en la consulta de datos, al tener que buscar en toda la colección cada vez que se haga una consulta, aunque sea muy simple, como la colección es tan grande la consulta tarda bastante. Se consideró que el timpo de carga no es aceptable.

La solución fue almacenar cada fuente de datos en una colección diferente, de modo que si queremos acceder a nos datos del ine, accedemos sólo a la colección del ine; si hacemos una consulta conjunta entre datos del ine y la agencia tributaria, sólo tendremos que comparar los valores en estas dos colecciones.

Como resultado obtenemos unas consultas mucho más rápidas, aunque complica un poco la consulta de datos y no es tan propio de las bases de

datos no relacionales.

5.7. Seguridad

Un tema importante a estudiar es si puede haber algún problema de seguridad en la aplicación.

Respecto a inyecciones NoSQL en la base de datos MongoDB. El plugin *flask-wtforms*, que es el encargado de renderizar los formularios, ya se encarga de sanear los entradas para prevenir inyecciones. En todo caso, como las únicas operaciones que se pueden hacer desde la aplicación web son de lectura, no sería demasiado importante que se consultasen más datos de los permitidos porque son públicos y están disponibles abiertamente.

En el peor de los casos, si se llegara a modificar o borrar la base de datos, sólamente habría que volver a ejecutar el script para actualizar las fuentes de datos para volver a generarla en buen estado.

Otro tema de seguridad es la entrada al introducir consultas calculadas. Internamete se trata esta entrada con un *eval* de python, lo que codacy considera como un problema de seguridad. Esta operación no se puede implementar con otros métodos como *literal_eval* porque se utilizan objetos de pandas. Como las únicas acciones que se pueden hacer son de escritura, no supone un gran problema, igual que en los párrafos anteriores, lo único que podría pasar es lanzar excepciones no tratadas.

Trabajos relacionados

Este capítulo está divido en dos secciones. Primero, se explica brevemente artículos de investigación de minería de datos relacionados con este proyecto. En la segunda sección se comparan algunas páginas web que visualizan datos públicos con este proyecto.

6.1. Artículos de investigación

Crime prediction through urban metrics and statistical learning

En este artículo se estudian las causas que pueden aumentar el índice de criminalidad utilizando técnicas de minería de datos. Para ello se utiliza un *Random Forest* para predecir el número de homicidios en una ciudad a partir de datos sociológicos y demográficos. [**?**]

Como resultado se obtiene un 97 % de precisión utilizando el coeficiente de determinación R^2 [**?**] en la predicción de índices de criminalidad y se ordenan los indicadores que causan estos crímenes en función de su importancia.

Detecting and investigating crime by means of data mining: a general crime matching framework

En este artículo, al igual que el anterior, se utilizan técnicas de minería de datos para identificar las características que pueden llevar a cometer un crimen [**?**].

En esta aproximación se toma como entrada informes policiales narrativos escritos en texto plano. De estos textos se extraen las características para

determinar la gravedad del delito utilizando análisis de grupos mediante *clustering*.

6.2. Páginas web

Mapa del paro

Mapa del paro[41] es una página web que nos permite visualizar el paro por municipios, provincias y comunidades en un mapa coroplético. Ver figura 6.4.

Los datos del paro contenidos en esta página web también están en este proyecto, con la ventaja de que además se pueden combinar con otros datos públicos y se puede descargar la información.

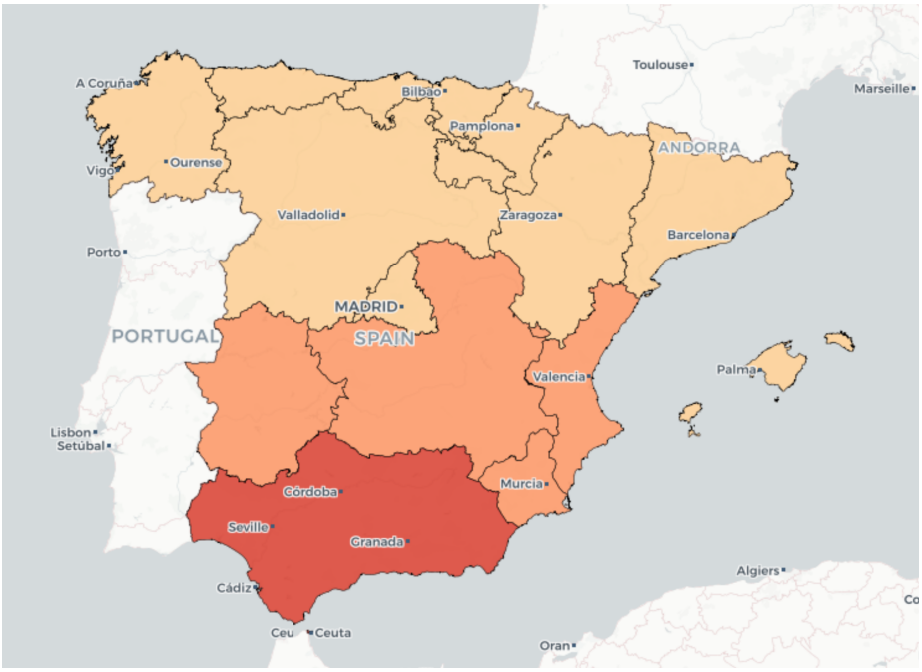

Figura 6.4: Paro por comunidades autónomas

Ciudatos

Ciudatos[42] nos permite visualizar varios datos públicos de Colombia mediante gráficas de barras y posicionarlos en el mapa. Ver figura 6.5.

[41] Mapa del paro: http://mapadelparo.com/
[42] Ciudatos: http://www.ciudatos.com/visualizacion

Esta herramienta no nos permite visualizar los datos con mapas coropléticos, simplemente sitúa los gráficos de barras de cada región en su correspondiente sitio.

Como ventaja nuestra aplicación tiene datos de España y permite visualizar datos resultantes de juntar varias subconsultas.

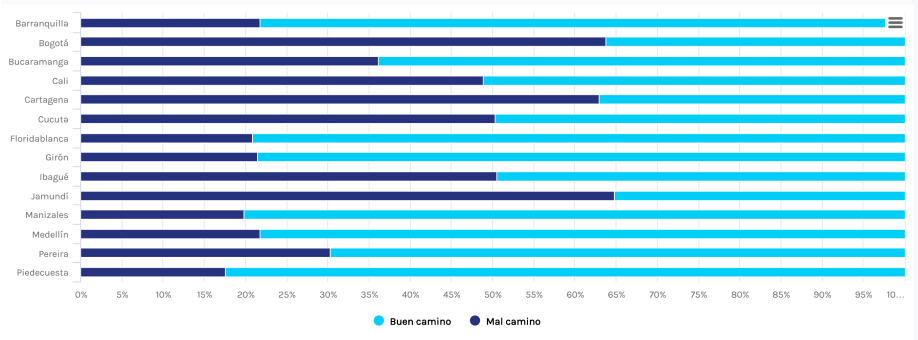

Figura 6.5: Bienestar por ciudades (Colombia)

Eurostat

Otra representación de datos públicos en mapas es Eurostat[43] .

En esta representación se muestra en el mapa los porcentajes de empleo en cada país de Europa. Ver figura 6.6.

Estos datos se escapan del alcance al que se quería llegar en este trabajo, pero podría considerarse para integrarlos en algún proyecto futuro.

[43]Eurostat: http://ec.europa.eu/eurostat/en/web/lfs/statistics-illustrated

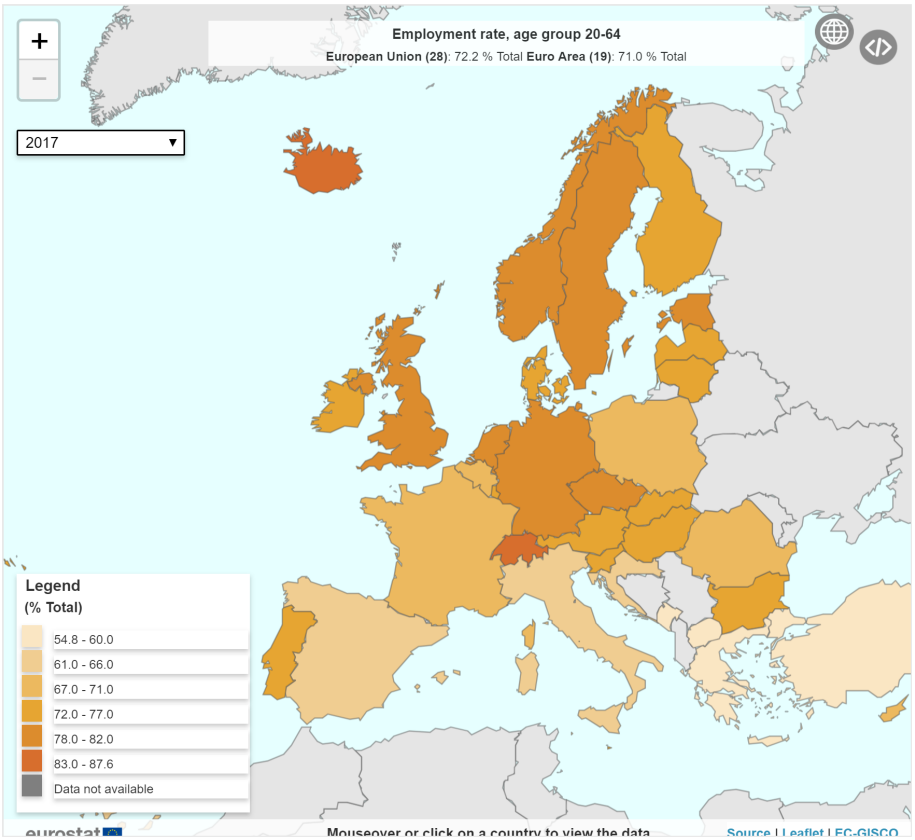

Figura 6.6: Porcentaje de empleo por país (Europa)

Comparación

Por último, se en la tabla 6.1, se ha realizado una comparación entre las características que ofrece este proyecto y las páginas web relacionadas.

Herramientas	TFG	Mapa del paro	Ciudatos	Eurostat
Exportación	✓	✗	✓	✓
Filtro de datos	✓	✗	✓	✓
Múltiples fuentes	✓	✗	✓	✗
Columnas calculadas	✓	✗	✗	✗
Mapas interactivos	✓	✓	✓	✓
Región	España	España	Colombia	Europa
Nivel	Provincias Municipios	Comunidades Provincias Municipios	Ciudades	Países

Tabla 6.1: Comparativa con herramientas de visualización de datos

Conclusiones y Líneas de trabajo futuras

En este capítulo se exponen las conclusiones derivadas del desarrollo del proyecto y comentarios para continuar con el trabajo.

7.1. Conclusiones

Tras haber acabado con la codificación del proyecto, creo que se han cumplido los objetivos que se habían planteado al comenzar con el proyecto. Como resultado se ha obtenido una herramienta que puede ser muy útil para ser utilizada en procesos de minería de datos que requieran datos demográficos y sociológicos en el ámbito español.

A nivel personal se han adquirido conocimientos de como utilizar *bases de datos no relacionales*, de las que no se había tratado nada en la carrera y me interesaba aprender.

También se han ampliados los conocimientos en el desarrollo de páginas web, tanto en la parte de *front-end* con el diseño y funcionamiento dinámico de la página, como en la parte de *back-end* con la codificación en una aplicación con Python, sin experiencia hasta ahora.

Otro tema tratado del que no conocía hasta ahora es la utilización y configuración de *servidores en la nube*, en este caso se han utilizado para desplegar la página web.

Puedo concluir que estoy satisfecho con el trabajo realizado en este proyecto, la experiencia ha sido muy positiva y he aprendido mucho sobre temas de los que conocía poco hasta ahora.

7.2. Líneas de trabajo futuras

Este trabajo es el inicio de un proyecto más completo y más ambicioso que se pretende continuar en trabajos de fin de grado futuros. A continuación se muestra una lista de posibles tareas a realizar en trabajos futuros:

- Mejoras las columnas calculadas:
  - Validar si la consulta es correcta dinámicamente con *Javascript*.
  - Permitir consultar más de una columna calculada.
  - Permitir elegir el nombre de las columnas calculadas.
  - Mejorar el autocompletado al introducir operadores.
- Hacer que los datos estén disponibles en la web semántica siguiendo la descripción de RDF [**?**].
- Mejorar la visualización de mapas: mostrar el valor de los atributos al pasar el ratón sobre una región, hacer que funcione el renderizado de mapas en Chrome. Puede que haya que cambiar de framework para representar mapas.
- Hacer pruebas para comprobar si las fuentes de datos siguen funcionando.
- Integrar más fuentes de datos estatales. Como por ejemplo:
  - Matrimonios por provincia de residencia del matrimonio, mes de celebración y forma de celebración del matrimonio [44].
  - Balances de criminalidad [45].
  - Estadísticas catastrales [46].
  - Información meteorológica [47].
- Integrar fuentes de datos de empresas privadas:
  - Tendencias de búsqueda como *Google Trends* [48].
  - Porcentajes de audiencia por municipio (televisión, radio, periódico).

[44] http://www.ine.es/dynt3/inebase/index.htm?padre=3406
[45] http://www.interior.gob.es/prensa/balances-e-informes/2017
[46] http://www.catastro.minhap.es/esp/estadisticas_2.asp
[47] http://www.aemet.es/es/datos_abiertos/catalogo
[48] https://trends.google.es/trends/?geo=ES

- Compilar la aplicación para juntar CSV en un ejecutable con *PyInstaller* o crear un paquete de *pip*.