UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster del

Máster Universitario en Ingeniería y Ciencia de Datos

IDENTIFICACIÓN DE ÁREAS DE INTERÉS PARA LA IMPARTICIÓN DE CURSOS DE FORMACIÓN

Daniel Ortega Diánez

Dirigido por: Dr. D. Agustín Carlos Caminero Herraez

Dr. D. José Manuel Cuadra Troncoso

Curso: 2022-2023: 1ª Convocatoria

Resumen

El mercado laboral español actual se caracteriza por presentar un desajuste entre la oferta y la demanda laboral. Una de las principales causas de dicho desajuste es la falta de disponibilidad de perfiles profesionales cualificados. Es por esta causa, y por la naturaleza dinámica propia de la sociedad del conocimiento, que es recomendable potenciar la investigación en las necesidades formativas del mercado laboral, acercando la oferta formativa de universidades y otros centros de formación a las necesidades actuales de las empresas.

En este contexto, para facilitar este acercamiento entre entidades formativas y empresas, y la detección temprana de las necesidades del mercado, es preciso disponer de herramientas automatizadas que recopilen información del laboral y la presenten de forma adecuada para la detección de áreas de interés para la elaboración de programas formativos.

Con este objetivo, este Trabajo Fin de Máster aborda el desarrollo de un sistema de información de apoyo a la toma de decisiones para la elección dichas áreas de interés, basándose en la información publicada en portales web de búsqueda de empleo.

En esta memoria se presentan las conclusiones obtenidas del estudio de las alternativas del panorama tecnológico actual para atender a las distintas áreas funcionales que aplican a este sistema, y expone los resultados del desarrollo e implantación de la plataforma integral para la extracción y presentación de datos de ofertas de empleo producto de este trabajo.

Palabras clave

Scraping web, Proceso ETL, Conteneirización, Cuadro de mando, Empleo, Formación

Abstract

The current Spanish labor market is characterized by presenting a mismatch between labor supply and demand. One of the main causes of this mismatch is the lack of availability of qualified professional profiles. It is for this reason, and due to the dynamic nature of the knowledge society, that it is advisable to promote research into the education needs of the labor market, bringing the training offer of universities and other training centers closer to the current needs of companies.

In this context, to facilitate this rapprochement between training entities and companies, and the early detection of market needs, it is necessary to have automated tools that collect labor information and present it adequately for the detection of areas of interest for the development of education programs.

To achieve this objective, this Final Master's Project implements the development of an information system to support decision-making for the choice of these areas of interest, based on the information published on job search web portals.

This work memory presents the conclusions obtained from the study of the alternatives of the current technological landscape to attend to the different functional areas that apply to this system, and exposes the results of the development and implementation of an integral platform for the extraction and presentation of data from job offers resulting from this work.

Keywords

Web Scraping, ETL Process, Containerization, Dashboard, Job, Education

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción, motivación y objetivos

Este primer capítulo tiene como finalidad presentar al lector el objeto de este Trabajo de Fin de Máster. Para ello, en primer lugar se expone qué es lo que se espera obtener de este trabajo, introduciendo sus objetivos particulares, para a continuación introducir cómo ha sido plasmado su desarrollo en la estructura de esta memoria, y finalmente incluir un mapeo entre los objetivos particulares del trabajo y los componentes de la estructura de la memoria.

1.1. Motivación y objetivos

El mercado laboral español actual se caracteriza por presentar un desajuste entre la oferta y la demanda laboral. Una de las principales causas de dicho desajuste es la falta de disponibilidad de perfiles cualificados [Servicio Público de Empleo Estatal (2023)]. Es por esta causa, y por la naturaleza dinámica propia de la sociedad del conocimiento, que es recomendable potenciar la investigación en las necesidades formativas del mercado laboral, acercando la oferta formativa de universidades y otros centros de formación a las necesidades actuales de las empresas.

El propósito de este trabajo es facilitar este acercamiento mediante la puesta a disposición de herramientas software que permitan la identificación de las necesidades laborales actualizadas de las empresas. Para ello, en este trabajo se aborda el desarrollo e implantación de un sistema de información de apoyo a la toma de decisiones en la elección de áreas de interés para la impartición de cursos de información, basándose en la información publicada en portales web de búsqueda de empleo.

Para llevar a cabo el desarrollo e implantación de dicho sistema se identifican los siguientes objetivos:

OBJ-01 Estudiar las tecnologías, conceptos y patrones de diseño aplicables a la solución final.

OBJ-02 Construcción de un motor de extracción de datos de portales web de empleo enfocado en facilitar las siguientes cuestiones:

- Inclusión sencilla de artefactos software parseadores de distintos portales de empleo.
- Adaptación de dichos artefactos a cambios de diseño de la interfaz gráfica de los portales.
- Incorporación dinámica y desacoplada de elementos software que permitan tratar los datos extraídos.

OBJ-03 Proporcionar interfaces gráficas dinámicas y herramientas de diseño on-line de las mismas, para la explotación en tiempo real de la información obtenida, con el propósito de la identificación de posibles áreas formativas de interés en función de su demanda en el mercado laboral.

OBJ-04 Implantación de un motor de flujos de trabajo que permita la ejecución independiente, fiable, periódica, automatizada y controlada, de los procesos implicados en:

- Las tareas de extracción de los datos obtenidos de los portales de empleo.
- El posterior tratamiento de los datos obtenidos para su puesta a disposición de las áreas usuarias implicadas.
- La notificación y/o persistencia de los productos resultantes de los procesos anteriores.

OBJ-05 Entrega empaquetada de la arquitectura completa del sistema que posibilite su despliegue en diferentes plataformas software de forma estandarizada, rápida, sencilla y automatizada.

OBJ-06 Implementar un sistema que lleve a cabo un uso integral de los componentes descritos sobre casos de uso reales de portales de empleo de distinta naturaleza.

OBJ-07 Elaborar de una memoria final en la que se detalle y evalúe el proceso para la consecución de cada uno de estos objetivos y sus resultados.

1.2. Estructura de la memoria

La memoria de este proyecto, producto de la consecución del objetivo 7 del apartado anterior, se estructura en los siguientes capítulos:

1. Introducción, motivación y objetivos. Presenta la motivación y objetivos de este trabajo y la estructura de la presente memoria.
2. Estado del arte. Desarrolla y evalúa las opciones contempladas para la consecución de los objetivos anteriormente presentados.
3. Detalle de la propuesta. Expone el análisis y diseño detallado del sistema, y los elementos más relevantes de la implementación de la solución tecnológica propuesta, dando cobertura a las necesidades de las distintas áreas funcionales derivadas de los objetivos planteados.
4. Evaluación. Ligado directamente al objetivo 7 del apartado anterior, evalúa la solución propuesta en el capítulo anterior, tomando como referencia su aplicación los resultados obtenidos en los casos de uso de portales de empleo de distinta naturaleza.
5. Conclusiones y trabajo futuro. Presenta las conclusiones de este proyecto y posibles trabajos futuros en relación con este trabajo.

1.3. Relación entre los objetivos del trabajo y la estructura de la memoria

Id.	Capítulo	Sección
OBJ-01	2. Estado del arte	Todas las secciones
OBJ-02	2. Estado del arte	2.2. Extracción de datos de portales web
	3. Detalle de la propuesta	3.1. Introducción
		3.2. Subsistema de extracción de datos
	4. Evaluación	4.2. Extracción de datos de portales web
OBJ-03	2. Estado del arte	2.3. Presentación y explotación de información
	3. Detalle de la propuesta	3.1. Introducción
		3.3. Subsistema de presentación y explotación de información
	4. Evaluación	4.3. Presentación de información

Id.	Capítulo	Sección
OBJ-04	2. Estado del arte	2.4. Planificación y ejecución de flujos de trabajo
	3. Detalle de la propuesta	3.1. Introducción
		3.4. Subsistema de planificación y ejecución de flujos de trabajo
	4. Evaluación	4.4. Planificación y ejecuciones de flujos de trabajo
OBJ-05	2. Estado del arte	2.5. Empaquetado del sistema
	3. Detalle de la propuesta	3.1. Introducción
		3.5. Infraestructura del sistema
	4. Evaluación	4.5. Infraestructura de despliegue
OBJ-06	3. Detalle de la propuesta	Todas las secciones
	4. Evaluación	Todas las secciones
OBJ-07	Todos los capítulos	Todas las secciones

Capítulo 2

Estado del arte

Con el objetivo de ubicar al lector en el contexto tecnológico presente en relación con los aspectos en los que se basa la solución que desarrolla este proyecto, en este capítulo se introducen las distintas opciones del panorama tecnológico actual tenidas en cuenta para la consecución de los objetivos del presente trabajo.

2.1. Introducción

Como se introdujo en el capítulo anterior, el mercado laboral español actual se caracteriza por presentar un desajuste entre la oferta y la demanda laboral. Una de las principales causas de dicho desajuste es la falta de disponibilidad de perfiles con una cualificación suficiente para satisfacer dicha demanda [Servicio Público de Empleo Estatal (2023)].

En este sentido, según refleja en [Adecco Group Institute (2022)], se estima que ocho de cada diez directores de recursos humanos expresan tener problemas a la hora de reclutar talento en sus empresas, indicándose además que una media del 9 % de las vacantes disponibles en el mercado de trabajo español se quedan sin cubrir.

Otro indicador de interés a este respecto es una de las conclusiones de la última "Encuesta trimestral de Coste Laboral" [Instituto Nacional de Estadística (2023)], en la que se revela una cifra de 149.645 empleos sin cubrir en España para el primer en el primer trimestre de 2023, la mayor registrada hasta el momento.

Teniendo estos aspectos en cuenta, es recomendable potenciar la investigación en las necesidades formativas del mercado laboral, acercando la oferta formativa de universidades y otros centros de formación a las necesidades actuales de las empresas.

Con este fin, en este trabajo se propone la puesta a disposición de herramientas software para facilitar dicho acercamiento. Estas herramientas deben servir como medio para la identificación de las necesidades laborales actualizadas de las empresas, presentando características relevantes de las vacantes laborales que ofrece el mercado que faciliten la identificación de áreas de interés para el

diseño de ofertas formativas orientadas a aumentar el éxito en la contratación.

A este respecto, tras haber realizado una exploración exhaustiva de soluciones, no se han encontrado soluciones software gratuitas y accesibles que atiendan a lo expresado, dando cobertura a la totalidad de los objetivos del presente trabajo para la explotación de información de portales web -de empleo o más generalistas- (ver 1.1. Motivación y objetivos). Es por esta circunstancia que en lo que sigue se plantean diferentes perspectivas del estado arte agrupadas según las áreas funcionales del trabajo, extraídas a partir de los objetivos referidos:

- Extracción de datos de portales web
- Presentación y explotación de información
- Planificación y ejecución de flujos de trabajo

Por último, en la sección Empaquetado del sistema, se presentan las tendencias tecnológicas actuales en relación con la distribución y despliegue de un sistema como el de este trabajo.

2.2. Extracción de datos de portales web

Este área funcional se encuentra directamente relacionado con el objetivo 2 del trabajo:

OBJ-02 Construcción de un motor de extracción de datos de portales web de empleo enfocado en facilitar las siguientes cuestiones:

- Inclusión sencilla de artefactos software parseadores de distintos portales de empleo.
- Adaptación de dichos artefactos a cambios de diseño de la interfaz gráfica de los portales.
- Incorporación dinámica y desacoplada de elementos software que permitan tratar los datos extraídos.

Este estudio se centra en la identificación de herramientas que, de forma forma rápida, simple y gratuita, den respuesta a los siguientes dos dimensiones:

- La naturaleza cambiante cambiante de los portales web actuales, principalmente causada por la necesidad de actualizaciones frecuentes de sus pilas tecnológicas, nuevas funcionalidades y cambios en el diseño de sus interfaces gráficas.
- La posibilidad de incorporación de cambios de forma eficaz y eficiente en los elementos software que extraen los datos de los portales web, para evitar pérdidas de información significativas en la serie temporal objeto de estudio.

El mercado tecnológico actual presenta una gran variedad de plataformas integrales de webcrawling, como *APIFY* (https://apify.com/), *import.io* (https://www.import.io/) o *parsehub* (https://www.parsehub.com/), no obstante, además de carecer de la flexibilidad y sencillez requeridas, la mayor parte de ellas de un modo u otro están ligadas a diferentes planes de pago. Es por ésto que el desarrollo de una solución a medida como motor de extracción de datos se considera la opción más viable para el caso que nos ocupa.

Para la construcción de un motor de extracción de datos que cumpla con los requerimientos de mantenibilidad anteriores es recomendado recurrir a paradigmas de programación y criterios de diseño consolidados en la industria del desarrollo software, como pueden ser la programación orientada a objetos o la programación funcional [Barnes and Kolling (2007); Fowler (2005); Gortázar et al. (2019); Hernández et al. (2003); Ostenero (2014); Sommerville (2005)] y patrones de diseño orientados a objetos [Eckel (2003); Fowler (2005); Freeman and Freeman (2004); Gamma et al. (2006); Larman (2010); Shalloway and Trott (2004)].

En cuanto a la aplicación de estos paradigmas y criterios de diseño, se requiere del uso de un lenguaje de programación consolidado, de alto nivel, que mantenga unos niveles aceptables de simplicidad, seguridad y adaptabilidad en la construcción/mantenimiento de software. La oferta de lenguajes que pueden servir a este cometido es muy amplia, presentándose como opciones más populares *Python*, *Javascript*, *Java* y *C++* [github.com (2022); jetbrains.com (2022); tiobe.com (2023)]. De las opciones indicadas, en los últimos años *Python* ha ido ganado en popularidad con respecto al resto de lenguages en el ámbito de la ingeniería y ciencia de datos debido a su facilidad aprendizaje, su orientación multiparadigma, y su amplio ecosistema de bibliotecas y frameworks para el tratamiento de datos. Entre dichas bibliotecas/frameworks las siguientes pueden ser de utilidad para el caso que nos ocupa:

- *Scrapy*: Se define como un framework de código abierto para la extracción de datos de sitios web de forma rápida, sencilla y extensible [scrapy.org (2023)]. Útil para el escaneo a gran escala de sitios web, aprovechando las características de navegación entre distintas páginas.

- *Beautiful Soup*: Definida como una librería para la extracción de datos de archivos XML y HTML, configurable a nivel del parseador de elección para este propósito [Richardson (2023)], muy eficaz en el escaneo de archivos de marcado de formá rápida y con bajo coste de desarrollo y mantenimiento.

- *Selenium*: Se compone de un conjunto de librerías para la interacción con páginas web mediante drivers que controlan navegadores web y procesan sus contenidos [Selenium community (2023)]. Resulta especialmente provechosa para la interacción con páginas dinámicas cuyo renderizado se hace del lado del cliente.

- Librerías para la operación vectorizada y/o tabulada de datos como *pandas* (https://pandas.

pydata.org/) y *NumPy* (https://numpy.org/), que simplifican la operativa datos estructurados en forma de vectores o tablas.

- Librerías para la descarga de datos mediante HTTP/FTP, como *Requests* (https://requests.readthedocs.io/en/latest/) y *urllib* (https://requests.readthedocs.io/en/latest/).

Por tanto, motivado por todo lo anteriormente expuesto, las características del trabajo y la oferta tecnológica actual parecen indicar que la solución más adecuada para este área es un desarrollo a medida, apoyado en un lenguaje de programación de alto nivel, como puede ser *Python* , que permita explotar sus mecanismos de extensión de componentes software base para la adaptación a nuevos requerimientos, y que disponga de un amplia oferta de librerías/frameworks para la extracción y tratamiento de la información en función de las características de cada portal web.

2.3. Presentación y explotación de información

Este área se encuentra ligada con siguiente objetivo del trabajo:

> **OBJ-03** Proporcionar interfaces gráficas dinámicas y herramientas de diseño on-line de las mismas, para la explotación en tiempo real de la información obtenida, con el propósito de la identificación de posibles áreas formativas de interés en función de su demanda en el mercado laboral.

Para este área se pretende obtener como resultado un sistema que permita, a partir de la información obtenida de los portales de empleo, la realización de operaciones de selección, clasificación y filtrado mediante gráficos dinámicos e interactivos que ayuden a la identificación de áreas funcionales de interés para la realización de cursos de formación. Para estos objetivos las opciones de visualización más adecuadas resultan ser los cuadros de mando (dashboards) [Zamorano et al. (2019)].

En la actualidad se dispone de gran variedad de opciones consolidadas para la elaboración de cuadros de mando y su puesta a disposición para su mantenimiento por parte del área usuaria, por lo que, a priori, y debido a su complejidad, no resulta razonable plantear un desarrollo a medida para el presente trabajo. A continuación se valoran tres de las opciones de más éxito para dicho cometido:

- *Kibana* (https://www.elastic.co/guide/en/kibana) Forma parte de la pila tecnológica de *Elasticsearch* (*ELK*), cuyo núcleo se compone del motor de búsqueda que le da nombre (*Elasticsearch*), que dota a *Apache Solr* (https://solr.apache.org/) -un potente producto de búsqueda basado en índices invertidos- de capacidades para su ejecución en clúster y de una *API REST* para la gestión y uso de sus documentos. Si bien *Kibana* fue creado originalmente para el análisis de logs, su uso se ha extendido en multitud de empresas y organizaciones para el análisis de datos alfanuméricos de cualquier tipo, debido a la versatilidad de sus gráficos, y a las posibilidades de escalado y velocidad de ejecución que le brinda *Elasticsearch*. Además de

dashboards, *Kibana* proporciona una suite integrada para el análisis de información (*Discover*), una interfaz web para la gestión de todo su stack tecnológico y servicios *REST* para la gestión de sus objetos.

- Tipos de orígenes de datos soportados: Sólo soportado origen de datos de *elasticsearch*.
- Licenciamiento: Versión gratuita disponible con licencia básica. Existen también opciones de pago con funcionalidades extendidas (https://www.elastic.co/es/subscriptions).

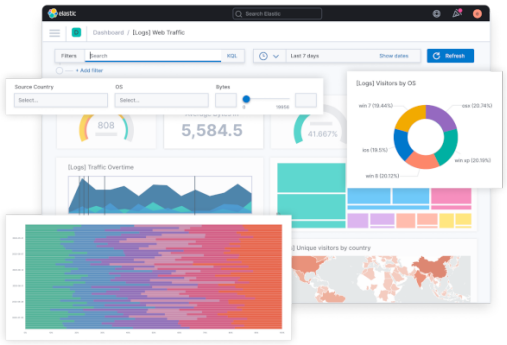


Figura 2.1: Ejemplo de cuadro de mando *Kibana*

- *Grafana* (https://grafana.com) *Grafana* surgió como un fork de *Kibana* para dar cobertura a soporte a métricas que no recibían demasiado mantenimiento por parte del equipo del *Kibana*. Se enfoca sobre todo en la visualización de series temporales de datos, especialmente para sistemas de monitorización, y en la actualidad es ampliamente utilizada como herramienta de visualización gracias su versatilidad en cuanto a los orígenes de sus fuentes de información se refiere.
  - Tipos de orígenes de datos soportados: Múltiples (*MySQL*, *PostgreSQL*, *Amazon Cloud*, *Azure*, *Elasticsearch*...)
  - Licenciamiento: Versión gratuita disponible como código abierto. Existe también una versión de pago con capacidades adicionales hospedada en la su propia infraestructura.

Figura 2.2: Ejemplo de cuadro de mando *Grafana*

- *Tableau* (https://www.tableau.com/) Tableau es una de las opciones comerciales más extendidas para la visualización de datos en el campo de análisis de negocio. Dispone de un ecosistema software muy completo para la gestión e implantación de visualizaciones, y se caracteriza por su facilidad de uso y su capacidad para extraer información de una gran cantidad de plataformas.

  - Tipos de orígenes de datos soportados: Múltiples (*CSV*, *Excel*, *Oracle*, *Sql Server*, *DB2*, *Azure*, *Google Cloud*...)

  - Licenciamiento: De pago en función de diferentes opciones de hosting on-premise y cloud.

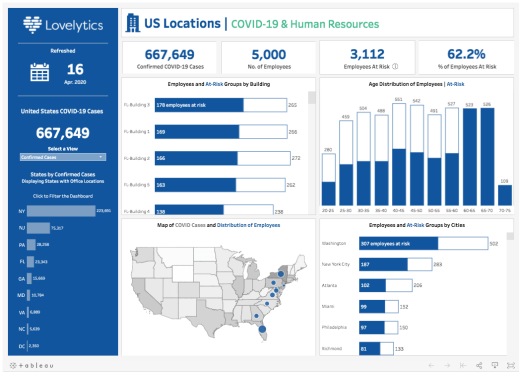


Figura 2.3: Ejemplo de cuadro de mando *Tableau*

Además de *Tableau*, existen más alternativas de pago muy competitivas, como *Microsoft Power BI* (https://powerbi.microsoft.com/es-es/) o *Qlik Sense* (https://www.qlik.com/us/), pero que quedan descartadas como opciones para el presente trabajo, debido a que ha decidido optar por tecnologías gratuitas. Por tanto, de las opciones presentadas, por su gratuidad, versatilidad y soporte por parte de la comunidad, tanto *Kibana* como *Grafana* pueden cubrir los requerimientos de este trabajo como herramientas de visualización.

2.4. Planificación y ejecución de flujos de trabajo

El objetivo del trabajo vinculado a este área funcional es el que sigue:

OBJ-04 Implantación de un motor de flujos de trabajo que permita la ejecución independiente, fiable, periódica, automatizada y controlada, de los procesos implicados en:

- Las tareas de extracción de los datos obtenidos de los portales de empleo.
- El posterior tratamiento de los datos obtenidos para su puesta a disposición de las áreas usuarias implicadas.
- La notificación y/o persistencia de los productos resultantes de los procesos anteriores.

Las publicaciones de las ofertas de trabajo en la mayor parte de los portales de búsqueda de empleo se caracterizan ser efímeras, es decir, las accesibles en un momento determinado suelen ser sólo aquéllas cuyas vacantes se encuentran sin cubrir. En el estudio de la información obtenida, para reducir la exposición a sesgos vinculados la duración de publicación de las ofertas, es recomendable que el sistema disponga de un componente software que periódicamente procese la información de distintos portales, haciendo uso del motor de extracción de datos, de modo que su los resultados persistan en el tiempo. De esta forma, escogiendo una frecuencia de muestreo adecuado, se dispondría de información de la mayor parte de las ofertas, independientemente de la duración de su publicación.

Se requiere, además, que estos datos obtenidos por el sistema de extracción sean a continuación limpiados, tratados, normalizados y almacenados para su inmediata explotación por parte del área usuaria desde el sistema de presentación y explotación.

La secuencia de ejecución descrita se corresponde con un proceso comúnmente conocido en el ámbito de las tecnologías de información como de extracción, transformación y carga: ETL (extract, transform, load) [El-Sappagh et al. (2011); Theodorou et al. (2017); Seenivasan (2023); Bansal (2014)]. La ejecución repetida de este tipo de procesos lleva aparejada la recomendación de disponer de mecanismos de registro, notificación y consulta de resultados de ejecución, para la identificación temprana de posibles errores por fallos de comunicación o cambios en las plataformas por los motivos expuestos en el apartado 3.2. Subsistema de extracción de datos, que puedan afectar a la calidad del juego de datos de estudio.

La oferta actual de productos software fiables y gratuitos que dan soporte a las cuestiones expuestas es amplia, por lo que no resulta recomendable acometer un desarrollo a medida de un planificador o un motor de ETL. A continuación se describen algunas alternativas:

- *Apache Airflow* (`https://airflow.apache.org/`) *Apache Airflow* es una plataforma de código abierto para la gestión de flujos de trabajo con un potente planificador tolerante a fallos. Es altamente escalable y extensible, y se caracteriza por la la definición de flujos de trabajo mediante grafos acíclicos dirigidos (DAG) programados directamente en sencillos scripts *Python*. Dispone de una interfaz web de usuario muy completa que permite la gestión y ejecución planificada de sus tareas, y la supervisión del resultado de sus ejecuciones. Dispone

además de una gran variedad de operadores y conectores para integrarse con distintas plataformas tecnológicas mediantes APIs, y de la posibilidad de programar conectores a medida.

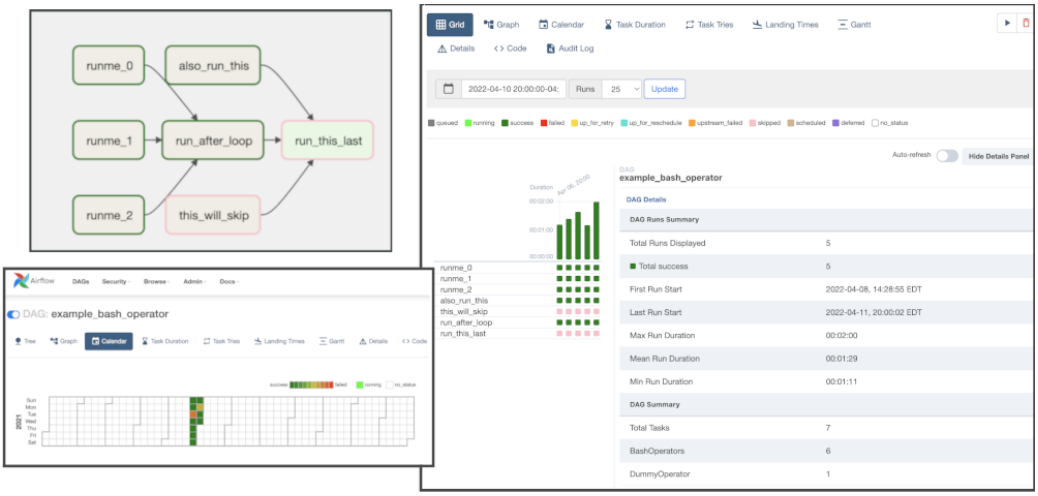


Figura 2.4: Interfaz web *Apache Airflow*

- *Argo Workflows* (https://argoproj.github.io/) *Argo Workflows* es la alternativa a *Apache Airflow* específica para *pods* de *Kubernetes*. Dispone también de una interfaz web para la gestión y planificación de sus flujos de trabajo, que también se definen mediante DAGs, pero en este caso mediante el patrón *GitOps* en diferentes formas, como ficheros *YAML*/*json*, aplicaciones *kustomize* o *helm charts*. También es altamente escalable, extensible mediante plantillas y proporciona integraciones nativas con múltiples plataformas.

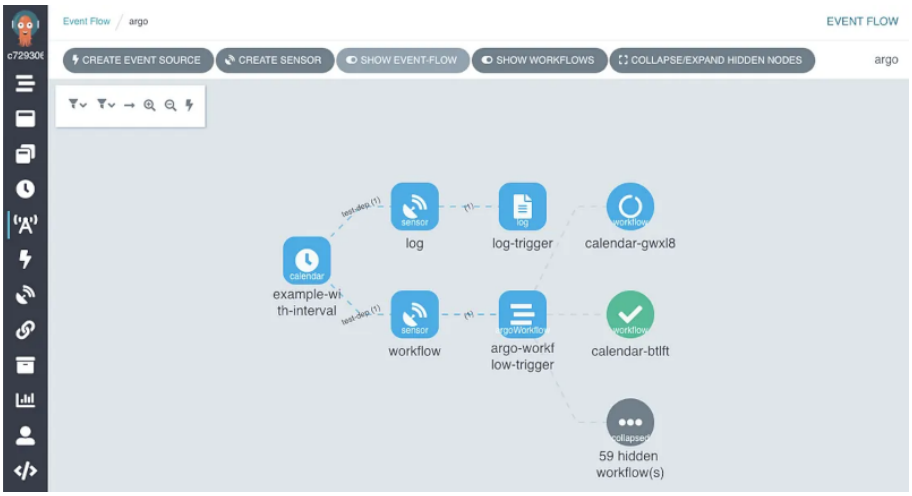


Figura 2.5: Interfaz web *Argo Workflows*

- *Apache NIFI* (https://nifi.apache.org/) *Apache NIFI* es un sistema de propósito específico que, mediante una interfaz web muy sencilla, mediante el uso de conectores multiplataforma y multiformato permite, sin necesidad de programar, definir y ejecutar flujos completos

de transformación de datos. Estos operaciones permiten hacer tareas múltiples como filtrados, validaciones, uniones, divisiones, modificaciones, etc. Además dispone de características para el desarrollo de componentes y servicios a medida. Dispone de una API REST para la ejecución de flujos.

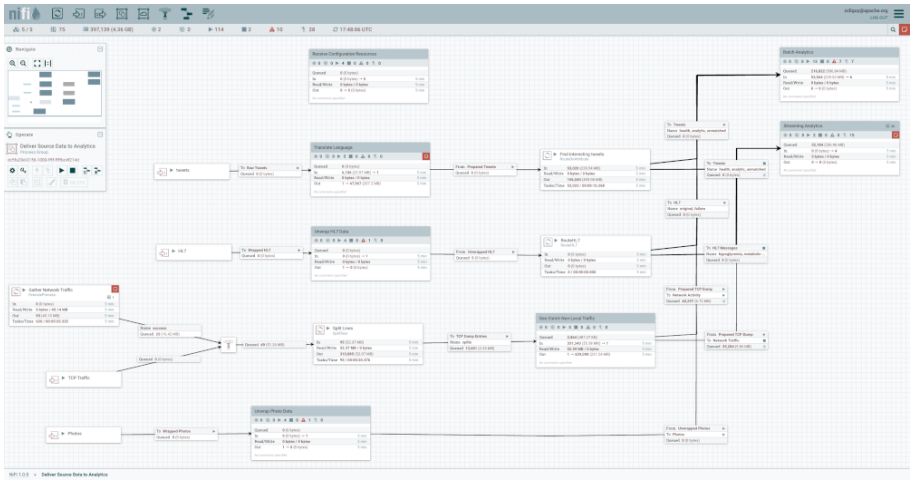


Figura 2.6: Interfaz web *Apache NIFI*

Otras soluciones equivalentes muy competitivas para la definición gráfica de transformaciones de datos son *Hitachi Vantara* (https://www.hitachivantara.com/) y *talend* (https://www.talend.com/), pero para las que la puesta en producción de sus flujos de transformación se requiere recurrir a plataformas que no son gratuitas.

Como resumen de lo concerniente a este área funcional, tras el estudio de varias opciones se concluye que la solución más indicada para el caso que nos ocupa es recurrir a un planificador de flujos de trabajo, como *Apache Airflow* o *Argo workflows*, para cubrir las necesidades del sistema en cuanto a la ejecución y supervisión de las tareas de ETL. Además, en el caso de disponer de juegos de datos que requieran transformaciones complejas podría llegar a ser interesante optar por una solución que combine planificadores de flujos de trabajo con herramientas específicas de diseño y ejecución de flujos de transformación de datos, integrando dentro de los flujos de trabajo de los planificadores llamadas al API de un sistema como *Apache NIFI* para la ejecución de dichas transformaciones mediante los flujos definidos en último este sistema o uno equivalente.

2.5. Empaquetado del sistema

El objetivo concerniente a esta sección es el que sigue:

OBJ-05 Entrega empaquetada de la arquitectura completa del sistema que posibilite su despliegue en diferentes plataformas software de forma estandarizada, rápida, sencilla y automatizada.

De las conclusiones de las secciones anteriores se deduce que el sistema que da respuesta a este trabajo se va a componer de múltiples componentes software que interactúan entre sí mediante protocolos de comunicaciones de red estandarizados. Para llevar a cabo el despliegue de dicha infraestructura en cualquier plataforma, de forma estandarizada, rápida y sencilla, es recomendable recurrir a capas software de virtualización que abstraigan y aíslen, en la medida de lo posible, la arquitectura y características del sistema operativo del servidor o servidores anfitriones de las requerida por dichos componentes software, y de la infraestructura de red subyacente.

Durante los últimos años las tendencias de virtualización en la industria software se suelen dirigir hacia soluciones orientadas a la conteneirización de servicios [Watada et al. (2019); Schenker et al. (2019)] para las cuestiones planteadas en esta sección. Un contenedor es una unidad de software ligera que empaqueta el código y todas sus dependencias, incluyendo todo lo necesario para la ejecución de una aplicación en distintos entornos. En gran medida, el éxito de este tipo de soluciones es debido a sus capacidades de escalabilidad y al ahorro en infraestructura que suponen estas arquitecturas frente a los modelos de virtualización tradicionales, especialmente aplicables en el ámbito de arquitecturas software orientadas a la ejecución de múltiples servicios ligeros e independientes que se comunican entre sí [Scheepers (2014); Turnbull (2014); da Cunha et al. (2021)].

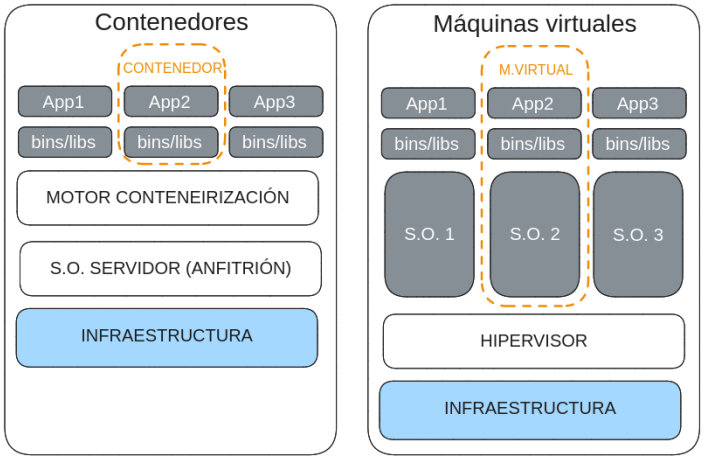


Figura 2.7: Arquitectura conteneirizada/máquinas virtuales

Para el empaquetado y despliegue de contenedores, *Docker* (https://www.docker.com/) se ha convertido en el estándar de facto gracias a su simplicidad de configuración y despliegue, y su registro de imágenes de contenedores centralizado [Rad et al. (2017); Nickoloff and Kuenzli (2019)]. Se trata de una plataforma integral que permite, a los desarrolladores y resto de actores implicados en el ciclo de vida del software, crear y probar contenedores rápidamente. Se encuentra disponible para múltiples arquitecturas hardware y sistemas operativos. Entre las herramientas que proporciona *Docker* que pueden ser de utilidad para este trabajo, cabe destacar de *Docker Compose*, que facilita la definición de la arquitectura de aplicaciones *Docker* basadas en múltiples contenedores.

En cuanto a la orquestación de contenedores en múltiples equipos anfitriones, *Docker* proporciona

Docker Swarm, que entre otras muchas características permite la definición de criterios adicionales de escalado de contenedores, de balanceo de carga o de actuación ante no disponibilidad de contenedores. Otra opción para la orquestación es *Kubernetes* (https://kubernetes.io/), un potente sistema para el automatizado de despliegues, escalado y gestión de contenedores. Si bien *Kubernetes* puede resultar algo más complejo de aprender que *Docker Swarm*, el primero ofrece interesantes opciones para la orquestación de contenedores a gran escala y su despliegue en infraestructuras cloud que le han convertido en líder en la industria en la orquestación de contenedores [Flexera (2023); Cormier (2022)].

En conclusión, las características de este trabajo apuntan a la utilización de tecnologías basadas en la conteneirización de servicios. La tecnología concreta a utilizar dependerá de las características de la infraestructura de despliegue, optando por una solución basada únicamente en *Docker* con *Docker Compose* para un sistema en el que se disponga de un único servidor anfitrión, y por *Docker Swarm* o *Kubernetes* para arquitecturas multinodo, decantándose por una u otra de estas dos últimas en función de las necesidades de escalado, automatización de la gestión y de la infraestructura de hosting.

2.6. Resumen

Como resultado del estudio del panorama tecnológico actual, en este capítulo se han presentado distintas alternativas como posibles soluciones para cada una de las áreas funcionales del sistema de información objetivo de este trabajo. Como se ha podido comprobar, el número de combinaciones de posibles soluciones es amplio. Es por este motivo que se requiere un análisis más detallado de las características de cada una de estas áreas para que, teniendo en consideración las particularidades de cada una de las alternativas estudiadas en este capítulo, se diseñe y construya una solución adecuada. Esta cuestión se aborda en el próximo capítulo.

Capítulo 3

Detalle de la propuesta

En este capítulo se expone la propuesta técnica que da solución a las necesidades planteadas en este trabajo y su aplicación a los dos casos de uso presentados con anterioridad. Para ello, en la primera sección, Introducción, se da una introducción a los elementos que componen el sistema y una descripción de la estructura seguida en las siguientes secciones de este capítulo, en las que se detalla la solución del presente trabajo.

3.1. Introducción

Componentes del sistema

Para dar cobertura a las áreas funcionales identificadas en el capítulo anterior (ver 2. Estado del arte) se ha diseñado una solución tecnológica basada en tres subsistemas que responden de forma independiente e individualizada a dichas necesidades:

- Subsistema de extracción de datos. Encapsula las necesidades relativas a la extracción directa de datos de los portales de empleo, abstrayendo al resto de subsistemas de estos detalles.
- Subsistema de presentación y explotación de información. Constituye la interfaz gráfica para la explotación de información por parte de los usuarios finales del sistema, para el estudio de datos y la identificación de áreas de interés en la elaboración de cursos de formación.
- Subsistema de planificación y ejecución de flujos de trabajo. Su objetivo principal es hacer de intermediario entre los dos subsistemas anteriores, ejecutando la extracción de datos de los portales de forma controlada y planificada, transformando dichos datos en información atendiendo a las necesidades del subsistema de presentación, y poniendo a su disposición dicha información.

En cuanto a componentes usuarios del sistema, los perfiles identificados y sus correspondientes responsabilidades son los que siguen:

- Desarrollador de software, ingeniero de datos y administrador de sistemas: Ejecución de las labores de mantenimiento, y desarrollo de nuevas funcionalidades y adaptaciones del sistema. Son los perfiles informáticos responsables del correcto funcionamiento técnico del sistema en todas las fases de su ciclo de vida.
- Científico de datos: Extracción de conocimiento y definición de modelos para este cometido a partir del estudio de datos.
- Usuario final: Consulta de las herramientas de visualización de información de ofertas de empleos para la identificación de áreas de interés para la impartición cursos de formación.

A continuación se incluye una representación de los subsistemas en la que se aprecia el flujo de datos descrito entre los distintos componentes, y los perfiles de sus usuarios potenciales:

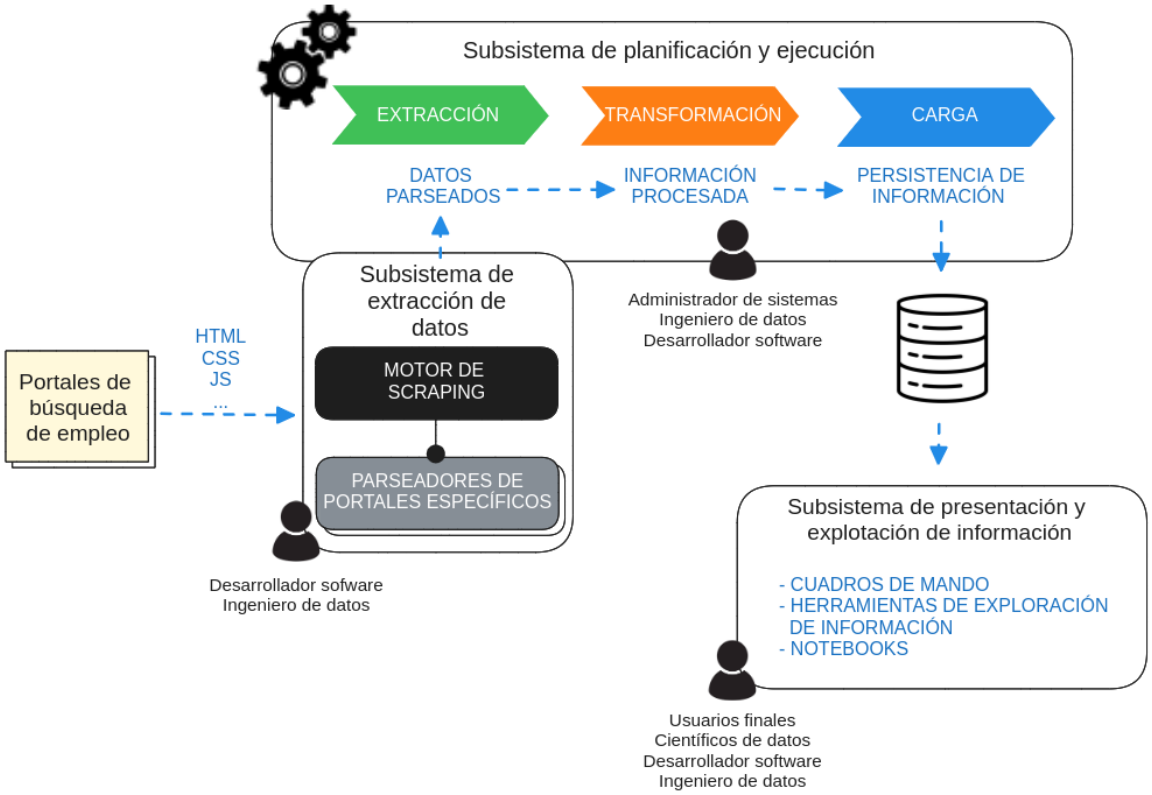


Figura 3.1: Flujo de información entre los componentes del sistema

Presentación de la solución en esta memoria

Para el análisis, diseño y construcción del sistema se ha seguido una metodología incremental [Kroll and Kruchten (2003); Zamorano et al. (2019); Larman (2010)] que ha abarcado el sistema completo, comenzando por los elementos de infraestructura y guiada en sus iteraciones por el análisis de los datos obtenidos para los casos de uso objeto de este trabajo. No obstante, para facilitar la comprensión del alcance del sistema por parte del lector de esta memoria, en las siguientes secciones

se describe el sistema abordando cada uno de los tres subsistemas referidos de forma individualizada, en lugar de seguir el orden cronológico de su desarrollo.

El orden de presentación de los distintos subsistemas responde a fines didácticos: en primer lugar se da una visión de las cuestiones relacionadas con la herramienta para la obtención los datos en origen (Subsistema de extracción de datos), a continuación se presenta el producto final para la presentación y explotación de información (Subsistema de presentación y explotación de información), y finalmente se describe la solución que permite la transición de datos entre ambos subsistemas (Subsistema de planificación y ejecución de flujos de trabajo).

Estas tres secciones comparten estructura, dividiéndose en los siguientes apartados:

1. **Análisis**, que aborda la descripción de la solución desde un punto de vista funcional.
2. **Diseño**, donde se presenta la solución técnica.
3. **Aplicación a los casos de uso**, donde se detalla la aplicación de los distintos subsistemas a dos portales de distinta naturaleza, de acuerdo con el objetivo 6 de este trabajo:

   > **OBJ-06** Implementar un sistema que lleve a cabo un uso integral de los componentes descritos sobre casos de uso reales de portales de empleo de distinta naturaleza.

   Estos casos de uso incluyen la explotación diaria de las ofertas publicadas dos portales de ofertas de empleo de España:

   - *Tecnoempleo* (`https://www.tecnoempleo.com/`), como origen de ofertas de empleo en el ámbito de las tecnologías de información.
   - *Infoempleo* (`https://www.infoempleo.com/`), que tiene un enfoque a ofertas de empleo de ámbito más generalista.

Para concluir este capítulo, en la sección Infraestructura del sistema, se presenta la arquitectura completa del sistema y la solución desarrollada para su empaquetado y distribución.

3.2. Subsistema de extracción de datos

Este subsistema da respuesta al área funcional "Extracción de datos de portales web", directamente relacionada con el objetivo 2 del trabajo:

OBJ-02 Construcción de un motor de extracción de datos de portales web de empleo enfocado en facilitar las siguientes cuestiones:

- Inclusión sencilla de artefactos software parseadores de distintos portales de empleo.
- Adaptación de dichos artefactos a cambios de diseño de la interfaz gráfica de los portales.
- Incorporación dinámica y desacoplada de elementos software que permitan tratar los datos extraídos.

Dentro del plano general del sistema, se puede identificar este subsistema como el componente que ejerce de interfaz entre los portales de empleo y el Subsistema de planificación y ejecución de flujos de trabajo, abstrayendo al resto del sistema de las características técnicas de cada portal e introduciendo el concepto abstracto de tarea de ejecución de extracción de la información de un portal.

3.2.1. Análisis

Para la extracción de los datos de portales web de empleo, el motor referido en el objetivo vinculado a este subsistema ha de interactuar con los componentes software de dichos portales web, entre los que se identifican dos tipos fundamentales:

- **Páginas de resultados de búsqueda**: Muestran la información, generalmente paginada, de la relación de ofertas de trabajo que cumplen con los criterios de búsqueda empleados, y dan acceso a las fichas de cada una de esas ofertas de empleo.
- **Fichas de detalle de oferta de empleo**: Aglutinan los datos relativos a los detalles de cada una de las ofertas de empleo.

Proceso de extracción de información

Este motor ha de facilitar el proceso de extracción de información de los componentes referidos, consistente en una secuencia iterativa de pasos de procesado y navegación por las distintas páginas vinculadas, que se resume en los siguientes puntos:

1. Carga de página inicial de resultados, a partir de los criterios de búsqueda indicados como punto de entrada.

2. Extracción de puntos de acceso a fichas de resultados de la página de resultados obtenida.
3. Carga de cada una de la fichas de resultados.
4. Extracción de datos de la ficha de resultados cargada.
5. Extracción de punto de acceso a la siguiente página de resultados de búsqueda -o característica de scroll al siguiente subgrupo de resultados en el caso de tratarse de paginado con carga dinámica de grupos de resultados por desplazamiento vertical en la página-.
6. Si se identifica en el paso anterior que hay más resultados, obtención de éstos y ejecución de nuevo de los pasos 2, 3, 4, 5, y 6 (si aplica).

A continuación se incluye un ejemplo ilustrativo en el que se muestra una representación gráfica de los componentes y el proceso descritos, para los que se han rodeado con cajas de borde rojo punteado los datos susceptibles de ser extraídos:

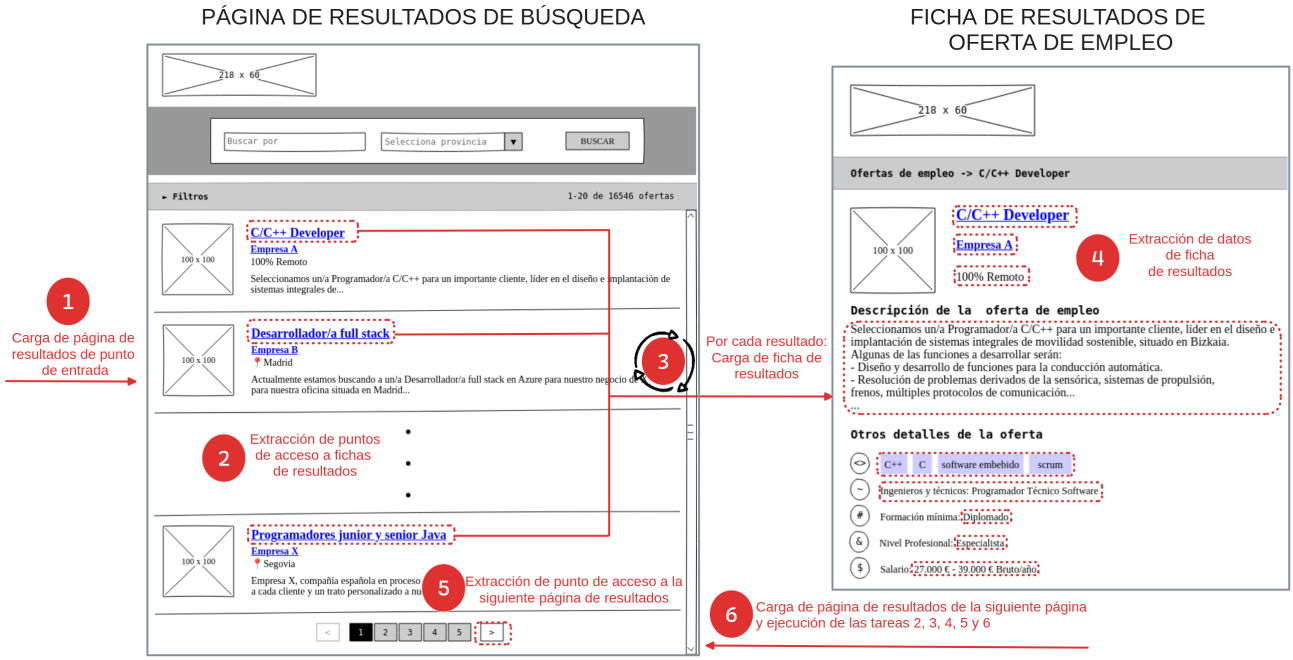


Figura 3.2: Ciclo de extracción de datos de un portal web de empleo

Motor de scraping e implementaciones específicas

Como se puede inferir de la descripción del objetivo de este subsistema, se espera que la librería que da respuesta a este motor disponga de un componente de propósito general, el **motor de scraping**, que abstraiga la ejecución el proceso de extracción, que es común para todos los portales, de los siguientes elementos:

- Las implementaciones dependientes de cada portal, en función de sus particularidades técnicas y funcionales, entre los que se encuentran:

- Los **parseadores de portales específicos**, encargados del procesamiento de los componentes de los portales web para la obtención particular de los datos requeridos en los pasos 2, 4 y 5 del ciclo de extracción de datos descrito en esta subsección.
- Los componentes de descarga de la información de portales (**requests**), encargados de la implementación de los protocolos de comunicación con los portales.

- Las características del tratamiento y utilización de los datos extraídos, requeridas para distintas ejecuciones de la librería, e implementadas por los distintos **observadores de resultados de parseo**, que acometen acciones relacionadas con el tratamiento de dichos datos para su persistencia o presentación durante la ejecución del proceso de extracción de información.

Las abstracciones descritas y su interactuación completa para distintas configuraciones tecnológicas y portales, se representan en el siguiente gráfico de flujo de datos de ejemplo:

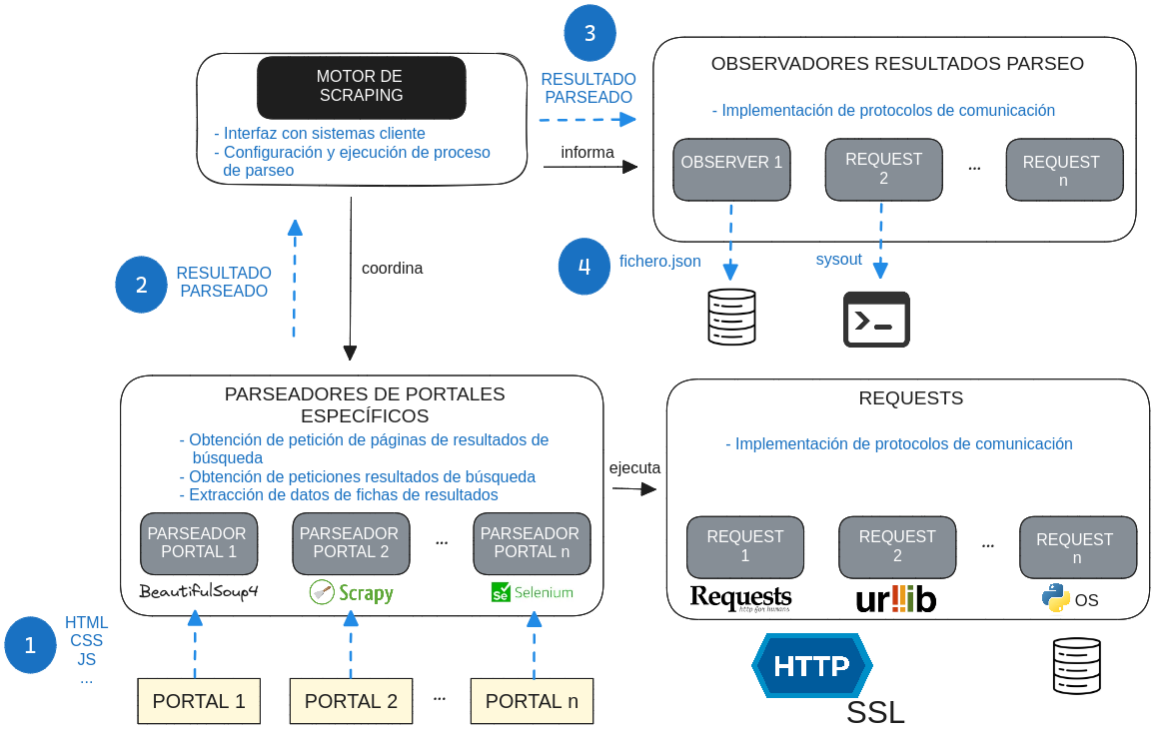


Figura 3.3: Flujo de datos entre componentes del motor de extracción

3.2.2. Diseño

Para el desarrollo de la biblioteca, siguiendo las conclusiones obtenidas del estudio del arte abordado en la sección 2.2. Extracción de datos de portales web, se ha recurrido a un lenguaje de alto nivel multiparadigma como *Python*. En su desarrollo se han aprovechado las características de extensibilidad, agregación, asociación y composición que brinda su parte de orientación a objetos,

aplicando soluciones basadas en patrones de diseño [Eckel (2003); Fowler (2005); Freeman and Freeman (2004); Gamma et al. (2006); Larman (2010); Shalloway and Trott (2004)] para dar respuesta a las características indicadas en el objetivo de este subsistema. Para mantener unos niveles adecuados de extensibilidad y mantenibilidad se han respetado en gran medida las recomendaciones y principios reflejados en [Martin (2009, 2003)].

La librería se ha diseñado de forma que provee un núcleo, que se ha denominado *uned.scraping_lib*, para que el desarrollador usuario sólo se tenga que preocupar de implementar, mediante extensiones de clases e interfaces, los elementos susceptibles de cambio que se identificaron en la subsección anterior (parseadores, requests y observadores). A continuación se incluye un diagrama de clases general que refleja las relaciones entre las entidades del núcleo (paquete *uned.scraping_lib*) y con sus extensiones (paquete *uned.scraping_impl*) :

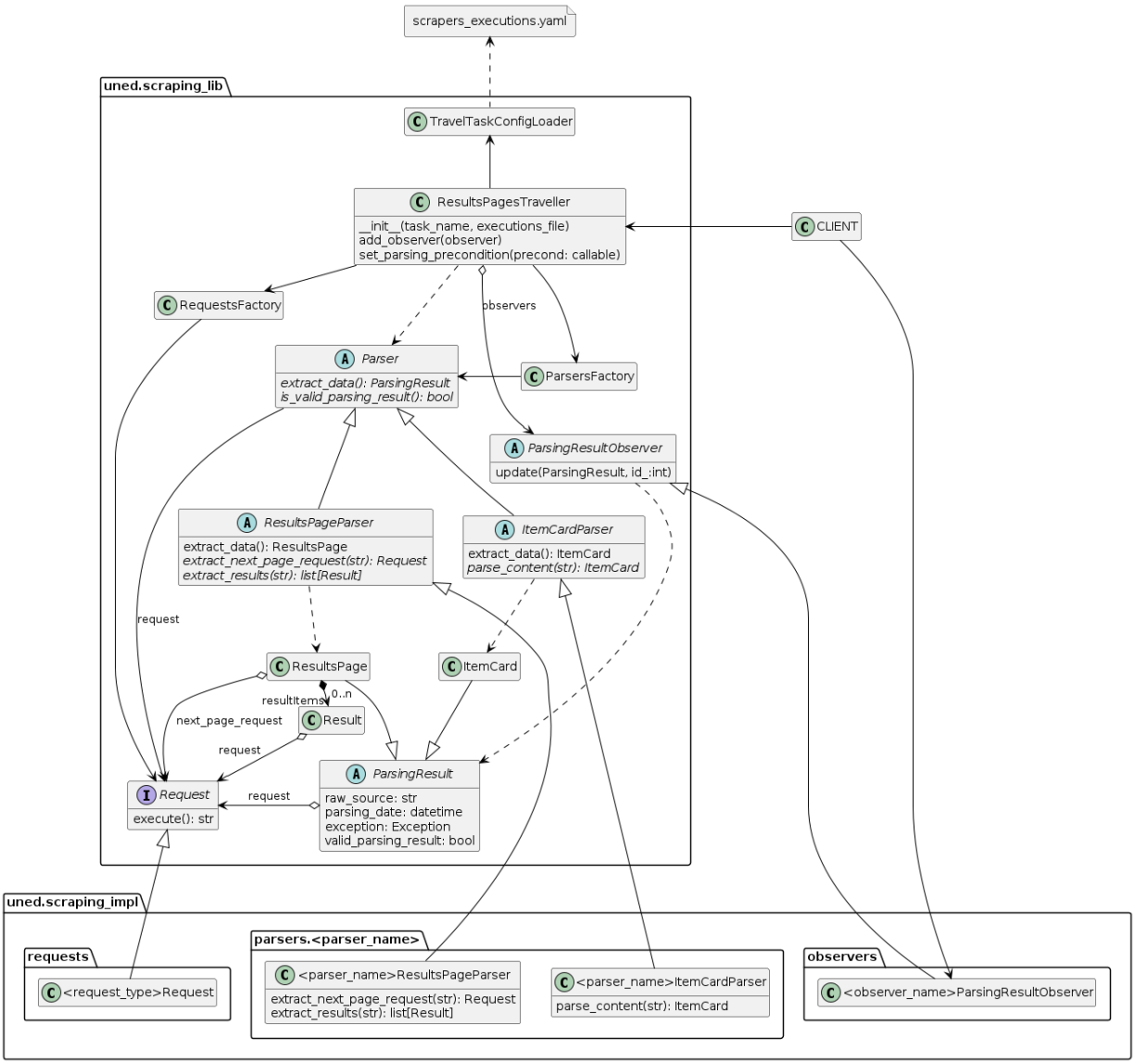


Figura 3.4: Diagrama de clases general de *uned.scraping_lib*

Parser se corresponde con la abstracción correspondiente a los extractores de información de páginas, siendo más concretamente ***ResultsPageParser*** e ***ItemCardParser*** las correspondientes a los parseadores de páginas de búsqueda y resultados respectivamente. Estas dos últimas clases abstractas han de extenderse para que, mediante la utilización del patrón método plantilla, el núcleo extraiga los elementos necesarios para la navegación entre los resultados de búsqueda y los datos concretos de cada página de detalle.

ParsingResult es la abstracción para los datos obtenidos de los parseos, que se concretan en ***ResultsPage*** para las páginas de búsqueda e ***ItemCard*** para las de detalle. Esta última puede ser extendida para ampliar su conjunto de atributos al ser utilizada como contenedor de datos para el objeto devuelto en la implementación del método de extracción de datos de las páginas de detalle (*ItemCardParser.**parse_content***), correspondiente al paso 4 del Proceso de extracción de información.

La interfaz ***Request*** tiene que ser extendida para la implementación de los protocolos de comunicación para la descarga de la información de portales, de los que sus objetos son devueltos en la implementación del método *ResultsPageParser.**extract_next_page_request***, cuya operativa se corresponde con el paso 5 del Proceso de extracción de información, y en la defínción de puntos de entrada e items de resultados.

Result representa los resultados de la página de búsqueda que son utilizados como objetos devueltos en la implementación del método *ResultsPageParser.**extract_results***, necesario para la implementación del paso 2 del Proceso de extracción de información.

ResultsPagesTraveller implementa el patrón fachada para el acceso de los clientes de la biblioteca a las funciones de parseo. De este modo, estos clientes únicamente han de definir las características de la tarea a ejecutar mediante un fichero de configuración, los observadores de los resultados y ejecutar el Proceso de extracción de información completo a través de la invocación de *ResultsPagesTraveller.**run***. Además, mediante el método ***set_parsing_precondition***, permite incluir una función de precondición que será evaluada durante el proceso para decidir si se procesa cada resultado de la búsqueda.

Para la incorporación dinámica y desacoplada de elementos software que permitan tratar los datos extraídos se ha recurrido a la implementación del patrón observador, cuyos suscriptores son definidos mediante la extensión de la clase abstracta ***ParsingResultObserver***. Como publicador de eventos ejerce *ResultsPagesTraveller*, que aporta el método ***add_observer*** para la adición de suscriptores que son informados del resultado del parseo de cada uno de los componentes web durante la ejecución de su método ***run***.

En cuanto a la utilización de los componentes anteriores, extendidos para la ejecución de la tarea de ejecución, la fachada del motor se encarga de cargar el fichero de configuración, mediante ***TravelTaskConfigLoader***, que aporta la información necesaria para la carga dinámica de los parseadores correspondientes, la request correspondiente a su punto de entrada y el tiempo de latencia entre ejecuciones de peticiones. Para la carga de los correspondientes componentes se hace uso del patrón

método factoría y de características propias de *Python* para la carga de dinámica de componentes a partir de nombres de ficheros de ficheros, implementadas en ***ParsersFactory*** y ***RequestsFactory***. (Para más información acerca del formato del fichero de configuración, acudir al punto Definición de tareas de extracción, en la página 28).

3.2.3. Aplicación a los casos de uso

A continuación se indican las adaptaciones realizadas para la aplicación del motor de parseo a los dos casos de uso descritos en 3.1. Introducción.

Extensiones de peticiones y observadores

En relación con el uso de la librería en lo que respecta a la extensión de ***Request***, se contemplan dos implementaciones:

1. ***HttpRequest***. Los portales de los casos de uso hacen uso de peticiones *HTTP* para la obtención de sus páginas de resultados y de detalle, con la particularidad de que *Infoempleo* utiliza peticiones *POST* para la parametrización de los accesos a las páginas de resultados de la búsqueda y *GET* para las de detalle, mientras que *Tecnoempleo* utiliza peticiones *GET* para ambos casos. Es por este motivo que se ha desarrollado esta extensión para la atención de cualquier petición *HTTP* independientemente del método utilizado, basada en la biblioteca *requests*, y cuyo uso podría extenderse a cualquier portal que haga peticiones de este tipo.

2. ***LocalFileRequest***. Implementación para la extracción de datos desde fichero de datos *HTML* almacenados localmente, con el objetivo de la ejecución de tests unitarios y de integración de la biblioteca, aplicando ejemplos de los casos de uso (ver "Pruebas automatizadas unitarias y de integración" en página 30).

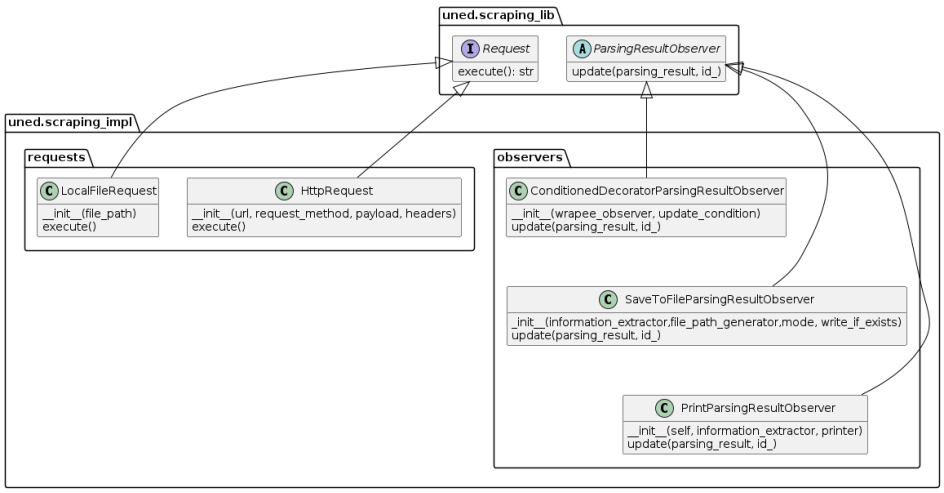


Figura 3.5: Diagrama de clases general de definición de peticiones y observadores para los casos de uso

Como definición de observadores, se han incluido tres opciones que dan cobertura a casuísticas de propósito general:

1. ***ConditionedDecoratorParsingResultsObserver***. Implementa el patrón decorador para la ejecución condicionada del observador decorado, basando su ejecución en el resultado de la ejecución de la condición indicada en su constructor para la entidad parseada en el momento de la publicación del evento de suscripción.

2. ***SaveToFileParsingResultObserver***. Su objetivo es el guardado de ficheros, aprovechando el uso de lambdas de la programación funcional de *Python* para la obtención de la ruta de destino y el contenido del fichero, en función de la entidad parseada para la que se publica el evento en el momento de ejecución.

3. ***PrintParsingResultObserver***. Aplicación para la impresión por consola de resultados, utilizando también lambdas para la extracción del contenido a imprimir de la entidad parseada.

Extensión de parseadores

Para la definición de los parseadores, en primer lugar se relacionan los atributos de las ofertas de trabajo. Como datos comunes a los dos casos de uso se identifican los siguientes:

		Ejemplos	
Atributos	**Multiv.**	**Infoempleo**	**Tecnoempleo**
id		2951615	rf-1973o674ck0c822951ja
Título		*Mozo/a Piscina/Playa*	*Analista Funcional Java*
Empresa		*Hotel ejemplo S.L.*	*Tecnoexample S.L.*
Localización		*Marbella (Málaga)*	*Sevilla*
Experiencia		*Entre 3 y 5 años de experiencia*	*3-5 años*
Vacantes		*5*	*5*
Tipo de contrato		*Contrato de duracion determinada*	*Indefinido*
Salario		*Entre 23.000 y 24.000€ Brutos/anuales*	*27.000 € - 33.000 € Bruto/año*
Categoría o nivel	x	*Técnico*	*Empleado*
Jornada laboral		*Jornada Completa*	*Jornada completa*
Descripción		*Proceso de selección continuo...*	*En Tecnoexample estamos...*
Modalidad de presencialidad		*Teletrabajo parcial*	*(Híbrido)*
Clasificación	x	*atencion al cliente / agente seguros*	*Java / Spring*
Puesto	x	*Técnico de Mantenimiento / Electricista*	*Analista Programador*

Cuadro 3.1: Atributos comunes casos de uso

Como se puede observar en los ejemplos mostrados de la tabla anterior, los valores de varios atributos no se encuentran normalizados. El propósito del parseo dentro de esta fase es la extracción de los valores de los valores de los campos en bruto (ver figura Flujo de información entre los componentes del sistema en página 18). Será en fases posteriores a la extracción de datos, pertenecientes al

Subsistema de planificación y ejecución de flujos de trabajo, donde se apliquen las transformaciones requeridas a dichos valores.

En las siguientes tablas se muestran los atributos específicos para cada portal, que no se disponen en el otro portal:

Atributos	Multiv.	Ejemplos
Área funcional		*Ingeniería y producción*
Sub-área funcional		*Instalación y mantenimiento*
Funciones		*- Su misión principal es ...*
Requisitos		*- Trato correcto y agradable...*
Se ofrece		*- Coche de empresa...*

Cuadro 3.2: Atributos específicos *Infoempleo*

Atributos	Multiv.	Ejemplos
Titulación académica		*FP2/Grado Superior*
Idioma	x	*Inglés (Medio)*

Cuadro 3.3: Atributos específicos *Tecnoempleo*

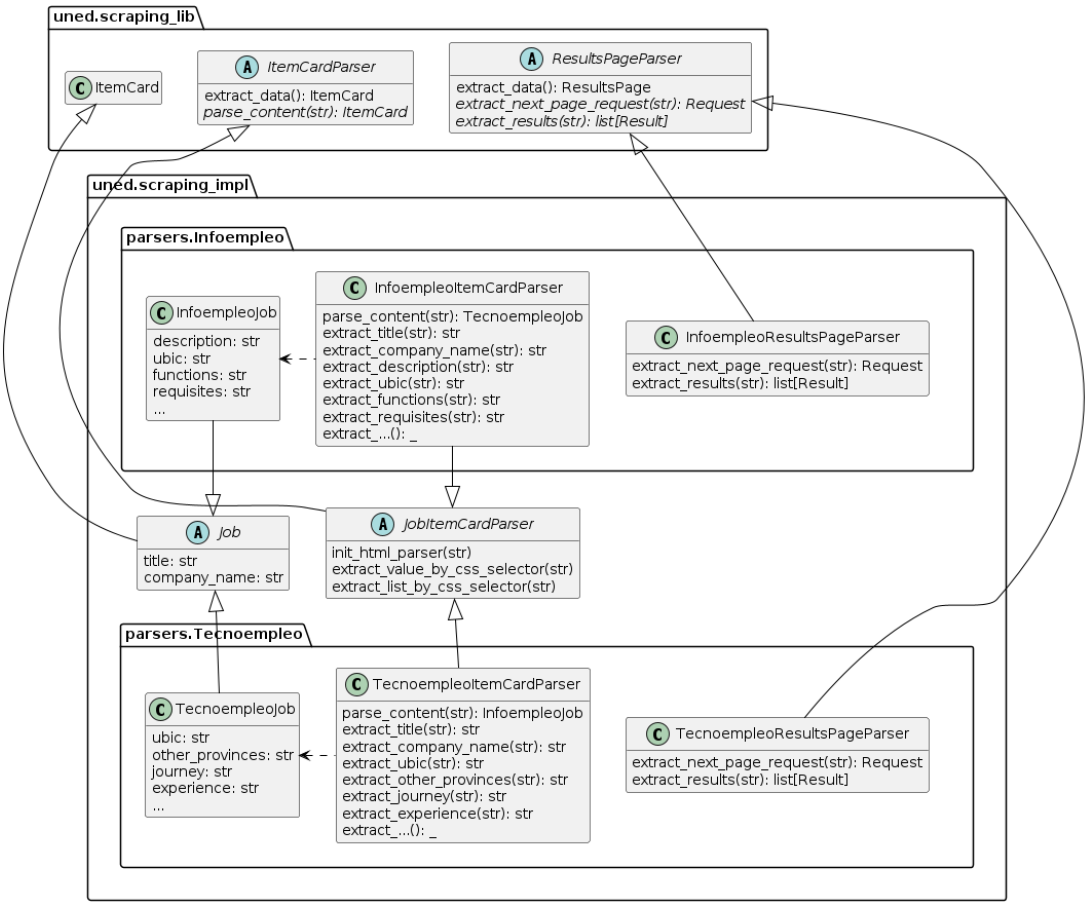


Figura 3.6: Diagrama de clases general de implementación del extractores de datos de webs para los casos de uso

En la figura 3.6 (página 27) se muestra un diagrama de clases general con las implementaciones específicas realizadas para los casos de uso, en el que se reflejan las implementaciones de las clases contenedoras de los valores tras el parseo de los atributos referidos de las ofertas de empleo (clases ***Tecnoempleo*** e ***Infoempleojob***) y el resto de implementaciones de parseadores.

Ambos portales web de búsqueda de empleo hacen uso de la renderización de los ficheros *HTML* finales en servidor, por lo que el *HTML* procesado por los parseadores es el definitivo, no requiriéndose transformaciones de éstos en cliente en tiempo de ejecución. Es por esto que se ha recurrido a la librería *Beautiful Soup* (ver sección 2.2. Extracción de datos de portales web en la página 6) para el parseo de las páginas web de los dos portales

Para la extracción de contenidos de las fichas de detalle (paso 4 del Proceso de extracción de información) se ha implementado el parseador correspondiente extrayendo las características comunes de parseo a la clase ***JobItemCardParser***, y extendiendo dicha clase para el parseo de los atributos específicos de los detalles de cada portal en ***TecnoempleoItemCardParser*** e ***InfoempleoItemCardParser***. De manera análoga para los pasos 2 y 5 del Proceso de extracción de información, se han definido las clases ***TecnoempleoResultsPageParser*** e ***InfoempleoResultsPageParser*** como extractoras de datos de las páginas de resultados.

Definición de tareas de extracción

Si bien la cuestión de la definición de las tareas de extracción corresponde más a la invocación del motor de extracción desde el proceso *ETL* del Subsistema de planificación y ejecución de flujos de trabajo, se ha creído conveniente incluirlo en esta subsección para ilustrar sus características.

El motor permite la definición de varias tareas de extracción de datos por parseador. Una tarea de extracción de datos se compone de tres unidades de información:

1. Nombre del parseador (atributo ***parser***), que relaciona la tarea con los parseadores.
2. Datos del punto de entrada para la extracción de datos, necesario para la ejecución del paso 1 del Proceso de extracción de información (atributo ***entrypoint_request***).
3. Tiempo de latencia, en segundos, entre ejecuciones de peticiones de páginas de resultados de búsqueda y de detalle (atributo ***delay_between_page_loads***).

Para los casos de uso de los portales que nos ocupan se necesitan únicamente dos tareas de ejecución, de nombres ***infoempleo_24hrs*** y ***tecnoempleo_24hrs***, pues sólo se va a requerir un punto de entrada por cada portal al no existir la necesidad de aplicar distintos criterios de filtrado para un mismo parseador. Como criterio de filtrado sólo aplica que se obtengan las ofertas de empleo publicadas en las últimas 24 horas para ambos casos. Por otra parte, el tiempo de latencia entre cargas de página se ha establecido en un segundo para evitar sobrecargar de peticiones los portales web.

A continuación se incluye el contenido del fichero *YAML* diseñado para la configuración de dichas tareas conforme a las características mencionadas (figura 3.7 en página 29).

```yaml
__anchors__:
  entrypoint_request:
    &common_http_headers
    headers:
      Access-Control-Allow-Origin: '*'
      Access-Control-Allow-Headers: 'Content-Type'
      Access-Control-Max-Age: '3600'
      User-Agent: 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0'


infoempleo_24hrs:
  parser: "Infoempleo"
  entrypoint_request:
    request_type: "Http"
    params:
      url: "https://www.infoempleo.com/trabajo/"
      request_method: "POST"
      payload:
        job_type: "empleo"
        search: ""
        region: ""
        diasPublicacion: "1"
        retribucionMin: "0"
        experienciaMin: "0,10"
        jornadaLaboral: ""
        tipoContrato: ""
        categoriaPuesto: ""
        pagina: "1"
        teletrabajo: ""
        ordenacion: "fechaAlta"
      <<: *common_http_headers
  delay_between_page_loads: 1  # SECONDS

tecnoempleo_24hrs:
  parser: "Tecnoempleo"
  entrypoint_request:
    request_type: "Http"
    params:
      url: "https://www.tecnoempleo.com/ofertas-trabajo/?ult_24h=1"
      request_method: "GET"
      <<: *common_http_headers
  delay_between_page_loads: 1  # SECONDS
```

Figura 3.7: Contenido del fichero de configuración de tareas de extracción de información

En el fichero se incluyen tres elementos de primer nivel:

- Un primer elemento para la definición de anclas para su reutilización en el resto del documento, donde se han especificado las características de las cabeceras *HTTP* que son comunes a los puntos de entrada de las dos tareas de extracción.

- Dos últimos elementos, en los que se definen las dos tareas de extracción de datos, donde cabe ilustrar la lógica seguida por el motor para la carga dinámica de componentes:

    1. El atributo *parser* es utilizado para localizar el parseador de la tarea (por ejemplo, para el valor ***Infoempleo*** se cargan las clases parseadoras ***Infoempleo****ResultsPageParser* e ***Infoempleo****ItemCardParser* del módulo ***Infoempleo*** del paquete *uned.scraping_impl.parsers*).

    2. El atributo *request_type* de los *entrypoint_request* (para el valor ***Http*** se carga la clase de nombre ***Http****Request* del módulo *requests* del paquete *uned.scraping_impl*)

Pruebas automatizadas unitarias y de integración

El desarrollo del núcleo de la biblioteca del motor de extracción de datos y de sus extensiones para la implementación de los casos de uso se ha acometido en ciclos iterativos basados en tests automatizados unitarios y de integración. Estos tests se han concebido para ser utilizados y extendidos para ser utilizados como pruebas de regresión en usos futuros de la biblioteca.

Se han desarrollado un total de 68 tests que cubren la mayor parte de los caminos críticos de ejecución de la librería para los casos de uso. Para su implementación se ha recurrido al framework de testing nativo *Python unittest*.

Estructura de ficheros

Para concluir con la descripción técnica de este subsistema, en el cuadro siguiente se muestra la estructura de los ficheros de sus componentes, junto con una descripción de los más relevantes. Como se puede apreciar, para facilitar su mantenimiento, la estructura de ficheros de los paquetes y módulos del núcleo de la biblioteca ha sido replicada para los ficheros de sus tests correspondientes:

Estructura de ficheros fuente	Estructura de ficheros de test	Descripción
src	**tests**	
	resources	**Paquete de recursos de tests**
	Infoempleo	
	DetailedItemPageOK.html	
	(...)	
	scraping_executions.yaml	Fichero de configuración de tareas
	Tecnoempleo	
	DetailedItemPageOK.html	
	(...)	
uned	**uned_tests**	
	ResourcesFilesLoader.py	
scraping_impl	**scraping_impl**	**Paquete de implementaciones**
observers.py		Módulo de observadores
parsers	**parsers**	**Paquete de parseadores**
	HttpRequestTestProxy.py	
Infoempleo.py	test_Infoempleo.py	*Parseadores de Infoempleo*
JobItemCardParser.py		
Job.py		
Tecnoempleo.py	test_Tecnoempleo.py	*Parseadores de Tecnoempleo*
requests.py	test_request.py	Módulo de requests
scraping_lib	**scraping_lib**	**Paquete del núcleo**
ItemCardParser.py	test_ItemCardParser.py	Módulos del núcleo
ItemCard.py		
Parser.py		
ParsersFactory.py	test_ParsersFactory.py	
ParsingResultObserver.py		
ParsingResult.py		
Request.py		
RequestsFactory.py		
Result.py		
ResultsPageParser.py	test_ResultsPageParser.py	
ResultsPage.py		
ResultsPagesTraveller.py	test_ResultsPagesTraveller.py	
TravelTaskConfigLoader.py	test_TravelTaskConfigLoader.py	
utils.py	test_utils.py	

Cuadro 3.4: Estructura de ficheros *uned.scraping_impl* y *uned.scraping_lib*

3.3. Subsistema de presentación y explotación de información

El propósito de este subsistema es atender a las demandas del área funcional Presentación y explotación de información, derivada del objetivo 3 de este trabajo:

> **OBJ-03** Proporcionar interfaces gráficas dinámicas y herramientas de diseño on-line de las mismas, para la explotación en tiempo real de la información obtenida, con el propósito de la identificación de posibles áreas formativas de interés en función de su demanda en el mercado laboral.

Este subsistema ejerce como medio de interactuación con el sistema por parte del área usuaria final, encargados de la toma de decisiones acerca de los cursos de formación, y de los científicos de datos cuyo cometido es la extracción de conocimiento y definición de modelos para este cometido.

3.3.1. Análisis

Para la consecución de las metas indicadas en el objetivo de este sistema se ha detectado la necesidad disponer de las siguientes herramientas:

- **Cuadro de mando**: Necesario para la identificación de áreas funcionales de interés en el ámbito de las ofertas de trabajo por parte del área usuaria final, con la presentación de información a través de gráficos que permitan realizar operaciones de selección, clasificación y filtrado. Este elemento ha de ser diseñado a medida atendiendo a las características de la información de los portales de empleo.

- **Herramienta gráfica para el análisis de datos de propósito general**: Cuyo objetivo es el acceso a la información de los juegos de datos para el estudio más pormenorizado de la información del que ofrece un cuadro de mando. Esta herramienta debe permitir la obtención de informes dinámicos de dichos datos y la realización de operaciones de combinación y agregación de sus atributos. Esta herramienta será utilizada por usuarios avanzados, como los científicos de datos o ingenieros de datos, para la detección de características de la información que pueden traducirse en posteriores modificaciones para mejoras del proceso *ETL*, del cuadro de mando, o en los modelos de explotación de la información.

- **Cuadernos interactivos**: Se utilizarán por los usuarios avanzados, con los mismos objetivos que la herramienta anterior, como herramienta interactiva para el estudio de los datos de los portales de empleo mediante un lenguaje de programación, permitiendo el uso de los elementos software utilizados por los procesos *ETL*.

Para facilitar el trabajo de los actores implicados desde cualquier plataforma tecnológica, todas herramientas deben ser accesibles remotamente mediante un cliente ligero estándar, como puede ser un navegador web.

Además, en el caso de acceso al cuadro de mando por parte de los usuarios finales, se ha de contar con una plataforma que permita la modificación, exportación e importación sencilla del cuadro de mando, y de mecanismos para la autorización y autenticación de sus usuarios.

3.3.2. Diseño

De las opciones tecnológicas tratadas en el estudio del estado del estado del arte, en su sección 2.3. Presentación y explotación de información (página 8), se ha decidido basar la solución para las dos primeras herramientas presentadas en la subsección anterior en la pila tecnológica de *Elasticsearch*. Esta solución presenta la ventaja de disponer de una suite que, además de responder a todas las necesidades planteadas en relación con las herramientas gráficas de explotación de datos, aporta un potente motor de índices invertidos que responde a las necesidades de persistencia de los datos de las ofertas de empleo (ver sección 3.1. Introducción en página 17). Se ha valorado además que *Elasticsearch* dispone de un alto nivel de escalabilidad, permitiendo el despliegue de su motor de índices en clusters y la división de sus índices en *shards* distribuidos, y una extensa *API REST* para interactuar con todos los servicios de la pila.

Por otra parte, para atender a la demanda de una herramienta de explotación de información mediante cuadernos interactivos, se recurre al producto de *JupyterLab*, ampliamente utilizado en la industria de ingeniería y ciencia de datos, y con soporte a más de 40 lenguajes de programación.

En la siguiente figura se representan los servicios implicados en la solución y las relaciones entre sí:

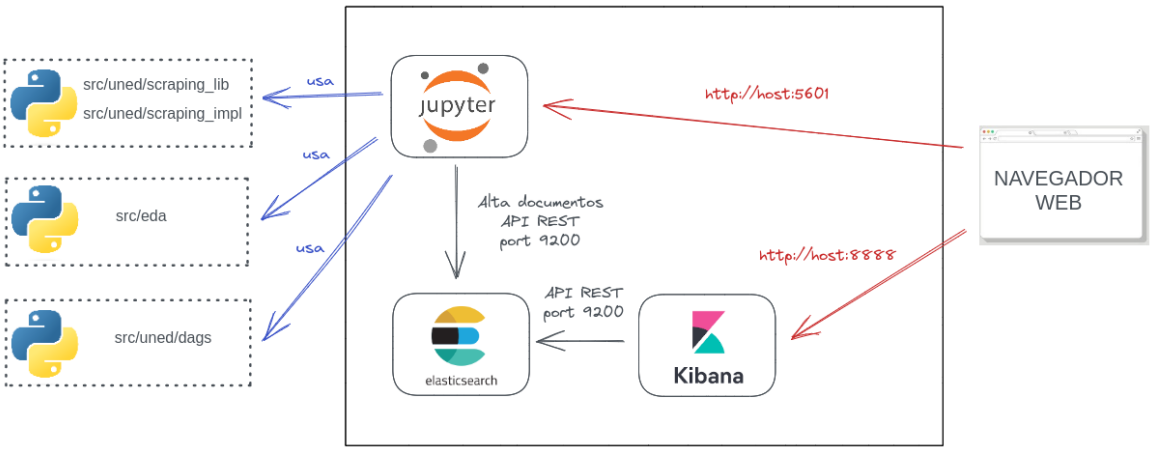


Figura 3.8: Diagrama de servicios de presentación y explotación de información

Como se puede deducir de la figura anterior, las características de esta solución permiten a usuarios avanzados el acceso por medio de *notebooks* a todas las funcionalidades del proceso ETL,

desarrolladas en *Python*. Aportando la solución, además de los *notebooks* utilizados para el estudio de datos de los casos de uso realizado para la implementación de este trabajo, los artefactos necesarios para lanzar el proceso completo de ETL para la puesta a disposición sus datos transformados en la herramienta de explotación de información, sin necesidad de recurrir al Subsistema de planificación y ejecución de flujos de trabajo.

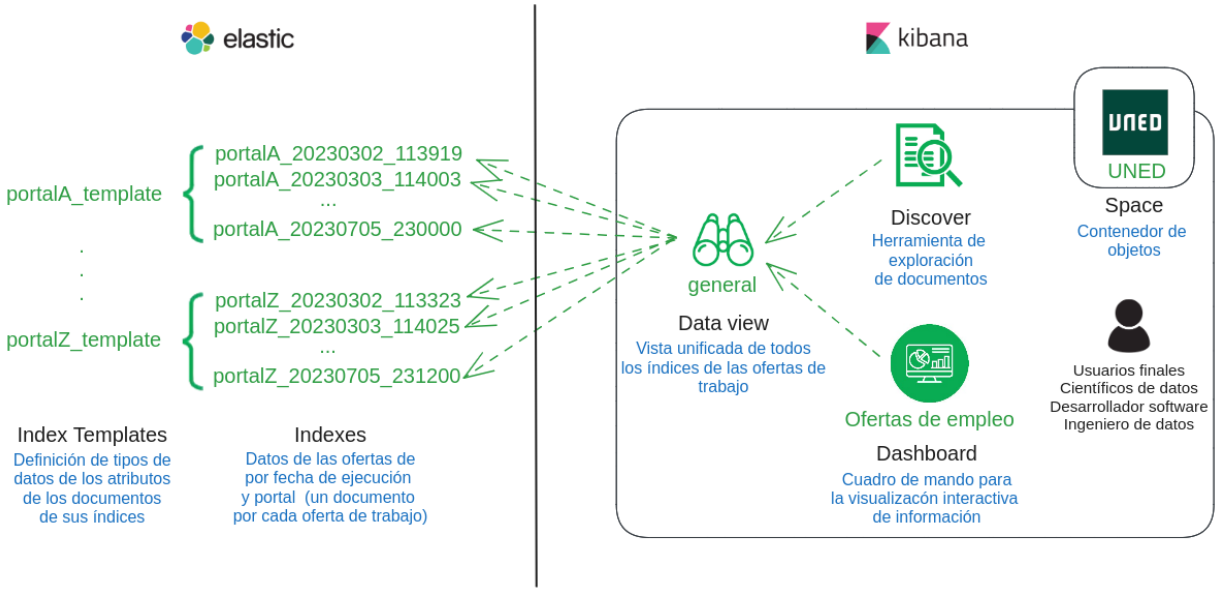


Figura 3.9: Diseño de solución de presentación y explotación de información: objetos elk

En cuanto al diseño de la solución en la pila *Elasticsearch*, representado en la figura 3.9, se ha optado por los siguientes criterios:

1. **Uso de *Index Templates***. Las plantillas de índices de Elasticsearch permiten definir plantillas para la especificación de las características de los índices cargados en la plataforma cuyos nombres cumplan con los patrones con especificados en dichas plantillas. Definen las características de mapeo de los campos de los índices y sus características de replicación y *shards*. Facilitan de este modo la aplicación unificada de dichos criterios a conjuntos de índices, evitando tener que hacerlo cada vez que se incluya un nuevo índice o recurrir al sistema de inferencia de tipos automático de *Elasticsearch*. Para el caso que nos ocupa se ha decidido crear una plantilla por portal de empleo, pues cada portal de empleo tiene un modelo de datos diferente.

2. **Índices por producto y dimensión temporal**: Como criterio de agrupación de documentos de ofertas de empleo, con el objetivo de facilitar el mantenimiento de la información y dada la flexibilidad que permite *Elasticsearch* para este tipo de decisiones por su orientación inicial a la explotación de la información de ficheros logs, se ha decidido crear un índice por cada portal e instante de subida de dicha información, creando índices cuyos nombres siguen el formato <nombre_portal>_yyyyMMdd_HHmmss.

3. ***Data view* de confluencia**: Las vistas de datos de *Kibana* son el elemento software para la obtención de información de los índices para su explotación por parte de las herramientas visuales. En este sentido, con el objetivo de unificar toda la información de los portales de empleo en una sola vista, se ha decidido crear una sola vista de datos general que permita disponer una visión completa de las ofertas de empleo desde las distintas herramientas de visualización. Esta vista dispondrá además de la opción de filtrado de resultados por el portal de origen.

4. **Cuadro de mando unificado**: Para obtener una visión integral del estado de las ofertas de trabajo se ha decidido diseñar un único cuadro de mando que aglutina la información de todas las ofertas, permitiendo además la identificación y filtrado por portal de origen, y los campos comunes y específicos de dichos portales.

5. ***Espacio* específico**: Como se comentó en la sección 2.3. Presentación y explotación de información (página 8) *Kibana* proporciona una interfaz con múltiples opciones para la gestión integral de la pila *ELK*. Es por este motivo que se ha recurrido a la creación de un único espacio para los portales de empleo para que, en combinación con la incorporación de perfiles de usuario específicos, se limite el acceso de los usuarios a los objetos de explotación de datos de los portales de empleo, restringiendo la utilización del resto de funcionalidades a los administradores de la plataforma.

3.3.3. Aplicación a los casos de uso

Definición de variables de visualización

El desarrollo de los objetos que componen la solución de visualización de este subsistema tiene relación directa con las características de los dos casos de uso descritos en 3.1. Introducción. Para ello, en primer lugar se ha hecho un estudio estadístico datos de los portales de origen, que ha concluido en la descripción de variables de visualización de la tabla 3.5. Variables de visualización (página 35).

En dicha tabla se observan tres tipos de orígenes de variable, indicando con leyenda *"C"* -para las variables comunes a los dos orígenes de datos-, *"I"* -para las variables exclusivas de *Infoempleo*-, y *"T"* -para las exclusivas de *Tecnoempleo*-. Conforme a lo reflejado en el diseño de la solución visto en la subsección anterior, este modelo de datos ha dado origen a dos plantillas de índices, una por portal de empleo: *infoempleo_template* y *tecnoempleo_template*, que comparten definición para sus atributos en el caso de la variables comunes, añadiendo las específicas para cada portal en cada caso.

Variables	Origen	Tipo	Multiv.	Val. nulos	Compuesta	Ejemplo
Origen	C	Cualitativa-Nominal				*Tecnoempleo*
Fecha de publicación	C	Cuantitativa-Discreta				*01/06/2023*
id	C	Cualitativa-Nominal				*rf-1973o674ck0c822951ja*
Título	C	Cualitativa-Nominal				*Analista Funcional Java*
Empresa	C	Cualitativa-Nominal		×		*Tecnoexample S.L.*
Localización	C	Cualitativa-Nominal		×		*Málaga*
Experiencia mínima requerida	C	Cuantitativa-Discreta			×	{ *"gte": 3, "lte": 5* }
Vacantes	C	Cuantitativa-Discreta				*5*
Tipo de contrato	C	Cualitativa-Nominal		×		*Indefinido*
Salario medio	C	Cualitativa-Contínua		×	×	{ *"gte": 27000, "lte": 33000, "mean": 30000* }
Categoría o nivel	C	Cualitativa-Nominal	×	×		*Técnico / Especialista*
Jornada laboral	C	Cualitativa-Nominal		×		*Jornada Completa*
Modalidad de presencialidad	C	Cualitativa-Nominal		×		*Híbrido*
Clasificación	C	Cualitativa-Nominal	×	×		*[JAVA, SPRING]*
Puesto	C	Cualitativa-Nominal	×	×		*Analista Programador*
Área funcional	C	Cualitativa-Nominal				*Tecnología e informática*
Sub-área funcional	I	Cualitativa-Nominal		×		*Programación*
Titulación académica	T	Cualitativa-Ordinal		×		*F2 - FP2/G. Superior*
Idioma	T	Cualitativa-Nominal	×	×		*Inglés (Medio)*

Cuadro 3.5: Variables de visualización

Como se puede observar en los ejemplos aportados en la definición de las variables de la tabla

3.5, en comparación con el modelo de datos original de los portales de información presentado en la subsección 3.2.3. Aplicación a los casos de uso del Subsistema de extracción de datos, se muestra un modelo de datos con información normalizada, que deberá ser procesada desde el Subsistema de planificación y ejecución de flujos de trabajo para cumplir con las características requeridas por este Subsistema.

Para cumplir con los requerimientos de filtrado por portal de origen se incluye además el campo origen de datos, que tomará para los casos de uso los valores "*Infoempleo*" y "*Tecnoempleo*".

Cuadro de mando y herramienta gráfica de explotación de datos de propósito general

El cuadro de mando diseñado como parte del trabajo es una propuesta que representa la distribución de la información de las distintas variables centrada principalmente en el número de vacantes de empleo ofrecidas por las ofertas de trabajo. No obstante, como se ha indicado anteriormente, una de las causas de elección de esta pila tecnológica ha sido que permite que de forma sencilla, mediante una interfaz web, se incorporen al cuadro de mandos nuevos elementos gráficos al cuadro atendiendo a otros criterios o se modifiquen los ya existentes.

A continuación se muestra el diseño general del cuadro de mando, con las tres secciones que lo componen resaltadas:

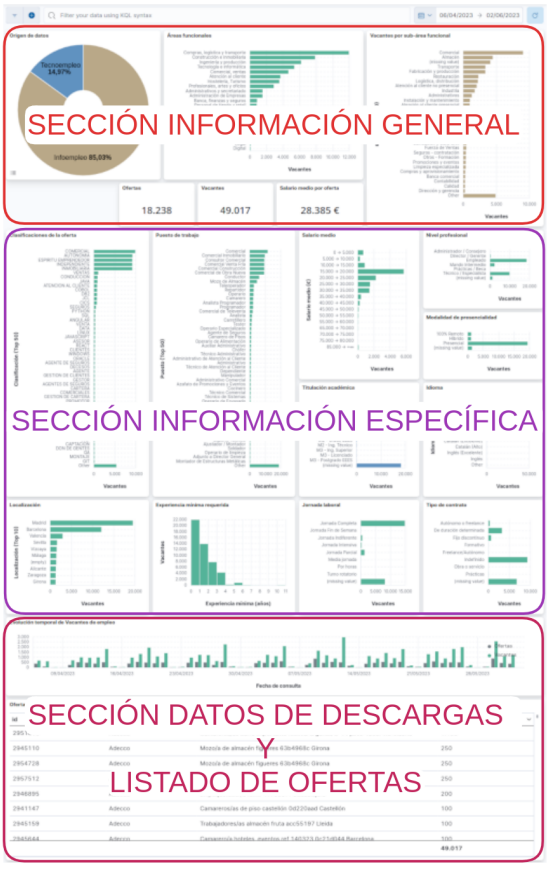


Figura 3.10: Visión general del cuadro de mando

Como se ha indicado en la relación de variables de visualización, existen tres variables exclusivas de alguno de los portales de empleo. Estas variables han sido incluidas en algunas de las visualizaciones del cuadro de mando, pero para no inducir a errores de interpretación se ha seguido el siguiente código de colores según la procedencia de las variables representadas:

Variables comunes
Variables exclusivas de Infoempleo
Variables exclusivas de Tecnoempleo

Cuadro 3.6: Códigos de colores de gráficos del cuadro de mando

Todos los gráficos del cuadro de mando son interactivos, permitiendo pulsar sus valores para la incorporación de filtros, provocando que dinámicamente se recalculen y rendericen el resto de gráficos teniendo en cuenta los filtros aplicados.

Para facilitar el trabajo de interpretación de los gráficos por parte del usuario en el caso de las variables con un número considerable de valores, se ha restringido la visualización de elementos a un número de elementos adecuado, determinado en función de la distribución observada en el juego de datos obtenido para los casos de uso. Esta circunstancia se informa en los gráficos en los que ha sido aplicada con la leyenda *"(Top <n>)"*, siendo <n> el número de elementos mostrados.

En el caso de requerirse un estudio más detallado de los datos se ha puesto a disposición del espacio de trabajo la herramienta de propósito general *Discover*, accesible desde el propio cuadro de mando y que no tiene las limitaciones propias en el acceso a los datos propias de un cuadro de mando, permitiendo la combinación, visualización y exportación de datos de variables de forma personalizada. A continuación se muestra una captura de pantalla de la herramienta:

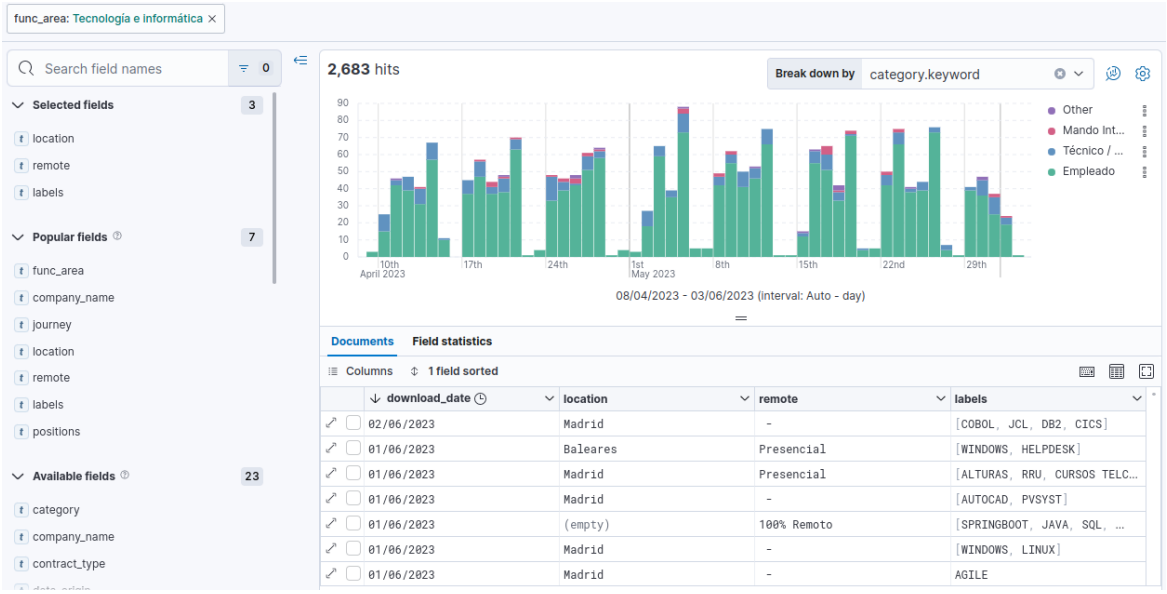


Figura 3.11: Visión de la herramienta gráfica de explotación de datos de propósito general: *Discover*

Para concluir esta sección, a continuación se presenta con mayor detalle el contenido de las secciones que componen el cuadro de mando:

Sección de información general

El objetivo de esta sección es dar una visión general, a través de las clasificaciones principales de las ofertas de empleo, como son el portal de origen y el área a la que pertenecen, para la aplicación de criterios de filtrado generales para la identificación de áreas de interés. Además incluye la visualización de indicadores generales de agregación, como es el caso del número total de *ofertas* y *vacantes* de empleo, y el *salario medio por oferta* de trabajo .

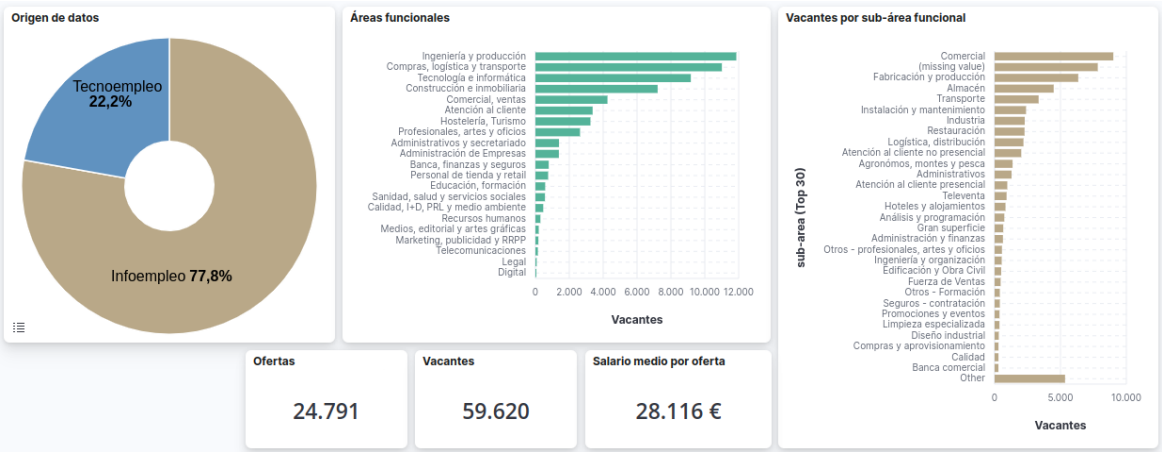


Figura 3.12: Cuadro de mando: sección de información general

Pulsando una de las dos opciones del gráfico circular, relativo al *origen de datos*, se posibilita mostrar en el cuadro de mando los resultados para uno los portales de empleo estudiados.

En la figura de esta sección se puede apreciar el criterio indicado anteriormente para la distinción los gráficos con variables exclusivas de un portal específico, como es el caso de la *sub-área funcional*, variable específica del portal *Infoempleo*.

Sección de información específica

En esta sección se representan el resto de variables de interés de los portales para la identificación de áreas de interés, desde una perspectiva de la distribución del número de ofertas vacantes de empleo para los criterios de búsqueda aplicados al cuadro de mando (ver figura 3.13 en página 39).

Esta sección incluye dos variables exclusivas del portal *Infoempleo* que se han considerado de interés para los objetivos de la visualización: *Titulación académica* e *Idioma*.

Figura 3.13: Cuadro de mando: sección de información específica

Sección de datos de descargas y listado de ofertas

El primer gráfico de esta última sección se ha incluido para facilitar la interpretación y filtrado de resultados a partir de su evolución temporal, mostrando recuentos diarios de vacantes y ofertas de empleo en función de sus fechas de publicación.

Para concluir se ha incluido una tabla que permite al usuario visualizar valores identificativos de referencia de las ofertas de empleo que cumplen los criterios de selección aplicados al cuadro de mando.

Figura 3.14: Cuadro de mando: sección de datos de descargas y listado de ofertas

3.4. Subsistema de planificación y ejecución de flujos de trabajo

El objetivo de este subsistema es atender a las cuestiones planteadas para el área funcional Planificación y ejecución de flujos de trabajo, que se derivan del objetivo 4 de este trabajo:

OBJ-04 Implantación de un motor de flujos de trabajo que permita la ejecución independiente, fiable, periódica, automatizada y controlada, de los procesos implicados en:

- Las tareas de extracción de los datos obtenidos de los portales de empleo.
- El posterior tratamiento de los datos obtenidos para su puesta a disposición de las áreas usuarias implicadas.
- La notificación y/o persistencia de los productos resultantes de los procesos anteriores.

3.4.1. Análisis

Como se ha visto anteriormente, la característica de la temporalidad de las publicaciones de la ofertas de empleo de los portales de búsqueda de empleo requiere disponer de este subsistema de orquestación, que ha de cumplir con los siguientes cometidos funcionales:

1. Extracción de los datos de los portales de origen, mediante el Subsistema de extracción de datos.
2. Transformación de los datos extraídos para cumplir los requerimientos de información del Subsistema de presentación y explotación de información.
3. Carga de la información transformada en el Subsistema de presentación y explotación de información.
4. Ejecución de los pasos anteriores en un ciclo continuo, con una frecuencia de ejecución predefinida y adecuada a las características de cada portal, que permita el acceso inmediato a la información procesada por parte del área usuaria.
5. Puesta a disposición de los usuarios técnicos de mecanismos que permitan la notificación de resultados obtenidos de los flujos anteriores y su persistencia.

3.4.2. Diseño

Para dar respuesta a las necesidades expresadas, una vez evaluadas las opciones presentadas en la sección 2.4. Planificación y ejecución de flujos de trabajo (página 10), se ha optado por las siguientes

opciones tecnológicas:

- Recurrir a una solución basada en *Apache Airflow* como planificador de flujos de trabajo, debido a la ventaja que supone su definición de flujos en *Python* para la integración directa en éstos del motor de extracción de datos de portales de empleo, y también a que la solución de infraestructura se va a limitar a la utilización de *Docker* (ver sección 3.5. Infraestructura del sistema en página 49), lo que descarta la solución de *Argo workflows*.
- En cuanto a la utilización de un componente específico de diseño de flujos de transformación de datos, como *Apache NIFI*, y una vez estudiadas las características de las de las transformaciones de información de los casos de uso, no se ha creído conveniente su utilización, pues la naturaleza de las transformaciones a aplicar en los datos de origen no justifica la complejidad adicional que le añadiría al sistema la incorporación este elemento software.

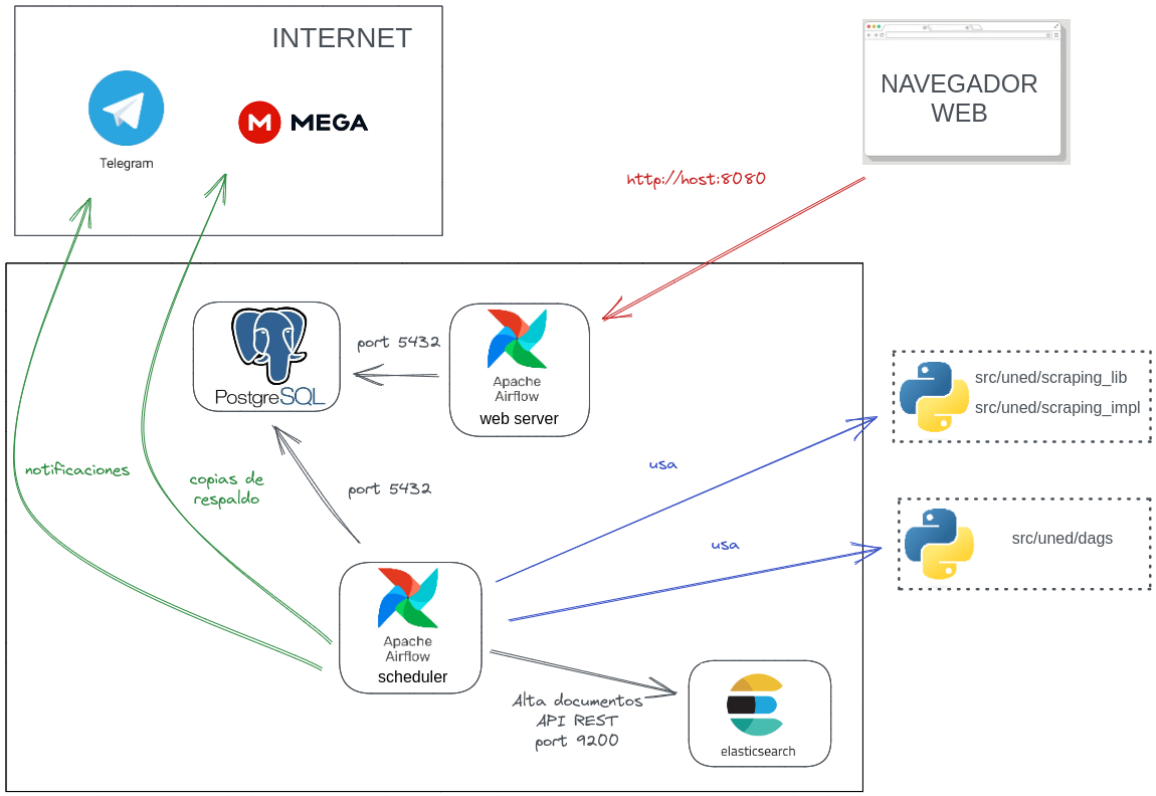


Figura 3.15: Diagrama de servicios del subsistema de planificación y ejecución de flujos de trabajo

En la figura anterior se refleja la arquitectura de servicios de la solución, en la que se muestran los dos servicios de internet utilizados para la notificación y persistencia de datos, y en la que se presentan los tres servicios que componen este subsistema:

- El servicio de persistencia de los datos de ejecución y configuración del planificador, para el que se ha optado por *PostgreSQL* como sistema gestor de base de datos relacional.

- El planificador, encargado de la ejecución de los flujos de trabajo que para su cometido cuenta con acceso al motor de extracción de datos de portales web, los ficheros de definición de flujos de trabajo y la interfaz *REST* de *Elasticsearch* para la puesta a disposición del alta de la información obtenida en el Subsistema de presentación y explotación de información.

- El servidor web, encargado de facilitar una interfaz web para el configuración y seguimiento de la ejecución de los flujos de trabajo.

Como se ha indicado anteriormente, para la definición de flujos de trabajo, *Airflow* utiliza especificaciones en *Python* que son recuperadas de los ficheros leídos por el planificador desde un directorio indicado en su configuración. *Airflow* explota la idea de los grafos acíclicos dirigidos para la definición de flujos, a los que denomina *DAG*, y que definen las tareas que los componen, sus relaciones y dependencias. Para ilustrar lo expresado con aplicación a este trabajo, a continuación se incluye la definición de un *DAG* de ejemplo correspondiente al flujo de extracción de datos de este trabajo, mediante el uso de decoradores de *Airflow*:

Listado de código 3.1: Definición de DAG de extracción de datos

```
DAG_ID = "portal_X_extract"
TASK_NAME = "portalX_ultimas_24_horas"

@dag(
    dag_id=DAG_ID,
    tags=['extract', TASK_NAME],
    start_date=datetime(2023, 4, 5, tz='Europe/Madrid'),
    schedule="0 8,15,22 * * *",
    default_args={"retries": 1, "retry_delay": duration(hours=1)},
 )
def dynamic_generated_dag():
    @task
    def execute_scraping_tasks(scraping_task_name: str):
        return execute_scraping_traveller(scraping_task_name)

    @task(outlets="portalX_dataset")
    def save_summary_task(summ, task_n):
        save_summary(summ, task_n)

    summary = execute_scraping_tasks(TASK_NAME)
    save_summary_task(summary, TASK_NAME)

dynamic_generated_dag()
```

El *DAG* del bloque de código anterior, que por la configuración indicada en la parametrización del decorador correspondiente al *DAG(@dag)* se denominaría *"portal_X_extract"*, se compone de dos tareas consecutivas, que se denominan *"execute_scraping_tasks"* y *"save_summary_task"*. Su repre-

sentación gráfica se muestra en el bloque "DAG EXTRACCIÓN" de la figura 3.16 (página 43).

En el contexto del planificador de *Airflow*, se distinguen dos tipos de eventos que provocan la ejecución de un los *DAGs* aplicables a esta solución:

1. **El cumplimiento de una planificación temporal**, como en el caso del ejemplo *DAG* anterior (ver listado de código 3.1) se produce diariamente a las 8h, 15h y 22h (determinado por la expresión de cron *"0 8,15,22 * * *"*).

2. **La actualización de un dataset**, evento que en el ejemplo mencionado se puede apreciar que se define en el decorador de la tarea *save_summary_task* (atributo *outlet* de su decorador *@task*), provocando que cada vez que esta tarea se ejecute de forma exitosa, se informe al sistema de la actuación del dataset correspondiente (en este caso *"portalX_dataset"*) y se ejecuten los *DAG* que dependan de esta actualización.

Habiendo ya introducido los artefactos necesarios para la definición y ejecución de flujos de trabajo, en el siguiente diagrama se representan los *DAG* construidos para la solución a este trabajo:

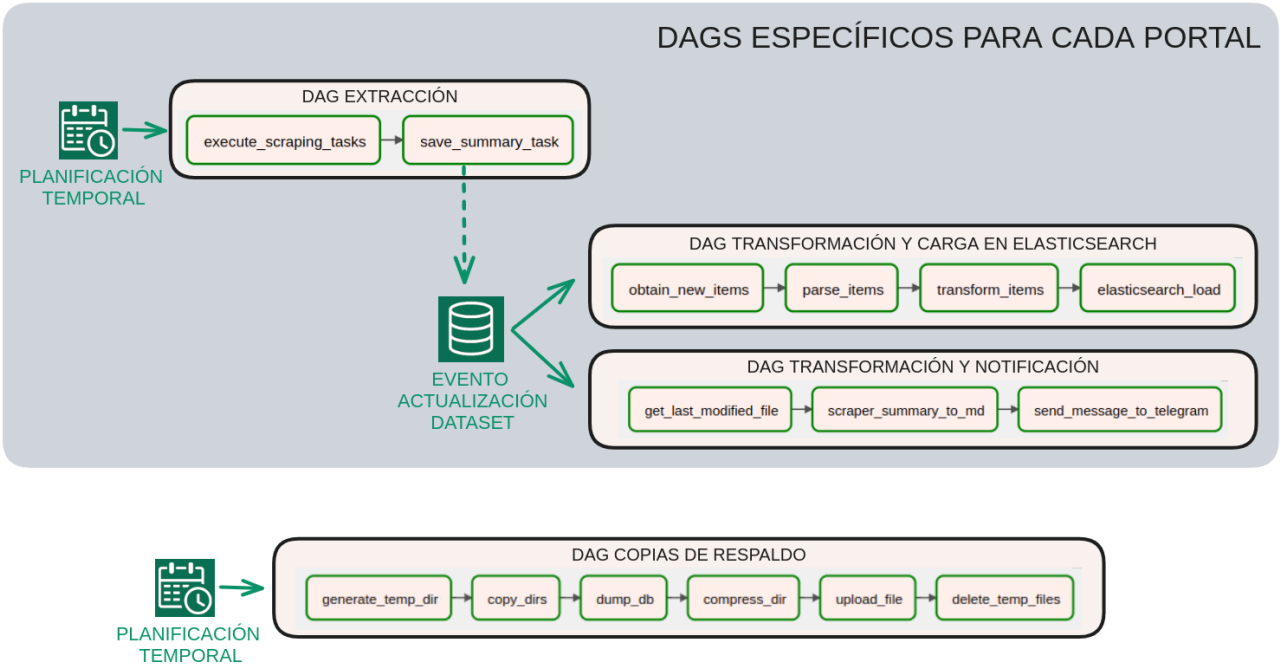


Figura 3.16: Diagrama de infraestructura subsistema de planificación y ejecución de flujos de trabajo

El primer grupo de *DAGs* del diagrama anterior, son los específicos para cada portal de empleo, para los que se han implementado plantillas *Python* que dan de alta sus *DAGs* correspondientes. Lo componen los siguientes flujos de trabajo:

- **DAG de extracción**: Ejecutado mediante planificación temporal, se encarga de la ejecución del motor de parseo para extracción de los datos de un portal y el guardado temporal de sus

resultados. Guarda un registro de los elementos procesados para evitar descargas innecesarias de fichas de detalle previamente procesadas, haciendo uso de las precondiciones de parseo del motor de de extracción de datos. Una vez finalizada la tarea de extracción de datos del portal, guarda un informe de resultados de dicha tarea e informa al sistema de la actualización de su dataset. Esta notificación de actualización provoca la ejecución de los dos siguientes *DAGs*, que para facilitar su mantenibilidad y posibilitar su ejecución manual, se ha decidido desarrollar como flujos independientes:

1. **DAG de transformación y carga en *Elasticsearch***. Este *DAG* identifica los nuevos ítems de ofertas de empleo descargados, los transforma de acuerdo al formato requerido para la carga en *Elasticsearch* de la información del portal de empleo procesado, y finalmente realiza esta carga para su consumo por parte del área usuaria desde el Subsistema de presentación y explotación de información, haciendo uso del *API REST* de *Elasticsearch*.
2. **DAG de transformación y notificación**: Se encarga de notificar los resultados del proceso de extracción a partir del último informe de resultados del proceso de ejecución del *DAG* de extracción. Por su gratuidad, alta disponibilidad y simplicidad de *API*, se ha escogido *Telegram* (`https://telegram.org/`) como plataforma de mensajería para la notificación de eventos, pero *Airflow* dispone de conectores para otras muchas plataformas que pueden remplazar fácilmente a este producto.

Por último, se ha construido un último *DAG* independiente para la realización de copias de respaldo de los datos extraídos de los portales y de la información de *Airflow*. Se trata de un *DAG* de ejecución temporal planificada, cuyas tareas se encargan de hacer una exportación de la base de datos de *Airflow*, comprimir la información extraída de los portales en un fichero, y subir los ficheros resultantes a un espacio de la plataforma de almacenamiento en línea *MEGA* (`https://mega.io/`).

Todos estos procesos se han desarrollado de forma que la información de las conexiones a servicios externos sea gestionable desde la interfaz de de conexiones nativa de *Airflow*:

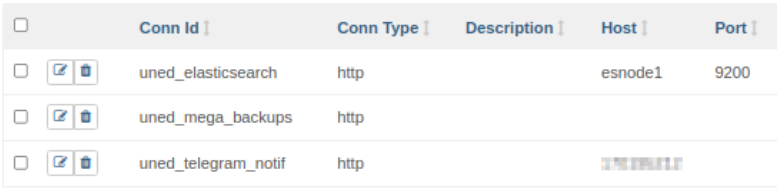


Figura 3.17: Interfaz web *Airflow*: Conexiones

3.4.3. Aplicación a los casos de uso

Para la aplicación de los procesos *ETL* desarrollados a los casos de uso descritos en la sección 3.1, se han adaptado los *DAGs* específicos de portales. A continuación se muestra la presentación

de estos flujos de trabajo en la interfaz web de *Airflow*, en la que se han resaltado los específicos de *Infoempleo* para la ilustrar el bloque de adaptación para cada caso de uso:

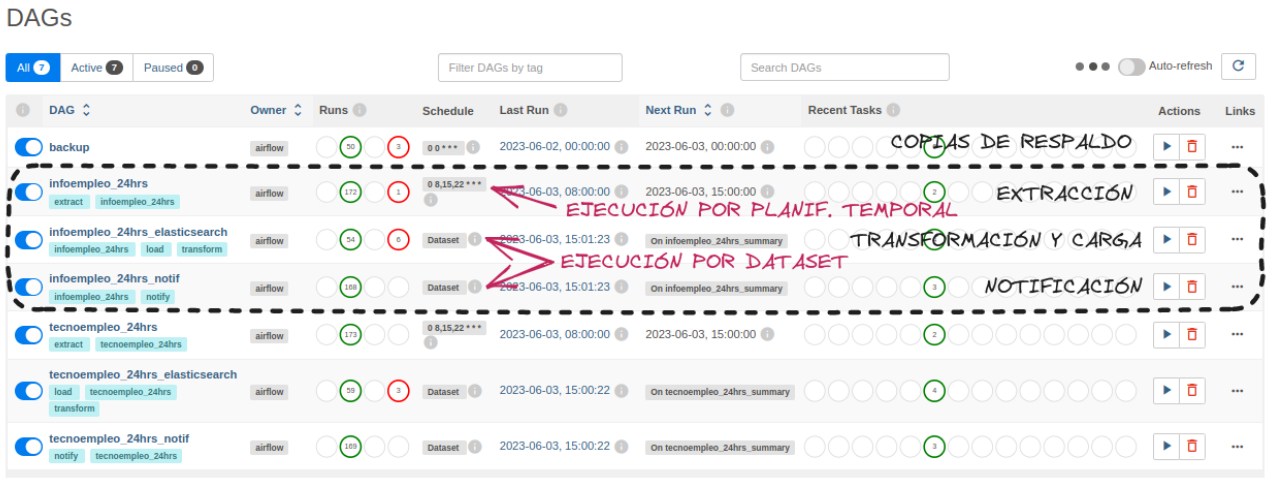


Figura 3.18: Interfaz web *Airflow*: DAGs

Para la implementación de los DAGs específicos de los portales se han hecho las siguientes adaptaciones:

DAGs de extracción:

Se han implementados los métodos correspondientes a dos *DAG*, *infoempleo_24hrs* y *tecnoempleo_24hrs*, que se corresponden con las dos tareas de extracción que recuperan las ofertas de empleo de las últimas 24 horas de los portales de los casos de uso en el momento de su ejecución. La definición de estas tareas para el motor de extracción se mostró en la sección 3.2.3. Definición de tareas de extracción (página 28). Se mantiene para estos *DAGs* la configuración vista en el apartado anterior, es decir, se ejecutan tres veces por día para la recuperación de las ofertas nuevas exclusivamente. La ejecución exitosa de estos dos *DAG* reporta al sistema la actualización de los datasets *infoempleo_24hrs_summary* y *tecnoempleo_24hrs_summary*, que disparan la ejecución de sus respectivos DAG de transformación relacionados, según se representa en siguiente captura de la interfaz web de Airflow:

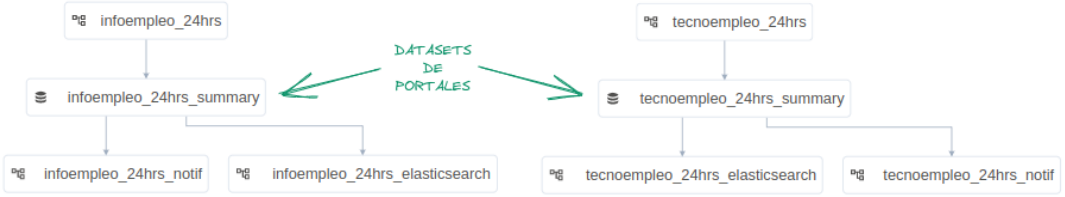


Figura 3.19: Interfaz web *Airflow*: Dependencias entre dags

***DAGs* de transformación y carga/notificación:**

Para la notificación de resultados se han generado los dos DAGs relacionados en la figura 3.19 (página 3.19), *"infoempleo_24hrs_notif"* y *"tecnoempleo_24hrs_notif"*. Estos flujos siguen una implementación común que no ha tenido que ser adaptada para cada caso de uso.

En cuanto a las tareas de transformación de datos para carga en *Elasticsearch*, incluidas en los *DAGs "infoempleo_24hrs_elasticsearch"* y *"tecnoempleo_24hrs_elasticsearch"*, se han hecho dos personalizaciones de la solución estándar presentada en el apartado anterior:

1. Para la tarea procesamiento de los ficheros de resultados extraídos, se ha hecho un desarrollo específico para mejorar el rendimiento monohilo que por defecto ofrecen las tareas *Python* en *Airflow*, optando por una ejecución multiproceso que aprovecharía las capacidades de concurrencia de la arquitectura del sistema donde se ejecute el planificador.

2. Se han implementado las tareas de procesamiento de datos de cada portal para cubrir las necesidades para la carga de información en *Elasticsearch* (subsección 3.3.3 en página 34), a partir de las características de los datos obtenidos por el motor de extracción de datos (subsección 3.2.3 en página 25). Las operaciones de procesado aplicadas para cada atributo de los portales de empleo se reflejan en la tabla 3.7 (página 47), cuya descripción es la siguiente:

   a) **Operaciones de limpieza**:

      1) Incorporación de nuevos atributos no existentes en origen, de las que cabe aclarar la introducción del atributo *Origen* con los valores *"Infoempleo"* y *"Tecnoempleo"*.
      2) Sustitución de valores nulos por valores por defecto, entre las que cabe mencionar que el atributo *"Área funcional"* inicialmente sólo existente para el portal *Infoempleo*, se incluye también para el portal de ámbito específico (*Tecnoempleo*), con el valor *"Tecnología e informática"*.

   b) **Operaciones de filtrado**, que abarcan la exclusión de los campos textuales que no se ha previsto explotar desde *Kibana*.

   c) **Operaciones de transformación**:

      1) Normalizaciones aplicadas a los atributos que requieren una terminología común para la explotación de información desde ambos portales.
      2) Extracciones de valores, que se traducen en el parseo de campos textuales de los que se parsean para la extracción de información específica relacionadas con las variables de visualización.
      3) Cálculos previos, que se aplican a los campos para no sobrecargar de cálculos adicionales al stack *ELK*.

Variables	Origen	Multiv.	Nulos	Operaciones de procesado			Ejemplos	
				Limpieza	Filtrado	Transformación	Extracción	Transformación
Origen	C			Incorp.				*Tecnoempleo*
Fecha de Publicación	C			Incorp.				*01/06/2023*
id	C						*rf-1973o674ck0c822951ja*	*rf-1973o674ck0c822951ja*
Título	C						*Analista Funcional Java*	*Analista Funcional Java*
Empresa	C						*Tecnoexample S.L.*	*Tecnoexample S.L.*
Localización	C		x			Extracción val. Normalización	*Marbella (Málaga)*	*Málaga*
Experiencia	C		x			Extracción val.	*3-5 años*	*{ "gte": 3, "lte": 5 }*
Vacantes	C			Sustit. nulos				*1*
Tipo de contrato	C					Normalización	*Contrato Indefinido*	*Indefinido*
Salario	C		x			Extracción val. Cálculo medias	*27.000 € - 33.000 € Bruto/año*	*{ "gte": 27000, "lte": 33000, "mean": 30000 }*
Categoría o nivel	C	x	x			Normalización	*Técnico*	*Técnico / Especialista*
Jornada laboral	C		x			Normalización	*Jornada completa*	*Jornada Completa*
Descripción	C				Exclusión		*En Tecnoexample estamos...*	
Modalidad de Presencialidad	C		x			Normalización	*(Híbrido)*	*Híbrido*
Clasificación	C	x	x			Normalización	*Java / Spring*	*[JAVA, SPRING]*
Puesto	C	x	x				*Analista Programador*	*Analista Programador*
Área funcional	I		x	Sustit. nulos				*Tecnología e informática*
Sub-área funcional	I		x				*Programación*	*Programación*
Funciones	I		x		Exclusión		*- Su misión principal es ...*	
Requisitos	I		x		Exclusión		*- Trato correcto y agradable...*	
Se ofrece	I		x		Exclusión		*- Coche de empresa...*	
Titulación Académica	T		x			Normalización	*FP2/Grado Superior*	*F2 - FP2/G. Superior*
Idioma	T	x	x				*Inglés (Medio)*	*Inglés (Medio)*

Cuadro 3.7: Tareas de procesado de datos

Con el objetivo de ilustrar el resultado de la adaptación a los casos de uso, a continuación se muestra una de las representaciones de la interfaz web de *Airflow*. Se trata de una de las vistas de ejecuciones que ofrece la plataforma para las ejecuciones del *DAG* de transformación y carga en *Elasticsearch* aplicado al portal *Infoempleo*:

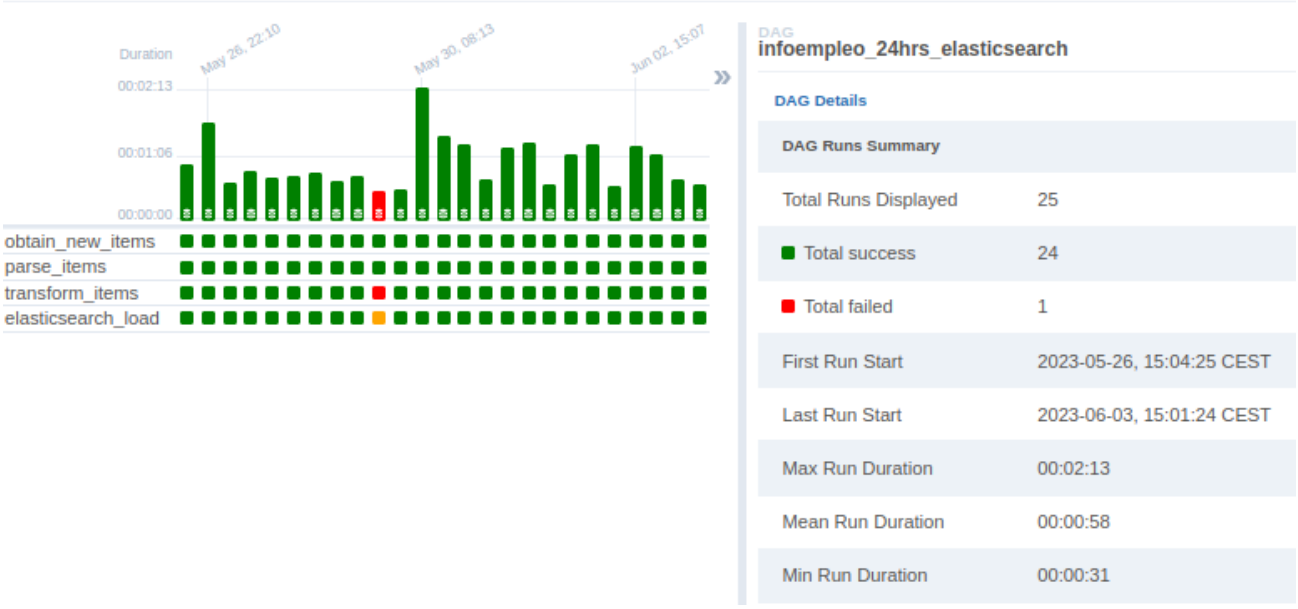


Figura 3.20: Interfaz web *Airflow*: Ejecución DAG transformación y carga en *Elasticsearch*

Por último, para concluir esta sección en la siguiente tabla se relacionan y describen los ficheros de código fuente utilizados para el desarrollo de esta solución, que incluyen el paquete relativo a la definición de *DAGs* y los dos paquetes del motor de extracción, cuyos contenidos se encuentran especificados en 3.2.3. Aplicación a los casos de uso (página 25)

Estructura de ficheros fuente	Descripción paquetes y módulos
src	**Directorio de código fuente**
uned	
dags	**Definición de DAGs de *Airflow***
backup.py	**DAGs** de copias de respaldo
extract.py	**DAGs** de extracción de datos
extract_traveller_execution.py	Objetos auxiliares para la ejecución del motor de extracción de datos
notification.py	**DAGs** de notificación
scraping_executions.yaml	**Fichero de definición de tareas de extracción**
transform_load_elasticsearch.py	Transformaciones de datos para la carga en *Elasticsearch*
transform_load_parallel_parse.py	Objetos para la ejecución concurrente de parseos
transform_load.py	**DAGs** de transformación y carga en *Elasticsearch*
scraping_impl	**Núcleo del motor de extracción de datos**
(...)	
scraping_lib	**Implementaciones del motor de extración de datos**
(...)	

Cuadro 3.8: Estructura de ficheros del Subsistema de planificación y ejecución de flujos de trabajo

3.5. Infraestructura del sistema

Esta sección describe la solución construida para atender al objetivo 5 de este trabajo:

> **OBJ-05** Entrega empaquetada de la arquitectura completa del sistema que posibilite su despliegue en diferentes plataformas software de forma estandarizada, rápida, sencilla y automatizada.

Como solución tecnológica general, de las opciones presentadas en el tema anterior (sección 2.5 en página 13) se ha decidido optar por una virtualización basada en *Docker* y *Docker Compose*, motivado por que en principio los requerimientos de almacenamiento y procesado del trabajo no justifica optar por una solución en clúster. De todas formas, en el caso de que por condicionantes ajenos a los estudiados en este trabajo se requiriese su despliegue en otra de las soluciones contempladas, la solución ha sido diseñada de forma que fuese fácilmente adaptable.

A continuación se muestra un diagrama general de la solución:

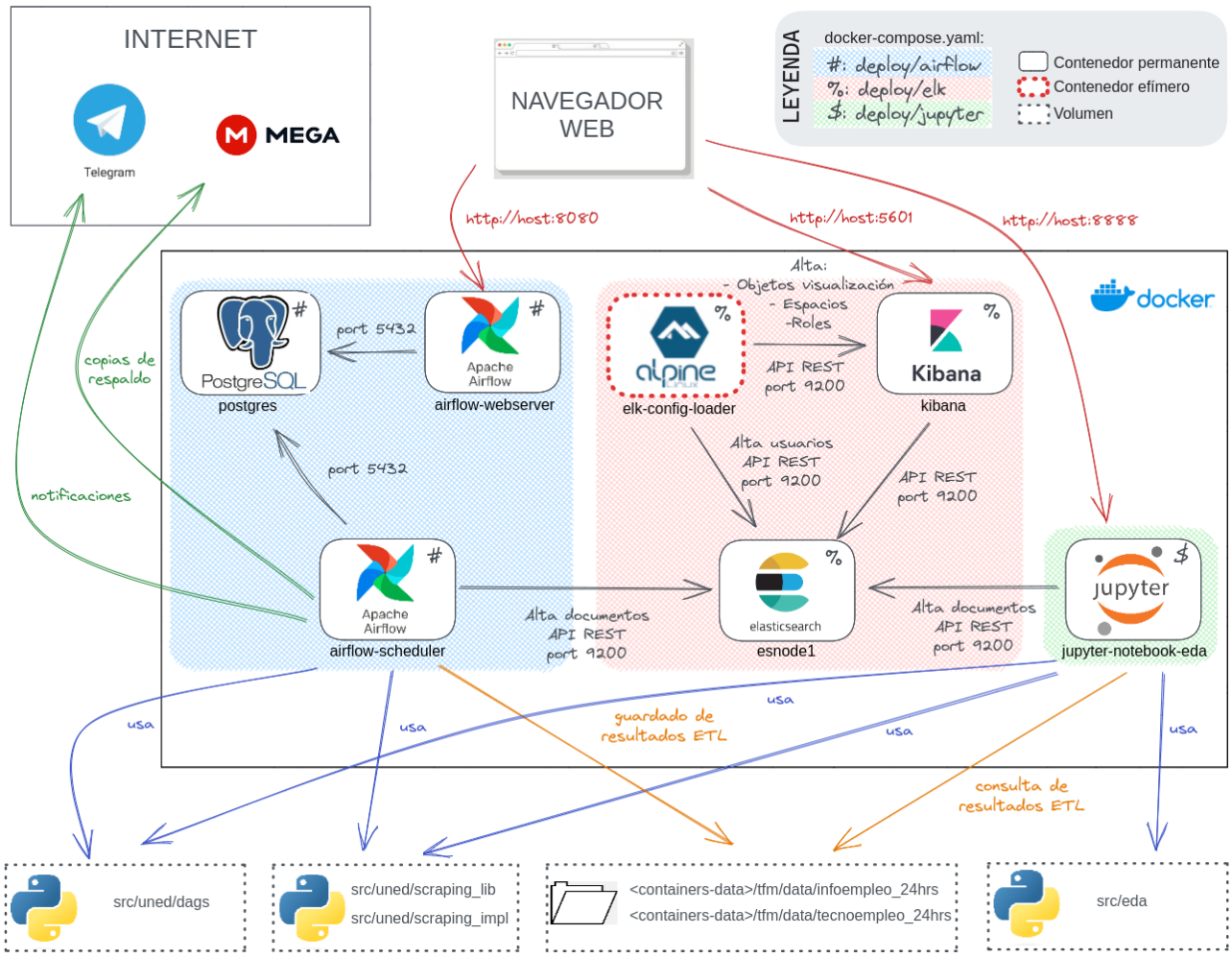


Figura 3.21: Diagrama general de infraestructura

Como se puede apreciar en el diagrama anterior se ha optado por la definición de tres ficheros *Docker Compose*, que comparten red para la comunicación entre sus servicios y exponen los puertos *HTTP* de acceso a los servicios web descritos en secciones anteriores, en función de las tecnologías desplegadas: *Airflow*, pila *ELK* y *JupyterLab* .

Antes de comentar las particularidades de cada una de estas tres definiciones de servicios, a continuación se presenta la estructura de ficheros completa que compone del proyecto, de la que para facilitar su interpretación se ha excluido el detalle de los ficheros descritos en secciones anteriores (tabla 3.4 en página 30 y tabla 3.8 en página 48):

Ficheros	Descripción
├── deploy	**Ficheros de despliegue en infraestructura**
│ ├── airflow	**Ficheros de despliegue de *Airflow***
│ │ ├── airflow-jobs-scrapper.Dockerfile	Definición de imagen personalizada de *Airflow*
│ │ ├── connections.yaml	Fichero de definición de conexiones a servicios externos
│ │ └── docker-compose.yaml	Fichero de definición de servicios
│ ├── elk	**Ficheros de despliegue de pila *ELK***
│ │ ├── config	**Ficheros de configuración para la carga de la configuración inicial**
│ │ │ ├── config.json	
│ │ │ ├── elk_objects_export.ndjson	
│ │ │ ├── indexes_templates.json	
│ │ │ └── space_logo.gif	
│ │ ├── docker-compose.yml	Fichero de definición de servicios
│ │ ├── elk-config-loader.Dockerfile	Definición de imagen personalizada de *Elasticsearch*
│ │ └── run.sh	Fichero de arranque de servicios
│ └── jupyter	**Ficheros de despliegue de *JupyterLab***
│ ├── docker-compose.yaml	Fichero de definición de servicios
│ └── jupyter-notebook-eda.Dockerfile	Definición de imagen personalizada de *JupyterLab*
├── eda	**Notebooks y ficheros auxiliares utilizados para el EDA**
│ ├── common_eda.py	
│ ├── infoempleo.ipynb	
│ ├── tecnoempleo.ipynb	
│ └── update_date.ipynb	
├── src	**Código fuente**
│ └── uned	
│ ├── dags	**Definición de DAGs de *Airflow***
│ │ └── (...)	
│ ├── elasticsearch	**Herramienta de carga de objetos de la pila *ELK***
│ │ └── config_loader.py	
│ ├── scraping_impl	**Núcleo del motor de extracción de datos**
│ │ └── (...)	
│ └── scraping_lib	**Implementaciones del motor de extracción de datos**
│ └── (...)	
├── tests	**Tests unitarios y de integración**
│ └── uned_tests	
│ ├── resources	**Recursos de tests**
│ │ └── (...)	
│ ├── scraping_impl	**Núcleo del motor de extracción de datos**
│ │ └── (...)	
│ └── scraping_lib	**Implementaciones del motor de extracción de datos**
│ └── (...)	
├── requirements_airflow_backups_env.txt	**Listados bibliotecas subsistema de planificación y ejecución de flujos de trabajo**
├── requirements_airflow.txt	
├── requirements_elk_cfg.txt	**Listados bibliotecas subsistema de presentación y explotación de información**
├── requirements_jupyter_eda.txt	
└── requirements_lib.txt	**Listado bibliotecas subsistema de extracción de datos**

Cuadro 3.9: Estructura de la solución

La estructura de ficheros se compone de cinco tipos de elementos en su primer nivel, cuyos

contenidos son los que siguen:

- **Directorio *deploy***: Definiciones de servicios y ficheros de configuración para las tres divisiones presentadas.
- **Directorio *eda***: *Notebooks* para el estudio estadístico de datos realizado para este proyecto y ficheros auxiliares utilizados por estos cuadernos .
- **Directorio *src***: Código fuente correspondiente al motor de extracción de datos de portales, definición de grafos acíclicos dirigidos del Subsistema de planificación y ejecución de flujos de trabajo y la herramienta de carga inicial de objetos de la pila *ELK*.
- **Directorio *tests***: Ficheros de pruebas unitarias y de integración del motor de extracción de datos y de sus implementaciones para los casos de uso de este trabajo.
- **Ficheros de requerimientos de bibliotecas *Python***: Relación de ficheros textuales que contienen la relación de bibliotecas *Python* y sus versiones para su importación en cada una de las partes que componen la solución a este trabajo.

Definición de servicios de *Airflow*

La definición de los contenedores de *Airflow* consta de tres servicios principales, el correspondiente a su base de datos -basado en una imagen oficial de la base de datos *Postgres* en su versión 13- y los dos servicios de *Airflow*, correspondientes al planificador y su servidor web. Para la ejecución de los dos servicios se ha creado una imagen de *Docker* personalizada, basada en una versión ligera oficial de *Apache Airflow* 2.5.3, que monta los entornos virtuales necesarios e incorpora las librerías Python requeridas, indicadas en los ficheros *requirements_airflow.txt* y *requirements_airflow_backups_env.txt*.

Para el guardado de resultados intermedios de los procesos *ETL* ejecutados, se ha definido una variable de entorno para la el montaje del volumen correspondiente, *UNED_SCRAPERS_DATA_DIR*. Por otra parte, se incluye la variable *UNED_ELASTIC_CERTIF_DIR* para informar la ruta local de los certificados *SSL* de las instancia de *Elasticserach* arrancada.

Además de los volúmenes correspondientes a las variables de entorno anteriores, para la ejecución de los procesos ETL, se ha configurado el servicio correspondiente al contenedor del planificador de *Airflow* para que mapee automáticamente como volúmenes todas las rutas de */src* correspondientes a la definición de *DAGs* y al motor de extracción de datos.

Definición de servicios de la pila *ELK*

Esta solución consta de una solución basada en dos contenedores de ejecución permanente; el primero de ellos se corresponde con un nodo *Elasticsearch* (servicio *esnode1*) y el segundo a la instancia de *Kibana*, ambos en la versión 8.7.0.

Para la atención de la demanda de un despliegue automatizado de la solución se ha recurrido a lo siguiente:

- Implementación de una herramienta *Python* para la subida de los objetos de la solución a la pila ELK, que recupera definiciones de los objetos *JSON* que componen la solución (espacios, roles, usuarios, dashboard, vista, índices y plantillas de índices), ubicados en el directorio *deploy/elk/config*, y los sube la pila haciendo uso de las interfaces REST de Kibana y Elasticsearch. El módulo *Python* donde se encuentran los componentes de esta herramienta es *uned.elasticsearch.config_loader*.

- Definición de un contenedor efímero para la ejecución de la herramienta de importación de objetos en la pila *ELK*, cuya arranque se ha vinculado un perfil perfil de *Docker Compose "config-load"* para que se produzca a demanda de la personas administradoras de los contenedores. Para este servicio se ha creado una imagen específica de *Docker*, a partir de una imagen oficial del intérprete de *Python 3.10* ligera, basada *Linux Alpine*, para la que se ha personalizado el punto de entrada para que ejecute la herramienta de subida de objetos y se ha incluido la carga de las bibliotecas necesarias para este cometido.

Para la securización de las peticiones *HTTP* a *Elasticsearch* se ha recurrido a la generación de certificados *SSL* la primera vez se arranca su contenedor. Estos certificados se almacenan en el directorio *certs* de la ruta informada como directorio de datos de elastic mediante la variable de entorno *ELASTIC_STACK_PROJ_DIR*.

Definición de servicios de *JupyterLab*

Este último bloque de servicios consta de un único contenedor para el arranque de la ejecución de cuadernos mediante *JupyterLab*. Este servicio utiliza una imagen personalizada que se basa en la imagen oficial de *JupyterLab* para Python 3.10, a la que se añaden en su carga las librerías específicas utilizadas en los EDA y las requeridas por la motor de extracción de datos de portales.

Para habilitar la ejecución desde notebooks de tareas de extracción, transformación y carga de datos, este servicio monta como volúmenes, además del directorio específico que contiene los notebooks de los EDA, tanto los directorios correspondientes al código fuente del trabajo, como el que alberga los certificados de Elasticsearch y el directorio destino de las transformaciones de *Airflow*.

3.6. Resumen

En este capítulo, de cada una de las áreas identificadas en este trabajo, se han expuesto sus necesidades funcionales y técnicas para, seguidamente, presentar una solución que alcanza el objetivo de desarrollar un sistema que da cobertura a dichas necesidades y finalmente ser aplicado a dos casos de uso reales.

Alcanzado este punto resulta importante evaluar la adecuación de dicho sistema, una vez desarrollado e implantado, a los objetivos presentados en la sección 1.1. Motivación y objetivos, cuestión que se aborda en el siguiente capítulo.

Capítulo 4

Evaluación

Atendiendo a los objetivos de este trabajo, detallados en la sección 1.1. Motivación y objetivos, este capítulo tiene como finalidad la evaluación del presente trabajo, expresada en su séptimo objetivo:

> **OBJ-07** Elaborar de una memoria final en la que se detalle y evalúe el proceso para la consecución de cada uno de estos objetivos y sus resultados.

Para llevar a cabo esta evaluación, en las secciones que componen este capítulos se comenta el nivel de adecuación de la solución desarrollada a dichos objetivos.

4.1. Introducción

Como se ha expuesto en el capítulo 1. Introducción, motivación y objetivos, el propósito de este trabajo es dar cobertura a la identificación de las necesidades del mercado laboral mediante el desarrollo e implantación de un sistema de información de apoyo a la toma de decisiones en la elección de áreas de interés para la impartición de cursos de información, basándose en la información publicada en portales web de búsqueda de empleo.

Para la consecución de dicho cometido, de acuerdo con el objetivo 1 de este trabajo, que bajo estas líneas se refleja, en el capítulo 2. Estado del arte de esta memoria se ha plasmado un resumen de la distintas opciones del panorama tecnológico actual tenidas en cuenta para la consecución de los objetivos del presente trabajo.

> **OBJ-01** Estudiar las tecnologías, conceptos y patrones de diseño aplicables a la solución final.

Este estudio ha tenido influencia directa en la fundamentación de la toma de decisiones relativas al diseño e implementación de la solución final, permitiendo optar por la adopción de soluciones técnicas con la seguridad de disponer de un estudio previo de distintas alternativas existentes en el momento actual, aumentando la probabilidad de éxito para las distintas opciones escogidas en el

desarrollo de la solución final. Todas estas decisiones se han reflejado y justificado en las secciones del capítulo 3. Detalle de la propuesta de la presente memoria.

Además, como resultado de este estudio previo se ha obtenido la segregación de la solución a este trabajo en sus distintas áreas funcionales, lo que ha facilitado las labores de las fases posteriores del desarrollo del sistema, permitiendo la división en sus distintos subsistemas, garantizando el desarrollo un sistema basado en componentes escalables, cohesivos y con un bajo nivel de acoplamiento entre ellos.

Para la valoración del nivel de adecuación de la solución técnica al resto de objetivos del sistema, se ha aplicado dicha solución a dos portales de búsqueda de empleo, cuyos detalles en cuanto a su descripción y desarrollo se exponen en el capítulo 3. Detalle de la propuesta de esta memoria. Esta aplicación a los casos de uso tienen relación con el siguiente objetivo:

OBJ-06 Implementar un sistema que lleve a cabo un uso integral de los componentes descritos sobre casos de uso reales de portales de empleo de distinta naturaleza.

La adecuación del sistema este último objetivo ha sido completa, pues se ha logrado adaptar la solución desarrollada a los dos casos de uso, y se ha implantado la solución en distintas arquitecturas hardware. El motor de extracción de datos, en el momento de la redacción de esta memoria, ha estado en permanente funcionamiento durante dos meses, obteniéndose durante este periodo un total de 21.003 ofertas de empleo.

En las siguientes secciones se evalúa la aplicación de la solución a estos casos de uso desde el punto de vista de cada una de las áreas funcionales identificadas en este sistema y, finalmente, desde el de su infraestructura de despliegue. En todas estas secciones se comienza por reflejar en primer lugar el objetivo del trabajo al que hacen referencia, para a continuación evaluar sus resultados en base a los resultados obtenidos.

4.2. Extracción de datos de portales web

OBJ-02 Construcción de un motor de extracción de datos de portales web de empleo enfocado en facilitar las siguientes cuestiones:

- Inclusión sencilla de artefactos software parseadores de distintos portales de empleo.
- Adaptación de dichos artefactos a cambios de diseño de la interfaz gráfica de los portales.
- Incorporación dinámica y desacoplada de elementos software que permitan tratar los datos extraídos.

Como se ha comentado anteriormente, el desarrollo de este sistema se ha abordado iterativamente, acometiendo en primer lugar el desarrollo de los elementos de infraestructura, para a continuación trabajar en distintas iteraciones de distinto alcance para dar cobertura a los objetivos de este trabajo. Entre esos elementos de infraestructura se encuentra la biblioteca correspondiente al motor de extracción de datos, para la que su aplicación a los casos de uso se ha abordado en distintas iteraciones en momentos diferentes.

Esta forma de desarrollar el sistema ha permitido apreciar las bondades de disponer de una biblioteca que abstraiga al desarrollador de los detalles del proceso completo de extracción de datos. Esto está motivado porque se han tenido que abordar, en diferentes momentos discontinuos en el tiempo, distintos desarrollos relacionados con la extracción de datos de los dos portales de los casos uso, demostrándose esta librería de utilidad en dos aspectos fundamentalmente:

- Permitiendo al desarrollador centrarse exclusivamente en implementar la funcionalidad de extracción de las características concretas de las páginas que aplicaban en cada iteración, de forma independiente para cada portal y despreocupándose de los detalles del Proceso de extracción de información (página 20).

- Habilitando una interfaz de interactuación con los portales para el Subsistema de planificación y ejecución de flujos de trabajo, para cuyas tareas implicadas en la extracción de datos no se han visto afectadas en los aspectos relacionados con la ejecución del mecanismo de ejecución del motor, durante los cambios de atributos de cada iteración del desarrollo. Demostrándose también como una herramienta útil para la abstracción de estos detalles de cara a los clientes del motor de extracción.

Estos dos aspectos tienen influencia directa en la valoración de las tres cuestiones que figuran en el objetivo que da pie a esta sección, pues tienen relación con la demostración de la utilidad del motor para facilitar la inclusión de cambios en los portales en distintos niveles y la demostración de la utilidad de su uso como herramienta para la incorporación desacoplada de elementos software de extracción de datos.

Desde otra perspectiva de la valoración de la adecuación de los resultados obtenidos para la extracción de datos de portales web, como se presentó en el apartado "Pruebas automatizadas unitarias y de integración" (página 30), como parte del desarrollo de la solución a este trabajo se han incluido 68 tests automatizados unitarios y de integración. Estos tests abarcan la mayor parte de los caminos críticos en el uso de la librería y evalúan la correcta obtención de la totalidad de las más de 30 características a extraer de los casos de uso. En términos absolutos, en la ejecución de los tests se ha alcanzado una cobertura del $90\,\%$ de las sentencias del código fuente del motor de extracción de datos y sus implementaciones de casos de uso, con un $100\,\%$ de tasa de éxito en su ejecución para distintos ejemplos de los casos de uso. En la tabla 4.1 (pagina 58) se refleja el detalle de la cobertura de los tests mencionados para los distintos módulos que componen este área funcional.

Como resumen, se puede afirmar, tras la aplicación de la solución para la extracción de datos a los casos de uso de los dos portales de empleo, que se ha obtenido un resultado muy satisfactorio tanto en aspectos relacionados con la extensibilidad, simplicidad y mantenibilidad de la biblioteca, como en términos absolutos de valoración de su fiabilidad y testeabilidad.

Paquete	Módulo	Sentencias	Sentencias no cubiertas	Tasa de cobertura
uned.scraping_impl.parsers	Infoempleo	180	4	98 %
	Job	8	0	100 %
	JobItemCardParser	27	1	96 %
	Tecnoempleo	191	5	97 %
uned.scraping_impl	observers	40	24	40 %
	requests	43	3	93 %
uned.scraping_lib	ItemCard	5	0	100 %
	ItemCardParser	21	2	90 %
	Parser	12	2	83 %
	ParsersFactory	18	0	100 %
	ParsingResult	10	0	100 %
	ParsingResultObserver	6	1	83 %
	Request	9	2	78 %
	RequestsFactory	12	0	100 %
	Result	9	1	89 %
	ResultsPage	10	1	90 %
	ResultsPageParser	29	3	90 %
	ResultsPagesTraveller	46	29	37 %
	TravelTaskConfigLoader	25	0	100 %
	utils	54	0	100 %
	Total	**755**	**78**	**90 %**

Cuadro 4.1: Sentencias de código fuente cubierto en tests automatizados

4.3. Presentación de información

OBJ-03 Proporcionar interfaces gráficas dinámicas y herramientas de diseño on-line de las mismas, para la explotación en tiempo real de la información obtenida, con el propósito de la identificación de posibles áreas formativas de interés en función de su demanda en el mercado laboral.

Para valorar la adecuación de la solución aportada para este área funcional, aplicada al desarrollo de los casos de uso, resulta conveniente hacer también mención a la influencia del ciclo iterativo del desarrollo de la solución a este trabajo. En las iteraciones iniciales de dicho ciclo se desarrollaron dos cuadros de mando, uno por portal, que finalmente se unificaron en uno solo tras la depuración de los procesos de procesamiento de datos. Para acometer esta evolución guiada por el análisis de las características de los juegos de datos, se ha revelado de gran utilidad el uso tanto de cuader-

nos interactivos para la ejecución de código, como de la herramienta gráfica de propósito general (*Discover*).

Por otra parte, el rediseño constante de los cuadros de mando para la adaptación a los cambios derivados de las distintas iteraciones del ciclo de desarrollo y su integración final en un solo cuadro de mando general, ha requerido un uso intensivo de la funcionalidad de diseño online de cuadros de mando de *Kibana* y de las herramientas auxiliares para la gestión, exportación e importación de los objetos de la pila *ELK*, obteniendo en todo momento resultados satisfactorios tras su utilización.

En cuanto al cuadro de mando diseñado para responder a la solución en su aplicación a los casos de uso, ha quedado patente su utilidad para la identificación de la distribución de las vacantes de las ofertas de empleo desde más de 20 perspectivas diferentes, que permiten la identificación de áreas de interés de distinta naturaleza para elaboración de cursos de formación. La representación de sus visualizaciones gráficas interactivas desde todos estos puntos de vista, que habilitan la incorporación en tiempo real de criterios de filtrado al cuadro de mando completo, resulta clave para la identificación de dichas áreas.

Las múltiples iteraciones del ciclo de desarrollo también han forzado a que durante el desarrollo de la solución se haya tenido que hacer un uso frecuente de estas funcionalidades gráficas para los distintos cuadros de mando que se han ido desarrollando, alcanzando siempre un rendimiento sobresaliente en cuanto a precisión de resultados y tiempos de respuesta para diferentes volúmenes de información. Estos aspectos avalan la idoneidad de las decisiones tomadas en el diseño y construcción de la arquitectura subyacente, detallados en la sección 3.3. Subsistema de presentación y explotación de información (página 32).

En conclusión, se puede decir que el resultado obtenido responde a los criterios de corrección, fiabilidad y usabilidad requeridos para el objetivo de esta sección, mientras la satisfactoria respuesta recibida por las herramientas utilizadas, junto con la similitud del proceso de desarrollo con respecto al ciclo de vida futuro de la solución, garantiza que la solución dispone de las características de estabilidad y adaptabilidad necesarias para la adopción de futuras evoluciones del subsistema.

4.4. Planificación y ejecuciones de flujos de trabajo

OBJ-04 Implantación de un motor de flujos de trabajo que permita la ejecución independiente, fiable, periódica, automatizada y controlada, de los procesos implicados en:

- Las tareas de extracción de los datos obtenidos de los portales de empleo.
- El posterior tratamiento de los datos obtenidos para su puesta a disposición de las áreas usuarias implicadas.
- La notificación y/o persistencia de los productos resultantes de los procesos anteriores.

Fruto de la implantación del Subsistema de planificación y ejecución de flujos de trabajo para los casos de uso de los dos portales de empleo, que ha abarcado el periodo comprendido entre el 7 de abril de 2023 y el 12 de junio de 2023, se ha comprobado la correcta ejecución planificada de los procesos extracción, transformación, notificación y puesta a disposición del Subsistema de presentación y explotación de información para un total de 21.003 ofertas de empleo (17.892 correspondientes al portal *Infoempleo* y 3.111 a *Tecnoempleo*). En la siguiente figura se representan los elementos procesados para distintas escalas temporales:

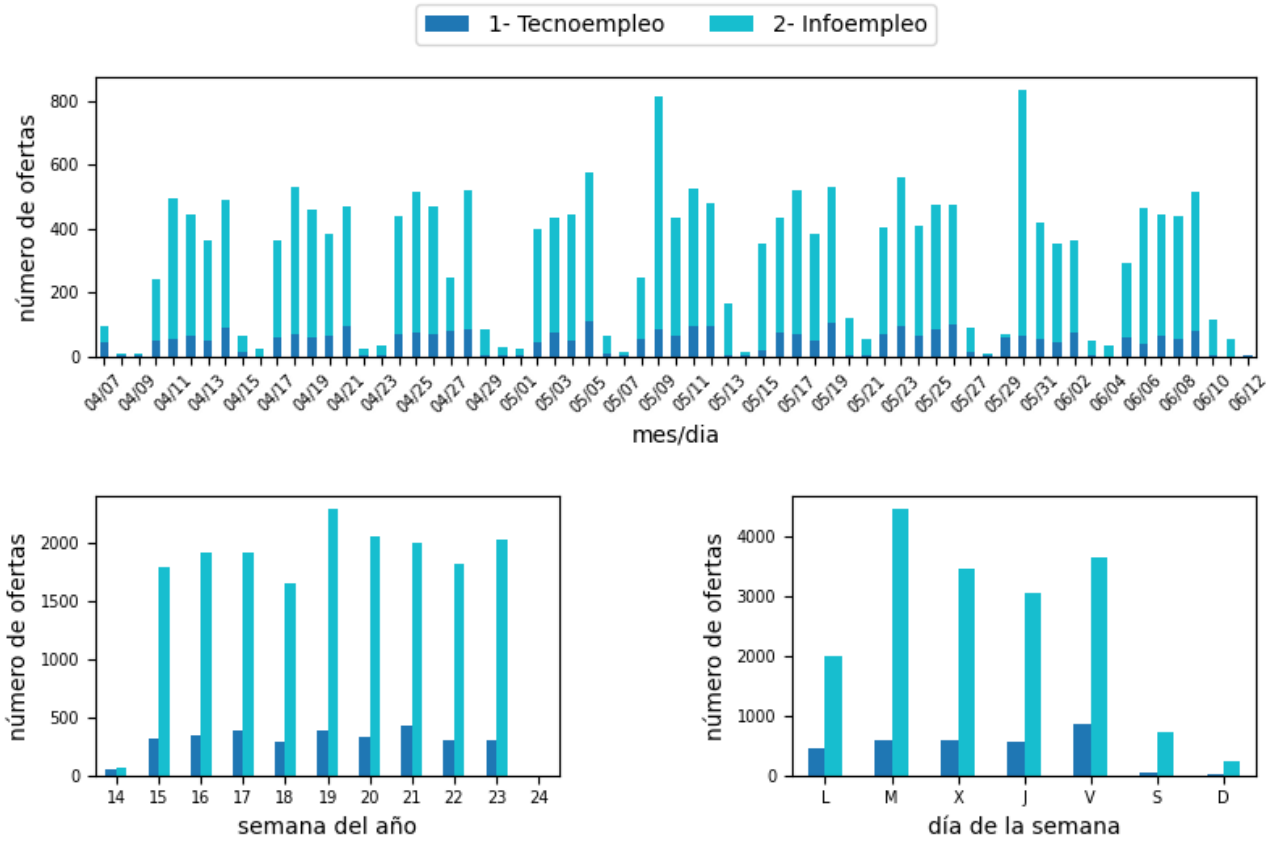


Figura 4.1: Ofertas de empleo procesadas

En cuanto los procesos de extracción de datos, cabe mencionar que inicialmente se fijó una ventana temporal de obtención de las ofertas publicadas en las últimas 24 horas desde el momento de extracción de datos, para ambos portales de los casos de uso, definiendo una frecuencia de ejecución de tres veces al día, como se indicó en el apartado "Definición de tareas de extracción" (página 28). En principio esta configuración se estableció con la intención de que fuera temporal, pues lo que se perseguía era la obtención rápida de un juego de datos válido para el desarrollo del producto y acumular pruebas suficientes de la ejecución de la plataforma. No obstante, muestreos posteriores de los datos del tiempo de permanencia de las ofertas de trabajo en las plataformas, parecen indicar que la opción escogida es correcta para un entorno productivo, pues la proporción de las ofertas que permanecen un único día en la plataforma es mayor de la esperada inicialmente.

En la figura 4.2 (página 61) se incluye un diagrama donde se aprecia su distribución:

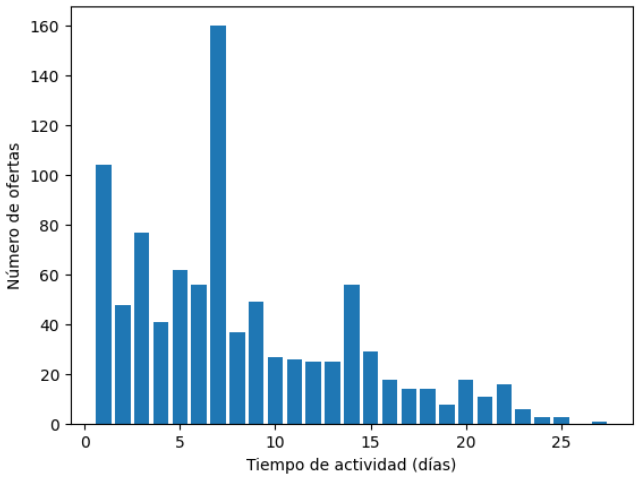


Figura 4.2: Distribución del tiempo actividad de ofertas de empleo

Durante el proceso desarrollo e implantación de la solución para este área funcional se han llevado a cabo el siguiente número aproximado de ejecuciones de los distintos flujos de trabajo desde distintas instalaciones:

- 500 del proceso de extracción de datos.
- 250 del proceso de transformación y carga en *Elasticsearch*.
- 400 del proceso de transformación y notificación.
- 77 del proceso de copias de respaldo.

En todas estas ejecuciones, la elección tecnológica para el planificador se ha demostrado fiable, atendiendo puntualmente a los requerimientos planificación en su ejecución y, en la ejecución de las flujos de trabajo, mostrándose siempre estable en la recuperación ante fallos de implementación de las tareas o de no disponibilidad de servicios externos. Además se ha consolidado como una solución ligera para la potencia y versatilidad que ofrece, no llegando a *1* GB de memoria consumida en estado de espera.

Otro de los puntos fuertes de la plataforma que ha sido aprovechado en las distintas iteraciones de desarrollo es la adaptabilidad dinámica a cambios de definición de sus *DAGs*, pues los cambios en código de las definiciones de dichos flujos se aplican en caliente tras la modificación del código de sus ficheros.

En relación con el control de los flujos de trabajo, también han resultado de utilidad las interfaces disponibles para la ejecución manual y configuración, además de las múltiples opciones de visualización de los resultados de las ejecuciones de los *DAGs*. A continuación se muestra uno de

las múltiples vistas que ofrece la plataforma de planificación y ejecución para la visualización de resultados, aplicada al flujo de extracción de datos de uno de los portales de empleo:

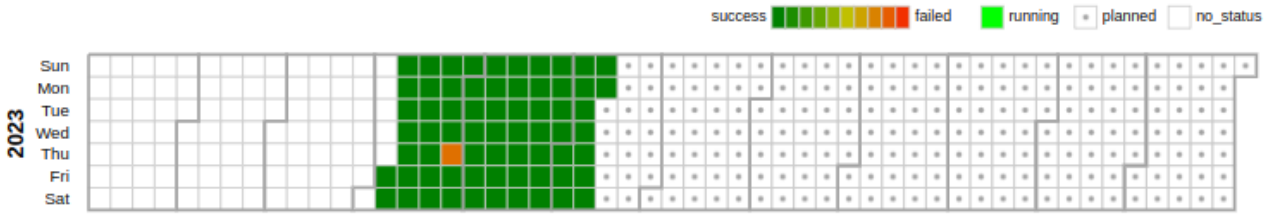


Figura 4.3: Interfaz web *Airflow*: Resultado de ejecuciones para *infoempleo_24hrs*

Una de las limitaciones de la plataforma que ya se ha tratado en la sección 3.4. Subsistema de planificación y ejecución de flujos de trabajo (página 46), es la restricción de *Airflow* para la ejecución monohilo de las tareas que componen los *DAGs*, aspecto que se ha resuelto recurriendo a una implementación específica que hace uso de un entorno *Python* virtual para la ejecución de las tareas de preprocesamiento de los ficheros de los resultados extraídos. Este desarrollo se ha comprobado como un éxito para los casos de uso, llegando a reducir el tiempo de ejecución de las tareas afectadas en hasta un orden de magnitud, dependiendo de las características de las unidades de procesamiento de la máquina donde se ejecute el planificador.

En suma, dejando de lado esta última limitación de la plataforma, que ha sido soslayada con el desarrollo indicado, durante el proceso de desarrollo de este trabajo se ha demostrado que la solución técnica a este área ha resultado ser fiable, extensible y mantenible, cumpliendo satisfactoriamente a nivel funcional y de rendimiento los requisitos de su objetivo, y a un coste contenido en cuanto al consumo de recursos técnicos.

4.5. Infraestructura de despliegue

OBJ-05 Entrega empaquetada de la arquitectura completa del sistema que posibilite su despliegue en diferentes plataformas software de forma estandarizada, rápida, sencilla y automatizada.

Las características del empaquetado de sistemas en contenedores *Docker*, en el que se basa la solución de este trabajo, por definición da respuesta al requerimiento de despliegue estandarizado en múltiples plataformas software, como se ha visto en el apartado 2.5. Empaquetado del sistema (página 13).

En la definición de la arquitectura de la solución, se ha prestado especial atención a que todas las imágenes de contenedores utilizadas dispongan de versiones que soporten ser ejecutadas en distintas arquitecturas hardware, habiéndose comprobado el despliegue de la solución completa en

arquitecturas *amd64* y *arm*. Más concretamente, la solución ha sido implantada y probada tanto ordenadores personales convencionales, como en micro computadores, como la *Raspberry Pi 4 Model B*, y en máquinas virtuales basadas en *Oracle VM VirtualBox* y *VMWare*. A continuación se incluye un listado de características hardware de varias de estas máquinas, junto con sus recursos técnicos disponibles para su ejecución:

	Procesador			
Equipo	**Descripción**	**Cores**	**Threads por core**	**Memoria RAM**
Laptop 1	Intel® Core™ i7-7700HQ CPU @ 2.80GHz	4	2	24 GB
Laptop 2	Intel® Core™ i7-4510U CPU @ 2.00GHz	4	2	8 GB
Raspberry Pi 4 Model B	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz	4	1	8 GB
vmware esxi riopar UNED	Intel® Xeon® 3204 CPU @ 1.90GHz	4	1	8 GB

Cuadro 4.2: Arquitecturas de ejecución de la solución

En todas las arquitecturas indicadas la solución se ha ejecutado con fluidez y se ha comprobado que las características desarrolladas para aprovechar las capacidades de ejecución multihilo de los equipos anfitriones se ejecutan según lo esperado en todas ellas.

Los requerimientos de rapidez y sencillez de despliegue demandados en el objetivo de este apartado también han sido satisfechos, pues, gracias a la utilización de *Docker Compose* para la definición de servicios, junto con el mapeo automático de *DAGs* por parte de *Airflow* y a la herramienta desarrollada para la importación de objetos en la pila *ELK*, se consigue la implantación de la solución completa con la ejecución de sólo tres comandos (ver A. Manual de configuración, ejecución y acceso al sistema de información en la página 75).

4.6. Resumen

Tras la exposición realizada en este capítulo se puede concluir que, una vez desarrollada e implantada la solución propuesta, y tras analizar y evaluar los resultados obtenidos para sus casos de uso, este trabajo ha cumplido con los objetivos inicialmente previstos.

Para finalizar este trabajo, resta indicar las conclusiones obtenidas y posibles evoluciones futuras, cuestiones que se tratan en el siguiente capítulo.

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones del proyecto en base al grado de cumplimiento de los objetivos iniciales del mismo, se proponen trabajos futuros para su evolución y, por último, se indica la referencia a las contribución en forma de código fuente de la solución a este trabajo.

5.1. Conclusiones

En el desarrollo de los capítulos anteriores ha quedado demostrado que con la construcción del producto software resultado de este trabajo se ha logrado el propósito principal previsto. Este propósito se resume en la puesta a disposición, con el objetivo de acercar la oferta formativa de centros educativos a las necesidades actuales de las empresas, de un sistema de información de apoyo a la toma de decisiones en la elección de áreas de interés para la impartición de cursos de información, basado en la explotación de los datos de las publicaciones de ofertas laborales en portales web de búsqueda de empleo.

En la ejecución de dicho cometido se han cumplido los objetivos iniciales del proyecto documentados en la presente memoria, concretándose en las siguientes acciones:

- El estudio inicial del panorama tecnológico actual, en el que se han identificado las áreas funcionales que componen la solución y estudiado las soluciones software aplicables a cada una de estas áreas.

- El desarrollo del sistema de información de apoyo a la toma de decisiones, basado en los siguientes componentes:

    - Una biblioteca como **motor de extracción de datos de portales web de búsqueda de empleo**, que abstrae de los detalles de extracción de datos de cada portal al resto de componentes del sistema, introduciendo el concepto de tarea de extracción de datos de un portal.

  - El **subsistema para la presentación y explotación de información de portales de búsqueda de empleo**, que constituye la interfaz gráfica para la explotación de información por parte de los usuarios finales del sistema, para el estudio de datos y la identificación de áreas de interés en la elaboración de cursos de formación.
  - El **subsistema de planificación y ejecución de flujos de trabajo**, cuyo cometido es hacer de intermediario entre los componentes anteriores, ejecutando la extracción de datos de los portales de forma controlada y planificada, transformando dichos datos en información atendiendo a las necesidades del subsistema de presentación, y poniendo a su disposición dicha información.

- Por último, se ha podido demostrar la utilidad del sistema de información mediante su aplicación a dos casos de uso reales de portales web de búsqueda de empleo en España, de los que que se ha explotado su información durante un periodo aproximado de dos meses y cuyos resultados han sido expuestos en la evaluación de resultados de esta memoria.

Como conclusión, se puede afirmar que se ha obtenido un sistema información que, además de cumplir con las expectativas funcionales requeridas para la identificación de las áreas formativas mencionadas, reúne las características de robustez, extensibilidad, simplicidad, usabilidad y portabilidad requeridos para su explotación en entornos profesionales.

5.2. Trabajos futuros

Como ha quedado patente en esta memoria durante el estudio de las alternativas el panorama tecnológico actual y el diseño de la propuesta, una de las premisas de este trabajo ha sido contar con una herramienta que permita ser ampliada de forma sencilla. Este objetivo ha sido alcanzado sobre las bases del desarrollo de un sistema basado en componentes que se caracterizan por su división en diferentes capas de abstracción y la disponibilidad de funcionalidades para la adición de nuevos elementos software que doten de nuevas características al sistema.

Es por este motivo que las posibilidades de extensión de lo desarrollado en trabajos futuros son muy amplias. No obstante, además de la evolución evidente consistente en la incorporación de nuevos portales de empleo, a continuación se proponen cuatro líneas de actuación futura que el autor de este trabajo ha considerado de interés:

- Diseño e incorporación de tesauros que permitan la explotación jerarquizada de las áreas de interés desde distintas perspectivas.

- Adaptación del proceso de transformación y explotación de datos a nuevas tipologías de fuentes de datos, como portales web de cursos de formación, para el estudio de la cobertura de la oferta formativa actual para las áreas de interés detectadas.

- Incorporación al proceso de transformación de datos de capacidades para el procesamiento del lenguaje natural, con el objetivo de la identificación de nuevas áreas de interés procedentes de las descripciones de las ofertas de trabajo.
- Aplicación de técnicas de clusterización en la explotación de los datos extraídos que permitan identificar patrones no evidentes para la identificación de necesidades formativas multidisciplinares.
- Adición de capacidades multilenguaje a los procesos de procesamiento y explotación de datos.

5.3. Contribuciones

Durante el desarrollo de este trabajo se ha utilizado un servicio web público basado en *Git* (https://gitlab.com/) como herramienta de control de versiones del presente documento y del código fuente que lo acompaña, cuyo repositorio se encuentra referenciado en la siguiente dirección web: https://gitlab.com/DODG/tfm.

Bibliografía

Adecco Group Institute (2022). Informe sobre perfiles deficitarios y escasez de talento en españa.

Bansal, S. K. (2014). Towards a semantic extract-transform-load (etl) framework for big data integration. In *2014 IEEE International Congress on Big Data*, pages 522–529.

Barnes, D. and Kolling, M. (2007). *Programación orientada a objetos con Java*.

Cormier, P. (2022). The state of enterprise open source: A red hat report. https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022.

da Cunha, H. G. V. O., Moreira, R., and de Oliveira Silva, F. (2021). A comparative study between containerization and full-virtualization of virtualized everything functions in edge computing. In *Advanced Information Networking and Applications: Proceedings of the 35th International Conference on Advanced Information Networking and Applications (AINA-2021), Volume 2*, pages 771–782. Springer.

Eckel, B. (2003). *Thinking in Patterns: Problem-Solving Techniques using Java*.

El-Sappagh, S. H. A., Hendawi, A. M. A., and El Bastawissy, A. H. (2011). A proposed model for data warehouse etl processes. *Journal of King Saud University-Computer and Information Sciences*, 23(2):91–104.

Flexera (2023). 2023 state of the cloud report.

Fowler, M. (2005). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA.

Freeman, E. and Freeman, E. (2004). *Head First Design Patterns*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (2006). *Patrones de diseño*. Addison-Wesley, Boston, MA.

github.com (2022). The state of opensource software: The top programming languages. https://octoverse.github.com/2022/top-programming-languages.

Gortázar, F., Martínez, R., and Fresno, V. (2019). *Lenguajes de programación y procesadores*.

Hernández, R., Lázaro, J. C., Dormido, R., and Ros, S. (2003). *Estructuras de Datos y Algoritmos*.

Instituto Nacional de Estadística (2023). Encuesta trimestral de coste laboral (etcl). primer trimestre de 2023.

jetbrains.com (2022). The state of developer ecosystem 2022. https://www.jetbrains.com/lp/devecosystem-2022/.

Kroll, P. and Kruchten, P. (2003). *The rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley Professional.

Larman, C. (2010). *UML y Patrones*.

Martin, R. C. (2003). *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.

Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

Nickoloff, J. and Kuenzli, S. (2019). *Docker in action*. Simon and Schuster.

Ostenero, F. L. (2014). *Teoría de los lenguajes de programación*.

Rad, B. B., Bhatti, H. J., and Ahmadi, M. (2017). An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228.

Richardson, L. (2023). Beautiful soup documentation. https://www.crummy.com/software/BeautifulSoup.

Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison. In *21st twente student conference on IT*, volume 21.

Schenker, G. N., Saito, H., Lee, H.-C. C., and Hsu, K.-J. C. (2019). *Getting Started with Containerization*.

scrapy.org (2023). Scrapy official documentation. https://scrapy.org/.

Seenivasan, D. (2023). Etl (extract, transform, load) best practices. *International Journal of Computer Trends and Technology*, 71(1):40–44.

Selenium community (2023). The selenium browser automation project. https://www.selenium.dev/documentation/about/.

Servicio Público de Empleo Estatal (2023). Tendencias del mercado de trabajo en españa.

Shalloway, A. and Trott, J. R. (2004). *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley, Boston, MA, second edition.

Sommerville, I. (2005). *Ingeniería del software*. seventh edition.

Theodorou, V., Abelló, A., Thiele, M., and Lehner, W. (2017). Frequent patterns in etl workflows: An empirical approach. *Data & Knowledge Engineering*, 112:1–16.

tiobe.com (2023). Tiobe index. https://www.tiobe.com/tiobe-index/.

Turnbull, J. (2014). *The Docker Book: Containerization is the new virtualization*. James Turnbull.

Watada, J., Roy, A., Kadikar, R., Pham, H., and Xu, B. (2019). Emerging trends, techniques and open issues of containerization: a review. *IEEE Access*, 7:152443–152472.

Zamorano, M. R., Yuste, A. R., Abad, L. T., and Gómez, A. (2019). *Visualización de los Datos*.

Nomenclatura

API Application Program Interface

DAG Grafo Acíclico Dirigido

ELK Pila tecnológica Elasticsearch

ETL Proceso de extracción, transformación y carga

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

REST Representational State Transfer

SSL Secure Sockets Layer

Anexo A

Manual de configuración, ejecución y acceso al sistema de información

En este anexo se presentan los requisitos necesarios para la configuración y ejecución del sistema de información resultado del presente trabajo. Consta de las siguientes secciones:

A.1. Requisitos mínimos

Los requisitos mínimos de hardware recomendados para una correcta ejecución de la plataforma en el equipo anfitrión son:

- **RAM**: 8 *GB*
- **CPU**: 4 cores a 1.8 *GHz*
- **Espacio libre en disco**: 50 *GB*

En cuanto a requisitos software, se requiere tener instalada la versión 24 de *Docker Engine* y *Docker Compose*. Además, este equipo ha de disponer de conexión a internet abierta para la descarga de las imágenes base de los contenedores desde *Docker Hub*.

A.2. Estructura de ficheros

Este apartado presenta los ficheros necesarios para la configuración y ejecución de la plataforma, para lo que a continuación se describen aquellos que tienen relación directa con la implantación del sistema, obviando la parte relativa al código fuente y definición de las librerías externas:

Ficheros	Descripción
deploy	**Ficheros de despliegue en infraestructura**
airflow	**Ficheros de despliegue de *Airflow***
.env	Definición de de variables de entorno de *Airflow*
airflow-jobs-scrapper.Dockerfile	Definición de imagen personalizada de *Airflow*
connections.yaml	Fichero de definición de conexiones a servicios externos
docker-compose.yaml	Fichero de definición de servicios
elk	**Ficheros de despliegue de pila *ELK***
config	**Ficheros de configuración para la carga de la configuración inicial**
config.json	Configuración para la subida automatizada de objetos a la pila *ELK*
elk_objects_export.ndjson	Objetos de *Kibana*
indexes_templates.json	Definición de plantillas de índices
space_logo.gif	Logo del espacio *UNED*
.env	Definición de de variables de entorno de *Elasticsearch*
docker-compose.yml	Fichero de definición de servicios
elk-config-loader.Dockerfile	Definición de imagen personalizada de *Elasticsearch*
run.sh	Fichero de arranque de servicios
jupyter	**Ficheros de despliegue de *JupyterLab***
.env	Definición de de variables de entorno de *JupyterLab*
docker-compose.yaml	Fichero de definición de servicios
jupyter-notebook-eda.Dockerfile	Definición de imagen personalizada de *JupyterLab*
eda	**Notebooks y ficheros auxiliares utilizados para el EDA**
(...)	
src	**Código fuente**
uned	
dags	**Definición de DAGs de *Airflow***
scraping_executions.py	Fichero de definición de tareas de extracción de datos
(...)	
(...)	
tests	**Tests unitarios y de integración**
(...)	
(...)	

Cuadro A.1: Estructura de ficheros de implantación y configuración de la solución

Como se expresa en la estructura de ficheros anterior (ver contenidos del directorio *deploy*), la solución consta tres infraestructuras software que se despliegan de forma independiente, dos correspondientes al Subsistema de presentación y explotación de información (pila *ELK* y *JupyterLab*) y una tercera para el Subsistema de planificación y ejecución de flujos de trabajo (*Airflow*).

Cada una de estas infraestructuras tiene su propios ficheros de configuración y su propio fichero *docker-compose.yml*, que define los contenedores docker que las componen y que son parametrizados usando variables de entorno, definidas en sus respectivos ficheros *.env*.

A.3. Configuración

En las siguientes subsecciones se muestran los ficheros de configuración de cada una de las infraestructuras relacionadas en la sección anterior, comenzando por los ficheros *.env*, cuyos contenidos

se encuentran en su mayor parte autoexplicados en los comentarios de las variables.

A.3.1. Pila *ELK*

(Directorio: *deploy/elk*)

A continuación se muestra el contenido del fichero de variables entorno para la ejecución de esta pila. En cuanto a su contenido es importante tener presente el adecuado dimensionamiento el uso de memoria máxima de los nodos *Elasticsearch*, pues en su configuración por defecto tienden a consumir toda la memoria libre de la máquina donde se ejecutan. Esta limitación se configura a través de los atributos de la máquina virtual de los nodos de *Elasticsearch* (variable *ELASTIC_NODE_JAVA_OPTS*), y ha sido dimensionada en $3GB$ para los casos de uso de este trabajo.

```
# Identificador por defecto para la creación de directorios en el equipo anfitrión
ELASTICSEARCH_UID=1000

# Versión del stack ELK
STACK_VERSION=8.7.0

# Puerto de exposición al host anfitrión del API HTTP de Elasticsearch
ES_PORT=9200

# Puerto de exposición al host anfitrión de la interfaz web de Kibana
KIBANA_PORT=5601

# Password del usuario administrador 'elastic'
ELASTIC_PASSWORD=uned100*

# Password del usuario 'kibana_system'
KIBANA_PASSWORD=kibana

# Directorio del equipo anfitrión para el almacenamiento de ficheros de la instalación del stack
ELASTIC_STACK_PROJ_DIR=~/containers-data/tfm/elk

# Configuración de zona horaria
SCRAPERS_TIMEZONE=Europe/Madrid

# Ubicación del equipo anfitrión de la herramienta de subida automática de ficheros al stack
ELASTIC_CONFIG_LOADER_FILE=../../src/uned/elasticsearch/config_loader.py

# Limitación de uso de memoria de Elasticsearch
ELASTIC_NODE_JAVA_OPTS=-Xmx3G
```

Figura A.1: Fichero *.env* pila *ELK*

Además de la configuración de las variables de entorno descritas en la figura anterior, como parte del desarrollo de la herramienta de importación de objetos en la pila *ELK* para este trabajo se ha definido el fichero *config/config.json* para la identificación de los objetos que se han de dar de alta de forma automatizada (contenidos en listado de código A.1). En concreto, la descripción de los atributos de primer nivel de este fichero es la siguiente:

- *spaces*: Datos los espacios *Kibana* a dar de alta y referencia al fichero de sus correspondientes imágenes de espacio.

- *roles*: Definición de nuevos roles, donde se especifican sus niveles de acceso a objetos específicos de *Elasticseach* y *Kibana*.
- *users*: Datos de nuevos usuarios y sus asignaciones de roles.
- *indexesTemplatesFile*: Ubicación del fichero donde se encuentran definidas las plantillas de índices de *Elasticsearch*.
- *kibanaObjects*: Ubicación del fichero de exportación/importación de objetos *Kibana*.

Listado de código A.1: Fichero de *config.json* de la pila *ELK*

```
"spaces": [
  {
    "id": "uned",
    "name": "UNED",
    "initials": "U",
    "description": "",
    "color": "#03473A",
    "disabledFeatures": [],
    "imageFile": "space_logo.gif"
  }],
"roles": [
    {
      "name": "uned_role",
      "metadata": {
        "version": 1
      },
      "elasticsearch": {
        "cluster": [],
        "indices": [
          {
            "names": ["infoempleo*", "tecnoempleo*"],
            "privileges": ["read", "view_index_metadata"],
            "allow_restricted_indices": false
          }
        ]
      },
      "kibana": [
        {
          "base": [],
          "feature": {
            "dashboard": ["read"]
          },
          "spaces": ["uned"]
        }
```

```
        ]
      }],
  "users": [
    {
      "username": "uned_user",
      "password" : "uned100*",
      "roles": ["uned_role"],
      "full_name": "",
      "email": ""
    }],
  "indexesTemplatesFile": "indexes_templates.json",
  "kibanaObjects": [
      {
        "spaceId": "uned",
        "objectsFile": "elk_objects_export.ndjson"
      }]
```

A.3.2. Airflow

(Directorio: *deploy/elk*)

El contenido del fichero de configuración de variables de entorno para la ejecución de *Airflow* se muestra a continuación:

```
# Identificador por defecto para la creación de directorios en el equipo anfitrión
AIRFLOW_UID=1000

# Directorio del equipo anfitrión para el almacenamiento de ficheros de la instalación de Airflow
AIRFLOW_PROJ_DIR=~/containers-data/tfm/airflow

# Directorio del equipo anfitrión de ubicación de los certificados SSL de conexión al stack ELK
UNED_ELASTIC_CERTIF_DIR=~/containers-data/tfm/elk/certs

# Directorio del equipo anfitrión de ubicación del código fuente del proyecto
AIRFLOW_DAGS_DIR=../../src

# Directorio del equipo anfitrión para el almacenamiento de ficheros de la base de datos de Airflow
POSTGRES_DATA_DIR=~/containers-data/tfm/postgres

# Directorio del equipo anfitrión para el almacenamiento de ficheros de resultados intermedios de
# ejecución de los procesos ETL de los DAGs de búsqueda de empleo
UNED_SCRAPERS_DATA_DIR=~/containers-data/tfm/data

# Configuración de zona horaria
SCRAPERS_TIMEZONE=Europe/Madrid

# Usuario y contraseña de acceso web a airflow
_AIRFLOW_WWW_USER_USERNAME=uned
_AIRFLOW_WWW_USER_PASSWORD=uned100*
```

Figura A.2: Fichero *.env Airflow*

Para la configuración de *Airflow* se incluye un fichero específico para la definición de conexiones a servicios externos (contenido mostrado en la Figura A.3 de la página 80), que para los servicios utili-

zados para los casos de uso contemplados en este trabajo a continuación se describen sus contenidos más relevantes:

- *uned_mega_backups*: Conexión a *MEGA.nz* para su uso por el *DAG* copias de respaldo.
  - *login*: usuario de *MEGA.nz*.
  - *password*: contraseña de *MEGA.nz*.
  - *schema*: directorio de guardado de copias de respaldo dentro de la estructura ficheros del usuario de *MEGA.nz*.
- *uned_telegram_notif*: Conexión a *Telegram* para los *DAGs* de notificación.
  - *host*: Identificador de la conversación a la que dirigir las notificaciones.
  - *password*: Identificador del bot de *Telegram* de envío de notificaciones.
- *uned_elasticsearch*: Conexión a *Elasticsearch* para los *DAGs* de transformación y carga.
  - *extra*: Ruta ubicación del fichero de clave pública para la conexión mediante *https* a *Elasticsearch*.
  - *host*: Nodo de conexión a *Elasticsearch*.
  - *login*: Usuario de conexión a *Elasticsearch*.
  - *password*: Contraseña de conexión a *Elasticsearch*.
  - *port*: Puerto de conexión a *Elasticsearch*.

```yaml
uned_mega_backups:
  conn_type: http
  description: ''
  extra: ''
  host: ''
  login: pupil@alumno.uned.es
  password: password
  port: null
  schema: backups

uned_telegram_notif:
  conn_type: http
  description: ''
  extra: ''
  host: 'conversation_id'
  login: ''
  password: telegram_bot_id
  port: null
  schema: ''

uned_elasticsearch:
  conn_type: http
  description: ''
  extra: /home/airflow/elastic_certif/ca/ca.crt
  host: esnode1
  login: elastic
  password: uned100*
  port: 9200
  schema: ''
```

Figura A.3: Fichero de *connections.yaml Airflow*

A continuación se muestra el contenido del fichero *YAML* diseñado para la definición y configuración de tareas de extracción de datos de portales, ubicado en *src/uned/dags/scraping_executions.py*. Si bien se trata de un fichero más propio de la definición de los *DAGs* de extracción que de configuración de la infraestructura de despliegue, se ha decidido incluir en esta sección para facilitar su identificación:

```yaml
__anchors__:
  entrypoint_request:
    &common_http_headers
    headers:
      Access-Control-Allow-Origin: '*'
      Access-Control-Allow-Headers: 'Content-Type'
      Access-Control-Max-Age: '3600'
      User-Agent: 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0'


infoempleo_24hrs:
  parser: "Infoempleo"
  entrypoint_request:
    request_type: "Http"
    params:
      url: "https://www.infoempleo.com/trabajo/"
      request_method: "POST"
      payload:
        job_type: "empleo"
        search: ""
        region: ""
        diasPublicacion: "1"
        retribucionMin: "0"
        experienciaMin: "0,10"
        jornadaLaboral: ""
        tipoContrato: ""
        categoriaPuesto: ""
        pagina: "1"
        teletrabajo: ""
        ordenacion: "fechaAlta"
      <<: *common_http_headers
  delay_between_page_loads: 1  # SECONDS

tecnoempleo_24hrs:
  parser: "Tecnoempleo"
  entrypoint_request:
    request_type: "Http"
    params:
      url: "https://www.tecnoempleo.com/ofertas-trabajo/?ult_24h=1"
      request_method: "GET"
      <<: *common_http_headers
  delay_between_page_loads: 1  # SECONDS
```

Figura A.4: Fichero scraping_executions.yaml

En el fichero se incluyen tres elementos de primer nivel para su aplicación a los casos de uso:

- Un primer elemento para la definición de anclas para su reutilización en el resto del documento, donde se han especificado las características de las cabeceras *HTTP* que son comunes a los puntos de entrada de las dos tareas de extracción.

- Dos últimos elementos, en los que se definen las dos tareas de extracción de datos y sus tiempos de latencia, que son descritos en más detalle en la sección 3.2.3. Definición de tareas de extracción (página 28).

A.3.3. JupyterLab

(Directorio: *deploy/jupyter*)

El detalle de configuración del entorno ejecución de *JupyterLab* se corresponde íntegramente con las definiciones de variables de entorno en su fichero *.env*, cuyo contenido se muestra a continuación:

```
# Directorio del equipo anfitrión para el almacenamiento de ficheros de resultados intermedios de
# ejecución de los procesos ETL de los DAGs de búsqueda de empleo
UNED_SCRAPERS_DATA_DIR=~/containers-data/tfm/data

# Directorio del equipo anfitrión de ubicación de los certificados SSL de conexión al stack ELK
UNED_ELASTIC_CERTIF_DIR=~/containers-data/tfm/elk/certs

# Directorio del equipo anfitrión de ubicación del código fuente del proyecto
SRC_DIR=../../src

# Directorio del equipo anfitrión de ubicación de los notebooks
NOTEBOOKS_DIR=../../eda

# Token por defecto para la conexión web a JupyterLab
JUPYTER_TOKEN=uned100*

# Configuración de zona horaria
SCRAPERS_TIMEZONE=Europe/Madrid
```

Figura A.5: Fichero *.env Jupyterlab*

A.4. Ejecución y acceso

La ejecución de las tres infraestructuras software sigue el modelo estandarizado para la orquestación mediante *Docker Compose* en cuanto a la descarga de imágenes de contenedores, y su creación y ejecución a partir de las definiciones de servicios basada en ficheros *docker-compose.yml*. Estos ficheros de definición de servicios se encuentran en la raíz de los directorios correspondientes a cada una de las tres infraestructuras (ver tabla A.1 en página 76).

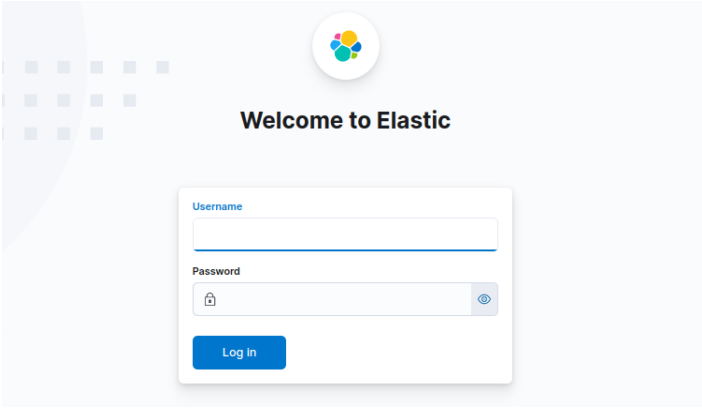
A continuación se indican los comandos que hay que ejecutar[1] desde el directorio de cada infraestructura para arrancarlas y cómo acceder a sus componentes principales de sus interfaces web, cuyas direcciones web y credenciales de acceso por defecto vienen relacionados en la sección A.5. Relación de direcciones web y credenciales de acceso por defecto (página 86).

A.4.1. Pila ELK

- **Directorio:** *deploy/elk*
- **Comandos de ejecución:**
  1. Arrancar la pila *ELK*:

     `docker compose up`
  2. Una vez arrancada la pila *ELK*, ejecutar la herramienta de importación de objetos [2]:

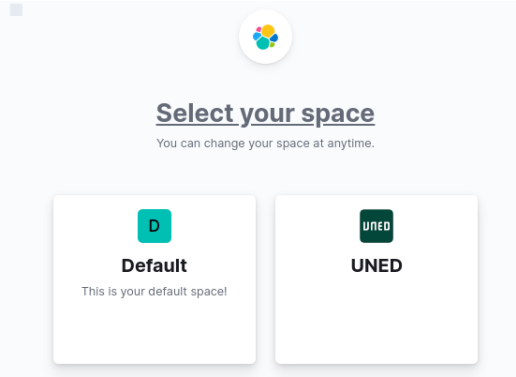
     `docker compose --profile config-load up elk-config-loader`
- **Acceso al cuadro de mando:**
  1. Autenticarse en la interfaz web mediante el usuario y contraseña correspondientes:



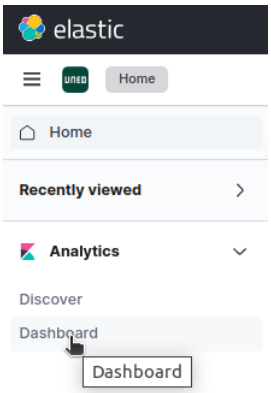
[1] Según la configuración de la instalación de *Docker*, es posible que el sistema requiera para ejecutar los comandos indicados a continuación haya que elevar los privilegios de ejecución (*sudo*).

[2] Este comando puede ejecutarse tantas veces como se necesite utilizar la herramienta de importación de objetos. En cada ejecución se volverán a crear los objetos referenciados en el fichero *config/config.json* (ver sección A.3.1. Pila *ELK* en página 77 para más información).

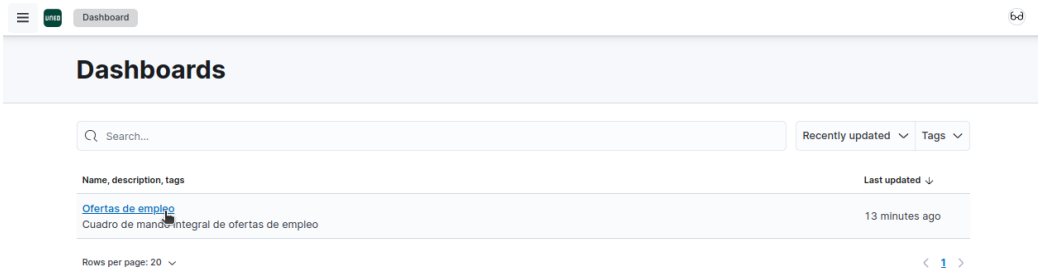
2. Si el usuario tiene acceso a más de un espacio, como por ejemplo ocurre con el usuario administrador "*elastic*", se mostrará la siguiente interfaz donde hay que seleccionar el espacio *UNED*:



3. Pulsar en el menú y después en *Analytics/Dashboard*:



4. Pulsar en el enlace *"Ofertas de empleo"*:



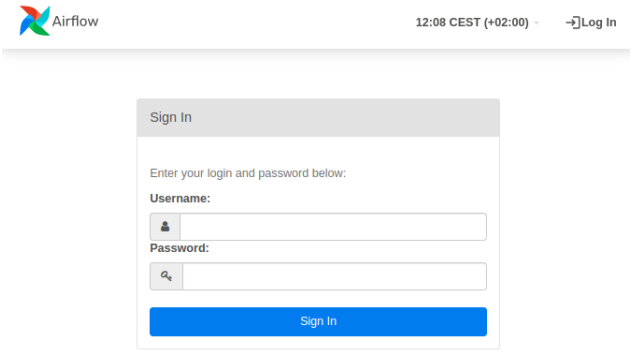
A.4.2. Airflow

- **Directorio:** *deploy/airflow*

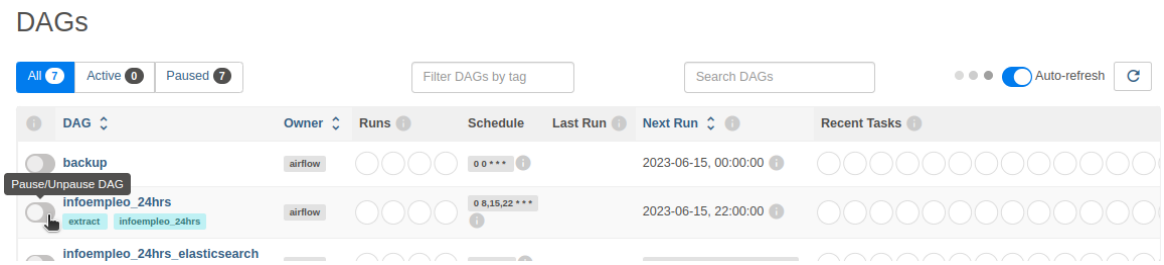
- **Comando de ejecución:** `docker compose up`

- **Acceso a interfaz web y activación de *DAGs*:**

  1. Autenticarse en la interfaz web mediante el usuario y contraseña correspondientes:



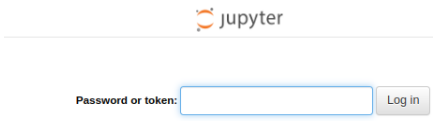
  2. Una vez autenticado por primera vez es importante activar los *DAGs* pulsando en el control que se indica en la siguiente imagen:



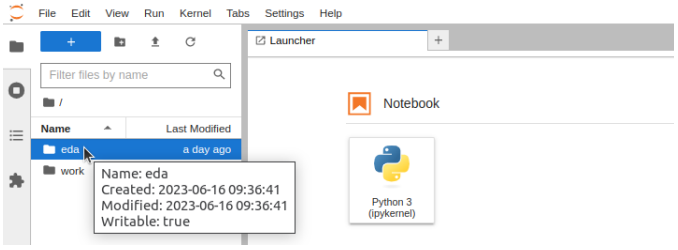
A.4.3. JupyterLab

- **Directorio:** *deploy/jupyter*
- **Comando de ejecución:** `docker compose up`
- **Acceso a interfaz de visualización y ejecución de los scripts de *EDA*:**

  1. Autenticarse en la interfaz web mediante el token de autorización correspondiente:



  1. Pulsar en el directorio *eda*:



A.5. Relación de direcciones web y credenciales de acceso por defecto

A.5.1. Datos de acceso pila *ELK*

- **Dirección web:** http://localhost:5601
- **Usuario de consulta:**
  - Usuario: *uned_user*
  - Contraseña: *uned100**
- **Usuario administrador:**
  - Usuario: *elastic*
  - Contraseña: *uned100**

A.5.2. Datos de acceso *Airflow*

- **Dirección web:** http://localhost:8080
- **Usuario:** *uned*
- **Contraseña:** *uned100**

A.5.3. Datos de acceso *JupyterLab*

- **Dirección web:** http://localhost:8888
- **Token de acceso:** *uned100**