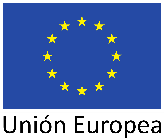




MANUAL TÉCNICO



Programa Operativo Fondo Europeo de Desarrollo Regional Aragón 2014-2020

*Construyendo Europa desde Aragón*

Contenido

[1. Requisitos del entorno 4](#_Toc58600896)

[1.1 Componentes principales 4](#_Toc58600897)

[1.1.1 Python3 4](#_Toc58600898)

[1.1.2 Entornos virtuales 4](#_Toc58600899)

[1.1.3 Apache Airflow 4](#_Toc58600900)

[1.1.4 PostgreSQL 4](#_Toc58600901)

[2. Desarrollos realizados 6](#_Toc58600902)

[2.1 Ingesta de los datos 6](#_Toc58600903)

[2.1.1 Ingesta de los artículos del BOE 7](#_Toc58600904)

[2.1.2 Ingesta de los artículos de Aragón 7](#_Toc58600905)

[2.1.3 Ingesta extra 9](#_Toc58600906)

[2.2 Conversión de los documentos 9](#_Toc58600907)

[2.2.1 De PDF a TXT 9](#_Toc58600908)

[2.2.2 De XML a TXT 9](#_Toc58600909)

[2.2.3 De HTML a TXT 10](#_Toc58600910)

[2.3 Extracción de la información 10](#_Toc58600911)

[2.3.1 Extracción mediante reglas 10](#_Toc58600912)

[2.3.2 Extracción mediante el modelo NER 12](#_Toc58600913)

[2.3.3 Extracción de puestos de tablas de anexos 13](#_Toc58600914)

[2.3.4 Modelo jerárquico 14](#_Toc58600915)

[2.4 Inserción en la base de datos 16](#_Toc58600916)

[2.5 Cierre de ofertas 16](#_Toc58600917)

[3. Base de datos 17](#_Toc58600918)

[3.1 Estructura 17](#_Toc58600919)

[3.2 Detalle de los campos 18](#_Toc58600920)

[3.2.1 Tabla Fecha 18](#_Toc58600921)

[3.2.2 Tabla Puesto 18](#_Toc58600922)

[3.2.3 Tabla Oferta 18](#_Toc58600923)

[3.2.4 Tabla Convocatoria 19](#_Toc58600924)

[3.3 Creación de las tablas 19](#_Toc58600925)

[3.4 Vistas 21](#_Toc58600926)

Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Descripción |
| 1.0 |  | Hiberus | Primera versión (incompleta) del documento |

# Requisitos del entorno

Para un correcto funcionamiento de la aplicación desarrollada en el proyecto se requiere de un entorno que contenga ciertas herramientas o*pen source.* Dichas herramientas permiten construir entornos de ejecución aislados, mantener las dependencias necesarias, almacenar los resultados obtenidos u orquestar procesos de manera automática, entre otras.

## Componentes principales

En este apartado se introducen brevemente los componentes que se han instalado y desplegado para crear un entorno de ejecución válido y funcional.

### Python3

Python es un lenguaje de programación interpretado y multiparadigma que soporta programación imperativa, orientada a objetos y funcional. Es uno de los lenguajes más utilizados de la actualidad (2020), ya que se diseñó para aumentar la productividad y la legibilidad del código. Además, a lo largo de los años se han publicado diferentes módulos y *frameworks* que, unido a su sencillez y su velocidad de desarrollo, lo han convertido en una de las mejores opciones a la hora de desarrollar aplicaciones relacionadas con el campo del dato.

Las versiones principales de Python son Python2 (versión antigua y sin soporte) y Python3. Este proyecto ha sido íntegramente desarrollado con Python3, tanto para construir el código necesario como para ejecutar dicho código (el denominado intérprete de Python3 es el proceso encargado de la ejecución de programas escritos en este lenguaje).

### Entornos virtuales

Los entornos virtuales de Python permiten encapsular dependencias para desacoplar los requisitos de una aplicación del sistema en la que se integra, generalmente un servidor. Son muy útiles cuando se despliegan aplicaciones (construidas en Python) en servidores compartidos por distintos proyectos, donde cada uno puede requerir diferentes versiones de un mismo paquete. El uso de entornos virtuales está considerado como una buena práctica ya que, de manera general, evita problemas de incompatibilidad y brinda al desarrollador un amplio control sobre el entorno en el que su código es ejecutado.

En este proyecto se han desplegado dos entornos virtuales, uno en el servidor *frontend* y otro en el *backend*. Posteriormente en este documento se detalla el proceso de creación de ambos entornos.

### Apache Airflow

Apache Airflow es una plataforma *open source* escalable y dinámica utilizada para diseñar, programar y orquestar tareas, ETLs o procesos. Su funcionamiento se basa en ejecutar DAGs (grafos acíclicos dirigidos), definidos como un conjunto de tareas que deben ejecutarse en un orden predefinido sin lazos de realimentación (p. ej. ETL: extracción 🡪 transformación 🡪 carga). Airflow fue desarrollado y liberado por la empresa Airbnb, pero hoy en día la mantiene la organización Apache.

Una de sus características principales, y que le diferencian de otros orquestadores de *workflows,* es que todas las tareas ejecutadas por la plataforma son programadas y desarrolladas con el lenguaje Python. Además, incluye módulos para ejecutar diferentes acciones (conexiones SSH, FTP, etc.), soporta integración junto a otros servicios (HDFS, Spark, PostgreSQL, GCP, AWS, Azure, etc.) y dispone de diferentes modos de ejecución que le permiten ajustarse al sistema sobre el que está desplegado. Por ello, Apache Airflow tiene importantes aplicaciones en proyectos relacionados con el procesamiento, ingesta o manipulación de datos.

Apache Airflow ha sido utilizado en este proyecto para automatizar y orquestar el proceso de ingesta, procesado y almacenamiento de los Boletines Oficiales.

### PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales orientado a documentos de código abierto. Entre otras funcionalidades, PostgreSQL permite almacenar datos de manera estructurada en forma de tablas, para posteriormente acceder a esa información y poder consumirla de la forma que se desee.

En este proyecto PostgreSQL se ha utilizado tanto para almacenar la información generada por Apache Airflow como para guardar el resultado del proceso de extracción y procesado de Boletines Oficiales. Para visualizar y consumir directamente la información extraída de los Boletines se han creado diferentes vistas, detalladas en la sección 3.4.

Una vez se cumplen estos requisitos, el siguiente paso es realizar la creación del entorno, como se explica en el apartado 1 del Manual de explotación y mantenimiento.

# Desarrollos realizados

En esta sección se van a detallar los desarrollos realizados para el presente proyecto, que sigue el flujo indicado en la Ilustración 2.



Ilustración . Flujo de tratamiento de los artículos

## Ingesta de los datos

El primer paso del flujo de la información es la ingesta de artículos. Esta se ejecuta con el fichero ingesta.py, indicando el día del que se quieren obtener los documentos en formato ‘aaaammdd’ y el directorio base donde se van a descargar. En el directorio base el programa va a crear una estructura como la siguiente:

|  |
| --- |
| directorio\_base  └───dia\_indicado  ├───apertura  │ ├───html  │ ├───info  │ ├───pdf  │ │ └───rotados  │ └───xml  └───cierre  │ ├───html  │ ├───info  │ ├───pdf  │ │ └───rotados  │ └───xml |

En el directorio base **se crea una carpeta por día ingestado**, con dicho día como nombre. Dentro de este se guardan los ficheros XML que funcionan a modo de sumarios de los diferentes tipos de boletines, y se crean dos carpetas, apertura y cierre, donde se guardan los artículos de apertura de convocatorias y de cierres de las mismas, respectivamente. Ambas siguen una **estructura común**: dentro de ellas se crean las carpetas html, xml, pdf e info. Dentro de las tres primeras se almacenan los artículos en los respectivos formatos. En pdf, también se crea una carpeta llamada rotados, donde se almacenan los PDFs de los artículos del BOE rotados en 90º. Por su parte, en info se va a guardar un fichero XML por artículo tratado, donde se va a ir almacenando la información obtenida para los diferentes campos.

Todo esto se encarga de crearlo y completarlo el programa ingesta.py. La única condición es que el directorio base exista. Por otra parte, el programa descarga los artículos de los boletines cuyos ficheros de configuración estén en la carpeta ficheros\_configuracion, en el mismo directorio que la carpeta ingesta, con el nombre ‘fuente\_conf.xml’, siendo fuente el boletín. Si la fuente indicada es diferente a las cinco tratadas, tratará de realizar la ingesta extra. Si no, hará la ingesta con los ficheros ingesta\_aragon.py e ingesta\_boe.py.

### Ingesta de los artículos del BOE

La ingesta de los artículos del BOE se realiza con el programa ingesta\_boe.py y sigue el esquema de la Ilustración 3, que a continuación se va a detallar.

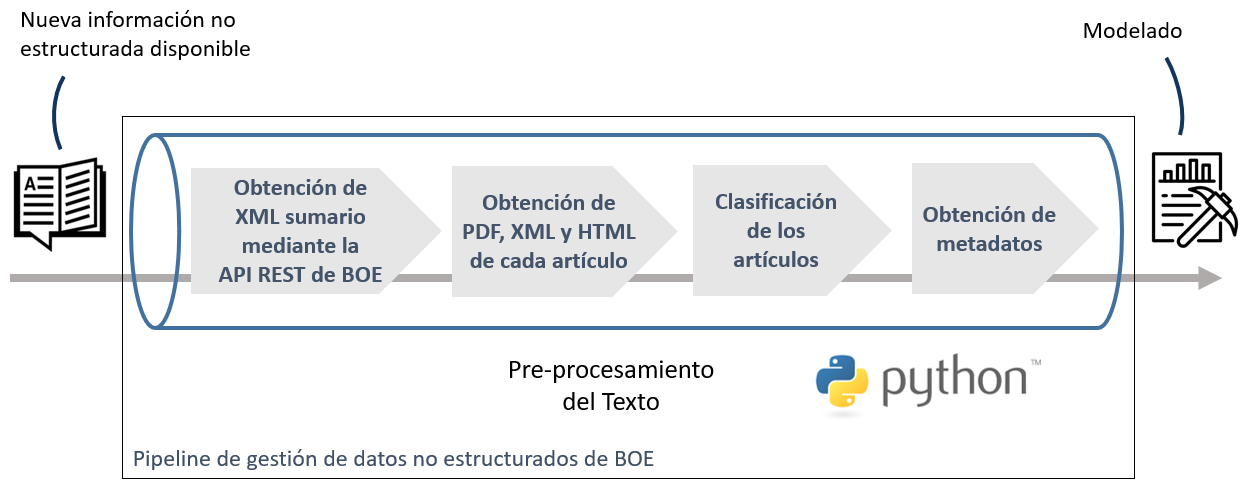


Ilustración . Pipeline de ingesta de artículos del BOE

El primer paso es obtener, mediante la API que proporciona el sitio web del BOE, el **XML sumario** con los enlaces a los artículos del día, divididos por secciones y departamentos. Este fichero se almacena por si se quiere consultar posteriormente. De este se extraen los sufijos de las **URLs** que llevan a los artículos, permitiendo su descarga. Sin embargo, previamente se consulta el título de cada artículo de las secciones indicadas. Los títulos sirven para, comparándolos con las cadenas definidas en el fichero de configuración auxiliar, decidir si cada artículo es una apertura de oferta, un cierre o nada (comprobando primero si no es de empleo, después si es de apertura y finalmente si es de cierre. Si no se encuentra ninguno de los tres, también es descartado). En el último caso, el artículo se descarta y no se descarga. En los otros dos, se descarga en el directorio correspondiente en sus diferentes formatos (incluido el PDF rotado). Para más información sobre estos primeros pasos, véase el apartado 2.2 del documento de validación del análisis exploratorio.

Para cada artículo no descartado, una vez se ha descargado en los distintos formatos se rellena el fichero de información con los metadatos interesantes que se obtienen del XML del artículo, indicados en el fichero de configuración en etiquetas\_xml/a\_guardar.

### Ingesta de los artículos de Aragón

La ingesta de los artículos de los boletines BOA, BOPH, BOPZ y BOPT se realiza con el programa ingesta\_aragon.py y sigue el esquema de la Ilustración 4, que a continuación se va a detallar.

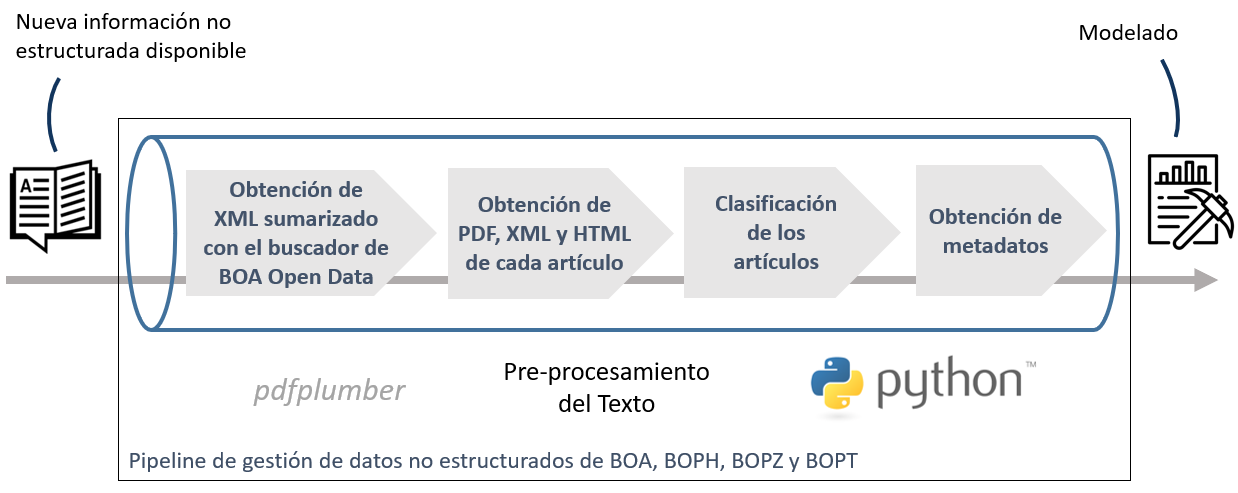


Ilustración . Pipeline de ingesta de artículos de boletines aragoneses

El esquema respecto al del BOE es el mismo, pero tiene algunas diferencias sustanciales. Por ejemplo, el BOE ofrece una API a la que se le indica únicamente el día y devuelve un XML, que es fijo, con los artículos de ese día. En cambio, para estos boletines, se ofrece simplemente un buscador que devuelve un XML con los artículos que cumplen las condiciones indicadas en los parámetros. A estos XML se les ha llamado **XML Sumarizados**, para diferenciarlos con los XML Sumario del BOE. Así, el primer paso es obtener los XML Sumarizados del día para cada tipo de boletín (y en el caso del BOA, para cada sección indicada en su fichero de configuración). Estos ficheros también se almacenan para su posterior consulta, por si el personal lo requiriera.

A continuación, en el caso del **BOA**, se lee el título para cada artículo y **se decide** si es un artículo de apertura, de cierre o de nada, descartando estos últimos, como ocurre con el BOE. En este caso, primero se comprueban las cadenas de no empleo, después comprueba si aparece la cadena ‘por la que se deja sin efecto’, que indica cierre, y a continuación apertura y cierre. Si no encuentra nada, se descarta también. En el caso de los provinciales, para esta decisión también se usa el texto completo, debido a que los títulos de estos no son tan descriptivos ni vienen completos. La lógica de los **BOPs** es más compleja, debido a las observaciones que se han realizado en los mismos, que varían mucho:

* Si encuentra, en el título alguna cadena que indique que no es de empleo específica de los BOPs o de las generales, o si encuentra en el texto alguna cadena que indique que no es de empleo específica de los BOPs, o si encuentra en el texto más de 5 cadenas que indiquen que no es de empleo de las generales: se descarta.
* Sino, si encuentra alguna cadena de cierre específica de los BOPs en el título: es de cierre.
* Sino, si encuentra más de una cadena de apertura específica de los BOPs en el título o encuentra ‘bases de la convocatoria’ en el texto: es de apertura.
* Sino, si encuentra una cadena de cierre específica de los BOPs en el primer tercio del texto: es de cierre.
* Sino, si se encuentra más de una cadena de apertura específica de los BOPs en el título o en el texto: es de apertura.
* Sino, se descarta.

Esta lógica se ha desarrollado calibrando y tratando de clasificar correctamente los artículos de provinciales vistos, muy variables en estructura y escritura.

El siguiente paso es **almacenar los artículos** con sus distintos formatos. Como no se ofrece el formato XML para cada artículo, se guarda el propio registro del XML Sumarizado como XML del artículo, en un fichero independiente. El pdf se obtiene a partir de uno de los campos del XML, donde se ofrece un enlace al mismo, que hay que procesar. Por su parte, el HTML se ha de formar en el código a partir de los parámetros anteriores y la posición del artículo en el XML Sumarizado. Para más información de la recuperación de todos estos documentos, véase el apartado 2.1.2.3 del documento de validación del análisis exploratorio.

Para cada artículo no descartado, una vez se ha descargado en los distintos formatos se rellena el fichero de información con los metadatos interesantes que se obtienen del XML del artículo, indicados en el fichero de configuración en etiquetas\_xml/a\_guardar.

### Ingesta extra

La ingesta extra, creada para los boletines no contemplados, sigue la misma estructura que las anteriores. Se ha tratado de generalizarlo lo máximo posible, por ello:

* La recuperación de un supuesto XML Sumario se realiza con una parametrización equivalente a la del BOE, donde solo sea necesario pasar el día a la API indicada en el fichero de configuración.
* Los registros en este XML Sumario se entiende que siguen una estructura coherente, indicada también en el fichero de configuración. Se ha contemplado también la posibilidad de secciones, ahí indicadas también.
* Las etiquetas a almacenar son las que aparezcan en el fichero de configuración, incluidas algunas con enlaces al documento en diferentes formatos, también indicados (si no encuentra una de XML, guarda la parte del registro de forma independiente, como con los boletines aragoneses). O el título o el texto han de poder recuperarse. Sino, el artículo se descarta ya que no se podría obtener nada. Se comprueba el tipo de artículo comprobando primero las cadenas de apertura y luego las de cierre.
* Se completa el fichero de información con las etiquetas indicadas en el fichero de configuración.

Es muy complicado que esta ingesta funcione de forma totalmente correcta sin revisiones ni un análisis previo de los boletines que se van a obtener, pero **se ha tratado de hacer lo más general posible**, dándole mucha importancia al fichero de configuración que el usuario crearía y completaría. Para más información, consúltese el apartado 2 del manual del usuario.

## Conversión de los documentos

La conversión de los documentos, referida a la transformación de los artículos en su formato inicial a texto plano, almacenado en un fichero de formato TXT, se realiza con el programa **conversion\_a\_texto.py**. Este programa se lanza con el directorio donde se encuentran las carpetas de apertura y cierre de las que se van a recuperar los documentos. También se le pueden pasar como argumentos la ruta al fichero auxiliar si este es movido de su directorio y un booleano que hace que el texto resultante aparezca cortado a los 500 caracteres, haciéndolo más legible (especialmente para el etiquetado posterior). El programa invoca al script correspondiente según el tipo de boletín tratado y el formato por defecto indicado en el fichero de configuración auxiliar. Por defecto, únicamente el BOPT lee del XML (porque en el PDF aparecen trozos de otros artículos), mientras que todos los demás lo obtienen de los PDF (ya que en XML y HTML algunas tablas y otra información importante no aparecen).

### De PDF a TXT

Para obtener el texto limpio de los PDFs se han estudiado concienzudamente para encontrar los puntos de corte. Estos son necesarios para evitar extraer textos como cabeceras, pies o información adicional que en ocasiones aparece en los laterales. Con estos puntos de corte es posible hacer una especie de recorte a cada página para extraer únicamente el texto que aparece dentro del mismo. De esta forma, el primer paso es crear esa ***bounding box***a partir de los puntos de corte indicados en el fichero de configuración correspondiente (en caso de no encontrarlos, se han colocado unos puntos por defecto, tratando de hacerlos generales). Luego, se extrae y concatena el texto obtenido de todas las páginas recortadas. Después, **se limpia** de forma general (como juntando palabras de diferentes líneas si la final termina con un renglón), y con algunos tratamientos específicos en función del tipo de boletín tratado. Finalmente, se guarda el texto. También se ha creado la lógica necesaria para obtener la fecha de publicación, que aparece en las cabeceras y se han estudiado sus puntos de corte y formato para cada tipo de boletín. Sin embargo, finalmente no se utiliza debido a que se obtiene como metadato en el paso anterior.

### De XML a TXT

Con esta forma, la recuperación del texto limpio es bastante sencilla:

* En el caso del BOE, el texto viene indicado por párrafos con etiquetas <p>, dentro de la etiqueta <texto>, por lo que basta con ir concatenando estos párrafos.
* En el caso de los boletines aragoneses, el texto viene en la etiqueta <texto>.

### De HTML a TXT

Con esta forma, la recuperación del texto limpio se realiza utilizando el paquete BeautifulSoup4. Basta utilizar la función get\_text() para recuperar todo el texto de la página, teniendo únicamente que observar los dos posibles casos para cortar el texto anterior y el posterior.

## Extracción de la información

En esta sección se van a detallar los desarrollos realizados para la extracción de la información de los textos, que se realiza con el fichero extraccion.py, donde se encuentran las invocaciones a los demás y el modelo jerárquico.

### Extracción mediante reglas

La extracción de información mediante reglas se realiza en el programa extraccion\_reglas.py, que es invocado por extraccion.py. Los **pasos** seguidos para ello son los siguientes:

* Primero, se leen algunos campos ya extraídos como metadatos como apoyo para la extracción de reglas. Por ejemplo, se lee si se ha obtenido el rango para evitar extraerla y para, en caso de que este sea una orden, saber si hay que extraer la ID de la orden.
* Después, se carga el modelo ‘es\_core\_news\_md’ de Spacy y se adapta con nlp() a los textos a tratar (título y texto del artículo). Esto lo que hace es adaptar los textos al modelo de noticias en español que tiene Spacy.
* Luego, se obtienen tanto los términos del tipo de la convocatoria y las expresiones regulares, todo ello del fichero regex.xml, ubicado en ficheros\_configuracion.
* A continuación, se aplican las expresiones regulares de los campos indicados. Algunas de ellas de forma específicas, otras de forma general (condición *else*). Algunos de los campos tienen un pequeño procesamiento, como las páginas web (hay que descartar páginas web inválidas, detectadas porque en ocasiones vienen con un espacio en blanco en medio de estas por el formato erróneo de los pdfs).
* Finalmente, el programa devuelve un diccionario con todos los textos detectados y las entidades a los que corresponden.

#### Expresiones regulares

En este apartado se van a detallar las expresiones regulares realizadas para los diferentes campos que detectan. Estas se almacenan en regex.xml, dentro de ficheros\_configuracion, haciendo muy sencilla su modificación en caso de requerirlo.

##### Cuerpo

La expresión del cuerpo contempla cualquier cuerpo que empiece por dicha palabra y termine en punto o coma. También le da importancia a “cuerpos docentes universitarios”, que aparece en múltiples ocasiones. La expresión completa es:

|  |
| --- |
| ([Cc]uerpo( de)? [a-zA-Zá-úÁ-Ú ]+|cuerpos docentes universitarios)(\.|\,) |

##### Escala

Al igual que con el cuerpo, se contemplan los que empiezan por la palabra escala, hasta que encuentra un carácter diferente de letra o espacio. La expresión es:

|  |
| --- |
| [Ee]scala [a-zA-Zá-úÁ-Ú ]+ |

##### Subescala

Equivalente a la escala, pero con la palabra subescala:

|  |
| --- |
| [Ss]ubescala [a-zA-Zá-úÁ-Ú ]+ |

##### Grupo

En el grupo se ha incluido la lógica conocida sobre el mismo, por lo que aparecen todos los grupos que pueden aparecer, tras las palabras grupo, subgrupo y ambas. También códigos de estatutarios:

|  |
| --- |
| ([ABCE]{1}[0-9]{3}|([Gg]rupo|[Ss]ubgrupo|[Gg]rupo\/[Ss]ubgrupo):? (A1|A2|A|B|C1|C2|C|E|GP1|GP2|GP3|GP4|GP5)\/?(A1|A2|A|B|C1|C2|C|E|GP1|GP2|GP3|GP4|GP5)?) |

##### Web

Para la página web se ha utilizado una *regex* habitual:

|  |
| --- |
| (http|www)[a-zA-Z\.\:\/\-\\_]+ |

##### Email

Para el correo electrónico también se ha utilizado una *regex* bastante habitual para estos casos:

|  |
| --- |
| [a-z0-9]+[\.\_]?[a-z0-9]+[@]\w+[.]\w{2,3} |

##### Plazo

El plazo tiene una lógica más complicada ya que, además de detectarlo bien, es necesario dividir la frase encontrada para entender sus partes con el objetivo de conseguir las fechas de presentación de solicitudes. Por ello, se ha dividido en cinco partes, que se concatenan para encontrar el plazo, pero se utilizan por separado para dividirlo:

|  |  |
| --- | --- |
| Num\_dias | ([a-zA-Z]+|[0-9]+|[a-zA-Z]+ y [a-zA-Z]+)( \([0-9]+\))? |
| Tipo\_dias | (días|DÍAS|día|DÍA|meses|MESES|mes|MES)( hábiles| HÁBILES| naturales| NATURALES| hábil| HÁBIL| natural| NATURAL)? |
| Contexto\_1 | \,? [a-zá ]\*(del|desde( el)?) |
| Dia\_inicio | (día siguiente|día|siguiente|la) |
| Contexto\_2 | [a-z ]+publicación |

##### Fecha\_Disposicion

Para la fecha de disposición se han realizado tres expresiones regulares diferentes. Para conocer la fecha de disposición del propio artículo se utilizan las aparecidas en reglas/apertura/fecha\_disposicion, teniendo en cuenta el rango del artículo:

|  |  |
| --- | --- |
| Resolución | de [0-9]+ de [a-z]+ de [0-9]+ |
| Orden | de [0-9]+ de [a-z]+ de [0-9]+ |

Sin embargo, para los artículos de cierre, donde interesa ver las fechas de disposición de las convocatorias de apertura, se ha creado otra expresión regular que las detecte, siendo más complicado por aparecer en medio del texto o del título:

|  |
| --- |
| (Resolución[a-zá-úA-ZÁ-Ú0-9, ]\*|Orden [A-Z]+\/[0-9]+\/[0-9]+,?) de [0-9]+ de [a-z]+( de [0-9]+|) |

##### ID\_Orden

Para los artículos que son órdenes, se ha realizado una regla que detecte el ID de las mismas:

|  |
| --- |
| (Orden|orden|ORDEN) [a-zA-Z]+\/[0-9]+\/[0-9]+ |

##### Puesto

A pesar de que el texto es habitual recogerlo de tablas o del NER, también se han realizado una serie de reglas para cuando el texto viene fijado o contextualizado por otras palabras, indicadas también a continuación:

|  |  |
| --- | --- |
| Denominación: puesto | (DENOMINACIÓN|Denominación|denominación|denominacion|IDENTIFICACIÓN|Identificación)((de la[s]?|del|de los)? (plaza[s]?|puesto[s]?))?:[a-zA-Zá-úÁ-Ú\/\,\“\”\"\' ]+(\.|\n|\\n) |
| Cuerpo, escala, puesto. | Cuerpo [a-zA-Zá-úÁ-Ú\/\, ]+, Escala [a-zA-Zá-úÁ-Ú\/\, ]+, [a-zA-Zá-úÁ-Ú\/ ]+\. |
| especialidad puesto. | especialidad [a-zA-Zá-úÁ-Ú\/ ]+\. |
| IDENTIFICACIÓN DE LA PLAZA: puesto. | IDENTIFICACIÓN DE LA PLAZA:[a-zA-Zá-úÁ-Ú\/ ]+\. |

##### Num\_Plazas

Para la identificación del número de plazas se recogen tanto las ocasiones en las que aparece etiquetado (“Número de plazas: 2”) y en las que no, aunque lo que aparezca sea “un puesto vacante”. También se detecta “bolsa de trabajo”, indicador de que no hay plazas.:

|  |
| --- |
| (BOLSA DE TRABAJO|Bolsa de trabajo|Bolsa de Trabajo|bolsa de trabajo|Número de plazas: ([a-zA-Z]+|[0-9]+)|(Una|una|[a-zA-Z]+ y [a-zA-Z]+|[a-zA-Z]+|[0-9]+) (plaza|puesto|vacante)[s]?( vacante[s]?)?) |

### Extracción mediante el modelo NER

Una vez se tiene entrenado el modelo NER, la extracción de campos es sencilla: hay que segmentar el texto del artículo (ya que los textos largos no los acepta) y **guardar todos los campos** **que encuentre** en los distintos segmentos. Esto lo realiza extraccion\_ner.py, que devuelve un diccionario con las entidades tras obtenerlas (y hacer un pequeño procesamiento a las escalas).

#### Entrenamiento del modelo NER

El entrenamiento del modelo [NER](https://spacy.io/usage/training#ner) (Reconocimiento de Entidades Nombradas), de la librería *open-source* [Spacy](https://spacy.io/), se llevó a cabo tras etiquetar manualmente 228 documentos, como se explica en el apartado 3 del plan de pruebas. En el apartado 4 del mismo documento se explican los **cambios realizados** a los ficheros de estas herramientas para el correcto funcionamiento del NER. Además, se cambiaron algunos de los **metadatos** que utiliza el entrenamiento como la tasa de *dropout* o el número de iteraciones, finalmente utilizadas 50 iteraciones, punto en el que ya ha olvidado las etiquetas que tenía el modelo “es\_core\_news\_md” predefinido y, además de aprender nuestras etiquetas, detecta satisfactoriamente los textos, algo que se nota especialmente en la reducción del tamaño, ya que con menos iteraciones los textos devueltos contienen contextos posteriores largos, que no interesan para los campos. Estos entrenamientos se realizaron etiquetando documentos hasta que las **métricas** del propio modelo fueron constantemente satisfactorias, **mostrando que se había evitado el *underfitting* y el *overfitting*** (apoyados también en diferentes evaluaciones con otros documentos con los que no se había entrenado). Las métricas para el modelo final, el entrenado con 228 documentos y 50 iteraciones, son:

|  |
| --- |
| Scores del modelo de 50 iteraciones, con un tiempo de 2356.44807100296 segundos:  >> La precision es 79.5774647887324, el recall es 69.90411382616765 y el f-score es 74.42779515890004  >> Métricas por entidades (propias):  web:  Precision: 85.97883597883597  Recall: 86.20689655172413  F-Score: 86.09271523178808  puesto:  Precision: 68.07909604519774  Recall: 53.08370044052864  F-Score: 59.65346534653465  tipo\_convocatoria:  Precision: 76.74858223062382  Recall: 59.443631039531475  F-Score: 66.996699669967  grupo:  Precision: 93.33333333333333  Recall: 89.74358974358975  F-Score: 91.50326797385621  escala:  Precision: 61.702127659574465  Recall: 57.08661417322835  F-Score: 59.304703476482615  cuerpo:  Precision: 73.10344827586206  Recall: 74.12587412587412  F-Score: 73.6111111111111  email:  Precision: 100.0  Recall: 97.67441860465115  F-Score: 98.8235294117647  num\_plazas:  Precision: 93.86189258312021  Recall: 95.07772020725389  F-Score: 94.46589446589448  plazo:  Precision: 94.27480916030534  Recall: 87.27915194346289  F-Score: 90.64220183486236 |

Para evaluar estos modelos se utilizaron, especialmente, documentos con los que no se había entrenado el modelo, como se comenta en el plan de pruebas.

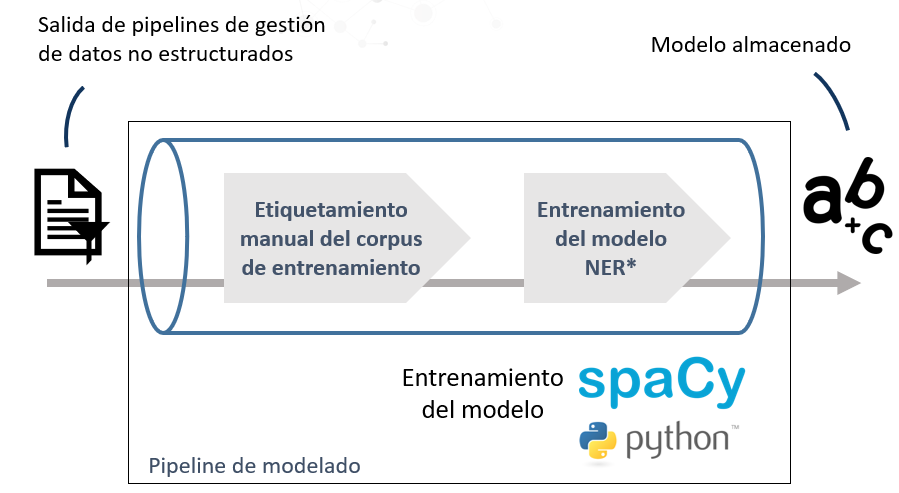


Ilustración . Pipeline de modelado

### Extracción de puestos de tablas de anexos

Las tablas que contienen múltiples puestos ofertados pueden aparecer en los anexos tanto del BOE como del BOA. La extracción de los mismos se realiza con el fichero extracción\_tablas.py, que utiliza también el programa puestos\_tablas\_anexos.py. El primero llama al segundo **tanto para el PDF normal como el rotado** (en el caso del BOE), y limpia las listas resultantes y decide cuál de las dos es la correcta, en caso de haber una correcta. El segundo es el que realiza la extracción propiamente dicha. Estos son los pasos que se sigue con cada página del fichero PDF indicado:

* Primero, se comprueba si hay una tabla en la página, y si esta contiene puestos. Esto se realiza consultando la cabecera, que puede ser la primera o la segunda fila de la tabla. Si en una de sus columnas aparece una de las cadenas consideradas en el fichero auxiliar, se entiende que es una tabla de ofertas de puestos y se procede a su extracción.
* Se guarda el índice de la columna que contiene una de esas cadenas, ya que será la columna donde aparezcan los puestos.
* Se recuperan las celdas de dicha columna, siempre y cuando la primera columna no esté vacía y contenga un número (que es el número de la orden). También se realiza una comprobación por si el texto de la celda es continuación del texto de la anterior, resultando en un único puesto.

**Se almacena toda la celda como puesto ofertado**, de forma que en ocasiones aparecerá más información de la debida, ya que de vez en cuando aparece el centro directivo del puesto actual o del siguiente en la propia celda. Anteriormente este texto se limpiaba, pero se han encontrado evidencias de tablas en las que no se pueden limpiar, ya que en ocasiones el centro directivo aparece encima del texto y en otras debajo. Además, en cuanto a formato cambia mucho (minúsculas o mayúsculas, terminando en punto o no, etc.)

### Modelo jerárquico

Este modelo es el que decide **de cuál de los extractores se va a recoger la información de cada campo**. Forma parte del fichero extraccion.py. Este programa es el que llama a los extractores, limpia las salidas y realiza estas decisiones. En términos generales, la función principal es evaluar\_articulo(), que primero realiza dichas invocaciones, luego limpia cada campo y decide con qué información quedarse, y finalmente escribe la información decidida en el fichero de información.

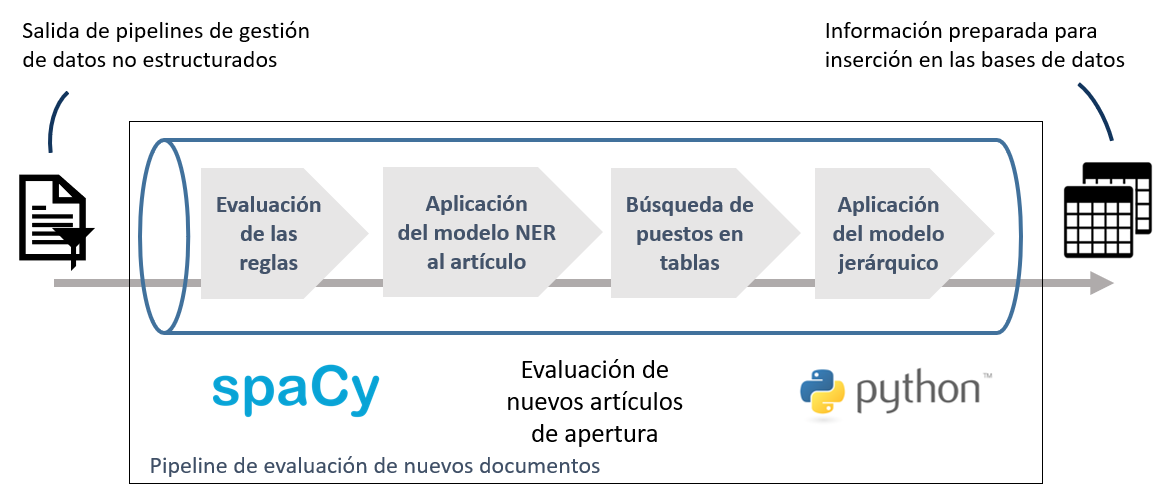


Ilustración . Pipeline de evaluación de nuevos artículos de apertura

Desglosando cada campo, la obtención de la información es como sigue:

* **Fuente de datos:** Metadatos.
* **Fecha de publicación:** Metadatos.
* **Enlace a la convocatoria:** Metadatos.
* **Órgano convocante:** Metadatos.
* **Título:** Metadatos.
* **URI ELI:** Metadatos.
* **Enlace al cierre:** Metadatos del cierre.
* **Estado:** Implícito en apertura y cierre.
* **Rango:** Metadatos.
* **ID\_Orden:**
  + Si no está en los metadatos y se ha conseguido alguno con las reglas, coger el primero.
  + Sino, ‘-’.
* **Fecha\_Disposicion:** 
  + Si no está en los metadatos y se ha conseguido alguno con las reglas, coger la primera.
  + Sino, se utiliza la fecha de publicación. Esto se hace especialmente por la no aparición de fechas de disposición en muchos boletines provinciales, y la necesidad de tener una para el id de oferta.
* **Plazo:**
  + Si hay en las reglas, coger el primero.
  + Si no y si hay en NER, coger el primero.
  + Si no, ‘-‘.
* **Fecha\_inicio\_presentación\_solicitudes/Fecha\_fin \_presentación\_solicitudes:**
  + Se entiende mediante control de patrones lo que indica el plazo para calcularlo, con ayuda de la fecha de publicación.
  + ‘-’ si plazo es ‘-’.
* **Tipo\_convocatoria:**
  + Si en el título, mediante las reglas, se ha detectado un término de “ambos” o términos de ambos: “Libre+Interna”
  + Si no, si en el título, mediante las reglas, se han detectado términos de uno: Ese.
  + Si no, si en el texto, mediante las reglas, se ha detectado un término de “ambos” o términos de ambos: “Libre+Interna”
  + Si no, si en el texto, mediante el NER, se ha detectado un término de “ambos” o términos de ambos: “Libre+Interna”
  + Si no, si en el texto, mediante las reglas, se ha detectado términos de uno: Ese
  + Si no, si en el texto, mediante el NER, se ha detectado términos de uno: Ese
  + Si no, “Libre”.
* **Datos\_contacto:**
  + Si hay webs con reglas, la primera.
  + Si no, si hay email con reglas, el primero.
  + Si no, si hay webs con NER, la primera.
  + Si no, si hay email con NER, el primero (por ser el campo menos etiquetado).
  + Si no, ‘-’.
* **Num\_plazas:**
  + Si en el título, mediante las reglas, se encuentra alguno, elegirlo (el primero).
  + Si no, elegir la ocurrencia más alta de entre las encontradas en el texto por reglas y/o NER, si esta es mayor o igual que el número de puestos. En caso contrario, elegir el número de puestos encontrados.
* **Cuerpo:**
  + Si el NER ha encontrado, coger el primero.
  + Si no, coger el primero de las reglas.
  + Si no, ‘-’.
* **Escala:**
  + Si el NER ha encontrado, coger la primera.
  + Si no, coger la primera de las reglas.
  + Si no, ‘-’.
* **Subescala:**
  + Si el NER ha encontrado, coger la primera.
  + Si no, coger la primera de las reglas.
  + Si no, ‘-’.
* **Puestos:**
  + Se cogen los puestos obtenidos de tablas de anexos.
  + Si no hay, se coge el primero detectado por reglas.
  + Si no, se coge el primero detectado con el NER.
  + Si no, coge la escala. (En almacenamiento.py)
  + Si no, coge el cuerpo. (En almacenamiento.py)
  + Si no, ‘-’.

La extracción de información de artículos de cierre es diferente, y de ella se encarga la función evaluar\_tablas\_cierre(), que busca puestos en tablas para cerrarlos. Si no encuentra, también busca en reglas y NER de la misma forma que en los de apertura.

## Inserción en la base de datos

El último paso para los artículos de apertura de ofertas es insertar a información obtenida en la base de datos. De esto se encarga el fichero almacenamiento.py, una vez están creadas y se tienen permisos para la inserción. La función principal de este programa es almacenar(). A ésta se le pasan principalmente la raíz del fichero de información tratado y la conexión abierta a la base de datos, que se abre en la función main(). La función obtiene primero todos los campos para después comprobar si el número de campos obtenido es mayor o igual al umbral indicado en el fichero auxiliar. Si no pasa la condición, se escribe el nombre del artículo (del tipo fuente\_dia\_serial) en el fichero ‘articulos\_no\_insertados.log’. En cambio, si la pasa, **se insertan las cuatro fechas y los puestos detectados si todavía no lo están, la convocatoria y, finalmente, una oferta por cada puesto obtenido.**

## Cierre de ofertas

En el caso de los artículos de cierre ofertas, el último paso es cerrar las ofertas abiertas cuyos campos coincidan. Esto se realiza en cierres\_convocatorias.py. El primer paso es recuperar fuente de datos, órgano convocante y fecha de disposición del artículo de cierre. Si en el cierre aparece uno de los términos de cierre sin puestos (como ‘dejar sin efecto’ o ‘desierta’), y hay alguna oferta abierta con los mismos atributos se cierra esa oferta. Si no aparece uno de ellos, pero hay ofertas abiertas con los mismos atributos, se recuperan de la base de datos todos los puestos de estas ofertas y se buscan en el texto. **Se cierran las ofertas cuyos puestos se encuentren en el texto**, a razón de una por cada ocurrencia. Cabe destacar que, cuando se encuentra un puesto en el texto, se elimina de este, con el objetivo de que un mismo trozo de texto no cierre más de una oferta. Al cerrar una oferta, su estado pasa a ‘Cerrada’, y enlace\_cierre pasa a ser la URL del artículo que la ha cerrado.

# Base de datos

En esta sección se va a explicar todo lo relacionado con la base de datos creada. Esta base de datos utiliza el sistema gestor de bases de datos PostgreSQL 9.6.12 y se le ha llamado empleo\_publico\_aragon. Para crearla, se ha ejecutado la siguiente secuencia en PostgreSQL:

*CREATE DATABASE empleo\_publico\_aragon;*

## Estructura

Una vez se cerró el análisis exploratorio completamente, se pasó a realizar el esquema de la base de datos. El primer paso para ello fue, estudiando todos los requerimientos y limitaciones que daban los campos, crear el **modelo E/R** (Entidad/Relación) que explicara cómo iba a ser luego la base de datos. Este modelo puede verse en la Ilustración 2. Cabe destacar que el color de los atributos tiene un carácter únicamente informativo: los atributos verdes fueron los que se contemplaban en el pliego como necesarios, los azules son los que se incorporaron para cumplir con las vistas mencionadas en este, y los amarillos son los que el cliente pidió a posteriori. Se procede a detallar las diferentes entidades y relaciones entre sí:

* **Puesto:** La entidad ‘puesto’ se ha formado con la denominación de este, su cuerpo, su escala y su subescala. De esta forma, cada combinación de estos cuatro atributos formará un puesto (indicado por un id). Cada puesto (dicha combinación) puede aparecer en N ofertas.
* **Fecha:** Cada fecha cuenta con su día, su mes, su año, el día de la semana (lunes, martes…), su formato textual (dd/mm/aaaa) y su id. Cada fecha puede aparecer en N convocatorias, con diferentes relaciones, que se detallan a continuación.
* **Convocatoria:** Entidad que se refiere a todo el artículo manejado. Incluye los atributos que son troncales a todas las ofertas en este incluidas. Se incluyen como opcionales, además de los atributos que se incluyeron como opcionales en el propio pliego (URI\_ELI y datos\_contacto), los que tienen naturaleza opcional (ID\_Orden) y los que no son siempre detectados (como el rango o el grupo). Sus cuatro atributos de fecha son las relaciones con dicha entidad. Cada convocatoria puede tener N ofertas.
* **Oferta:** Cuenta con un estado, que puede ser ‘Abierta’ o ‘Cerrada’ y, en este segundo caso, un enlace\_cierre. Cada oferta pertenece a una convocatoria y puede incluir un puesto.

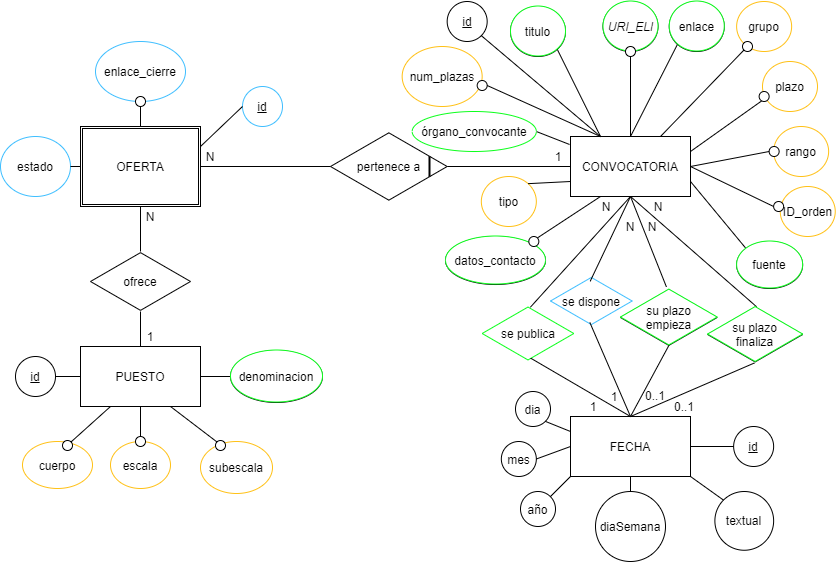


Ilustración . Modelo E/R de la base de datos

A partir de este modelo E/R se pasó al **esquema relacional**, que ya definía la estructura de las tablas de la base de datos. Este esquema puede verse en la Ilustración 3. A partir de este esquema y la opcionalidad indicada en el modelo anterior, se procedió a la creación de las tablas.

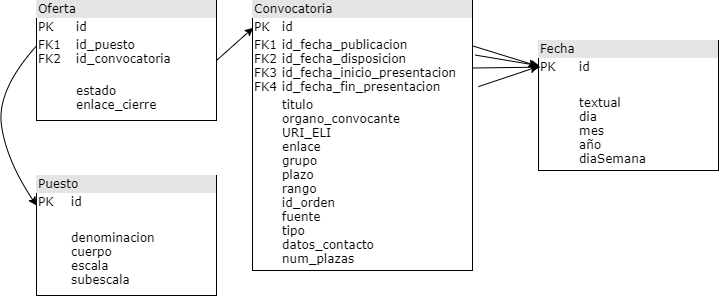


Ilustración . Esquema relacional de la base de datos

## Detalle de los campos

En esta sección se va a explicar brevemente qué contiene cada uno de los campos de la base de datos.

### Tabla Fecha

Los campos de la tabla fecha son los siguientes:

* **ID:** Es el identificador de la fecha. Para optimizar posibles joins y darle información real al id se ha decidido que éste no fuera un número secuencial, sino que contuviera la propia fecha que, al fin y al cabo, va a ser única en la base de datos. De esta forma, el id es la fecha que guarda en formato ‘aaaammdd’.
* **Textual:** Es un campo varchar que contiene la fecha indicada con formato ‘dd/mm/aaaa’.
* **Dia:** Contiene el día en forma numérica de la fecha.
* **Mes:** Contiene el mes en forma numérica de la fecha.
* **Año:** Contiene el año en forma numérica de la fecha.
* **DiaSemana:** Contiene el día de la semana, en forma textual y en español, de la fecha.

### Tabla Puesto

Los campos de la tabla puesto son:

* **ID:** Es el identificador artificial del puesto. Es un serial.
* **Denominación:** Es el propio puesto, en formato textual. En caso de no haberlo y haber escala, se incluye aquí la escala.
* **Cuerpo:** Es el cuerpo, en formato textual.
* **Escala:** Es la escala, en formato textual.
* **Subescala:** Es la subescala, en formato textual.

### Tabla Oferta

En cuanto a los campos de la tabla oferta:

* **ID:** Es el identificador de la oferta. Es textual y se compone de campos de otras tablas: ‘fuente\_organoconvocante\_denominacionpuesto\_num’, siendo num un secuencial propio que aumenta si los otros tres campos son equivalentes a los de otro registro de la base de datos.
* **Estado:** Puede ser ‘Abierta’ o ‘Cerrada’, indicando el estado de la oferta.
* **Enlace\_cierre:** Cuando la convocatoria está abierta tiene valor nulo. Cuando se cierra, contiene la URL del artículo que cerró dicha oferta.
* **Id\_puesto:** FK del id del puesto ofertado.
* **Id\_convocatoria:** FK de la convocatoria a la que pertenece la oferta.

### Tabla Convocatoria

Los atributos de la tabla convocatoria son los siguientes:

* **ID:** Es el identificador artificial de la convocatoria. Es un serial.
* **Titulo:** Es el título de la convocatoria.
* **Organo\_convocante:** Contiene el órgano que realizó la convocatoria.
* **URI\_ELI:** Contiene, en caso de existir, la URI ELI que identifica a la convocatoria de forma unívoca.
* **Enlace:** Contiene la URL al artículo de la convocatoria.
* **Grupo:** Contiene, en formato textual, los grupos profesionales indicados en la convocatoria como requisitos.
* **Plazo:** Contiene, en formato textual, el plazo de presentación de solicitudes indicado en la convocatoria.
* **Rango:** Es el rango de la convocatoria, especialmente ‘Resolución’ y ‘Orden’.
* **Id\_orden:** Es la ID de la orden de la convocatoria, en caso de que su rango lo sea, en formato textual.
* **Fuente:** Indica la fuente de los datos (el boletín del que se extrajo el artículo).
* **Tipo:** Indica si la convocatoria ofrece plazas de tipo ‘Libre’, de tipo ‘Interna’ o de ambas (‘Libre+Interna’).
* **Datos\_contacto:** Contiene los datos de contacto de la convocatoria (sitios web o correos electrónicos), en caso de que los haya.
* **Num\_plazas:** Indica el número de plazas totales ofertados en la convocatoria.
* **Id\_fecha\_publicacion:** FK de la fecha en la que se publicó la convocatoria.
* **Id\_fecha\_disposicion:** FK de la fecha de disposición de la convocatoria.
* **Id\_fecha\_inicio\_presentacion:** FK de la fecha de inicio de presentación de solicitudes.
* **Id\_fecha\_fin\_presentacion:** FK de la fecha de fin de presentación de solicitudes.

## Creación de las tablas

En esta sección se van a mostrar las creaciones realizadas para las diferentes tablas, con toda la información ya decidida en los anteriores apartados.

Para crear la tabla fecha:

|  |
| --- |
| *CREATE TABLE FECHA (*  *id int PRIMARY KEY,*  *dia int NOT NULL,*  *mes int NOT NULL,*  *año int NOT NULL,*  *diaSemana varchar NOT NULL,*  *textual varchar NOT NULL*  *);* |

Para crear la tabla convocatoria:

|  |
| --- |
| *CREATE TABLE CONVOCATORIA (*  *id serial PRIMARY KEY,*  *organo\_convocante varchar NOT NULL,*  *titulo varchar NOT NULL,*  *URI\_ELI varchar,*  *enlace varchar NOT NULL,*  *grupo varchar,*  *plazo varchar,*  *rango varchar,*  *id\_orden varchar,*  *fuente varchar NOT NULL,*  *tipo varchar NOT NULL,*  *datos\_contacto varchar,*  *num\_plazas int,*  *id\_fecha\_publicacion int NOT NULL,*  *id\_fecha\_disposicion int NOT NULL,*  *id\_fecha\_inicio\_presentacion int,*  *id\_fecha\_fin\_presentacion int,*  *CONSTRAINT fk\_fecha\_publicacion*  *FOREIGN KEY (id\_fecha\_publicacion) REFERENCES FECHA (id),*  *CONSTRAINT fk\_fecha\_disposicion*  *FOREIGN KEY (id\_fecha\_disposicion) REFERENCES FECHA (id),*  *CONSTRAINT fk\_fecha\_inicio\_presentacion*  *FOREIGN KEY (id\_fecha\_inicio\_presentacion) REFERENCES FECHA (id),*  *CONSTRAINT fk\_fecha\_fin\_presentacion*  *FOREIGN KEY (id\_fecha\_fin\_presentacion) REFERENCES FECHA (id)*  *);* |

Para crear la tabla puesto:

|  |
| --- |
| *CREATE TABLE PUESTO (*  *id serial PRIMARY KEY,*  *denominacion varchar NOT NULL,*  *cuerpo varchar,*  *escala varchar,*  *subescala varchar*  *);* |

Finalmente, para crear la tabla oferta:

|  |
| --- |
| *CREATE TABLE OFERTA (*  *id varchar PRIMARY KEY,*  *enlace\_cierre varchar,*  *estado varchar,*  *id\_convocatoria int NOT NULL,*  *id\_puesto int NOT NULL,*  *CONSTRAINT fk\_convocatoria*  *FOREIGN KEY (id\_convocatoria) REFERENCES CONVOCATORIA (id),*  *CONSTRAINT fk\_puesto*  *FOREIGN KEY (id\_puesto) REFERENCES PUESTO (id));* |

## Vistas

El último paso fue crear las vistas requeridas en el pliego. Para ello, se tuvieron en cuenta, además de con qué condiciones debían estar enfocadas las vistas, qué atributos mostrar en cada una de ellas.

Así, se creó la vista **‘ofertas\_en\_plazo\_presentacion’**, que muestra todas las ofertas que, a fecha actual, están en plazo de presentación de solicitudes (es decir, que la fecha de inicio es pasada o actual y la fecha de fin es actual o futura):

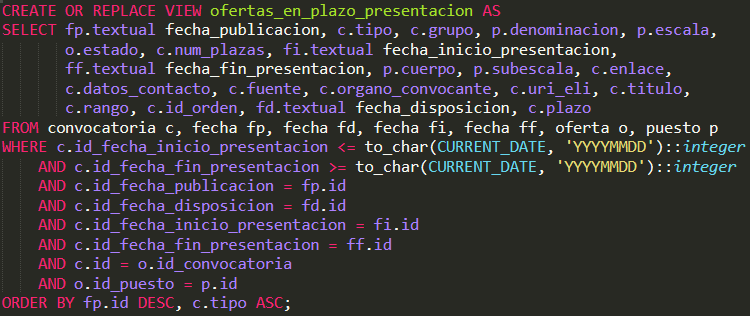


Ilustración . Vista ofertas\_en\_plazo\_presentacion

La siguiente vista creada fue **‘ofertas\_abiertas’**, que muestra las ofertas con estado ‘Abierta’:

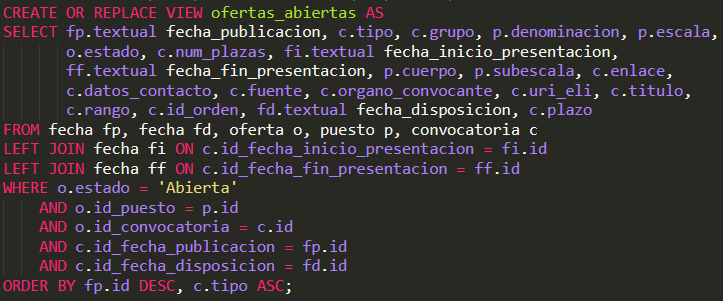


Ilustración . Vista ofertas\_abiertas

Por último, se creó la vista **‘ofertas\_historicas’**, que muestra todas las ofertas, abiertas o cerradas, cuya fecha de publicación fuera igual o posterior al 1 de enero de 2020:

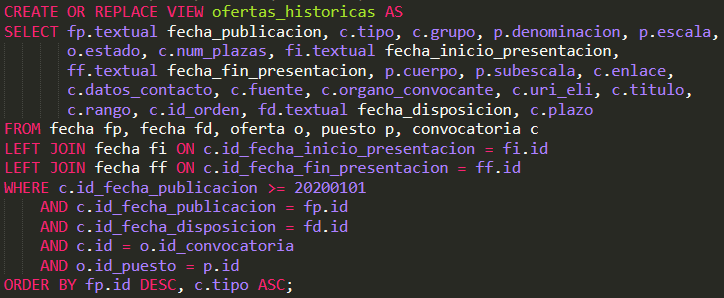


Ilustración . Vista ofertas\_historicas