Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Aplicación para análisis estadístico de titulares de periódico

Autor:
D. Samuel Garrido Cuesta

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Aplicación para análisis estadístico de titulares de periódico

Autor:

D. Samuel Garrido Cuesta

Tutor:

D. Quiliano Isaac Moro Sancho

Contenido

1. Introducción

1.1. Objetivos

El objetivo de este trabajo es elaborar una aplicación que permita realizar estadísticas sencillas sobre titulares de periódicos. La aplicación constará de dos partes fundamentales, cada una de las cuales proporcionará distinta funcionalidad:

- Una herramienta para recolectar datos de artículos de páginas web de periódicos, que utilice técnicas de *web scraping* o similares para analizar el HTML de las páginas, y que permita almacenar la información obtenida en una base de datos.
- Una herramienta para realizar estadísticas sobre el conjunto de datos recolectados, que permita visualizarlas mediante diagramas y que cuente con una interfaz gráfica de usuario.

El trabajo abarcará varias fases del desarrollo de software:

- Planificación inicial
- Análisis de software
- Diseño de software
- Diseño de bases de datos
- Implementación
- Testeo del software

El presente informe documenta la realización del trabajo y proporciona una descripción del software desarrollado. Se incluye también un apartado de anexos en el que poder consultar el manual de usuario de la aplicación, la forma de instalarla, o los archivos incluidos en el proyecto.

1.2. Planificación

En esta sección se describe la planificación inicial ideada para completar el trabajo en 300 horas. El desarrollo del proyecto se ha dividido en cinco etapas, las cuales no tienen por qué cumplirse de forma rigurosa, pero darán una idea general del transcurso del proyecto. Las etapas son las siguientes:

- Análisis de software
    - Duración: 20 horas

    - El objetivo de esta etapa es definir la funcionalidad de la aplicación. Se elaborará una descripción relativamente detallada de los objetivos que se pretenden conseguir y de la forma en que funcionará la aplicación y se realizará el análisis de requisitos del sistema y de casos de uso.

- Diseño de software
    - Duración: 20 horas
    - El objetivo de esta etapa es elaborar el diseño de la arquitectura del sistema y de los casos de uso. El resultado será un modelo de la arquitectura y un modelo de los casos de uso identificados en la etapa anterior; todo ello quedará plasmado en un conjunto de diagramas UML.
- Búsqueda de herramientas
    - Duración: 5 horas
    - El objetivo de esta etapa es buscar las herramientas apropiadas para implementar la aplicación y elegir a qué plataformas o sistemas operativos se quiere dar soporte.
- Creación del software
    - Duración: 245 horas
    - El objetivo de esta etapa es implementar el programa en base a la descripción realizada. El resultado será una aplicación funcional que cumpla los objetivos y requisitos descritos.
- Creación del manual
    - Duración: 10 horas
    - El objetivo de esta etapa es elaborar el manual de la aplicación y crear las herramientas necesarias para poder instalarla en las distintas plataformas.

La creación del software se llevará a cabo de una forma similar a la metodología ágil (*agile software development*). Este tipo de metodologías emplean un enfoque iterativo en el que la funcionalidad del producto se va ampliando y refinando de forma gradual. En cada iteración se incorporan al sistema algunos de los requisitos, obteniendo como resultado un producto funcional, el cual será limitado en las primeras iteraciones pero cada vez más completo a medida que éstas transcurran. Los requisitos del sistema y el diseño elaborados en las dos primeras etapas no son definitivos, y podrán evolucionar a medida que se desarrolla el software. Este enfoque de trabajo se opone a los métodos estructurados, en los que se sigue una planificación rígida, el producto final se construye desde el primer momento con el objetivo de ajustarse a unos requisitos y un diseño estrictos, y se hace mucho énfasis en la elaboración de documentación extensiva.

La etapa de creación del software constará de un número indefinido de iteraciones. Cada iteración tendrá una duración indeterminada e implicará las siguientes tareas:

- Añadir, modificar o mejorar los requisitos y/o el diseño
- Elegir los requisitos a implementar
- Implementar los requisitos
- Testear el código añadido
- Arreglar posibles bugs

Trabajando de esta forma se garantiza que, si los objetivos establecidos no son realistas y el tiempo no alcanza para cumplirlos todos, la aplicación desarrollada será completamente funcional, aunque no cumpla con todos los requisitos que se plantearon inicialmente.

El enfoque de trabajo descrito presenta algunas similitudes con el método Scrum, sin embargo en ningún momento se pretende seguir de forma rigurosa una metodología en concreto. Las metodologías ágiles están enfocadas al ámbito empresarial y al trabajo en equipo, nada de lo cual se aplica a este proyecto. Lo que aquí se describe es la forma de trabajo individual que es más cómoda para mí.

Como último punto cabe mencionar que, debido a la situación excepcional en la que se ha desarrollado el trabajo —estado de confinamiento por cuarentena y suspensión de las clases presenciales— la comunicación con el tutor se ha llevado a cabo de forma telemática, por email y videoconferencia.

1.3. Costes

A continuación se incluye una estimación de los costes requeridos para completar el proyecto.

- Equipo de trabajo. Para desarrollar el software se deberá contar con un equipo de trabajo. Se utilizará un ordenador portátil de gama media, con un precio aproximado de 400 €.
- Conexión a internet. La conexión a internet será necesaria para poder descargar las herramientas y librerías requeridas y consultar la documentación online. Suponiendo una jornada de trabajo de 3 o 4 horas diarias, las 300 horas del proyecto se completarían en aproximadamente 3 meses. Para un precio del ADSL de 25 € al mes, la conexión a internet costaría 75 €.
- Sueldo del ingeniero. Para un sueldo promedio en el ámbito de la informática de 29000 € al año o 16 €/h, el salario del ingeniero por 300 horas de trabajo sería de 4800 €.

Equipo de trabajo	400
Conexión a internet	75
Sueldo del ingeniero	4800
Total	**5275 €**

2. Análisis

2.1. Descripción

La funcionalidad básica de la aplicación consistirá en recuperar el titular y la fecha y hora de publicación de los artículos publicados en el servidor web de uno o varios periódicos, almacenar esta información en una base de datos o algún sistema de almacenamiento similar y proporcionar una interfaz gráfica de usuario que permita realizar estadísticas sencillas sobre el conjunto de los datos recopilados y visualizarlas mediante diagramas.

Para realizar la búsqueda de artículos en un periódico se hará uso de la sección de últimas noticias. Se asume que esta sección cumple una serie de condiciones:

- Está presente en todos los periódicos
- Contiene un número fijo de artículos
- Los artículos son los más recientemente publicados
- Los artículos contienen, directa o indirectamente, los datos requeridos

En el último punto, cuando se habla de que los artículos contienen los datos requeridos, se está haciendo referencia a que los artículos en la sección —entendiendo como artículo el elemento que represente uno— contienen, o bien el titular y la fecha de publicación, o bien un enlace a otra página donde encontrarlos.

El proceso de búsqueda consistirá de un bucle en el que en cada iteración se consultarán los artículos presentes en ese momento en la sección de últimas noticias. El ritmo de publicación de artículos es obviamente inferior a la velocidad de trabajo de un ordenador, por lo que en cada iteración se obtendrán seguramente varios artículos repetidos. Estos artículos deberán ser discriminados para no almacenar información redundante. Si se hace uso de una base de datos relacional podemos obviar este aspecto, ya que si se elige como clave primaria el titular o la fecha de publicación del artículo, a la hora de realizar las inserciones los artículos con clave primaria repetida se desecharían. La funcionalidad de discriminación sin embargo se incluirá en la propia aplicación, para no sobrecargar al gestor de bases de datos y a la red.

La forma de dirscriminar será la siguiente. Al terminar cada iteración, la fecha del artículo más reciente consultado se guardará en un archivo log. Este archivo será leído al comenzar cada iteración, y en caso de recopilar artículos anteriores, estos se desecharán.

La recolección de los datos se realizará mediante técnicas de *web scraping*. El código HTML de las páginas se procesará en busca de los elementos que contengan la información relevante. Algunos periódicos puede que dispongan de una API REST mediante la cual consultar sus artículos, lo cual

constituye un método de consulta más sencillo que realizar web scraping. Aunque la aplicación no utilizará APIs REST, esta característica se tendrá en cuenta para posibles ampliaciones futuras, por lo que la parte dedicada a la recolección de datos tendrá un diseño libre que permita adaptarse lo mejor posible a la naturaleza de cada periódico.

La realización de estadísticas se realizará desde una interfaz gráfica de usuario. Esta interfaz funcionará de forma independiente a la parte de recolección y podrá ejecutarse desde otro equipo. Las estadísticas podrán realizarse sobre uno o varios periódicos, y el conjunto de artículos a consultar podrá restringirse mediante su fecha de publicación introduciendo un rango de fechas. El usuario seleccionará todos estos parámetros mediante controles gráficos o *widgets*. La interfaz deberá ser lo más sencilla posible y estar despejada, de forma que el énfasis recaiga en los diagramas.

Algunos parámetros de la aplicación, tales como los periódicos que consultar a la hora de recolectar artículos o la base de datos a la que conectarse, deberán poder ser elegidos por el usuario; por ello la aplicación deberá contar con un sistema de configuración personalizable. La parte de recolección utilizará el enfoque tradicional Unix: la configuración se almacenará en un archivo de texto, el cual deberá ser editado antes de cada ejecución si se quiere modificar algún parámetro. La configuración relativa a la parte de estadísticas también se almacenará en un archivo de texto, pero esta podrá ser editada desde la propia interfaz gráfica, por lo que el archivo podrá ser modificado durante la propia ejecución.

El formato de los archivos de configuración no es crítico, y la única condición que se impone es que la semántica sea sencilla. Para desarrollar el proyecto se ha elegido TOML, un lenguaje minimalista y bien estandarizado, pensado específicamente para archivos de configuración. Está basado en elementos clave-valor y es muy similar al lenguaje INI, utilizado tradicionalmente en sistemas Linux y Windows. La aplicación se regirá por la versión 1.0.0 del lenguaje, que a fecha de la elaboración de este informe es la última especificación publicada.

2.1.1. Estadísticas

Para la realización del trabajo se han definido tres estadísticas, las cuales recibirán los nombres que se indican a continuación:

- Búsqueda de palabras clave
- Longitud media del titular
- Ranking de palabras

La estadística *búsqueda de palabras clave* obtendrá el número de titulares que contienen una o varias palabras clave. Podrá aplicarse sobre uno o varios periódicos y restringiendo los artículos a

aquellos publicados en un determinado intervalo de fechas. El método de visualización de esta estadística será mediante un diagrama de barras.

La estadística *longitud media del titular* calculará la media aritmética del tamaño de los titulares, entendiendo como tamaño el número de palabras que contienen. Podrá aplicarse sobre uno o varios periódicos y restringiendo los artículos a aquellos publicados en un determinado intervalo de fechas. El método de visualización de esta estadística será mediante un diagrama de barras.

La estadística *ranking de palabras* obtendrá las palabras más utilizadas en los titulares y mostrará el número total de apariciones de cada una. Podrá aplicarse sobre uno o varios periódicos y restringiendo los artículos a aquellos publicados en un determinado intervalo de fechas. El método de visualización de esta estadística será mediante un diagrama de barras.

La estadística *ranking de palabras* no se aplicará sobre todas las palabras presentes en los titulares, sino que se considerará un conjunto de palabras irrelevantes las cuales no serán tenidas en cuenta a la hora de realizarla. Los titulares de los artículos se almacenarán íntegramente en la base de datos, por lo que la aplicación deberá filtrarlos antes de realizar la estadística. Se considerarán palabras irrelevantes las siguientes categorías gramaticales:

- Preposiciones
- Pronombres
- Determinantes
    - Artículos
    - Demostrativos
    - Posesivos
    - Interrogativos/exclamativos
- Adverbios
    - Interrogativos/exclamativos
- Predeterminantes
- Conjunciones
- Contracciones

Adicionalmente, algunos adverbios de tiempo, lugar, modo, negación y duda también se considerarán palabras irrelevantes. El usuario deberá poder añadir o retirar las palabras que considere oportunas de esta lista.

La combinación de periódicos, el rango de fechas y cualquier otro parámetro relativo a las estadísticas será seleccionado por el usuario a través de la interfaz gráfica. La unidad mínima en la escala de fechas serán los días, por lo que no podrán especificarse horas, minutos ni segundos. Para realizar las estadísticas *ranking de palabras* y *búsqueda de palabras clave* no podrá distinguirse entre mayúsculas y minúsculas, de manera que las palabras “Hola” y “HOLA” se contabilicen como la misma.

Las estadísticas *longitud media del titular* y *búsqueda de palabras clave* podrán realizarse agrupando por fechas o agrupando por periódicos. Si se agrupa por fechas, la estadística *longitud media del titular* calculará el tamaño del titular de todos los periódicos juntos para cada día dentro del rango de fechas. Si se agrupa por periódicos, calculará la longitud del titular de cada periódico teniendo en cuenta los artículos publicados dentro del rango de fechas. Las estadística *búsqueda de palabras clave* funcionará de forma análoga. En los diagramas de barras utilizados para visualizar estas estadísticas, cada barra representará:

- Un día, si se agrupa por fechas
- Un periódico, si se agrupa por periódicos

En las figuras 1 y 2 se ejemplifica el comportamiento esperado de los diagramas dependiendo del modo de agrupamiento.

Figura 1. Diagrama de barras agrupando por fechas

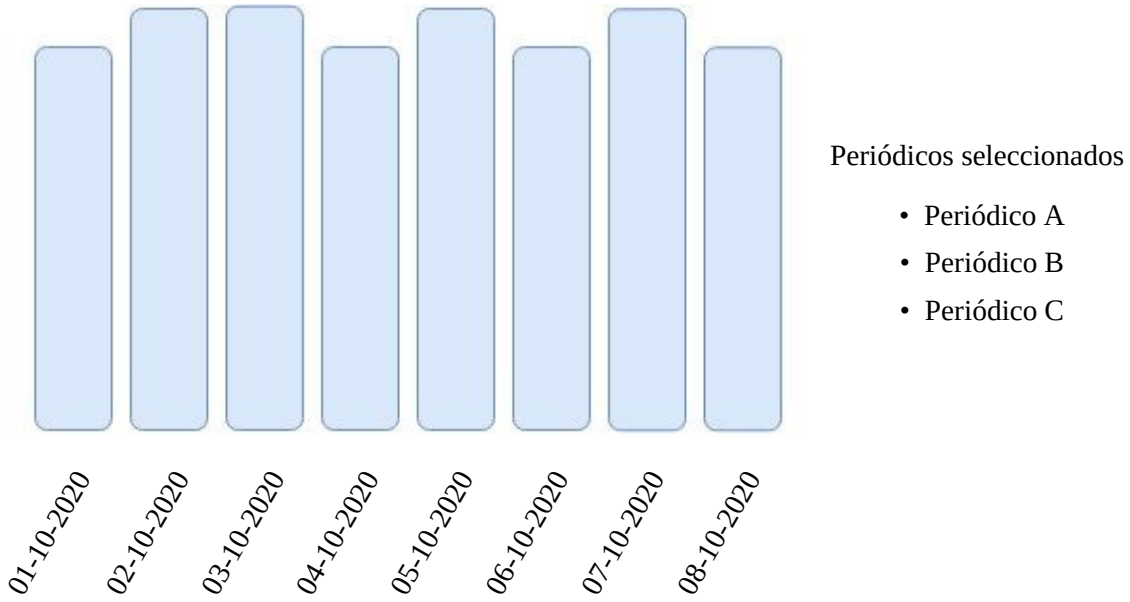
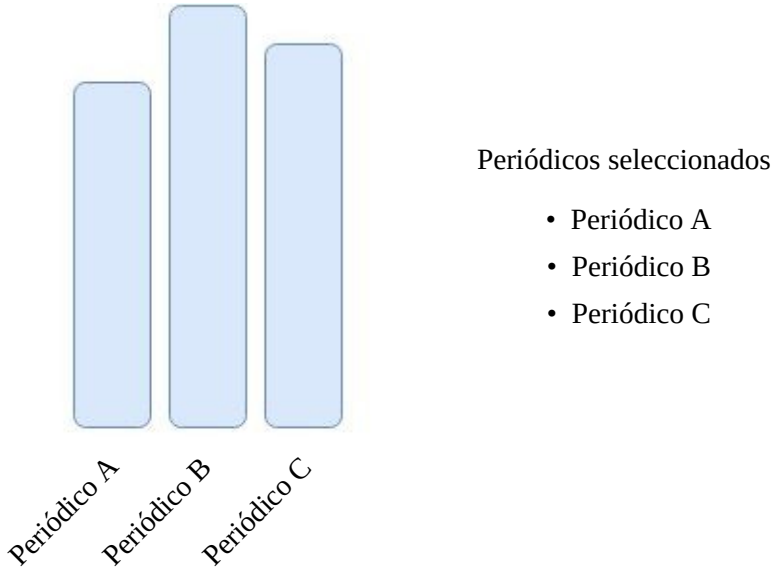


Figura 2. Diagrama de barras agrupando por periódicos



La estadística *ranking de palabras* podrá realizarse de forma global o agrupando por periódicos. Si se agrupa por periódicos, se mostrarán las palabras más usadas para cada uno de los periódicos. Si se realiza de forma global, se mostrarán las palabras más usadas para todos los periódicos juntos —todo esto teniendo en cuenta el rango de fechas—. En las figuras 3 y 4 se ejemplifica el comportamiento esperado de los diagramas para esta estadística.

Figura 3. Diagrama *ranking de palabras* agrupando por periódicos

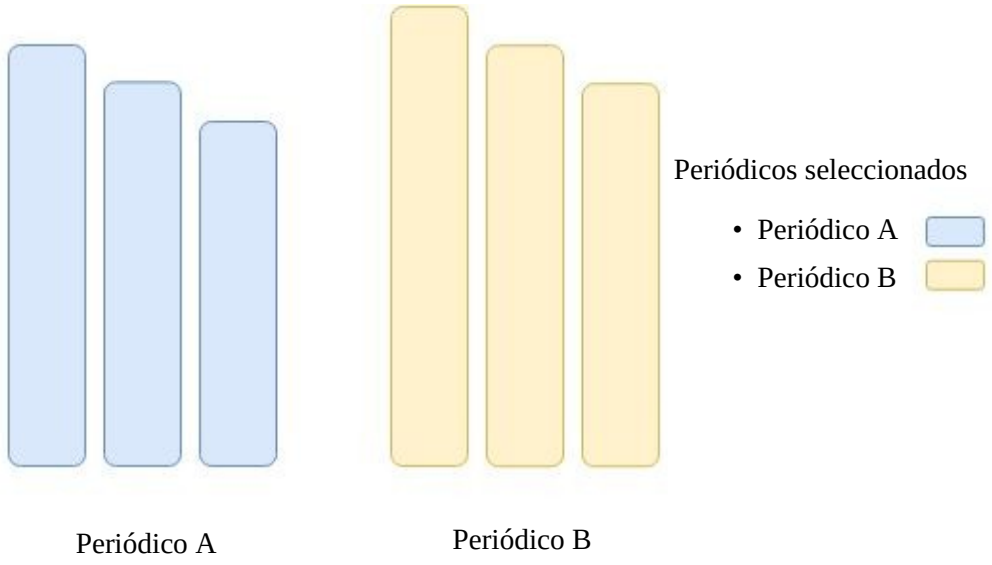
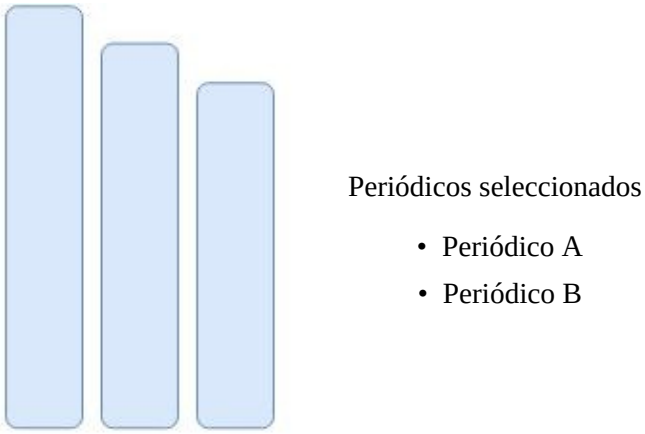


Figura 4. Diagrama *ranking de palabras* global



2.1.2. Periódicos

Los siguientes periódicos han sido seleccionados para realizar el proyecto:

- El País
- El Mundo

Ambos cuentan con una sección de últimas noticias que cumple las condiciones especificadas en la sección 2.1. El proyecto se ajustará al formato de HTML empleado a fecha de elaboración de este informe. Este formato podría cambiar en un futuro, quedando el software inutilizado.

2.2. Requisitos funcionales

En las tablas 1 y 2 se muestra la versión final de los requisitos funcionales de la aplicación, elaborados a partir de la descripción inicial del sistema realizada en la sección 2.1.

Tabla 1. Requisitos funcionales de la parte de recolección

1	El sistema deberá recopilar artículos de páginas web de periódicos
2	El sistema deberá almacenar los artículos en una base de datos
3	El sistema deberá guardar el titular de los artículos

4	El sistema deberá guardar la fecha de publicación de los artículos
5	El nombre de la base de datos deberá ser configurable
6	La dirección de la base de datos deberá ser configurable
7	El puerto de la base de datos deberá ser configurable
8	La cuenta para acceder a la base de datos deberá ser configurable
9	La contraseña para acceder a la base de datos deberá ser configurable
10	Los periódicos a consultar deberán ser configurables
11	El sistema deberá leer la configuración de un archivo de texto
12	El archivo de configuración deberá utilizar formato TOML v1.0
13	El sistema deberá permitir elegir el archivo de configuración a utilizar
14	El sistema deberá utilizar un archivo de configuración por defecto si este no se especifica
15	El sistema deberá notificar un error y abortar si la configuración no se puede leer
16	El sistema deberá notificar un error y abortar si la configuración es errónea
17	El sistema deberá funcionar de forma no interactiva
18	El sistema deberá funcionar de forma indefinida una vez ejecutado
19	El sistema deberá poder ser detenido sin afectar a la recolección
20	El sistema deberá notificar los errores por salida de comandos

Tabla 2. Requisitos funcionales de la parte de estadísticas

21	El sistema deberá manejarse mediante una interfaz gráfica de usuario (GUI)
22	El sistema deberá permitir realizar las estadística *longitud media del titular*
23	El sistema deberá permitir realizar la estadística *ranking de palabras*

24	El sistema deberá permitir realizar la estadística *búsqueda de palabras clave*
25	El sistema deberá visualizar las estadísticas mediante diagramas
26	El sistema deberá permitir seleccionar los periódicos para cada estadística
27	El sistema deberá permitir ajustar el rango de fechas para cada estadística
28	El sistema deberá permitir ajustar los parámetros específicos de cada estadística
29	La dirección de la base de datos deberá ser configurable
30	El puerto de la base de datos deberá ser configurable
31	El nombre de la base de datos deberá ser configurable
32	La cuenta para acceder a la base de datos deberá ser configurable
33	La contraseña para acceder a la base de datos deberá ser configurable
34	El sistema deberá almacenar la configuración en un archivo de texto
35	El archivo de configuración deberá utilizar formato TOML v1.0
36	El sistema deberá notificar un error si la configuración no se puede leer
37	El sistema deberá notificar un error si la configuración es inválida
38	El sistema deberá notificar un error si las consultas en la base de datos fallan
39	El sistema deberá permitir editar la configuración mediante la GUI

2.3. Requisitos no funcionales

En la tabla 3 se muestran los requisitos no funcionales de la aplicación, obtenidos a partir de la descripción inicial del sistema realizada en la sección 2.1.

Tabla 3. Requisitos no funcionales del sistema

1	El sistema deberá contar con una GUI sencilla

2	El sistema deberá contar con una GUI que haga énfasis en los diagramas
3	El sistema deberá contar con una GUI que responda de forma fluída

2.4. Casos de uso

En esta sección se describen los casos de uso del sistema. Los casos de uso se aplican únicamente a la parte de estadísticas, ya que la parte de recolección se ejecuta de forma no interactiva.

Caso de uso “Búsqueda de palabras clave”

1. El usuario selecciona los periódicos
2. El usuario introduce el rango de fechas
3. El usuario selecciona la estadística “Búsqueda de palabras clave”
4. El usuario introduce las palabras clave
5. El usuario selecciona el modo de agrupamiento
6. El usuario selecciona la opción “Consultar estadística”
7. El sistema realiza la consulta en la base de datos
    7.1. Si la consulta falla, el sistema informa de ello y el caso de uso termina
8. El sistema realiza las estadística y muestra los resultados en un diagrama

Caso de uso “Longitud media del titular”

1. El usuario selecciona los periódicos
2. El usuario introduce el rango de fechas
3. El usuario selecciona la estadística “Longitud media del titular”
4. El usuario selecciona el modo de agrupamiento
5. El usuario selecciona la opción “Consultar estadística”
6. El sistema realiza la consulta en la base de datos
    6.1. Si la consulta falla, el sistema informa de ello y el caso de uso termina
7. El sistema realiza la estadística y muestra los resultados en un diagrama

Caso de uso “Ranking de palabras”

1. El usuario selecciona los periódicos
2. El usuario introduce el rango de fechas
3. El usuario selecciona la estadística “Ranking de palabras”
4. El usuario selecciona el modo de agrupamiento
5. El usuario selecciona la opción “Consultar estadística”
6. El sistema realiza la consulta en la base de datos
    6.1. Si la consulta falla, el sistema informa de ello y el caso de uso termina
7. El sistema realiza la estadística y muestra los resultados en un diagrama

Caso de uso “Editar configuración”

1. El usuario selecciona la opción “Editar configuración”
2. El sistema muestra la ventana de configuración
3. El usuario introduce los parámetros
    3.1. Si el usuario selecciona “Cancelar”, el caso de uso termina
    3.2. Si el usuario selecciona “Aceptar”
        3.2.1. El sistema actualiza la configuración
        3.2.2. El sistema escribe la configuración en el archivo
        3.2.3. El sistema crea una nueva conexión con la base de datos
            3.2.3.1. Si la conexión no se realiza con éxito, el sistema informa de ello

3. Diseño de la base de datos

El sistema de almacenamiento de la aplicación consistirá en una base de datos relacional. La base de datos albergará dos tipos de entidades diferentes; una se corresponderá con los periódicos a consultar y otra con los artículos recopilados. La información que deberá recoger cada una de estas entidades queda reflejada en la tabla 4.

Tabla 4. Entidades de la base de datos

Entidad	**Información**
Periódico	Nombre
Artículo	Titular
	Periódico al que pertenece
	Fecha y hora de publicación
	Número de palabras
	Número de caracteres

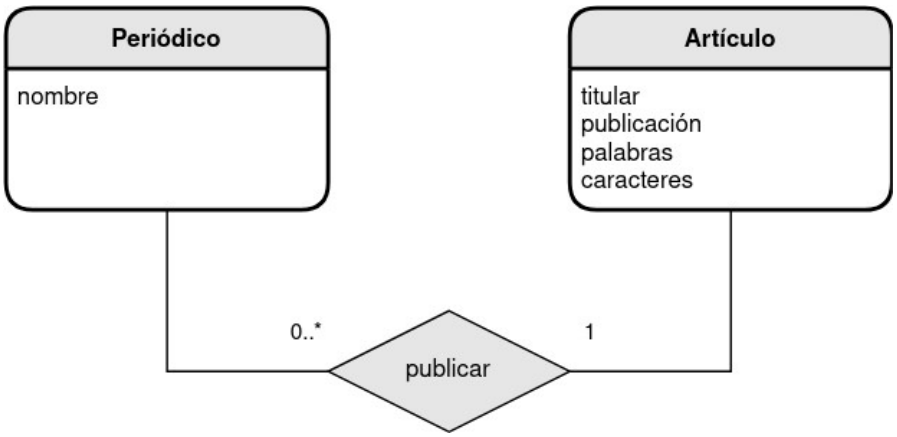
De entre toda esta información se identifica una relación, cuyas características se muestran en la tabla 5.

Tabla 5. Relaciones de la base de datos

Relación	**Entidad**	**Cardinalidad**
Publicar	Artículo	1..1
	Periódico	0..1

Toda esta información se plasma en un modelo relacional. El diagrama de entidad-relación para el modelo se muestra en la figura 5.

Figura 5. Diagrama de entidad-relación de la base de datos



La base de datos contendrá dos tablas, una para cada entidad. La relación entre ambas entidades se implementará mediante una clave foránea. Para ahorrar espacio de almacenamiento la tabla Periódico contará con un campo adicional, denominado identificador, que consistirá en un número entero. Este campo será al que haga referencia la clave foránea de los artículos. La estructura de la base de datos se muestra en la figura 6.

Figura 6. Estructura de la base de datos

periódico (identificador, nombre)

artículo (periódico, titular, publicación, palabras, caracteres)

4. Diseño software

4.1. Arquitectura

La aplicación está formada por tres sistemas de funcionalidad diferente, organizados en una arquitectura por capas en la que algunos sistemas dependen de otros. Los sistemas serán referenciados en este documento mediante los siguientes nombres:

- Sistema de recolección
- Sistema de estadísticas
- Sistema de almacenamiento

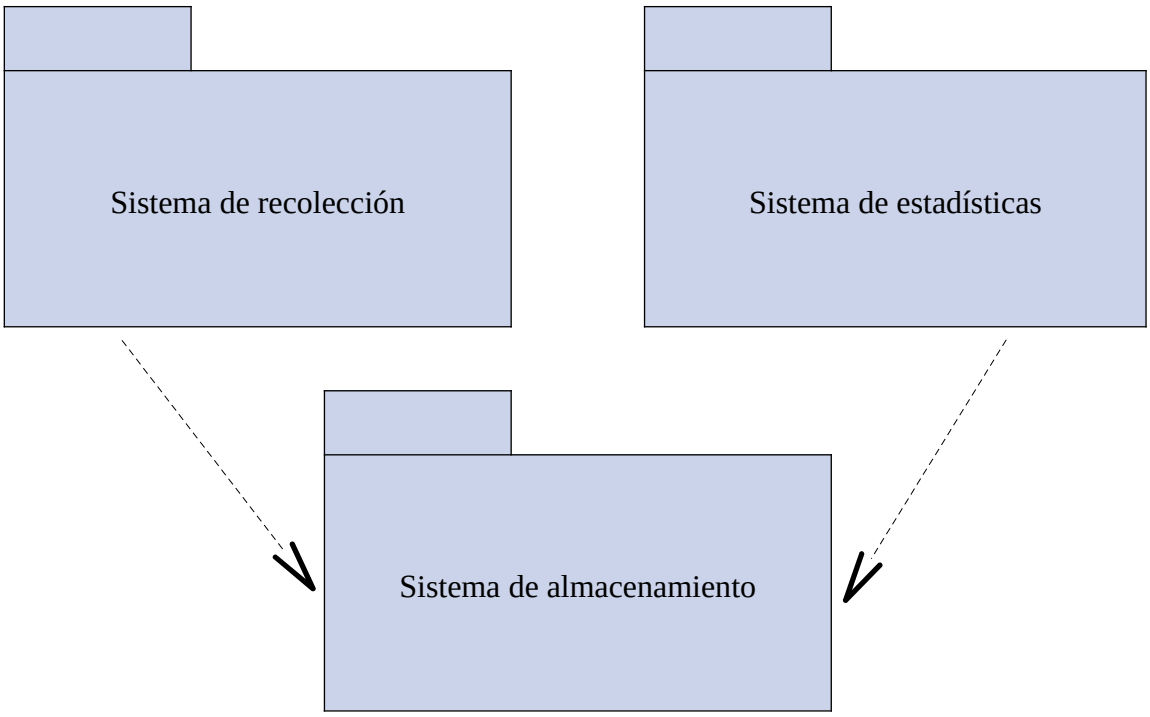
El sistema de almacenamiento constituye el método de acceso a la base de datos, y de él dependen los otros dos sistemas. Se encarga de gestionar las comunicaciones con el sistema gestor de bases de datos (SGBD) y actúa como una abstracción de él para las otras dos capas, proporcionando una interfaz que oculte los detalles irrelevantes y que permita realizar consultas o inserciones de forma sencilla, sin necesidad de conocer las especificaciones o el lenguaje de manipulación de datos. Las modificaciones en la estructura de la base de datos o el cambio a un SGBD diferente deberán de ser transparentes para las otras dos capas, siendo el sistema de almacenamiento el único que deba sufrir cambios.

El sistema de recolección se encarga de recopilar los artículos. Su cometido será obtener los datos pertinentes de las páginas web de los periódicos e insertarlos en la base de datos; esto último lo hará por medio del sistema de almacenamiento. El sistema de recolección funciona de forma independiente al sistema de estadísticas y está pensado para ser ejecutado como un proceso en segundo plano o *daemon*, por lo que no requiere de interacción con el usuario más allá de ser iniciado o detenido.

El sistema de estadísticas es el que permite realizar y visualizar las estadísticas definidas en la sección 2.1.1. El sistema de estadísticas se encarga de la interación con el usuario, la cual se realizará por medio de una interfaz gráfica de usuario (GUI). A la hora de realizar las estadísticas, obtendrá los datos pertinentes por medio del sistema de almacenamiento.

El diagrama de paquetes de alto nivel de la aplicación se muestra en la figura 7.

Figura 7. Diagrama de paquetes de la aplicación



En las siguientes secciones se describe el diseño adoptado para cada uno de los tres sistemas descritos. Las clases en este apartado hacen referencia a clases UML; es importante recalcar que estas no tienen por que ser implementadas mediante clases de programación orientada a objetos. De la misma forma, los métodos aquí descritos deberán aparecer obligatoriamente en la aplicación final, pero esta podrá contar con métodos o funciones adicionales a fin de poder realizar unit testing o mejorar la legibilidad del código fuente.

4.1.1. Sistema de almacenamiento

El sistema de almacenamiento está formado por una única clase denominada Database. Cada instancia de la clase encapsula una conexión con la base de datos, y contiene todos los parámetros y la lógica necesarios para realizar las diferentes consultas e inserciones requeridas por los otros dos sistemas.

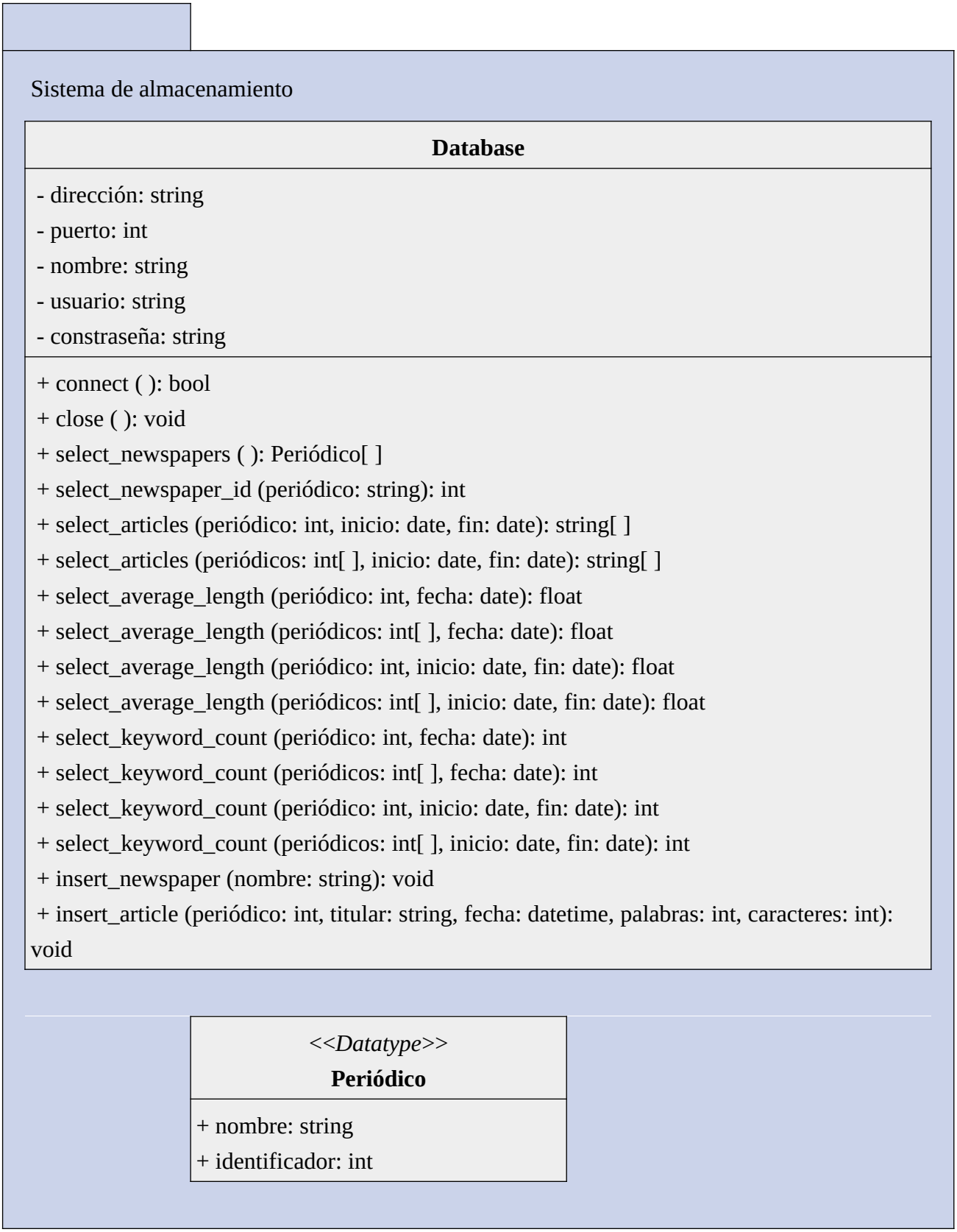
La clase Database contiene los siguientes métodos:

- connect Establece la conexión con la base de datos.
- close Cierra la conexión con la base de datos.
- insert_newspaper Inserta un periódico.
- insert_article Inserta un artículo.

- select_newspaper_id — Devuelve el identificador asociado a un periódico en la base de datos.
- select_newspapers — Devuelve el nombre y el identificador de todos los periódicos presentes en la base de datos.
- select_articles — Devuelve los titulares de los artículos de uno o varios periódicos presentes en la base de datos publicados durante un intervalo determinado de fechas.
- select_average_length — Devuelve la longitud media del titular de los artículos publicados en una fecha o un intervalo de fechas determinado de uno o varios periódicos.
- select_keyword_count — Devuelve el número de artículos de uno o varios periódicos publicados en una fecha o un intervalo de fehcas determinado cuyos titulares contienen una o varias palabras clave.

El diagrama de clases completo para el sistema de almacenamiento se muestra en la figura 8.

Figura 8. Diagrama de clases para el sistema de almacenamiento



4.1.2. Sistema de recolección

El sistema de recolección deberá procesar páginas web de distintos periódicos, cada uno de las cuales utilizará un formato de HTML distinto para sus artículos. Algo que se pretende evitar es el tener que modificar el sistema completo cada vez que se añada un nuevo periódico o que la estructura HTML de alguna de las páginas cambie. Por este motivo el sistema deberá presentar obligatoriamente un diseño modular. Por cada periódico que se quiera consultar existirá una instancia de una clase que se encargue de recopilar artículos única y exclusivamente de ese periódico.

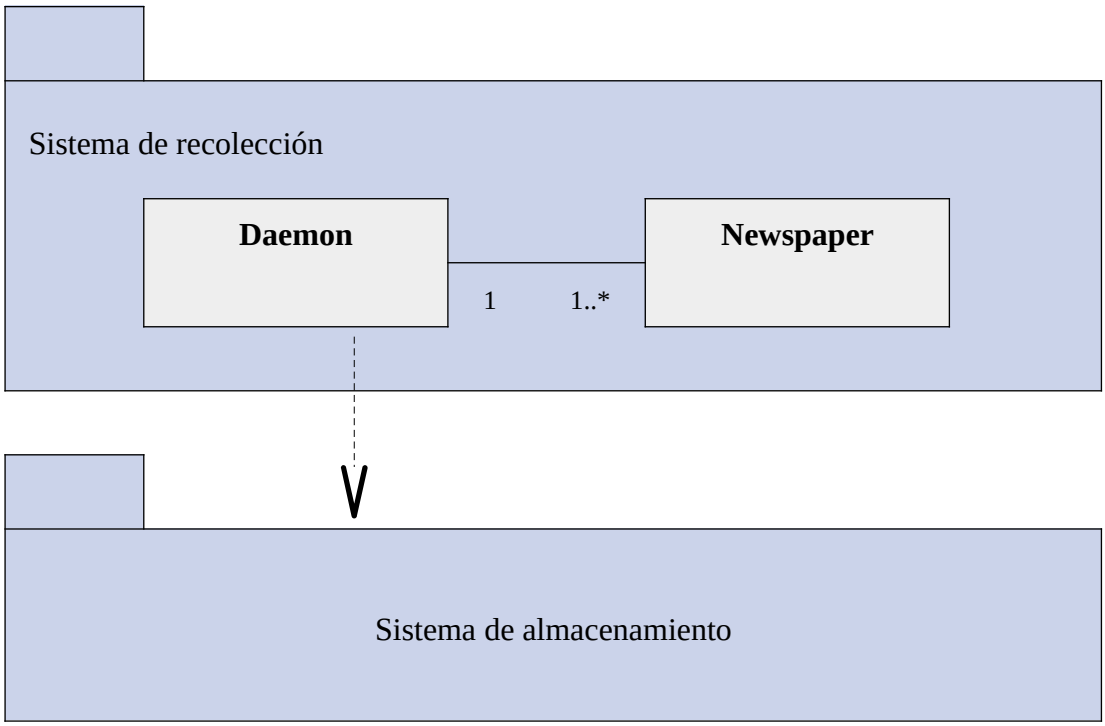
Para facilitar la incorporación de nuevos periódicos el sistema se diseñará mediante una arquitectura de plugins, de forma que de todas las clases disponibles se carguen de forma dinámica sólo aquellas asociadas a los periódicos que el usuario haya seleccionado antes de la ejecución del programa. Este enfoque permitirá añadir o retirar periódicos del programa simplemente moviendo archivos a un directorio concreto, sin necesidad de modificar el código fuente de la aplicación.

El sistema de recolección consta de una clase principal, denominada Daemon, y una clase Newspaper que actúa como un plugin. Los plugins se encargan de recopilar los artículos de la sección de últimas noticias de sus respectivos periódicos; la clase principal se encarga de ejecutar los plugins, validar y procesar los datos recopilados y almacenarlos en la base de datos. Los plugins funcionan de forma independiente a la clase principal y no tienen ningún tipo de conociemiento sobre esta.

La clase principal hará uso de los métodos proporcionados por la clase Database del sistema de almacenamiento para realizar las consultas e inserciones oportunas en la base de datos.

Una primera aproximación del diagrama de clases para el sistema de recolección se muestra en la figura 9.

Figura 9. Diagrama de clases inicial para el sistema de recolección



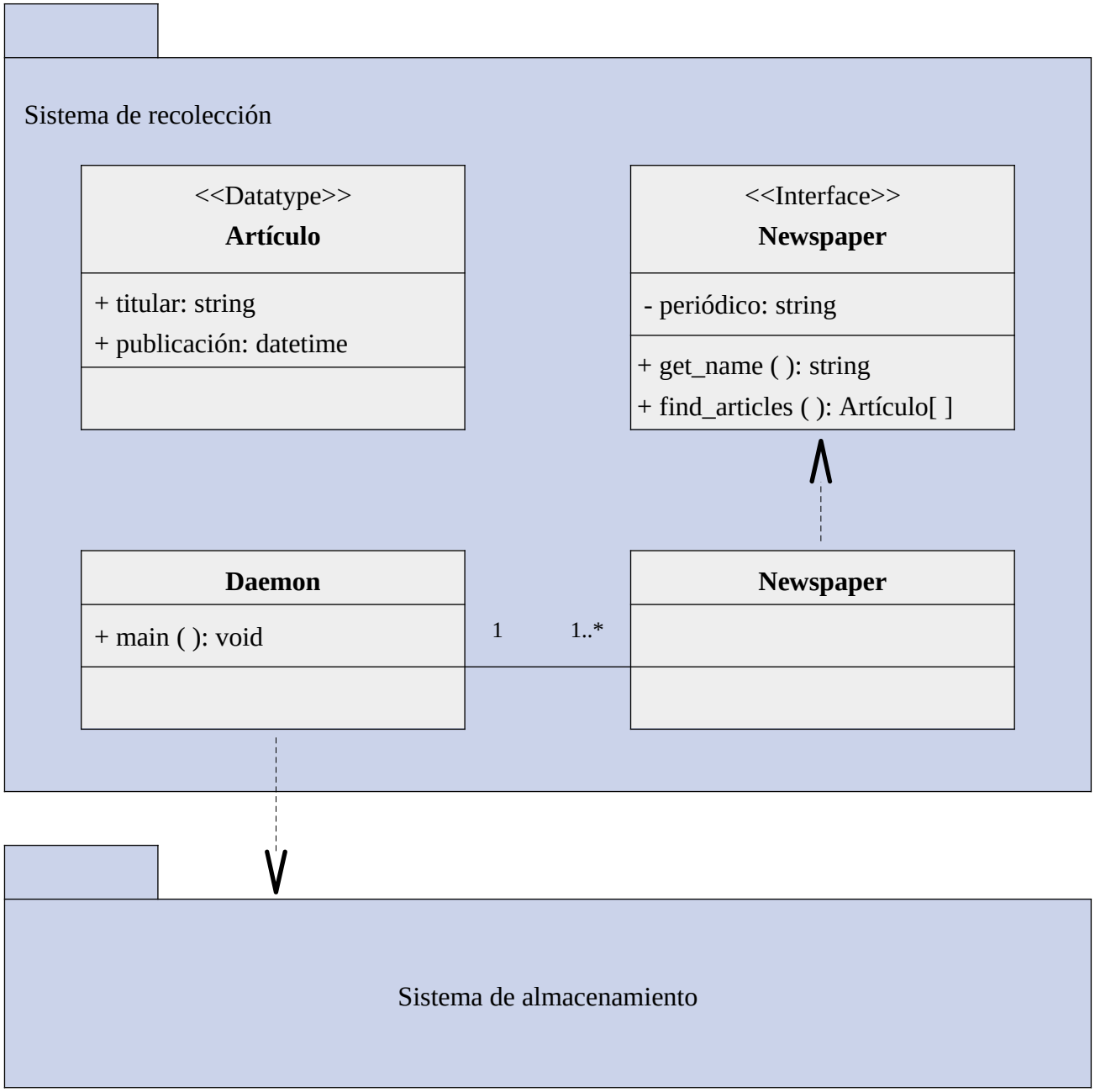
El diseño de los plugins estará sometido al menor número de restricciones posible, de forma que la implementación sea relativamente libre y cada plugin pueda adaptarse lo mejor posible a las características de su periódico. La clase Plugin contiene los siguientes métodos:

- get_newspaper Devuelve el nombre del periódico asociado a él
- find_articles Devuelve los artículos de la sección de últimas noticias

Dado que los plugins funcionan de forma independiente y no necesitan comunicarse en ningún momento con la clase principal, esta no requiere de ningún método más allá de aquel que actúe como punto de entrada. Por tanto, la clase Daemon contiene únicamente el método main.

El diagrama de clases completo para el sistema de recolección se muestra en la figura 10.

Figura 10. Diagrama de clases para el sistema de recolección

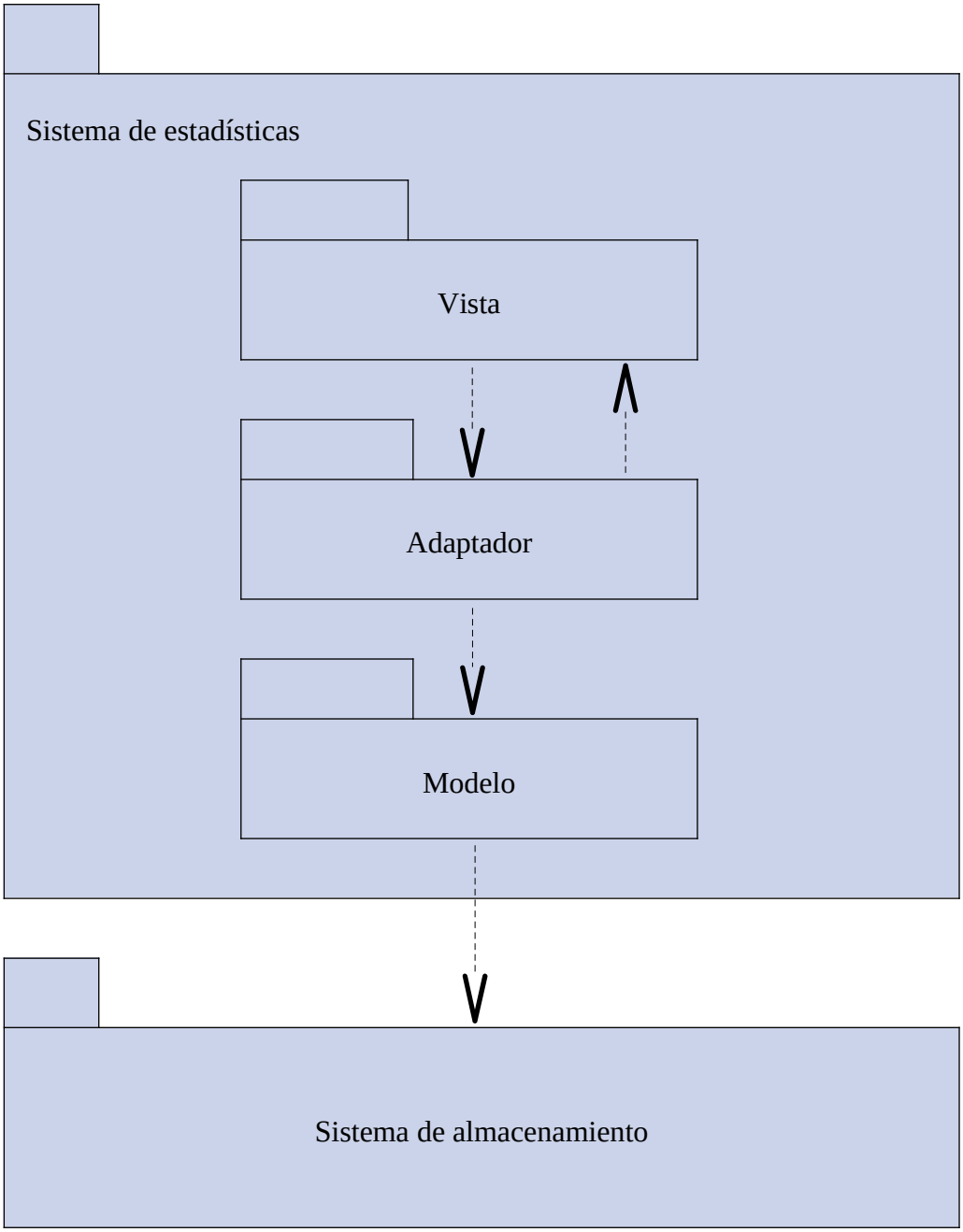


4.1.3. Sistema de estadísticas

El sistema de estadísticas utiliza el patrón Modelo-Vista-Adaptador (MVA). Este patrón está pensado para facilitar el mantenimiento de grandes aplicaciones que cuenten con una interfaz gráfica de usuario, separando la funcionalidad relativa a la interfaz con la funcionalidad relativa a los datos, de manera que los cambios en uno no afecten al otro. El patrón MVA está formado por tres capas: modelo, vista y adaptador. La vista se encarga de gestionar la interfaz gráfica de usuario, el modelo se encarga de la manipulación de los datos, y el adaptador se encarga de gestionar la lógica de la aplicación, actuando de intermediario entre la vista y el modelo.

En la figura 11 se muestra el diagrama de paquetes de alto nivel para el sistema de estadísticas, reflejando la organización por capas del patrón MVA.

Figura 11. Diagrama de paquetes para el sistema de estadísticas



El modelo se encarga de gestionar el acceso a los datos. Cuenta con métodos para llevar a cabo las consultas e inserciones requeridas por esta parte de la aplicación, proporcionando los datos en la forma adecuada para el adaptador.

La vista se encarga de gestionar la interfaz gráfica, y será la capa con la que interactúe el usuario. Su cometido es construir los componentes gráficos o *widgets* de la interfaz, obtener los datos de entrada del usuario y visualizar los resultados obtenidos del modelo en forma de diagramas.

La vista no contiene ningún tipo de lógica, y se encarga únicamente de la funcionalidad relacionada con los propios widgets. Cualquier acción del usuario que escape a este ámbito será relegada al adaptador para que sea él quien la gestione.

El adaptador proporciona la lógica de la aplicación y actúa de intermediario entre la vista y el modelo. El adaptador se encarga de procesar los eventos generados en la vista, generar las estadísticas a partir de los datos proporcionados por el modelo y comunicar los resultados a la vista.

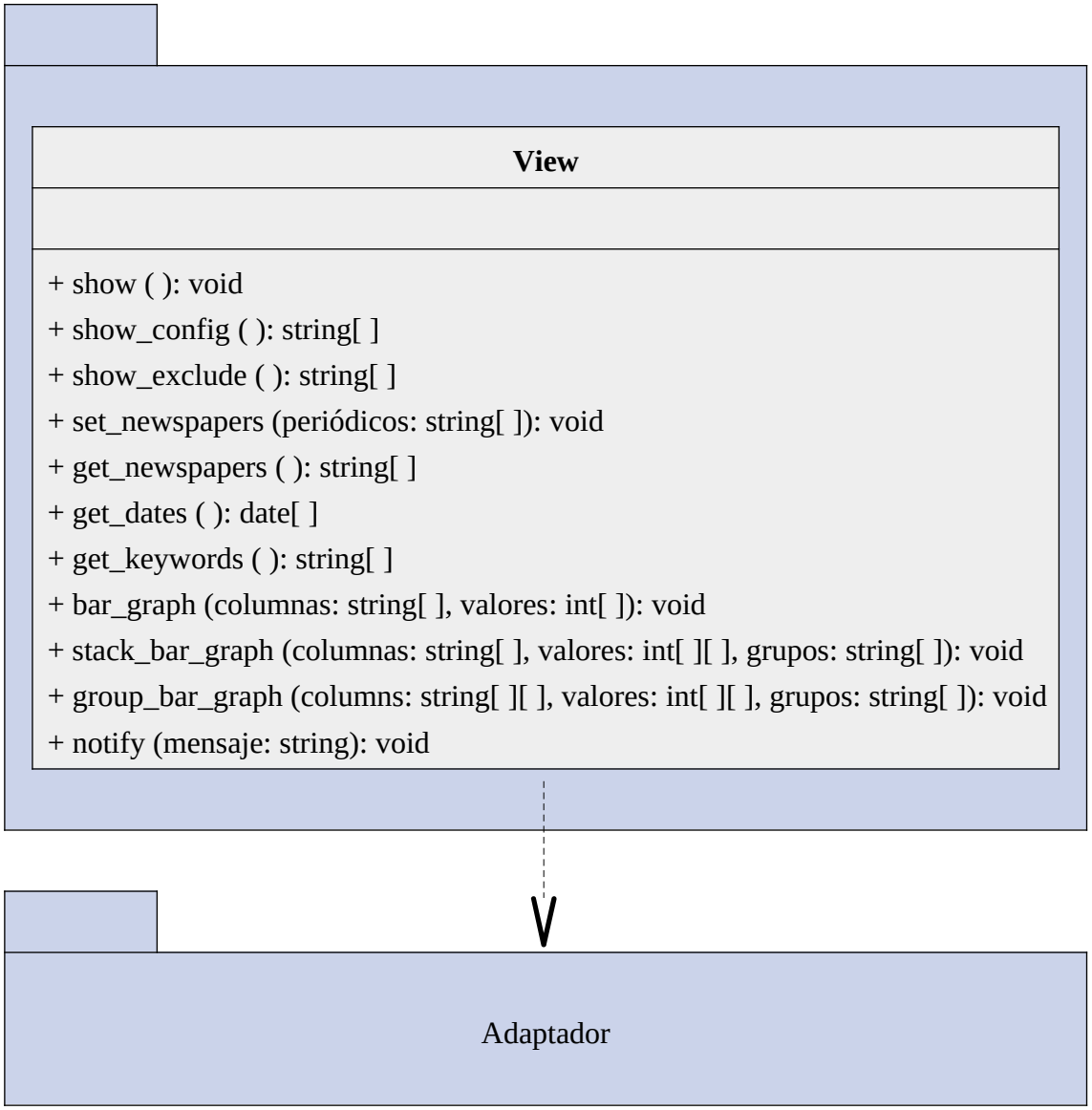
Vista

La vista está formada por una sóla clase denominada View, que gestiona los widgets y la funcionalidad relativa a ellos. La clase View contiene los siguientes métodos:

- show — Muestra la interfaz gráfica
- show_config — Muestra una ventana para editar la configuración. Devuelve la nueva configuración en caso de que el usuario realice algún cambio
- show_exclude — Muestra una ventana para editar la lista de palabras a excluir de las estadística “Ranking de palabras”. Devuelve la nueva lista en caso de que el usuario realice algún cambio
- set_newspapers — Puebla la lista de periódicos seleccionables
- get_newspapers — Devuelve los nombres de los periódicos seleccionados
- get_dates — Devuelve las fechas introducidas
- get_keywords — Devuelve las palabras clave introducidas
- bar_graph — Crea un gráfico de barras
- stack_bar_graph — Crea un gráfico de barras apiladas
- group_bar_graph — Crear un gráfico de barras agrupadas
- notify — Muestra un mensaje

El diagrama de clases para la vista se muestra en la figura 12.

Figura 12. Diagrama de clases para la vista



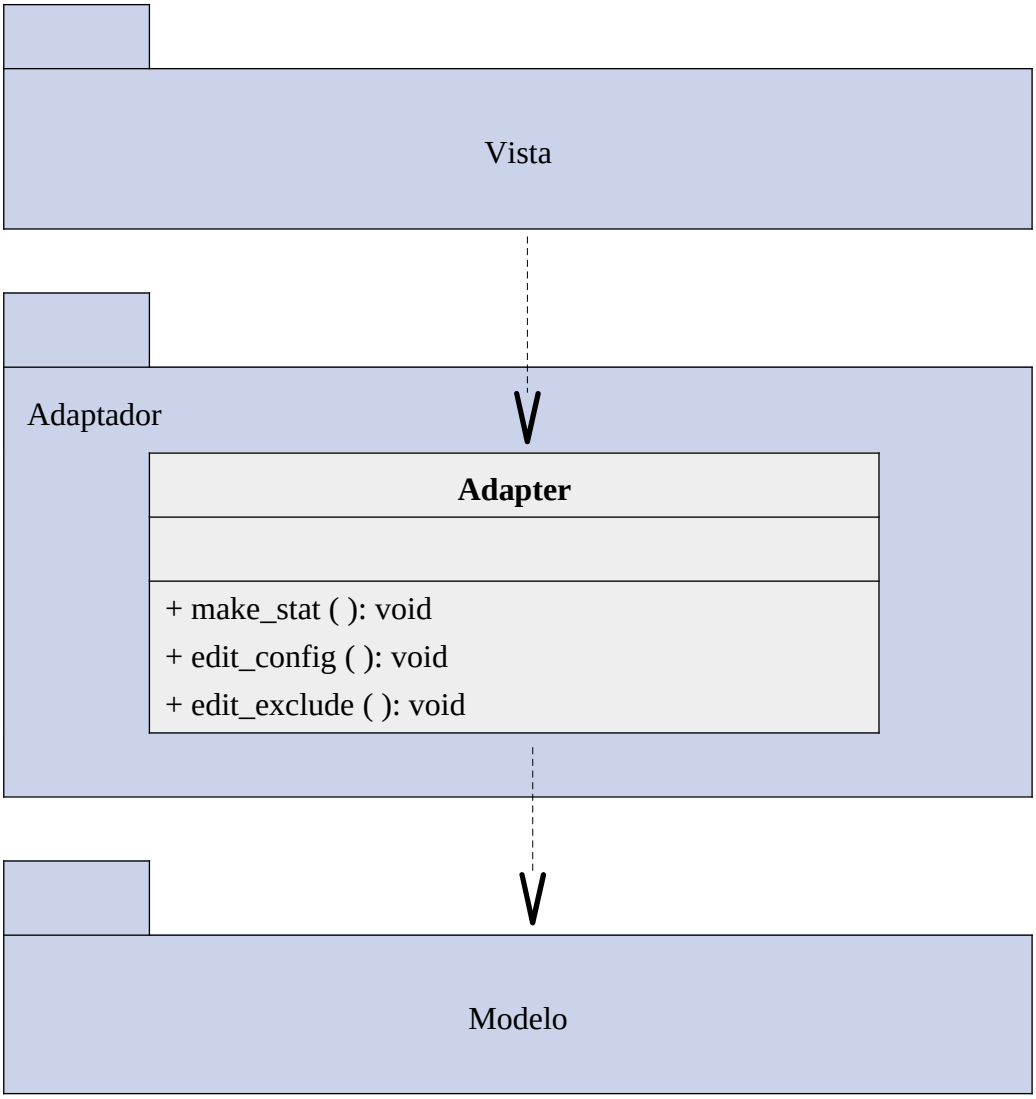
Adaptador

El adaptador está formado por una clase denominada Adapter. Los métodos de Adapter serán invocados por View, y los métodos de View serán invocados por Adapter. Los métodos de la clase Adapter son los siguientes:

- make_stat Procesa el evento “Generar estadística”
- edit_config Procesa el evento “Editar configuración”
- edit_exclude Procesa el evento “Editar palabras a excluir”

El diagrama de clases para el controlador se muestra en la figura 13.

Figura 13. Diagrama de clases para el controlador.

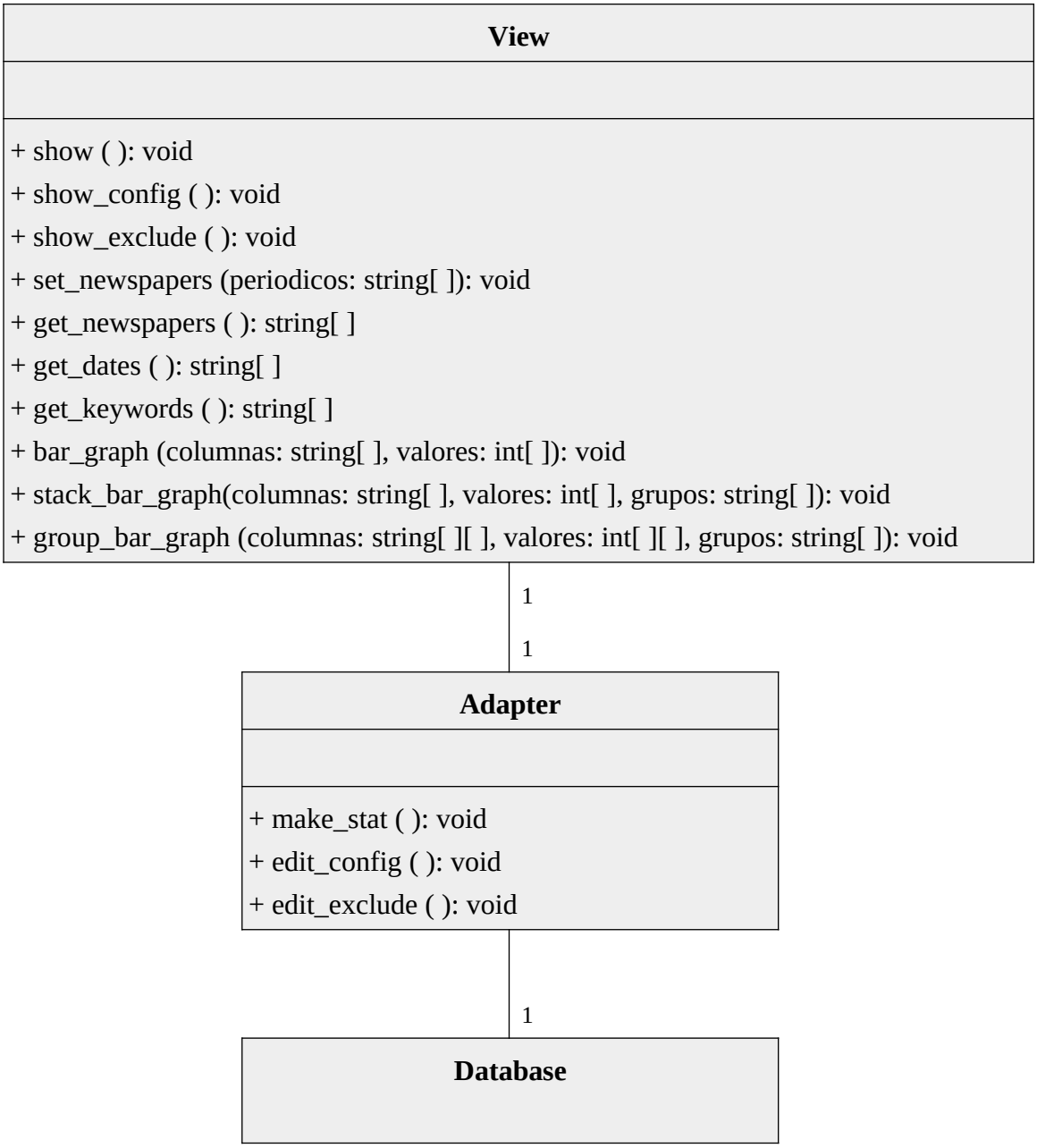


Modelo

En la parte relativa al acceso y la manipulación de datos, el sistema de estadísticas no requiere de ninguna funcionalidad más allá de aquella que ya proporciona el sistema de almacenamiento; por ello la clase Database del sistema de almacenamiento actuará también como modelo en el sistema de estadísticas.

El diagrama de clases completo para el sistema de estadísticas se muestra en la figura 14.

Figura 14. Diagrama de clases para el sistema de estadísticas



4.2. Casos de uso

En la figura 15 se muestra el diagrama de secuencia para el caso de uso “Editar configuración”. En la figura 16 se muestra el diagrama de secuencia para el caso de uso “Búsqueda de palabras clave”. Los casos de uso para el resto de estadística son análogos a este.

Figura 15. Diagrama de secuencia para el caso de uso “Editar configuración”

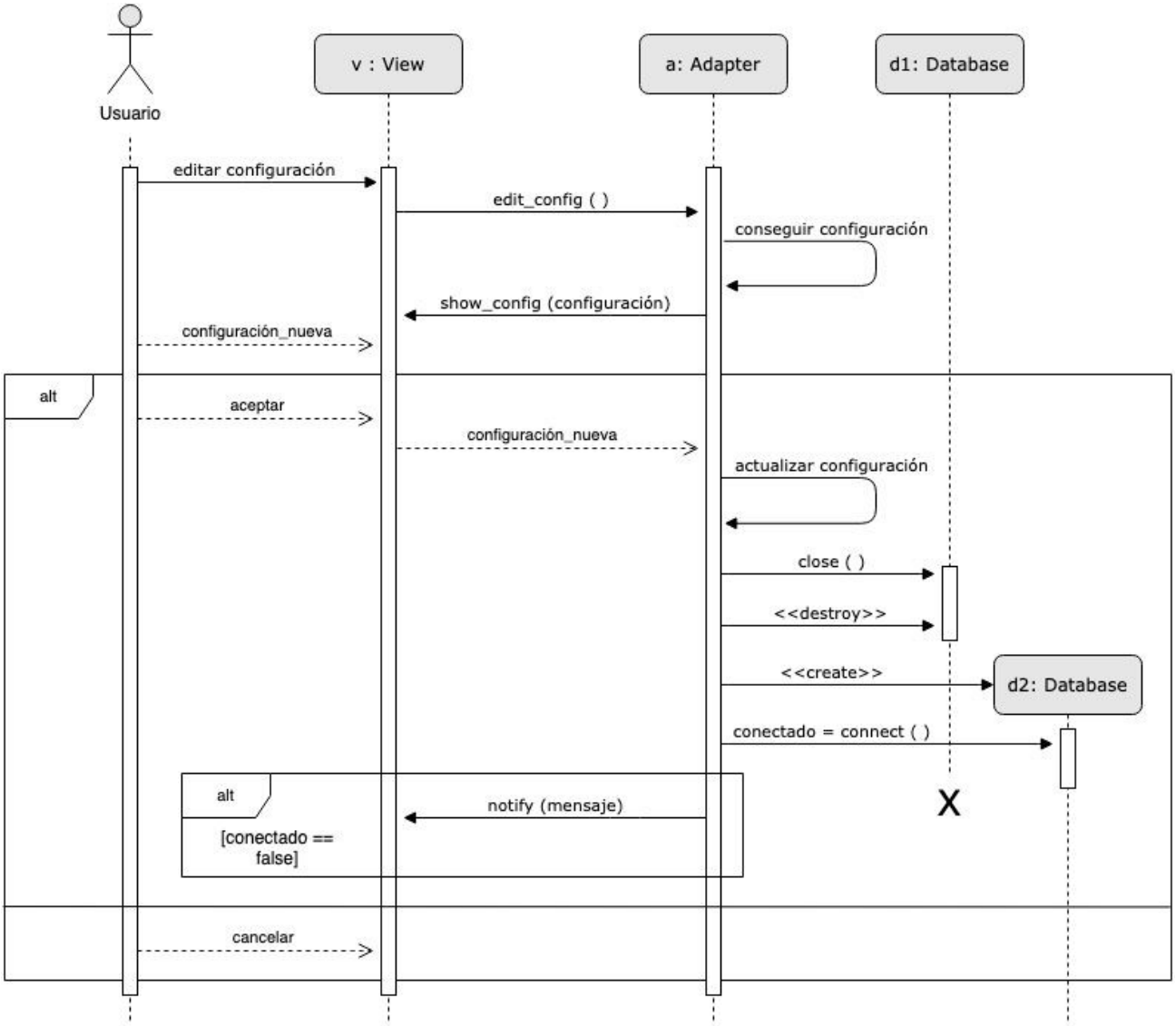
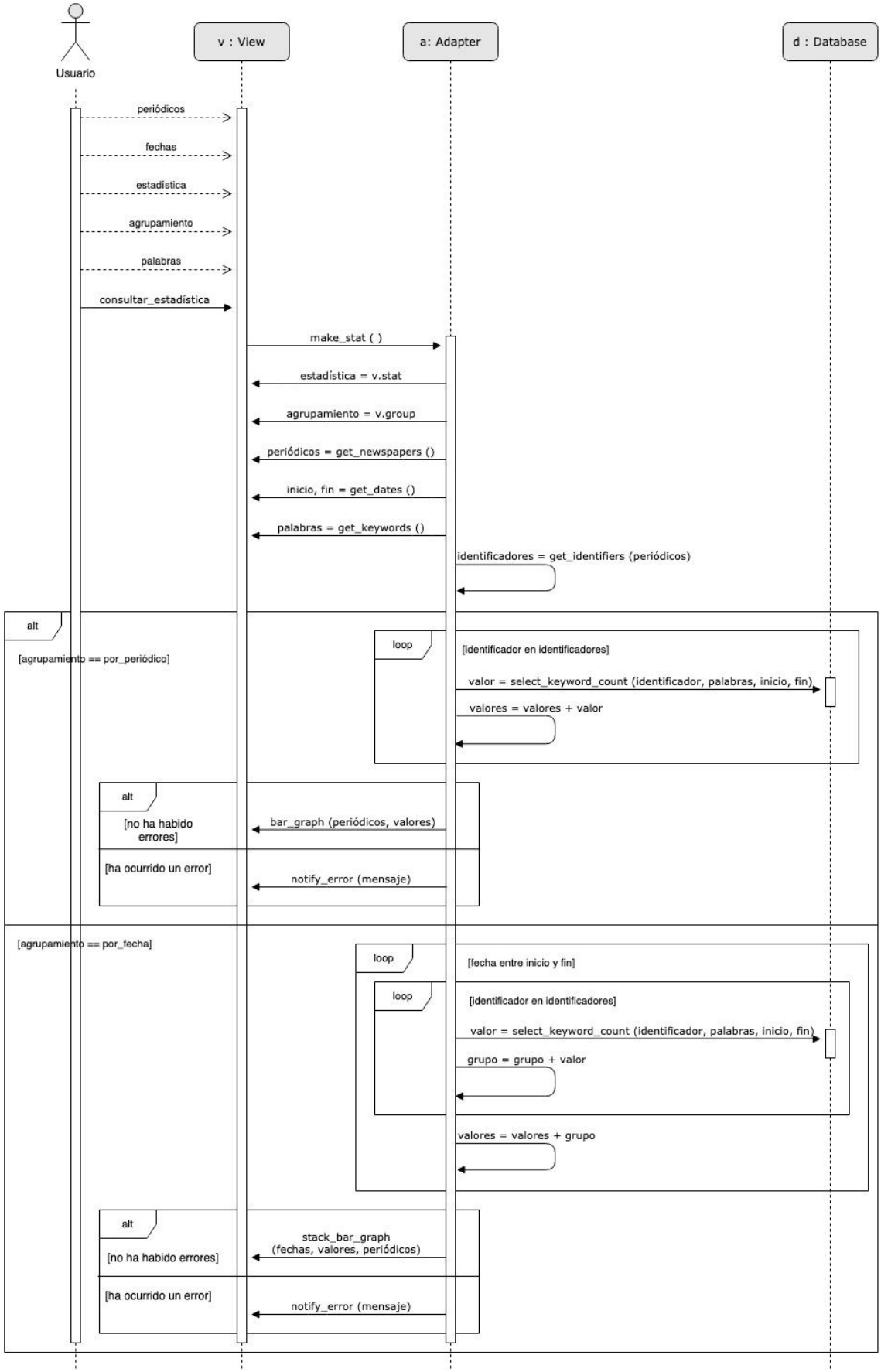


Figura 16. Diagrama de secuencia para el caso de uso “Búsqueda de palabras clave”



5. Diseño de pruebas

En esta sección se detalla la funcionalidad de la aplicación dependiente de factores externos, indicando las posibles situaciones en las que puede incurrir, cuales se consideran válidas y cuales no. Se entiende por factores externos aquellos que escapen al control del propio programa, tales como las acciones del usuario, las aplicaciones de terceros de las que dependa, las operaciones en el sistema de archivos, etc. Esta información se utilizará en la fase de implementación para elaborar los unit tests.

En la tabla 6 se listan todas las operaciones dependientes de factores externos en el sistema de recolección. Para cada una de estas operaciones se identifican, en la tabla 7, todos los posibles casos en los que podrá incurrir que se podrían considerar anómalos —los casos que no se incluyan en la tabla se asume que son válidos—. Junto con cada caso se incluyen dos campos indicando la naturaleza y la forma de proceder si se incurre en él. Los campos son los siguientes:

- Error. Un valor “Sí” indica que el caso se considerará un error o situación anómala. Un valor “-” indica que el caso se considerará admisible y dentro del funcionamiento normal del programa.
- Crítico. En los casos considerados erróneos, este campo indica la gravedad del error. Un valor “Sí” indica que el programa deberá terminar de forma excepcional en caso de incurrir en él. Un valor “-” indica que el programa podrá continuar funcionando si incurre en el caso, pero deberá notificar el error de algún modo.

De la misma forma, en la tabla 8 se listan las operaciones dependientes de factores externos en el sistema de estadísticas, y en la tabla 9 se identifican los posibles casos en los que podrán incurrir. El sistema de estadísticas no puede terminar de forma excepcional al ocurrir un error, por ello cada caso va acompañado de un único campo que indica si constituye un error; en caso afirmativo, este deberá notificarse de algún modo en la interfaz gráfica de usuario.

Tabla 6. Operaciones dependientes de factores externos en el sistema de recolección

Operación	**Descripción**
1	Buscar archivo de configuración
2	Leer archivo de configuración
3	Parsear archivo de configuración
4	Establecer conexión con la base de datos
5	Importar módulos de los periódicos

6	Recopilar artículos
7	Consultar periódicos en la base de datos
8	Insertar periódicos en la base de datos
9	Insertar artículos en la base de datos

Tabla 7. Comportamiento del sistema de recolección

	Caso	**Error**	**Crítico**
1	El programa se invoca indicando la ruta al archivo	-	-
	El programa se invoca sin argumentos	-	-
2	La ruta al archivo de configuración no existe	SÍ	SÍ
	La ruta no se corresponde con un archivo	SÍ	SÍ
	No se dispone de permisos para leer el archivo	SÍ	SÍ
	La lectura falla	SÍ	SÍ
3	El formato no se ajusta a TOML	SÍ	SÍ
	Faltan parámetros obligatorios	SÍ	SÍ
	Hay parámetros repetidos	SÍ	SÍ
4	No se puede conectar	SÍ	SÍ
	La conexión se realiza pero la base de datos especificada no existe	SÍ	SÍ
	La conexión se realiza pero la cuenta de usuario especificada no existe	SÍ	SÍ
	La conexión se raliza pero la contraseña especificada no es válida	SÍ	SÍ
5	La ruta al directorio donde buscar los módulos no existe	SÍ	SÍ
	La ruta al directorio donde buscar los módulos no se corresponde con un directorio	SÍ	SÍ
	No se dispone de permisos para leer el directorio	SÍ	SÍ
	Alguno de los módulos especificados no existe	SÍ	SÍ

	No se dispone de permisos para leer alguno de los módulos	SÍ	SÍ
	Alguno de los módulos contiene código Python incorrecto	SÍ	SÍ
	Alguno de los módulos no se ajusta al formato establecido	-	-
6	No se puede establecer la conexión	SÍ	-
	Se recibe un código de error HTTP	SÍ	-
	La extracción de datos del HTML no se realiza correctamente	SÍ	-
7	La conexión ya establecida con la base de datos falla	SÍ	-
	La tabla que se quiere consultar no existe	Sí	-
	El periódico que se quiere consultar no existe	SÍ	-
	No se dispone de permisos de lectura en la tabla	Sí	-
8	La conexión ya establecida con la base de datos falla	Sí	-
	Se produce una violación de clave primaria	SÍ	-
	La tabla en la que se quiere insertar no existe	SÍ	-
	No se disponde de permisos de escritura en la tabla	SÍ	-
9	La conexión ya establecida con la base de datos falla	SÍ	-
	Se produce una violación de clave primaria	SÍ	-
	Se produce una violación de clave foránea	SÍ	-
	La tabal en la que se quiere insertar no existe	SÍ	-
	No se disponde de permisos de escritura en la tabla	SÍ	-

Tabla 8. Operaciones dependientes de factores externos en el sistema de estadísticas

Operación	**Descripción**
1	Leer archivo de configuración
2	Parsear archivo de configuración
3	Establecer conexión con la base de datos

4	Consultar periódicos de la base de datos
5	Consultar artículos de la base de datos
6	Editar configuración

Tabla 9. Comportamiento del sistema de estadísticas

	Caso	**Error**
1	La ruta no existe	-
	La ruta no se corresponde con un archivo	SÍ
	No se dispone de permisos de lectura sobre el archivo	SÍ
2	El formato no se ajusta a TOML	SÍ
	Faltan parámetros obligatorios	SÍ
	Hay parámetros repetidos	SÍ
3	No se puede conectar con el SGBD	SÍ
	La base de datos no existe	SÍ
	La cuenta de usuario no existe	SÍ
	La contraseña no es válida	SÍ
4	La conexión ya establecida con la base de datos falla	SÍ
	La tabla que se quiere consultar no existe	SÍ
	El periódico que se quiere consultar no existe	SÍ
	No se dispone de permisos de lectura en la tabla	SÍ
5	La conexión ya establecida con la base de datos falla	SÍ
	La tabla que se quiere consultar no existe	SÍ
	El periódico que se quiere consultar no existe	SÍ
	No se dispone de permisos de lectura en la tabla	SÍ
6	La configuración introducida por el usuario es errónea	SÍ

	La ruta al archivo de configuración no existe	-
	La ruta no se corresponde con un archivo	SÍ
	No se dispone de permisos de escritura sobre el archivo	SÍ
	La escritura falla	SÍ

6. Desarrollo

6.1. Herramientas

Para desarrollar el proyecto se utilizará el lenguaje de programación Python 3. Python es un lenguaje interpretado, de propósito general y multiplataforma. Toda aquella funcionalidad requerida por el proyecto que no se incluya por defecto en el lenguaje o en la librería estándar se logrará mediante el uso librerías externas. Python cuenta con un amplio catálogo oficial de liberías externas y con su propio gestor de paquetes, por lo que la instalación de dependencias debería ser trivial. La versión utilizada para desarrollar el proyecto es Python 3.8; no se garantiza que la aplicación funcione con versiones anteriores. En los subsiguientes apartados de este documento se utilizará terminología específica de Python.

Como sistema de almacenamiento se ha elegido MariaDB, una base de datos realacional bastante popular que ofrece un alto rendimiento. MariaDB es compatible con MySQL, otro sistema gestor ampliamanete utilizado, por lo que ambos podrán intercambiarse de forma transparente para la aplicación.

El proyecto se enfocará principalmente en Linux. Se dará soporte también al sistema operativo Windows, pero la aplicación no será testeada extensivamente en esta plataforma. Al estar escrita en el lenguaje Python, la aplicación debería poder funcionar en otros sistemas operativos sin necesidad de alterar el código fuente. Del mismo modo, la base de datos podrá ser albergada en cualquier equipo sin importar el sistema operativo, ya que MariaDB proporciona implementaciones para varias plataformas.

6.2. Librerías

En esta sección se detallan las librerías externas elegidas para implementar la funcionalidad presentada en el apartado 2. Se considera librerías externas a todos aquellos componentes que no vengan incluidos en el propio lenguaje Python o en la librería estándar. Para consultar el listado de módulos empleados y sus versiones, véase el anexo 8.6.

Para realizar el *web scraping* se ha optado por Beautiful Soup, una librería de extracción de datos de archivos HTML y XML. Beautiful Soup permite mapear contenido HTML y XML en una estructura de datos en forma de árbol, y proporciona una interfaz sencilla para navegar dicho árbol, buscar las etiquetas pertinentes y extraer los datos.

La funcionalidad relacionada con HTTP la proporcionará urllib. Este módulo permite realizar conexiones HTTP y HTTPS con un servidor web y descargar los archivos asociados a una determinada URL.

La conexión con la base de datos se realizará mediante PyMySQL. Este módulo permite establecer conexiones con una base de datos MySQL o MariaDB, realizar consultas e insertar datos en ella. Python cuenta con una especificación oficial que define una API estándar para aquellos módulos relacionados con SQL. La mayoría de módulos para bases de datos implementan esta API, por lo que el código de la aplicación relativo a este ámbito debería ser similar independientemente de cuál de ellos se use.

Para crear los diagramas se hará uso de Matplotlib. Matplotlib es una librería de creación de gráficos enfocada al uso matemático ampliamente utilizada en el mundillo del análisis de datos. Es extremadamente versátil y se integra de forma nativa con varias plataformas gráficas, permitiendo embeber diagramas en ventanas u otros widgets de manera sencilla.

La interfaz gráfica de usuario se construirá por medio de tkinter, que es considerada la plataforma gráfica *de facto* de Python. Tkinter proporciona toda la funcionalidad requerida por la aplicación; funciona de forma similar a cualquier otra librería basada en widgets, puede integrarse de forma con Matplotlib y cuenta con una gran cantidad de documentación disponible en internet debido a su larga trayectoria. El módulo tkinter suele venir instalado por defecto con Python.

Para analizar el contenido de los archivos de configuración se utilizará el módulo toml. Este módulo permite transformar cadenas de texto formateadas según el lenguaje TOML en un diccionario, una estructura de datos muy fácil de navegar. Una vez volcada la configuración en el diccionario, la comprobación de los parámetros y su formato deberá resultar trivial.

Para realizar las pruebas se utilizará el módulo unittest. Esta librería proporciona una funcionalidad de unit testing análoga a la de cualquier otra plataforma o lenguaje de programación. Permite diseñar tests que validen el comportamiento y los datos devueltos por cada método o función de forma sencilla, y muestra los resultados de la ejecución de los tests mediante una interfaz intuitiva.

6.3. Implementación

En este apartado se describe la implementación en lenguaje Python del proyecto a partir del diseño descrito en la sección 4.1. Se explican las decisiones de implementación tomadas, así como todas las consideraciones tenidas en cuenta a la hora de escribir el código fuente, y se documentan todas las clases y funciones que conforman la aplicación.

Al desarrollar la aplicación no se ha prestado especial atención al rendimiento ni a la eficiencia. En su lugar se ha priorizado la legibilidad y la sencillez del código. Se ha tratado de respetar las guías de estilo de Python en todo momento, pero este aspecto se considera poco relevante y puede que no se hayan seguido de forma rigurosa.

El proyecto ha sido desarrollado mediante un enfoque modular. El código fuente se ha dividido en multitud de funciones y/o clases, que constituyen una unidad lógica lo más pequeña posible y abarcan funcionalidad independiente. Este enfoque a la hora de escribir el programa tiene tres propósitos: facilitar el mantenimiento de la aplicación, aumentar la legibilidad del código fuente y permitir realizar un testeo efectivo mediante baterías de prueba *unit testing*.

La forma de abordar la gestión de errores es la siguiente. Cuando una función contiene un sólo punto de fallo o contiene varios puntos pero todos constituyen el mismo tipo de fallo, la función indicará si ha tenido éxito mediante el valor de retorno. Un valor False —o None en caso de que la función devuelva datos— indicará que un error ha sucedido. Cuando una función puede fallar de varias formas distintas, siendo cada fallo un tipo de error conceptualmente distinto, la función hará uso de excepciones para distinguir entre ellos.

El sistema de almacenamiento se ha implementado mediante una clase denominada DBManager, que contiene exactamente los mismos métodos descritos en la clase Database de la fase de diseño. En el contexto de la aplicación, se han identificado tres tipos de error que pueden ocurrir al manejar la base de datos: error en la conexión, error de integridad y error genérico. Un error en la conexión se considera cualquier problema de conectividad con la base de datos. Un error de integridad lo constituyen una violación de clave primaria o foránea. Un error genérico engloba cualquier otro tipo de fallo: consultas erróneas, falta de permisos, etc. Los errores genéricos se considera que son improbables y poco relevantes. Para representar estos errores se han definido tres tipos de excepción, así como un tipo adicional del que heredan las otras tres.

El sistema de recolección se ha implementado mediante un conjunto de funciones estáticas, dos clases denominadas DaemonConfig y Plugin, y dos módulos Python que contienen la funcionalidad relativa a la consulta de los periódicos. Las clases DaemonConfig, Plugin y las funciones estáticas se corresponderían con la clase Daemon de la fase de diseño.

La clase DaemonConfig se encarga de abstraer y gestionar la configuración: actúa como un contenedor que almacenará los parámetros del programa durante el tiempo de vida de este, y proporciona un método para analizar el contenido del archivo de configuración y extraer los parámetros de éste.

Los módulos de los periódicos se corresponden con la clase Newspaper de la fase de diseño, y se han implementado mediante dos funciones estáticas. No implementan ninguna interfaz, pues se considera que la estructura es trivial. Para facilitar la gestión de estos módulos se cuenta con la clase Plugin, que sirve para encapsular y simplificar la funcionalidad relativa a la carga. Una instancia de Plugin representa un periódico; contiene métodos para cargar el módulo, así como intermediarios para invocar a los propios métodos del módulo.

Para llevar a cabo la recolección, los módulos descargan la sección de últimas noticias de sus respectivos periódicos, vuelcan el HTML en un diccionario y realizan la búsqueda de los datos en

este diccionario. Si los datos pertinentes se encuentran en otra página cuyo enlace se indica en la sección de últimas noticias, se descarga esta página y se realiza exactamente el mismo proceso.

En el sistema de estadísticas la vista se ha implementado mediante una clase denominada GUIView, que se corresponde con la clase View de la fase de diseño. El adaptador se ha implementado mediante dos clases, una denominada GUIAdapter, que contiene la lógica principal, y otra denominada GUIConfig, que encapsula la funcionalidad relativa a la configuración. La clase DBManager actúa como modelo.

La clase GUIConfig funciona de forma similar a la clase DaemonConfig descrita anteriormente; almacena los parámetros del programa durante el tiempo de vida de este y proporciona métodos para obtener la ubicación del archivo de configuración, analizar su contenido y formatear los parámetros en el formato adecuado para ser escrito.

A continuación se indican todas las iteraciones llevadas a cabo durante la etapa de creación del software, indicando los requisitos implementados y el resultado obtenido en cada una de ellas. Las iteraciones se listan en el orden en que fueron realizadas; los requisitos pueden consultarse en las tablas de la sección 2.2.

1. Requisitos implementados: 1
    - Resultado: módulos Python para consultar los artículos de la sección de últimas noticias de dos periódicos.
    - Se implementa la descarga de las páginas web y el análisis del código HTML.
2. Requisitos implementados: 2, 3, 4
    - Resultado: una librería que permite conectar con la base de datos e insertar artículos y periódicos.
    - Se crea la base de datos. Se crea la clase DBManager con un método para conectar con la base de datos y dos métodos para realizar inserciones.
3. Requisitos implementados: 2, 3, 4, 17
    - Resultado: una aplicación funcional que al ser ejecutada consulta todos los artículos de la sección de últimas noticias de los dos periódicos e intenta insertarlos en una base de datos.
    - Se crea la clase Plugin y se implementa parte de la funcionalidad de la clase Daemon del diseño.
4. Requisitos implementados: 5, 6, 7, 8, 9, 10, 11, 12, 13
    - Resultado: la aplicación utiliza la base de datos y los periódicos especificados por el usuario.

- Se crea la clase DaemonConfig, que implementa la funcionalidad relativa al análisis de la configuración, y se incorporan funciones para leer archivos de texto.

5. Requisitos implementados: 18, 19

   - Resultado: la aplicación funciona de forma indefinida y es capaz de discriminar artículos.

   - Se implementa el bucle principal de la aplicación y el uso de archivos log para almacenar la fecha del artículo más reciente de cada iteración.

6. Requisitos implementados: 15, 16

   - Resultado: la aplicación reacciona correctamente al encontrarse con errores de lectura o una configuración inválida.

   - Se implementa la gestión de errores en la clase DaemonConfig y en la funcionalidad relativa a la lectura de archivos.

7. Requisitos implementados: 14

   - Resultado: el archivo de configuración se busca en una ubicación por defecto si este no se indica.

   - Se implementa la búsqueda del archivo de configuración por defecto en las diferentes plataformas.

8. Requisitos implementados: 21

   - Resultado: la aplicación cuenta con un nuevo ejecutable, el cual pone en marcha una interfaz gráfica sin ningún tipo de funcionalidad.

   - Se crea la clase GUIView y se implementan los widgets.

9. Requisitos implementados: 25

   - Resultado: la interfaz gráfica permite visualizar diagramas de barras, pero aún carece de funcionalidad.

   - Se implementan los métodos para crear diagramas de barras en la clase GUIView.

10. Requisitos implementados: 26, 27, 28

    - Resultado: la interfaz gráfica cuenta con algo de lógica, pero aún carece de funcionalidad.

    - Se crea la clase GUIAdapter. Se implementa la lógica para consultar y validar los parámetros introducidos por el usuario en GUIAdapter y GUIView.

11. Requisitos implementados: 29, 30, 31, 32, 33, 34, 35

    - Resultado: la interfaz gráfica realiza una conexión con la base de datos que especifique el usuario.

    - Se crea la clase GUIConfig, que implementa la funcionalidad relativa al análisis de la configuración, y se incorporan métodos para leer archivos de texto a la clase GUIAdapter.

12. Requisitos implementados: 22

    - Resultado: la interfaz gráfica permite realizar la estadística “Longitud media del titular”.

    - Se incorporan los métodos de consulta oportunos a la clase DBManager y se implementa la lógica necesaria en la clase GUIAdapter.

13. Requisitos implementados: 23

    - Resultado: la interfaz gráfica permite realizar la estadística “Búsqueda de palabras clave”.

    - Se incorporan los métodos de consulta oportunos a la clase DBManager y se implementa la lógica necesaria en la clase GUIAdapter.

14. Requisitos implementados: 24

    - Resultado: la interfaz gráfica permite realizar la estadística “Ranking de palabras”.

    - Se incorporan los métodos de consulta oportunos a la clase DBManager y se implementa la lógica necesaria en la clase GUIAdapter.

15. Requisitos implementados: 36, 37, 38

    - Resultado: la interfaz gráfica reacciona correctamente al encontrarse con errores de lectura, consultas fallidas o una configuración inválida.

    - Se crea un mecanismo de notificación de mensajes en la clase GUIView. Se implementa la gestión de errores en la clase GUIConfig, en los métodos para leer archivos y en los métodos encargados de realizar las estadísticas en la clase GUIAdapter.

16. Requisitos implementados: 39

    - Resultado: la configuración puede editrase desde la interfaz gráfica.

- Se implementa una ventana de edición en la interfaz gráfica y se incorporan métodos para formatear la configuración y escribir en archivos en las clases GUIConfig y GUIAdapter.

17. Requisitos implementados: 28

- Resultado: la aplicación permite editar las palabras a excluir para realizar la estadística “Ranking de palabras”.
- Se crea un ventana de edición en la interfaz gráfica y se implementa la lógica necesaria en la clase GUIAdapter.

En los siguientes apartados se documentan las clases y funciones que conforman la aplicación.

6.3.1. DBManager

DBManager (address, port, name, user, password)

Constructor.

address: string

Dirección IP o nombre de dominio de la base de datos

port: int

Puerto de la base de datos

name: string

Nombre de la base de datos

user: string

Cuenta de usuario para la base de datos

password: string

Contraseña de la cuenta

connect ( )

Establecer conexión con la base de datos. Devuelve True si la conexión se realiza con éxito, o False si ocurre algún error.

close ( )

Cerrar la conexión con la base de datos. Si no hay ninguna conexión abierta, este método no hace nada.

select_newspaper_id (name)

Consultar identificador asociado a un periódico. Devuelve un int, o None si el periódico no existe.

name: string

Nombre del periódico

select_newspapers ( )

Consultar nombre e identificador de todos los periódicos. Devuelve una lista de tuplas. Cada tupla contiene dos elementos: el primero es un string con el nombre del periódico, el segundo es un int con el identificador del periódico.

select_articles (identifier, start_date, end_date)

Consultar titulares de los artículos de uno o varios periódicos. Devuelve una lista de strings.

identifier: int o lista de ints

Identificador del periódico o los periódicos a consultar

start_date: date o datetime

Fecha mínima de publicación a la que restringir la búsqueda

end_date: date o datetime

Fecha máxima de publicación a la que restringir la búsqueda

select_keyword_count (identifiers, keywords, start_date, end_date)

Consultar el número de titulares que contienen una o varias palabras clave.

identifiers: int o lista de ints

Identificador del periódico o los periódicos a consultar

keywords: lista de strings

Palabras clave a buscar

start_date: date o datetime

Fecha de publicación mínima a la que restringir la búsqueda

end_date: date o datetime (opcional)

Fecha de publicación máxima a la que restringir la búsqueda. Si no se especifica, se consultarán los artículos publicados en la fecha indicada por start_date

select_average_length (identifier, start_date, end_date)

Consultar longitud media de los titulares de uno o varios periódicos.

identifier: int o lista de ints

Identificador del periódico o los periódicos a consultar

start_date: date o datetime

Fecha de publicación mínima a la que restringir la búsqueda

end_date: date o datetime (opcional)

Fecha de publicación máxima a la que restringir la búsqueda. Si no se especifica, se consultarán los artículos publicados en la fecha indicada por start_date

insert_newspaper (name)

Insertar un periódico.

name: string

Nombre del periódico

insert_article (newspaper, title, publication, words, chars)

Insertar un artículo

newspaper: int

Identificador del periódico al que pertenence el artículo

title: string

Titular del artículo

publication: datetime

Fecha de publicación del artículo

words: int

Número de palabras que contiene el artículo

chars: int

Número de caracteres que contiene el artículo

format_date (date)

Formatear fecha como cadena de texto. Devuelve un string. Este método es de uso interno.

date: date

Fecha a formatear

format_datetime (datetime)

Formatear fecha y hora como cadena de texto. Devuelve un string. Este método es de uso interno.

datetime: datetime

Fecha y hora a formatear

Los métodos de la clase DBManager —a excepción de los métodos de uso interno— podrán lanzar las siguientes excepciones:

- ConnectionError Conexión fallida
- IntegrityError Violación de clave primaria o foránea
- GenericError Error sin especificar

Todos los métodos podrán lanzar ConnectionError y GenericError. Los métodos que realicen inserciones pordrán lanzar IntegrityError.

6.3.2. GUIView

GUIView ( )

Constructor.

show ( )

Iniciar la interfaz gráfica.

show_config ( )

Mostrar ventana de configuración. Si el usuario selecciona “Aceptar”, devuelve una tupla de strings con los parámetros de la configuración introducidos. Si el usuario selecciona “Cancelar”, devuelve None.

show_exclude ( )

Mostrar ventana de palabras a excluir. Si el usuario selecciona “Aceptar”, devuelve una lista de strings con las palabras introducidas. Si el usuario selecciona “Cancelar”, devuelve None.

set_newspapers (names)

Rellenar la lista de periódicos seleccionables.

names: lista de strings

Nombres de los periódicos

get_newspapers ( )

Consultar los periódicos seleccionados por el usuario. Devuelve una lista de strings o una lista vacía.

get_dates ( )

Consultar las fechas seleccionadas por el usuario. Devuelve una tupla con seis strings, respectivamente: día inicial, mes inicial, año inicial, día final, mes final, año final

get_keywords ( )

Consulta las palabras clave introducidas por el usuario. Devuelve uns lista de strings o una lista vacía.

bar_graph (columns, values)

Construir un diagrama de barras.

columns: lista de strings o lista de dates

Nombres de cada columna

values: lista de ints, lista de floats o lista de Decimals

Valores para cada columna

group_bar_graph (columns, values, groups)

Construir un diagrama de barras agrupadas.

columns: lista de listas de strings

Nombres de cada columna. Deberá contener una lista de strings por cada grupo especificado

values: lista de listas de ints

Valores para cada columna. Deberá contener una lista de ints por cada grupo especificado

groups: lista de strings

Nombres de los grupos

stack_bar_graph (columns, values, legend)

Construir diagrama de barras apiladas.

columns: lista de strings

Nombres de cada columna.

values: lista de lista de ints

Valores para cada columna. Deberá contener una lista de ints por cada grupo especificado.

groups: lista de strings

Nombres de los grupos

notify (message)

Mostrar notificación

message: string

Mensaje a mostrar

make_new_graph (columns)

Reservar espacio para un diagrama. Este método es de uso interno.

columns: int

Número de columnas que tendrá el diagrama

validate (text)

Comprobar que una cadena de texto tiene formato numérico. Este método es de uso interno.

6.3.3. GUIAdapter

GUIAdapter (view)

Constructor.

view: GUIView

Instancia de la vista a la que vincular el adaptador.

edit_config ( )

Mostrar ventana de configuración. Si el usuario hace algún cambio, actualiza la configuración y crear una nueva conexión con la base de datos.

edit_exclude ( )

Mostrar ventana de palabras a excluir. Si el usuario hace algún cambio, actualiza la lista de palabras a excluir.

make_stat ( )

Realizar estadística y crear diagrama.

connect_database ( )

Conectar con la base de datos y poblar la vista con los periódicos disponibles. Este método es de uso interno sólo.

date_range (start_date, end_date)

Iterar sobre un intervalo de fechas, extremos incluidos. Este método es de uso interno.

start_date: datetime

Fecha inicial.

end_date: datetime

Fecha final.

top_words (words, count)

Consultar las palabras más comunes de una lista de strings. Devuleve una lista de tuplas; cada tupla contiene un string con una palabra y un int indicando el número de apariciones de la palabra. Este método es de uso interno.

words: lista de strings

Palabras sobre las que realizar la búsqueda

count: int

Número de palabras al que restringir la búsqueda

read_file (path)

Leer contenido de un archivo. Devuelve un string, o None si la lectura falla. Este método es de uso interno.

path: string

Ruta al archivo

get_directory (path)

Consultar directorio en el que está contenido un archivo. Devuelve un string. Este método es de uso interno

path: string

Ruta al archivo

connect_database ( )

Conectar con la base de datos. Este método es de uso interno.

6.3.4. GUIConfig

GUIConfig ( )

Constructor.

config_path ( )

Consultar ruta al archivo de configuración. Devuelve un string.

parse (text)

Parsear texto en formato TOML y extraer la configuración de él. Devuelve True si el formato del texto es válido y la extracción se realiza con éxito, False si no.

text: string

Texto a parsear

format ( )

Formatear configuración como TOML. Devuelve un string.

6.3.5. Daemon

read_config ( )

Leer y parsear el archivo de configuración. Devuelve un objeto DaemonConfig, o termina el programa si ocurre un error.

check_directories (directory1, directory2)

Comprobar que los directorios existen. Termina el programa si no existen.

directory1: string

directory2: string

Directorios a comprobar

load_modules (directory, names)

Importar módulos. Devuleve una lista de Plugins, o termina el programa si ocurre algún error.

directory: string

Directorio que contiene los módulos

names: lista de strings

Nombres de los módulos, sin la extensión .py

connect_database (address, port, name, user, password)

Conectar con la base de datos. Devuleve un objeto DBManager, o termina el programa si la conexión no se puede establecer.

address: string

Nombre de dominio o dirección IP de la base de datos

port: int

Puerto de la base de datos

name: string

Nombre de la base de datos

user: string

Cuenta de la base de datos

password: string

Contraseña de la cuenta

get_newspapers (database, plugins, log)

Consultar nombre, identificador y fecha del último artículo procesado de los periódicos. Devuelve una lista de diccionarios.

database: DBManager

Base de datos a consultar

plugins: lista de Plugins

Periódicos a consultar

log: string

Directorio que contiene los archivos log

get_newspaper_id (database, names)

Consultar identificadores de los periódicos. Devuelve una lista de ints.

database: DBManager

Base de datos a consultar

names: lista de strings

Nombres de los periódicos a consultar

get_log_files (directory, plugins)

Obtener nombre de los archivos log de los periódicos

directory: string

Directorio que contienen los archivos log

plugins: lista de Plugins

Periódicos

get_last_article_dates (files)

Consultar fecha de publicación de los últimos artículos recopilados. Devuelve una lista cuyos elementos pueden ser datetimes, o None si el archivo no se ha podido leer.

files: lista de strings

Rutas a los archivos log

insert_article (database, identifier, article, most_recent_date)

Insertar artículo en la base de datos. Devuelve la fecha del artículo más reciente.

database: DBManager

Base de datos donde realizar la inserción

identifier: int

Identificador del periódico

article: (string, datetime)

Artículo a insertar. El primer elemento es el titular, el segundo elemento es la fecha de publicación.

main_loop (database, newspapers)

Bucle principal. Recopilar artículos e insertarlos en la base de datos

database: DBManager

Base de datos en la que realizar las inserciones

newspapers: lista de diccionarios

Periódicos. Cada elemento contiene el nombre, identificador, ruta al archivo log, fecha del último artículo consultado y Plugin de un periódico.

config_path ( )

Consultar ruta al archivo de configuración. Devuelve un string.

should_be_inserted (article, log_date)

Determinar si un artículo debe ser insertado en la base de datos

article: tupla de la forma (string, datetime)

Artículo

log_date: datetime

Fecha de publicación del más reciente de los artículos procesados hasta ahora

parse_date (text)

Extraer fecha y hora de una cadena de texto. Devuelve un datetime. Si el formato es incorrecto, devuelve la fecha mínima (UNIX epoch)

text: string

Texto con la fecha y hora

read_file (path)

Leer contenido de un archivo. Devuelve un string, o None si la lectura falla.

path: string

Ruta al archivo

write_date (path, date)

Escribir fecha en un archivo. Devuleve True si la escritura se realiza correctamente, False si no

path: string

Ruta al archivo

date: datetime

Fecha a escribir

log (message)

Imprimir mensaje por salida de error.

message: string

Mensaje a imprimir

abort (message)

Imprimir mensaje por salida de error y terminar el programa con código de error.

message: string

Mensaje a imprimir

6.3.6. DaemonConfig

DaemonConfig ( )

Constructor.

parse (text)

Analizar texto en formato TOML y extraer la configuración de él. Devuelve True si el formato del texto es válido y la extracción se realiza con éxito, False si no.

text: string

Texto a parsear.

6.3.7. Módulos de periódicos

get_newspaper ( )

Consultar nombre del periódico. Devuelve un string.

find_articles ( )

Recopilar artículos. Devuelve una tupla con dos elementos: una lista de tuplas y un bool. Las tuplas de la lista contendrán un string con el titular del artículo y un datetime con la fecha de publicación del artículo. El bool indica si ha ocurrido algún error al procesar alguno de los artículos: un valor True significa que todo ha ido correctamente. Si el proceso falla completamente, en lugar de una lista de tuplas devuelve None.

6.4. Pruebas

El correcto funcionamiento de la aplicación se comprobará mediante baterías de prueba *unit testing*. Las funciones que dependan del sistema de archivos y el comportamiento de la interfaz gráfica no se testearán, por lo que deberá analizarse esta parte del código con especial detenimiento para garantizar su correcto funcionamiento. Cuando una función o una clase dependa de otra clase, se elaborará una clase de pruebas que simule a la clase verdadera.

Para testear la clase DBManager se utilizará una base de datos de prueba, que se creará mediante la herramienta Docker. Docker proporciona una funcionalidad similar a una máquina virtual: permite crear contenedores que contengan un sistema de archivos completo en los que ejecutar aplicaciones de forma aislada. El motivo por el que se utilizará Docker es para aislar la base de datos de prueba del resto, minimizando así la posibilidad de cometer errores que puedan afectar a la integridad de los datos. Docker presenta una característica muy útil para esta situación, y es que los cambios realizados en un contenedor no se guardarán para la siguiente ejecución. De esta forma, cuando se quiera realizar el test bastará con reiniciar el contenedor y la base de datos retornará a su estado original.

Se elaborará una clase unit test que evalúe todos los métodos de DBManager. La comprobación de los métodos para realizar consultas es trivial; basta con comparar los datos obtenidos y los que se sabe que están presentes en la base de datos. Si ambos coinciden, la consulta se ha realizado con éxito y ha sucedido tal y como se planeó. La evaluación de los métodos para realizar inserciones requiere comprobar que los datos están presentes en la base de datos; para ello se utilizarán los propios métodos de DBManager. Al ser testeado cada método por separado, esto no debería suponer ningún inconveniente.

La parte del código que no se ha testeado son la clase GUIView y algunas funciones del módulo daemon. Las baterías de prueba desarrolladas no se documentan en este informe, pues se considera

que la sintáxis de los unit test es trivial y el código fuente es autoexplicativo. Para saber qué archivos contienen las pruebas y cómo ejecutarlas, véase el anexo 8.5.

7. Conclusión

El trabajo se ha completado habiendo cumplido todos los objetivos establecidos.

- Se ha realizado el análisis de software
- Se ha elaborado el diseño de software
- Se ha diseñado la base de datos
- Se han implementado todos los requisitos iniciales
- Se han creado varias baterías de prueba para testear la aplicación

El proyecto ha transcurrido según la planificación establecida. Todos los requisitos elaborados en la primera etapa de la planificación han sido implementados, y algunas modificaciones y mejoras han sido añadidas durante la etapa de creación del software. La aplicación desarrollada se ciñe al diseño establecido en la segunda etapa de la planificación, y es totalmente funcional.

7.1. Posibles ampliaciones

A continuación se listan algunas sugerencias para ampliar la funcionalidad de la aplicación en un hipotético proyecto futuro.

- Implementar nuevas estadísticas.
- Añadir soporte para APIs REST.
- Indicar la estructura HTML de los periódicos en archivos de configuración, en lugar de embeberla en el código fuente.

8. Anexos

8.1. Instalación

Junto con los archivos del proyecto se incluye un script Python ejecutable, install.py, que permite instalar la aplicación en sistemas Linux y Windows. El script deberá ejecutarse desde el directorio del proyecto y con privilegios de administrador/root. Si se quiere comprobar la ubicación de los archivos sin realizar modificaciones se puede invocar de la siguiente forma:

```
install.py --files
```

Una vez instalada la aplicación, y antes de comenzar a probarla, deberá editarse la configuración relativa a la base de datos; puede consultarse el manual de usuario en el anexo 8.2. Para crear una base de datos con artículos de prueba consultar el anexo 8.4.

8.2. Manual de usuario

La aplicación consta de dos comandos de consola: newscollect y newstats.

El comando newscollect pone en marcha el proceso de recolección de artículos. Para poder funcionar deberá leer su configuración de un archivo de texto; si la lectura falla o la configuración no es válida, el programa terminará. El comando puede ser invocado con un argumento o sin argumentos. Si se invoca con un argumento, este indicará la ruta al archivo donde se leerá la configuración. Si se invoca sin argumentos, newscollect buscará la configuración en un archivo por defecto. La ubicación por defecto del archivo de configuración para las distintas plataformas es la siguiente:

- Linux /etc/newscollect
- Windows %ProgramFiles%\newslib\newscollect

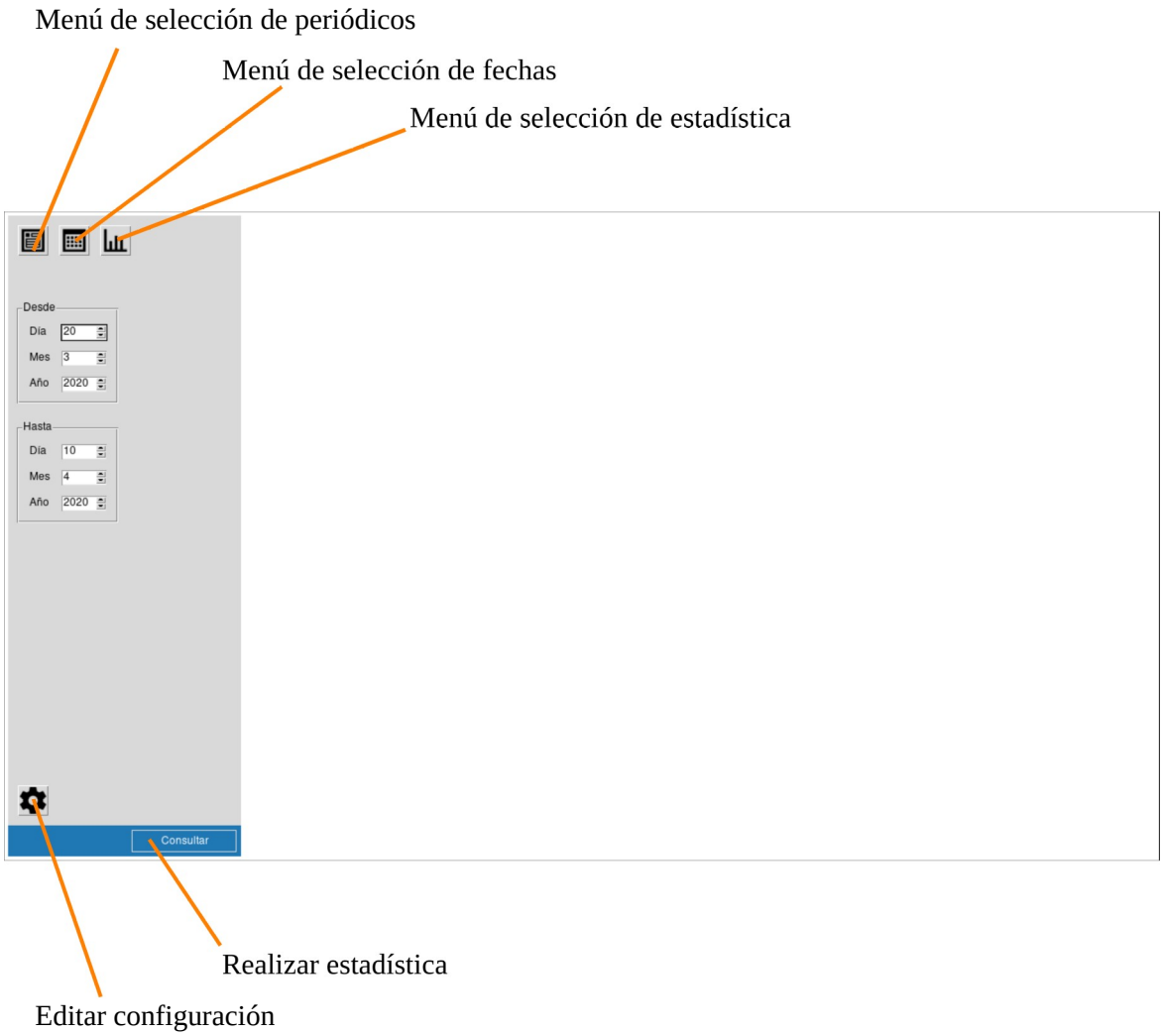
El comando newscollect está pensado para ejecutarse como un proceso en segundo plano, por lo que una vez iniciado se mantendrá funcionando de forma indefinida hasta que el usuario lo detenga. La forma de detenerlo es enviando la señal de parada; esto se puede lograr pulsando la combinación de teclas Ctrl + C desde el terminal en el que ha sido invocado. Si la aplicación se ha instalado en un equipo Linux, newscollect puede ponerse en marcha como un servicio systemd utilizando el comando systemctl. De lo contrario, newcollect deberá ejecutarse con privilegios root —la aplicación realmente no necesita privilegios para funcionar; esto es así por como se han estructurado los archivos y directorios—.

El comando newstats pone en marcha la interfaz gráfica de usuario, y se invoca sin ningún argumento. Guarda su configuración en un archivo de texto; este archivo puede ser editado manualmente o desde la propia interfaz gráfica. La ubicación del archivo de configuración para las distintas plataformas es la siguiente:

- Linux ~/.config/newstats/config
- Windows %APPDATA%\Local\newstats\config

La interfaz gráfica cuenta con tres menús en el panel lateral, desde los cuales pueden introducirse todos los parámetros relativos a la realización de estadísticas. Los menús se pueden navegar con los botones de la parte superior. La configuración de la base de datos puede editarse haciendo clic en el botón de ajustes de la parte inferior.

Una vez introducidos todos los parámetros, haciendo clic sobre el botón “Consultar” de la parte inferior aparecerá el diagrama. Si las fechas introducidas no son válidas —por ejemplo, si se especifica el día 30 para el mes de febrero— se mostrará un mensaje de error y la estadística no se realizará.



8.2.1. Configuración

En esta sección se detalla el formato empleado por los archivos de configuración de la aplicación. Los archivos deberán adecuarse al lenguaje TOML v1.0. Para ver ejemplos de configuración, consultar el anexo 8.3.

La configuración de newscollect consta de las secciones *database* y *modules*. La sección *database* contiene parámetros relativos a la conexión con la base de datos; la sección *modules* contiene parámetros relativos a la utilización de módulos o plugins de periódicos.

Sección `[database]`

`address` Dirección IP o nombre de dominio de la base de datos [cadena de texto].

`port` Puerto de la base de datos. Este parámetro es opcional; si no se incluye se utilizará el puerto por defecto para MySQL [número entero].

`name` Nombre de la base de datos [cadena de texto].

`user` Cuenta de usuario para la base de datos [cadena de texto].

`password` Contraseña para la cuenta de usuario. Este parámetro es opcional; si no se incluye se asumirá que la cuenta no utiliza contraseña [cadena de texto].

Sección `[modules]`

`src` Directorio que contiene los módulos de los periódicos. La ruta puede ser absoluta o relativa. Si se utiliza una ruta relativa, el programa deberá ejecutarse desde el directorio adecuado [cadena de texto].

`log` Directorio donde almacenar los archivos log de los periódicos. La ruta puede ser absoluta o relativa. Si se utiliza una ruta relativa, el programa deberá ejecutarse desde el directorio adecuado [cadena de texto].

`use` Listado de módulos a utilizar. Por cada entrada de la lista se buscará, en el directorio especificado en `src`, un archivo cuyo nombre sea la entrada de la lista con la extensión `.py` [vector de cadenas de texto]

La configuración de newstats consta de una sola sección, *database*, la cual contiene parámetros relativos a la conexión con la base de datos.

Sección `[database]`

`address` Dirección IP o nombre de dominio de la base de datos en la que consultar los

artículos [cadena de texto].

`port` Puerto de la base de datos. Este parámetro es opcional; si no se incluye se utilizará el puerto por defecto para MySQL [número entero].

`name` Nombre de la base de datos [cadena de texto].

`user` Cuenta de usuario para la base de datos [cadena de texto].

`password` Contraseña para la cuenta de usuario. Este parámetro es opcional; si no se incluye se asumirá que la cuenta no utiliza contraseña [cadena de texto].

8.3. Ejemplos de configuración

Ejemplo de archivo de configuración para newscollect.

```
[database]
address  = "10.0.0.20"
port     = 3306
name     = "articulos"
user     = "atanarico"
password = "1234"

[modules]
src = "/var/lib/newscollect/newspapers"
log = "/var/lib/newscollect/log"
use = [
  "elpais",
  "elmundo"
]
```

Ejemplo de archivo de configuración para newstats.

```
[database]
address  = "localhost"
port     = 3306
name     = "articulos"
user     = "chindasvinto"
password = "1234"
```

8.4. Base de datos

Junto con los archivos del proyecto se incluye un directorio sql, que contiene scripts relacionados con la base de datos. Los scripts tables.sql y users.sql han sido los utilizados para, respectivamente, crear las tablas y crear los usuarios de la base de datos utilizada durante el desarrollo. El script backup.sql contiene una copia de todo el contenido que se ha ido almacenando durante la elaboración del proyecto, y ha sido creado mediante el comando mysqldump.

Para replicar la base de datos y poder probar la aplicación se puede proceder de la siguiente forma (desde la línea de comandos de MySQL):

```
create database prueba;
use prueba;
source backup.sql
```

Si la base de datos se crea manualmente es importante respetar la estructura de las tablas, y que sus nombres sean *newspapers* y *articles* —tabla de periódicos y tabla de artículos, respectivamente —. De lo contrario la aplicación lanzará un error cuando se ejecute.

8.5. Ejecución de pruebas

Los archivos de pruebas se encuentran en el directorio test. Si la aplicación está instalada en el equipo, para lanzar las pruebas basta con ejecutar los archivos Python que comienzen con el prefijo test-, con la salvedad de aquellos que se indican a continuación.

El archivo testguiadapter.py contiene los unit test de la clase GUIAdapter. Antes de ejecutarse deberá sustituirse la librería dbmanager.py por el archivo test/dbmanager.py, el cual contiene una clase que simula a DBManager. Dependiendo de la plataforma en la que se haya instalado la aplicación la ubicación de dbmanager.py será diferente; esta puede comprobarse mediante los scripts de instalación.

El archivo testdbmanager.py, que contiene los unit test para la clase DBManager, requiere que previamente se haya puesto en marcha un contenedor Docker con la base de datos de prueba. Asumiendo que se trabaja sobre un equipo Linux y que Docker está instalado, las instrucciones para ejecutar este test son las siguientes:

1. Situarse en el directorio test/docker y ejecutar el script launch-db.sh
2. Esperar a que el contenedor acepte conexiones. Para comprobar esto se puede utilizar el comando docker logs
3. Ejecutar testdbmanager.py

Cada vez que se quiera repetir el test deberá volver a ejecutarse el script launch-db.sh.

8.6. Dependencias

A continuación se listan los módulos Python empleados por la aplicación. Los módulos pueden instalarse mediante el gestor de paquetes del sistema o mediante el comando pip; si se utiliza pip en Linux es importante hacer la instalación de forma global —ejecutando con privilegios root— para que la aplicación pueda importarlos independientemente del usuario que se utilice. Junto con cada módulo se indica la versión utilizada durante el desarrollo; no se garantiza que la aplicación funcione con versiones anteriores, aunque lo más probable es que esto no suponga ningún problema.

Tabla 10: Listado de módulos Python

Módulo	Paquete Ubuntu	Versión
bs4	python3-bs4	0.0.1
matplotlib	python3-matplotlib	3.2.1
pymysql	python3-pymysql	0.9.3
tkinter	python3-tk	8.6.10
toml	python3-toml	0.10.0
urllib3	python3-urllib3	1.25.9

El módulo tkinter suele venir instalado por defecto con Python. Para realizar las pruebas se ha utilizado el módulo unittest, que también suele venir instalado por defecto.

9. Referencias

Beautiful Soup, https://www.crummy.com/software/BeautifulSoup/
Último acceso: 1 septiembre 2020

Bob Hughes y Mike Cotterell (2009), *Software Project Management*, 5ª ed. McGraw-Hill

David Rojo, Anna Ginès y Neus Margalló (2019), *Estado del mercado laboral en España*, Infojobs

Docker, https://www.docker.com/
Último acceso: 1 septiembre 2020

MariaDB, https://mariadb.org/
Último acceso: 1 septiembre 2020

Matplotlib, https://matplotlib.org/
Último acceso: 1 septiembre 2020

PyMySQL, https://pypi.org/project/PyMySQL/
Último acceso: 1 septiembre 2020

Python, https://www.python.org/
Último acceso: 1 septiembre 2020

MariaDB Docker image, https://hub.docker.com/_/mariadb/
Último acceso: 1 septiembre 2020

M. Fowler (2002), *Patterns of Enterprise Application Architecture*, 1ª ed. Addison-Wesley

Style Guide for Python Code, https://www.python.org/dev/peps/pep-0008/
Último acceso: 1 septiembre 2020

Tkinter, https://docs.python.org/3/library/tkinter.html
Último acceso: 1 septiembre 2020

TOML, https://github.com/toml-lang/toml
Último acceso: 1 septiembre 2020

TOML Python, https://pypi.org/project/toml/
Último acceso: 1 septiembre 2020

unittest, https://docs.python.org/3/library/unittest.html
Último acceso: 1 septiembre 2020

urllib3, https://pypi.org/project/urllib3/
Último acceso: 1 septiembre 2020