ANEXO IV – DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Septiembre 2023

ÍNDICE

LISTA DE ILUSTRACIONES

1. INTRODUCCIÓN

En este anexo se describirán las características técnicas del proyecto. Como se ha mencionado en la memoria principal, el proyecto ha sido realizado en el lenguaje de programación Python y la página web se ha utilizado HTML y CSS.

Primeramente, se mostrará la estructura de los archivos de código fuente de las dos partes del desarrollo del proyecto, el algoritmo de *web scraping* y la página web.

Posteriormente se analizará en profundidad los distintos ficheros de código que conforman ambas partes, explicando las distintas clases, con sus métodos y atributos, sus cometidos y características generales a tener en cuenta.

También se detallará el todo el proceso de instalación que tiene que llevarse a cabo para poner en ejecución todo el proyecto, indicando qué programas, lenguajes o sistemas son necesarios para que cualquiera pueda ponerlo en marcha.

2. ESTRUCTURA DEL PROYECTO

En este apartado se mostrará la estructura de directorios en la que está dividido el proyecto. Hay dos partes diferenciadas, cada una con su árbol de directorios y ficheros propio, la parte de *web scraping* por un lado y la parte de aplicación web por otro.

El código fuente consiste en todos los archivos .py que están en el árbol de directorios.

2.1. Estructura ficheros del algoritmo de *web scraping*

En la siguiente imagen viene indicada la estructura de ficheros que llevan a cabo todo el algoritmo de *web scraping* que obtiene los catálogos de las marcas de Zara, H&M y Carhartt. Como se ha mencionado en la memoria principal del proyecto, para realizar este algoritmo se ha utilizado el *framework* de Python Scrapy, especializado en *web scraping*.

Se ha organizado el código en directorios englobando los elementos arquitectónicos comunes de Scrapy, es decir, las arañas se agrupan en el directorio *spiders*, las clases de *Item Pipelines* en el directorio *pipelines*, los *middlewares* en el directorio *middlewares*, las clases de ítems en *ítems*.

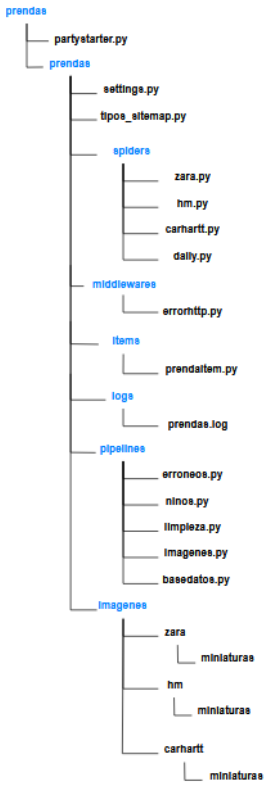

Ilustración 1 - Estructura de ficheros del web scraping

2.2. Estructura ficheros de la página web

En la siguiente imagen viene indicada la estructura de Ficheros que llevan a cabo el funcionamiento de la página web. Para el desarrollo de la página web se ha utilizado, como se ha mencionado en la memoria principal, el *framework* de Python Flask, especializado en desarrollo de páginas web.

El código fuente consiste en todos los archivos .py.

La estructura del código que se ha seguido es en el módulo principal, *galería_gestion*, los ficheros de código de gestión de la aplicación web y en la carpeta *templates* los ficheros HTML.

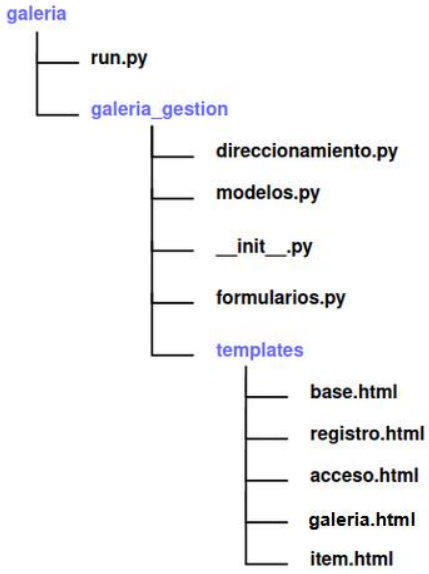

Ilustración 2 - Estructura de ficheros de la aplicación web

3. BIBLIOTECAS EXTERNAS

Para la realización de cualquier proyecto *software* está a la orden del día el uso de bibliotecas externas no desarrolladas por el usuario para poder realizar todo tipo de tareas. En este apartado se indicarán qué bibliotecas han sido usadas y un enlace a sus respectivas páginas web de documentación:

- SQLAlchemy: se trata de una biblioteca cuya función es la de abstraer el acceso a bases de datos relacionales que usen SQL usando lenguaje Python en vez de sentencias nativas de SQL. https://docs.sqlalchemy.org/
- re: es una biblioteca especializada en el uso de expresiones regulares. https://docs.python.org/3/library/re.html

4. WEB SCRAPING

Dada la organización de ficheros indicada en "Ilustración 1" se irán mostrando por partes la documentación técnica empezando por las *Spiders* o arañas, posteriormente con *Ítem Pipelines* o clases de procesado de los ítems, a continuación, los *Middleware* o clases de procesado de peticiones y respuestas y se acabará con los *Scripts* que se han utilizado, tanto los de investigación de las webs como el de ejecución de las arañas. Como se ha podido ver en la estructura del proyecto, cada una de estas partes tiene su directorio asociado excepto los *scripts*.

4.1. Spiders

- Clase *ZaraSpider*:
    - Fichero: zara.py.
    - Descripción: araña que recorre la web de Zara a través del *sitemap*.
    - Hereda de: *SitemapSpider*.
    - Atributos:
        - *nombre* = "Zara": nombre de la araña para gestión de Scrapy.
        - *allowed_domains* = ["zara.com"]: dominio permitido para que recorra la araña.
        - *sitemap_urls* = ['https://www.zara.com/sitemaps/sitemap-es-es.xml.gz']: URL del *sitemap* a recorrer.
        - *sitemap_rules* = [('^((?!/help|-mkt[0-9]+|-pG[0-9]+|-pT[0-9]+|-p[1-9]|look-|kids|nino|girl|boy|baby|pendiente|abalorio|accessori|accesori|pack|boxer|mochila|calcetin|sombra|bolso|gafa|lip-|zapat|shoe|mini--|sujetador|refill|wallpaper|perfume|beachwear|beauty|lingerie|bag).)*$', 'parse')]: palabras que en caso de contener cada URL del *sitemap* no se le enviará petición a dicha URL.
        - *custom_settings* = {"IMAGES_STORE": 'imagenes/zara'}: configuración de la araña. Directorio donde se guardarán las imágenes descargadas de los productos de Zara.
    - Métodos:
        - *__init__():*
            - Descripción: constructor de *ZaraSpider*.
            - Parámetros:
                - *productos_bd*, *list*: lista de URLs de los productos de Zara almacenados en la base de datos.

- *sitemap_filter():*
  - Descripción: Método cuya función es la de recorrer el *sitemap* y hacer un filtrado de las distintas URLs. El filtrado de entradas se hace en base a la etiqueta XML 'changefreq', de forma que diariamente se recorran las URLs con valor 'daily', en este caso las tipo -l y semanalmente los lunes las tipo -p. Se aplican primeramente los filtros de sitemap_rules y posteriormente se ejecuta este método.
  - Parámetros
    - *entries*,: elementos URL del *sitemap*.
  - Devuelve:
    - *entry*,: elementos URL del *sitemap* cuya URL es tipo -l o tipo -p.
- *parse()*:
  - Descripción: Procesa la respuesta de cada petición realizada a las URLs provenientes de sitemap_filter(). Los enlaces tipo -p se procesan como productos y los tipo -l como listas de productos.
  - Parámetros:
    - *response*, *Response*: respuesta que contiene el HTML de un producto o de una lista de productos.
  - Devuelve:
    - *PrendaItem* o *Request*: ítem en caso de ser una URL tipo -p o petición de un producto si es una URL tipo -l.
- *parse_lista():*
  - Descripción: Se obtienen todos los enlaces tipo -p, es decir, todos los productos del enlace tipo -l, y se envían peticiones a dichos enlaces -p cuya respuesta será procesada en parse_producto(). Estas listas -l se analizan de forma diaria.
  - Parámetros:
    - *response*, *Response*: respuesta que contiene un HTML de lista de productos.
  - Devuelve:
    - *Request*: petición a enlaces tipo -p de productos.

- *parse_productos():*
  - Descripción: Se procesa cada página de producto. Se hace parsing para obtener todos los campos menos Descripción.
  - Parámetros:
    - *response*, *Response*: respuesta que contiene el HTML de un producto.
  - Devuelve:
    - *PrendaItem*: ítem poblado con todos los campos menos descripción.

- Clase HmSpider
  - Fichero: hm.py
  - Descripción: araña que recorre la web de H&M a través de la API JSON.
  - Hereda de: *Spider*.
  - Atributos:
    - *name*= "hm": nombre de la araña para gestión de Scrapy.
    - *allowed_domains* = ["hm.com"]: dominios que le está permitido recorrer.
    - *start_urls* = ["https://www2.hm.com/es_es/hombre/compra-por-producto/view-all.html", "https://www2hm.com/es_es/mujer/compra-por-producto/view-all.html"]: URLS a las que se mandará petición primero.
    - *custom_settings* = {"IMAGES_STORE": 'imagenes/hm'}: ajustes de la araña. Directorio donde se guardarán las imágenes descargadas de los productos de H&M.
  - Métodos:
    - *parse():*
      - Descripción: se gestiona la respuesta a las URLs de start_urls. Se obtienen el número de ítems de hombre o de mujer y se preparan parámetros para llamar a la API JSON.
      - Parámetros:
        - *response*, *Response*: respuesta tipo *Response* a un objeto *Request* de las URLs de *start_urls*.

- Devuelve:
    - *list*: lista de *Request* de peticiones a la API JSON.

- *crear_peticiones_json()*:
    - Descripción: se crean las URLs de la API JSON que devuelven todos los ítems de hombre y de mujer y se realizan dichas peticiones.
    - Parámetros:
        - *genero*, *string*: cadena que contiene el género, puede ser hombre o mujer.
        - *prefijo_genero*, *string*: prefijo asociado al género, 'fa5b' para hombre o '30ab' para mujer.
        - *numero_articulos*, *int*: número máximo de artículos de hombre o de mujer que hay en H&M.
        - *tamano_bloque*, *int*: número de productos que contendrá el archivo JSON.
    - Devuelve:
        - *list*: lista de *Request* por partes a la API JSON hasta cubrir el número total de productos de cada género.

- *parse_item_but_description():*
    - Descripción: Se hace *parsing* de un producto obteniendo de los Atributos de los ítems del archivo JSON todos los campos menos descripción y composición, que no se encuentran en ningún atributo. Para obtener estos se hará una petición a la URL del producto.
    - Parámetros:
        - *response*, *Response*: respuesta que contiene un archivo JSON con productos.
        - *genero*, *string*: género, hombre o mujer, al que pertenecen los ítems del archivo JSON.
    - Devuelve:
        - *Request*: petición de un producto.

- *parse_item_description_composition():*
    - Descripción: Obtención de los campos de descripción y composición. Se termina de poblar el ítem y se devuelve.

- Parámetros:
  - *response*, *Response*: respuesta que contiene el HTML de un producto.
  - *callback_item*, *PrendaItem*: ítem que le faltan los campos de descripción y composición.
- Devuelve:
  - *PrendaItem*: ítem poblado con todos los campos.

- Clase *CarharttSpider*:
  - Fichero: carhartt.py.
  - Descripción: Araña que recorre la web de Carhartt WIP a través del *sitemap*.
  - Hereda de: *SitemapSpider*.
  - Atributos:
    - *name* = "carhartt": nombre de la araña para gestión de Scrapy.
    - *allowed_domains* = ["carhartt-wip.com"]: dominios que le está permitido recorrer.
    - *sitemap_urls* = ['https://www.carhartt-wip.com/es/sitemap-products.xml']: URL del *sitemap* a recorrer.
    - *sitemap_rules* = [('^((?!accesorio|gadget|ropa-interior|zapato).)*$', 'parse')]: palabras que en caso de contener cada URL del *sitemap* no se le enviará petición a dicha URL.
    - *custom_settings* = {"IMAGES_STORE": 'imagenes/carhartt',}: configuración de la araña. Directorio donde se guardarán las imágenes descargadas de los productos de Carhartt.
  - Métodos:
    - *parse():*
      - Descripción: Se procesa cada página de producto. Se hace *parsing* para obtener todos los campos.
      - Parámetros:
        - *response*, *Response*: respuesta que contiene el HTML de un producto.
      - Devuelve:
        - *PrendaItem*: ítem poblado con todos los campos.

4.2. Ítem Pipelines

- Clase *PipelineErroneos*:
    - Fichero: erroneos.py.
    - Descripción: clase de *Ítem Pipelines* cuya función es la de descartar a prendas que carezcan de algún campo como nombre, precio, url_imagen, género o descripción.
    - Excepciones:
        - *DropItem*: excepción que se lanza para descartar un ítem.
    - Métodos:
        - *process_item():*
            - Descripción: función de procesamiento del ítem. Descarta un ítem en caso de que carezca de algún campo.
            - Parámetros:
                - *item*, *PrendaItem*: ítem a procesar.
                - spider, Spider: araña asociada a esta clase de Ítem Pipeline.
            - Devuelve:
                - *PrendaItem*: se devuelve el ítem para que siga pasando por las clases de *Item Pipelines*.
- Clase *PipelineNinos*:
    - Fichero: ninos.py.
    - Descripción: clase de *Item Pipelines* cuya función es la de descartar a prendas de Zara que sean de niños.
    - Excepciones:
        - *DropItem*: excepción que se lanza para descartar un ítem.
    - Métodos:
        - *process_item():*
            - Descripción: Función de procesamiento del ítem. Descarta un ítem si contiene el valor 'NIÑOS' en el campo de género, como ocurre con los ítems de niños de Zara.
            - Parámetros:
                - *item*, *PrendaItem*: ítem a procesar.

        - *spider*, *Spider*: araña asociada a esta clase de *Item Pipeline*.

    - Devuelve:

        - *PrendaItem*: se devuelve el ítem para que siga pasando por el resto de las clases de *Item Pipelines*.

- Clase *PipelineLimpieza*:

    - Fichero: limpieza.py.

    - Descripción: función de procesamiento del ítem. Se limpian y estandarizan los valores de los campos de los productos para que todos tengan el mismo formato y sean más fáciles de procesar.

    - Métodos:

        - *process_item():*

            - Descripción: función de procesamiento del ítem. Llama a distintos métodos para que hagan labores de limpieza y estandarización de los campos de los ítems.

            - Parámetros:

                - *item*, *PrendaItem*: ítem a procesar.

                - *spider*, *Spider*: araña asociada con esta clase de *Item Pipeline*.

            - Devuelve:

                - *PrendaItem*: se devuelve el ítem para que siga pasando por el resto de clases de *Item Pipelines*.

        - *crear_id():*

            - Descripción: crea el id de cada ítem con los últimos 10 dígitos del número entero producido por el algoritmo de *hashing* MD5. Se utiliza como elemento de entrada para hacer el hash la URL del producto ya que es única.

            - Parámetros:

                - url, string: URL del producto.

            - Devuelve:

                - *int*: identificador del producto.

- *limpieza_composicion():*
  - Descripción: se asigna el valor 'sin_composicion' a los productos que no tengan composición y se eliminan los caracteres inútiles y se normaliza el valor de composición de los productos de Carhartt porque contiene caracteres *unicode* extraños.
  - Parámetros:
    - *item*, *PrendaItem*: ítem a procesar.
  - Devuelve:
    - *string*: composición del ítem.
- *limpieza_nombre():*
  - Descripción: si el producto no es de Carhartt se pone en formato xx y en caso de ser Carhartt se añade una traducción del tipo de producto que es al nombre pues este se encuentra en inglés
  - Parámetros:
    - *item*, *PrendaItem*: ítem a procesar.
  - Devuelve:
    - *string*: nombre del producto.
- *limpieza_color():*
  - Descripción: de la cadena obtenida del XPath se extrae el color y en caso de haber varios, el más importante.
  - Parámetros:
    - *color_entrada*, *string*: cadena que contiene el color.
  - Devuelve:
    - *string*: color extraído de *color_entrada*.
- *asignar_color():*
  - Descripción: a la cantidad de colores de entrada que hay, se le asigna un color predefinido.
  - Parámetros:
    - *color_entrada*, *string*: color aislado.
  - Devuelve:
    - *string*: color de los tipos prefijados.

- Clase *PipelineImagenes*:
  - Fichero: imágenes.py.
  - Descripción: función de procesamiento del ítem. Se encarga de descargar las imágenes cuyas URL contiene cada ítem.
  - Excepciones:
    - *DropItem*: excepción que se lanza para descartar un ítem.
  - Métodos:
    - *get_media_requests():*
      - Descripción: Obtiene las URLs de las imágenes de un ítem.
      - Parámetros:
        - *item*, *PrendaItem*: ítem a procesar.
        - *info*,: información de la descarga.
      - Devuelve:
        - *Request*: petición de descarga de la imagen del campo url_imagen.
    - *generador_ruta_imagen():*
      - Descripción: Genera el nombre de la ruta desde el directorio IMAGES_STORE.
      - Parámetros:
        - *miniaturas*, *string*: cadena que contiene o 'miniaturas' o None.
        - *item*, *PrendaItem*: ítem a procesar.
      - Devuelve:
        - *string*: ruta de la miniatura o imagen desde el directorio IMAGES_STORE.
    - *file_path():*
      - Descripción: devuelve el nombre de la ruta donde está descargada la imagen del producto.
      - Parámetros:
        - *request*, *Request*: solicitud del producto.
        - *response*, *Response*: respuesta del producto, valor por defecto None.

- *info*,: información de la descarga, valor por defecto None.
- *item*, *PrendaItem*: *PrendaItem*, valor por defecto None.

- Devuelve:
  - *string*: nombre de la ruta de la imagen.

- *thumb_path():*
  - Descripción: devuelve el nombre de la ruta donde está descargada la miniatura del producto.
  - Parámetros:
    - *request*, *Request*: solicitud del producto.
    - thumb_id, int: identificador de la miniatura.
    - *response*, *Response*: respuesta del producto, valor por defecto None.
    - *info*,: información de la descarga, valor por defecto None.
    - *item*, *PrendaItem*: ítem a procesar, valor por defecto None.
  - Devuelve:
    - *string*: nombre de la ruta de la minitatura.
- *item_completed():*
  - Descripción: se llama a este método cuando se han descargado todas las imágenes de un ítem.
  - Parámetros:
    - *results*,: resultados de la descarga.
    - *item*, *PrendaItem*: ítem a procesar.
    - *info*,: información de la descarga.
  - Devuelve:
    - *PrendaItem*: se devuelve el ítem para que siga pasando por las clases de *Item Pipelines*.

- Clase *PipelineBaseDatos*:
  - Fichero: basedatos.py.

- Descripción: clase de Ítem Pipelines cuya función es la de insertar los ítems en la base de datos.
- Excepciones:
  - *DropItem*: excepción que se lanza para descartar un ítem.
- Métodos:
  - *open_spider():*
    - Descripción: método que se activa al abrirse la araña asociada a esta clase de *Item Pipelines*. Su función es la de crear una conexión a la base de datos para posteriormente acceder a ella.
    - Parámetros:
      - *spider*, *Spider*: araña asociada a esta clase de *Item Pipelines*.
  - *close_spider():*
    - Descripción: método que se activa al cerrarse la araña asociada a esta clase de *Item Pipelines*. Su función es la de cerrar la conexión a la base de datos de forma segura.
    - Parámetros:
      - *spider*, *Spider*: araña asociada a esta clase de *Item Pipelines*.
  - process_item():
    - Descripción: función de procesamiento del ítem. Se inserta un ítem en la base de datos.
    - Parámetros:
      - *item*, *PrendaItem*: ítem a procesar.
      - *spider*, *Spider*: araña asociada a esta clase de *Item Pipeline*.
    - Devuelve:
      - *PrendaItem*: se devuelve el ítem para que siga pasando por las clases de *Item Pipelines*.

4.3. Ítems

- Clase *PrendaItem*:
  - Fichero: prendaitem.py.
  - Descripción: clase que representa una prenda de ropa.
  - Hereda de: *Item*.
  - Atributos:
    - *id*, *int*: identificador de la prenda.
    - *nombre*, string: nombre de la prenda.
    - *precio*, *float*: precio de la prenda.
    - *url_imagen*, *string*: URL de la imagen de la prenda.
    - *ruta_imagen*, *string*: ruta de la imagen en el sistema de archivos.
    - *ruta_miniatura*, *string*: ruta de la miniatura de la imagen en el sistema de archivos.
    - *genero*, *string*: genero de la prenda.
    - *color*, *string*: color de la prenda.
    - *descripcion*, *string*: descripción de la prenda.
    - *marca*, *string*: marca de la prenda.
    - *composicion*, *string*: composición de la prenda.
    - *url*, *string*: URL de la prenda.
    - *fecha*, *datetime*: fecha de obtención de la prenda.

4.4. Middlewares

- Clase *ErrorHttpDownloaderMiddleware*:
  - Fichero: errorhttp.py.
  - Descripción: *DownloaderMiddleware* cuya función es la de cerrar una araña si esta ha recibido más de *max_errores* códigos HTTP 403 seguidos porque si se han recibido tantos significa que se le ha prohibido el acceso a la página.
  - Atributos:
    - *contador_zara*, *int*: número de errores 403 seguidos ha recibido la araña Zara.
    - *contador_hm, int*: número de errores 403 seguidos ha recibido la araña H&M.

- *contador_carhartt*, *int*: número de errores 403 seguidos ha recibido la araña Carhartt.
- *previo_zara*, *boolean*: si en la respuesta anterior Zara ha recibido un código 403.
- *previo_hm*, *boolean*: si en la respuesta anterior H&M ha recibido un código 403.
- *previo_carhartt*, *boolean*: si en la respuesta anterior Carhartt ha recibido un código HTTP 403.
- *max_errores*, *int*: número máximo de errores 403 que se pueden recibir seguidos antes de cerrar la araña correspondiente.

- Métodos:
  - *__init__():*
    - Descripción: constructor de *ErrorHttpDownloaderMiddleware*.
  - *process_response*():
    - Descripción: función de procesamiento de respuestas de los Middlewares. Llama a la función *gestion_codigo_error()* para analizar la respuesta que ha recibido la araña spider y se actualizan los valores de contador de errores 403 y el estado previo del error HTTP.
    - Parámetros:
      - *request*, *Request*: petición asociada con la respuesta recibida.
      - *response*, *Response*: respuesta recibida.
      - *spider*, *Spider*: araña que ha recibido la respuesta.
    - Devuelve:
      - *Response*: respuesta recibida.
  - *gestion_codigo_http():*
    - Descripción: si se reciben *max_errores* códigos HTTP 403 seguidos, se cierra la araña que ha recibido dicho código porque se le ha prohibido el acceso a la página web. En caso de recibir un código 200 se reinicia el contador.
    - Parámetros:
      - *estatus_http*, *int*: código HTTP recibido en la última respuesta.

- *contador*, *int*: contador de códigos HTTP 403.
- *previo*, *boolean*: indica si en la previa respuesta se recibió un código 403 (True) o no (False).
- *spider*, *Spider*: araña que ha recibido la respuesta.

- Devuelve:
  - *int*: se devuelve el contador para actualizarlo.
  - *Boolean*: se devuelve el estado para actualizarlo.

4.5. Scripts

- Nombre: partystarter.py:
  - Descripción: ejecuta las arañas Zara, H&M y Carhartt.
  - Excepciones:
    - *DropItem*: excepción que se lanza para descartar un ítem.
  - Funciones:
    - *create_db_connection():*
      - Descripción: Crea la conexión con la base de datos
      - Devuelve:
        - *conn*,: objeto de conexión a la base de datos.
        - *cursor*,: cursor de la conexión a la base de datos.
    - *get_zara_database_urls():*
      - Descripción: se obtienen las URLs de todos los productos de Zara de la base de datos.
      - Parámetros:
        - *cursor*,: cursor de la base de datos.
      - Devuelve:
        - *list*: lista de URLs de productos de Zara.
    - *close_db():*
      - Descripción: se cierra la conexión con la base de datos.
      - Parámetros:
        - *conn*,: conexión con la base de datos.
        - *cursor*,: cursor de la base de datos.

        - *main():*
            - Descripción: Se obtienen las URLs de los productos de Zara, se pasan a la araña de Zara por parámetro y se ejecutan las arañas de Zara, H&M y Carhartt. Estas dos últimas de forma semanal los lunes.

- Nombre: tipos_sitemap.py
    - Descripción: lista el número de URLs de cada tipo para un *sitemap* dado.
    - Funciones:
        - *analizador_sitemap():*
            - Descripción: lista el número de URLs de cada tipo para un *sitemap* dado.

- Nombre: daily.py
    - Clase: *DailySpider*
    - Hereda de: *Spider*
    - Descripción: araña que recorre los tipos de URLs tipo -l y tipo -pT y analiza si los productos que contienen estas páginas se encuentran en el *sitemap* o no.
    - Métodos:
        - *from_crawler():*
            - Descripción: método de instanciación de la araña que registra las señales de abrir araña, *opened_spider* y araña cerrada, *closed_spider*.
            - Parámetros:
                - *crawler*, *Crawler*: clase base de la que parten todos los componentes de Scrapy.
            - Devuelve:
                - *spider*, *Spider*: araña a crear.
        - *spider_opened():*
            - Descripción: abre el acceso al fichero.
            - Parámetros:
                - *spider*, *Spider*: araña que ha sido abierta.
        - *spider_closed():*
            - Descripción: se cierra el acceso al fichero.

- Parámetros:
  - *spider*, *Spider*: araña que ha sido cerrada.

- *parse():*
  - Descripción: se elaboran listas de enlaces tipo -p, -l y -pT y para ser gestionadas posteriormente.
  - Parámetros:
    - *response*, *Response*: respuesta a analizar.

- *chequear_en_sitemap():*
  - Descripción: se comprueba si hay algún enlace tipo -p de todas las URLs extraídas entre el total de enlaces tipo -p del *sitemap*.
  - Parámetros:
    - *total_productos*, *list*: total de enlaces tipo -p del *sitemap*.
    - *links*, *list*: conjunto de enlaces tipo -p que están en el atributo *link_padre*
    - *link_padre*, *string*: enlace tipo -l o tipo -p que contiene los enlaces tipo -p.

- parse_l():
  - Descripción: se obtienen las URLs de los enlaces tipo -p en las páginas de tipo -l.
  - Parámetros:
    - *response*, *Response*: respuesta a analizar.

- parse_pT():
  - Descripción: se obtienen las URLs de los enlaces tipo -p de las páginas de tipo -pT.
  - Parámetros:
    - *response*, *Response*: respuesta a analizar.

5. PÁGINA WEB

Para tratar la documentación de la parte de página web, primero se mostrarán los distintos *scripts* del directorio *galería_gestion* que son los que se encargan del funcionamiento de la página web y posteriormente se profundizará en *run.py* que es el que se encarga de arrancar la aplicación web.

- Nombre: direccionamiento.py:
    - Descripción: contiene las diferentes rutas de la aplicación web y las funciones que se realizan para cada una.
    - Rutas:
        - *'/' o '/market':*
            - Descripción: muestra los productos de la plataforma pudiendo estar filtrados y los filtros. Los filtros existentes son, tipo de prenda, geénro, marca, color y composición. Su función asociada es *market_page()*.
            - Métodos HTTP:
                - GET: se solicita el listado total de productos.
                - POST: se solicita un listado de productos para los filtros asociados a esta petición.
            - Devuelve:
                - *market.html:* documento HTML con los productos.
        - *'/registro':*
            - Descripción: muestra una página de registro en la que el cliente puede crear una cuenta de usuario para la página. Contiene un formulario de campo como nombre, nombre de usuario, email, contraseña y confirmar contraseña.
            - Métodos HTTP:
                - GET: se solicita la página de registro.
                - POST: se solicita crear un perfil de usuario para los parámetros introducidos en el formulario.
            - Devuelve:
                - *registro.html:* documento HTML el formulario de registro.
        - *'/login':*
            - Descripción: procesa el inicio de sesión del usuario.

- Métodos HTTP:
  - GET: se solicita la página de registro.

- Devuelve:
  - *registro.html:* documento HTML el formulario de registro.

- *'/logout':*
  - Descripción: cierra sesión para el usuario que lo solicita.
  - Métodos HTTP:
    - GET: se solicita la página de registro.
- *'/<item_id>:*
  - Descripción: devuelve la página del ítem cuyo identificador es el que aparece en la ruta.
  - Métodos HTTP:
    - GET: se solicita la página del ítem.
  - Devuelve:
    - *item.html:* documento HTML el formulario de registro.
- *'/obtener_imagen:*
  - Descripción: devuelve la imagen solicitada.
  - Parámetros:
    - *id*: identificar del ítem cuya imagen se quiere obtener.
    - *miniatura*: si la imagen que se obtiene es miniatura. Puede tomar como valor 0, no se quiere obtener miniatura o 1 sí se quiere obtener miniatura.
  - Métodos HTTP:
    - GET: se solicita la imagen de un ítem.
  - Devuelve:
    - Imagen en formato JPG.

- Nombre: formularios.py.
  - Descripción: archivo que contiene los formularios que se usarán para registrar e iniciar sesión usuarios.

- Clases:
  - Nombre: *FormularioRegistro*
    - Hereda de: *FlaskForm*
    - Descripción: formulario del documento *registro.html* para registrar un usuario nuevo en el sistema.
    - Atributos:
      - *nombre_usuario*: contiene el nombre de usuario necesario para iniciar sesión. Como restricciones tiene que ser una cadena de entre 5 y 15 caracteres, es de relleno obligatorio y no se autocompleta.
      - *email*: contiene el correo electrónico del usuario. Contiene las validaciones típicas de un *email*, el arroba, dominio y nombre anterior a la arroba.
      - *contrasena*: campo que contiene la contraseña del usuario. Tiene que ser una cadena de mínimo 6 caracteres y es de relleno obligatorio.
      - *conf_contrasena*: campo que contiene la contraseña del usuario para confirmar que coincide con el atributo *contraseña* y asegurarse que no se introduce una contraseña errónea. Es de relleno obligatorio.
      - *aceptar*: botón que inicia la petición POST para registro del usuario con los atributos incluidos de *nombre_usuario, email, contraseña.*
    - Métodos:
      - *validate_nombre_usuario()*:
        - Descripción: comprueba que el nombre introducido al registrar un usuario no existe ya en la base de datos.
        - Parámetros:
          - *nombre_usuario_validar, string*: nombre de usuario que se quiere validar.
        - Excepciones:
          - *ValidationError*: excepción que se lanza en caso de que el nombre de usuario ya exista en la base de datos.

- *validate_email()*:
  - Descripción: comprueba que el email introducido al registrar un usuario no exista ya en la base de datos.
  - Parámetros:
    - *email_validar, string*: email que se quiere validar.
  - Excepciones:
    - *ValidationError*: excepción que se lanza en caso de que el email ya exista en la base de datos.

- Nombre: *FormularioAcceso*
- Hereda de: *FlaskForm*
- Descripción: formulario del documento *acceso.html* para iniciar sesión de un usuario.
- Atributos:
  - *nombre_usuario*: contiene el nombre de usuario que se va a validar para iniciar sesión. Campo que tiene que completarse obligatoriamente.
  - *contrasena*: contiene la contraseña del usuario que se va a validar para iniciar sesión. Campo que tiene que completarse obligatoriamente.
  - *aceptar*: botón para enviar petición POST al servidor con los campos *nombre_usuario* y *contraseña* para que sean verificador para iniciar sesión.

- Nombre: modelos.py.
  - Descripción: contiene las clases los modelos que se van a extraer de la base de datos, especificando a qué tabla se corresponden, sus columnas y el tipo de dato de cada columna.
  - Clases:
    - Nombre: *Usuario*.
      - Hereda de: *Model, UserMixin*.
      - Descripción: se trata del modelo del usuario almacenado en la página web.

- Atributos:
    - *id*: identificador del usuario. Se trata de un número tipo *int*, que es clave primaria y se forma mediante autoincremento.
    - *nombre_usuario:* nombre del usuario. Se trata de una cadena de caracteres único en toda la base de datos y con un máximo de 15 caracteres.
    - *email:* correo electrónico del usuario. Se trata de una cadena de caracteres de un máximo de 50 caracteres única en la base de datos.
    - *hash_contrasena:* digestión de la contraseña producida por un algoritmo de *hashing*. Se trata de una cadena de un máximo de 64 caracteres.
- Métodos:
    - *contrasena():*
        - Descripción: es una propiedad, es decir, un método público que devuelve la contraseña y que se accede a él como si fuera un atributo más de la clase.
        - Devuelve:
            - String: devuelve la contraseña del usuario.
    - *contrasena():*
        - Descripción: es un *setter* asociado a la propiedad *contrasena()* de forma que genera el *hash* de la contraseña que se le pasa como parámetro y se asocia al atributo de clase *hash_contrasena*.
        - Atributos:
            - Contraseña_texto_llano, string: contraseña del usuario.
    - *Comprobar_hash_contrasena()*:
        - Descripción: comprueba si el *hash* de la contraseña que se pasa como parámetro es igual al *hash_contrasena* del usuario al que se quiere acceder sesión.

- Atributos:
  - *Posible_contrasena*, string: contraseña con la que se quiere iniciar sesión.
- Devuelve:
  - *Boolean*: *True* si el *hash* de *posible_contrasena* es igual al *hash_contrasena* del usuario al que se quiere iniciar sesión y *False* en caso contrario.

- Métodos:
  - *Load_user():*
    - Descripción: se trata de un método *callback* cuya finalidad es revalidar la sesión si se ha iniciado sesión previamente.
    - Parámetros:
      - *User_id:* identificador del usuario.
    - Devuelve:
      - *Usuario*: objeto Usuario cuya sesión está activa.
  - Nombre: Item
    - Hereda de: *Model*.
    - Descripción: se trata del modelo de la clase *PrendaItem* que se ha utilizado en *web scraping* y que se utilizará ahora para manejar los ítems que se presenten en la aplicación web.
    - Atributos:
      - *Id:* identificador del ítem. Se trata de un tipo de dato *Bigint* que es clave primaria en la base de datos.
      - *Nombre:* nombre del ítem. Se trata de una cadena de un máximo de 100 caracteres.
      - *Precio:* precio del ítem. Es un número de tipo *float*.
      - *Genero:* género del ítem. Es una cadena de caracteres de un máximo de 10 caracteres.
      - *Color:* color del ítem. Cadena de caracteres de un máximo de 100 caracteres.
      - *Descripción:* descripción del ítem. Cadena de caracteres de un máximo de 800 caracteres.

- *Marca:* marca del ítem. Cadena de caracteres de un máximo de 10 caracteres.
- *url:* URL del ítem. Cadena de caracteres de máximo de 200 caracteres.
- *fecha:* fecha de extracción del ítem. Es una variable de tipo *DateTime*.
- *composición:* material del que está compuesto el ítem mayoritariamente. Es una cadena de caracteres con un máximo de 100 caracteres.
- *ruta_imagen:* ruta del servidor en la que se encuentra la imagen del producto. Es una cadena de caracteres de un máximo de 60 caracteres.
- *ruta_miniatura:* ruta del servidor en la que se encuentra la miniatura del producto. Es una cadena de caracteres de un máximo de 100 caracteres.

- Métodos:
  - *__repr__()*:
    - Descripción: devuelve una representación del ítem en cuestión en forma de cadena de caracteres.
    - Devuelve:
      - String: devuelve una cadena de caracteres del ítem en la que se especifica su *id* y su *nombre*.

- Nombre: __init__.py.
  - Descripción: importa los módulos necesarios para que los *direccionamiento.py, modelos.py* y *formularios.py* el módulo *galería_gestion*, que es el que define este fichero, funcionen correctamente e inicializa variables de entorno necesarias para el funcionamiento de la aplicación web, como la URI de la base de datos o el directorio de las imágenes.
- Nombre: run.py
  - Descripción: ejecuta la página web de forma que pueda usarse.