## Main

```
//Bucle principal del user-agent.
//Crea constantes, inicializa variables, obtiene el id a tratar e itera formando la url, y compobando excepciones
//y respuestas del servidor, sigue iterando o realizando el tratamiento de errores.

Inicializar cabecera del user-agent

Inicializar Serie de los elementos ya tratados a partir del json del dataset.

Inicializar intentos
Inicializar repetición

Crear lista de elementos a tratar(Serie elementos tratados, inicio, limite)
Obtener primer id

Mientras existan elementos en la lista, durante un número marcado de reintentos y si no se para la extracción:

        Formar la url [https://reports.exodus-privacy.eu.org/es/reports/ + id]

        Try
                Obtener el html [request.get(url, cabecera)]
        Capturar Error de conexión (ConnectionError)
                Gestionar tiempo(motivo = error conexión, intento)
                Repetir
        Capturar Error de timeout (Timeout)
                Gestionar tiempo(motivo = timeout, intento)
                Repetir
        Capturar Error de proxy
                Parar con aviso(motivo = error de proxy)
        Capturar Error SSLError
                Parar con aviso (motivo = error SSL)
        Cualquier otra cosa
                Parar con aviso (motivo = e)

        Comprobar request.status
                Entre 500 y 600 (Errores de servidor)
                        Avisar
                        Gestionar tiempo (motivo = error servidor, intento)
                        Repetir
                Entre 400 y 500 (Errores de cliente)
                        Avisar
                        Gestionar tiempo (motivo = error cliente, intento)
                        Repetir
                Entre 300 y 400 (Redirecciones)
                        Parar con aviso
                Entre 201 y 300
                        Avisar
                200
                        dict = Rastrear el html(html)
                        Añadir el diccionario json a series_elementos_tratados
                        Inicializar intentos
                        Si no es el ultimo:
                                Obtener nuevo id a tratar
                                Gestionar tiempo(motivo = ok, intento)
                        si es el último:
                                Parar con aviso

Fin mientras
```

## Crear lista elementos a tratar

```
//Devuelve una lista de enteros para iterar con aquellos elementos que no han
//sido tratados previamente o se han tratado con error desde un número inicial
//hasta un límite fijado.

//IN:  Serie_Elementos_Tratados: De ella obtiene información de los elementos que deban tratarse
               inicio: Establece el id para comenzar a iterar.
               Limite: Establece un límite, marcado desde el inicio, para la iteración.
//OUT: Lista_Iteracion: Una lista de números convertidos a string para realizar el tratamiento.
```

Crear una lista range(...) desde el inicio hasta el límite marcados.
Devolver aquellos que no estén en la serie de elementos tratados.

## Gestionar tiempo

```
//Herramienta para ralentizar/mantener la ejecución del user-agent en el caso de errores en el proceso

//IN:  motivo: Indicador del motivo de invocación al gestor de tiempo
//OUT: {sin salida} la función detiene el proceso el tiempo necesario.

retraso = ESTANDAR * intento

Comprobar motivo:
       error conexión:       retraso = ESTANDAR_ERROR_CONEXION * intento
       timeout: retraso = ESTANDAR_TIMEOUT * intento
       error servidor: ESTANDAR_ERROR_SERVIDOR * intento
       error cliente: ESTANDAR_ERROR_CLIENTE * intento
F Comprobar

time.sleep(retraso)
```

## Rastrear el html

```
//Este módulo se encarga de parsear el html, obtener la información de atributos del dataset,
//organizarlos en su bloque Json y añadirlo al dataset.

//IN: html obtenido desde el servidor
//OUT: un elemento diccionario con la información de cada atributo a capturar

Crear el objeto beautifulsoup con el html

        //Id de la aplicación
        Buscar tag = soup.find('input', {'name': 'next'})
        Obtener el atributo value [ej. /reports/42/]
        Quedarse con el contenido desde la segunda barra hasta el final sin contar la última barra.
        Crear entrada en dict 'Id': id

        //Nombre de la aplicación
        Buscar el tag h1
        Obtener el texto tag.string
        trim del texto
        Crear entrada en dict 'Name': nombre

        //Número de rastradores
        Buscar el tag <a class="section-link" href = "#trackers">
        Avanzar a su hijo <span class="badge badge-pill badge-danger reports">12</span> y
        Obtener el texto
        Crear entrada en dict 'Tracker_count': cuenta_rastreadores

        //Número de permisos
        Buscar el tag <a class="section-link" href = "#permissions">
        Avanzar a su hijo <span class="badge badge-pill badge-danger reports">14</span> y
        Obtener el texto
        Crear entrada en dict 'Permissions_count': cuenta_permisos

        //Version, descargas y fecha de análisis
        Buscar entre los tags <div class="col-md-8 col-12"> aquel que en el texto contenga “Versión”
        Obtener el texto tag.text
        Operar para obtener la versión, las descargas y la fecha de análisis
                version = re.search(r'([0-9]*\.)+[0-9]*', descr).group()
                descargas = re.search(r'Descargas: ([0-9]*(,[0-9]+)*(\+)*)', descr).group(1)
                fecha = re.search(r'creado el ([0-9]{1,2}) de (.*?) de ([0-9]{4})', descr)

        //Trackers
        Buscar entre los tags <div class="col-md-8 col-12"> aquel que en el texto contenga “rastreadores en la aplicación”
        Inicializar la lista de rastreadores
        Para todos los tags dentro del <div>, distinguir aquellos que sean elementos Tag de BeautifulSoup, aquellos que sean <p> y tengan atributos.
                Obtener el texto del tag como nombre del tracker
                Inicializar los propósitos
                        Dentro del tag, iterar por todos los elementos <span> si los contiene.
                                Cada tag es un propósito del tracker padre que se acumula en una lista.
                Añadir el nombre y la lista de propósitos como diccionario a la lista de rastreadores.

        //Permisos, permisos peligrosos
        Buscar entre los tags <div class="col-md-8 col-12"> aquel que en el texto contenga “permisos en la aplicación”
        Inicializar la lista de permisos
        Para todos los tags dentro del <div> buscar los tag <span> con al atributo data-placement = top
                Acumular el permiso del texto de <span> en la lista.
        Realizar la cuenta en el bloque de los tags de imagen con título “Protection level: dangerous”
```

//Pais, desarrollador
Buscar el tag b que encierra el texto Emisor
Saltar hasta el tag hermano <sam>
Extraer del texto “country” y “organizationName”
Tener en cuenta que existen varios formatos para la información de Emisores clave=valor o clave:valor.
Tener en cuenta la posibilidad de que la coma también puede ser parte del valor y no solo delimitar los pares clave-valor.

//Icono
Buscar el tag img con el atributo class = rounded y obtener el link del atributo src

imagen = Obtener icono (ruta)

Si se ha determinado que el icono esté incluido como atributo en el dataset → guardar imagen
Si se ha determinado que el icono esté como anexo externo, guardar imagen en id.png.

Se conforma un componente de diccionario con los atributos rescatados

## Obtener icono

```
//Este módulo se encarga de conectarse al servidor para obtener la imagen y devolverla como un objeto interpretable
//por el dataset.

//IN: url de la imagen obtenida desde el procesamiento del html

//OUT: binario de la imagen representado en un objeto propicio para dataset

A la url base del web scraper https://reports.exodus-privacy.eu.org/es se concatena la ruta pasada por parámetro.
try
        Se obtiene la imagen con la request
exception (errores)
        Devolver nada

Si status = 200:
        Se procesa el atributo content de la request obtenida.
                Reducir tamaño
                Obtener la lista de componentes RGBA.
                        Comprobar:
                                La imagen solo tiene el canal de transparencia → Conformar un RGB a negro e incorporar la componente A como atenuador.
                                La imagen no tiene canal de transparencia → incorporar al RGB un nuevo vector de transparencias a 255(transparencia máxima)
Si no:
        Devolver nada

Se devuelve la imagen tratada.
```