GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2015 / 2016

EUSKAL CRAWLER

MEMORIA DEL PROYECTO

DATOS DE LA ALUMNA O DEL ALUMNO		**DATOS DEL DIRECTOR O DE LA DIRECTORA**	
NOMBRE	DAVID	NOMBRE	JUAN ANTONIO
APELLIDOS	LÓPEZ ANGULO	APELLIDOS	PEREIRA VARELA
		DEPARTAMENTO	LENGUAJES Y SISTEMAS INFORMÁTICOS
FDO.:		FDO.:	
FECHA: 10-06-2016		FECHA: 10-06-2016	

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Descripción del proyecto

Euskal Crawler se pensó como un proyecto que automatizara la obtención de datos de empresas situadas en la comunidad autónoma del País Vasco para posteriormente analizar sus páginas web, detectar cuales están construidas con CMS (Content Management System, sistema de gestión de contenido) y si esos CMS son vulnerables.

Inicialmente, esta automatización se diseñó para que fuera llevada a cabo mediante un *web crawler*[1] que rastreara la red almacenando datos, en especial direcciones de páginas web, de las empresas situadas en la comunidad autónoma del País Vasco. De ahí proviene el nombre del proyecto.

Esta idea inicial se descartó por la dificultad que entrañaba determinar si una URL pertenecía a una empresa situada en la comunidad autónoma del País Vasco o pertenecía a cualquier otro lugar o ni siquiera era una página web de una empresa.

Se exploraron entonces otras alternativas y se optó por la técnica conocida como *web scraping*[2]. Utilizando esta técnica Euskal Crawler obtiene las páginas web de empresas situadas en la comunidad autónoma del País Vasco a través de listados públicos alojados en la red. De esta forma nos cercioramos que son empresas de la comunidad autónoma del País Vasco ya que estos listados públicos están filtrados por provincias.

Posteriormente, tras almacenarlas de manera automatizada en una base de datos las analiza en busca de datos de interés y posibles vulnerabilidades en sus CMS (en el caso de que estén diseñados con uno de estos sistemas).

[1] Sistema de rastreo automatizado de páginas web a través de las URL (Uniform Resource Locator, localizador de recursos uniforme) que encuentra en sus cuerpos.

[2] Técnica de obtención de datos mediante la cual un sistema accede a listados de páginas web para simular la navegación de un usuario y extraer la información necesaria.

Si el script encuentra una vulnerabilidad, envía un aviso de manera automática a los propietarios de la página web, a través de un correo electrónico. Periódicamente vuelve a escanear el listado de páginas web en busca de nuevas vulnerabilidades y actualiza la base de datos.

La parte visible del proyecto consiste en una página web donde usuarios propietarios o administradores de sitios webs pueden registrarse, acceder, y escanearlos para comprobar si tienen vulnerabilidades. Así mismo, este sitio web mostrará información anónima estadística sobre los tipos de vulnerabilidades por regiones y otros datos de interés.

1.2. Motivación

La principal motivación personal de este proyecto es que trata un tema que me gusta y despierta mi interés. Esto propicia una mayor implicación, más ganas por aprender nuevos conocimientos que complementen los adquiridos en el aula para llevar a cabo las tareas necesarias que lograrán el alcance de todo el potencial que se puede desarrollar en este proyecto.

Existen otras motivaciones de carácter más general. La prestigiosa empresa Forbes ha publicado que uno de los principales motivos por los que ha salido a la luz el reciente caso de fraude fiscal de Panamá fue debido a agujeros de seguridad en versiones de Wordpress y Drupal desactualizadas. [1]

En ese caso es de agradecer la falta de actualización tecnológica por parte de la empresa afectada, sin embargo, existen muchas empresas honradas en nuestro entorno en una situación similar, utilizando CMS desactualizados con importantes agujeros de seguridad. Esto supone un riesgo potencial para esas empresas y una oportunidad de negocio sin explotar.

La última motivación relevante consiste en almacenar datos con fines estadísticos. Obtener información acerca de los CMS más utilizados y los CMS más vulnerables en un determinado territorio.

1.3. Planteamiento

Como ya se ha dicho previamente este proyecto se planteó inicialmente como un *web crawler* que con un pequeño listado de direcciones a páginas web seleccionadas manualmente fuera capaz de rastrear esas páginas en busca de nuevos enlaces para determinar si pertenecían a empresas vascas y si fuera así almacenarlas en una base de datos.

Pero a pesar de plantear seriamente el tema, buscar información y barajar algunas soluciones para resolver el problema que ocasionaba la determinación de la procedencia de una página web, se descartó esta idea original. El principal motivo es que el coste temporal para llevarlo a cabo era muy elevado en comparación con otras opciones, como el *web scraping* sobre webs con los datos ya filtrados por empresa y localidad.

Por este motivo se descartó la idea original, se rediseñó la ejecución del proyecto y se procedió entonces a desarrollar un script que obtuviera de manera automatizada los enlaces de listados públicos de empresas vascas previamente localizados mediante *web scraping*.

1.4. Propósito

El propósito de este proyecto es crear una aplicación útil para el entorno empresarial que contribuya a concienciar sobre la seguridad en Internet, disminuir el número de sitios webs vulnerables e inculcar la necesidad de constante actualización en propietarios y administradores.

También trata de elaborar una base de datos sobre los sistemas más utilizados por las empresas y las vulnerabilidades más comunes a disposición del público en general.

2. Planteamiento inicial

2.1. Objetivos

Este proyecto tiene dos objetivos principales. El primero consiste en obtener mediante *web scraping* un listado de direcciones de páginas web de empresas que desarrollen su actividad en la comunidad autónoma del País Vasco. Una vez obtenidas se almacenan en una base de datos para analizarlas en busca de vulnerabilidades en sus CMS en el caso de que estén construidas bajo este tipo de gestores. Posteriormente avisar de manera automatizada a través de correo electrónico a las empresas afectadas.

El segundo objetivo principal consiste en realizar una página web donde cualquier persona pueda comprobar si un sitio web tiene vulnerabilidades en los CMS. También consiste en crear unas estadísticas de todos los sitios web analizados y ofrecerlos de manera visual en forma de grafico al público en general.

2.2. Alcance

En este apartado vamos a ver las funcionalidades a desarrollar en el proyecto para alcanzar los objetivos fijados. A lo largo del proyecto se han ido realizando modificaciones sobre el EDT original a medida que iban surgiendo problemas o necesidades. A continuación se adjunta el EDT final del proyecto.

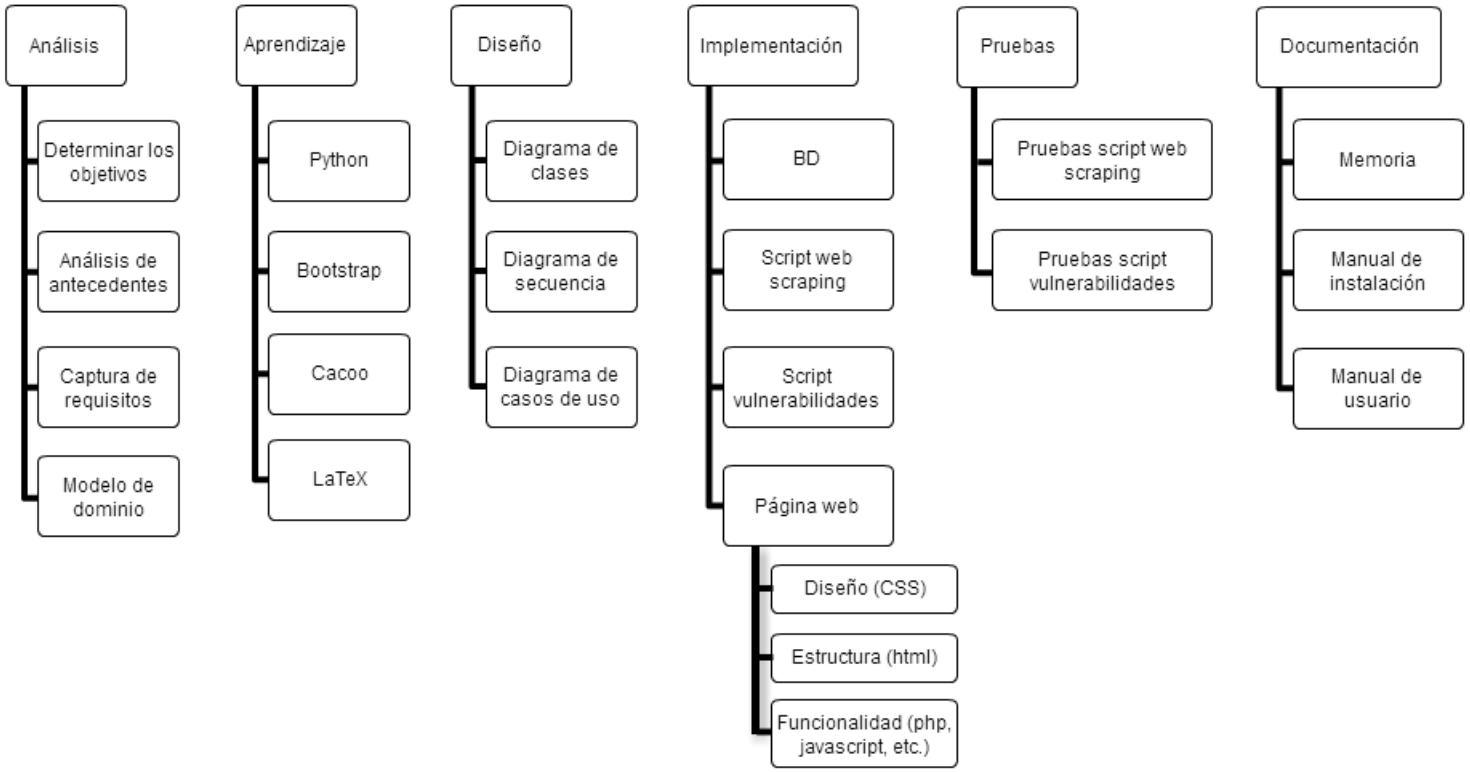
Análisis
Determinar los objetivos
Análisis de antecedentes
Captura de requisitos
Modelo de dominio
Aprendizaje
Python
Bootstrap
Cacoo
LaTeX
Diseño
Diagrama de clases
Diagrama de secuencia
Diagrama de casos de uso
Implementación
BD
Script web scraping
Script vulnerabilidades
Página web
Diseño (CSS)
Estructura (html)
Funcionalidad (php, javascript, etc.)
Pruebas
Pruebas script web scraping
Pruebas script vulnerabilidades
Documentación
Memoria
Manual de instalación
Manual de usuario

Figura 1: EDT.

En la figura 1 podemos ver los principales grupos de tareas que se han llevado a cabo en el proyecto y las subtareas por cada grupo. A continuación se detallan los grupos:

- **Análisis.** Lo primero y fundamental de un proyecto es hacer un buen análisis. Este paso, si se hace bien y se le dedica tiempo, facilitará muchas fases posteriores del proyecto. Es tiempo de proyectar y detallar lo máximo posible todas las tareas futuras, siendo siempre conscientes de que es posible de que necesitemos modificar el análisis inicial a medida que se vaya desarrollando el proyecto.

- **Aprendizaje.** El aprendizaje fue una tarea fundamental en el proyecto ya que trabajé con herramientas y lenguajes de programación que no había utilizado hasta ahora, como Cacoo o Python, por ejemplo. El sistema de aprendizaje fue autodidáctico a través de Internet. La información fue obtenida de foros, blogs, tutoriales y videos. Cuando me surgía una duda acudía a un buscador para consultarla, buscaba y preguntaba hasta solventarla. Cuando no hallaba respuesta acudía al tutor.

- **Diseño.** Este grupo de tareas está enfocado a detallar de manera precisa y cuantificable las partes y los procesos del software que se desarrollará en la fase de implementación. Es el momento de pasar lo abstracto a diagramas ordenados que nos ayuden a la hora de escribir el código.

- **Implementación.** La implementación es la parte más tediosa y larga de todo el proyecto. Se trata de pasar la teoría a la práctica. Y a veces lo que hemos proyectado en la fase de análisis y diseño es complicado de implementar y hay que buscar otras soluciones más sencillas. O puede que nos hayamos equivocado en fases previas y solo nos demos cuenta en la fase de implementación.

- **Pruebas.** Yo diría que esta es la fase más importante de todo el proyecto porque determina la calidad del mismo. Cuantas más pruebas diferentes y minuciosas reciba el sistema mejor sabremos cómo se comporta en distintas situaciones.

- **Documentación.** Este es un grupo de tareas que se ha ido desarrollando a lo largo de todo el proyecto. Borradores de documentos a medida que se avanza durante las etapas. Y al final, cuando todo esté listo, es hora de juntar todos los borradores, pasarlos a limpio y darles formato.

A continuación se detallan las tareas individualmente:

- **Análisis.**
    - **Determinar los objetivos.**
        - Grupo: Análisis.
        - Duración: 5 horas.
        - Descripción: Determinar los objetivos que han de obtenerse con la realización de este proyecto.
    - **Análisis de antecedentes.**
        - Grupo: Análisis.
        - Duración: 15 horas.
        - Descripción: Recopilar información sobre trabajos o proyectos similares a Euskal Crawler que se realizaron con anterioridad.
    - **Captura de requisitos.**
        - Grupo: Análisis.
        - Duración: 20 horas.
        - Descripción: Determinar los requisitos funcionales y no funcionales que ha de cumplir el proyecto.
    - **Modelo de dominio.**
        - Grupo: Análisis.
        - Duración: 15 horas.
        - Descripción: Proyectar un modelo abstracto del funcionamiento de todo el conjunto de la aplicación y de sus partes.

- **Aprendizaje.**
  - **Python.**
    - Grupo: Aprendizaje.
    - Duración: 30 horas.
    - Descripción: Este lenguaje de programación no lo había utilizado nunca antes. Cuando ya controlas otros lenguajes de programación no es complicado aprender uno nuevo. Sin embargo, si me llevó un tiempo familiarizarme con la sintaxis de Python.
  - **Bootstrap.**
    - Grupo: Aprendizaje.
    - Duración: 15 horas.
    - Descripción: Desconocía de su existencia hasta este proyecto. Es un *framework* muy útil y muy sencillo de implementar, aunque se complica cuando quieres modificar algo porque aplica estilos predefinidos.
  - **Cacoo.**
    - Grupo: Aprendizaje.
    - Duración: 10 horas.
    - Descripción: El aprendizaje de esta herramienta web fue sencillo e intuitivo.
  - **LaTeX.**
    - Grupo: Aprendizaje.
    - Duración: 25 horas.
    - Descripción: Un aprendizaje lento y fraccionado ya que LaTeX tiene muchísimas posibilidades a explotar.
  - **Scrapy.**
    - Grupo: Aprendizaje.
    - Duración: 40 horas.
    - Descripción: El aprendizaje de este framework fue un poco más lento ya que era totalmente desconocido para mí. [2]

- **Diseño.**
  - **Diagrama de clases.**
    - Grupo: Diseño.
    - Duración: 30 horas.
    - Descripción: Realizar el diagrama de clases tanto del script de *web scraping* como el script de vulnerabilidades.
  - **Diagrama de secuencia.**
    - Grupo: Diseño.
    - Duración: 40 horas.
    - Descripción: Realizar el diagrama de secuencia tanto del script de *web scraping* como el script de vulnerabilidades para determinar los procesos de ejecución de la aplicación.
  - **Diagrama de casos de uso.**
    - Grupo: Diseño.
    - Duración: 30 horas.
    - Descripción: Realizar el diagrama de casos de uso de la página web para ver que funciones o tareas puede realizar cada actor. Realizar previamente la jerarquía de actores.
- **Implementación.**
  - **BD.**
    - Grupo: Implementación.
    - Duración: 25 horas.
    - Descripción: Creación de la base de datos y de las tablas necesarias en función de las necesidades del sistema.
  - **Script web scraping.**
    - Grupo: Implementación.
    - Duración: 80 horas.
    - Descripción: Programar en Python el código del script de web scraping que obtendrá el listado de datos de empresas y lo almacenará en la base de datos. Para ello se requiere de Scrapy.

- **Script de vulnerabilidades.**
  - Grupo: Implementación.
  - Duración: 70 horas.
  - Descripción: Programar en Python el código del script de vulnerabilidades que obtendrá los datos almacenados en la base de datos y los analizará en búsqueda de vulnerabilidades.
- **Página web.**
  - Grupo: Implementación.
  - Duración: 100 horas.
  - Descripción: Realización de la página web, estructura, diseño, funcionalidad, etc. generando gráficos en base a los datos almacenados previamente por los scripts.

- **Pruebas.**
  - **Pruebas script web scraping.**
    - Grupo: Pruebas.
    - Duración: 25 horas.
    - Descripción: Realizar pruebas sobre el script de *web scraping* con diferentes parámetros.
  - **Pruebas script de vulnerabilidades.**
    - Grupo: Pruebas.
    - Duración: 25 horas.
    - Descripción: Realizar pruebas sobre el script de vulnerabilidades con diferentes parámetros.

- **Documentación.**
  - **Memoria.**
    - Grupo: Documentación.
    - Duración: 80 horas.
    - Descripción: Realizar la memoria del proyecto que se llevará a cabo a lo largo del mismo, desde que comience el proyecto hasta que termine.

2.3. Planificación temporal

Ya conocemos las tareas a realizar. Ahora pasaremos a organizarlas y determinar una previsión del tiempo que podría tomar cada una de ellas. Esto se ha hecho a través de diagramas de Gantt (figuras 2-10).

Análisis:

1. Determinar los objetivos del proyecto
2. Análisis de antecedentes
3. Captura de requisitos
4. Modelo de dominio

Aprendizaje:

5. Aprendizaje de Phyton
6. Aprendizaje de Bootstrap
7. Aprendizaje de Cacoo
8. Aprendizaje de LaTeX
9. Aprendizaje de Scrapy

Diseño:

10. Diagrama de clases
11. Diagramas de secuencia
12. Diagramas de casos de uso

Implementación:

13. BD
14. Script web scraping
15. Script vulnerabilidades
16. Página web
    a) Diseño
    b) Estructura
    c) Funcionalidad

Pruebas:

17. Pruebas script *web scraping*
18. Pruebas script de vulnerabilidades

Documentación

19. Memoria

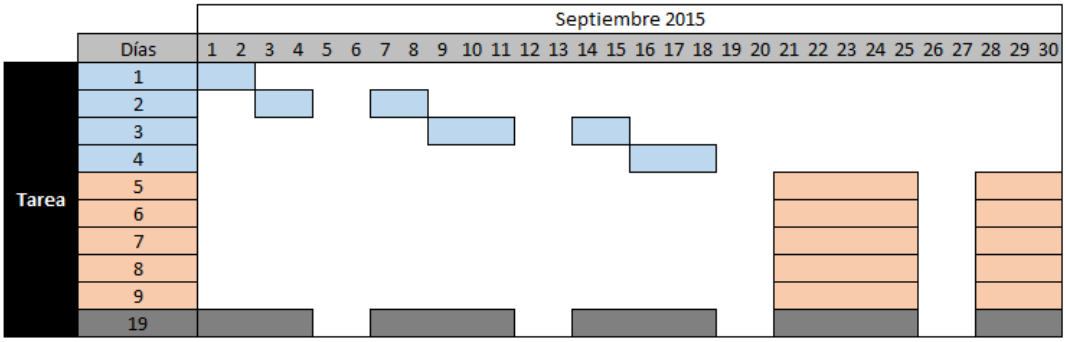

Figura 2: Diagrama de Gantt. Septiembre de 2015.

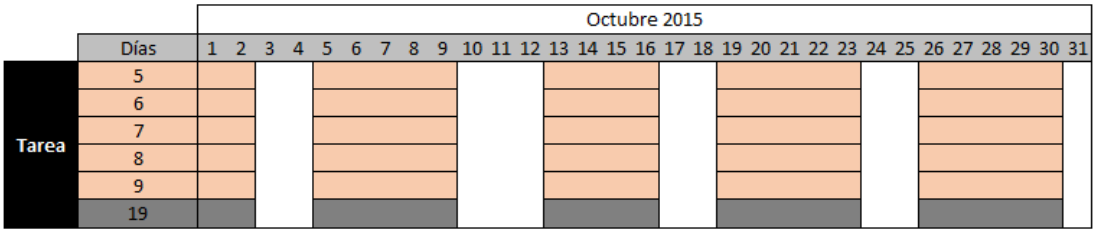

Figura 3: Diagrama de Gantt. Octubre de 2015.

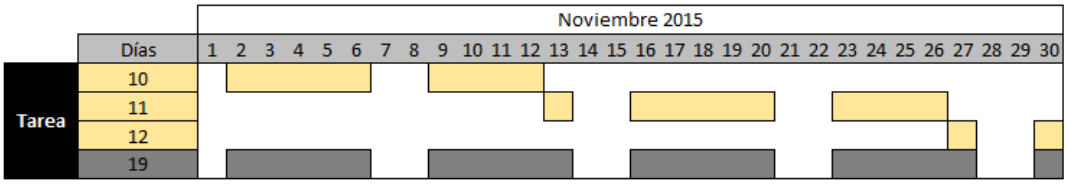

Figura 4: Diagrama de Gantt. Noviembre de 2015.

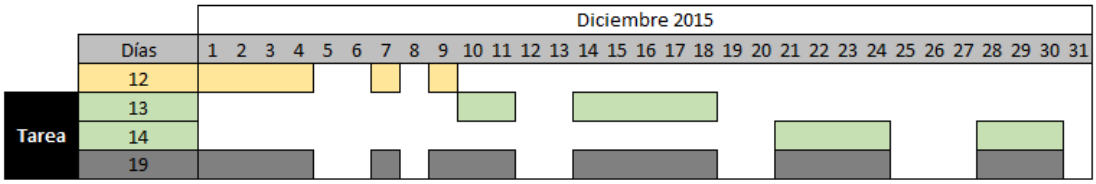

Figura 5: Diagrama de Gantt. Diciembre de 2015.

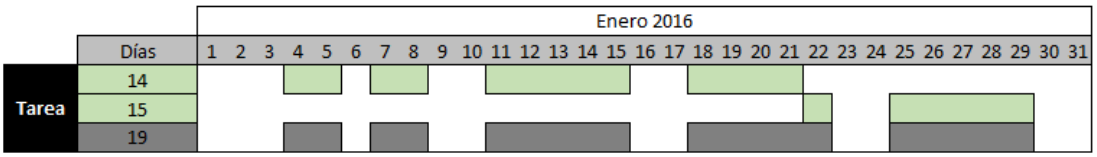

Figura 6: Diagrama de Gantt. Enero de 2016.

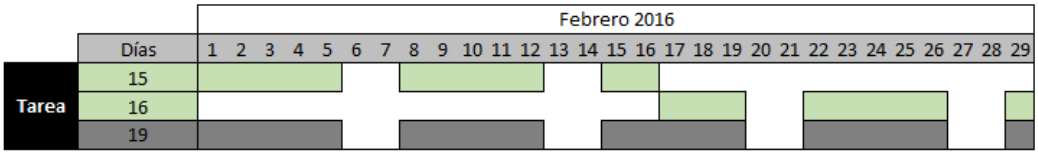

Figura 7: Diagrama de Gantt. Febrero de 2016.

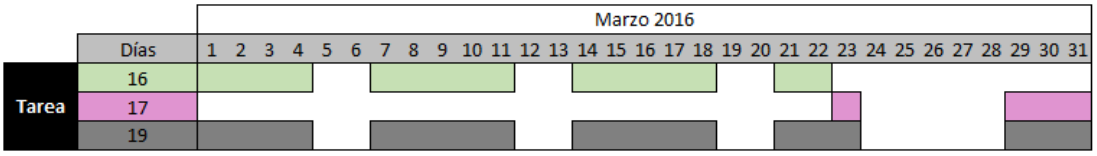

Figura 8: Diagrama de Gantt. Marzo de 2016.

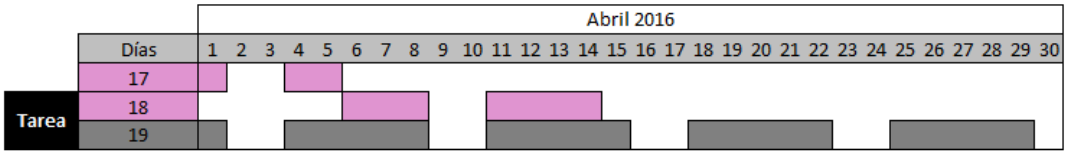

Figura 9: Diagrama de Gantt. Abril de 2016.

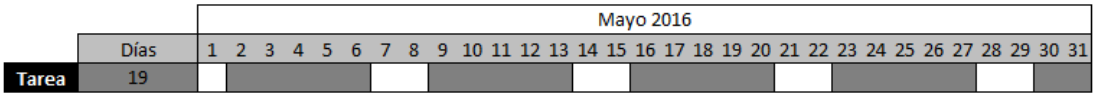

Figura 10: Diagrama de Gantt. Mayo de 2016.

Se han excluido días festivos, fines de semana y vacaciones, como si se tratara de un trabajo por cuenta ajena. Se han proyectado 20 horas de trabajo a la semana (media jornada).

Algunas tareas se solapan el tiempo porque se ha previsto que puedan realizarse simultáneamente, como por ejemplo, elaborar una parte de la documentación al mismo tiempo que se va desarrollando el código.

2.4. Herramientas

En esta sección aparecen todas las herramientas utilizadas en el proyecto en todas sus etapas y para que se ha utilizado cada una de ellas.

- **PyCharm.** Es un IDE desarrollado por la compañía Jetbrains. Permite trabajar con frameworks como Django, Flask o Bootstrap entre otros y con lenguajes de programación como HTML, CSS, JavaScript y Python. Posee un editor inteligente que detecta y resalta los errores. [3]

  Esta herramienta se ha utilizado para el desarrollo de todo el código del proyecto, para los scripts desarrollados en Python, tanto los scripts de *webs craping* que obtienen la información de listados web como los scripts que almacenan dicha información en la base de datos, y para la realización completa de la página web.

  Se ha elegido esta herramienta porque es el principal entorno de desarrollo integrado basado en Python, que es el lenguaje utilizado para los scripts del proyecto.

  Alternativas descartadas: Eclipse.

- **Cacoo.** “Es una herramienta en línea que permite crear de manera colaborativa una variedad de organizadores gráficos tales como mapas mentales, wireframes, diagramas UML, y de redes, entre otros. Esta herramienta ofrece una licencia de uso gratuito y otra comercial; esta última ofrece funcionalidades adicionales a las de la versión gratuita.” [4]

  Esta herramienta se ha utilizado para la realización de los diagramas de clase, de secuencia y de casos de uso. También ha sido utilizado para la creación del EDT.

  Se ha elegido esta herramienta por su popularidad, su sencillez de uso y la posibilidad de monitoreo de los diagramas por parte del tutor.

- **Plot.ly.** Es una herramienta web para graficar y analizar datos colaborativamente. Permite trabajar con diferentes formatos de datos como excel, CSV o texto. Ofrece una amplia variedad de gráficos: de columnas, de líneas, circulares, histogramas, de puntos, mapas de calor, 3D, etc. Permite trabajar y crear gráficos con una herramienta propia o utilizar sus API para crear gráficos con otras herramientas o en otras plataformas, como Matlab, R, Python o JavaScript. [5] [6]

  Esta herramienta se ha utilizado para crear los gráficos que aparecen en la página web con los datos de los CMS más utilizados y los más vulnerables.

  Se ha elegido esta herramienta por su popularidad de uso, su buena reputación y la variedad de opciones de uso que ofrece. Pero sobre todo porque permite trabajar con JavaScript para poder así integrarlo en la web.

- **ShareLaTeX.** Es un editor de LaTeX online para crear documentos de gran calidad. Posee una interfaz fácil de utilizar que permite gestionar los proyec-

tos cómodamente mediante un árbol de archivos. Sus principales características son: coloreado de sintaxis para el código, resaltado de errores y vista previa del PDF resultante disponible en todo momento pulsando el botón Recompilar. También permite añadir colaboradores al proyecto para editar al mismo tiempo un documento y chatear desde la misma ventana. [7] [8]

Esta herramienta se ha utilizado para realizar la documentación final aportando calidad tanto al aspecto como al contenido.

Se ha elegido esta herramienta porque entre otras muchas herramientas que permiten trabajar con LaTeX esta es la que más se ajusta a mis necesidades, que son: la previsualización del resultado, poder compartir el documento a través de la misma aplicación y sobre todo el autoguardado online que periódicamente, cada pocos segundos, guarda los cambios a medida que vas escribiendo, lo cual disminuye considerablemente el riesgo de perder los avances o incluso el documento completo.

- **Github.** "Es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. Aloja el repositorio de código y brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de eso, ofrece la posibilidad de contribuir a mejorar el software de los demás." [9]

  Esta herramienta se ha utilizado para el almacenamiento seguro y el control de todo el código desarrollado durante el proyecto, tanto para los scripts como para la página web. Además ha permitido una comunicación fluida con el tutor, permitiendo un seguimiento por su parte de toda la ejecución del proyecto.

Se ha elegido esta herramienta debido a que era una herramienta ya conocida con la que había trabajado a lo largo del grado, por lo tanto no requería un aprendizaje previo lo cual beneficiaba a los costes temporales del proyecto. Además era compatible con el IDE seleccionado previamente.

- **Bootstrap.** "Es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice." [10]

  Este framework se ha utilizado para mejorar el estilo de la página web.

  Se ha elegido este framework por su facilidad de implementación que favorece una reducción en los tiempos de desarrollo de la página web.

- **MySQL.** Es un sistema de administración de bases de datos relacionales. Utiliza un sistema de tablas para organizar la información. Fue desarrollado en lenguaje C y se adapta a muchos entornos de desarrollo permitiendo la interactuación con lenguajes de programación muy utilizados, como PHP, Python, Java, etc. [11]

  Se ha utilizado esta herramienta para la creación de todas las tablas que componen la base de datos del proyecto.

  Se ha elegido esta herramienta debido a que es gratuita, es ampliamente conocida y se ha trabajado con ella a lo largo del grado. Además permite una sencilla interactuación con lenguajes de programación previamente escogidos.

2.5. Gestión de riesgos

En esta sección vamos a ver cuáles son los riesgos que pueden afectar negativamente a la ejecución del proyecto, como preverlos y en caso de que finalmente sucedan, a pesar de la prevención, como gestionarlos.

- **Falta de conocimientos**

Descripción	En este proyecto se va a trabajar con muchas herramientas y lenguajes de programación desconocidos, como Python, Cacoo o Plot.ly. Durante el proceso de elaboración de los scripts o de la página web pueden surgir momentos en los que queramos que la aplicación realice una determinada función y no sepamos cómo llevarla a cabo.
Prevención	Prepararse lo mejor posible para obtener el máximo de información sobre las herramientas y los lenguajes de programación antes de su utilización.
Plan de contingencia	Detener la actividad hasta solventar la duda o el problema consultando en internet (tutoriales, foros, blogs, etc.) o al tutor.
Probabilidad	Media.
Impacto	Crítico.

Tabla 1: Riesgo de falta de conocimientos.

- **Accidente o enfermedad**

Descripción	Enfermedad común, accidente o lesión física que afecte a las capacidades para continuar realizando el trabajo con normalidad.
Prevención	Cuidarse, comer bien y no coger frio.
Plan de contingencia	En caso de incapacidad para continuar, detener toda actividad hasta la recuperación.
Probabilidad	Improbable.
Impacto	En función de la enfermedad o la lesión puede ser nulo si se trata de un resfriado, por ejemplo, o muy crítico si se trata de una lesión en las muñecas u otra enfermedad o lesión que incapacite físicamente para continuar el trabajo.

Tabla 2: Riesgo de accidente o enfermedad.

- **Fallos de hardware o software**

Descripción	Algún componente del hardware o software produce un error, se estropea o deja de funcionar. Pueden ser tanto fallos eléctricos, como físicos, virus u otros problemas con el software.
Prevención	Mantener el hardware limpio de polvo y evitar el derramado de líquidos. Mantener instalado y actualizado un antivirus en el equipo informático. Tener actualizados todos los drivers, programas y herramientas a utilizar.
Plan de contingencia	Si se trata de hardware, tratar por todos los medios a disposición de solventar el problema y reparar el componente dañado, si es posible. Si se trata de software reinstalar el driver o programa dañado. En casos más extremos formatear el equipo informático.
Probabilidad	Baja.
Impacto	En caso de inutilización absoluta del hardware por incendio, rotura, fallos electrónicos por vertido de líquidos u otros problemas supondrá un coste económico (volver a comprar otro equipo informático).

Tabla 3: Riesgo de fallos de hardware o software.

- **Fallo en la estimación temporal**

Descripción	Durante la ejecución del proyecto es posible que una o varias tareas requieran más tiempo del estimado.
Prevención	Revisar otros proyectos con tareas similares para hacerse una idea general del coste temporal. Detallar bien todos los pasos y todas las subtareas que se realizarán para ajustar más el tiempo a la realidad.
Plan de contingencia	Si se produce el caso en que una tarea sobrepase la estimación deberá hacerse una nueva estimación con la situación actual.
Probabilidad	Alta.
Impacto	Retrasos en la entrega del proyecto.

Tabla 4: Riesgo de fallo en la estimación temporal.

- **Problemas de compatibilidad**

Descripción	Durante la elaboración del código pueden surgir problemas de compatibilidad entre versiones o entre módulos distintos. Por ejemplo, Python tiene varias versiones y el código escrito en una versión antigua no es compatible con una versión nueva. O también se podría dar el caso de un script en Python que no es compatible con la versión del servidor que utilizamos para realizar un script en la página web.
Prevención	Conocer bien todos los lenguajes de programación y las herramientas que vamos a utilizar. Informarse sobre las versiones de cada uno de ellos y si son compatibles.
Plan de contingencia	Si por cualquier motivo sucediera una incompatibilidad a pesar de la prevención se deberá solventar reescribiendo el código para que sea compatible o buscando otra herramienta que cumpla la misma función y sea compatible.
Probabilidad	Media.
Impacto	Sería un coste temporal imprevisto añadido.

Tabla 5: Riesgo de problemas de compatibilidad.

- **Pérdida de código o documentación**

Descripción	Por algún descuido o fallo informático es posible perder todo o parte del código y de la documentación realizados hasta un punto determinado.
Prevención	Elegir herramientas con autoguardados en la nube, hacer copias de seguridad diariamente tanto local como online y poner especial cuidado cuando estemos realizando alguna tarea del proyecto.
Plan de contingencia	En el caso de perder todo el proyecto y que este estuviera bastante avanzado hablar con el tutor. Volver a hacer una estimación temporal de todo el proyecto desde cero. Si es una pérdida parcial reajustar las estimaciones temporales del proyecto y rehacerla lo antes posible, ya que una vez hecho es más fácil volverlo hacer.
Probabilidad	Baja
Impacto	Muy alto en el caso de perder todo. Leve si solo se pierde una parte.

Tabla 6: Riesgo de pérdida de código o documentación.

2.6. Evaluación económica

En esta sección se realiza un cálculo del coste total del proyecto teniendo en cuenta tanto la mano de obra, como los costes ocasionados por la adquisición o uso de software y hardware. También se incluyen otros gastos derivados de la ejecución del proyecto.

- **Mano de Obra**

  Para realizar el cálculo del coste de la mano de obra se ha consultado el convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública que establece un salario mínimo de 1057,19 euros al mes (plus de convenio incluido) para un programador junior a jornada completa [12].

  Se ha contrastado esta información con ofertas de portales de empleo online, como infojobs, infoempleo y jobandtalent. Comprobando que diferentes empresas ofrecen puestos de trabajo para programadores junior con salarios anuales entre 12.000 y 18.000 euros. Podemos verificar que la información es realista y por lo tanto que este proyecto podría haberse llevado acabo cobrando un salario similar.

  Como el proyecto necesita de 10 meses para su ejecución, pero solo a media jornada, podemos establecer un coste de mano de obra total de 5285,95 euros.

- **Software y hardware**

  Todas las licencias del software utilizado son gratuitas, excepto para el sistema operativo utilizado (Windows 7) que se incluye en el precio del ordenador.

El equipo informático ha costado en total 900 euros. Estimando que la duración del equipo es de 3 años, el precio de amortización para dos meses sería de:

$$900\ € / 3\ \text{años} / 12\ \text{meses} = 25\ €/\text{mes}$$

Como la duración del proyecto es de 10 meses, la amortización total del equipo durante su ejecución ascendería a 250 euros.

- **Otros gastos**

La compra de un dominio .com para el sitio web asciende a 10 euros.

El equipo informático tiene una potencia de 0.09 kW y será necesario un uso del mismo de 1000 horas, por lo tanto el consumo energético total será de 90 kWh. Con el precio actual de 0.13 €/kWh en España (esto es una estimación ya que el precio real variará en función de la compañía a la que se contrate el servicio de suministro) el gasto por consumo eléctrico sería de 11,7 €.

El consumo de internet sería 240 euros que derivan de la contratación de una tarifa plana con un proveedor de servicios a 24 euros mensuales aproximadamente.

El precio del hosting del servidor no viene incluido ya que nos lo proporcionan de manera gratuita.

- **Gastos totales**

  En la siguiente tabla se resumen todos los gastos originados en el diseño y elaboración del proyecto.

Descripción	Desglose	Total (euros)
Mano de obra	10 meses x 1057,19 €/mes	5285,95
Software y hardware	(900 € / 3 años / 12 meses) * 10 meses	250
Dominio		10
Electricidad	90 kWh x 0.13 €/kWh	11,7
Internet	10 meses x 24 €/mes	240
	Coste total del proyecto	5797,65

Tabla 7: Coste económico total del proyecto.

3. Análisis de antecedentes

Euskal Crawler es un proyecto único y sin precedentes, o al menos no conozco ninguno. Este proyecto obtiene información sobre empresas de la comunidad autónoma del País Vasco, la analiza buscando la versión del CMS, vulnerabilidades en el mismo y otra información de interés y con toda esta información almacenada crea estadísticas en forma de gráficos fácilmente consultables por usuarios sin habilidades ni conocimientos avanzados de programación.

Este conjunto de funciones es lo que hace único a Euskal Crawler. No existe ningún sistema que ofrezca información acerca de los CMS y las vulnerabilidades de páginas web de empresas de la comunidad autónoma del País Vasco. Sin embargo, si podemos encontrar proyectos anteriores a Euskal Crawler que analizan individualmente una URL para obtener información sobre que CMS utiliza y si tiene vulnerabilidades. Y estos son los proyectos que pasaremos a detallar a continuación.

3.1. WIG

WIG (WebApp Information Gatherer) [13] es un proyecto desarrollado por jekyc alojado en github. Puede identificar un gran número de CMS y otro tipo de aplicaciones administrativas. Está basado en checksums y coincidencia de strings para determinar las diferentes versiones de CMS. Con estas herramientas calcula una puntuación para cada CMS y su versión. Muestra cada CMS con la versión más probable del mismo. El cálculo de la puntuación está basada en pesos y el número de veces que aparece.

WIG también trata de detectar el sistema operativo en el que está basado el servidor. Esto lo hace a través de una base de datos de valores de cabecera conocidos para diferentes sistemas operativos, que permite a WIG determinar las versiones de Microsoft Windows o Linux.

WIG tiene una especial relevancia para Euskal Crawler ya que se ha reutilizado su código para el escaneo de URL.

El problema de WIG, y aquí es donde entra la innovación de Euskal Crawler, es que es un proyecto con mala accesibilidad y usabilidad. El usuario medio no sabría cómo utilizar WIG, es muy complicado para ellos. También es mejorable la visualización de los resultados. Unos resultados más visuales hacen más atractivo y sencillo el manejo de una aplicación.

Si su uso es complicado, su modificación todavía más, y este es otro de los problemas de WIG. Ya que era un proyecto desarrollado bajo Python 2 y se necesitó añadir funcionalidades que solo estaban disponibles para Python 3 nada funcionaba bien, pero estos problemas se verán más detalladamente en la sección de desarrollo.

3.2. CMS explorer

Como WIG, existen más aplicaciones similares de escaneo de vulnerabilidades, pero todas tienen el mismo problema, una mala usabilidad y accesibilidad para el usuario con conocimientos medios de informática. El pequeño comerciante de un barrio de Bilbao que tiene una página web que le hicieron hace unos cuantos años no sabe utilizar estas aplicaciones, ni siquiera sabe de su existencia. Por eso el objetivo de Euskal Crawler es hacer llegar a pequeños y grandes comerciantes una información que puedan entender, visualizar y manejar.

Pasaremos a ver algún proyecto más, similar a WIG, siendo conscientes de que a efectos prácticos son muy escasas sus diferencias.

El siguiente proyecto se llama CMS explorer. Es un script originalmente escrito en Perl, a diferencia de WIG que estaba escrito en Python. Sirve para detectar módulos, plugins, componentes o temas de CMS que estén siendo ejecutados por un sitio web.

CMS explorer sin embargo, no realiza comprobaciones de seguridad directas. Lo que si hace es revelar los archivos ocultos que no están normalmente accesibles a través de los WebClient.

Actualmente el código ha sido migrado a Python. [14] [15]

3.3. Acunetix WVS

Otra aplicación de este tipo es Acunetix WVS (Web Vulnerability Scanner). Es una aplicación desarrollada por la empresa Acunetix. A diferencia de WIG o CMS explorer Acunetix WVS no se centra exclusivamente en los CMS sino que escanea y analiza una página web en busca de cualquier tipo de vulnerabilidad. Este proyecto se centra en las vulnerabilidades de plataformas como el comercio electrónico, blogs, páginas de inicio de sesión y otros contenidos dinámicos.

A diferencia de WIG o CMS explorer, proyectos que están alojados en github como código abierto, Acunetix WVS es código cerrado. Es una aplicación que tiene web propia donde se puede consultar la información y escanear un sitio web y es visualmente más atractivo. Trabaja con un montón de prestigiosas marcas, como AVG o Sony.

Esta aplicación se centra en los ataques Cross-site Scripting, SQL Injection, DOM-based XSS y CSRF Attacks. Está disponible para descargarlo o para escanear un sitio web desde su misma página. Y genera una gran variedad de informes técnicos disponibles para los desarrolladores o los propietarios de un sitio web. Pero a diferencia de Euskal Crawler, no automatiza la recogida de información de empresas ni genera estadísticas sobre un territorio consultables públicamente.

Anteriormente tenían una versión de pago y una versión gratuita con menos funcionalidades. Pero actualmente, al parecer, han eliminado la versión gratuita y solo tienen la versión de pago con 14 días de prueba gratuitos. [16]

3.4. Otros proyectos

Como ya he dicho anteriormente hay muchas aplicaciones y proyectos que realizan la misma función, escanear un sitio web en busca de vulnerabilidades. Los hay más específicos, como por ejemplo los que buscan vulnerabilidades en CMS. O más generales, como ya hemos visto con Acunetix WVS.

Se puede conocer más de estas aplicaciones en el blog de Hack Players donde han escrito un artículo en el que analizan más de sesenta aplicaciones.

4. Captura de requisitos

En esta sección se detallará la captura de requisitos. Es un paso fundamental en el desarrollo del proyecto y es necesario dedicarle tiempo y hacerlo lo mejor posible para un desarrollo más fluido de la etapa de elaboración.

4.1. Requisitos funcionales

Primero pasaremos a detallar los requisitos funcionales. Estos requisitos definen las diferentes funciones que ofrece la aplicación. Entre ellas están:

- Cualquier usuario, registrado o sin registrar, puede ver todo el contenido público de la página web.
- Un usuario anónimo puede registrarse.
- Un usuario registrado puede ingresar a la zona de usuario.
- Un usuario registrado puede buscar vulnerabilidades en una o más páginas web.
- La función de buscar vulnerabilidades solo mostrará si una página web es vulnerable o no con el objetivo de mejorar el CMS de la misma. En ningún caso se informará al usuario de cuáles son las vulnerabilidades detectadas ni de como explotarlas.

4.2. Requisitos no funcionales

Los requisitos no funcionales no se encargan de definir que funciones puede realizar el sistema sino cómo se comporta y como es su manejo. Son los siguientes:

- La página web será accesible desde cualquier parte del mundo con conexión a internet.
- La página web, la base de datos y los scripts asociados responderán en un tiempo razonable siempre que no existan problemas físicos con el servidor u otros problemas ajenos al proyecto.
- La página web será entendible y fácil de manejar.

4.3. Casos de uso

Los casos de uso son una representación gráfica de la secuencia de actividades necesaria para realizar un proceso. Siempre es un actor el que inicia una actividad, que puede desencadenar otras actividades. Cada actividad se denomina caso de uso. Los actores y los casos de usos se comunican entre sí mediante relaciones. A continuación se detallan los tipos de relaciones:

- **Comunicación.** Este tipo de relación se da exclusivamente entre un actor y un caso de uso para indicar el inicio de un proceso mediante la acción externa o interna de un actor. Se le asigna la etiqueta «communicate» aunque normalmente se suele omitir.

- **Inclusión.** Cuando la ejecución de un caso de uso implica directamente la ejecución de otro caso de uso utilizamos la relación de inclusión. Hasta la versión 1.2 de UML se utilizaba la etiqueta «uses» que se sustituyó por «include» en la versión 1.3.

- **Extensión.** Cuando un caso de uso es una parte de otro caso de uso y es dependiente de él, es decir, que no tiene sentido la ejecución aislada de la actividad que representa, utilizamos la relación de extensión. Para esta relación se utiliza la etiqueta de «extends».

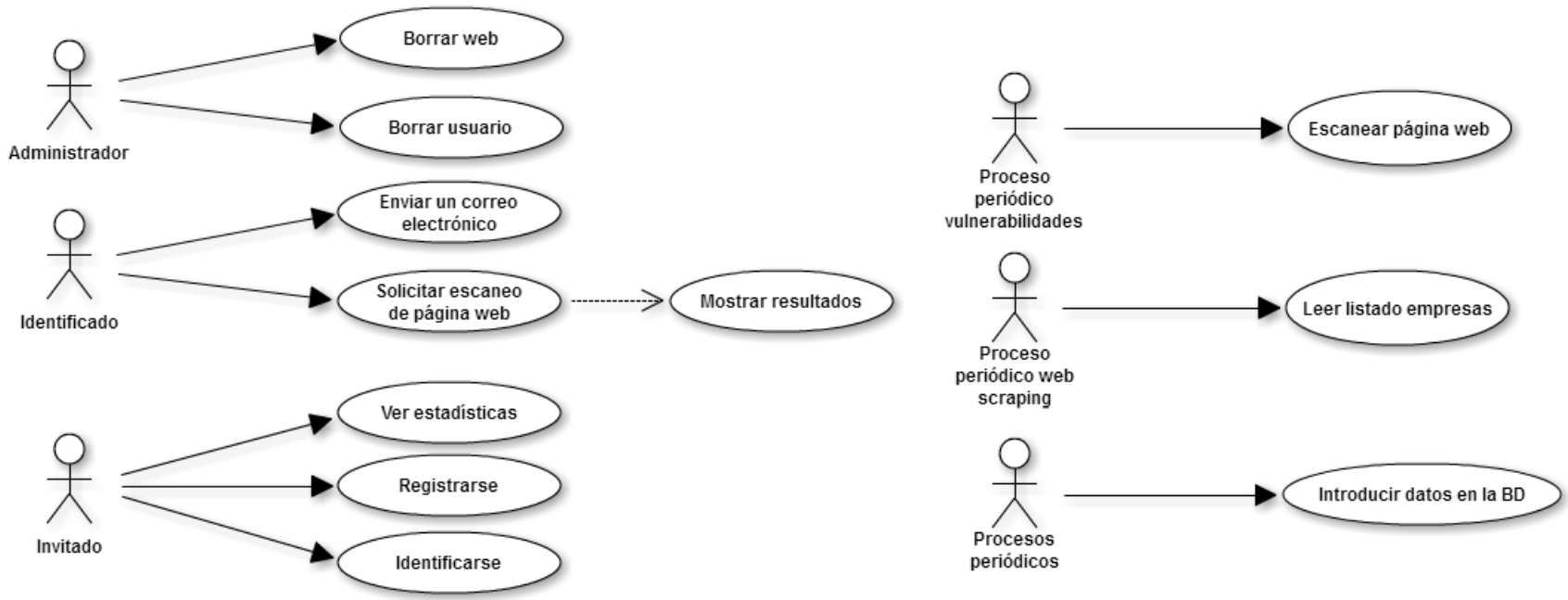

Figura 11: Diagrama de casos de uso.

Para el actor administrador existen los siguientes casos de uso (Figura 11):

- **Borrar la página web.** El administrador puede borrar la página web y todo el contenido asociada a la misma.
- **Enviar correo electrónico.** El administrador puede enviar un correo electrónico a una empresa o un usuario registrado para informarle sobre vulnerabilidades o cuestiones relacionadas con la página web.
- **Escanear una página web.** El administrador puede escanear manualmente una página web en busca de vulnerabilidades en su CMS.
- **Ver estadísticas.** El administrador puede ver las estadísticas mostradas públicamente que han sido generadas automáticamente por la página web.
- **Actualizar base de datos.** Permite la modificación de los datos almacenados en la base de datos, la eliminación de los mismos o la inclusión de datos nuevos.

Para el actor identificado existen los caso de uso enviar correo electrónico, escanear una página web y ver las estadísticas (casos de uso descritos anteriormente).

Para el actor invitado existen los siguientes casos de uso (Figura 11):

- **Registrarse.** Permite el registro en la página web mediante un formulario a toda persona desconocida que acceda a la misma.
- **Identificarse.** Permite la identificación en la página web para acceder al menú de usuario a toda persona desconocida que acceda a la misma con la condición de que previamente se haya registrado.
- **Ver estadísticas.** Descrito anteriormente.

Para el actor script vulnerabilidades existen los caso de uso actualizar base de datos y escanear un página web (casos de uso descritos anteriormente).

Para el actor script *web scraping* existen los siguientes casos de uso (Figura 11):

- **Actualizar base de datos.** Descrito anteriormente.
- **Identificarse.** Obtiene mediante *web scraping* un listado público de empresas y lo ordena recogiendo los datos de interés para introducirlos en la base de datos posteriormente.

4.4. Jerarquía de actores

A continuación se presenta la jerarquía de actores que refleja todos los actores que actúan o pueden actuar sobre el sistema desarrollado. Se muestra en la Figura 12.

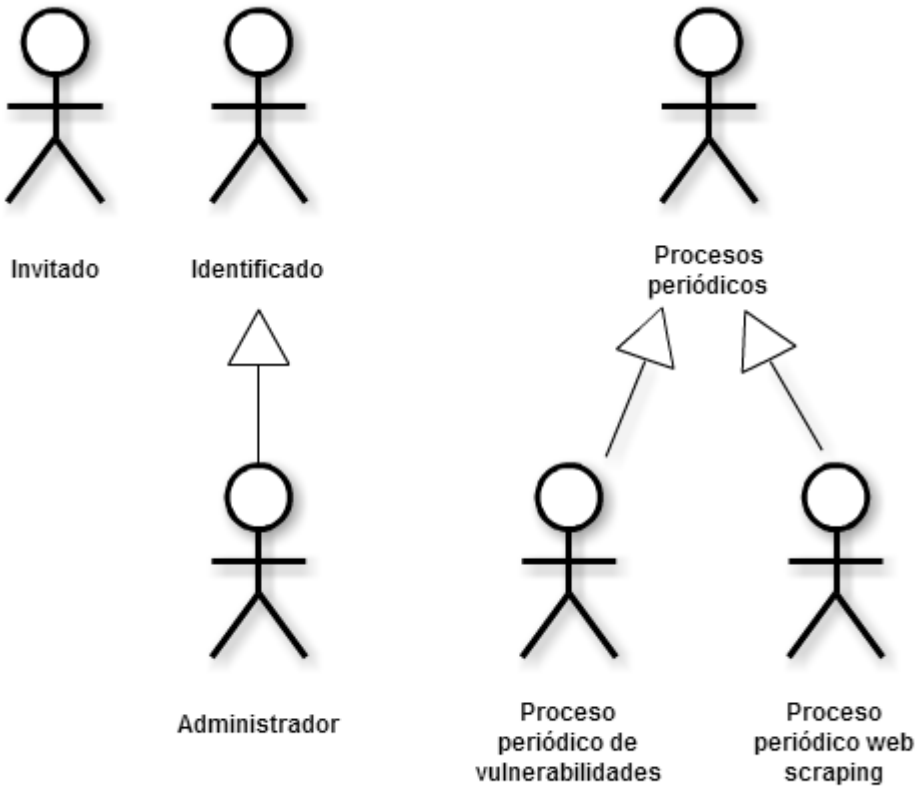

Figura 12: Jerarquía de actores.

- **Invitado.** Todo aquel usuario desconocido para la aplicación que accede a la página web.
- **Identificado.** Todo aquel usuario que se ha registrado e identificado en la página web.

- **Administrador.** El administrador tiene todos los derechos, privilegios y responsabilidades sobre la totalidad del proyecto.
- **Script vulnerabilidades.** Es un script desarrollado en Python que interactúa con el sistema accediendo a la base de datos y buscando vulnerabilidades en cada una de las webs que estén almacenadas.
- **Script web scraping.** Es un script desarrollado en Python que se encarga de obtener información acerca de empresas a través de listados públicos y las almacena en la base de datos.

4.5. Modelo de dominio

El modelo de dominio es una representación de la realidad de forma abstracta. Se guía por las reglas de UML pero no hace referencia a ninguna entidad de código o lenguaje de programación. Una entidad en el modelo de dominio representa objetos, sistemas, personas o cosas que interactúan entre sí.

La finalidad de esta representación es comprender y visualizar todas las entidades que participan en el sistema y de qué forma interactúan desde el punto de vista abstracto.

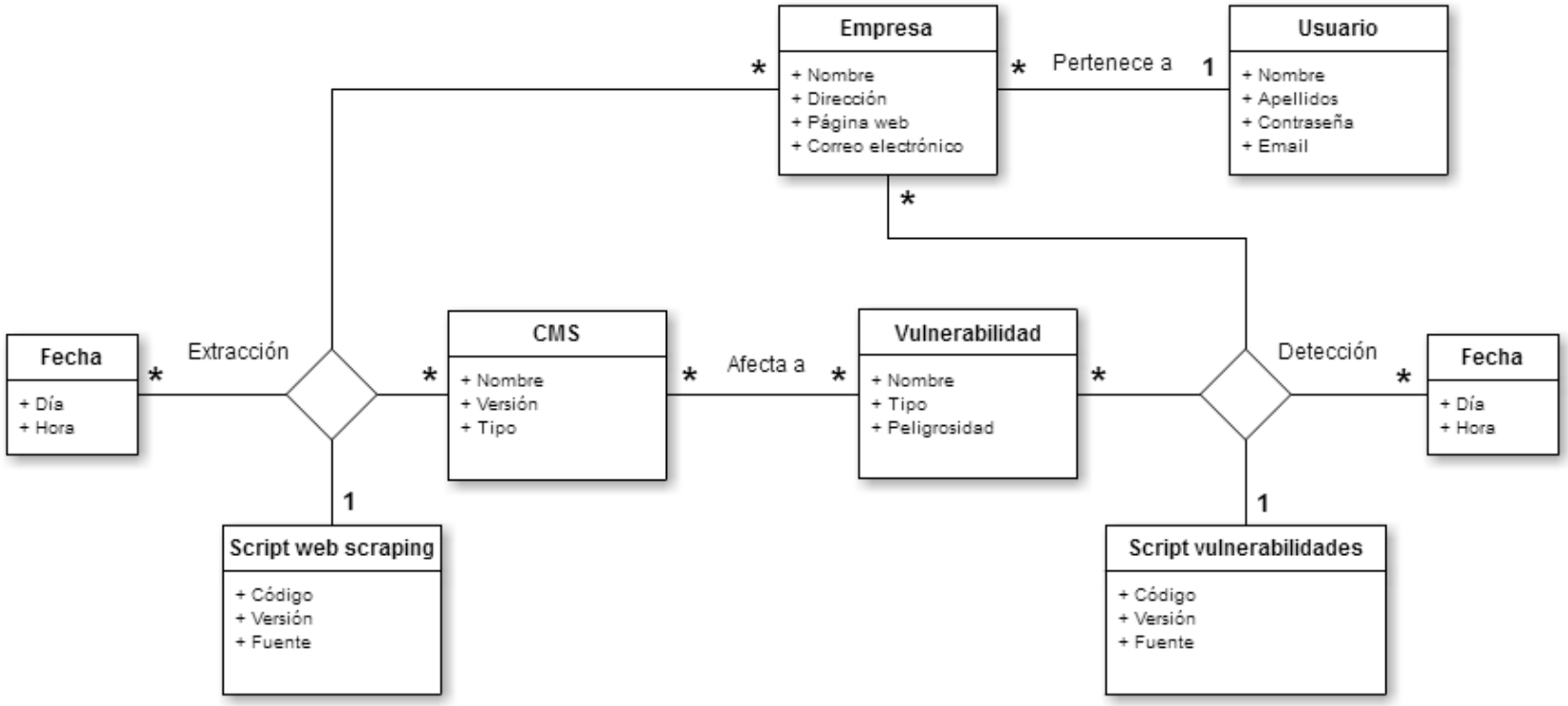

Figura 13: Modelo de dominio.

En la figura 13 podemos ver el modelo de dominio de la aplicación. A continuación se describen las entidades:

- **Empresa.** Esta entidad representa a los datos que se almacenaran de una empresa que son necesarios para generar posteriormente las estadísticas.
- **Usuario.** Esta es la entidad que interactua con el sistema, tiene datos que lo identifican y puede generar datos de una empresa a través de un escaneo.
- **Página web.** Esta entidad almacena los datos referentes a la página web que permiten el acceso a la misma y su identificación.
- **Script.** Esta entidad es la encargada de generar datos que se incluirán en la entidad de la página web.

5. Análisis y diseño

Pasamos ahora a describir la fase de análisis y diseño. Esta fase es fundamental ya que es la fase previa al desarrollo. El objetivo es trasformar la realidad y las ideas capturadas en la fase de captura de requisitos y transformarlas en conceptos aplicables al mundo de la programación.

Se trata de analizar todos los aspectos del sistema y recogerlos en un formato estandarizado que consiste en diagramas, de clase y de secuencia, que determinarán las entidades de la aplicación y como han de comportarse. También es momento de representar las entidades que se almacenarán en la base de datos.

5.1. Base de datos

El proyecto contará con una única base de datos que estará formada por tablas que recogerán toda la información que se necesita almacenar.

En la figura 14 se puede apreciar todas las tablas y sus atributos, así como las claves primarias y las claves secundarias con las que se relacionan entre sí.

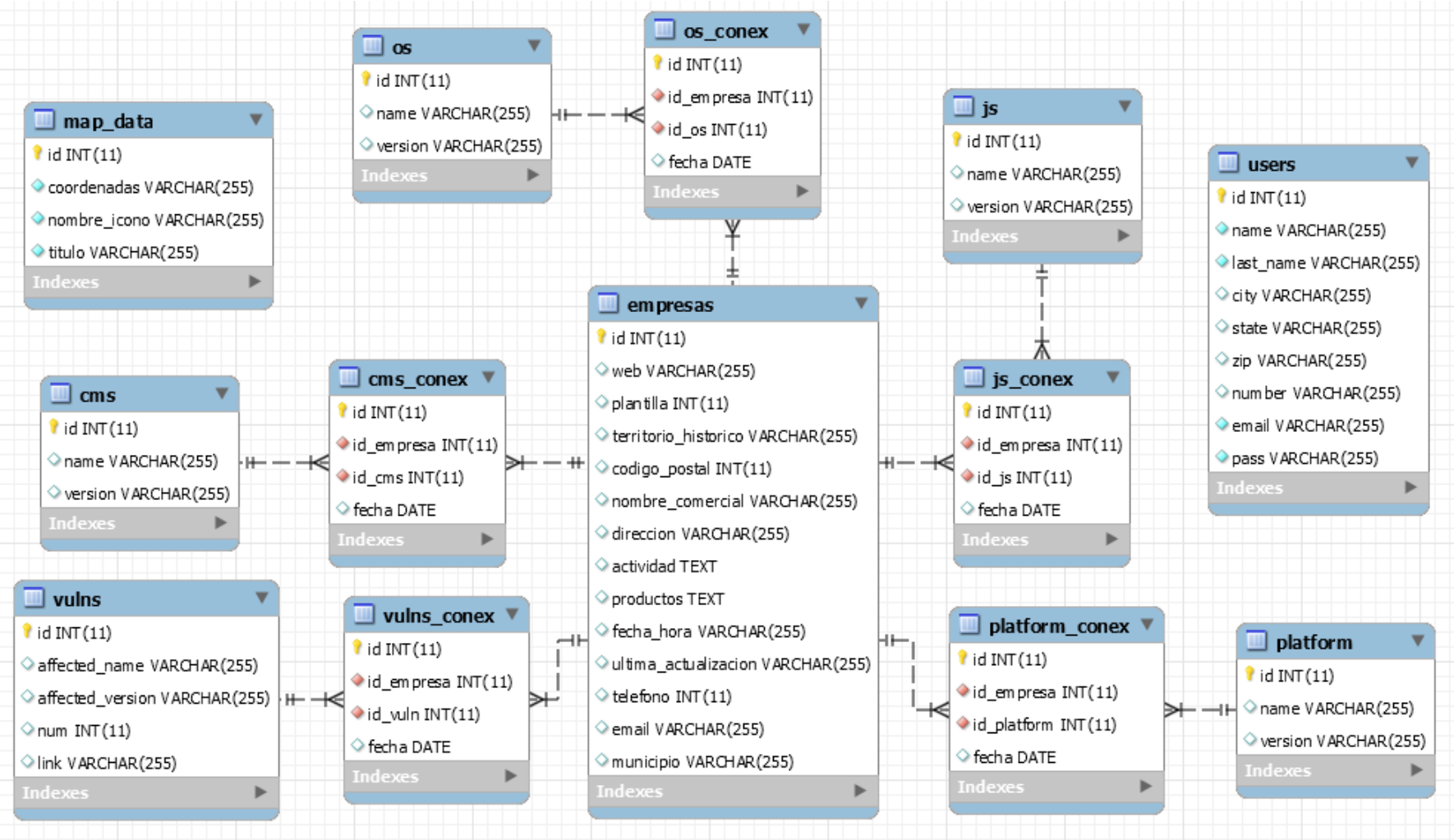

Figura 14: Diagrama de base de datos.

A continuación se describen detalladamente todas las tablas de la base de datos:

- **map_data**

  Esta tabla guarda los datos relacionados con el mapa de la página web. Se obtiene la dirección física de cada empresa con un CMS identificado y se rastrea a través del api de google maps. Esto devuelve unas coordenadas que se almacenan debido a que google solo permite un número reducido de peticiones por segundo. Por lo tanto si no guardábamos esta información cargar el mapa tardaba algunos minutos, lo cual no era práctico. También se guardan el nombre del icono, ya que cada CMS tendrá su propio icono en el mapa, y el título que no es más que el nombre del CMS y su versión. Esta tabla no se relaciona con ninguna otra.

- **users**

  Esta tabla guarda los datos relacionados con los usuarios de la página web. Cada vez que un usuario se registra a través del formulario se ejecuta un script en PHP que envía la información a la base de datos. El nombre, los apellidos, el email y la contraseña son campos obligatorios para crear una nueva entrada. Los demás campos, ciudad, provincia, código postal y número de teléfono, son opcionales. Esta es la otra tabla que tampoco se relaciona con ninguna otra.

- **empresas**

  Esta es la tabla principal donde se guardan todos los datos de las empresas obtenidos mediante el script de web scraping. A continuación se detallan sus atributos:

  - **web** La URL de la página web de la empresa.
  - **plantilla** El número de empleados que trabajan en la empresa.
  - **territorio_historico** La provincia en la que está alojada la empresa.
  - **codigo_postal** El código postal en el que está alojada la empresa.
  - **nombre_comercial** El nombre comercial registrado por la empresa.
  - **direccion** La dirección física en la que está alojada la empresa.

- **actividad** Una descripción sobre la actividad comercial a la que se dedica la empresa.
- **productos** Una lista de los productos o servicios que ofrece la empresa.
- **fecha_hora** La fecha y la hora en la que se registró esta empresa en la base de datos.
- **ultima_actualizacion** La fecha de la última vez que se modificó algún dato relacionado con la empresa en la fuente.
- **teléfono** El número de teléfono de la empresa.
- **email** El correo electrónico de la empresa.
- **municipio** El municipio en el que está alojada la empresa.

- **os**

  Tabla que almacena los sistemas operativos diferentes encontrados en los servidores de las empresas escaneadas.

- **js**

  Tabla que almacena las distintas versiones de JavaScript utilizadas por las páginas web de las empresas escaneadas.

- **platform**

  Tabla que almacena las distintas plataformas utilizadas por los servidores de las empresas escaneadas.

- **cms**

  Tabla que almacena todas las versiones diferentes de los CMS encontrados entre las páginas web de las empresas escaneadas.

- **vulns**

  Tabla que almacena todos los tipos de vulnerabilidades distintas relacionadas con los CMS encontrados, el número de vulnerabilidades que afectan a una misma versión de CMS y el enlace que lleva la página donde se detalla información sobre las mismas.

- **os_conex**

  Esta tabla relaciona un elemento de la tabla os con un elemento de la tabla empresas. Se creó para evitar repeticiones en la tabla os.

- **js_conex**

  Esta tabla relaciona un elemento de la tabla js con un elemento de la tabla empresas. Se creó para evitar repeticiones en la tabla js.

- **platform_conex**

  Esta tabla relaciona un elemento de la tabla platform con un elemento de la tabla empresas. Se creó para evitar repeticiones en la tabla platform.

- **cms_conex**

  Esta tabla relaciona un elemento de la tabla cms con un elemento de la tabla empresas. Se creó para evitar repeticiones en la tabla cms.

- **vulns_conex**

  Esta tabla relaciona un elemento de la tabla vulns con un elemento de la tabla empresas. Se creó para evitar repeticiones en la tabla vulns.

5.2. Diagrama de clases

El diagrama de clases reflejará fielmente la estructura de nuestra aplicación. Este tipo de diagramas representan las clases de un sistema que utiliza un lenguaje orientado a objetos. Diseñaremos las clases, sus atributos y sus métodos que posteriormente serán implementados en el lenguaje Python.

Se han dividido los diagramas en dos. Uno para el script de web scraping y otro para el script de vulnerabilidades que tendrá más clases ya que el código es mucho más complejo.

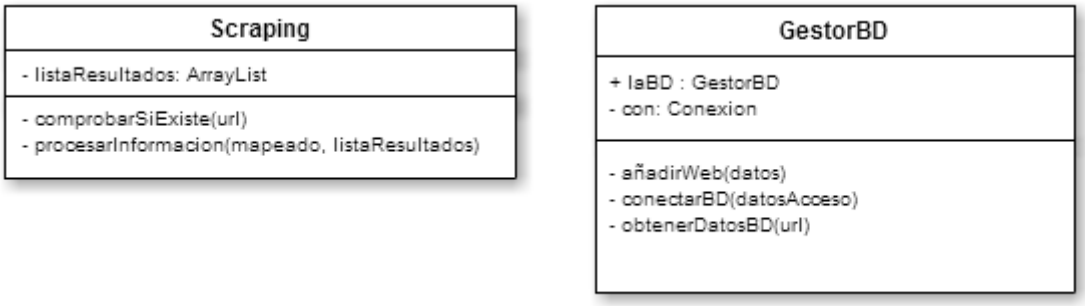

Figura 15: Diagrama de clases (Scrip de web scraping versión 1).

A continuación se explicará detalladamente cada clase:

- **Scraping**

  Es la clase principal del script de web scraping. Se encarga de recoger los datos del fichero JSON, comprobar si alguno ya está en la base de datos y enviar los que no están para su almacenamiento.

- **GestorBD**

  Se encarga de todas las comunicaciones con la base de datos del script de web scraping, tanto obtener todos los datos que había previamente de la tabla de empresas como ejecutar las sentencias SQL necesarias para introducir los datos nuevos.

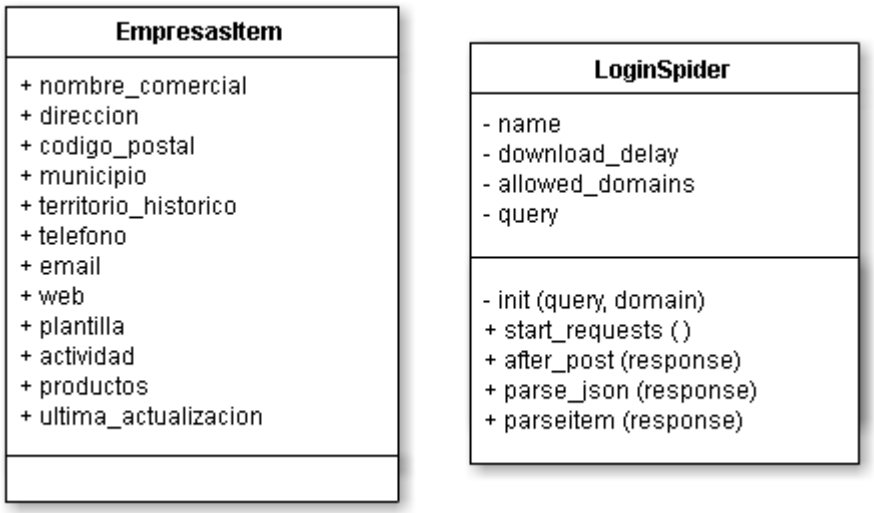

Figura 16: Diagrama de clases (Scrip de web scraping versión 2).

A continuación se explicará detalladamente cada clase:

- **EmpresasItem**

  Esta clase tiene como finalidad guardar toda la información relacionada con una empresa.

- **LoginSpider**

  Es la clase principal. Rellena los datos de búsqueda de cada empresa, accede al enlace de detalles, extrae los contenidos de cada empresa, los guardar en la clase Empresa y repite el proceso hasta terminar todas las empresas de la página. Pasa a la siguiente página y vuelve a repetir desde el principio, hasta llegar a la última página de esta búsqueda.

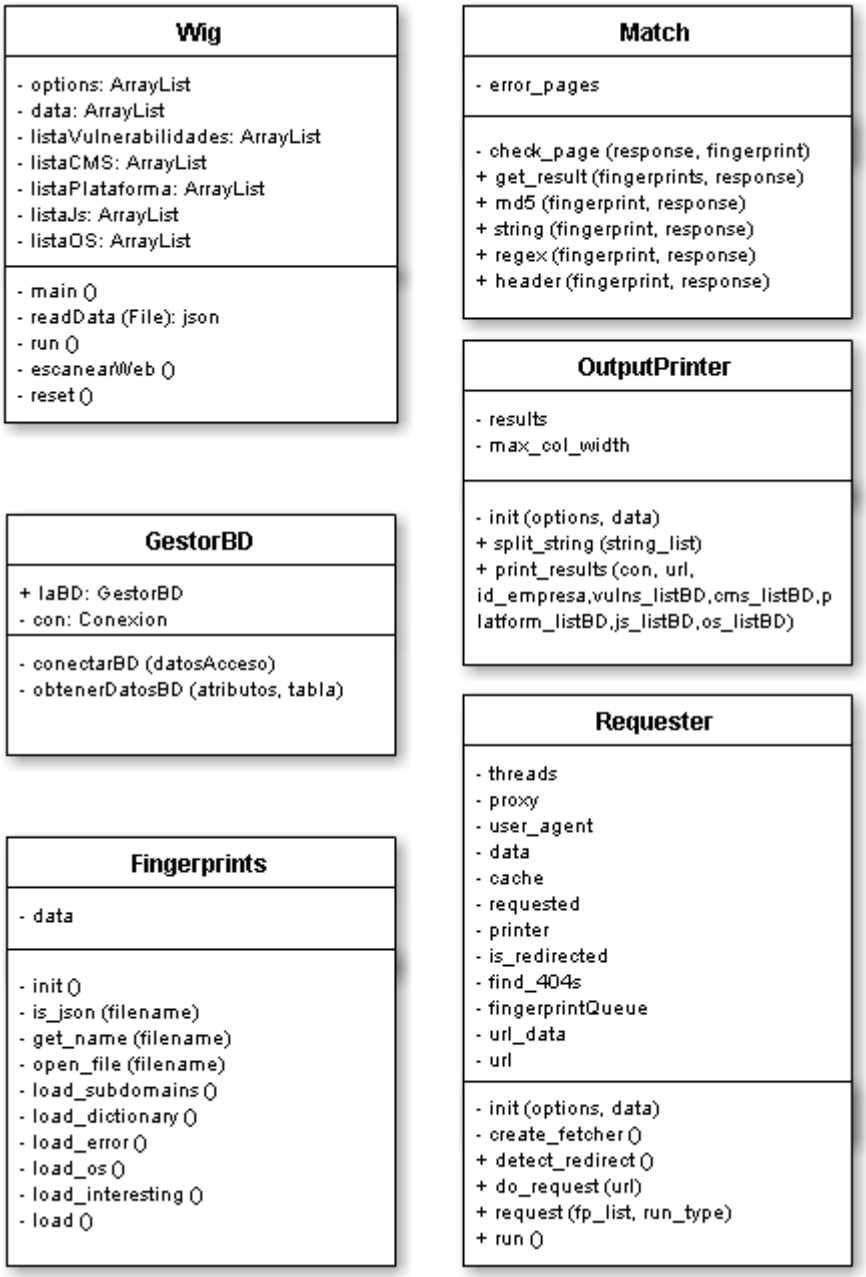

Figura 17: Diagrama de clases (Script de vulnerabilidades parte 1).

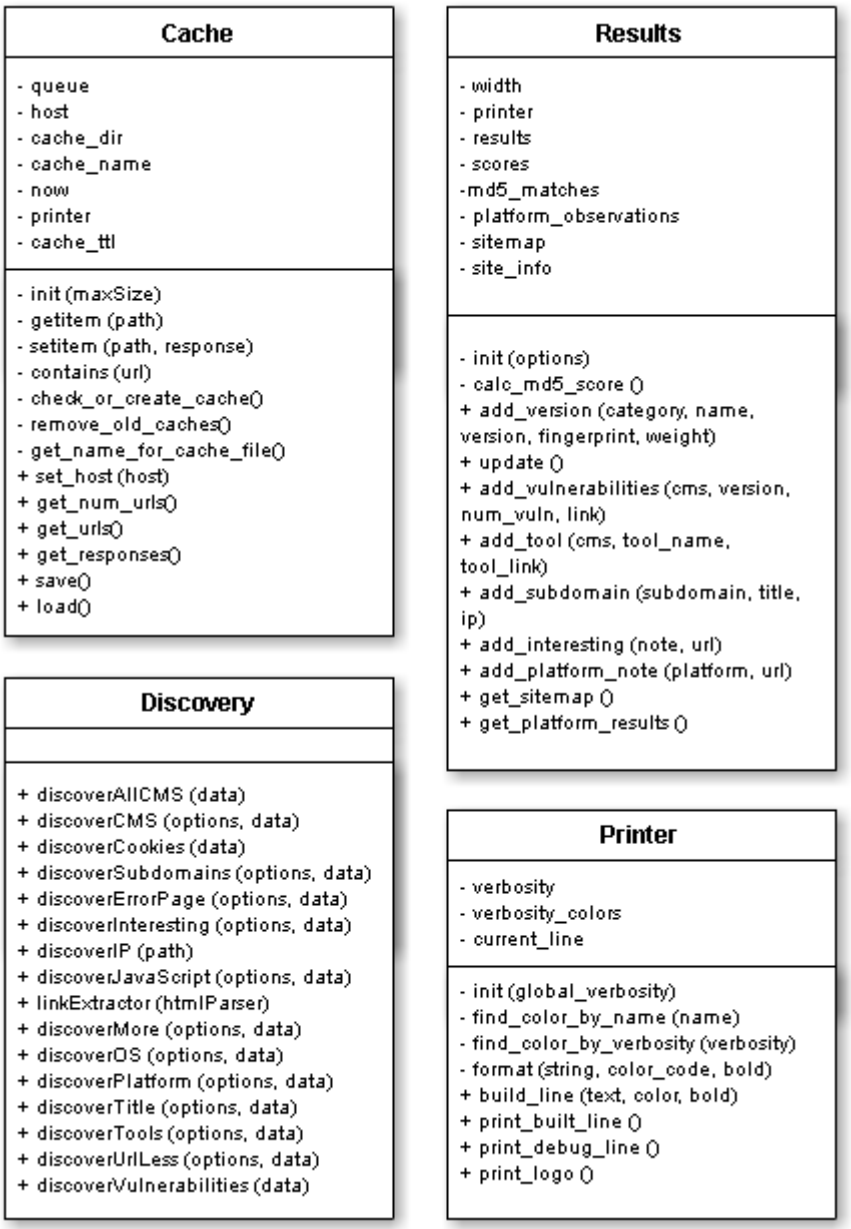

Figura 18: Diagrama de clases (Script de vulnerabilidades parte 2).

A continuación se explicará detalladamente cada clase:

- **Wig**

  Es la clase principal del script de web de vulnerabilidades. Se encarga de leer los ficheros de configuración y gestionar el resto de clases.

- **GestorBD**

  Se encarga de todas las comunicaciones con la base de datos del script de web vulnerabilidades. Obtiene todos los datos de la tabla de empresas para devolvérselos a la clase Wig y ejecuta sentencias SQL para introducir los datos obtenidos sobre vulnerabilidades, cms, etc.

- **Fingerprints**

  Se encarga de cargar los ficheros JSON con información para identificar los diferentes sistemas operativos que se ejecutan en el servidor, las versiones de JavaScript que utiliza la página web que se va a analizar, las posibles plataformas que esté utilizando y el CMS con la que esté construida. También carga información almacenada de CVE details para determinar si el CMS detectado tiene alguna vulnerabilidad.

- **Printer**

  Se encarga de imprimir por línea de comandos información de interés sobre el desarrollo del escaneo de páginas web o si se ha producido algún error.

- **Match**

  Se encarga de registrar las coincidencias encontradas entre la información obtenida de la página web y la información almacenada que previamente había sido cargada por la clase Fingerprints.

- **Cache**

  Se encarga de cargar la cache y toda la información relacionada (host, número de URL, respuesta del servidor, etc.) y de guardarla.

- **OutputPrinter**

  Se encargar de gestionar todas las comunicaciones con la clase GestorBD y la clase Printer para indicarles que se ha de imprimir por consola o que se ha de almacenar en la base de datos. Cualquier dato de salida pasa por esta clase.

- **Results**

  Se encarga de almacenar temporalmente toda la información obtenida acerca de la página web escaneada (versión, vulnerabilidades, herramientas, subdominios, etc.).

- **Requester**

  Se encargar de gestionar si la URL a escanear redirige a otra URL diferente. Además también realiza las peticiones a la página web.

- **Discovery**

  Se encarga de detectar toda la información de la página web escaneada (versión de CMS, cookies, ip, JavaScript, herramientas, etc.)

5.3. Diagrama de secuencia

En esta sección se van a representar los diagramas de secuencia en los que se puede ver cómo interactúan las clases entre sí para realizar una determinada función.

Se muestran en orden el diagrama de secuencia del script de web scraping versión 1, el del script de web scraping versión 2 y el del script de vulnerabilidades:

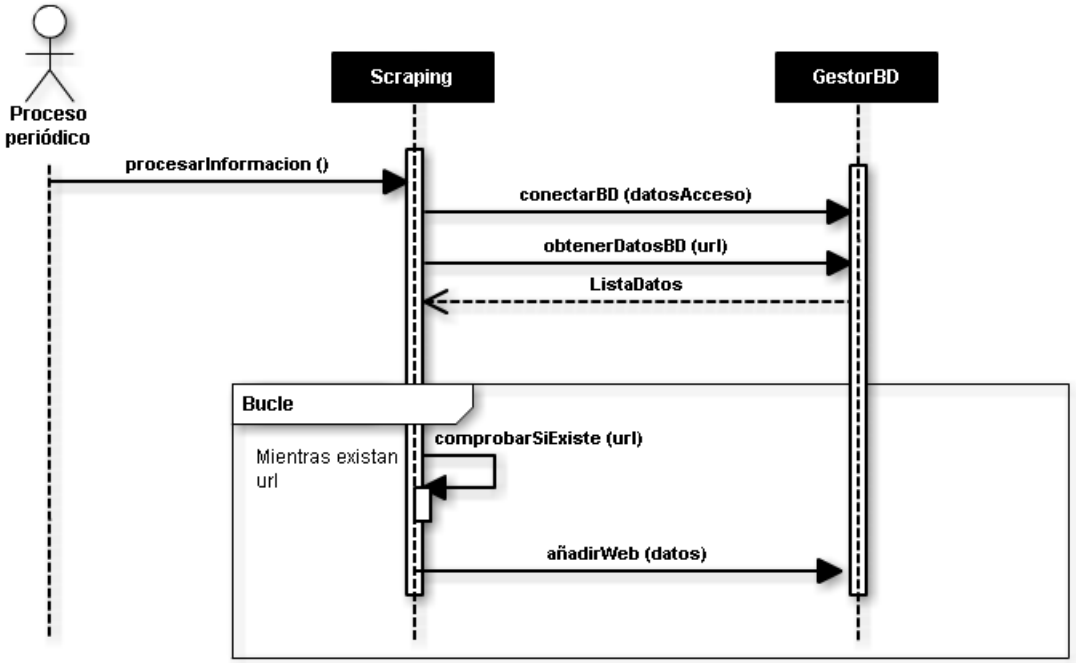

Figura 19: Diagrama de secuencia del script de web scraping versión 1.

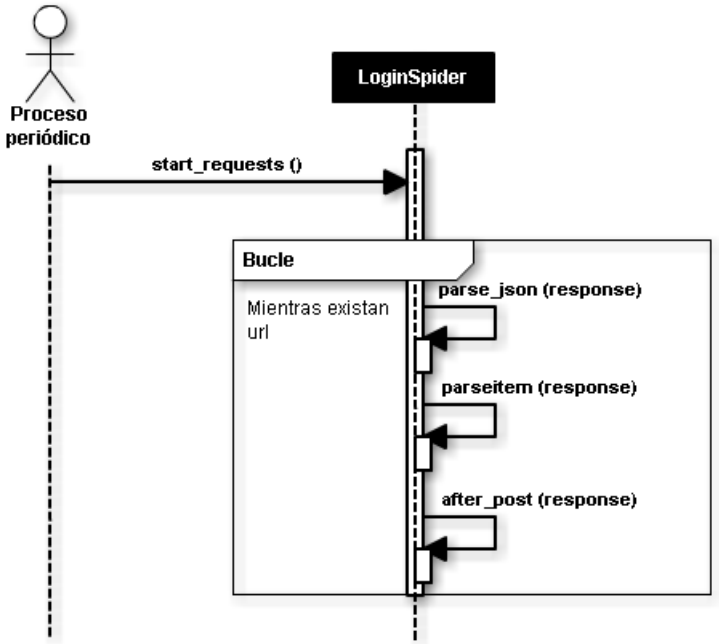

Figura 20: Diagrama de secuencia del script de web scraping versión 2.

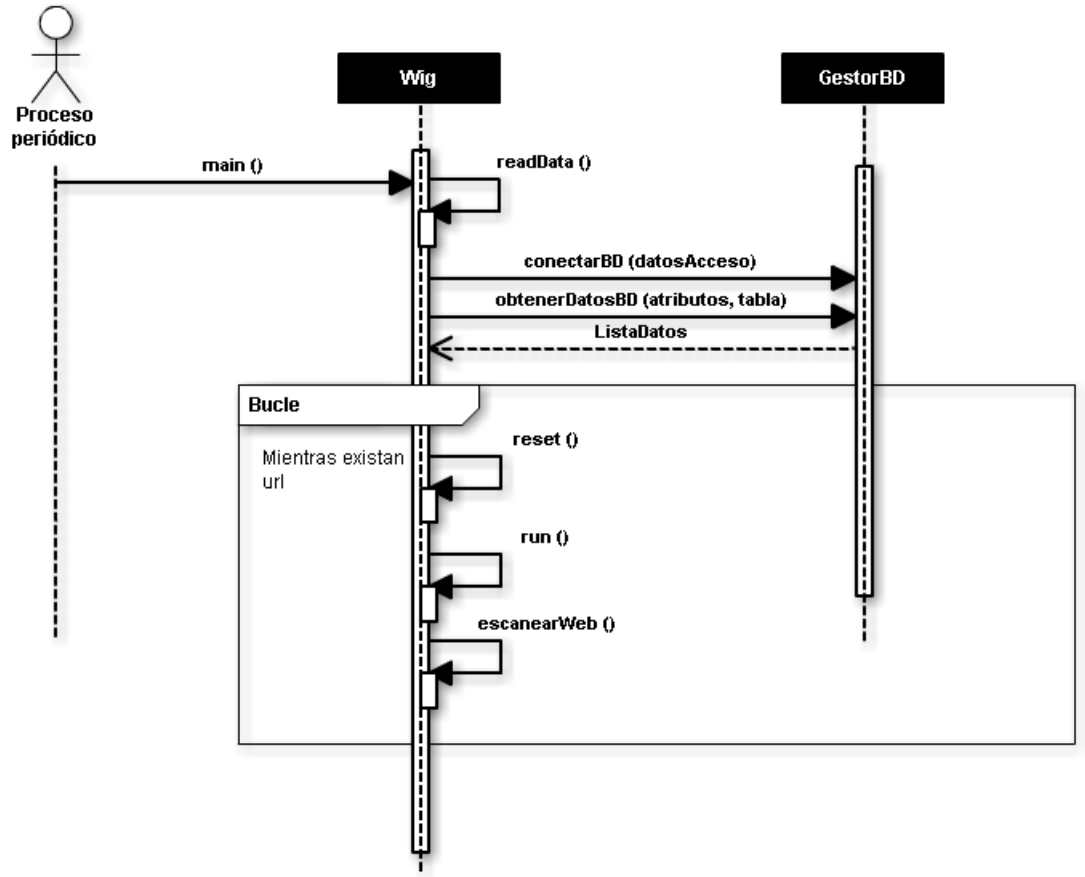

Figura 21: Diagrama de secuencia del script de vulnerabilidades (parte 1).

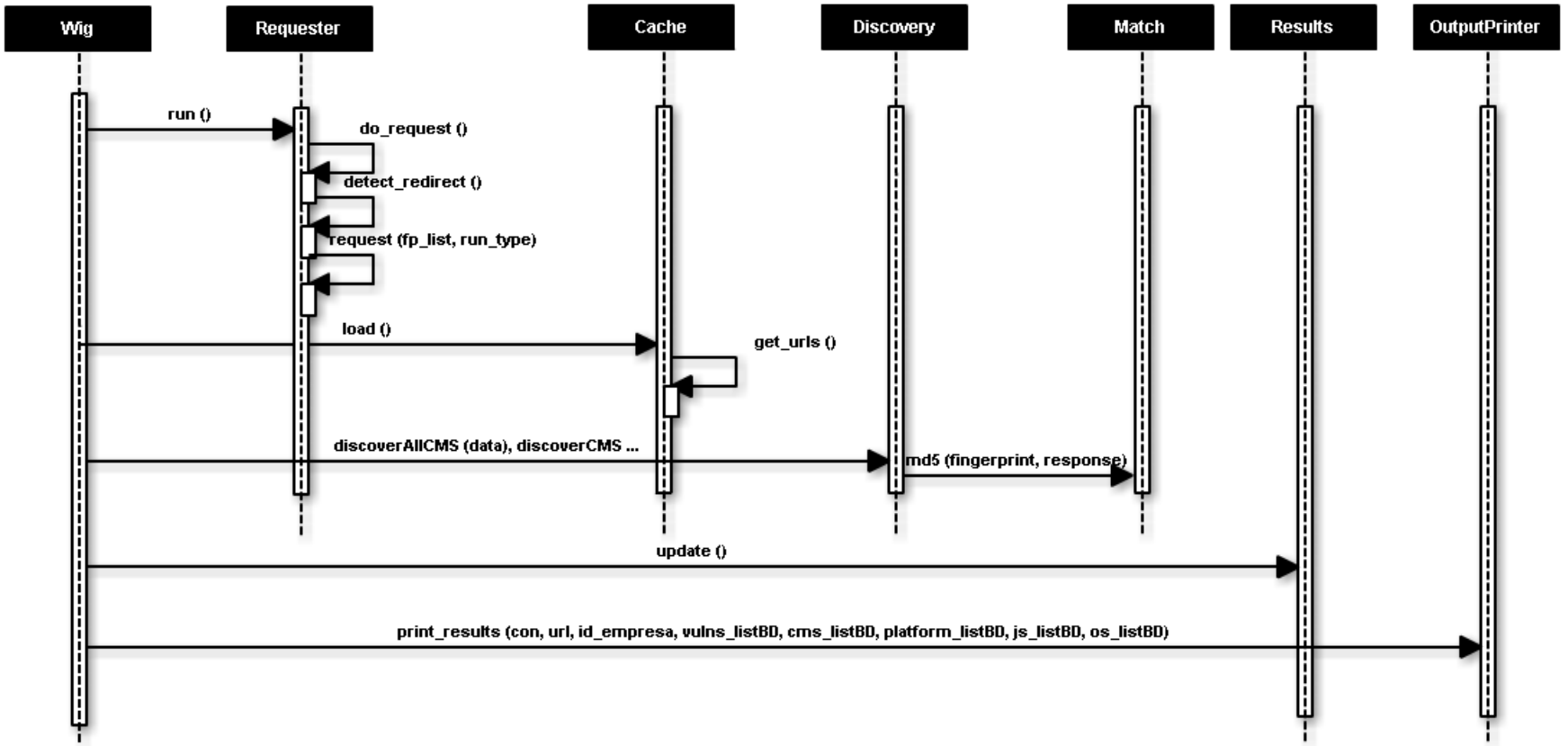

Figura 22: Diagrama de secuencia del script de vulnerabilidades (parte 2).

6. Desarrollo

En esta sección vamos a ver paso a paso como ha sido el desarrollo del proyecto desde el inicio hasta el final.

6.1. Import.io

Import.io es una herramienta online que facilita mucha el *web scraping*. A primera vista parece simple de configurar, pero no es tan sencillo. Permite descargar su aplicación y generar un script siguiendo algunos pasos mucho más sencillos que escribir el código.

Esta herramienta se utilizó para obtener un listado de nombres de empresa y sus URL del sitio web de páginas amarillas. A continuación pasaré a detallar su funcionamiento:

Lo primero que hay que hacer, una vez abierta la aplicación, es escoger la opción de Crawler (Figura 23). Esto nos va a permitir que el script navegue a través de varias URL diferente de una misma página web buscando listados similares al que le enseñemos.

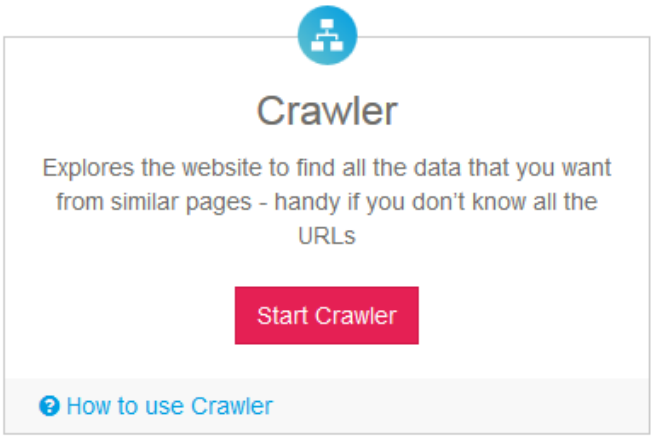

Figura 23: Import.io. Seleccionar crawler.

Una vez hecho esto se abre una nueva ventana donde debemos introducir la URL donde esté el listado con los datos que queremos obtener. En este caso hemos elegido la página web de páginas amarillas (Figura 24). Este ha sido un listado que previamente hemos buscado filtrando para que nos muestre las empresas de Bizkaia de un determinado sector.

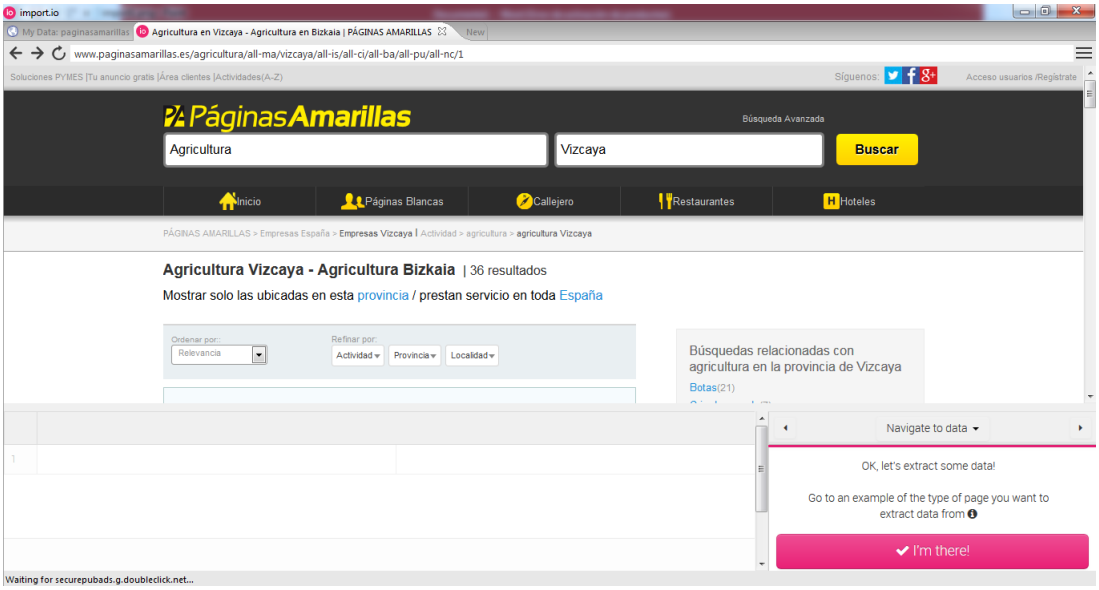

Figura 24: Import.io. Introducir la URL.

Después nos ofrecerá opciones para configurarlo según nuestras necesidades. Tenemos que escoger la opción de múltiples entradas para que seleccione varios elementos de la misma página porque se trata de un listado (Figura 25).

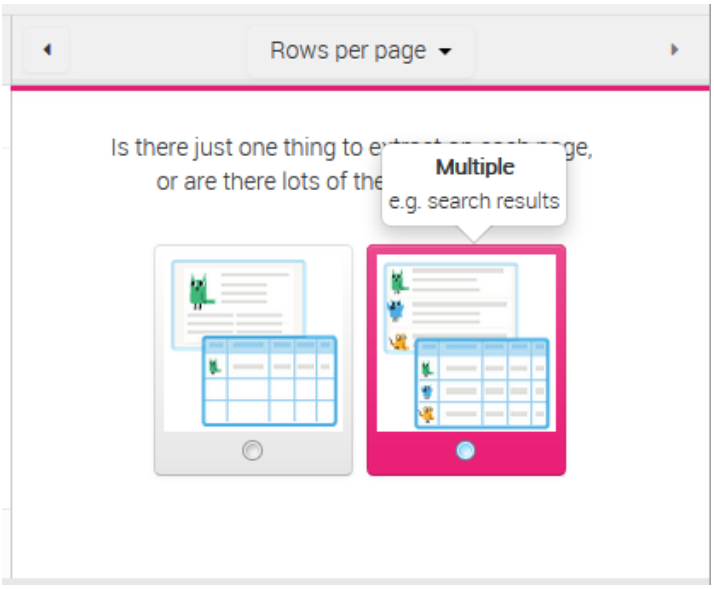

Figura 25: Import.io. Seleccionar multiple.

Seguidamente le indicamos qué datos ha de recoger por cada elemento. En nuestro caso solo nos interesa el nombre de la empresa y la URL (Figura 26). Serían de interés otros datos pero esta herramienta no nos permite seleccionar datos que no estén visibles en la misma página.

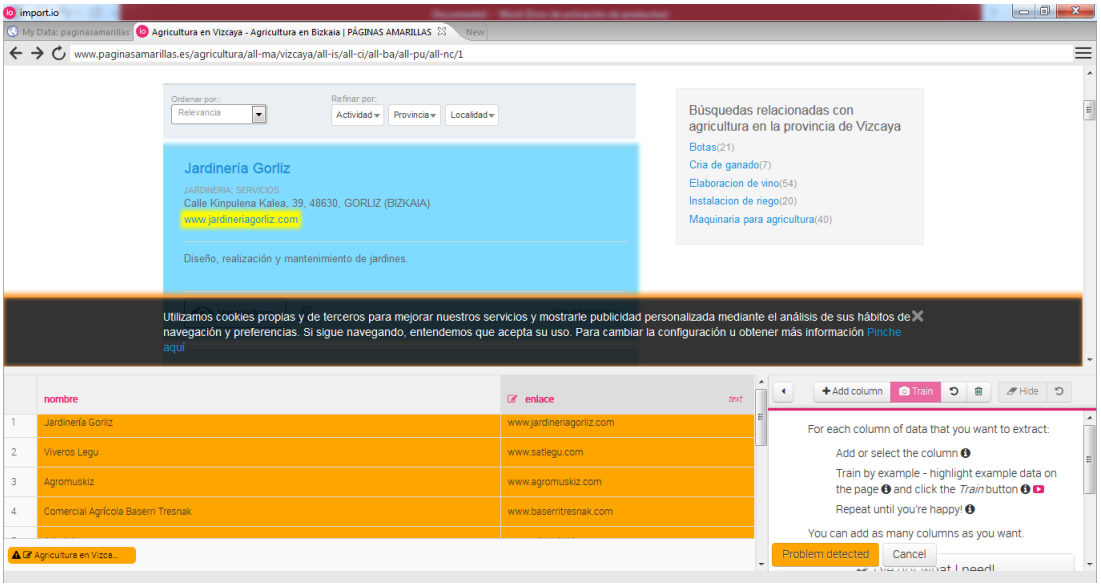

Figura 26: Import.io. Seleccionar los datos.

Por último solo tenemos que añadir cuatro páginas similares más para que el sistema compruebe si realmente se está seleccionando lo deseado. A partir de aquí, si todo va bien, una vez que carguemos una página similar el sistema detectará automáticamente los nombres y las URL del nuevo listado.

Cuando terminamos aparece un botón “Run Crawler” que nos lleva a una pantalla en la que podemos configurar el rastreo de listados a través de toda la web, en nuestro caso páginas amarillas, con los parámetros que le indiquemos (Figura 27).

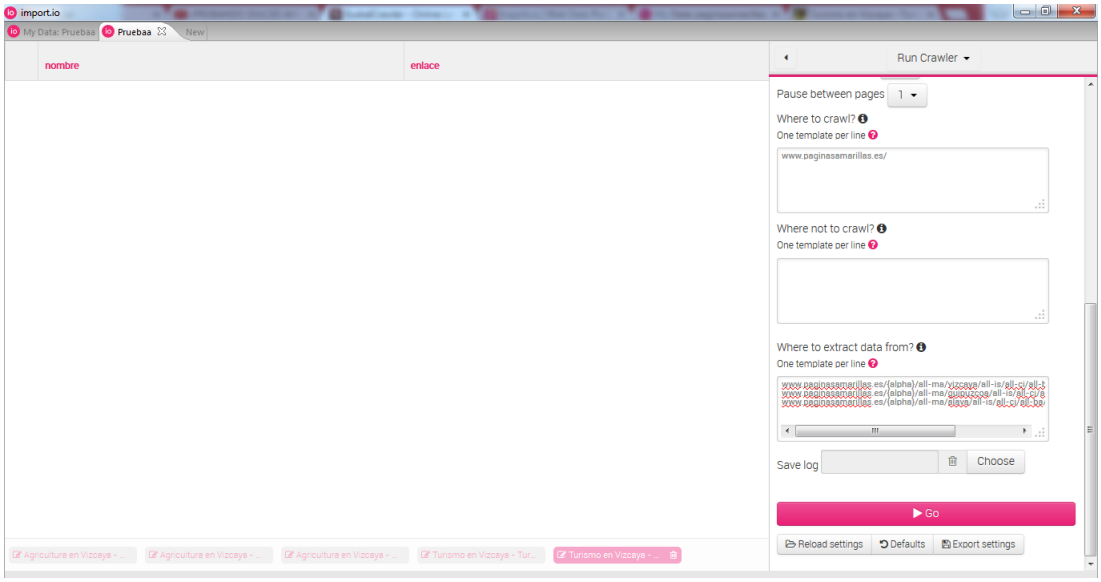

Figura 27: Import.io. Ejecutar la búsqueda.

Como se puede apreciar en la imagen Import.io ofrece varias opciones para configurar el rastreo de entradas. Nos interesa concretamente “Where to extract data from?” en la que sustituiremos el sector (agricultura) por la cadena {alpha} para indicarle que busque listados de todos los sectores. También sustituiremos con la cadena {num}$ donde iría el número de la página para indicarle que rastree en todas las páginas de resultados que encuentre por sector. Por último añadiremos dos nuevas URL idénticas pero cambiando el nombre de la provincia para obtener resultados de las tres provincias vascas.

También se puede ejecutar import.io a través de la línea de comandos indicándole estos mismos parámetros en un archivo de configuración e indicándole también un fichero de autentificación con nuestros datos. Este sería el comando necesario:

./import.io -crawl <crawl-config.json><auth-config.json>

Esto genera un fichero en formato JSON con todos los resultados obtenidos.

6.2. Script web scraping

Ahora, esta información en formato JSON la introduciremos en la base de datos. Para ello he diseñado un script que he llamado script de web scraping (aunque el scraping ya esté realizado).

Este script se encargará de leer el fichero en formato JSON y transformar los datos en sentencias SQL para ser enviadas a la base de datos. Puede parecer una tarea sencilla pero durante su ejecución surgieron bastantes problemas que detallo a continuación.

6.2.1. Problemas con versiones de Python

A la hora de buscar ejemplos de código Python para leer ficheros en formato JSON encontré ejemplos escritos en Python 2. Este proyecto se está desarrollando bajo Python 3. Y para mi sorpresa y desagrado estas versiones no son compatibles y algunas instrucciones de código de Python 2 no servían o funcionaban mal. Esto me dio bastantes dolores de cabeza y perdí mucho tiempo buscando las librerías, módulos e instrucciones de Python 3 necesarias para realizar una tarea relativamente sencilla.

6.2.2. Problemas con JSON

Otro de los problemas a los que tuve que enfrentarme en esta fase inicial del desarrollo fue un fallo de import.io al almacenar la información en formato JSON. En algunas entradas guardaba dos valores para un atributo. Es decir, si el atributo es URL, guardaba un array de dos elementos, con la URL y otro valor desconocido e inútil.

El problema es que al parsear la información daba error en estas entradas porque no se esperaba un array, sino sencillamente un valor. Este problema fue difícil de identificar. Se solucionó filtrando las entradas con arrays en sus atributos para seleccionar el elemento del array que interesaba.

A continuación muestro un ejemplo:

- {“enlace”:“www.url1.com”,“nombre”:“NombreEjemplo1”}

  (Línea esperada en el fichero en formato JSON que genera import.io)

- {“enlace”:[“www.url2.com”,“StringInfructuoso”],“nombre”:“NombreEjemplo2”}

  (Línea no esperada que causó el problema)

6.2.3. Problemas con los apóstrofos

Puede parecer una tontería pero hasta los detalles más pequeños pueden parar la ejecución de todo un proyecto cuando se trata de código. En este caso fueron los apóstrofos que llevaban algunos nombres de las empresas. Interferían en la secuencia SQL, no se enviaba completa y por lo tanto el script fallaba. La solución fue analizar todos los nombres en busca de apóstrofos y eliminarlos cuando se detectara alguno.

6.3. Script de web scraping versión 2

El problema de utilizar import.io sobre un listado de empresas como páginas amarillas era que proporcionaba muy poca información de utilidad sobre las empresas. Así que decidí buscar otras fuentes de información y encontré CIVEX, que es un buscador de empresas gestionado por el gobierno vasco.

El problema que me surgió con CIVEX es que no muestra un listado de empresas para poder hacer el web scraping, sino que primero hay que introducir una cadena de texto a buscar y entonces muestra unos resultados. Por lo tanto no se podía utilizar import.io sobre él.

Así que decidí diseñar un script desde cero para que rastreara los resultados devueltos por cadenas de texto como aa, ab, ac, etc. El script accede a la página de CIVEX, hace la búsqueda, recoge los resultados, los almacena en la base de datos y repite el proceso.

6.4. Script de vulnerabilidades

Para este script se ha tomado como base el proyecto WIG. El problema es que WIG estaba diseñado para escanear una única URL por línea de comandos. Por lo tanto la primera modificación necesaria fue obtener toda la información almacenada en la base de datos para procesarla.

Una vez obtenidas todas las URL de la base de datos este script escanea una a una las URL buscando coincidencias de strings y ficheros conocidos para identificar el CMS y su versión más probables. También obtiene otro tipo de información como el sistema operativo que utiliza el servidor, la versión de JavaScript y si tiene alguna plataforma. Toda esta información la obtiene del servidor y de cabeceras con etiquetas como 'x-powered-by'.

Posteriormente y con la versión del CMS ya obtenida se compara contra una base de datos de vulnerabilidades online (CVE Details [3]) para conocer si esa versión tiene vulnerabilidades, si es así se obtiene el número de las mismas y el link donde se indican todos los detalles de la vulnerabilidad.

Por último solo falta almacenar toda esta información en la base de datos. Y seguidamente se repite este proceso en bucle para cada URL.

6.4.1. Problemas con el módulo de MySQL

Para la comunicación entre Python y MySQL es necesario instalar un módulo en Python e importarlo en el código donde vaya a utilizarse. Esto me dio bastantes problemas a la hora de llevarlo a la práctica. Estuve buscando un módulo un módulo oficial en la página de Python y por foros, entre otros encontré mysqlclient pero fallaban al instalarlos o al ejecutarlos en el equipo. Así que finalmente escogí un módulo llamado MySQLdb, que no es un módulo oficial, pero se instalaba y se ejecutaba correctamente.

El problema con este módulo es que estaba desarrollado para Python 2. Y aunque en mi equipo funcionaba bajo Python 3 al probarlo en otros equipos fallaba. Por

[3] http://www.cvedetails.com/

lo tanto, se descartó y se pasó a utilizar el módulo pymysql que estaba desarrollado para Python 3 y funcionaba sin mayor problema.

6.5. Página web

A continuación se detallará el desarrollo del sitio web. Se utilizó también PyCharm como IDE para el desarrollo del código ya que permite también trabajar con HTML, PHP, JavaScript y Bootstrap.

6.5.1. Estructura inicial y diseño

Para la estructura inicial de la web se escogió una plantilla de Bootstrap llamada Simply Me. Esto agilizó el proceso de creación y permitió un estilo elegante y uniforme para la página web. Después fue necesario modificar la barra de navegación para crear las secciones necesarias y crear también las páginas que conformarían la web (estadísticas, contacto, conocenos, home, acceso, registro, etc...).

Después se fueron añadiendo textos, imágenes, también se fue modificando el estilo (centrar textos, cuadrar imágenes, cambiar colores, etc.) para hacer más atractivo el diseño de la página.

6.5.2. Gráficos

Una vez hecha la estructura y el diseño iniciales se pasó a configurar los gráficos, que es la parte fundamental de la página web. Se probaron varias herramientas pero al final se escogió Plot.ly por su popularidad, su facilidad de uso y su versatilidad.

Inicialmente se diseñó únicamente un gráfico circular (gráfico de tarta). Un script en PHP recoge la información necesaria de la base de datos, para ello tiene que acceder a la tabla que guardan los CMS únicos que se han registrado y a la tabla donde se relacionan esos CMS con las empresas para saber cuántos hay de cada uno. También se recogen las versiones de cada CMS que están registradas en las mismas tablas.

Con estos datos recogidos ya se puede mostrar el gráfico de los CMS más utilizados. Faltaría ahora reflejar cuáles de ellos tienen más o menos vulnerabilidades. Esto se ha hecho con un código de diez colores en un degradado entre rojo y verde, siendo verde el color que representa el CMS o la versión que tiene un número bajo de vulnerabilidades y el rojo el que tiene un número alto de vulnerabilidades (Figura 28).

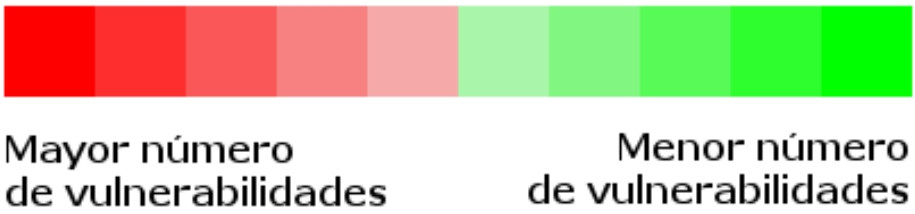

Figura 28: Ejemplo del degradado de colores del sitio web.

Posteriormente se eliminó la leyenda que tienen los gráficos de Plot.ly por defecto y se creó una leyenda propia para poder filtrar las estadísticas por versiones, es decir, que si haces click en algún elemento de la leyenda, que serán nombres de CMS, la aplicación genera otro gráfico exclusivamente con las versiones de ese CMS.

Por último también se diseñó una lista de checkbox para poder filtrar los datos por provincias. De esta manera se pueden ver los datos de una, dos o las tres provincias de la comunidad autónoma del País Vasco, dependiendo de los checkbox que estén seleccionados. Y para hacerlo más ágil y cómodo no es necesario hacer click en ningún botón para generar el nuevo gráfico, se genera de manera automática al modificar algún campo de checkbox (Figura 29).

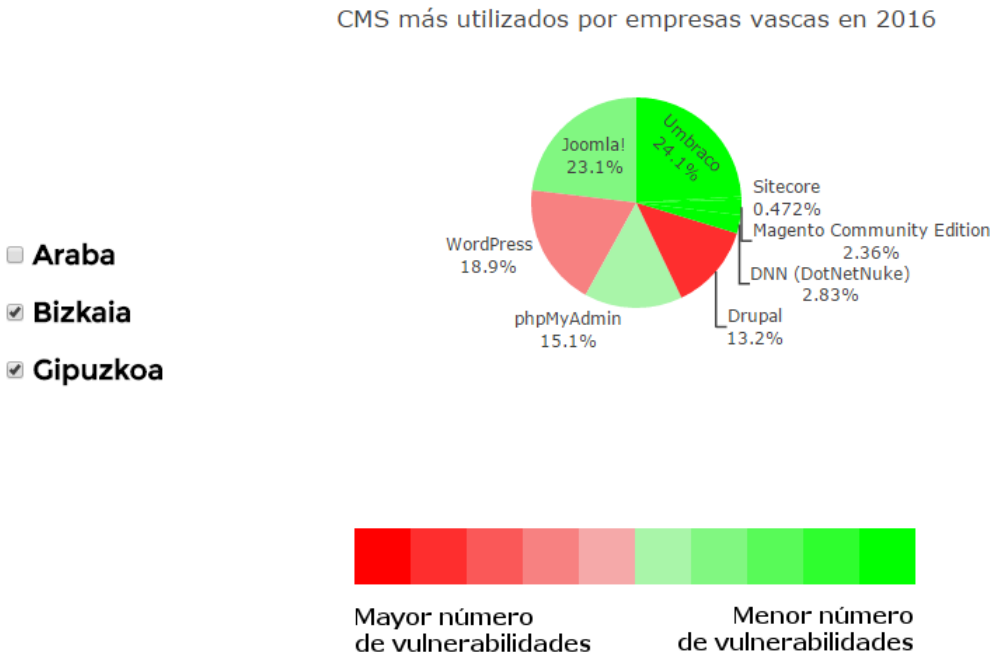

Figura 29: Ejemplo del checkbox para filtrar por provincias en el sitio web.

Una vez acabado el gráfico circular se modificaron los estilos para que se vieran bien la leyenda y la lista de checkbox. Y a continuación se diseñó otro gráfico, esta vez un gráfico de barras, porque me pareció que quizás el gráfico circular no era la mejor manera de representar los CMS más utilizados y con el gráfico de barras se puede ver de forma más clara.

6.5.3. Mapa

Para el mapa se utilizó la versión 3 del API de Google Maps. Primero se genera un mapa vacío indicándole las coordenadas y el zoom correcto para que muestre la comunidad autónoma del País Vasco.

Después falta cargar los marcadores. Para esto se creó un script en PHP que obtiene la información necesaria de la base de datos. Primero se obtienen todos los elementos de la tabla que relaciona los CMS y las empresas. Posteriormente, con los id de las empresas se obtiene la dirección física de cada uno de ellos y con los id de los CMS se obtiene el CMS para asignarle el icono de ese CMS al marcador.

Una vez obtenida toda la información necesaria de la base de datos se le pasa el listado de direcciones a la api de google maps para que las transforme en coordenadas de latitud y longitud.

Y ya solo falta crear los marcadores con el listado de coordenadas y el listado de los iconos personalizados para cada CMS y añadirlos al mapa.

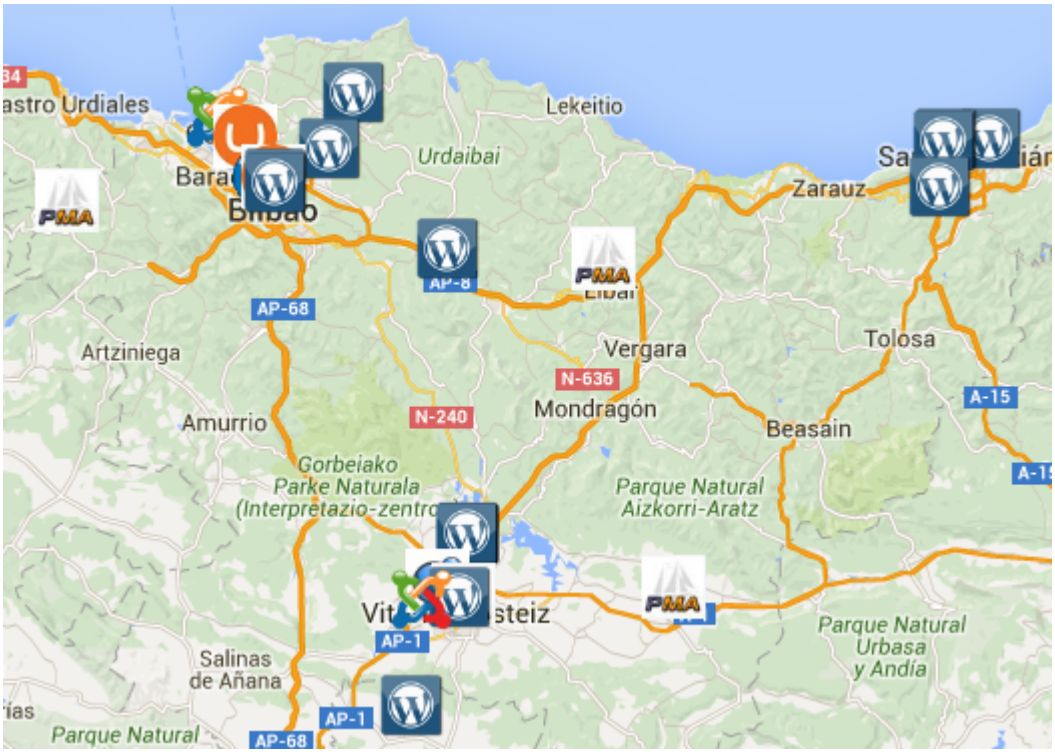

Figura 30: Ejemplo del mapa del sitio web.

6.5.4. Formularios

Para los formularios se escogió unas plantillas de formularios de Bootstrap. Se modificaron según las necesidades de nuestra aplicación, añadiendo campos que faltaban o eliminando lo que sobraban. Y se añadieron a la página web.

Posteriormente se crearon los scripts necesarios en PHP. Para el formulario de contacto se recogen los campos introducidos por el usuario (el título, el mensaje y su correo electrónico) y se envían a un script que ejecuta una instrucción que envía el correo electrónico a mi bandeja de entrada. Se ha utilizado el protocolo SMTP.

Ejemplo de instrucción para el envío de correos electrónicos:

```
mail($emailDestinatario, $título, $mensaje, $emailUsuario);
```

Para el formulario de registro, a diferencia del formulario de contacto, es necesario acceder a la base de datos para obtener la información sobre los usuarios existentes. Con esta información se comprueba que el usuario no se registre con un email ya registrado. Se creó un script que aparte de comprobar que el email no existe en la base de datos también se encarga de comprobar que los campos obligatorios han sido introducidos. Si todo está correcto se crea un usuario nuevo y se introduce en la base de datos.

Para el formulario de acceso fue necesario crear una variable de sesión que registrara si un usuario había accedido al sistema. Dependiendo de esta variable se modifica parte de la barra de navegación eliminando el login y registro (que ya carecen de sentido) y añadiendo una sección de usuario y un enlace para que el usuario pueda cerrar sesión.

6.5.5. Idiomas

Para los idiomas se han creado tres ficheros con listas para todas las cadenas de texto de la página web (un fichero por cada idioma). Se ha añadido una nueva variable de sesión que registre el idioma, por defecto el español, y que el usuario puede cambiar en cualquiera de las páginas con un selector situado en la barra inferior.

Y donde antes se escribía el texto en español ahora se ha cambiado por una variable PHP que accede a la lista correspondiente, dependiendo de la variable de sesión, para recoger una cadena de texto. De este modo están todas las cadenas de texto recogidas en un mismo fichero y es más sencillo modificarlas, eliminarlas o añadir nuevas.

6.5.6. Problemas con la api de google maps

Los problemas con la api de google maps empezaron con sus versiones. Empecé utilizando la versión 2 para crear el mapa vacío y centrarlo. Posteriormente probé a añadir un marcador y funcionaba bien. El problema vino al añadir una lista de marcadores, no mostraba ninguno. También daba problemas a la hora de cambiar el icono, así que terminé utilizando la versión 3.

El otro problema de esta api fue a la hora de cargar los marcadores en el mapa. Al parecer, google, de forma gratuita, no permite más de una petición por segundo para trasformar una dirección en coordenadas de longitud y latitud. Por lo que fue necesario poner un retraso al script para que permitiera obtener todas las coordenadas. El problema es que los marcadores tardaban en cargarse algunos minutos y esto no era práctico ni cómodo.

La solución a este problema fue obtener todos los marcadores previamente y almacenarlos en la base de datos, en una nueva tabla destinada únicamente para este fin. Posteriormente cuando cualquier usuario acceda a la sección de mapa solo será necesario cargarlos desde la base de datos y mostrarlos, por lo que puede hacerse mucho más rápido y es más cómodo para el usuario.

6.5.7. Problemas con Bootstrap

Bootstrap puede ser una herramienta muy útil y que te ahorre mucho tiempo pero si no se tiene mucho conocimiento sobre ella puede resultar frustrante intentar diseñar algo. El principal problema de Bootstrap es las plantillas ya diseñadas tienen hojas de estilo muy complejas y con muchos parámetros. Por lo tanto, si deseamos modificar algo, se nos va a hacer complicado y se puede descuadrar todo. Por ejemplo, si quieres añadir un campo más a un formulario, es posible que se descuadre todo el formulario y algunas áreas de texto se salgan de los márgenes.

7. Verificación y evaluación

En esta sección se explicarán las pruebas realizadas durante el proyecto. Se han realizado pruebas sobre los scripts y sobre la página web para comprobar su correcto funcionamiento en situaciones normales y situaciones inusuales o extremas.

Las pruebas realizadas son de caja negra, es decir, que solo se tiene en cuenta la entrada al sistema y el resultado de la salida, sin considerar lo que suceda internamente. Se han descartado otro tipo de pruebas porque estas son las más útiles para el sistema y las que determinarán como se comportará ante un usuario.

Para todas las pruebas se ha establecido un color rojo que representa que la prueba no se ejecutó como debería y un color verde que representa que la prueba sí se ejecutó como debería.

7.1. Pruebas script web scraping

A continuación se muestran las pruebas realizadas en el script de web scraping:

Prueba	Resultado esperado	Resultado obtenido
Sin conexión	Mostrar un mensaje de que no hay conexión a internet.	No se ejecuta el script.
Texto, que no es una URL, en el campo de la URL	No almacenarlo en la base de datos	Se almacena en la base de datos.
Apóstrofos que pueden interferir en la sentencia SQL	Eliminar los apóstrofos encontrados.	Se eliminan los apóstrofos encontrados.
Falta el fichero de configuración	Mostrar un mensaje de que falta el fichero de configuración.	El script falla.
Ejecutar con conexión	Guardar todos los datos en la base de datos.	Guarda todos los datos en la base de datos.
Ejecutar con fallos en el fichero de datos en formato JSON	Mostrar mensaje de error.	Muestra el mensaje de error y la aplicación se cierra.

Tabla 8: Pruebas script de web scraping.

7.2. Pruebas script de vulnerabilidades

A continuación se muestran las pruebas realizadas en el script de vulnerabilidades:

Prueba	Resultado esperado	Resultado obtenido
Sin conexión	Mostrar un mensaje de que no hay conexión a internet.	Se ejecuta el script pero no escanea ninguna página web.
URL erronea	No escanearla y continuar.	No la escanea y continúa.
Página web sin CMS	Escanear la web, detectar que no tiene CMS y descartarla.	Escanea la web, detecta que no tiene CMS y la descarta.
Falta el fichero de configuración	Mostrar un mensaje de que falta el fichero de configuración.	Muestra el mensaje.
Datos erróneos en el fichero de configuración	Mostrar un mensaje de que hay datos erróneos en el fichero de configuración.	Muestra el mensaje.
Con conexión	Escanear todas las webs correctas.	Escanea las webs.

Tabla 9: Pruebas script de vulnerabilidades.

7.3. Pruebas página web

A continuación se muestran las pruebas realizadas en la página web:

Prueba	Resultado esperado	Resultado obtenido
Envío de formulario de registro vacío	Mostrar un mensaje de que no se ha rellenado el formulario.	Muestra el mensaje.
Envío de formulario de registro con algunos campos obligatorios sin rellenar	Mostrar un mensaje de que no se han rellenado campos obligatorios.	Muestra el mensaje.
Envío de formulario de registro con todos los campos obligatorios rellenados	Almacenar los datos en la base de datos y crear un nuevo usuario.	Almacena los datos en la base de datos y crea un nuevo usuario.
Introducir una cadena de texto aleatoria en el campo de correo electrónico	Mostrar un mensaje de error avisando de que el campo introducido no es un correo electrónico.	Almacena los datos y crea un nuevo usuario.
Introducir datos erróneos en el formulario de acceso	Mostrar un mensaje de que el usuario o la contraseña son incorrectos.	Muestra el mensaje.
Introducir datos correctos en el formulario de acceso	Acceder a la zona de usuario.	Accede a la zona de usuario.

Tabla 10: Pruebas página web.

8. Conclusiones

Ahora toca el momento de la reflexión y la autocrítica. Evaluar objetivamente el trabajo realizado y destacar que se hizo bien y puntualizar también que se hizo mal y por qué. El objetivo es hacer un resumen de los resultados obtenidos y aprender de la elaboración de un proyecto desde su principio a su fin.

También daré una opinión personal sobre el proyecto. Ha sido una experiencia totalmente nueva y muy diferente a como lo imaginaba, y han sucedido muchas cambios y situaciones inesperadas.

8.1. Cambios

Primero pasaremos a ver los cambios principales que han sucedido a lo largo del proyecto. Estos cambios se han debido a decisiones que había que tomar frente a un problema o por la mala gestión del tiempo y del trabajo realizado.

8.1.1. De Crawler a Scraping

El primer cambio del proyecto fue casi al principio. Durante toda la fase de planificación se pensó y se diseñó un *web crawler* para obtener las URL e información de empresas. De hecho también parte de la etapa de captura de requisitos y análisis, se llegó incluso a realizar un diagrama de clases para un *web crawler*.

Pero a la hora de llevarlo a cabo y tras varias consultas con el tutor se decidió descartar la idea por su complejidad. El *web crawler* en si no es excesivamente complicado, más bien lo era determinar si una web pertenecía a una empresa de la comunidad autónoma del País Vasco solo con una URL.

Se optó entonces por una solución más sencilla pero igual de efectiva, el *web scraping*. Y se volvió a plantear el proyecto desde el principio, reestructurando los diagramas y diseño.

8.1.2. De Python a PHP

Este cambio no hace referencia al script de *web scraping* ni al script de vulnerabilidades. Este cambio se produjo en los scripts que generaban los datos que se enviaban a la página web para generar los gráficos y el mapa.

Inicialmente se desarrollaron en Python, ya que era el lenguaje con el que había estado trabajando hasta ahora. Pero a la hora de implementarlos me di cuenta de que mi servidor Apache no disponía de soporte para Python y preferí traducirlos a PHP, que si esta soportado en mi servidor Apache, antes que buscar un servidor que permitiera la integración de código Python o un módulo de ejecución de scripts Python para mi servidor Apache.

8.1.3. Planificación temporal

La planificación temporal fue uno de los mayores quebraderos de cabeza del proyecto. Se realizaron tres planificaciones temporales diferentes a lo largo del proyecto. Esto fue debido a que surgieron varios problemas que retrasaron la ejecución del proyecto.

El primer problema fue por motivos personales y laborales, estuve dos meses con muy poco tiempo para la realización de las tareas y prácticamente no avance nada. Otro de los problemas que ha retrasado el proyecto y obligado a la reestructuración de la planificación temporal ha sido mi falta de constancia, y para esto no hay ninguna excusa, costaba mucho ponerse a hacer las tareas. Los días iban pasando y el avance del proyecto era muy lento, por lo que hubo que hacer una segunda reestructuración de la planificación.

8.1.4. Herramientas

- **Gráficos.** Se probaron varias herramientas para el desarrollo de los gráficos interactivos de la página web. Entre ellos estaban Highcharts [17] y Charts.js [18]. Finalmente la elección fue Plot.ly porque es una herramienta más conocida y ofrece muchas posibilidades, tipos de gráficos y lenguajes con los que trabajar.

Es lógico pensar que es más probable que trabaje de nuevo con Plot.ly en el futuro debido a su popularidad. No obstante nunca está de más probar y conocer otras herramientas similares.

- **Diagramas.** En un principio se empezó a trabajar con la misma herramienta con la que se había trabajado durante el grado: Visual Paradigm [19]. Este programa, entre otras cosas, permite la elaboración de diagramas UML con bastante calidad.

  El problema fue que cuando surgió la necesidad de compartir estos diagramas con el tutor para su revisión nos dimos cuenta de que no era lo más rápido ni lo más práctico. No hay compatibilidad entre todas las versiones de Visual Paradigm y por lo tanto si tenemos versiones diferentes instaladas no podemos visualizar o modificar los archivos enviados.

  La solución, propuesta por el tutor, fue usar Cacoo, que se trata de una herramienta online en la que puedes invitar a otras personas a visualizar un diagrama. Y como ventaja no es necesario instalar ningún software por lo que los diagramas pueden visualizarse en cualquier dispositivo en cualquier momento.

8.2. Resultado

El resultado final ha sido satisfactorio. Sí que es cierto que es un proyecto que se ha alargado excesivamente en el tiempo y que se podía haber hecho muchas más cosas, pero se ha hecho un gran trabajo, muy costoso, que ha llevado mucho tiempo de realizar y el resultado es algo innovador, útil y sofisticado.

El resultado final podrá verse y utilizarse desde cualquier dispositivo, a través de la página web de Euskal Crawler. También estará el código de todos los scripts alojado en github y la documentación alojada en ADDI.

8.3. Reflexión personal

Este proyecto empezó hace casi un año. Cuando me puse a hacerlo nunca imaginé que me llevaría tanto tiempo realizarlo. Mi idea era acabarlo en tres o cuatro meses. Pero han pasado muchas cosas desde el inicio que han retrasado su ejecución, como ya se ha explicado anteriormente.

El principal problema de un TFG es que nadie va a exigirte nada. Eres tú el que tiene que exigirse a sí mismo, y eso es bastante duro. Pero está bien que sea así y es necesario porque te prepara para el mundo laboral real. Es uno mismo el que se exige que objetivos cumplir, cuanto tiempo dedicar y el nivel de calidad y de implicación con una tarea.

En el transcurso de este proyecto me he dado cuenta de muchas cosas y me ha permitido conocerme mejor a mí mismo. Me falta constancia e implicación en la ejecución del proyecto. También he sido consciente de mi inexperiencia para llevar a cabo un proyecto real desde cero hasta el final. Y mi principal punto débil es la documentación ya que me lleva un tiempo excesivo y no consigo realizarla tan completa y precisa como otros compañeros.

A pesar de todas las dificultades y carencias me siento muy orgulloso de todo el trabajo que he realizado, que no ha sido nada fácil. Y cuando ves el resultado final sientes que has hecho algo importante y aunque sabes que aún te falta mucha experiencia y tiempo para adquirirla miras adelante con ilusión y con ganas de aprender y de implicarte más.

9. Trabajo futuro

Esta aplicación es una versión bastante básica del proyecto. Se puede mejorar muchos aspectos y añadir nuevas funcionalidades. Aunque el trabajo que se proyectó se ha realizado casi en su totalidad hay muchos campos en los que se puede seguir mejorando y otros campos a los que se les pueden añadir nuevas opciones para ampliar la calidad del proyecto y también para ofrecer más información.

Ahora pasaremos a ver estas mejoras que se ha pensado que se podrían llevar a cabo si se continuara con el desarrollo de este proyecto:

9.1. Diseño de la página web

El diseño de la página web está hecho con Bootstrap. Hay muchos aspectos descuidados del diseño ya que se ha priorizado la funcionalidad del proyecto. Los aspectos que se podrían mejorar en cuanto al diseño son los siguientes:

- **Colores y formas.** Se podrían mejorar los colores de la página web así como las formas y las secciones para darle un aspecto más atractivo, profesional y serio.

- **Reducción de la ventana del navegador.** Cuando se reduce la ventana del navegador la barra de navegación superior de la página web se descuadra y los textos se salen de la misma. Convendría reducir el tamaño de la letra o generar listas desplegables cuando se reduce la ventana del navegador. O en su defecto estudiar otras alternativas.

- **Gráficos.** El estilo de la pantalla de gráficos es muy pobre. Convendría mejorar el checkbox lateral y la lista de la derecha para darle un estilo más fresco y futurista. También sería conveniente añadir algún efecto al gráfico para hacerlo visualmente más atractivo.

- **Formularios.** Los formularios son formularios predefinidos de Bootstrap que han sido modificados según las necesidades pero no se le ha añadido ningún estilo adicional. Convendría retocarlos para darles un toque personal y que todos siguieran el mismo tipo de estilo.

- **Zona de usuario.** La zona de usuario también es muy pobre en cuanto a estilo y convendría hacerlo más atractivo y diferente al resto de la página web para que el usuario se sienta cómodo cuando inicie sesión y sea fácil diferenciar cuando ha cerrado la sesión.

9.2. Contenido de la página web

Actualmente el contenido de la página web es muy escaso. Se ofrece muy poca información sobre lo que hace o como funciona Euskal Crawler. Se podrían crear más secciones con funcionalidades nuevas, como por ejemplo un blog donde se iría informando de las novedades en cuanto a seguridad en los CMS.

Otra sección que podría añadirse a la página web es un foro donde empresas y desarrolladores podrían discutir sobre la seguridad de sus sistemas y ponerse en contacto para mejorar la seguridad de sus servidores.

9.3. Zona de usuario

Actualmente la zona de usuario solo permite el escaneo de una página web y nada más. La idea sería añadir más secciones dentro de la zona de usuario para que se puedan realizar más tareas. Por ejemplo tener un registro de las páginas web que ha escaneado. Otra sección sería las estadísticas de sus propios sitios web para que pueda ver de un vistazo cuáles son sus sistemas más vulnerables, en el caso de que tuviera varios.

También sería conveniente una zona de configuración donde el usuario pudiera modificar sus datos personales, ya que es casi impresionable en cualquier página

web. Además podrían pensarse y diseñarse otras muchas funciones para darle vida a la zona de usuario.

9.4. Más idiomas

Solo hay tres idiomas disponibles para la página web: inglés, español y euskera. Lo cual está bastante bien considerando que otras páginas web solo están disponibles en un único idioma. Pero la idea es mejorar y estaría muy bien añadir otros idiomas para mejorar la accesibilidad del sitio. Pero para esto habría que contratar a un traductor experto.

9.5. Gráficos

En cuanto a los gráficos se pueden incluir más variedades de gráfico, por ejemplo gráficos en 3D o gráficos de puntos para que el usuario tenga más opciones a la hora de visualizar la información y le sea más cómodo consultarla. Simplemente sería añadir nuevas pestañas a la sección de estadísticas o unificarlos todos en una sola página y permitirle al usuario seleccionar el tipo de gráfico con un listado desplegable.

También podrían añadirse nuevas opciones al gráfico. Actualmente solo permite filtrar los datos de la comunidad autónoma del País Vasco. Se podría modificar para que permitiera filtrar por país y después por provincia o región. También podría agregarse información adicional sobre el listado de CMS consultado en cada gráfico.

9.6. Obtención de datos

La obtención de los datos de las empresas se hace a través del script de *web scraping* escrito en Python que se encarga de recoger la información de un listado de empresas de civex para introducirlo en la base de datos.

La idea sería buscar nuevas fuentes de datos para aumentar la base de datos o nuevas formas de obtención. Quizá con más tiempo y un proyecto futuro si se

pudiera diseñar un *web crawler* que tratara de determinar la procedencia de una página web y aumentar las estadísticas de forma nacional o global.

9.7. Mejoras en la búsqueda de vulnerabilidades

Actualmente el script de vulnerabilidades solo examina sitios web que estén construidos con CMS. Y solo detecta vulnerabilidades asociadas a la falta de actualización de los mismos. La idea sería diseñar un nuevo script que permitiera también la comprobación de otro tipo de vulnerabilidades para hacerlo más completo.

Hay muchos tipos de vulnerabilidades, amenazas y riesgos en cuanto a una página web se refiere. Es importante actualizar la versión del código en la que están construidos, ya que versiones más antiguas suelen tener un mayor número de agujeros de seguridad. También es importante cómo esté diseñado y construido el sitio web para evitar ataques como SQL injection, por ejemplo. Una gran parte de los propietarios de sitios web desconocen esto y ampliar el script para detectar más vulnerabilidades serviría para ayudarles e informarles.

Glosario

- **API.** Aplication Programming Interface (Interfaz de Programación de Aplicaciones). “Es una llave de acceso a funciones que podemos utilizar de un servicio web provisto por un tercero, dentro de nuestra propia aplicación web, de manera segura y confiable.” [20]

- **CMS.** Content Management System (Sistema de gestión de contenidos). “Es un software que permite la creación y administración de los contenidos de una página Web, principalmente, de forma automática.” [21]

- **EDT.** Estructura de Desglose del Trabajo. “Es una descomposición jerárquica, orientada al producto entregable del trabajo que será ejecutado por el equipo del proyecto, para lograr los objetivos del proyecto y crear los productos entregables requeridos.” [22]

- **IDE.** Un IDE (Integrated Development Environment) o ambiente de desarrollo integrado es una aplicación informática compuesta por un conjunto de herramientas que son generalmente un editor de código, un compilador, un depurador y un constructor de interfaz gráfica que facilitan la labor del desarrollador. Existen IDE para casi cualquier lenguaje de programación. [23] [24]

- **SQL.** Structured Query Language (Lenguaje de consulta estructurado). “Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.” [25]

- **UML.** Unified Modeling Language (Lenguaje Unificado de Modelado). “Es un estándar para la representación de procesos o esquemas de software (programas informáticos).” [26]

- **URL.** Uniform Resource Locator (Localizador de Recursos Uniforme). “Se trata de la secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.” [27]

Bibliografía

[1] BeauHD. "Outdated and vulnerable wordpress, drupal versions contributed to panama papers breach". 06 de Abril de 2016. Enlace: https://news.slashdot.org/story/16/04/07/008230/outdated-and-vulnerable-wordpress-drupal-versions-contributed-to-panama-papers-breach.

[2] Dimitrios Kouzis-Loukas. "Learning Scrapy" (2016), Ed. Packt.

[3] Jetbrains. "Jetbrains Pycharm Preview (EAP)". Enlace: https://confluence.jetbrains.com/pages/viewpage.action?pageId=23004355.

[4] Universidad de Antioquia. "¿Qué es cacoo?". Enlace: http://aprendeenlinea.udea.edu.co/boa/contenidos.php/c7149543e9ad3ef665662a0aee87caf9/1035/estilo/aHR0cDovL2FwcmVuZGVlbmxpbmVhLnVkZWEuZWR1LmNvL2VzdGlsb3MvYXp1bF9jb3Jwb3JhdGl2by5jc
/1/contenido/1Que_es_Cacoo/cacoo.html.

[5] Suzanne Rubinstein. "Analiza y visualiza datos en equipo con Plot.ly". Enlace: http://tecducacion.com/sistema-operativo/navegadores/analiza-y-visualiza-datos-en-equipo-con-plot-ly/.

[6] Luis Castellanos. "Plot.ly". 07 de Agosto de 2015. Enlace: https://dtyoc.com/2015/08/07/plot-ly/.

[7] Juan David Quiñónez. "ShareLaTeX, editor de LaTeX colaborativo, online y con decenas de plantillas". 08 de Septiembre de 2013. Enlace: http://wwwhatsnew.com/2013/09/08/sharelatex-editor-latex-online-colaborativo/.

[8] Guillermo Julián. "ShareLaTeX, edición online y colaborativa de documentos LaTeX". 21 de Diciembre de 2012. Enlace: http://www.genbeta.com/web/sharelatex-edicion-online-y-colaborativa-de-documentos-latex.

[9] Luciano Castillo. "¿Qué es GitHub?". Enlace: http://conociendogithub.readthedocs.io/en/latest/data/introduccion/.

[10] Johanny Solis. "¿Qué es bootstrap y cómo funciona en el diseño web?". 26 de Septiembre de 2014. Enlace: http://www.arweb.com/chucherias/editorial/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web.htm.

[11] Esepe Studio. "¿Qué es MySQL?". 18 de Agosto de 2005. Enlace: http://www.esepestudio.com/noticias/que-es-mysql.

[12] BOE. "XVI Convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública". 04 de Abril de 2009. Enlace: https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf.

[13] Jesper Kückelhahn. "WIG - WebApp Information Gatherer". Enlace: https://github.com/jekyc/wig.

[14] "CMS Explorer". Enlace: https://github.com/delta24/updated_cms_explorer.

[15] "New Lists for CMS Explorer". 13 de Diciembre de 2015. Enlace: http://viyatb.in/2015/New-Lists-for-CMS-Explorer/.

[16] Acunetix. Enlace: http://www.acunetix.com/.

[17] HighCharts. Enlace: http://www.highcharts.com/.

[18] Chart.js. Enlace: http://www.chartjs.org/.

[19] Visual Paradigm. Enlace: https://www.visual-paradigm.com/.

[20] "¿Qué es y para qué sirve una API?". Enlace: http://www.internetya.co/que-es-y-para-que-sirve-una-api/.

[21] Alicia Cañellas Mayor. "CMS, LMS y LCMS. Definición y diferencias". Enlace: http://www.centrocp.com/cms-lms-y-lcms-definicion-y-diferencias/.

[22] Kemuel Micael Francisco Mateo. "La EDT como Herramienta de Gestión del Alcance de Proyectos". 18 de Diciembre de 2012. Enlace: http://www.eoi.es/blogs/mintecon/2012/12/18/la-edt-como-herramienta-de-gestion-del-alcance-de-proyectos/.

[23] EcuRed. "IDE de Programación". Enlace: http://www.ecured.cu/IDE_de_Programaci%C3%B3n.

[24] Daniel Maldonado. "¿Qué son los IDE de Programación?". 03 de Septiembre de 2007. Enlace: https://elcodigok.blogspot.com.es/2007/09/que-son-los-ide-de-programacin.html.

[25] Micho García y Jorge Arévalo. "Conceptos básicos de SQL". 15 de Octubre de 2013. Enlace: http://geotalleres.readthedocs.io/es/latest/conceptos-sql/conceptos_sql.html.

[26] César Krall. "¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.". Enlace: http://aprenderaprogramar.es/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

[27] "Definición de URL". Enlace: http://definicion.de/url/.