eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

EUSKALCRAWLER, UN DETECTOR DE VULNERABILIDADES WEB

Alumno/Alumna: Gallego González, Iñigo

Director/Directora (1): Pereira Varela, Juanan

Curso: 2017-1018

Fecha: 30-01-2018

0. Resumen

Durante este proyecto, se ha desarrollado un *detector de vulnerabilidades* en Django combinando la funcionalidad de dos herramientas (Whatweb y Uniscan), que permiten a cualquier usuario que tenga una página web conocer los datos de su página y detectar sus vulnerabilidades.

La primera herramienta es *WhatWeb*, una herramienta que reconoce tecnologías Web, incluyendo los sistemas de gestión de contenidos (CMS), plataformas de blogs, bibliotecas JavaScript, servidores web, etc. Actualmente contiene más de 900 *plugins* que permiten reconocer tecnologías diferentes. Además identifica versiones, direcciones de correo electrónico, números de cuenta, errores SQL, etc. Para obtener dicha información, realiza una serie de consultas a las páginas web de forma pasiva, es decir, realizando consultas no muy seguidas en un determinado espacio de tiempo para que la web no sospeche y le bloquee el acceso.

La segunda herramienta es *Uniscan*, un escáner de vulnerabilidades Web que está licenciado bajo GNU GENERAL PUBLIC LICENSE 3.0 (GPL 3). Esta herramienta permite encontrar, además, emails de contacto para poder ponerse en contacto con el desarrollador de la página en caso necesario. Además, también realiza consultas de forma pasiva.

Además, para asegurar que cualquier usuario no pueda consultar la información de cualquier pagina, sino que sólo la suya, se ha implementado un sistema de seguridad basado en verificar si un usuario es el dueño de la página web que quiere escanear.

Índice

0 Resumen **3**

1 Definiciones y vocabulario técnico **9**

2 Introducción **11**
2.1 Origen del proyecto 13
2.2 Descripción y situación del proyecto 13
2.3 Motivaciones para la elección del proyecto 15

3 Planteamiento inicial **17**
3.1 Objetivos 17
3.2 Herramientas utilizadas 19
3.3 Arquitectura 21
3.4 Alcance 23
3.4.1 Aprendizaje 24
3.4.2 Organización 26
3.4.3 Captura de requisitos 28
3.4.4 Análisis y diseño 29
3.4.5 Implementación y desarrollo 30
3.4.6 Pruebas 33
3.4.7 Documentación 35
3.4.8 Resumen de la planificación realizada 38
3.5 Planificación temporal 40
3.6 Evaluación económica 42
3.6.1 Mano de obra 42
3.6.2 Gasto de *software* 43
3.6.3 Gasto de *hardware* 43
3.6.4 Gastos totales 44
3.6.5 Posibilidades de negocio 44
3.7 Riesgos 44
3.7.1 Enfermedad o lesión 46
3.7.2 Problemas con el equipo informático 47
3.7.3 Problemas con las herramientas de desarrollo 47
3.7.4 Discontinuidad en el proyecto 48
3.7.5 Falta de conocimientos para poder comenzar el desarrollo 49

4 Antecedentes **51**
4.1 EuskalCrawler 51
4.2 Situación actual del proyecto 53

5 Captura de requisitos **55**
5.1 Jerarquía de actores 55
5.2 Casos de uso 55
5.2.1 Empresario 55
5.2.2 Usuario registrado 56
5.3 Modelo de dominio 58

6 Análisis y diseño **61**
6.1 Análisis de elementos 61
6.2 Diagrama de clases 62
6.3 Diagramas de secuencia 63

7 Elección de lenguajes y tecnologías **67**
7.1 Elección del SDK 67
7.2 Elección del lenguaje de programación 67
7.3 Elección del entorno de desarrollo 68
7.4 Elección de los escáneres web 68

8 Desarrollo **69**
8.1 Necesidades del cliente 69
8.1.1 Mejora del escaneo de las páginas web 69
8.1.2 Mejora en la seguridad de la aplicación 70
8.2 Desarrollo de la base de datos 71
8.3 Desarrollo de la aplicación 72

9 Pruebas **75**
9.1 Pruebas de la verificación de las páginas web 75
9.2 Pruebas de la normalización de los datos 76
9.3 Pruebas del registro 77
9.4 Pruebas de la identificación 78
9.5 Pruebas del escaneo de las páginas webs 78

10 Conclusiones **79**
10.1 Análisis entre planificación estimada y real 79
10.1.1 Objetivos 79
10.1.2 Herramientas utilizadas 79
10.1.3 Alcance 80
10.1.4 Evaluación económica 83
10.2 Lineas futuras 86
10.3 Reflexión personal 86

11 Direcciones Webs utilizadas **89**

A Anexo 1: Manual de instalación 91
A.1 Instalación de Django 91
A.2 Instalación de python y sus librerías 91
A.3 Instalación del proyecto 91
A.4 Instalación de Uniscan 92
A.5 Instalación de Whatweb 92
A.6 Instalación de la base de datos 93
A.7 Preparación de la confirmación de correos 93
A.8 Lanzamiento del servidor 93

Índice de figuras

1 Explicación de la verificación. 14
2 Esquema de la arquitectura de un detector de vulnerabilidades. 22
3 Diagrama EDT por bloques de nivel 1. 24
4 Diagrama EDT del bloque de aprendizaje. 25
5 Diagrama EDT del bloque de organización. 27
6 Diagrama EDT del bloque de captura de requisitos. 28
7 Diagrama EDT del bloque de análisis y diseño. 29
8 Diagrama EDT del bloque de implementación y desarrollo. . . 31
9 Diagrama EDT del bloque de pruebas. 33
10 Diagrama EDT del bloque de documentación. 35
11 Diagrama Gantt 41
12 Funcionalidades antiguas y nuevas 52
13 Jerarquía de actores 56
14 Diagrama de casos de uso inicial de Empresario 57
15 Diagrama de casos de uso inicial de Usuario Estándar 57
16 Diagrama de modelo de dominio 58
17 Diagrama de clases 62
18 Diagrama de secuencia del escaneo 64
19 Diagrama EDT final del bloque de aprendizaje. 80

Índice de tablas

1 Dependencias y duración de las tareas (1). 38
2 Dependencias y duración de las tareas (2). 39
3 Costes totales del proyecto 44
4 Dependencias y duración de las tareas finales(1). 83
5 Dependencias y duración de las tareas finales (2). 84
6 Costes finales del proyecto 85

1. Definiciones y vocabulario técnico

Antes de comenzar con la introducción y la explicación del desarrollo del proyecto, es importante conocer algunos de los términos que se utilizan durante este documento. Para poder entender lo que se explica, es necesario conocer el significado de los siguientes tecnicismos:

- **Vulnerabilidad**: Es todo aquello que hace que una página sea menos segura o que esté desactualizada, por eso, en este proyecto, vulnerabilidad se entenderá como fallo en la seguridad o utilización de versiones obsoletas.
- **Detector de vulnerabilidades o escáner web**: Es una herramienta utilizada para poder localizar las versiones de la página así como errores en la base de datos de la misma. Para ello se basa en las *fingerprints*.
- ***Fingerprints***: Son las huellas que se pueden seguir de las páginas web para poder detectar qué tipo de *software* está utilizando y, por lo tanto, sus posibles vulnerabilidades.
- **Base de datos**: Es una colección de información organizada que permite a un sistema informático realizar consultas rápidamente para extraer los datos que realmente le interesan.
- ***Python***:Del inglés pitón (de ahí que su logo sea dos serpientes entrelazadas), es un lenguaje de programación que se creó para poder crear códigos de programación legibles, de ahí que sea uno de los lenguajes preferidos por los profesores de secundaria para introducir a los alumnos en el mundo de la programación. Es un lenguaje que soporta programación orientada a objetos, programación imperativa y, en menor medida, programación funcional. Su página oficial es: `https://www.python.org/`
- ***Framework* o marco de trabajo**: Es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.
- **Django**:Es un *framework*, escrito en Python, que utiliza un diseño Modelo-Vista-Controlador, y que permite tener el contenido de la pagina de una forma muy ordenada proporcionando servicios como sesiones, creación de cookies, autenticación de usuarios, soporte para bases de datos y una página de administración para gestionar contenido, entre otros. Su página oficial es: `https://www.djangoproject.com/`

- **HTTP o *Hypertext Transfer Protocol***: Es un protocolo de comunicación que permite las transferencias de información entre servicios y clientes que utilizan páginas web.
- **CMS (*Content Management System*)**:Es un programa desarrollado para que cualquier usuario pueda administrar y gestionar contenidos de una web con facilidad y sin conocimientos de programación Web.Algunos ejemplos son: WordPress, PrestaShop, Joomla, Magento...

2. Introducción

La seguridad en informática ha sido un tema preocupante tanto para los empresarios como para sus clientes a la hora de implantar un sistema web para gestionar las diferentes operaciones de una empresa. Por ello, muchas entidades o sujetos independientes han visto en este campo un método de negocio, ya sea para proteger o para atacar los diferentes sistemas informáticos. Por un lado, las empresas que están orientadas a campos que distan mucho de la informática deben gastar un gran presupuesto en seguridad informática para proteger sus sistemas, para ello, contratan un servicio y/o personal que ofrezca garantías de que no serán vulnerables a ataques para obtener información de su empresa o para robarles. Por otro lado, numerosos *hackers* son contratados por empresas para enseñar a sellar las brechas de seguridad de sus sistemas informáticos, ya que, al conseguir encontrar vulnerabilidades en los sistemas cuyos propietarios no creían existentes, también pueden ayudar a solventarlos.

El primer paso para que nuestro sistema informático sea más seguro es conocer qué tecnologías está utilizando y saber si hay actualizaciones posibles. Mantener el sistema actualizado garantiza que esté dotado de posibles correcciones de *bugs*, nuevas funcionalidades o, lo más importante, de corrección en fallos de seguridad.

También es importante conocer si, a la hora de implementar el sistema, se ha cometido algún error de programación o configuración que pueda suponer una brecha de seguridad en el mismo y, desgraciadamente, una de las formas para conocer estos casos, es cuando el sistema es atacado. Por ello, se crearon los escáneres de seguridad.

Existen numerosos escáneres web que permiten identificar vulnerabilidades a diferentes niveles, pero se han detectado casos en los que no son precisos al 100 % a la hora de detectar la causa de que un sitio web sea vulnerable. Esto se debe a que es muy difícil saber determinar que versiones utiliza una página web a la hora de hacer consultas para sacar información de ellas.

Esto provoca que usuarios externos no sean capaces de obtener información relevante que les pueda ayudar para un posterior ataque más agresivo que pueda poner en peligro todo el sistema. En consecuencia, y dado que los escáneres web pretenden sonsacar información sin ser detectados, muchos no son capaces de conseguir la suficiente información u obtienen información equivocada. Por ello, diferentes escáneres pueden recibir información diferente y muchos son mejores adquiriendo un tipo de información u otra.

El objetivo de este Trabajo de Fin de Grado (TFG) es implementar un escáner web que combine los resultados de varios escáneres, ganando así una mayor precisión y cantidad de información a la hora de informar a los usuarios.

2.1. Origen del proyecto

La idea del proyecto original fue de Juanan Pereira, docente en la Universidad del País Vasco (UPV/EHU).

Juanan quería desarrollar un detector de vulnerabilidades para las páginas web que fuera preciso y que permitiera notificar a los gerentes de las empresas del País Vasco que su web no era del todo segura, y alegando los motivos. Para ello, vio en los escáneres web una oportunidad para poder detectar este tipo de incidencias y lo propuso como TFG a algunos de sus alumnos.

Cuando el profesor Pereira le propuso al alumno David López Angulo la realización de este proyecto, David aceptó realizar algo que ayudara a todas las empresas del País Vasco. Desarrolló el sistema en HTML, CSS, JavaScript y Python. Planteando el proyecto de manera que un escáner observara las empresas y mostrara en una página web la información que había obtenido. Para escanear las empresas utilizó Wig[1], es decir, una sola tecnología, lo que provocó una serie de problemas que se comentarán durante este documento, y que fueron una de las razones por las que se decidió mantener el proyecto para mejorar lo que David realizó. Además, no se implementó un panel de control y tampoco un sistema de verificación de la propiedad de una web.

2.2. Descripción y situación del proyecto

Este TFG es la continuación y mejora del proyecto "Euskalcrawler", realizado por David López Angulo [Angulo, 2016]durante el año 2016. Como ya se ha comentado anteriormente, el proyecto consiste en el desarrollo de un detector de vulnerabilidades en páginas web con el que los gerentes de las empresas puedan detectarlas y lleguen a mejorar su sistemas de seguridad. Para ello el gerente se dará de alta en el servicio y pedirá información sobre su página web. El sistema diariamente hará un escaneo de algunas de las empresas registradas para continuar al día siguiente con las restantes. A todas las empresas de las que tenga información se les mostrará cuando el usuario introduzca el nombre de su empresa. Para mejorar la seguridad y evitar que cualquiera escanee cualquier empresa, el sistema pedirá al gerente que cree una determinada URL dentro de su página web con un determinado contenido que consistirá en un número de 16 dígitos pseudoaleatorios para confirmar que él es el dueño. En la figura 1 se puede observar ésta funcionalidad de forma más clara.

[1]Link de descarga de Wig: `https://github.com/jekyc/wig`.

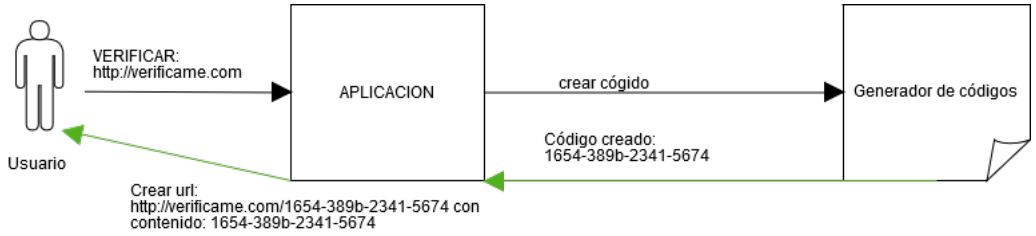

Figura 1: Explicación de la verificación.

El sistema realizado para este proyecto es un escáner de vulnerabilidades llamado "Euskalcrawler". Su base ha sido desarrollada utilizando las librerías de "Django"[2]. Como casi todos escáneres web, "Euskalcrawler" muestra la información de lo que encuentra de una forma clara y ordenada.

Hasta el momento se utilizan dos tecnologías de detección de versiones y vulnerabilidades (WhatWeb y Uniscan), pero el proyecto está preparado para añadir más de una forma más o menos sencilla, sin tener que modificar en gran medida el código existente .

Además de la página web donde se realizarán todas los operaciones orientadas a los usuarios y se mostrará la información deseada, Django también proporciona un panel de administración donde poder administrar el sistema de una forma muy sencilla.

[2]Link de Django: `https://www.djangoproject.com/`.

2.3. Motivaciones para la elección del proyecto

Hay cinco motivos principales que me han llevado a la elección de este proyecto.

2.3.0.1. Aprendizaje

Cuando Juanan me ofreció la posibilidad de realizar un proyecto que iba a ser realizado en un lenguaje como Python, que en la carrera no lo había tocado de una forma tan profunda, y con un *framework* que estructura la información de la web como Django, acepté sin pensármelo dos veces ya que, para mí, todo lo que sea aprender más supone una mejora para mi futuro laboral y poder ser más competente.

Curiosidad

Cuando Juanan expuso en una presentación temas de los proyectos que ofrecía, el tema de la seguridad en informática llamó mi atención. Me empecé a plantear las preguntas: ¿Cómo se puede atacar a una página web? ¿Cómo protegerla? Juanan me dio las pautas para poder buscar escáneres web y averiguar cómo funcionaban, además, me enseñó a buscar yo mismo el mismo tipo de información de las páginas web. Como me llamó tanto la atención la cantidad de información que se podía obtener simplemente realizando unas pocas consultas en una página web, decidí que desarrollaría un escáner web para mi TFG.

Me pareció un reto interesante

Cuando me dirigí a Juanan para preguntarle sobre los proyectos disponibles, él me propuso este y, como las asignaturas orientadas a la seguridad que se imparten en el grado me parecieron muy interesantes y eché de menos profundizar un poco más en el tema, acepté sin pensármelo dos veces. El hecho de coger un proyecto que alguien había empezado, comprenderlo y mejorarlo, y además de eso, empezar sin saber casi nada al respecto, me pareció un gran reto a superar. Esto me motivó a escoger el proyecto, aunque me decepcioné un poco al tener que desechar todo lo referente al código que se había hecho hasta el momento.

El problema a solventar me afecta

Soy una persona que me gustaría dedicarme al desarrollo de aplicaciones web. El hecho de que pudiera crear una funcionalidad que me permitiera escanear mis futuras páginas web me pareció muy atractivo y muy interesante, ya que no sólo me ayudaría a mí, sino que podría ser útil a otros desarrolladores

Va a ser utilizado por empresas reales

Todo proyecto tiene el fin de llegar a ser utilizado. El hecho de que la versión que existía hasta ese momento no iba a ser usada, me impulsó a crear un versión mejorada que permitiera la introducción de mejoras de forma sencilla para que el escáner que realizo en el TFG nunca se quedara obsoleto o desactualizado.

3. Planteamiento inicial

En esta sección se explican los objetivos, las herramientas utilizadas, el alcance, la planificación temporal y la evaluación económica y de riesgos que se han realizado para este proyecto. También se presenta la arquitectura que se utiliza para una mejor comprensión del funcionamiento de un escáner web.

3.1. Objetivos

Los objetivos del proyecto se han dividido en cuatro secciones. Así, quedan bien agrupados lor objetivos en las diferentes funcionalidades desarrolladas durante el proyecto. Además, al ser la continuación de un proyecto del que se desechó lo realizado, en este se han recogido algunos de los objetivos de su antecesor.

Escáner web

En esta sección se muestran los objetivos principales que ha de cumplir el escáner. Su completo desarrollo se ha basado en los siguientes objetivos:

- **Funcionalidades completas y sin fallos**: Al ser un escáner que requiere de tecnologías externas, esas tecnologías deben ser adaptadas a las necesidades del proyecto y sin que provoquen ningún tipo de error. Han de realizarse todos los cambios necesarios para que los usuarios puedan solicitar el servicio que se ofrece, recibiendo siempre la información necesaria. No se permite que generen ningún tipo de error o *warning* en los *logs*, ya que esto puede afectar al desarrollo de este proyecto, pero también puede llegar a repercutir en otros.

- **Información clara y comprensible**: Todos los datos que el escáner consiga recopilar deberá ir acompañado de una etiqueta que identifique a qué hacen referencia los diferentes datos. Además, al utilizar varios escáneres web, los datos deberán ser normalizados y filtrados, mediante un proceso de votación y un método de desempate en caso de que varios escáneres se contradigan a la hora de encontrar información (por ejemplo, que uno diga que está usando un CMS como Joomla y dos digan que está usando un CMS como WordPress).

- **Escaneo periódico**: Cada cierto tiempo se deberá hacer un escaneo de todas las empresas que se han registrado en el sistema y de las que ya

estaban registradas, hay que establecer un período de tiempo adecuado para intentar agilizar el servicio.

- **Seguridad y privacidad**: Este apartado hace referencia a la seguridad informática, y el sistema deberá asegurar que sólo los dueños de las empresas puedan consultar la información obtenida por el escáner, de lo contrario, se estará poniendo en peligro la seguridad y la protección de datos de los usuarios.

Página web

En esta sección se muestran los objetivos principales que ha de cumplir la página web. Su completo desarrollo se ha basado en los siguientes objetivos:

- **Diseño claro y atractivo**: Los elementos de la página deberán estar ordenados y para realizar las diferentes operaciones que se ofrecen, el usuario no deberá dar más pasos que los estrictamente necesarios.
- **Funcionalidades básicas**: Dado que la página es un detector de vulnerabilidades, deberá ofrecer ese servicio, y dado que se pretende que los usuarios se sientan seguros, se deberá ofrecer un servicio de identificación y registro para poder guardar información acerca de las páginas web y sus dueños.
- **Verificación de los correos electrónicos y las páginas web**: Dado que el servicio registrará a los usuarios mediante su email, el sistema deberá asegurarse de que dicho correo es válido y existe para posteriores notificaciones a los usuarios. Así, también deberá asegurarse de que las webs registradas en el sistema estarán verificadas por los usuarios. de esta forma, únicamente el usuario que verificó la página web podrá consultar la información obtenida.

Generales y de documentación En esta sección se muestran los objetivos principales que ha de cumplir la documentación y el proyecto de forma general. Su completo desarrollo se ha basado en los siguientes objetivos:

- **Documentación bien definida**: Ya que este proyecto va a ser utilizado por empresas reales, la documentación ha de estar bien definida y en apartados específicos. De esta forma, si surge alguna duda, problema o error, siempre se podrá consultar el apartado correspondiente de este documento para poder solventarlos lo más rápido posible. También se

han añadido comentarios al código para que se pueda identificar fácilmente la función que ejerce cada una de las funcionalidades implementadas. Además, esto hará más fácil las modificaciones y actualizaciones que se quieran realizar en el futuro.

- **Preparado para modificaciones y nuevas funcionalidades**: El detector de vulnerabilidades y páginas web deben ser fáciles de modificar y se tiene que poder añadir nuevas funcionalidades sin problemas. Además, tiene que estar siempre preparado para las nuevas actualizaciones que se creen en las librerías de Django, Python, así como también en las librerías de las tecnologías que se fueran a utilizar para escanear las webs.

3.2. Herramientas utilizadas

Aquí se detallan las diferentes herramientas que se utilizarán para el desarrollo del proyecto. También se explica, de manera resumida, en qué partes del proyecto se plantea usar cada una de ellas.

- **Bitbucket**: Es un sistema de control de versiones que consta de repositorios privados gratuitos. Desde este sistema se puede clonar lo que exista en un repositorio para poder modificarlo en local y después subir el nuevo código a ese repositorio. Es una herramienta muy útil para proyectos en los que participa mucha gente, ya que en todo momento todos los desarrolladores pueden acceder a un código actualizado con una simple instrucción. El código del proyecto se guardará en un repositorio privado de este sistema. Su página oficial es: `https://bitbucket.org/product`.

- **Git**: Es un sistema de gestión de versiones de código abierto. Está integrado en sistemas como GitHub, Bitbucket o GitLab y se puede usar en casi todo tipo de plataformas. Es el programa que usará para compartir las diferentes versiones del código con otros potenciales desarrolladores. Su página oficial es: `https://git-scm.com/`.

- **Django**: Es un *framework* para aplicaciones web gratuito y open source, escrito en Python que ayuda a desarrollar sitios web respetando el patrón de diseño conocido como Modelo–vista–controlador. Todo lo referente a *software* del proyecto se realizará con ésta herramienta.

- **HTML**: También conocido como *HyperText Markup Language*, es un lenguaje de programación basado en etiquetas. Este lenguaje es ejecu-

tado por el navegador, que es el encargado de interpretar el código y mostrar los elementos de las páginas web. La página web del escáner utilizará este lenguaje para mostrar los componentes en la web.

- **CSS**:Cascading Style Sheets, u Hojas de Estilo en Cascada es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación. El diseño visual de la página web será implementado con este lenguaje.

- **Python**: Es un lenguaje de programación interpretado que es principalmente usado en páginas web dinámicas del lado del cliente. Gran parte de la funcionalidad de la página web, así como el trato de las librerías de Whatweb y Uniscan y los resultados obtenidos se realizará con este lenguaje.

- **Uniscan**:Uniscan es un escáner de vulnerabilidades Web, dirigido a la seguridad informática, cuyo objetivo es la búsqueda de vulnerabilidades en los sistemas web. Está desarrollado en Perl y tiene un fácil manejo de expresiones regulares. Una de las herramientas utilizadas para escanear las webs es ésta. La página donde descargarlo es: `https://github.com/poerschke/Uniscan.git`.

- **Perl**:es un lenguaje de programación diseñado por Larry Wall en 1987. Perl toma características del lenguaje C, así como otros lenguajes interpretados de muchos otros lenguajes de programación. Este lenguaje es en el que se ha desarrollado Uniscan.

- **Whatweb**:WhatWeb es una herramienta reconoce tecnologías Web, incluyendo los sistemas de gestión de contenidos (CMS), plataformas de blogs, bibliotecas JavaScript, servidores web, etc. Actualmente contiene más de 900 *plugins* los cuales permiten reconocer tecnologías diferentes. Además identifica versiones, direcciones de correo electrónico, números de cuenta, errores SQL, etc. Ésta es otra tecnología utilizada para el escaneo de páginas web. Su página oficial es: `https://whatweb.net/`.

- **MySQL**: Es el sistema gestor y de administración de bases de datos de código abierto más usado en el mundo. La base de datos del proyecto se creará con este sistema.

- **MySQL Workbench**: Es una herramienta visual para trabajar con MySQL. Tiene funciones de desarrollo, administración de usuarios y da la posibilidad de hacer copias de seguridad.

- **Pycharm**: Es un entorno de desarrollo que permite escribir código en Python, HTML y CSS y además incluye herramientas para la compilación y comprobación del funcionamiento del código.
- **Overleaf**: Overleaf es una herramienta de escritura y publicación colaborativa *online* de LaTeX y Rich Text. Con ella, se puede ver en directo cómo se va modificando el fichero PDF resultante del código que se escribe. Además, ofrece control de versiones del documento. La memoria del proyecto será redactada en LaTeX con esta herramienta web. Su página oficial es: `https://www.overleaf.com/`.
- **Cacoo**: Es una aplicación para la creación de diagramas compartibles. Permite realizar diagramas, prototipos e incluso crear planos de edificios. El poder modificar un diagrama a tiempo real y permitir la coedición la hacen una herramienta muy útil.
- **GanttProject**: Es un programa para la creación de diagramas Gantt de manera sencilla. Con solo crear el diagrama Gantt, este programa es capaz de crear un diagrama PERT del proyecto que se esté realizando. Los diagramas Gantt de este proyecto serán realizados con GanttProject. Su página oficial es: `http://www.ganttproject.biz/`.
- **SendGrid**: es un proveedor de envío de mensajes a correos electrónicos que se utilizará para mandar mensajes de confirmación de cuentas de los usuarios. Su página oficial es: `https://sendgrid.com/`

3.3. Arquitectura

En este apartado, se explica cómo es la arquitectura de un detector de vulnerabilidades y el proceso que se lleva a cabo desde que se escanea una página web hasta que se muestra en la aplicación.

Detector de vulnerabilidades

Para hacer funcionar un escáner web se necesita una base de datos MySQL, un *framework* como Django, uno o más subescáneres, un *script* de normalización de resultados, un proveedor de envío de mensajes, un verificador de urls y el interprete del lenguaje de programación. Como en este proyecto se usará Python, el ejemplo de esta sección se hará con ese lenguaje de programación.

Antes de comenzar, en la Figura 2 se muestra un esquema que se seguirá para realizar la explicación de la arquitectura.

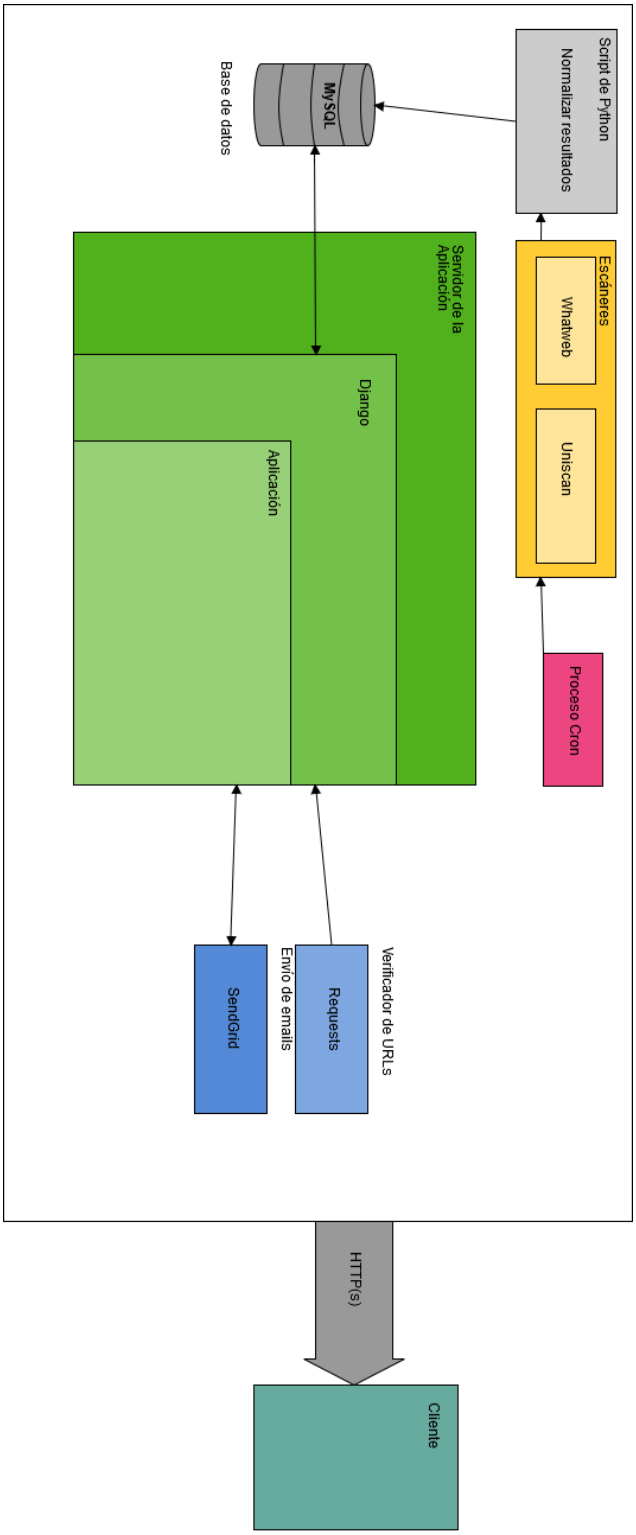

Figura 2: Esquema de la arquitectura de un detector de vulnerabilidades.

Cuando un usuario nuevo quiere darse de alta en el servicio, lo primero que debe hacer es registrarse, indicando su correo electrónico y una contraseña, que se guardarán en la base de datos. A continuación, por medio del proveedor de envío de mensajes, recibirá en su correo electrónico la confirmación de su cuenta. El usuario sólo tendrá que seguir el link para completar su registro.

Una vez registrado deberá identificarse, indicando su correo electrónico y su contraseña, la base de datos comprobará si son correctos y, en caso de que sean correctos, el usuario estará ya conectado a la aplicación.

En la página principal de la aplicación aparecerán todas las empresas que el usuario ha intentado consultar, y si están verificadas o no, además, también dispondrá de un formulario para añadir más empresas.

Para añadir una empresa a la lista de consulta, deberá indicar el nombre de la página web de la empresa en el formulario, que será almacenado en la base de datos con un código de verificación, Cada página web será almacenada en la base de datos con su código y por quién ha sido verificada. Este último dato sólo se rellenará cuando la web haya sido verificada.

Hasta que el usuario no verifique una página web, no podrá consultar su información. Suponiendo que una página está verificada, el usuario podrá acceder a su información siempre que lo requiera.

Para garantizar que todas las páginas son escaneadas s, un proceso cron se encargará periódicamente de escanear todas aquellas que su última fecha escaneo sea la más antigua, para garantizar la información almacenada sobre las páginas web está actualizada. La fecha del último escaneo lo consultará en la base de datos para determinar qué páginas se deben escanear.

3.4. Alcance

Para definir el alcance de este proyecto se ha realizado un diagrama EDT. Se han identificado siete bloques principales: aprendizaje, organización, captura de requisitos, análisis y diseño, implementación y desarrollo, pruebas y documentación.

1. **Aprendizaje**: Bloque al que pertenecen las tareas relacionadas con el estudio de las diferentes herramientas usadas durante el proyecto.

2. **Organización**: Bloque que contiene las tareas necesarias para el buen desarrollo del proyecto.

3. **Captura de requisitos**: Bloque que agrupa las tareas que recogen los datos necesarios para reconocer las funcionalidades a realizar durante el desarrollo del proyecto.

4. **Análisis y diseño**: En este bloque se encuentran las tareas que se encargan de describir gráficamente el código del proyecto.

5. **Implementación y desarrollo**: Este bloque alberga todas las tareas que impliquen la escritura de código en cualquier tipo de lenguaje de programación.

6. **Pruebas**: Bloque que recoge las tareas con los métodos utilizados para la comprobación del correcto funcionamiento de las funcionalidades desarrolladas para el proyecto.

7. **Documentación**: A este último bloque pertenecen las tareas que requieran la redacción de información sobre el proyecto. Se realizará a lo largo de todo el proyecto.

Cada bloque del EDT es precursor al siguiente descrito en la lista, exceptuando el aprendizaje y la documentación. Las tareas de estos dos bloques se desarrollan en paralelo a las demás durante todo el proyecto.

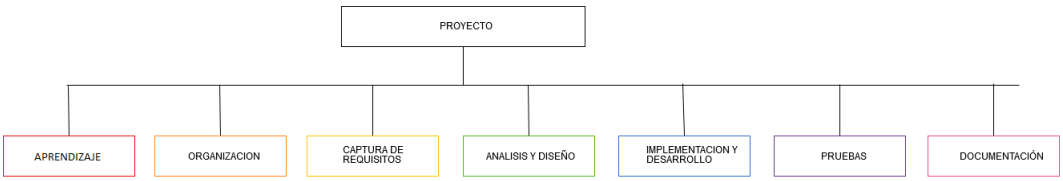

Figura 3: Diagrama EDT por bloques de nivel 1.

La Figura 3 muestra la división del proyecto en los bloques mencionados previamente.

A continuación se mostrarán detalladamente los datos sobre la planificación de cada uno de los bloques junto con las tablas y figuras correspondientes.

3.4.1. Aprendizaje

En este apartado se muestran los detalles de las tareas relacionadas con el aprendizaje del uso de los programas y librerías que se usarán en el proyecto.

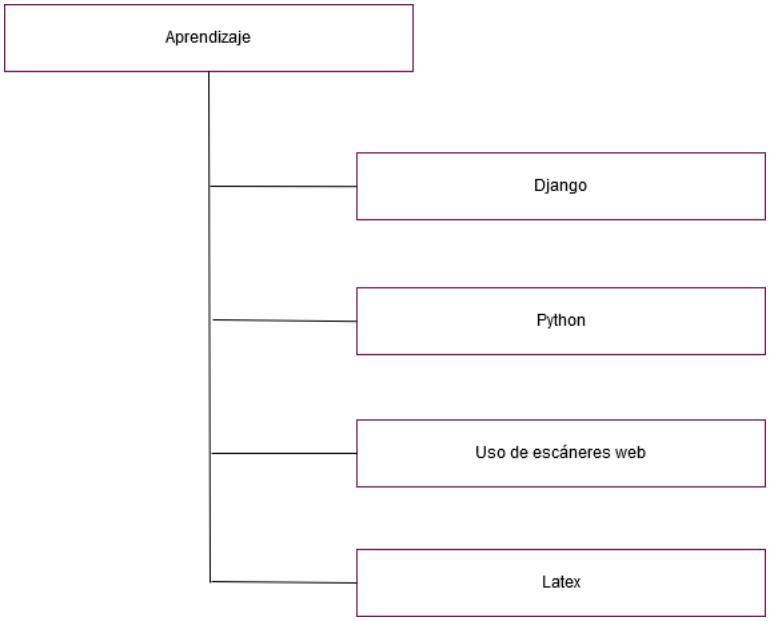

Figura 4: Diagrama EDT del bloque de aprendizaje.

La Figura 4 muestra el apartado del EDT en el que se pueden observar las tareas a realizar durante el aprendizaje. En las siguientes tablas se muestran los detalles de cada tarea.

Aprendizaje de Django
Paquete de trabajo: Aprendizaje. **Duración estimada**: 50 horas.
Descripción: Estudio de la sintaxis y el uso del *framework* de Django [Azzopardi, 2016].
Salidas/Entregables: N/A.
Recursos necesarios: Manual de Django y Python 2.7.x.

Aprendizaje de Python
Paquete de trabajo: Aprendizaje. **Duración estimada**: 30 horas.
Descripción: Estudio de la sintaxis y el uso del lenguaje Python.
Salidas/Entregables: N/A.
Recursos necesarios: Manual de Python y Python 2.7.x

Uso de escáneres web
Paquete de trabajo: Aprendizaje. **Duración estimada**: 4 horas.
Descripción: Estudio del uso de escáneres web y en qué se basan para recoger la información
Salidas/Entregables: N/A.
Recursos necesarios: Instrucciones de un escáner web y ordenador con conexión a Internet

Aprendizaje de LaTeX
Paquete de trabajo: Aprendizaje. **Duración estimada**: 30 horas.
Descripción: Aprendizaje del uso y sintaxis del lenguaje LaTeX para la redacción de textos, creación de tablas, inserción de imágenes, etc.
Salidas/Entregables: Documentación.
Recursos necesarios: Página web `https://www.overleaf.com` y manuales de LaTeX.

3.4.2. Organización

Aquí se expondrán los detalles de todas las tareas que serán necesarias para el correcto desarrollo del proyecto.

La Figura 5 muestra la descomposición de las tareas del bloque de organización. A continuación se detallan las especificaciones de cada tarea y su duración estimada.

Planteamiento de la aplicación
Paquete de trabajo: Organización. **Duración estimada**: 3 horas.
Descripción: En esta tarea se redactará un listado con las ideas de las nuevas funcionalidades que se puedan añadir al proyecto.
Salidas/Entregables: Documento con una lista de las ideas aplicables al proyecto.
Recursos necesarios: Papel y lápiz.

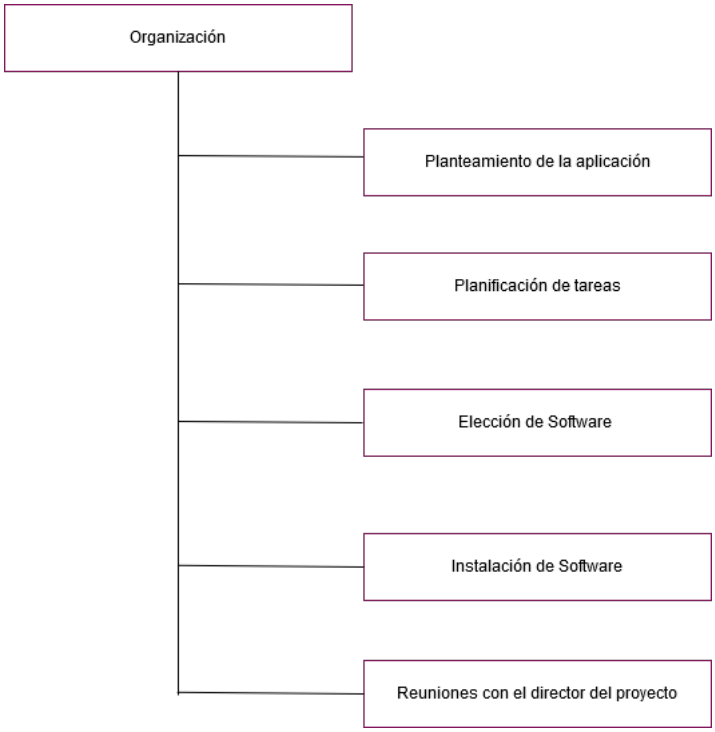

Figura 5: Diagrama EDT del bloque de organización.

Planificación de las tareas
Paquete de trabajo: Organización. **Duración estimada**: 5 horas.
Descripción: Identificación, descripción y planificación temporal de todas las tareas que tendrán que realizarse en el proyecto.
Salidas/Entregables: Diagrama EDT con las tareas a realizar junto con un documento en el que se detalla cada una de las tareas.
Recursos necesarios: Papel, lápiz y Cacoo.

Elección de software
Paquete de trabajo: Organización. **Duración estimada**: 8 horas.
Descripción: Selección de las herramientas *software* que se usarán en el proyecto.
Salidas/Entregables: Documento con el listado de los nombres de las herramientas a usar.
Recursos necesarios: Papel y lápiz.

Instalación de software
Paquete de trabajo: Organización. **Duración estimada**: 6 horas.
Descripción: Instalación de las herramientas necesarias para el desarrollo del proyecto.
Salidas/Entregables: N/A.
Recursos necesarios: Ordenador con conexión a Internet.

Reuniones con el director del proyecto
Paquete de trabajo: Organización. **Duración estimada**: 30 horas.
Descripción: Reuniones que se realizarán durante el desarrollo del proyecto con el director del mismo. En las reuniones se tratarán temas como problemas, modificaciones en la planificación o funcionalidades del proyecto, con el fin de tener un control periódico.
Salidas/Entregables: Un documento con las soluciones, modificaciones o tareas pendientes de las que se habla durante la reunión.
Recursos necesarios: N/A.

3.4.3. Captura de requisitos

En este apartado se presenta la planificación de las tareas que recogen los requisitos a cumplir por las herramientas desarrolladas durante el proyecto.

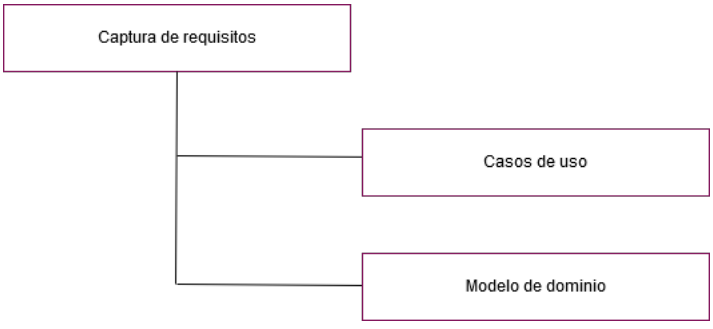

Figura 6: Diagrama EDT del bloque de captura de requisitos.

En la Figura 6 se muestra el desglose de tareas de esta sección. A continuación se muestran la información de la planificación de cada una de las tareas.

Casos de uso
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Identificar y recoger en un diagrama los casos de uso correspondientes al detector de vulnerabilidades.
Salidas/Entregables: Diagrama de casos de uso.
Recursos necesarios: Papel, lápiz y Cacoo.

Modelo de dominio
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 2 horas.
Descripción: Identificar las entidades necesarias para después, crear el diagrama del modelo de dominio de las herramientas desarrolladas en este proyecto.
Salidas/Entregables: Diagrama del modelo de dominio.
Recursos necesarios: Papel, lápiz y Cacoo.

3.4.4. Análisis y diseño

En esta sección se muestran las tareas contenidas por el bloque de análisis y diseño.

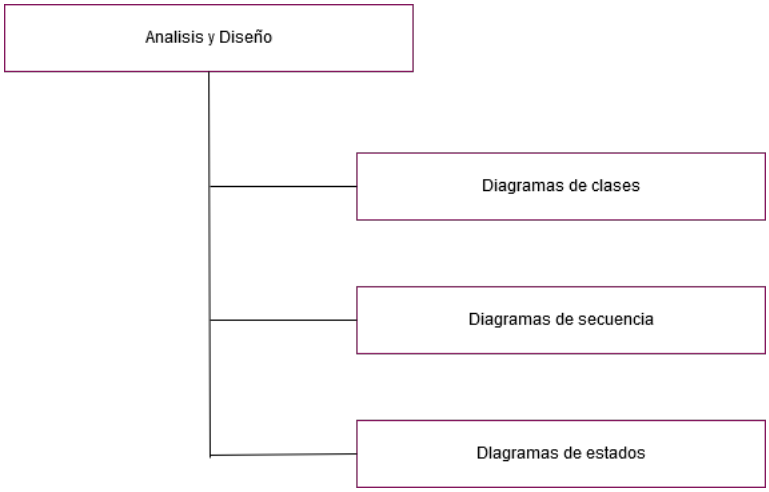

Figura 7: Diagrama EDT del bloque de análisis y diseño.

En la Figura 7 se puede contemplar la descomposición de tareas de este apartado. Ahora se expondrán los detalles y la planificación de cada una de esas ellas.

Diagrama de clases
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 4 horas.
Descripción: Desarrollo del diagrama de las clases que necesitará el escáner para cumplir con los objetivos del proyecto.
Salidas/Entregables: Diagrama de clases.
Recursos necesarios: Papel, lápiz y Cacoo.

Diagramas de secuencia
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 4 horas.
Descripción: Realización de los diagramas de secuencia del escaneo de empresas desde que lo solicita un usuario hasta que se muestra.
Salidas/Entregables: Diagramas de secuencia.
Recursos necesarios: Papel, lápiz y Cacoo.

Diagrama de estados
Paquete de trabajo: Análisis y diseño. **Duración estimada**: 2 horas.
Descripción: Diseño del diagrama que muestra los diferentes estados del escaner web durante la ejecución del escaneo.
Salidas/Entregables: Diagrama de estados.
Recursos necesarios: Papel, lápiz y Cacoo.

3.4.5. Implementación y desarrollo

En esta sección se muestran las tareas a realizar durante la implementación y desarrollo del proyecto.

Tal y como se muestra en la Figura 8, este apartado se divide en dos bloques principales. A continuación se describe cada uno de los bloques y se expone la planificación de las tareas que agrupan.

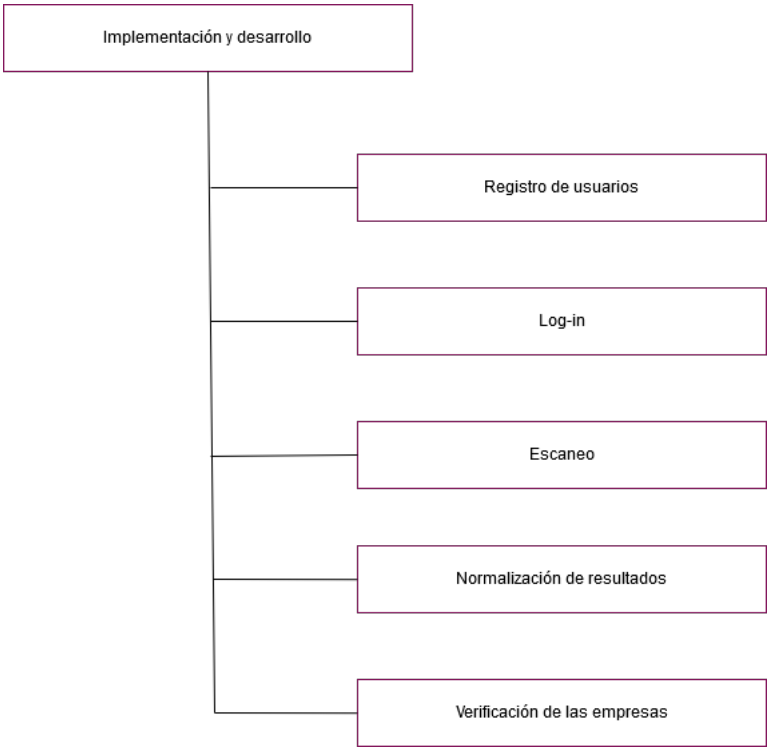

Figura 8: Diagrama EDT del bloque de implementación y desarrollo.

Escaneo
Paquete de trabajo: Implementación y desarrollo - Desarrollo del detector de vulnerabilidades. **Duración estimada**: 20 horas.
Descripción: Adaptación del código de los escáneres elegidos a la necesidades de la aplicación e implementación de código necesario para el almacenamiento en la base de datos
Salidas/Entregables: Ficheros .py con el código necesario para la correcta ejecución del escaneo.
Recursos necesarios: Pycharm, Django, Python 2.7.x y un ordenador con conexión a Internet.

Normalización de resultados
Paquete de trabajo: Implementación y desarrollo - Desarrollo del detector de vulnerabilidades. **Duración estimada**: 4 horas.
Implementación de código necesario para la normalización de los datos en un formato estándar para todos los escáneres y su almacenamiento en la base de datos
Salidas/Entregables: Ficheros .py con el código necesario para la correcta normalización de los datos.
Recursos necesarios: Pycharm, Django, Python 2.7.x, y un ordenador con conexión a Internet.

Verificación de las empresas
Paquete de trabajo: Implementación y desarrollo - Desarrollo del detector de vulnerabilidades. **Duración estimada**: 12 horas.
Implementación del código necesario para la correcta ejecución y visualización del panel de verificación de la aplicación.
Salidas/Entregables: Ficheros .py, .html y css con el código necesario para la correcta ejecución y visualización del panel de verificación de la aplicación.
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

Registro de usuarios
Paquete de trabajo: Implementación y desarrollo - Desarrollo del detector de vulnerabilidades. **Duración estimada**: 6 horas.
Descripción:Implementación del código necesario para el registro de los usuarios y el almacenamiento en la base de datos
Salidas/Entregables: Ficheros .py, .html y css con el código necesario para la correcta ejecución y visualización del panel de registro de la aplicación.
Recursos necesarios: Pycharm, Django, Python 2.7.x y un ordenador con conexión a Internet.

Log-in
Paquete de trabajo: Implementación y desarrollo - Desarrollo del detector de vulnerabilidades. **Duración estimada**: 6 horas.
Implementación del código necesario para la correcta ejecución y visualización del panel de identifiación de la aplicación.
Salidas/Entregables: Ficheros .py, .html y css con el código necesario para la correcta ejecución y visualización del panel de identificación de la aplicación.
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

3.4.6. Pruebas

En esta sección se exponen las diferentes pruebas a realizar para comprobar el correcto funcionamiento de la aplicación del detector de vulnerabilidades. Las pruebas se han hecho de forma manual.

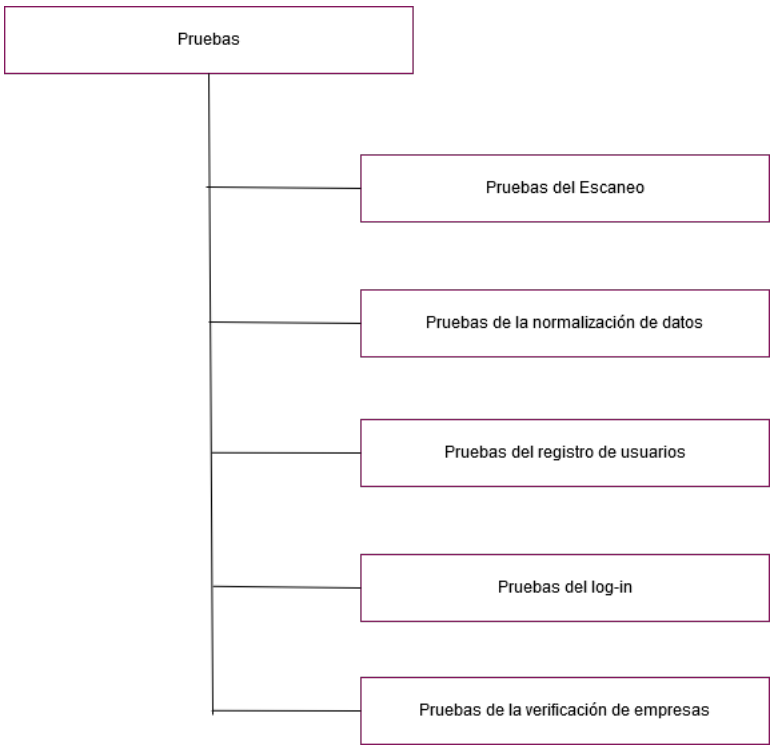

Figura 9: Diagrama EDT del bloque de pruebas.

En la Figura 9 se muestra el desglose de las tareas de este bloque.

Pruebas del escaneo
Paquete de trabajo: Pruebas. **Duración estimada**: 12 horas.
Descripción: Comprobación del correcto funcionamiento de las funcionalidades de los escáneres.
Salidas/Entregables: Informe de los errores que se puedan ocasionar al ejecutar los escaneo
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

Pruebas de la normalización de datos
Paquete de trabajo: Pruebas. **Duración estimada**: 4 horas.
Descripción: Comprobación de la correcta normalización de los datos.
Salidas/Entregables: Informe de los errores que se puedan ocasionar al ejecutar la normalización
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

Pruebas del registro de usuarios
Paquete de trabajo: Pruebas. **Duración estimada**: 2 horas.
Descripción: Comprobación del correcto funcionamiento del registro de usuarios así como del envío de los mensajes de confirmación
Salidas/Entregables: Informe de los errores que se puedan ocasionar al registrar un usuario
Recursos necesarios: SendGrid, Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

Pruebas del log-in
Paquete de trabajo: Pruebas. **Duración estimada**: 2 horas.
Descripción: Comprobación del correcto funcionamiento dela identificación de usuarios
Salidas/Entregables: Informe de los errores que se puedan ocasionar al identificar a un usuario
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

Pruebas de la verificación
Paquete de trabajo: Pruebas. **Duración estimada**: 6 horas.
Descripción: Comprobación del correcto funcionamiento dela verificación de las empresas
Salidas/Entregables: Informe de los errores que se puedan ocasionar al verificar una empresa
Recursos necesarios: Pycharm, Django, Python2.7.x y un ordenador con conexión a Internet.

3.4.7. Documentación

Por último, se detallan las tareas relacionadas con la documentación del proyecto. Este bloque se dividirá en dos secciones para, de esta manera, separar la documentación relacionada con la programación de la documentación de la memoria del proyecto.

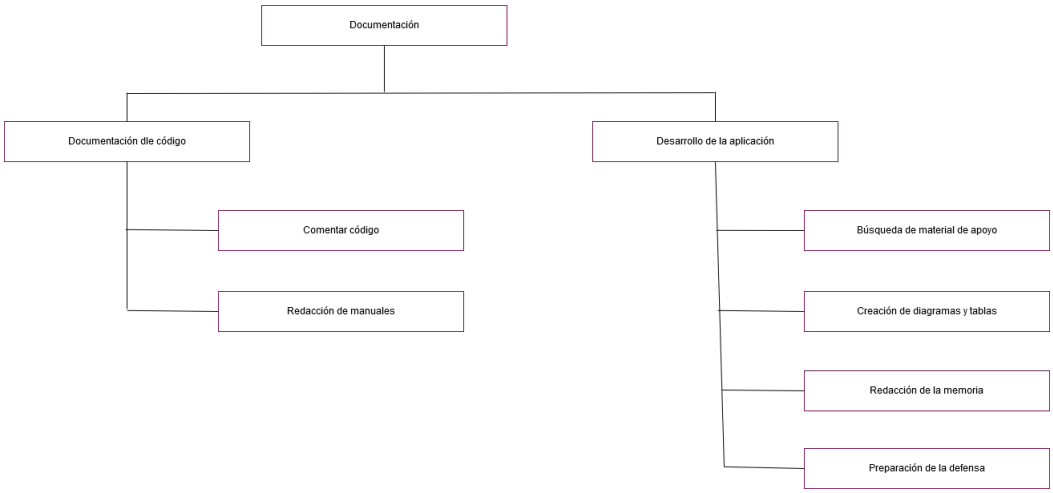

Figura 10: Diagrama EDT del bloque de documentación.

En la Figura 10 se muestra la división mencionada anteriormente junto con la descomposición de las secciones creadas. A continuación se describirá cada una de ellas y se mostrará la planificación realizada para cada tarea.

3.4.7.1. Documentación del código

En este apartado se recogen las tareas de explicar el código y redactar los manuales de uso e instalación de todo lo desarrollado durante el proyecto.

Comentar código
Paquete de trabajo: Documentación - Documentar código. **Duración estimada**: 4 horas.
Descripción: Explicación, mediante comentarios, de las clases, métodos e incluso lineas del código programadas.
Salidas/Entregables: Ficheros de código comentados.
Recursos necesarios: Editores de texto

Redacción de manuales
Paquete de trabajo: Documentación - Documentar código. **Duración estimada**: 6 horas.
Descripción: Redacción de los manuales de uso e instalación del software relativo al proyecto.
Salidas/Entregables: Manuales.
Recursos necesarios: Sublime Text 2.

3.4.7.2. Memoria del proyecto

La memoria del proyecto recoge el desarrollo del proyecto, problemas surgidos durante la realización del mismo, las conclusiones y líneas futuras que se pretendan implementar.

Búsqueda de material de apoyo
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 5 horas.
Descripción: Búsqueda de artículos, libros o documentos que tengan alguna relación con las herramientas usadas durante este proyecto.
Salidas/Entregables: Bibliografía.
Recursos necesarios: Ordenador con conexión a Internet.

Creación de tablas y diagramas
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 10 horas.
Descripción: Creación de las tablas y diagramas, independientes de los apartados de casos de uso y análisis y diseño, que se usarán en este documento.
Salidas/Entregables: Tablas y diagramas.
Recursos necesarios: Dia, Overleaf y GanttProject.

Redacción de la memoria
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 50 horas.
Descripción: Redacción de la memoria del proyecto.
Salidas/Entregables: Memoria del proyecto.
Recursos necesarios: Overleaf.

Preparación de la defensa
Paquete de trabajo: Documentación - Memoria del proyecto. **Duración estimada**: 25 horas.
Descripción: Elaboración y preparación de la presentación de la defensa del proyecto.
Salidas/Entregables: Presentación *Power Point*.
Recursos necesarios: Proyecto y documentación.

3.4.8. Resumen de la planificación realizada

En las siguientes tablas se muestran el nombre, duración y las dependencias de cada una de las tareas identificadas. Como se puede observar en la Tabla **??**, el número de horas totales que se estima para la duración del proyecto asciende a 382 horas.

ID	Nombre de la Tarea	PREDECESORAS	DURACION ESTIMADA
	APRENDIZAJE		**114**
1	Python	-	30
2	Django	1	50
3	Uso de escáneres web	-	4
4	Latex	-	30
	ORGANIZACIÓN		62
5	Planteamiento de la aplicación		3
6	Planificación de las tareas	5	5
7	Elección de software	6	8
8	Instalación de software	7	6
9	Reuniones con el director del proyecto	5	30
	CAPTURA DE REQUISITOS		4
10	Casos de uso	5	2
11	Modelo de dominio	10	2
	ANALISIS Y DISEÑO		10
12	Diagramas de clases	11	4
13	Diagramas de secuencia	12	4
14	Diagramas de estados	10	2

Tabla 1: Dependencias y duración de las tareas (1).

	IMPLEMENTACIÓN Y DESARROLLO		66
15	Registro de usuarios	-	6
16	Log-in	15	6
17	Verificación de las empresas	-	12
18	Escaneos de las webs	17	30
19	Normalización de datos	17,18	12
	PRUEBAS		26
20	Pruebas del escaneo	18,17	12
21	Pruebas de la normalización de datos	17,18,19,20	4
22	Pruebas del registro de usuarios	15	2
23	Pruebas del log-in	16	2
24	Pruebas de la verificación de las empresas	17	6
	DOCUMENTACIÓN		100
	Documentación del código		10
25	Comentar código	15,16,17,18,19	4
26	Redacción de manuales	15,16,17,18,19	6
	Memoria del Proyecto		90
27	Búsqueda del material de apoyo	5	5
28	Creación de tablas y diagramas	27	10
29	Redacción de la memoria	27,28	50
30	Preparación de la defensa	29	25
	TOTAL		382

Tabla 2: Dependencias y duración de las tareas (2).

3.5. Planificación temporal

Una vez se conocen las tareas a realizar, hay que planificar como se desarrollarán durante el tiempo que se ha planeado que dure el proyecto. Para ello se ha realizado el diagrama Gantt que se muestra en la Figura 11. En él, se pueden observar los diferentes bloques mencionados en el alcance del proyecto (Sección 3.4). Como bien se menciona en esa sección, el bloque de documentación ha sido dividido para facilitar el reconocimiento de las tareas.

Durante este proyecto se quiere simular un horario de un trabajador a media jornada que trabaja 28 horas semanales. Como el proyecto comienza el día 26 de agosto de 2016, y se ha planificado su finalización el día 25 de noviembre de 2017, hay un periodo de más o menos tres meses y medio. Suponiendo que un mes tiene más o menos cuatro semanas, se han realizado los siguientes cálculos:

$$Número\ de\ meses * Número\ de\ semanas * Horas\ de\ trabajo = 3{,}5 * 4 * 28 = 392\ horas \quad (1)$$

De las 392 horas resultantes, hay que quitar los días festivos, eventos y fines de semana, en los cuales no se trabajará, a menos que sea necesario para la finalización del proyecto en el plazo previsto. Se ha estimado que el total de horas de trabajo que se perderán durante esos días será de 4 horas. Teniendo en cuenta esto, las horas disponibles para trabajar en el proyecto quedarían de la siguiente manera:

$$Horas\ totales - Horas\ festivas = 392 - 4 = 388\ horas\ disponibles\ para\ trabajar \quad (2)$$

Teniendo en cuenta que la estimación del número de horas totales invertidas en el proyecto será de 382 horas (Sección 3.4.8), se puede apreciar que el tiempo está ajustado para poder entregar el proyecto a tiempo. Además, tal y como se muestra en el Gantt, muchas de las tareas se realizarán en paralelo a otras, ya que las dependencias no lo impiden.

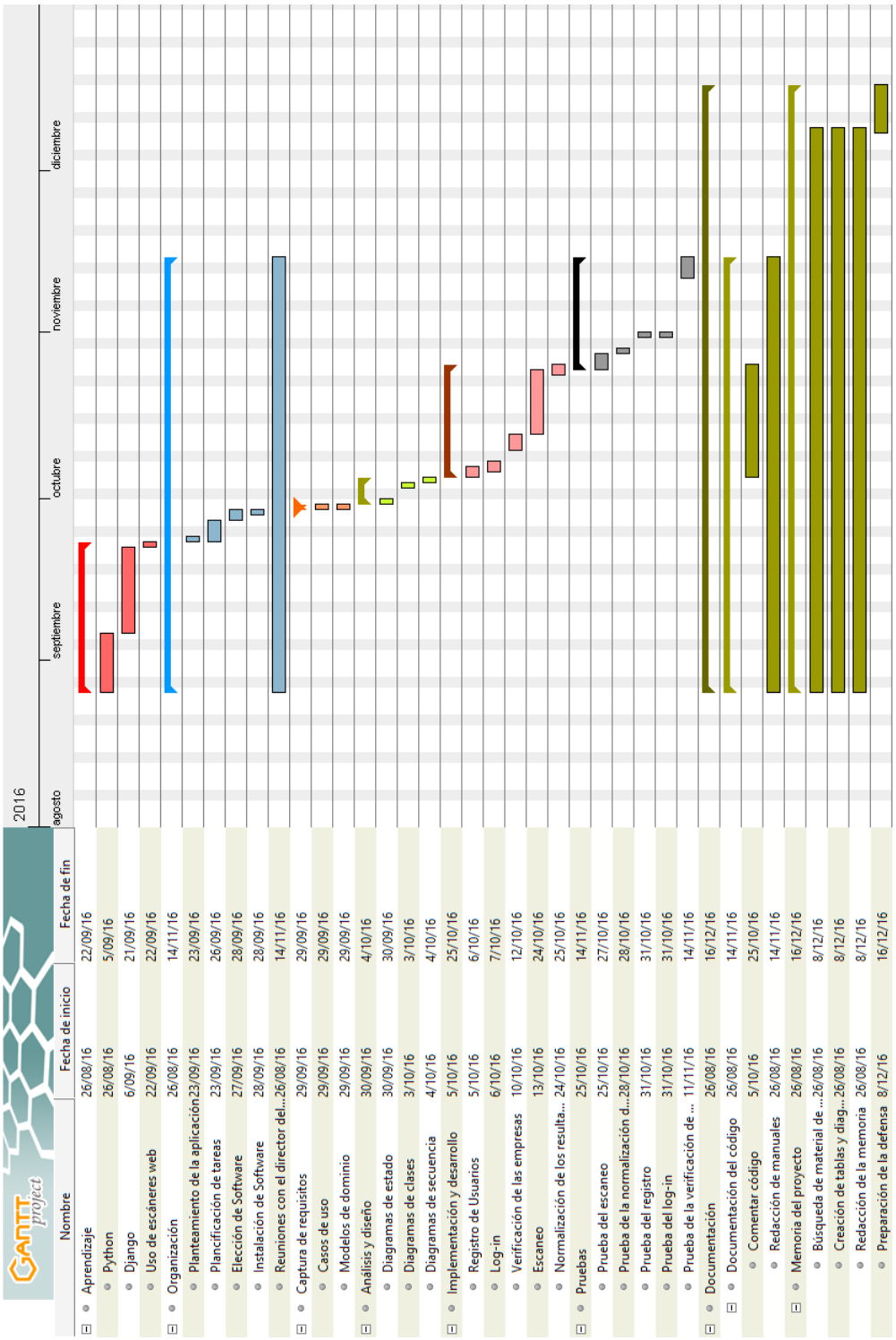

Figura 11: Diagrama Gantt

3.6. Evaluación económica

Aunque este proyecto se trate de un TFG del que no se espera obtener ningún beneficio económico, se ha realizado la siguiente valoración económica por si en un futuro se llegara a comercializar.

3.6.1. Mano de obra

En la publicación de tablas salariales del BOE el miércoles 15 de marzo de 2017, se indica que el sueldo neto de un programador informático es de 1.436,62€ mensuales. El salario por hora de trabajo se puede obtener mediante la siguiente ecuación:

$$Sueldo/hora = \frac{Sueldo\ mes}{Semanas/mes * Días\ trabajo/semana * Horas\ trabajo/día} \tag{3}$$

Un mes tiene, por lo general, 4 semanas de duración, y la jornada laboral está estipulada a 5 días de trabajo a la semana y 8 horas al día. Despejando la ecuación quedaría:

$$Sueldo/hora = \frac{1{,}436,62}{4 * 5 * 8} = \frac{1{,}436,62}{160} = 8,98 \tag{4}$$

Dando un total de 8,98€ por cada hora trabajada.

En la planificación del proyecto, se ha estimado que se trabajarán 382 horas, de las cuales, 114 se dedicarán al aprendizaje del uso de las herramientas que se utilizarán. Esas horas no se van a tener en cuenta a la hora de calcular los costes de mano de obra por lo que:

$$Horas\ totales - Horas\ aprendizaje = 382 - 114 = 268\ horas \tag{5}$$

Y multiplicado por el sueldo/hora establecido anteriormente se obtiene un total de:

$$Sueldo = Horas\ trabajo * Sueldo/hora = 268 * 8,98 = 2604{,}64 \tag{6}$$

El gasto económico en lo referente a la mano de obra sería de 2604,64€.

3.6.2. Gasto de *software*

Como todas las herramientas que se piensan utilizar para el desarrollo del proyecto son gratuitas, en este apartado el gasto en herramientas *software* será de 0€. En este caso, no se está teniendo en cuenta que la universidad ha proporcionado el servidor en el que se han realizado las pruebas del escáner ni que el escáner está en un servidor para poder tener una url de acceso.

Después de mirar varias herramientas, se tendría en cuenta que se hubiese usado un servidor de Microsoft Azure de un coste de más o menos 100€. Por lo que el gasto en *Software* si no se hubiera proporcionado un servidor donde desarrollar el proyecto ascendería a 100€/año

3.6.3. Gasto de *hardware*

En este apartado se tratarán los aspectos económicos relacionados con la amortización del material electrónico que se usará en la realización del proyecto.

3.6.3.1. Ordenador sobremesa Asus Z170A

La vida útil que se estima para este portátil es de unos 7 años (84 meses) y su precio fue unos 1300€ aproximadamente.

Su amortización mensual es:

$$Amortización\ mensual = \frac{Precio}{Vida\ útil} = \frac{1300}{84} = 15,47 \tag{7}$$

El resultado es 15,47€/mes y como el ordenador va a ser usado durante todo el proyecto (3 meses):

$$Amortización\ total = Amort.\ mensual * Meses = 15,47 * 3 = 46,43 \tag{8}$$

Dando un total de 46,43€ en lo referente a la amortización del ordenador.

3.6.3.2. Luz e Internet

Durante el desarrollo del proyecto también se realiza un gasto en electricidad e Internet. Como este gasto puede llegar a variar mucho dependiendo del coste de la luz y de la tarifa de Internet que se tenga, se da por hecho que el gasto total de este gasto indirecto suma aproximadamente 120€.

3.6.4. Gastos totales

En este último apartado de la evaluación económica, se calcula el gasto total que origina la realización del proyecto.

En la Tabla 3 aparece reflejada la suma de todos los gastos mencionados previamente. Dando un total de 4.140,06 € por el completo desarrollo del proyecto.

Tipo de gasto	***Gasto ocasionado (en €)***
Mano de obra	**2604,64 €**
Gasto en *software*	**100 €**
Gasto en *hardware*	**46,43 €**
Gastos indirectos	**120 €**
Total	**2871,07 €**

Tabla 3: Costes totales del proyecto

3.6.5. Posibilidades de negocio

Aunque las herramientas que se desarrollen en este TFG no se piensan comercializar, en caso de que se quisiera llegar a sacar algún beneficio económico de ellas, se ha planteado la siguiente opción.

Como las herramientas que se desarrollen tienen un propósito informativo para todas las empresas del País Vasco, para estimar los beneficios se va a suponer que en el País Vascos hay un total de 2000 empresas. Esto implica un mercado bastante amplio, en el cual se podrían llegar a vender licencias que permiten recibir notificaciones en tiempo real de las vulnerabilidades detectadas por un módico precio de 10 €. Esto implica que con solo 288 ventas de las 2000 posibles ya compensaríamos los gastos (sin tener en cuenta los riesgos). Todas las demás ventas serían beneficios.

3.7. Riesgos

En este apartado se hablará acerca de los posibles riesgos que pueden surgir durante la realización del proyecto, su plan de prevención y la probabilidad de que sucedan.

Se tendrá en cuenta que hay diferentes tipos de riesgos que dependiendo de su gravedad puedan afectar en mayor o menor grado al desarrollo del proyecto.

A la hora de calcular el impacto, se ha estipulado la jornada laboral de entre 4 y 8 horas al día.

3.7.1. Enfermedad o lesión

Consiste en la indisponibilidad por una causa médica, ya sea derivada del trabajo o por causas ajenas a este.

Prevención

- **Derivadas del trabajo**:
  - Mantener una postura correcta a la hora de trabajar.
  - Mantener la distancia correcta frente al monitor.
  - Realizar descansos periódicamente.
- **No derivadas del trabajo**:
  - Dormir las horas recomendadas por los expertos en salud.
  - Realizar actividad deportiva, sin ser de riesgo.

Plan de contingencia

- Acudir al médico para diagnosticar el alcance de la lesión o enfermedad.
- En caso de que el diagnóstico sea favorable, se continuará con el trabajo. Siempre intentando descansar más tiempo del que se hacía hasta el momento.
- En caso de que el diagnóstico sea desfavorable, se alargará el tiempo necesario para recuperarse y poder continuar con el trabajo.

Probabilidad

- Diagnóstico favorable. Baja: 15 %.
- Diagnóstico desfavorable. Muy baja: 5 %.

Impacto

- Diagnóstico favorable. 0,15 x 2 días = 2,4 horas. Impacto medio.
- Diagnóstico desfavorable. 0,05 x 7 días = 2,8 horas. Impacto medio.

3.7.2. Problemas con el equipo informático

Daños imprevistos, tanto de hardware como de software, debido a acciones relacionadas con el trabajo, transporte de material o agentes externos.

Prevención

- Mantener el equipo en unas condiciones ambientales adecuadas.
- Realizar una revisión mensual del equipo.
- Realizar copias de seguridad periódicamente y en varios dispositivos de almacenamiento.

Plan de contingencia

- ***Software***: Se procederá a restaurar la última copia de seguridad guardada y se procederá a realizar el trabajo que se haya perdido.
- ***Hardware***: Se procederá a usar otro ordenador, en el cual se volverán a instalar los programas requeridos y se pasará la copia del trabajo realizado.

Probabilidad

- Problema de *software*. Baja: 15 %.
- Problema de *hardware*. Baja: 5 %.

Impacto

- Fallo de *software*. 0,15 x 1 día = 1,2 horas. Impacto bajo.
- Fallo de *hardware*. 0,05 x 1 día = 0,4 horas. Impacto bajo.

3.7.3. Problemas con las herramientas de desarrollo

Es complicado predecir los problemas que pueden surgir con los programas que se usarán durante el proyecto, sobre todo sabiendo que no se tiene experiencia del uso de gran parte de ellos. Aun así es necesario tenerlos en cuenta.

Prevención

- Tener una buena curva de aprendizaje con los programas.
- Conocer los lugares de donde extraer información. Entre ellos manuales y páginas oficiales de las herramientas.

Plan de contingencia

- Buscar en Internet la solución al problema.
- Aplicar los conocimientos obtenidos con el aprendizaje de las herramientas para intentar solventar el problema.

Probabilidad

- Encontrar un problema y dar con la solución adecuada. Media: 45 %.

Impacto

- Solución no aparente. 0,45 x 3 días = 10,8 horas. Impacto medio.

3.7.4. Discontinuidad en el proyecto

Debido a asignaturas pendiente y a asuntos externos, es posible que el proyecto no se pueda desarrollar de forma continua y que haya paradas durante el desarrollo del mismo.

Prevención

- Intentar llevar las asignaturas al día y resolver los asuntos externos lo más rapido posible para una ágil reincorporación.
- Apuntar las tareas pendientes y por donde se debería continuar el desarrollo.

Plan de contingencia

- Revisar el documento donde se apuntaron las tareas pendientes del proyecto y revisar los archivos correspondientes
- Revisar los tutoriales sobre las herramientas utilizadas para un repaso rápido y poder recuperar los conocimientos olvidados.

Probabilidad

- Parar el desarrollo y continuar un período de tiempo largo después. Media: 45 %.

Impacto

- Solución no aparente. 0,85 x 3 meses = 1836 horas. Impacto muy alto.

3.7.5. Falta de conocimientos para poder comenzar el desarrollo

Debido a la falta de conocimientos respecto a los escáneres web y sus funcionalidad y a nunca haber utilizado uno, el no saber cómo empezar hace que dependa de bastante asistencia externa y que, por tanto, tenga que retrasar el desarrollo del proyecto para documentarme al respecto.

Prevención

- Buscar información actualizada y en páginas fiables para poder tener una buena base de información para empezar.
- Preguntar a personas que hayan trabajado en el sector de la seguridad en internet y que hayan manejado este tipo de herramientas.

Plan de contingencia

- Guardas las páginas web consultadas que han resultado ser útiles para posibles consultas futuras.
- Apuntar en un documento de texto toda la información recibida por las personas que han colaborado en encontrar la información necesaria para comenzar el desarrollo.

Probabilidad

- Retrasar el desarrollo del proyecto para documentarse sobre escáneres web y conocer cuáles se suelen utilizar. Media: 45 %.

Impacto

- Solución no aparente. 0,45 x 15 días = 162 horas. Impacto alto.

4. Antecedentes

En este capítulo, se van a mencionar los proyectos en los que nos hemos basado para el desarrollo. En el proyecto pionero, se desarrolló un escáner web conocido como EuskalCrawler . Su funcionalidad principal es poder consultar una página web y conocer las vulnerabilidades que detectaba, así como las versiones de su *software*.

4.1. EuskalCrawler

Es el resultado de un proyecto desarrollado por David López Angulo en HTML;CSS,JavaScript y Python sin ningún tipo de soporte.A continuación se muestra un esquema las funcionalidades antiguas del proyecto y las que se han añadido con esta actualización:

Figura 12: Funcionalidades antiguas y nuevas

4.2. Situación actual del proyecto

Este proyecto se centrará en implementar una aplicación desde cero con un framework de desarrollo, en lugar de HTML, CSS, JavaScript y Python, únicamente.

El registro y la identificación de usuarios cobra más sentido en esta versión. Esto se debe a que se realiza una verificación de las páginas de las empresas que se van a escanear en las que se ha de añadir una URL con un contenido específico basado en un número de 16 dígitos pseudo-aleatorios. Ésta operación, sólo la podría hacer el verdadero dueño de usa página y, que deberá estar registrado en la aplicación con un nombre de usuario único. Finalmente, una vez realizada la verificación, la página web queda asociada al usuario que la verificó para que sólo él pueda consultar la información.

Para garantizar la seguridad de los datos de los usuarios registrados, se ha realizado un cifrado de las contraseñas para que sea más difícil dar con ellas.

También se quiere mejorar la validez de los datos obtenidos por la aplicación, por lo que no se usará un sólo escáner de vulnerabilidades, sino que en esta ocasión se han utilizado dos, bastante conocidos a nivel informático para la detección de versiones y vulnerabilidades en las páginas web.

Por último, ya que se ha supuesto que un gerente puede tener varias empresas en marcha, se ha implementado una funcionalidad para que en la página principal de la aplicación, aparezca la lista de empresas del usuario registrado y su estado (verificada o no, y si ha sido escaneada).

5. Captura de requisitos

En este capítulo, se recogen los requisitos que ha de cumplir la herramienta desarrollada durante el proyecto, de manera que se satisfagan las necesidades de los usuarios que la utilicen.

Cabe destacar que durante el desarrollo, se han realizado cambios en todos los diagramas. Esto se ha debido a los diferentes factores que han afectado al proyecto. En las siguientes secciones, se mostrarán los diagramas relacionados a la captura de requisitos y sus modificaciones. Estas últimas, han sido provocadas por las nuevas necesidades que no se habían tenido en cuenta en el planteamiento inicial (Sección 3).

El cliente Juanan Pereira, identificó algunas necesidades imprevistas. Esas nuevas necesidades se exponen como modificaciones en esta sección.

5.1. Jerarquía de actores

Como se comentó en el apartado de objetivos (Sección 3.1), la seguridad es algo a tener en cuenta en este proyecto. Es por ello que el escáner tiene que verificar las empresas que se quiere escanear, para que un usuario no identificado no acceda a su información. Para ello, hay que registrarse y verificar la empresa introduciendo la información que la aplicación pida. De está manera el usuario se habrá registrado como cliente en nuestra aplicación. En la figura 13 se muestra la jerarquía de los actores.

5.2. Casos de uso

En este apartado, se muestran en diagramas las posibles acciones que puede tomar cada tipo de usuario mientras utiliza las herramientas desarrolladas durante este proyecto. Como se ha mostrado en la sección anterior (Sección 5.1), se han identificado dos tipos de actores, empresario y usuario estándar, que se describen a continuación junto con sus diagramas de casos de uso. Los detalles de las modificaciones se explicarán el capítulo de desarrollo (Sección 8).

5.2.1. Empresario

Este es el tipo de usuario registrado capaz de usar todas las funcionalidades de la aplicación. Si un usuario no registrado intenta realizar cualquier

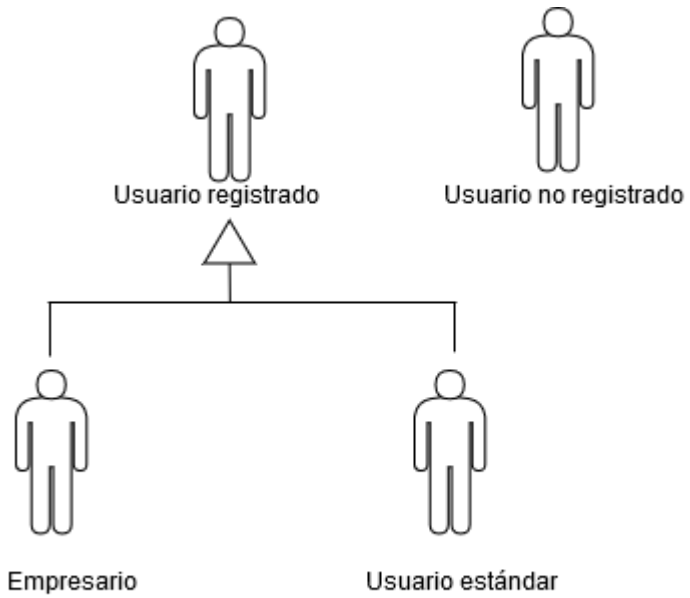

Figura 13: Jerarquía de actores

funcionalidad que no sea de registro o identificación, automáticamente se le redirigirá a dichos formularios.

Este usuario se supone que ya ha verificado una empresa, pero, aún así, puede añadir más empresas para consultar su información, como se muestra en el siguiente diagrama de la figura 14.

5.2.2. Usuario registrado

Este es el tipo de usuario registrado capaz de usar todas las funcionalidades de la aplicación. La única diferencia que tiene con el empresario es que no ha verificado ninguna página web, por lo tanto, no puede consultar la información de ninguna página web, hasta que la haya verificado. Como se muestra en el diagrama de la figura 15

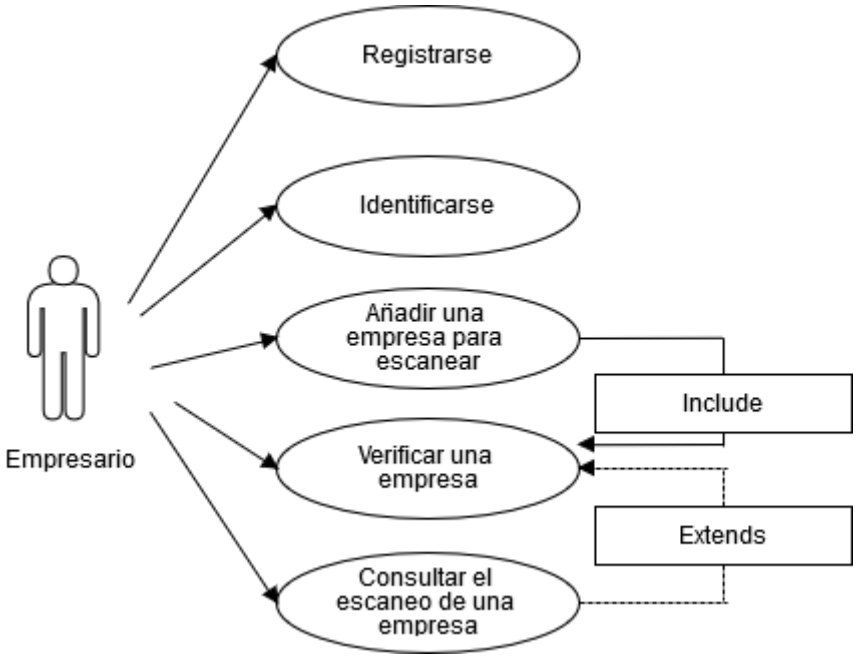

Figura 14: Diagrama de casos de uso inicial de Empresario

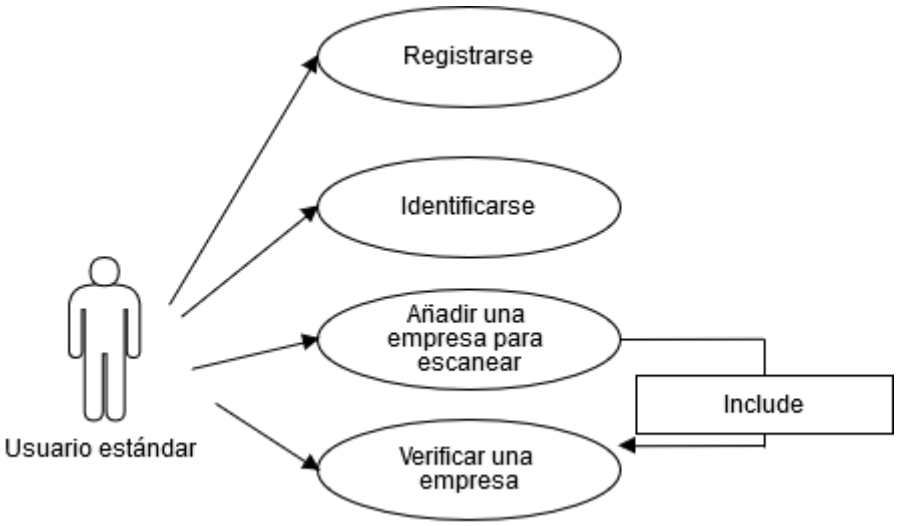

Figura 15: Diagrama de casos de uso inicial de Usuario Estándar

5.3. Modelo de dominio

El modelo de dominio es un modelo que representa, de manera conceptual, las entidades, relaciones y atributos relacionados con un problema específico. En esta sección, se muestra el modelo de dominio desarrollado durante este proyecto. Solo se mostrará la última versión.

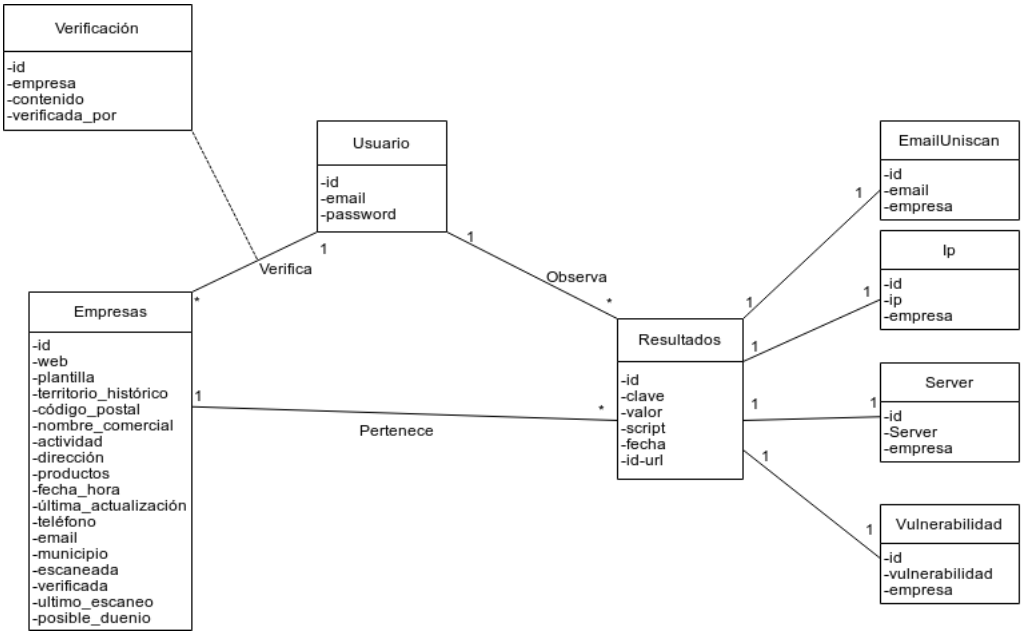

Figura 16: Diagrama de modelo de dominio

A continuación, se describe cada una de las entidades que aparecen en la Figura 16:

- **Empresa**: Es la entidad que representa a la empresa. Debido a que guardaba los datos que se muestran en el diagrama, se han conservado todos, pero realmente sólo interesan para el proyecto la web, verificada (que indica si ha sido verificada), escaneada (que indica si ha sido escaneada), el último escaneo (la fecha) y el posible dueño, es decir el usuario que solicitó escanear dicha empresa.

- **Usuario**: Es la entidad que encarna un grupo de usuarios. En ella se guarda un identificador, el email y el password con el que se identificará en la aplicación.

- **Verificación**: Es una entidad resultante de cuando un usuario intenta verificar una empresa, guarda un identificador, la empresa a la que pertenece la verificación, el contenido que debe contener la nueva url (que también será el título de la url) y por quién ha sido verificada.

- **Resultados**: Representa a todos los resultados normalizados de WhatWeb y Uniscan. Guarda un identificador, la clave, que será el tipo de resultado que se está guardando, el valor de ese resultado, el Script que encontró el dato (en este caso será WhatWeb o Uniscan) la fecha en la que se guardó y el identificador de la empresa a la que pertenece.

- **EmailUniscan,IP,*Server* y vulnerabilidad**t: son los resultados que recoge Uniscan, se han tenido que crear esas entidades de forma manual, ya que Uniscan no soporta el guardado en bases de datos, pero extrae los resultados a un fichero de texto plano. Por medio de búsquedas con expresiones regulares en el texto, se pueden guardar los resultados en la base de datos con éstas entidades.

- **DatosWhatweb**: No aparecen en el diagrama, debido a que no se han tenido que crear manualmente, si no que son creados automáticamente al instalar WhatWeb, todos los datos encontrados por Whatweb, son guardados en la entidad de Resultados.

Las relaciones de 1-* como por ejemplo las que se dan con el usuario y la empresa en el caso de la verificación significa que: un usuario puede verificar muchas empresas, pero una empresa sólo puede ser verificada por un usuario, además, como resultado, se crea una verificación.

Las relaciones de 1-1 que se dan en los resultados, significa que cada dato pertenece a 1 resultado y cada resultado sólo puede representar 1 dato.

El número de entidades mostradas en el diagrama no son todas las que realmente hay, ya que faltan todas las entidades creadas por Django y por los escáneres utilizados para este proyecto, el hecho de que no se muestren es por legibilidad del diagrama, ya que al mostrarlas, habría demasiadas entidades y relaciones entre ellas. Otro motivo por el que no se han mostrado, ha sido porque no han sido desarrolladas por mí, sino que han sido añadidas directamente a la hora de instalar Django y los escáneres utilizados.

6. Análisis y diseño

En esta sección, se exponen el diagrama de clases, diagrama de estados y diagramas de secuencia de la aplicación para el correcto funcionamiento del detector de vulnerabilidades.

6.1. Análisis de elementos

En este apartado se explican de una forma sencilla, aquellos elementos internos, que tienen que entenderse para el correcto desarrollo de los eventos que se producen cuando se quiere escanear una página web.

6.2. Diagrama de clases

Es un diagrama en el que se representa la estructura de un sistema. Para ello, se muestran las clases, métodos, atributos y relaciones de manera estática. En esta sección, se mostrarán las clases que se han tenido que crear de forma manual para el correcto funcionamiento de la aplicación, las clases predefinidas de DJango, no se tendrán en cuenta, ya que son automáticamente creadas por él y son necesarias para su funcionamiento y soporte.

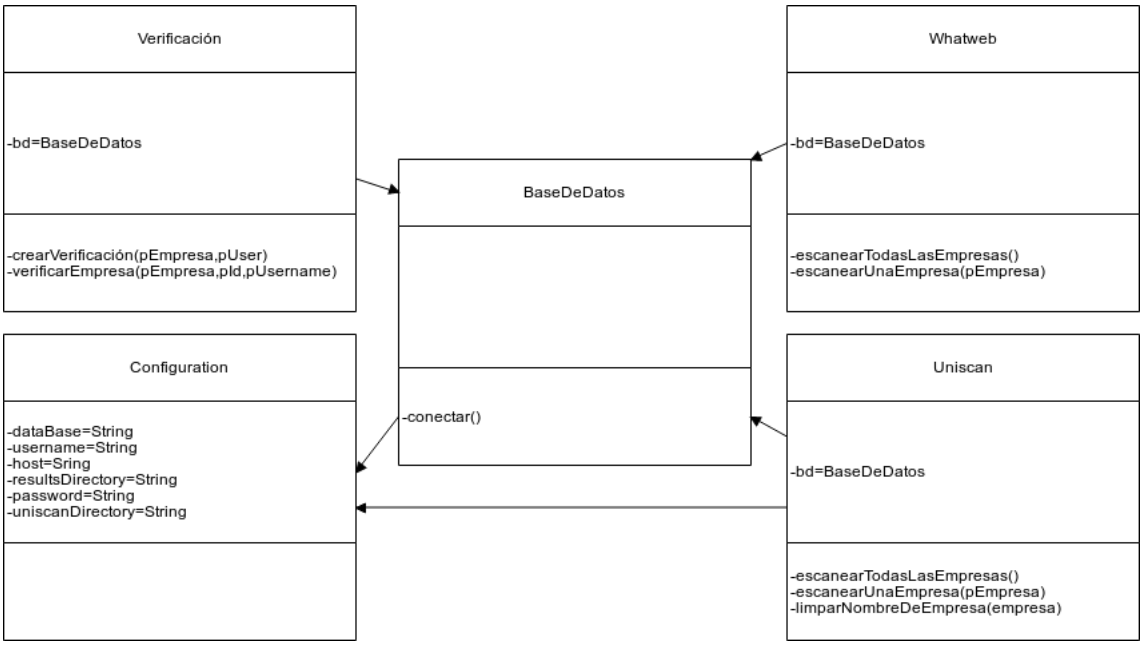

Figura 17: Diagrama de clases

A continuación se procede a explicar cada una de las clases del diagrama:

- ***Verificacion***: Esta clase es la encargada de crear las verificaciones para las empresas y también para verificarlas.
- **BaseDeDatos**: Esta clase se encarga de conectarse a la base de datos, requiere de la clase Configuration.
- ***Configuration***: Clase encargada de almacenar todos los datos necesarios para la configuración del proyecto, así como los datos referentes a la base de datos donde se va a conectar la aplicación, como también la localización de los archivos donde se quieren guardar los resultados de los escaneos. También guarda la localización de Uniscan, necesaria para poder llamar a los procesos de Uniscan.
- **Whatweb**: Clase que escanea las diferentes empresas utilizando Whatweb
- **Uniscan**: Clase que escanea las diferentes empresas utilizando Uniscan
- **NormalizarResultados**: Es un *script* que no aparece en el diagrama de clases, ya que no es una clase como tal, sino que es una función que se tuvo que incorporar para poder normalizar los datos de Whatweb y Uniscan, ya que los datos que recogen están en formatos muy diferentes.

Todas las clases descritas en este apartado han sido creadas o modificadas durante el desarrollo del proyecto. También es importante decir que detrás del funcionamiento del escáner, hay muchas clases que implementan las funcionalidades que tiene la aplicación. Como las ya mencionadas de Django, así como todos los subprocesos internos de Whatweb y Uniscan.

6.3. Diagramas de secuencia

Los diagramas de secuencia tienen como objetivo representar gráficamente la interacción entre los objetos de un sistema. Para este proyecto, se ha creado el diagrama de secuencia de la ejecución del escaneo y de todas las comprobaciones que se realizan durante el mismo.

El diagrama de secuencia que se muestra en la Figura 18, es la versión final. Se tuvieron que realizar algunos cambios en su estructuras debido a las nuevas necesidades del cliente y a algunas mejoras de eficiencia. Como se puede observar, no se ven las clases de los escáneres (Whatweb y Uniscan)

y tampoco la de la normalización de resultados. Esto es debido a que se les llama desde fuera de forma periódica por medio de procesos Cron, es decir, les llama el servidor, no la aplicación. Por otro lado, se observa que aparece una clase que no se ha mencionado hasta el momento (Views), ésta es una de las clase de Django que no se ha querido mostrar anteriormente, ya que son clases que no hay que implementar manualmente, sí que hay que implementar las funciones, pero la clase viene creada por defecto, para el correcto funcionamiento de Django. Para esta sección, se ha considerado que la aparición de esta clase es muy importante para poder entender correctamente el diagrama de secuencia.

Como ya se ha comentado en este capítulo, el escaneo se compone de diferentes estados. Para simplificar la explicación, se detallará lo que se realiza en cada estado.

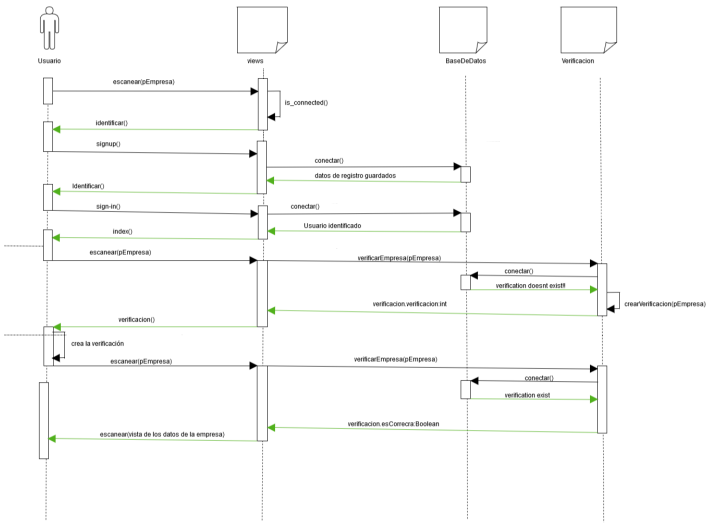

Figura 18: Diagrama de secuencia del escaneo

Al comienzo del diagrama,el usuario quiere escanear una página web. El sistema comprueba que el usuario está conectado e identificado a la aplicación, como no está conectado, le devuelve la vista de la identificación. El usuario no tiene cuenta, entonces le pide a la aplicación la vista del registro y mete sus datos, la clase *Views* se conecta con la base de datos y guarda los datos, internamente se ha mandado un correo electrónico al usuario, pero eso es parte de la función *Sign-up* de *Views*, por lo que no se muestra en el diagrama de secuencia. Una vez registrado el usuario, se le devuelve a la vista de identificación. El usuario quiere identificarse, para ello inserta los

datos de identificación, y la clase *Views* se comunica con BaseDeDatos para cotejar los datos, si son correctos, el usuario se conecta a la aplicación y se le devuelve a la página principal.En este punto, el usuario vuelve a solicitar escanear una página web. La clase *Views* llama a la clase Verificación para verificar esa empresa. El usuario no tiene la verificiación creada y entonces cierra la aplicación.Cuando el usuario crea la verificación vuelve a solicitar que se escanee la página web. La verificación encuentra los datos y, entonces, le muestra los datos en la *View* donde se muestran los datos.

7. Elección de lenguajes y tecnologías

A la hora de desarrollar un proyecto, es muy importante el hecho de escoger las herramientas adecuadas. La mala selección de estas, no solo implica la pérdida de tiempo de todos los integrantes del proyecto, sino que puede suponer la ruina económica de este o, en casos extremos, de una empresa completa.

Al tratarse de un proyecto en el que la implementación de código supone prácticamente el total de la realización, es realmente importante hacer una selección de los lenguajes de programación, librerías y programas a utilizar. Una correcta elección de estos puede ahorrar muchos problemas y horas de trabajo, incluso aunque no se conozcan, ya que unos lenguajes son más adecuados que otros para este tipo de aplicaciones. Es por ello que en este proyecto se plantearon varios lenguajes con los que trabajar. En esta sección se muestran los planteamientos que se realizaron a la hora de escoger un lenguaje de programación.

7.1. Elección del SDK

SDK son las siglas de *software development kit*, que hacen referencia al conjunto de herramientas que un desarrollador de *software* utiliza a la hora de crear aplicaciones para un sistema concreto.

Al comienzo de este proyecto se planteó la opción de desarrollarlo con el *framework* de Django, ya que permitía ordenar las páginas de la aplicación web de una forma coherente, así como también soportaba el manejo de bases de datos y de vista de las pantallas de forma dinámica.

Me pareció una idea interesante, pero tenía que tener en cuenta las horas que iba a emplear para aprender a utilizar ésta herramienta para la planificación del TFG, y, al ir completando el tutorial de Django, más me fui convenciendo de que estábamos escogiendo la opción correcta, por lo que en ningún momento se llegó a plantear otra opción.

7.2. Elección del lenguaje de programación

Cuando le pregunté a Juanan Pereira, el director de este proyecto, por los proyectos de TFG en los que se desarrollan aplicaciones web, me comentó que él veía una buena oportunidad para Django en éste, lo cual significaba tener que trabajar por medio del lenguaje de Python, además de HTML y

CSS, que ya conocía y que son básicos para el desarrollo de una aplicación web.

Este hecho me ayudó a decidirme por el lenguaje a escoger, pero, una vez más, debía tener en cuenta las horas de aprendizaje que me conllevaría aprender la correcta utilización de Python para la creación de aplicaciones web. Juanan conocía bien Django y Python, lo que suponía una ventaja que podría compensar las horas gastadas en el aprendizaje de Python y Django, ya que podría obtener ayuda de Juanan en caso de tener algún error del que no encontrara la solución.

Por todo ésto, y porque, además, me interesaba aprender un lenguaje nuevo, la opción de Python, junto con HTML y CSS, fue una decisión definitiva prácticamente desde el momento en el que se puso en práctica.

7.3. Elección del entorno de desarrollo

Ya tenía el lenguaje de programación, pero no quería ponerme a implementar el código en un fichero de texto plano sin ningún tipo de compilador, así que le consulté a Juanan al respecto, quien me recomendó el entorno de desarrollo de Pycharm.

Decidí instalarlo en mi ordenador para probarlo por mi cuenta, y pronto me dí cuenta de que era bastante intuitivo y no sólo era útil para desarrollar en Python, sino que también en HTML, CSS, JavaScript...lo que me iba a resultar muy útil para el desarrollo de la aplicación tanto a nivel funcional como también visual.

Por ello la elección de ésta herramienta también fue definitiva desde el principio.

7.4. Elección de los escáneres web

Ésta fue la elección más difícil de todas, de hecho, gran parte del proyecto se retrasó debido a ésta elección. Ya que Juanan me recomendó algunos que eran bastante conocidos y que se utilizaban bastante, pero tenían sus inconvenientes.

Los resultados obtenidos eran satisfactorios, pero si dispusiera de más tiempo, habría añadido más escáneres web para poder encontrar datos más fiables y más relevantes. En un futuro, hacer esto podría ser una buena vía para mejorar el proyecto.

8. Desarrollo

En este capítulo se van a detallar los pasos que se siguieron a la hora de desarrollar el detector de vulnerabilidades y su aplicación web. Además, se detallarán los pasos seguidos para implementar las necesidades que el cliente ha ido encontrando durante el desarrollo del proyecto. También se explicarán los problemas encontrados y las modificaciones que han sido necesarias para solucionarlos.

Como durante el desarrollo del proyecto han surgido nuevas necesidades y problemas técnicos que no se habían tenido en cuenta, el alcance y la planificación han sido severamente alterados. Estas modificaciones se detallan en el apartado de conclusiones (Sección 10).

8.1. Necesidades del cliente

Antes de comenzar a comentar cuál fue el proceso que siguió el desarrollo, se han de explicar las necesidades que el director y cliente, Juanan Pereira, identificó mientras se realizaba el diseño y la implementación del proyecto. Para ello, primero se explicarán las que afectaron al desarrollo de un buen escáner web y después las que afectaron a la aplicación.

8.1.1. Mejora del escaneo de las páginas web

Al ver el desarrollo del proyecto antecedente a este, no se tardó mucho en identificar los problemas que traía escanear las páginas web sólo con un escáner web, además de la mala elección de éste.

- **Falta de seguridad**: este apartado no se refiere a la seguridad en la aplicación, de la cual se hablará en otra sección. Más bien se refiere a la seguridad de que los datos recogidos por el escáner web son correctos y que no se está dando información falsa sobre la página de los clientes.

- **Fallos de escaneo**: este problema no se tardó en detectar, de hecho en, prácticamente, en todas las combinaciones que se hicieron para cotejar los datos recogidos por el escáner con los datos reales, se detectaba mínimo un error en los datos que iba capturando, por lo tanto, se había desarrollado un detector de vulnerabilidades que no capturaba los datos correctamente.

- **Falta de datos relevantes**: Lo que se entiende por un detector de vulnerabilidades es, precisamente, que detecta vulnerabilidades, y el término vulnerabilidad no se refiere únicamente a versiones obsoletas, sino también a fallos en la seguridad o puertas abiertas que se han dejado que pueden ser graves para la página web que se está escaneando. El detector de vulnerabilidades que se utilizaba en la versión anterior cumplía, cuando acertaba, con el primer tipo de vulnerabilidades, pero de las del segundo tipo, el cual es muy importante, no se observó que detectara ninguna.

Por esto, se necesitaba mejorar el escaneo de las páginas web, ya fuera utilizando otro escáner o la combinación de varios y, para tranquilizar al cliente a la hora de recibir los resultados de los escaneos, finalmente se optó por las dos cosas. Esto supuso un retraso de tres semanas en la implementación de la aplicación, ya que, como se ha comentado anteriormente, no se conseguía encontrar dos detectores de vulnerabilidades que satisficiera las necesidades para el desarrollo.

8.1.2. Mejora en la seguridad de la aplicación

En este apartado se recogen todas las necesidades de los clientes relacionadas con la seguridad en la aplicación y en la protección de los datos de los usuarios.

- **Verificación de las webs escaneadas**: Esta necesidad no era una necesidad inicial, pero conforme me iba reuniendo con Juanan, la vimos más necesaria, ya que a los usuarios de la aplicación no les interesaba que cualquiera pudiera ver la información de su empresa. Además, se estaba rozando la ilegalidad si el escáner recogía datos de la página web sin el permiso del dueño real de esa página web, por lo que verificar que el usuario que quería escanear una página web fuera el dueño de la misma se convirtió en una necesidad primordial.
- **Cifrado de contraseñas**: Ésta necesidad estaba presupuesta para el desarrollo de una aplicación relacionada con la seguridad, pero se observó que, en el antecedente a este proyecto, no se había llevado a cabo, lo cual era un grave error de seguridad para un desarrollador de aplicaciones web relacionadas con este campo de la informática.
- **Confirmación de emails**: Hoy en día, todas las aplicaciones web disponen de este servicio, y la aplicación desarrollada en este TFG no

se iba a quedar atrás, no fue una necesidad inicial, pero se decidió implementar más adelante. La implementación de una confirmación de emails por medio del servicio de SendGrid trajo sus problemas para poderla adaptar al código ya implementado, y supuso un retraso de una semana para el desarrollo del proyecto.

Todas estas necesidades se consiguieron implementar satisfactoriamente finalmente, llegando a implementar una aplicación robusta y segura.

8.2. Desarrollo de la base de datos

Durante el desarrollo del proyecto, la base de datos se ha tenido que ir modificando por dos razones principales. La primera, son todas las nuevas necesidades que se identificaban mientras se implementaba el proyecto. La segunda razón es por las librerías de Django.

Al ir añadiendo las nuevas funcionalidades, estas requerían de nuevos datos que no se veían reflejados en la base de datos. Es por ello que, necesidades como 'Verificación de las webs escaneadas" o " Mejora del escaneo de las páginas web", hicieron que se tuviera que modificar la base de datos para poder guardar la información que se necesitaba para implementarlas.

Las librerías de Django también estuvieron implicadas en las modificaciones realizadas en la base de datos. La base de datos desarrollada durante este proyecto se compone de tres partes que están relacionadas por una tabla. Una parte está formada por las tablas de la base de datos que las librerías de Django proporcionan para su correcto funcionamiento. La segunda parte, es la que contiene todas las tablas que los escáneres web utilizados durante el proyecto crean durante su instalación. Y la tercera parte, es la que se ha desarrollado durante el proyecto para guardar los datos necesarios para la realización de todas las funcionalidades mencionadas anteriormente.

El problema con la parte de la base de datos de las librerías de Django ha sido su sincronización con las otras partes. Django guarda el resto de las entidades en un fichero que se llama models.py, es decir, los modelos de la base de datos. Pero, si se quisiera añadir una nueva tupla a un modelo ya existente, había que modificar también ese fichero y luego sincronizar los datos con Django. Dicha sincronización era el principal problema, ya que muchas veces no permitía algunos de los cambios que se querían hacer de la base de datos, como por ejemplo cambiar el tipo de dato que se guardaba en una tupla concreta o la eliminación de tuplas existentes, ya que continuamente mandaba avisos de que se podían perder datos con el borrado de esas tuplas, aunque

estuvieran vacías, como era el caso. Para solucionar este problema, había que eliminar las copias de seguridad que almacenaba Django de las bases de datos para verificar la integridad de los datos. Como consecuencia, el desarrollador tiene que realizar sus propias copias de seguridad por si el cambio no era correcto y había que volver a la versión anterior.

Este último problema se hubiera solucionado simplemente haciendo que Django comprobara si al eliminar una tupla, realmente se pierden datos, y no valores *Null*, ya que, en mi caso, tenía en todo momento controlado lo que estaba eliminando y sabía que no perdía ningun dato, ya fuera importante o no.

El diagrama de la base de datos con la que se ha finalizado el proyecto se muestra en la sección 5.3. No se ha querido mostrar la base de datos del proyecto antecedente ya que no se heredó nada de ella, y la base de datos actual es totalmente independiente de la anterior. En dicha sección se explica porque parece que hay tan pocas entidades, y es que no se ha querido mostrar todas las tablas de Django ni las creadas por Whatweb, ya que el modelo de domino podía llegar a ser ilegible por la cantidad de entidades que había que mostrar.

8.3. Desarrollo de la aplicación

En un principio, se pensó que el desarrollo del escáner web no llevaría tanto trabajo como el que realmente se ha realizado. Esto se debió a que se pensó que se podría utilizar algo de lo que David López había desarrollado. Pero como ya se ha comentado durante este documento, se desechó todo lo que él realizo por los errores que tenía y porque no tenía ningún soporte de desarrollo, eran ficheros HTML, CSS, JavaSCript y Python, sin ningún *framework* que los soportara.

Antes de desechar el código de David, Juanan me prestó el código que él había desarrollado, para ver si se podía llegar a utilizar algo. El código se desechó rápidamente y no se pudo usar nada de lo desarrollado para aplicación actual, pero fue un código que se utilizó. El código desarrollado resultó muy útil para entender como se comunicaba una aplicación web con los escáneres de vulnerabilidades y también para entender en qué se basaban los detectores de vulnerabilidades y su comportamiento.

El escáner se comenzó a desarrollar de manera local. Las librerías de Django permiten desarrollar aplicaciones sin necesidad de un servidor web simplemente indicando que lo quieres desarrollar como *localhost*. Para ello

había que ejecutar un comando cada vez que se quería ver la aplicación y su funcionamiento y presentación de los datos.

Al comienzo del desarrollo de la aplicación, el trabajo se centró en encontrar escáneres web válidos y en adaptar el código que ellos tenían implementado para las necesidades del proyecto, llegando a implementar un código para que ellos se ejecutaran de forma correcta y para normalizar los resultados que recogían. Además, también había que hacer modificaciones en los nombres de las webs que se almacenaban en la base de datos para que estuvieran en un formato válido para los diferentes escáneres. Esta operación también debía realizarse para las futuras inserciones de webs en la base de datos a través de la aplicación.

Una vez realizadas las funcionalidades básicas para comenzar a escanear webs, se prosiguió al desarrollo de la aplicación completa con Django. Mientras se desarrollaba la aplicación en local, había reuniones periódicas con Juanan para comentar problemas y corregir errores. También se fueron viendo más necesidades para el desarrollo final del proyecto.

Uno de los principales problemas para el desarrollo fue la normalización de los datos de los escáneres, ya que uno guardaba los datos en la base de datos en un formato concreto, y el otro los mostraba en ficheros de texto plano en otro formato completamente distinto. Juanan me propuso una solución, la cual resulto ser muy eficaz. Dicha solución consistía, primero, en guardar los datos del escáner que mostraba los resultados por fichero de texto en la base de datos. Una vez hecho, me recomendó crear una tabla de resultados de tipo clave/valor para almacenar los datos de ambos escáneres en un mismo formato, además, esta tabla también mostraba qué escáner era el que había detectado los diferentes datos. El desarrollo de esta funcionalidad supuso una complicación importante, pero no se podía desechar ninguno de los dos escáneres ya que estaban obteniendo unos resultados mucho mejores de los esperados.

Una vez desarrollada la aplicación en local, había que llevarla a un servidor externo. Para ello, Juanan me prestó uno de sus servidores para que pudiera corregir todos los errores que se podrían ocasionar en un servidor externo, los cuales fueron más de los esperados, pero que se solventaron relativamente rápido.

Otra cosa a destacar sobre el desarrollo de la aplicación es el diseño de las interfaces, como se ha podido observar no se realizó ningún prototipo del diseño de las interfaces al comienzo del desarrollo del TFG. Por lo tanto se diseñó una interfaz que satisficiera todas la necesidades para las funcionalidades del proyecto, pero que visualmente no era muy atractiva. Por ello,

se decidió consultar con una diseñadora gráfica que, casualmente, era amiga mía, y se llama Meritxel Esteban Díez. Meritxel me dio varias ideas de acuerdo a las necesidades que debía solventar para que el cliente no tuviera que visualizar demasiadas pantallas a la hora de recorrer la aplicación, y para que visualmente se viera en un sitio que inspirara confianza, tranquilidad y seguridad. Por ello, me recomendó una paleta de colores para añadir al diseño y una distribución de los elementos ordenada y visualmente atractiva.

Finalmente, y debido a que fue una necesidad que se detectó prácticamente al final del desarrollo del proyecto, Juanan se dio cuenta de que cualquiera podía consultar los datos que se recogían por los escáneres, y esto había que solucionarlo implementando un sistema de verificación de páginas web que ha sido utilizado por Google. Dicho sistema consiste pedir al usuario que cree un contenido y una url que la aplicación le solicita, dicho contenido es aleatorio, de esta forma, si el usuario ha sido capaz de crear la url con ese contenido, no hay duda de que es el dueño de la página web de la que quiere ver la información.

9. Pruebas

En este capítulo se muestran las pruebas que se han realizado para la comprobación del correcto funcionamiento de las herramientas desarrolladas durante el proyecto. Hay que comentar que por falta de tiempo, las pruebas no se han podido automatizar.

9.1. Pruebas de la verificación de las páginas web

En este apartado se muestran las pruebas que se han realizado para comprobar que la verificación de las páginas web funciona correctamente. Para realizar la comprobación, se intentaron automatizar las pruebas existentes sin éxito. Esto se debe a que el usuario ha de estar registrado y, además, poseer páginas web para poder crear el contenido y la url que se le requieren. Las pruebas que se han podido hacer han tenido que ser con la propia página de EuskalCrawler, ya que yo era el dueño de esa página, y para hacer más pruebas he tenido que hacer que las demás páginas han sido verificadas por mi modificando la base de datos que guarda esa información. Además tampoco ha sobrado demasiado tiempo como para desarrollarlo antes de que termine el plazo de entrega del proyecto.

Envío de la verificación
Descripción de la prueba:Al intentar escanear una página, se ha de enviar una verificación al usuario para que verifique que él es el dueño de la página.
Salida esperada: Vista de la verificación con la URL y el contenido requerido para que se pueda llevar a cabo con éxito.
Resultado obtenido: Éxito.

Verificación sólo con el contenido de la página
Descripción de la prueba: Al intentar verificar la página, el usuario sólo crea el contenido requerido, pero no la URL.
Salida esperada: Vista de la verificación con la URL y el contenido requerido para que se pueda llevar a cabo con éxito.
Resultado obtenido: Éxito.

Verificación sólo con la URL
Descripción de la prueba: Al intentar verificar la página, el usuario sólo crea la URL, sin el contenido requerido.
Salida esperada: Vista de la verificación con la URL y el contenido requerido para que se pueda llevar a cabo con éxito.
Resultado obtenido: Éxito.

Verificación con la URL y el contenido creado por otro usuario
Descripción de la prueba: El usuario intenta escanear una página ya verificada, pero por un usuario diferente, no por él.
Salida esperada: Vista de la verificación con la URL y el contenido requerido para que se pueda llevar a cabo con éxito.
Resultado obtenido: Éxito.

Verificación con la URL y el contenido creado por el usuario
Descripción de la prueba: El usuario crea la verificación con el contenido y la URL Requeridas.
Salida esperada: Vista de los datos que se han escaneado de esa página, además del registro en la base de datos de la verificación por ese usuario.
Resultado obtenido: Éxito.

9.2. Pruebas de la normalización de los datos

En este apartado se muestran las pruebas que se han realizado para comprobar que los datos se normalizan correctamente. Tampoco se han podido automatizar las pruebas.

Normalización de datos nuevos, no existentes
Descripción de la prueba: El script realizará todos los *inserts* correspondientes en la base de datos sin duplicados de los datos nuevos, es decir, que no existían previamente.
Salida esperada: Base de datos sin problemas de integridad,concurrencia ni recurrencia en los datos.
Resultado obtenido: Éxito.

Normalización de datos ya existentes
Descripción de la prueba: El script realizará todos los *updates* correspondientes en la base de datos sin duplicar los existentes.
Salida esperada: Base de datos sin problemas de integridad, concurrencia ni recurrencia en los datos.
Resultado obtenido: Éxito.

9.3. Pruebas del registro

En este apartado se muestran las pruebas que se han realizado para comprobar que los usuarios se pueden registrar en la aplicación sin problemas y que se les envía los mensajes de confirmación de las cuentas. Tampoco se han podido automatizar las pruebas.

Registro de usuarios
Descripción de la prueba: Los usuarios son registrados y sus datos son guardados en la base de datos.
Salida esperada: Base de datos con los datos de los usuarios
Resultado obtenido: Éxito.

Usuario ya registrado se intenta registrar
Descripción de la prueba: Los usuarios intentan registrarse con emails ya confirmados y duplicados
Salida esperada: Mensaje informativo acerca de la incidencia
Resultado obtenido: Éxito.

Usuario se registra correctamente
Descripción de la prueba: La aplicación le debe enviar un mensaje para confirmar su cuenta
Salida esperada: correo electrónico en la cuenta del usuario con un link de confirmación.
Resultado obtenido: Éxito.

Usuario se registra correctamente
Descripción de la prueba: La aplicación le debe enviar un mensaje para confirmar su cuenta
Salida esperada: correo electrónico en la cuenta del usuario con un link de confirmación.
Resultado obtenido: Éxito.

9.4. Pruebas de la identificación

En este apartado se muestran las pruebas que se han realizado para comprobar que los usuarios se pueden identificar en la aplicación sin problemas y con seguridad. Tampoco se han podido automatizar las pruebas.

Identificación de un usuario correcta
Descripción de la prueba:Los usuarios intentan identificarse con datos correctos
Salida esperada: Página principal de la aplicación con el usuario identificado y con acceso a todas las funcionalidades.
Resultado obtenido: Éxito.

Identificación de un usuario incorrecta
Descripción de la prueba: Los usuarios intentan identificarse con datos erróneos
Salida esperada: Mensaje informativo acerca de la incidencia
Resultado obtenido: Éxito.

9.5. Pruebas del escaneo de las páginas webs

En este apartado se muestran las pruebas que se han realizado para comprobar que las páginas son escaneadas y sus datos almacenados en la base de datos. Tampoco se han podido automatizar las pruebas.

Escaneo de una página web existente
Descripción de la prueba:Se escanea una página web real y se recogen sus datos en la base de datos con la fecha del escaneo
Salida esperada: base de datos actualizada
Resultado obtenido: Éxito.

Escaneo de una página web no existente
Descripción de la prueba: Se intenta escanear una página que no existe.
Salida esperada: Error de HTTP 404
Resultado obtenido: Éxito.

10. Conclusiones

En este último capítulo, se va realizar un análisis entre la planificación que se estimó al comienzo del proyecto y cómo ha acabado realmente. También se van a plantear las posibles lineas futuras para el proyecto y, por último, una pequeña reflexión personal.

10.1. Análisis entre planificación estimada y real

A continuación, se exponen los aspectos que no se tuvieron en cuenta a la hora de planificar el proyecto y que han repercutido severamente en el desarrollo de este.

10.1.1. Objetivos

Al tener un proyecto antecedente, los objetivos finales han estado claros desde el principio, aunque durante el desarrollo del mismo se han detectado objetivos, como la verificación de las páginas web, que no se habían pensado desde el principio, pero que las necesidades de los usuarios han hecho que sean objetivos principales.

10.1.2. Herramientas utilizadas

Además de las herramientas mencionadas en el alcance (Sección 3.2), también se han usado las siguientes herramientas:

- **Cron**: Es un administrador regular de procesos en segundo plano en los sistemas operativos Unix. Se utiliza para ejecutar procesos o guiones en intervalos regulares. Este administrador de procesos es utilizado para escanear las páginas web de forma periódica. También llama al fichero que se encarga de la conversión de las grabaciones de voz periódicamente.

- **Requests**: Es una herramienta de Python que se utiliza para verificar urls y contenidos de las mismas, y ha resultado muy útil para la verificación de las páginas web que se querían utilizar.

10.1.3. Alcance

En cuanto al alcance, los bloques de tareas principales que aparecen en la Figura 3 se han mantenido iguales. Pero, dentro de cada bloque si se han tenido que añadir nuevos bloques debido a las nuevas funcionalidades que se identificaron durante el desarrollo.

Aprendizaje

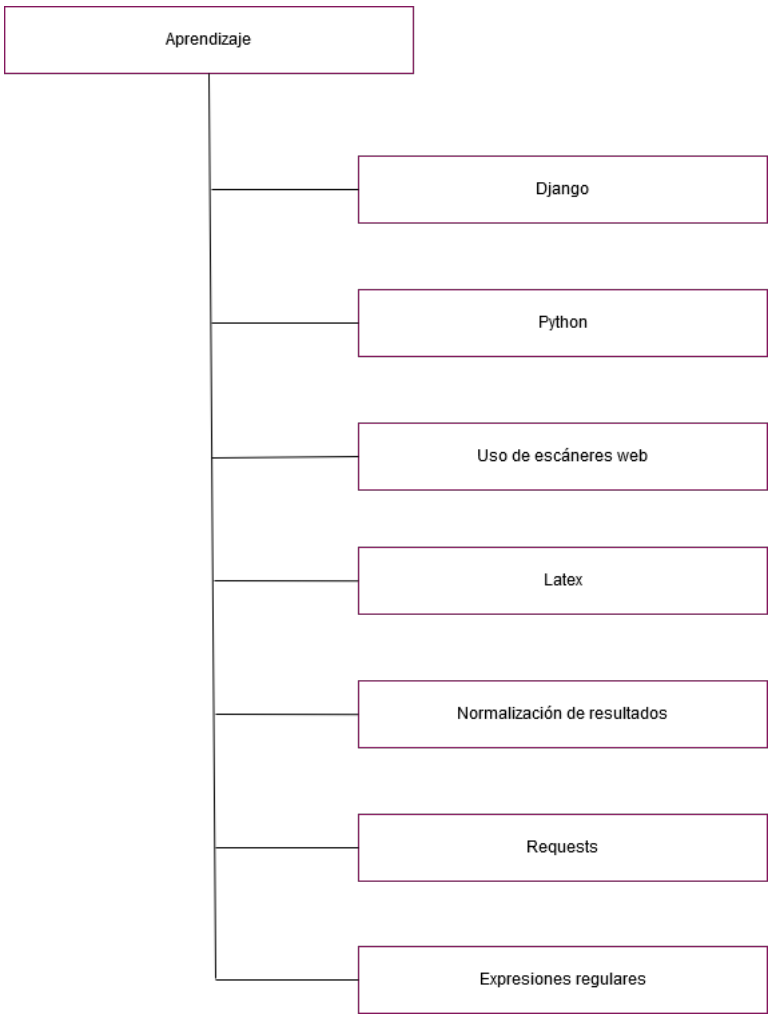

Figura 19: Diagrama EDT final del bloque de aprendizaje.

En este bloque se han añadido las tareas de aprendizaje de la librería de requests, la normalización de los datos y el uso de expresiones regulares.

Todas estas funcionalidades nuevas han sido orientadas a la seguridad de la aplicación, ya que se ha considerado que se estaban manejando datos bastante interesantes como para que la seguridad perjudicara a la protección de los datos de los usuarios.

Aprendizaje de Requests
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje del uso de la librería requests.
Salidas/Entregables: N/A.
Recursos necesarios: Python y Urls con contenido para verificar.

Aprendizaje de Expresiones regulares
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje del uso de expresiones regulares.
Salidas/Entregables: N/A.
Recursos necesarios: Python

Aprendizaje de normalización de datos
Paquete de trabajo: Aprendizaje.
Descripción: Aprendizaje de cómo normalizar resultados que están en formatos muy diferentes.
Salidas/Entregables: N/A.
Recursos necesarios: Python y una base de datos.

Organización

En este apartado, el tiempo utilizado ha sido mayor al estimado debido a la dificultad que supuso encontrar escáneres adecuados para poder obtener datos relevantes y correctos para los usuarios. Además, el haber tenido que reinstalar algunas de las herramientas por haber tenido que tocar archivos propios de ellas y haber afectado a las funcionalidades de la misma, también ha hecho que se aumente el tiempo utilizado en las tareas de este bloque.

Captura de requisitos

En la captura de requisitos se ha recortado tiempo respecto al estimado. Esto se debe a que se pensó que se tardaría más tiempo del que realmente se ha tardado en realizar los prototipos.

Análisis y diseño

En este bloque, se ha tardado más de lo esperado. Cada vez que aparecía una nueva funcionalidad o algún cambio en lo ya existente, había que modificar el diseño de la interfaz de la aplicación.

Implementación y desarrollo

Este bloque, al haberse identificado las nuevas funcionalidades en un momento relativamente cercano al inicial, no ha tenido grandes modificaciones a la hora de identificar las tareas que se iban a realizar por lo que, en la Figura 8, se muestran los apartados sin modificaciones.

Pruebas

Al no haber podido automatizar las pruebas por no tener muchas páginas web para escanear que fueran propias y por falta de tiempo, en este apartado se ha tardado menos tiempo del estimado.

Documentación

Debido a que la documentación se realizó completamente después de la implementación de todo el código y de la aplicación, no se ha visto modificada su planificación inicial y el tiempo de realización ha sido el estimado

Duración de las tareas

Por todos los cambios comentados en esta sección, la duración real a sobrepasado con creces lo que se estimó en la planificación inicial del proyecto.

Por las modificaciones y los problemas que se encontraron durante la evaluación de la herramienta, la duración de algunas de las tareas llevaron más de lo previsto. En las Tablas 4 y 5, se muestran todas las tareas, la estimación que se hizo en la planificación inicial y lo que ha durado realmente cada tarea.

Como se puede comprobar en las tablas, en el único bloque en el que se ha recortado tiempo ha sido en el de pruebas por no haber tenido tiempo suficiente para automatizarlas. En casi todos los demás bloques el tiempo se ha ajustado al planificado, salvo en organización, dónde se ha incrementado de forma notable, el proyecto en vez de durar las 382 para terminar, ha llevado 460 horas.Estas horas que se han estimado son sin tener en cuenta el tiempo en el que el proyecto estuvo parado debido a la discontinuidad del mismo. Dicho tiempo fue más o menos un año, debido a las asignaturas pendientes

ID	APRENDIZAJE	PREDECESORAS	DURACION ESTIMADA
1	Python	-	30
2	Django	1	50
3	Uso de escáneres web	-	4
4	Latex	-	30
5	Requests	1	4
6	Normalización de datos	1,3	12
7	Expresiones regulares	1	4
	ORGANIZACIÓN		
5	Planteamiento de la aplicación		4
6	Planificación de las tareas	5	6
7	Elección de software	6	30
8	Instalación de software	7	50
9	Reuniones con el director del proyecto	5	30
	CAPTURA DE REQUISITOS		
10	Casos de uso	5	2
11	Modelo de dominio	10	2
	ANALISIS Y DISEÑO		
12	Diagramas de clases		4
13	Diagramas de secuencia		4
14	Diagramas de estados		2

Tabla 4: Dependencias y duración de las tareas finales(1).

y a que me quedó una asignatura, la cual ya iba por la quinta convocatoria y cuya prioridad era absoluta, ya que no quería llegar a la sexta y, mucho menos, pedir la compensatoria para que se me aprobara, ya que era la única asignatura que me quedaba pendiente. Si a esto le sumamos los diferentes problemas personales que me surgieron, el proyecto estuvo parado mucho más tiempo del deseado y esto provocó un gran retraso en la finalización del mismo.

10.1.4. Evaluación económica

En cuanto a la evaluación económica, solo hay que tener en cuenta que el número de horas trabajadas ha sido mayor al previsto. Es por ello que hay que volver a realizar los cálculos del gasto en la mano de obra.

Como los cálculos de cuanto se paga por horas a un programador informático ya se tienen en la valoración económica (Sección 3.6, solo queda cambiar el número de horas por las que se han tardado realmente. Para ello, primero se van a restar las horas que realmente se han usado en el bloque de aprendizaje:

	IMPLEMENTACIÓN Y DESARROLLO		66
15	Registro de usuarios	-	6
16	Log-in	15	6
17	Verificación de las empresas	-	12
18	Escaneos de las webs	17	30
19	Normalización de datos	17,18	12
	PRUEBAS		26
20	Pruebas del escaneo	18,17	2
21	Pruebas de la normalización de datos	17,18,19,20	4
22	Pruebas del registro de usuarios	15	2
23	Pruebas del log-in	16	2
24	Pruebas de la verificación de las empresas	17	6
	DOCUMENTACIÓN		100
	Documentación del código		10
25	Comentar código	15,16,17,18,19	4
26	Redacción de manuales	15,16,17,18,19	6
	Memoria del Proyecto		90
27	Búsqueda del material de apoyo	5	5
28	Creación de tablas y diagramas	27	10
29	Redacción de la memoria	27,28	50
30	Preparación de la defensa	29	25
	TOTAL		460

Tabla 5: Dependencias y duración de las tareas finales (2).

$$Horas\ totales - Horas\ aprendizaje = 460 - 134 = 326\ horas \quad (9)$$

Multiplicando el resultado por el sueldo/hora que se calculó en la evaluación económica, se obtiene el siguiente resultado:

$$Sueldo = Horas\ trabajo * Sueldo/hora = 326 * 8,98 = 2927,48 \quad (10)$$

Con esto tenemos que el gasto real en mano de obra sería de 2927,48 €. Con este nuevo resultado, en la Tabla 6 se muestran los costes finales del proyecto.

Tipo de gasto	***Gasto estimado (en €)***	***Gasto real (en €)***
Mano de obra	**2604,64 €**	**2927,48 €**
Gasto en *software*	**110 €**	**110 €**
Gasto en *hardware*	**153,36 €**	**153,36 €**
Gastos indirectos	**150 €**	**150 €**
Total	**4.140,06 €**	**4.921,32 €**

Tabla 6: Costes finales del proyecto

10.2. Lineas futuras

Este proyecto ofrece una posibilidad de mejora prácticamente hasta donde se quiera llegar, ya que se pueden añadir más escáneres o actualizar los ya existentes, por ello, el código desarrollado en este proyecto está preparado para ser reciclado en caso de que se quieran añadir modificaciones o mejoras. El número de escáneres utilizados es 2, pero se pueden llegar a utilizar más, y, en vez de mostrar todos los resultados escaneados por todos los escáneres, se podrían filtrar y obtener sólo los más relevantes, hacer mecanismos de votación para extraer la información más correcta o cualquier tipo de funcionalidad que mejoren la eficiencia en la presentación de la información.

10.3. Reflexión personal

Para concluir con mi memoria del TFG, quiero comentar algunos aspectos que me han hecho disfrutar de este proyecto.

Uno de los aspectos más "frustrantes" de este proyecto fue, que lo empece sin conocimiento alguno de lo que iba a hacer. Mi única experiencia de gran parte de lo que he realizado, era el haber implementado una página web en uno de los proyectos de la asignatura de seguridad que cursé en tercer curso del grado. Todo lo realizado en este proyecto, ha sido a base de un aprendizaje continuo del que me siento orgulloso. He aprendido a usar muchas herramientas, he investigado otras que, aunque no haya llegado a usar, ya tengo conocimientos de ellas y, sobre todo, aun habiendo empezado todo el proyecto desde cero, he conseguido llegar hasta este punto en menos de un año.

Una de las experiencias que me llevo, es la de haber creado algo que, durante su desarrollo, no sólo me ha enseñado a escanear las páginas web, sino que también me ha enseñado por qué se han de proteger los datos y cómo protegerlos, además de qué datos son los más vulnerables.

Por todo ello, he de dar las gracias a Juanan, por dejarme tener esta experiencia y por ayudarme con los problemas que iban surgiendo durante el proyecto.

Bibliografía

[Angulo, 2016] Angulo, D. L. (2016). Euskalcrawler, memoria del proyecto. `https://addi.ehu.es/bitstream/handle/10810/19678/EuskalCrawler.pdf?sequence=2&isAllowed=y`.

[Azzopardi, 2016] Azzopardi, L. (2016). Learning django. In *Tango with Django. A beggineer guide to web development with Django 1.9*, pages 183–260.

11. Direcciones Webs utilizadas

- `https://whatweb.net/`:Para obtener información sobre el escáner de WhatWeb y poder descargarlo.
- `http://blindelephant.sourceforge.net/`:Para conocer uno de los escáneres web más conocidos y utilizados.
- `https://github.com/poerschke/Uniscan`:Para obtener información sobre el escáner de Uniscan y poder descargarlo.
- `https://platzi.com/blog/expresiones-regulares-python/`: para saber como se pueden hacer expresiones regulares en Python. Fue útil para la normalización de resultados.
- `http://blog.contraslash.com/registro-de-usuarios-y-confirmacion-por-email-en-d/`: Para saber cómo se pude hacer en Django la confirmación de emails.
- `https://sendgrid.com/`: Para descargar un servicio para el envío de emails para la confirmación.
- `https://stackoverflow.com/questions/30547580/django-update-models-py-when-database-changes`: Para modificar los archivos de Django para mantener la concurrencia con la base de datos.
- `https://stackoverflow.com/questions/1985383/update-django-database-to-reflect-changes-in-existing-models`:Para modificar la base de datos para mantener la concurrencia con los archivos de Django.
- `https://en.wikibooks.org/wiki/LaTeX/Mathematics`: Para conocer las diferentes expresiones de LaTeX para poder hacer ecuaciones
- `https://stackoverflow.com/questions/15138614/how-can-i-read-the-contents-of-an-url-with-python`: Se consultó para poder hacer la verificación de las empresas y las comprobaciones.
- `https://stackoverflow.com/questions/1905470/cannot-delete-or-update-a-parent-row-a-foreign-key-constraint-fails`: Cuando insertaba datos en la base de datos, había veces que no me dejaba borrarlos, debido a las dependencias de las claves extranjeras que se generaban.F

A. Anexo 1: Manual de instalación

A.1. Instalación de Django

Para este paso hay que tener instalado python. Una vez hecho, hay que ejecutar los siguientes comandos para instalar Django:

$pip install -U django==1.9.5 $pip install pillow $pip install –upgrade pip

Y crear un nuevo proyecto:

$django-admin.py startproject aplicacion

Dentro de esa carpeta que se nos ha creado llamada aplicación, hay que copiar el contenido de la carpeta euskalcrawler que se ha descargado de git (el contenido, no la carpeta), es decir:

$cd pathDelProyecto $cp -a euskalcrawler /path/de/aplicacion

A.2. Instalación de python y sus librerías

Lo primero es instalar python en la versión 2.7, para ello sólo hay que escribir los siguientes comandos:

$apt-get install python2.7 Los siguientes comandos nos permitirán poder utilizar MySQL

$sudo apt-get install libmysqlclient-dev $sudo apt-get install python-mysqldb

Y el siguiente comando son las librerías de python que necesitaremos para que todo funcione.

$sudo pip install requests $sudo pip install bcrypt

A.3. Instalación del proyecto

Lo primero es instalar git, lo cual también será necesario para la instalación de Uniscan.Para ello se abrirá la terminal de Linux y se introducirá el siguiente comando:

$sudo apt-get install git

Después hay que clonar el repositorio de git de la aplicación en su PC, para ello:

$git clone http://www.urlDeLaAplicación.fin

A.4. Instalación de Uniscan

Al instalar el proyecto, ya se habrá clonado también una carpeta llamada Uniscan. Para instalar Uniscan sólo se deberá colocar en la carpeta de Uniscan que está dentro del proyecto euskalcrawler con el siguiente comando:

$cd /pathDelProyecto/Uniscan

Una vez dentro sólo se debe ejecutar el siguiente comando, que instalará los módulos necesarios para que Uniscan funcione correctamente:

$./install-modules.sh -ejecutamos el archivo que tiene ese nombre $sudo apt-get install libmoose-perl -se recomienda si se tiene ubuntu 13.04, ya que si no dará un error de que falta ese paquete.

Si se quiere comprobar que todo funciona correctamente, o simplemente se quiere probar Uniscan, ejecutamos el siguiente comando, que hace que Uniscan escanee una url (se debe de estar colocado dentro de la carpeta de Uniscan).

uniscan -u http://objetivo -qweds -tardará un rato pero ya se estará escaneando la url objetivo

A.5. Instalación de Whatweb

Para ello lo primero es instalar Ruby, ya que whatweb requiere de dependencias con este software:

¡IMPORTANTE!: Whatweb no es compatible con Ruby 1.9.

sudo apt-get install ruby ruby-dev libopenssl-ruby

Y también se recomienda instalar las diferentes opciones para que whatweb guarde los resultados:

$sudo gem install json $sudo gem install bson $sudo gem install bson_ext $sudo gem install mongo $sudo gem install rchardet

Una vez instalado Ruby, sólo queda instalar Whatweb:

sudo apt-get install whatweb

Si se quiere probar whatweb, y que todo funciona correctamente:

whatweb http://objetivo

A.6. Instalación de la base de datos

Para este paso se recomienda usar el gestor de base de datos llamado MySQLWorkbench.

Lo primero es importar la base de datos que hay en git a MySQL.

Ahora hay que configurar una conexión para esa base de datos e indicarla en el fichero de configuración de euskalCrawler/configuration.py, indicando el nombre de usuario, el servidor, la contraseña, el nombre de la base de datos donde debe conectarse para que todo funcione correctamente.

También habrá que indicarle el path de la carpeta results para que pueda llevar los resultados al archivo que hay dentro.

Dentro de la carpeta Uniscan hay otro configuration.py, hay que hacer lo mismo solo que, además, debemos indicar también cuál es el path de Uniscan.

Por último, hay que ir al archivo settings.py de Django e indicarle ahí los datos de la base de datos, el archivo se encuentra dentro de la carpeta euskalcrawler.

A.7. Preparación de la confirmación de correos

En settings.py escribir la configuración del servidor smtp en la variable que se llama así.

En views.py cambiar localhost por la pagina real de la pagina y changeme@hotmail.com por el correo real que enviará los emails de confirmación.

A.8. Lanzamiento del servidor

Todo listo. Para lanzar el servidor sólo hay que ejecutar el siguiente comando:

$ python manage.py runserver 0.0.0.0:8000