UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG

Presentado por Nombre del alumno
en Universidad de Burgos — 31 de marzo de 2025
Tutor: nombre tutor

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Nombre del alumno, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 31 de marzo de 2025

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Descripción del contenido del trabajo y del estructura de la memoria y del resto de materiales entregados.

2. Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

3. Conceptos teóricos

En esta sección se definen aquellos conceptos que son necesarios conocer para comprender el resto del documento.

Clasificador

Un clasificador es un modelo o algoritmo que se utiliza para asignar una categoría o etiqueta a un conjunto de datos, basándose en sus características. En el contexto de la inteligencia artificial, el proceso de clasificación implica entrenar un modelo para que aprenda a predecir la clase o categoría correcta de nuevas observaciones, basándose en ejemplos previos conocidos como datos de entrenamiento.

Red neuronal (*Neural Network*)

Una red neuronal artificial es un modelo computacional inspirado en el funcionamiento de las redes neuronales biológicas y el cerebro humano. Este tipo de modelos suponen la base del aprendizaje automático (*deep learning / machine learning*) y han transformado radicalmente el campo de la inteligencia artificial en los últimos años debido a su capacidad de reconocer patrones complejos en grandes conjuntos de datos.

Internamente, una red neuronal se compone de capas interconectadas de unidades de procesamiento llamadas neuronas artificiales o perceptrones. Cada neurona puede recibir múltiples señales de entrada, aplicando sobre estas una transformación matemática que las combina, la forma de combinar dichas entradas depende de la función específica usada y un conjunto de pesos ajustables, posteriormente, usando la entrada combinada, se genera una salida a través de otra función matemática no lineal llamada, función

de activación. Dependiendo de la aplicación que se le quiera dar al modelo, existen diferentes tipos de redes neuronales que se ajustan mejor a unas tareas u otras. Los tipos principales son: redes *feedforward* para propósitos generales, redes recurrentes (RNN) para el procesamiento de datos secuenciales, redes convolucionales (CNN) para procesamiento de imágenes y visión por ordenador, o, recientemente, modelos de tipo *Transformer* para el procesamiento del lenguaje natural.

La forma en la cual la red aprende se lleva a cabo mediante un algoritmo llamado *backpropagation*, el cual ajusta los parámetros de las diferentes neuronas de manera iterativa para reducir una función de pérdida que mide el error de predicción. Este proceso de ajuste de los pesos se realiza mediante el algoritmo del descenso de gradiente y variaciones de este que intentan predecir el mejor ajuste para reducir la función de pérdida.

Las aplicaciones de este tipo de modelos se han extendido enormemente en los últimos años y hoy en día se aplican en infinidad de campos para la realización de tareas como el procesamiento y la generación del lenguaje natural, el análisis de imágenes médicas, los sistemas de recomendación, la detección de amenazas cibernéticas y muchas otras.

Tokenización

Los modelos de inteligencia artificial, en su más puro estado, son modelos matemáticos complejos los cuales consiguen aprender patrones de sus datos de entrada y permiten realizar posteriormente predicciones en base a dichos patrones.

El problema fundamental que esto presenta a la hora de procesar diferentes tipos de datos es principalmente que los modelos matemáticos se basan en el uso de números y la búsqueda de patrones en datos numéricos. Esto implica que si usamos datos que no sean numéricos, como puede ser el texto, caemos en un grave problema puesto que no podemos simplemente aplicar el modelo a este tipo de dato directamente puesto no está preparado para funcionar con el.

Una solución a este problema es emplear el proceso de tokenización, el cual consiste en romper el texto que se obtiene como entrada en unidades más sencillas llamadas tokens que el modelo pueda llegar a entender posteriormente. Este proceso puede realizarse de diferentes manearas, ya sea partiendo cada palabra en un token individual, partiendo las palabras en subpalabras o incluso procesando cada carácter de manera independiente.

El proceso de tokenización parece bastante trivial a primera vista pero presenta muchos retos como pueden ser los siguientes. Procesar diferentes idiomas se vuelve rápidamente un problema en función del método que se utilice para obtener los tokens, diferentes idiomas permiten contracciones o expresiones especiales las cuales pueden ser interpretadas de diferentes maneras y han de poder ser divididas en tokens que mantengan dicho significado. También es posible que un texto presente errores gramaticales, símbolos especiales o jerga específica de un campo que ha de ser entendida y tokenizada correctamente para mantener su significado.

Vocabulario

Para que un modelo pueda entender los tokens que se obtienen en el paso de la tokenización, es necesario convertir dichos tokens en números de alguna forma. Esto puede realizarse de forma sencilla mediante el uso de un diccionario o vocabulario para el modelo.

El vocabulario del modelo no es más que la colección de todos los tokens que se pretende que el modelo pueda comprender asociados a un valor numérico que siempre será el mismo. De esta forma, el texto de entrada puede ser dividido en tokens, los cuales, a su vez, pueden ser convertidos siempre en el mismo valor numérico mediante el uso de un diccionario. Permitiendo así que el modelo procese texto y pueda encontrar patrones en este.

De forma similar al proceso de la tokenización, la creación y selección de un vocabulario presenta muchos problemas que pueden no ser tan aparentes a primera vista, algunos de ellos son, la extensión del diccionario y el número de tokens que son necesarios en este para poder procesar diferentes idiomas o, la necesidad de poder incluir valores de control que puedan ser usados en casos en los que un token no exista o se le quiera dar una información especial a ciertos tokens en algunas circunstancias.

Framework web

Un *framework* web es un conjunto de herramientas, bibliotecas y componentes predefinidos que facilitan el desarrollo de aplicaciones y páginas web. Su propósito es ofrecer una estructura básica que los desarrolladores pueden utilizar para crear aplicaciones de manera más eficiente, sin tener que comenzar desde cero. Los *frameworks* web incluyen funcionalidades que resuelven tareas comunes, como el manejo de rutas (URLs), la conexión a

bases de datos, la autenticación de usuarios y la seguridad, lo que ahorra tiempo y esfuerzo durante el desarrollo.

Además, proporcionan una organización estructurada para el código, lo que facilita la colaboración entre desarrolladores y simplifica el mantenimiento de los proyectos a largo plazo. Muchos *frameworks* también incluyen mecanismos de seguridad incorporados para proteger las aplicaciones contra amenazas comunes y herramientas que automatizan tareas repetitivas, como la gestión de dependencias y la ejecución de pruebas. Todo esto permite construir aplicaciones web escalables, seguras y fáciles de mantener.

Algunos *frameworks* web populares son: Django (para Python), Ruby on Rails (para Ruby), Angular y React (para JavaScript), y Laravel (para PHP).

Malware

El término *malware*, (proveniente de *malicious software*) o software malicioso, se refiere a cualquier programa, código o script diseñado con el propósito explícito de causar daño, comprometer la seguridad, o realizar actividades no autorizadas en un sistema informático, una red o un dispositivo. Coloquialmente, se utiliza el término *malware* para referirse a una amplia gama de diferentes subcategorías de programas que puedan considerarse dañinos, cada una con características, formas de ataque y objetivos específicos, pero todas compartiendo la intención de perjudicar al usuario, robar información, o tomar el control de aquellos sistemas afectados.

Una clasificación bastante común del *malware* es la siguiente:

Virus

Un virus es considerado el comodín de los programas maliciosos, al menos desde un punto de vista coloquial, ambas palabras son intercambiables, en cambio, desde un punto de vista técnico, la definición de un virus informático es equiparable a su equivalente biológico. Un virus es un tipo de *malware* que requiere de un huésped para poder realizar su función, es decir, suele ir incrustado o adjunto a otros archivos, generalmente, ejecutables o documentos los cuales, una vez abiertos, permiten al virus infectar el sistema y realizar acciones no deseadas. Una característica importante de los virus, es el hecho de que pueden auto replicarse, es decir, una vez infectada una máquina, pueden emplear diferentes métodos y técnicas para propagarse a otras, comúnmente, estas suelen ser mediante el uso de la red o mediante el uso de medios físicos extraibles. Los virus suelen ser detectados mediante

firmas específicas, aunque las técnicas de ofuscación y polimorfismo pueden dificultar su identificación.

Gusano (*worm*)

Un gusano es un tipo de *malware* diseñado para propagarse automáticamente a través de las redes informáticas, explotando las vulnerabilidades de los diferentes sistemas o utilizando técnicas de ingeniería social para engañar a los usuarios. A diferencia de los virus, los gusanos no necesitan de un huésped al cual adjuntarse para poder desempeñar su función y para auto replicarse, puesto que pueden propagarse de manera independiente. Su principal objetivo es infectar tantos dispositivos como sea posible, lo que puede resultar en la saturación de las redes, la degradación del rendimiento del sistema, o la creación de puertas traseras para permitir el paso a otros tipos de *malware*. Los gusanos son particularmente peligrosos en entornos corporativos, donde pueden propagarse rápidamente a través de las redes internas de una empresa o una organización.

Troyano (*trojan*)

Un troyano es un tipo de *malware* que se hace pasar por un *software* legítimo o útil para engañar a los usuarios y lograr su ejecución. Una vez instalado, un troyano permite a un atacante acceder o controlar el sistema infectado de manera remota, sin el conocimiento del usuario. Los troyanos no se replican por sí mismos, pero pueden realizar gran variedad de acciones maliciosas, principalmente, robar información confidencial, instalar otros tipos de *malware*, o convertir el sistema en parte de una *botnet*. Su nombre proviene del mito del Caballo de Troya, perteneciente a la mitología griega, ya que, al igual que sucede en la historia, el troyano parece inofensivo en un principio pero oculta una gran amenaza en su interior.

Ransomware

El *ransomware* es un tipo de *malware* diseñado para cifrar los archivos del usuario o bloquear el acceso al sistema, exigiendo un rescate, *ransom* en inglés y generalmente en criptomonedas, a cambio de restaurar el acceso. Este tipo de *malware* ha ganado mucha popularidad en los últimos años debido a su impacto devastador en individuos, empresas e incluso instituciones gubernamentales. El *ransomware* suele propagarse a través de correos electrónicos de *phishing*, descargas maliciosas o *exploits* de vulnerabilidades. Una vez este es activado, el comportamiento más típico consiste en mostrar al usuario un mensaje (*ransom note*) con las instrucciones para pagar el

rescate. Es importante destacar que, aunque se pague la tasa correspondiente al rescate dentro del tiempo establecido, no hay ninguna garantía de que los atacantes cumplan con su promesa de desbloquear los archivos. Es por ello, por lo que, este tipo de *malware* es extremadamente peligroso puesto que juega con el factor de perder un recurso muy preciado, como puede ser la información, además de utilizar la desesperación de los usuarios en su contra.

Spyware o *Info stealer*

El *spyware* es un tipo de *malware* diseñado para recopilar información del usuario sin su consentimiento. Esta información puede incluir contraseñas, datos bancarios, historiales de navegación, métodos de entrada (como pueden ser las pulsaciones de las teclas de un teclado) o cualquier otro dato sensible que pueda posteriormente ser vendido o utilizado en contra de los usuarios. El *spyware* suele operar de manera sigilosa, sin mostrar signos evidentes de su presencia, lo que dificulta su detección.

En términos generales, existen dos variantes de *spyware*, la primera de ellas se instala en el sistema de forma permanente y siempre está activa, monitorizando la actividad del usuario de manera constante, mandando cualquier información sensible que este use, vea o teclee a un servidor externo para que los atacantes la puedan utilizar o vender, la única ventaja que presenta esta variante es que deja rastro y es más sencilla de detectar. Por otro lado, la segunda variante simplemente se ejecuta una vez, recopila toda la información que pueda y luego se borra a si misma para no dejar ni el más mínimo rastro de su ejecución. Para muchos, esta segunda variante es considerada incluso más peligrosa que la primera puesto que la víctima puede tardar semanas, meses o incluso años en darse cuenta de que su información a sido comprometida.

Además de robar información, algunos tipos de *spyware* pueden modificar la configuración del sistema, instalando *software* adicional o redirigiendo el tráfico de red, generalmente, para evitar su detección. Este tipo de *malware* es comúnmente distribuido a través de descargas no autorizadas, correos electrónicos de *phishing*, o *software* gratuito (*freeware*) que incluye componentes ocultos.

Adware

El *adware* es un tipo de *software* que muestra publicidad no deseada, a menudo de manera intrusiva, en el dispositivo del usuario. Aunque no

siempre es malicioso, el *adware* puede ser molesto y afectar negativamente a la experiencia del usuario. En algunos casos, el *adware* incluye funcionalidades adicionales para rastrear el comportamiento del usuario y mostrar anuncios personalizados, lo que puede considerarse una violación de la privacidad. El *adware* suele distribuirse junto con *software* gratuito, y los usuarios pueden instalarlo sin darse cuenta al aceptar los términos y condiciones sin leerlos detenidamente.

PUP (*Potentially Unwanted Program* o Programa Potencialmente no Deseado)

Un PUP (*Potentially Unwanted Program*, por sus siglas en inglés) es un tipo de *software* que, aunque no es necesariamente malicioso, puede ser considerado no deseado por el usuario. Los PUPs incluyen aplicaciones como barras de herramientas (*toolbars*), optimizadores de sistema, o *software* de publicidad que se instalan sin el consentimiento explícito del usuario. Aunque no siempre son dañinos, los PUPs pueden ralentizar el sistema, mostrar anuncios no deseados, o recopilar información acerca del usuario. Muchos antivirus y soluciones de seguridad clasifican los PUPs como una categoría separada de *malware*, ya que su impacto puede variar desde simplemente molesto hasta potencialmente peligroso.

Rootkit

Un *rootkit* es un conjunto de herramientas o *software* diseñado para otorgar a un atacante acceso privilegiado y persistente a un sistema, mientras oculta su presencia tanto del usuario como de cualquier *software* de seguridad que pueda estar presente en el sistema. Los *rootkits* suelen operar a nivel de *kernel* (conocido como Ring 0 en muchos casos) o del sistema operativo, lo que les permite manipular cualquier funcionalidad del sistema, incluso aquellas de las que el usuario posiblemente desconoce puesto que están ocultas por el sistema operativo para facilitar su uso. Esto permite a los *rootkits* ser uno de los *malware* más peligrosos debido a que pueden evadir la gran mayoría de soluciones de seguridad y antivirus puesto que operan con los mismos privilegios que estos o incluso más altos. Una vez instalado, un *rootkit* puede ser utilizado para instalar otros tipos de *malware*, robar información, o convertir el sistema en parte de una *botnet*. Debido a su capacidad para ocultarse, los *rootkits* son particularmente difíciles de detectar y eliminar, y a menudo requieren herramientas especializadas o en muchos casos, la reinstalación completa del sistema para poder deshacerse de ellos.

Botnet

Una *botnet* es una red de dispositivos infectados (llamados *bots* o *zombies*) controlados de manera remota por un atacante, conocido como *botmaster*. Los dispositivos infectados pueden incluir computadoras, servidores, dispositivos IoT, y otros equipos conectados a internet. Las *botnets* son utilizadas para realizar una variedad de tareas maliciosas, como pueden ser, ataques de denegación de servicio distribuido (DDoS), envío masivo de correos no deseados (*spam*), minería de criptomonedas, o robo de información masivo. Los dispositivos infectados suelen ser controlados a través de un servidor externo, y los usuarios generalmente no son conscientes de que su dispositivo forma parte de una *botnet*. La creación y gestión de *botnets* es una de las actividades que más dinero genera para los ciberdelincuentes, ya que les permite llevar a cabo ataques a gran escala con un impacto muy significativo.

Análisis estático

Técnica de detección de *malware* que se realiza sin la necesidad de ejecutar el programa en cuestión. Este método se basa en la obtención, inspección y evaluación de las características que se pueden extraer de un archivo binario, tales como su estructura, código fuente (si está disponible), indicios de obfuscación u otras técnicas de ocultación, cadenas de texto incrustadas en este, firmas digitales, huella digital *hash o signature* del archivo, secuencias de bytes concretas, cabeceras del programa, metadatos incrustados, desensamblado del ejecutable y otras propiedades que pueden ser extraídas directamente del archivo. Las ventajas que este enfoque presenta son, su simplicidad, rapidez y bajo coste computacional, ya que no requiere de entornos de ejecución específicos ni de hardware especializado para probar el comportamineto del programa. Sin embargo, el mayor problema de este tipo de análisis es su dificultad para detectar malware que utiliza técnicas avanzadas de ofuscación o cifrado, ya que estas prácticas dificultan la extracción de información útil del binario.

Análisis dinámico

Técnica de detección de *malware* que consiste en evaluar el comportamiento de un programa mediante su ejecución en un entorno controlado, con el objetivo de observar sus interacciones con el sistema operativo, los recursos de este y otros programas. En este enfoque, se monitorean actividades como la modificación de archivos, el tráfico de red generado, la creación de procesos o la inyección de código en estos, lo cual permite identificar patrones de

comportamiento asociados con programas maliciosos. A diferencia del análisis estático, el análisis dinámico ofrece una mayor precisión, ya que puede detectar comportamientos maliciosos que no son evidentes simplemente escanenado el archivo de manera estática, como el uso de técnicas de ofuscación. Sin embargo, este tipo de análisis sigue teniendo sus incovenientes, por un lado, es más complejo, requiere de más recursos computacionales y es más costoso de implementar, dado que involucra la ejecución real del código en un entorno controlado, generalmente una máquina virtual (*sandbox*). Por otro lado, también es poco eficiente contra casos en los que el *malware* detecta el hecho de que está siendo analizado y oculta su comportamiento malicioso. Además, puede no ser adecuado para dispositivos con recursos limitados, como dispositivos IoT o móviles, debido a sus altos requerimientos de hardware y tiempo.

Análisis híbrido

Metodo de detección de *malware* el cual combina las fortalezas tanto del análisis estático como del dinámico. En este método, el programa se ejecuta en un entorno controlado, y durante su ejecución, se realizan *dumps* de memoria de manera periódica o en respuesta a comportamientos sospechosos. Estos volcados de memoria son posteriormente analizados utilizando técnicas de análisis estático para identificar posibles patrones maliciosos, tales como la inyección de código en procesos ajenos, manipulación de memoria que no le pertenece al programa o modificaciones en partes protegidas de la memoria pertenecientes al sistema oeprativo. Este enfoque permite una detección más precisa de *malware* que utiliza técnicas avanzadas de ocultamiento, ya que combina la observación del comportamiento en tiempo real con la inspección detallada del estado de la memoria. Sin embargo, el análisis híbrido es el más complejo y costoso de implementar, ya que requiere tanto de infraestructura de virtualización como de herramientas para realizar un buen análisis de memoria. A pesar de todo, suele ofrecer los mejores resultados en términos de detección.

Huella digital (*fingerprinting*)

El fingerprinting o, la generación de huellas digitales de archivos, es una técnica utilizada para identificar de manera única un archivo mediante la aplicación de funciones criptográficas de *hashing*. Este proceso consiste en calcular un *hash* a partir del contenido completo del archivo utilizando algoritmos como MD5, SHA-1, SHA-256 u otros. El resultado es una cadena de longitud fija que actúa como un identificador único para ese archivo.

Cualquier modificación, por mínima que sea, en el contenido del archivo resultará en un *hash* completamente diferente, lo que permite detectar alteraciones o corrupciones en estos.

Esta técnica es muy utilizada en la verificación de la integridad de archivos, la detección de duplicados, y la identificación de malware conocido al comparar el *hash* que este genera con una base de datos de muestras previamente catalogadas. Sin embargo, una limitación importante del *fingerprinting* es su sensibilidad extrema a cambios mínimos, lo que dificulta la identificación de archivos que han sido ligeramente modificados pero que conservan una estructura o funcionalidad. Esto implica que incluso cambiar un bit en el *padding* del archivo, hace que este ya no se detecte como malware al tener una huella digital diferente.

Huella digital difusa (*fuzzy hashing*)

El *fuzzy hashing*, o *hashing* difuso, es una técnica que extiende el concepto del *hashing* tradicional al permitir la comparación de archivos basada en similitudes parciales en lugar de una coincidencia exacta. A diferencia del *hashing* convencional, que opera sobre el archivo completo, el *fuzzy hashing* divide el archivo en bloques o segmentos y calcula un *hash* para cada uno de ellos. Este enfoque por bloques permite identificar similitudes entre archivos incluso cuando solo una porción de su contenido ha sido modificada.

El *fuzzy hashing* es particularmente útil en el análisis forense digital y la detección de *malware*, ya que permite identificar variantes de archivos maliciosos que han sido modificaods para evadir su detección, pero que conservan partes significativas de su código original. Al comparar dos *hashes* difusos, es posible calcular un grado de similitud basado en la cantidad de bloques que coinciden entre ambos. Esto se logra mediante algoritmos especializados como SSDeep o TLSH, que están diseñados para generar *hashes* difusos y medir la similitud entre ellos.

3.1. Conceptos de LaTeX

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX [1].

3.2. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.3. Referencias

Las referencias se incluyen en el texto usando cite [1]. Para citar webs, artículos o libros [**?**], si se desean citar más de uno en el mismo lugar [**?**, **?**].

3.4. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

[1] Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

Figura 3.1: Autómata para una expresión vacía

3.5. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.6. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

4. Técnicas y herramientas

En este apartado se comentan y analizan las diferentes herramientas usadas en la realización de este proyecto.

4.1. PyTorch vs Keras

Características	**PyTorch**
Flexibilidad	Alta, permite crear redes neuronales personalizadas y complejas.
Facilidad de uso	Requiere más código y tiene una curva de aprendizaje más pronunciada
Personalización	Excelente para redes personalizadas y modelos avanzados.
Comunidad y soporte	Muy popular en la investigación académica y proyectos avanzados.
Uso principal	Investigación, redes neuronales complejas.

Tabla 4.1: Comparativa entre PyTorch y Keras

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

5. Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

6. Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Wikipedia. Latex — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=LaTeX&oldid=84209252, 2015. [Internet; descargado 30-septiembre-2015].