Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Título del Trabajo Fin de Grado

Alumno: Nombre y Apellidos

Tutor/a/es: Nombres y Apellidos

Título del TFG

Jaime Serrano Casado

Índice general

Índice de figuras

Índice de cuadros

Dedicado a
mi familia

Agradecimientos

Muchas gracias a todos

Resumen

Resumen que aparecerá en UVADOC.

Palabras claves: separadas por comas.

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

Se tratará brevemente de explicar cómo se organiza la Memoria del Trabajo Fin de Grado (TFG), del posible contenido de cada uno de los capítulos y secciones, así como de contenidos mínimos exigibles y algunas recomendaciones prácticas.

1.1. Introducción

La corrección de proyectos académicos mediante sistemas inteligentes actualmente está limitada, especialmente en la adaptación al nivel objetivo de los estudiantes en diferentes etapas formativas. Solucionar esta carencia es la motivación del presente Trabajo de Fin de Grado.

El sistema se intragra con el entorno académico con el fin de ofrecer al alumnado una retroalimentación más precisa y personal, alineada con los objetivos marcados por el profesorado.

Esta memoria se estructura en cuatro capítulos:

- Memoria descriptiva
- Documentación técnica del sistema
- Manuales de la aplicación
- Apéndices

1.2. Objetivos del trabajo

Se explicará también el alcance de la aplicación, sus restricciones y limitaciones, así como las perspectivas del trabajo.

El objetivo general es el diseño y desarrollo de un sistema inteligente que genere retroalimentación automática para proyectos académicos, adaptada al nivel requerido por el profesor segun la etapa formativa, mediante el uso de LLMs y la herramienta LangGraph.

La aplicación desarrollada permite analizar proyectos académicos de la asignatura Programación Orientada a Objetos (POO) y generar comentarios ajustados a los objetivos formativos de la asignatura mediante el análisis de la rubrica correspondiente, cubriendo la necesidad de una corrección más personalizada y del nivel objetivo. El sistema se integra en el entorno académico y está orientado a servir como apoyo al profesorado, sin sustituir el proceso de evaluación humana en ningún caso.

Los objetivos del trabajo son:

- A nivel personal, profundizar en el uso de tecnologías de inteligencia artificial aplicadas al ámbito educativo, así como en el diseño de sistemas basados en LLMs y flujos de trabajo mediante LangGraph.
- A nivel teórico, estudiar los fundamentos de la evaluación automática, los modelos de lenguaje de gran tamaño y su aplicación en entornos formativos.
- A nivel práctico, implementar una aplicación que mejore la calidad de la retroalimentación automática, teniendo en cuenta el nivel académico del alumnado.

El sistema desarrollado está concebido para su aplicación en un contexto académico, aunque su uso queda limitado al ámbito de una asignatura concreta por el momento. Entre las principales restricciones se encuentran la dependencia de la API de los LLMs empleados y la necesidad de una serie de rubricas para los trabajos a corregir. Como perspectiva futura, el proyecto podría ampliarse a otras asignaturas o niveles educativos, así como incorporar mecanismos de evaluación más avanzados y personalizados.

1.3. Entorno de aplicación

En los últimos años, el uso de inteligencia artificial en todos los ambitos ha experimentado un notable crecimiento, especificamente en el sector educativo diversos trabajos y herramientas tecnológicas han abordado la corrección automática de proyectos mediante reglas predefinidas o análisis estático de código, con resultados desiguales.

Las soluciones tradicionales presentan como principal ventaja su rapidez y consistencia en la evaluación, así como su capacidad para reducir la carga de trabajo del profesorado. No obstante, estas herramientas suelen ofrecer retroalimentación genérica, poco flexible y escasamente adaptada al nivel académico del estudiante, lo que limita su utilidad ya que la personalización es clave para el aprendizaje.

Más recientemente, el uso de modelos de lenguaje de gran tamaño ha abierto nuevas posibilidades en la generación de retroalimentación automática más rica y contextualizada. Sin embargo, muchos de los trabajos existentes no consideran de forma explícita el nivel formativo requerido segun la etapa formativa ni estructuran adecuadamente el proceso de generación de comentarios, lo que puede dar lugar a respuestas inconsistentes o poco alineadas con los objetivos docentes.

En este contexto este Trabajo de Fin de Grado propone un enfoque basado en el uso de LangGraph y LLMs, orientado a mejorar la calidad y adecuación de la retroalimentación automática en proyectos académicos. Este enfoque permite estructurar el proceso

de análisis y generación de respuestas, teniendo en cuenta el nivel exigido al estudiante y el contexto específico de la asignatura, justificando así la pertinencia y originalidad del trabajo desarrollado.

Capítulo 2

Metodología

En este capítulo se detallarán las cuestiones metodológicas, es decir, las metodologías y herramientas que se han utilizado para plantear el trabajo.

2.1. Proceso de desarrollo

Este apartado se refiere a si se ha utilizado un modelo de desarrollo iterativo, de prototipos, en cascada, etc (ciclo de vida del software). Por ejemplo, el siguiente diagrama refleja el modelo de desarrollo iterativo:

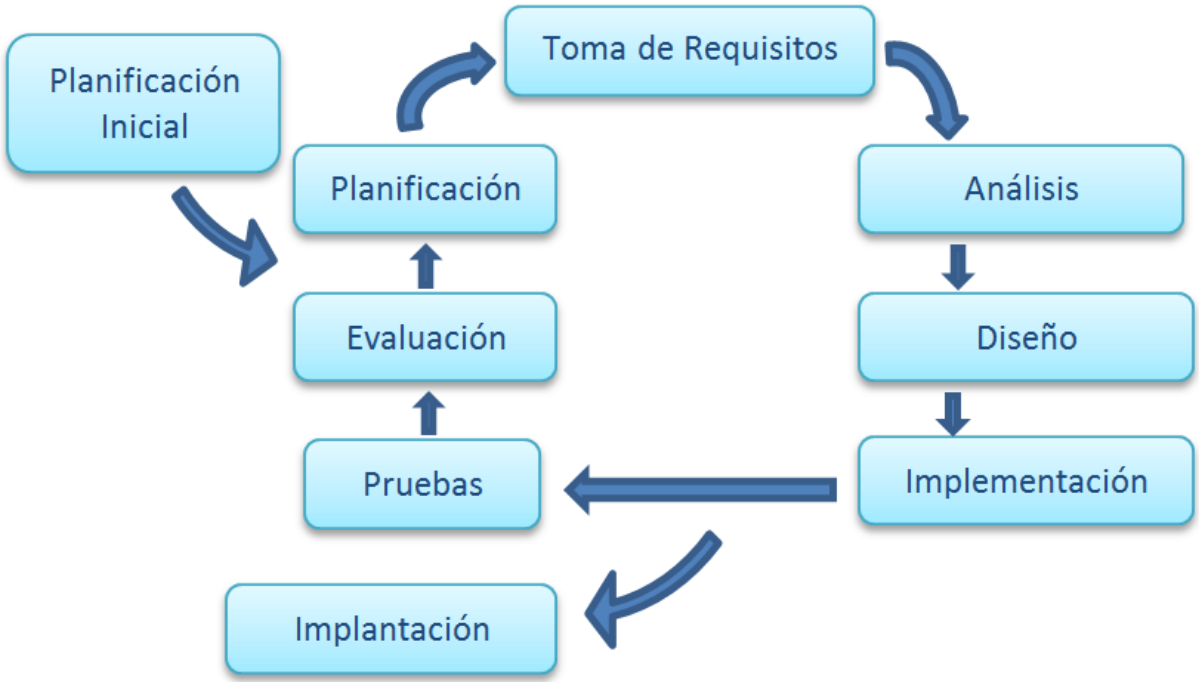

Figura 2.1: Modelo de desarrollo iterativo

También se especificará si se ha usado el paradigma de programación estructurada o programación orientada a objetos.

Asimismo, se explicará si he ha usado alguna metodología especial (metodologías ágiles, TDD, etc).

2.2. Herramientas utilizadas

En esta sección se entrará más en detalle en las tecnologías especiíficas que se han empleado para desarrollar la aplicación: lenguajes de programación empleados, gestor de bases de datos, herramientas usadas en la planficación y en la generación de documentación, sistema operativo, etc.

Asimismo, se defenderán los criterios por los que se han seleccionado las herramientas elegidas, de entre otras posibles, haciendo referencia a posibles ventajas e inconvenientes.

2.3. Arquitectura

Se explicará en este apartado la arquitectura de la aplicación, tanto a nivel lógico como físico. Por ejemplo, el diagrama 2.2 ilustra la arquitectura lógica cliente-servidor.

Sin embargo, el diagrama 2.3 representa una posible arquitectura física correspondiente a dicho moxelo lógico.

2.4. Definición de siglas y abreviaturas

Es interesante mostrar una tabla o lista de abreviaturas y siglas (acrónimos) de uso extendido en la Memoria, pero de los que es posible que el lector de la misma no tenga conocimiento o sean ambiguos.

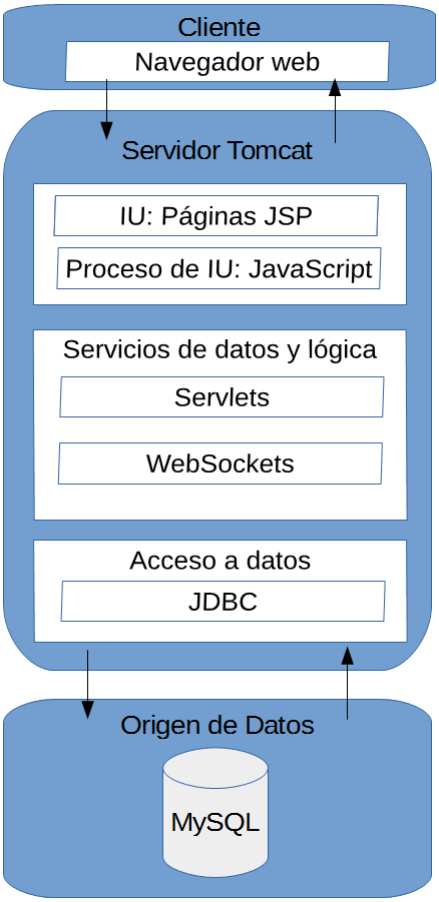

Figura 2.2: Arquitectura lógica

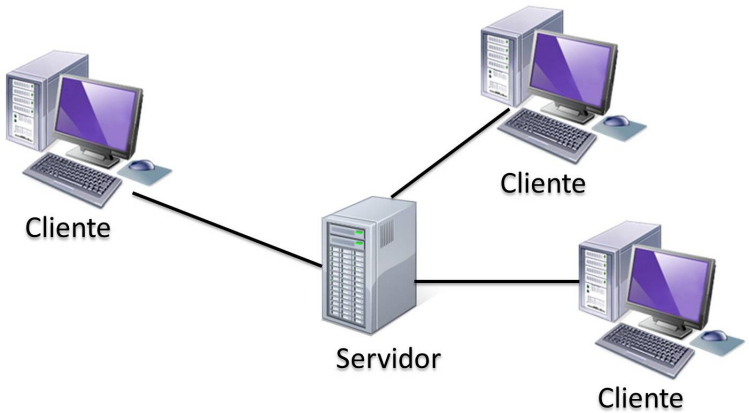

Figura 2.3: Arquitectura física

Capítulo 3

Planificación

En este capítulo se aboradarán las cuestiones relativas a la planificación del trabajo.

3.1. Estimación del esfuerzo

Para la planificación temporal y de costes pueden aplicarse al menos dos enfoques: el primero se basa en la estimación del esfuerzo mediante **puntos de función** y su transformación en líneas de código, con el fin de calcular el coste final mediante el método *COCOMO II*.

El segundo método consiste en la estimación de **puntos de Caso de Uso**, que se basa en un análisis previo de los requisitos funcionales de la aplicación.

Cualquiera que sea el método elegido, conviene justificar las ventajas e inconvenientes, aplicándolo al caso concreto del TFG.

3.2. Planificación temporal

En la planificación de tareas se tendrán el cuenta los objetivos y requisitos de la aplicación, así como la estimación del esfuerzo, pero también puede tenerse en cuenta la distinción en tareas de análisis, diseño, implementación y pruebas, en cada una de las posibles iteraciones del ciclo de vida del proyecto, y el marco temporal (plazo de entrega del proyecto).

El correspondiente diagrama de Gantt permite observar de forma gráfica la distribución temporal de las tareas:

3.3. Presupuesto económico

Para la estimación del presupuesto se tendrán en cuenta las herramientas utilizadas (hardware y software, con los factores de impacto que correspondan al duración del proyecto), junto con los recursos humanos necesarios, según la planificación de tareas, y el

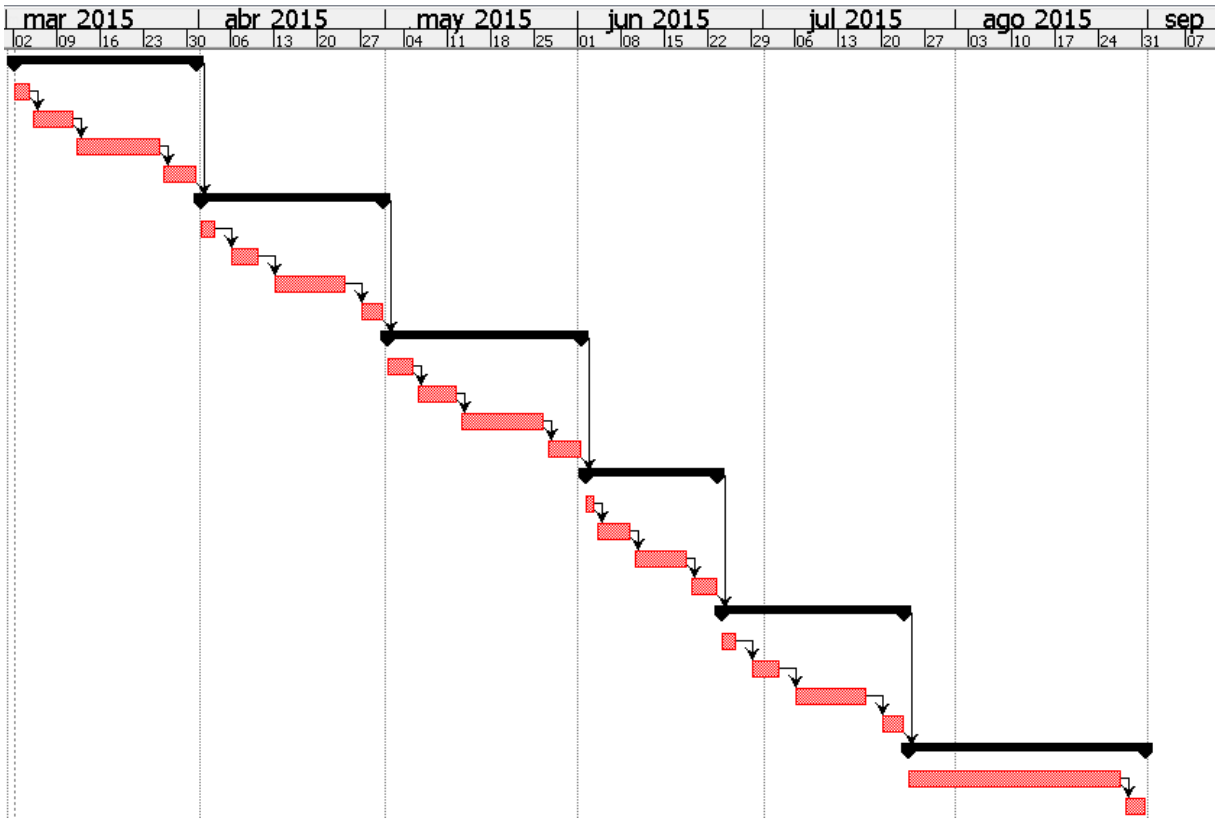

Figura 3.1: Diagrama de Gantt

tipo de rol (analista, programador, etc) correspondiente a cada tarea. Para ello conviene tener una tabla actualizada del coste por hora de cada tarea del proyecto, además del estudio previo de la planificación de tareas.

3.3.1. Hardware y software

Valoración del hardware y software utilizado en el proyecto.

3.3.2. Recursos humanos

Valoración de los recursos humanos

3.3.3. Presupuesto total

Se resumirán los conceptos económicos, por ejemplo en una tabla, con las estimaciones realizadas y el coste real, si procede:

Nota: es interesante analizar la discrepancia (si es que existe) entre el presupuesto y el coste real, y tratar de explicar sus causas.

	Hardware	**Software**	**RRHH**	**Total**
Estimación I	650,75 €	500	60000 €	*61150,75 €*
Estimación II	650,75 €	500	40000 €	*41150,75 €*
Coste real	650,75 €	400	50000 €	*51050,75 €*

Cuadro 3.1: Presupuesto total

Capítulo 4

Conclusiones

Este capítulo debe analizar los aspectos relativos a la consecución de los objetivos propuestos, y el grado de satisfacción de los mismos.

Asimismo, se puede abordar un análisis crítico de las ventajas en inconvenientes de las soluciones propuestas. en el trabajo.

Por otra parte, conviene resaltar la el grado de originalidad del TFG, así como el valor añadido que supone el TFG con respecto a los estudios de grado; dicho de otra forma, qué cosas nuevas se han aprendido y trabajado en el proyecto que no se habían aprendido a lo largo de la carrera.

Por último, es necesario proponer posibles mejoras y futuras ampliaciones del trabajo.

Parte II

Documentación técnica

Capítulo 5

Análisis

5.1. Requisitos

En este apartado se describen los requisitos del sistema que se va a desarrollar: requisitos funcionales, de interfaz de usuario, de información, etc. Asimismo, se especificará si la aplicación debe ser o no multiplataforma (o multilingüe), si se establecen restricciones de uso o de otro tipo (por ejemplo heredadas del entorno organizativo en el que se integrará), etc.

Para una mejor descripción de los requisitos, se usan los llamados casos de uso, basados en la identificación de actores y tareas.

A modo de ejemplo, este sería un caso de uso para la tarea "solicitar registro" en una aplicación WEB, que deba ser validado por un administrador:

Nombre e ID del CU	CU-01. Solicitar registro
Actor	Usuario
Descripción	El usuario envía sus datos de usuario al sistema, generando una solicitud de alta que deberá ser gestionada por un administrador.
Precondiciones	PRE-1. El usuario no está identificado en el sistema.
Postcondiciones	POST-1. La solicitud queda almacenada en el sistema.
Flujo normal	FN1 El actor introduce sus datos de usuario e indica al sistema que quiere solicitar el registro. FN2 El sistema comprueba los datos introducidos. FN3 Si los datos son correctos, el sistema almacena la solicitud de registro e informa al actor del resultado.
Flujo alternativo 1	FA3 Si los datos son incorrectos se informa al usuario del error y no se procede a almacenar la solicitud.

Flujo alternativo 2	FA3 Si no hay otros usuarios, el usuario pasa automáticamente al estado de activación y tendrá el rol de administrador.
Excepciones	E1 El usuario ha dejado campos requeridos sin rellenar. E2 El usuario o correo electrónico ya existen. E3 El usuario está bloqueado.
Prioridad	Alta
Otra info	El primer usuario registrado en la aplicación será el que adquiera el rol de administrador.

Cuadro 5.1: CU-01. Solicitar registro

5.2. Atributos de calidad

Se analizarán aquí los posibles indicadores de calidad de la aplicación desarrollada, a saber:

Tiempo de corrección : La aplicación deberá ser capaz de evaluar un proyecto académico de programación de hasta 2 000 líneas de código distribuidas en múltiples archivos en un tiempo máximo de 1 minuto, incluyendo el tiempo de comunicación con la API de la LLM, en un equipo local con 16 GB de RAM y CPU de 8 núcleos.

Uso de memoria : El consumo máximo de memoria de la aplicación cliente durante el proceso de corrección no deberá superar el 60 % de la memoria RAM disponible.

Ejecución concurrente : El sistema deberá permitir la evaluación concurrente de al menos dos proyectos de programación independientes mediante llamadas paralelas a la API, sin que el tiempo medio de respuesta por proyecto se incremente más de un 40 %.

Privacidad de los datos : El código fuente enviado a la API de la LLM será solo el contenido necesario para la evaluación y no se registrará ninguna informacion sensible del usuario. El codigo tampoco se almacenará de forma persistente en la aplicación cliente si este no lo aprueba.

Aislamiento de archivos : La aplicación deberá restringir el acceso a los archivos del sistema únicamente al directorio del proyecto a evaluar, impidiendo la lectura, ejecución o modificación de código fuera de dicho ámbito.

Gestión de errores : Ante fallos en la comunicación con la API de la LLM, tiempos de espera excedidos o respuestas inválidas, la aplicación deberá capturar la excepción,

informar al usuario de forma clara y permitir reintentar la operación sin finalizar abruptamente la ejecución.

Tolerancia a entradas inválidas : El sistema deberá detectar y rechazar proyectos que contengan archivos no soportados, dependencias inexistentes o código corrupto, notificando el error al usuario en un tiempo máximo de 10 segundos antes de iniciar el proceso de evaluación.

Estabilidad prolongada : La aplicación deberá mantener un funcionamiento estable durante sesiones continuas de al menos 2 horas evaluando múltiples proyectos de programación consecutivos, sin fugas de memoria, acumulación de procesos de red ni degradación significativa del rendimiento.

Capítulo 6

Diseño

Los aspectos de diseño depende mucho del paradigma elegido (programación orientada a objetos o no, por ejemplo).

6.1. Diseño de datos

En primer lugar podemos ver el modelo de datos que se usará para diseñar la base de datos, si esta es necesaria. Así, habrá que mostrar el diagrama entidad-relación, el modelo relacional, y el diccionario de datos, a nivel lógico, y por otra parte el diseño físico de la base de datos, y los perfiles de acceso a la misma por parte de los usuarios (si procede).

6.2. Diagramas de clase y de secuencia

Si la programación es orientada a objetos, es necesario especificar los diagramas de clases que se usan.

Los diagramas de secuencia describen la secuencia de pasos con que un usuario interacciona con el sistema, y están relacionados con los casos de uso de la etapa de análisis.

Es interesante también especificar si se ha hecho uso de diseño de patrones.

Capítulo 7

Implementación

En este capítulo se mostrarán los detalles más relevantes en cuanto a la implementación del sistema.

Capítulo 8

Pruebas

Este capítulo es muy importante, y muestra las pruebas que se han realizado para demostrar el funcionamiento del sistema.

Se referirán las estrategias de prueba utilizadas, y cuándo se han efectuados dichas pruebas.

Se realizarán pruebas de caja negra y de caja blanca, unitarias, de integración, de rendimiento, de seguridad, etc.

Se realizará una buena batería de pruebas para probar la funcionalidad de la aplicación, que está determinada por los requisitos de la misma, y se mostrarán los resultados (si estos son muchos, se pueden dejar los menos relevantes para el contenido del CD o para los Anexos de la Memoria).

Parte III

Manuales de la Aplicación

Capítulo 9

Manual de Instalación

Este capítulo se dedica a explicar los prerrequisitos técnicos, sistema operativo, etc, necesarios para instalar la aplicación desarrollada en el TFG.

Se explicará también todo lo necesario para conseguir e instalar el software de terceros que se precise con carácter previo.

Si se trata de una aplicación WEB, explicar cómo instalar y configurar los servidores necesarios para que funcione dicha aplicación, y cómo se realiza el despliegue de la misma.

Si se trata de una aplicación de escritorio, explicar el proceso de instalación y configuración en las plataformas correspondientes.

Capítulo 10

Manual de Usuario

Este capítulo final se dedica a desarrollar el manual del usuario final, así como del usuario administrador, si fuese necesario.

10.1. Manual de Usuario

Se recomienda que contenga la información relevante, lo más completa y clara posible, como si fuese el manual de un producto comercial, dirigido a un usuario no avanzado. Esta documentación pudiera coincidir (de hecho, se recomienda) con la AYUDA del programa.

10.2. Manual de Administración

Este manual sería mucho más técnico, pero asimismo igual de claro y completo, y estaría dirigido al administrador de la aplicación.

Parte IV

Apéndices

Apéndice A

Anexos

A.1. Información complementaria

En los anexos se pueden incluir todas aquellas cosas que pueden servir, a nivel teórico o práctico, para ayudar al lector a profundizar en ciertos aspectos de la memoria, como los prerrequisitos teóricos, descripción de alguna herramienta en particular, etc.

A.2. Diagramas y tablas

También se pueden incluir aquí las tablas, los casos de uso, o los resultados de prueba que no se hayan incluido en el capítulo correspondiente, para una mayor claridad en la lectura de la documentación técnica.

Apéndice B

Contenido del CD

Explicar el contenido del CD que se entrega junta a la Memoria, junto con el árbol de directorios correspondiente, para facilitar la navegación por los archivos del mismo.

Bibliografía

[1] P. Bourque y R.E. Fairley. *Guide to the Software Engineering Body of Knowledge, Version 3.0*. ACM-IEEE. 2014.

[2] *Function Point Languages Table*. Quantitative Software Management. Agosto 2015. URL: `http://www.qsm.com/resources/function-point-languages-table`.

[3] R. Pressman. *Ingeniería del Software*. McGraw-Hill, 2014.

[4] Adi Shamir. "How to share a secret". En: *Communications of the ACM* 22 (1979).