Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

Ingeniería Informática
Especialidad: Computación
Cuarto curso. Primer cuatrimestre

Proyectos.

Práctica 5: Documentación

Valentín Gabriel Avram Aenachioei
03524931C
p92avavv@uco.es

Curso académico 2022-2023
Córdoba, 9 de junio de 2023

Índice

Índice de figuras

En este documento se analizará la estructura y contenido del TFG nº 42, titulado *Sistema autocorrector de ejercicios de programación en Python*. En este análisis nos centraremos en la estructura seguida por el autor, comprobando que hace bien y que hace aspectos pueden llegar a ser erroneos o mejorables.

1. Estructura

1.1. Portada

Siendo lo primero que se ve del TFG la portada, esta tiene todo lo esperable, título, institución, autor, director, fecha, etc.

1.2. Introducción

El primer capítulo del trabajo es el de *Introducción*, donde el autor, en lugar de centrarse en introducir cual es el proyecto que ha realizado, parece centrarse en una introducción al propio problema, y no en el cómo se plantea la resolución de este. Solo los dos últimos párrafos se centran en el propio trabajo a realizar.

1.3. Definición del problema

El apartado de definición del problema, que debería tratar de resolver cuál es el problema a tratar y las necesidades que se cubren al desarrollar este trabajo, el autor se centra en otros puntos, llamandolo *Descripción Técnica del problema*. Estos puntos deberían estar fuera de la definición del problema, pues aunque importantes, se centran en muchas cosas salvo en definir el problema que busca tratar.

1.3.1. Funcionamiento

Describe brevemente el funcionamiento de las actuales soluciones al problema a tratar, sin hacer mención al propio funcionamiento de su proyecto. Se puede considerar como una explicación incorrecta del funcionamiento del proyecto, pero correcta definición del problema.

1.3.2. Entorno

Correcta descripción del entorno del problema inicial, tanto haciendo referencia a usuarios, hardware a usar y software.

1.3.3. Vida esperada

Correcta explicación de la vida útil esperada para el proyecto desarrollado, pero de nuevo, no sería parte de la definición del problema.

1.3.4. Ciclo de mantenimiento

De nuevo, correcto desarrollo, pero no sería parte de la definición del problema, si no del propio sistema desarrollado.

1.3.5. Competencia

Hace mención a un posterior desarrollo realizado en el capítulo de "Antecedentes". Podemos considerar que este apartado sobra en éste cápitulo, pues ni es parte de la definición del problema ni añade ninguna información relevante.

1.3.6. Estandarización

Breve explicación de los estándares a seguir en el desarrollo de la aplicación, sin entrar en detalle, y sin ser esto parte de la definición del problema.

1.3.7. Calidad y fiabilidad

Se nombran ciertas cualidades a cumplir, sin entrar en cómo o por qué. De nuevo, no debería ser parte de este capítulo.

1.3.8. Aspecto externo

Mención a la interfaz de usuario y a toda la documentación a realizar. Útil, pero no debería ser parte de este capítulo.

1.3.9. Programa de tareas

Descripción de las fases a realizar para desarrollar el proyecto. Breve explicación de cada fase del desarrollo, pero una vez más, no es parte de la definición del problema.

1.3.10. Pruebas

Explicación teórica y general de lo que son las pruebas de software, sin entrar en detalles para el proyecto concreto. Está fuera de lugar en este capítulo, además de no entrar en detalles.

1.4. Objetivos

Consta de una breve mención de los objetivos que persigue el proyecto, sin indicar que parte del problema original se desea solucionar a través de cada objetivo, ni el cómo. Ni siquiera se llegan a describir los objetivos a cumplir, solo se mencionan.

1.5. Antecedentes

Consta de una correcta descripción de los sistemas software, frameworks y sistemas usados, añadiendo algunas imágenes que realmente poco tienen que ver con el uso que se les da a esas herramientas en el proyecto, usadas únicamente para visualizar el software usado. No se hace mención a ningún antecedente al problema, o soluciones tratadas con anterioridad.

1.6. Restricciones

En este capítulo, el autor no ha conseguido explicar de forma correcta qué son los factores datos y factores estratégicos. Con respecto a los factores datos, el autor ha malentendido la definición de estos, poniendo como ejemplos de factores datos algunas decisiones de diseño. Además, no llega a incluir ningún factor dato como tal.

Con respecto a los factores estratégicos, estos si han sido interpretados de forma correcta, aunque algunos de los factores anteriormente nombrados

como factores dato deberían haber sido incluidos en esta categoría.

1.7. Recursos

En el capítulo de Recursos se especifican los recursos humanostanto el autor como los directores, los recursos software, únicamente los usados para el desarrollo, sin mencionar los necesarios para el uso y ejecución del sistema, y los recursos hardware usados para el desarrollo, de nuevo, sin especificar unos requisitos mínimos o recomendados para su ejecución.

1.8. Especificación de requisitos

En este capítulo, los requisitos se dividen en funcionales, no funcionales, y de información. La especificación de requisitos es corta, tiene problemas de duplicidad, requisitos poco desglosados y falta de requisitos funcionales. Aún así, con los requisitos presentes y su desarrollo es suficiente para entender su papel en el proyecto.

1.8.1. Requisitos de información

Generalmente bien definidos, aunque en llega a repetir requisítos (RINF-4 y RING-7 son el mismo requisito). Además, hay requisitos definidos como requisitos de información que deberían estar definidos como funcionales (RINF-6).

1.8.2. Requisitos funcionales

Los requisitos se definen de manera clara y concreta los objetivos de la aplicación. Algo a mejorar podría ser el requisito RF-6, el cuál se podría dividir en 3 requisitos diferentes, pudiendo entrar en mas detalle para cada uno.

1.8.3. Requisitos no funcionales

Solo se especifican tres requisitos, uno de los cuales (RNF-2) es muy subejtivo, poco especificado y no cuantificable. Por lo demás, son correctos

pero escasos.

1.9. Análisis funcional

En este capítulo se usa UML para definir el comportamiento de los actores del proyecto y los diagramas de casos de usos del sistema, se especifican los casos de uso y sus respectivas validaciones. La especificación de los actores es correcta y acorde al proyecto. El diagrama de los casos de uso se puede visualizar en la figura 1.

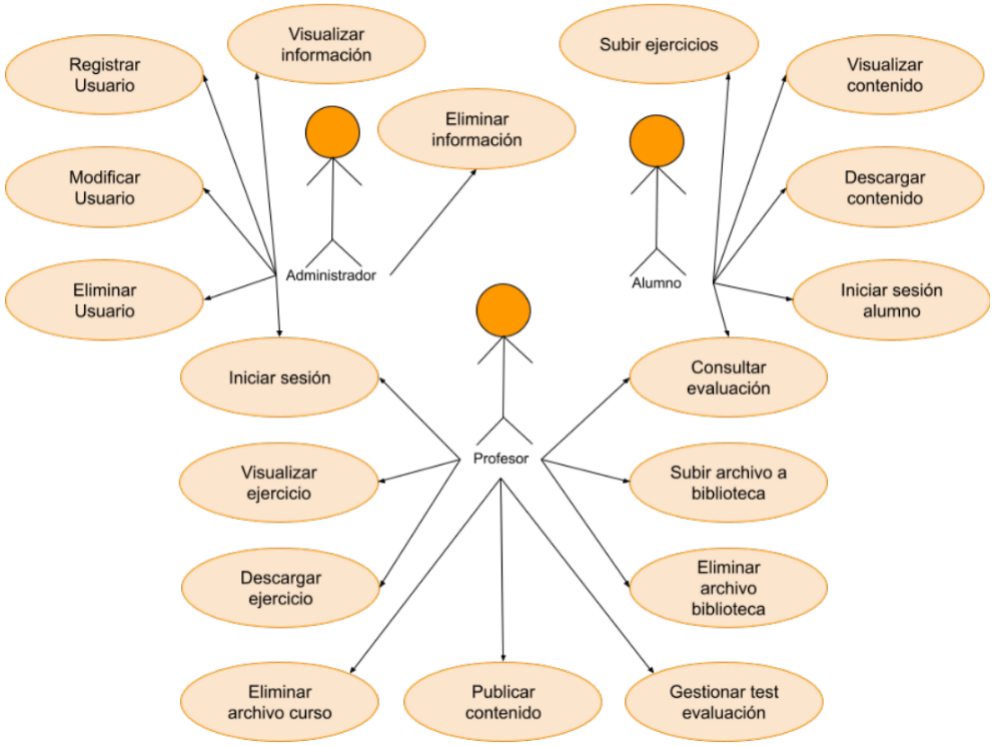

Figura 1: Diagrama de casos de uso

Se puede apreciar que el diagrama puede llegar a ser confuso. Los casos de uso no están englobados dentro de un sistema. Debería haber una generalización de varios actores aquí expuestos hacia un usuario general y que este realizara los casos de uso compartidos como iniciar sesión. Los casos de uso están bien definidos al menos.

La especificación de los casos de usos sigue correctamente el estándar UML con los aparatos típicos de estas representaciones. La especificación,

identificación, los actores, la validación y los flujos son correctos. Los casos de usos están relacionados con un requisito funcional, lo que es correcto.

1.10. Análisis del sistema

Este capítulo define lo relacionado con la base de datos. Se definen las entidades de manera simple y correcta, y las interrelaciones entre entidades. Generalmente, las interrelaciones son correctas, salvo un caso especifico.

Se describen las normalizaciones de la base de datos, su modelo relacional, y por último, parte del código usado para crear tablas y registro de usuarios, aunque no se indica todo el código usado para la creación de la base de datos.

1.11. Diseño del sistema

En este capítulo se definen la arquitectura a utilizar, así como las clases utilizadas en el programa, la organización de ficheros, las vistas y los servicios externos utilizados. La arquitectura utilizada está definida de manera clara y precisa, aunque en el diagrama de despliegue aparecen tecnologías que no se especificaron previamente en los factores estratégicos, como Bootstrap.

Posteriormente se definen mediante un diagrama de clases bastante simple, con tipos de clases: de control, de interfaz y de entidad. Esta manera de definir las clases mezcla UML con UML Components, pero puede ser correcta. Las clases están correctamente conectadas y son coherentes. Después de las clases viene un solo diagrama de secuencia que se utiliza para definir un caso de uso. El autor justifica que es importante representar este caso de uso, pero sería correcto representar cada caso de uso con su respectivo diagrama de secuencia.

Continúa con la correcta descripción de la organización de ficheros y las vistas. Ambas están definidas de forma coherente y clara. El último apartado es una especificación de distintas funcionalidades del programa, como el cifrado de la contraseña, el control de acceso a rutas o la autocorrección de ejercicios, además de servicios externos como Google PyDrive. Todo está correctamente definido y explicado.

1.12. Conclusiones

Se especifican tanto una conclusión a nivel técnico como posibles mejoras a futuro. Este último apartado se podría añadir en otro capítulo, ya que no sería una propia conclusión. En el apartado de conclusiones técnicas se mencionan las soluciones dadas, sin relacionarlas al respectivo problema que se ha llegado a solucionar. En el apartado de mejoras futuras, se hace mención a ciertos apartados a añadir en el proyecto. Prácticamente, no son mejoras al propio proyecto, si no actualizaciones añadiendo nuevas funcionalidades. Es correcto, pero lo ideal sería especificarlo en un capítulo separado.

1.13. Bibliografía

Se especifican todas las referencias usadas, tanto citadas en el respectivo texto como desarrolladas en el apartado de bibliografía. Como formato de bibliografía, se usa el formato IEEE. Además, se incluyen hiperenlaces a cada sitio web visitado.

1.14. Apéndices

Como apéndices, se incluyen dos de vital importancia, manuales de usuario y manuales de código.

1.14.1. Manual de usuario

El manual de usuario está bien desarrollado, diferenciando entre los distintos tipos de usuarios, incluyendo imágenes y capturas de pantallas para cada funcionalidad de las posibles.

1.14.2. Manual de código

En este se especifican todas las versiones de todos los sistemas software usados, un listado de directorios explicando brevemente sus contenidos, y el código fuente de la aplicación. Las siguientes 240 páginas incluyen todo el código fuente del sistema. Este apartado resulta prácticamente inútil, pues es todo el código, apenas comentado, siendo prácticamente ilegible, y casi imposible localizar cierto apartado, pues está dudosamente ordenado

2. Posibles modificaciones

Como aspectos faltantes en los documentos, tanto como que no se hayan añadido o no estén bien definidos o detallados, se pueden definir:

- En el capítulo *Objetivos*, faltaría detallar el qué se quiere solucionar o conseguir, describiendo brevemente como se plantea esa solución.
- En el capítulo *Antecedentes*, faltaría especificar antecedentes,tanto como ha evolucionado el problema, o como enfoques anteriores para solucionarlo.
- En el capítulo *Recursos*, estaría bien detallar los requisitos necesarios para ejecutar el programa, tanto software como hardware.
- Sería beneficioso añadir diagramas de secuencia de todos los casos de uso, o al menos alguno más.
- Añadir Bootstrap como factor estratégico en la parte de factores.
- En el manual de Código, sería útil añadir cierto orden y comentarios en el código para mejorar su legibilidad.

3. Consideraciones generales

En cuanto al formato se puede decir que es correcto, tanto en fuente de texto, tamaño, espaciadp, orden en cuanto a títulos y subtítulos. Faltarían más capítulos, y modificar los capítulos ya existentes, siguiendo las directrices indicadas en el apartado "Posibles modificaciones". La numeración de las páginas es correcta, añadiendo títulos y subtítulos a cada imagen y tabla añadida. Se podrían añadir menciones y referencias a cada imagen. Formato, texto, faltas ortográficas, numeración de páginas, subtítulos de imágenes y gráficas, etc, redacción clara y correcta, páginas en blanco entre capítulos, aunque los capítulos empiezan por páginas impares, así que podemos decir que el formato y estilo es el correcto.

En cuanto a índices, se diferencia correctamente entre un índice general, uno de tablas y uno de figuras, con su respectiva numeración de páginas en numeración romana. Un error de formato aparece en la bibliografía, pues entre las dos páginas que componen la bibliografía hay diferencias de margen entre ambas páginas.

Como último error de formato, sería el propio manual de código, pues es prácticamente ilegible, poco estilizado, sin un orden claro y poco comentado.