|  |  |  |
| --- | --- | --- |
| Una caricatura de una persona  Descripción generada automáticamente con confianza baja | UNIVERSIDAD DE BURGOS  ESCUELA POLITÉCNICA SUPERIOR  Grado en Ingeniería Informática |  |

**TFG del Grado en Ingeniería Informática**

**<Sistema de extracción y gestión automatizada de requisitos desde documentos PDF para proyectos de ingeniería>**

Presentado por Estíbalitz Díez en Universidad de Burgos - 15 de enero de 2025

Tutores: Dr. José Manuel Galán Ordax y Dr. José Ignacio Santos Martin

|  |  |  |
| --- | --- | --- |
| Una caricatura de una persona  Descripción generada automáticamente con confianza baja | UNIVERSIDAD DE BURGOS  ESCUELA POLITÉCNICA SUPERIOR  Grado en Ingeniería Informática |  |

D. José Manuel Galán Ordax y D. José Ignacio Santos Martin, profesores del departamento de Ingeniería de Organización, área de Organización de Empresas.

Exponen:

Que el alumno Dña. Estíbalitz Díez, con DNI 78.893.267R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado “***Sistema de extracción y gestión automatizada de requisitos desde documentos PDF para proyectos de ingeniería***”.

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

En Burgos, 15 de enero de 2025

Vº. Bº. del Tutor:

D. José Manuel Galán Ordax

Vº. Bº. del co-tutor:

D. José Ignacio Santos Martin

RESUMEN

Descriptores

ABSTRACT

Keywords

Índice general

[Índice general III](#_Toc177895238)

[Índice de figuras V](#_Toc177895239)

[Índices de tablas VI](#_Toc177895240)

[1. Introducción 1](#_Toc177895241)

[1.1. Estructura de la memoria 1](#_Toc177895242)

[1.2. Materiales adjuntos 1](#_Toc177895243)

[2. Objetivos del proyecto 3](#_Toc177895244)

[2.1. Objetivos generales 3](#_Toc177895245)

[2.2. Objetivos personales 3](#_Toc177895246)

[3. Conceptos teóricos 4](#_Toc177895247)

[3.1. XXX 4](#_Toc177895248)

[4. Técnicas y herramientas 5](#_Toc177895249)

[4.1. Técnicas metodológicas 5](#_Toc177895250)

[4.2. Patrones de diseño 6](#_Toc177895251)

[4.3. Control de versiones 6](#_Toc177895252)

[4.4. Alojamiento de repositorio 6](#_Toc177895253)

[4.5. Gestión del proyecto 6](#_Toc177895254)

[4.6. Comunicación 6](#_Toc177895255)

[4.7. Entorno de desarrollo integrado (IDE) 7](#_Toc177895256)

[4.8. Documentación de memoria 7](#_Toc177895257)

[4.9. Documentación del código 7](#_Toc177895258)

[4.10. Integración y despliegue continuos (CI/CD) 7](#_Toc177895259)

[4.11. Calidad y consistencia del código 7](#_Toc177895260)

[4.12. Cobertura de código 7](#_Toc177895261)

[4.13. Framework escritorio 7](#_Toc177895262)

[4.14. Sistema gestor de bases de datos 7](#_Toc177895263)

[4.15. Bibliotecas y librerías relevantes 8](#_Toc177895264)

[4.16. Desarrollo web 8](#_Toc177895265)

[4.17. Otras herramientas 8](#_Toc177895266)

[5. Aspectos relevantes del desarrollo del proyecto 9](#_Toc177895267)

[5.1. Inicio del proyecto 9](#_Toc177895268)

[5.2. Metodologías 9](#_Toc177895269)

[5.3. Formación 9](#_Toc177895270)

[5.4. Obtención y procesamientos de datos 9](#_Toc177895271)

[5.5. División por aplicaciones 9](#_Toc177895272)

[5.6. Fuentes de noticias 9](#_Toc177895273)

[5.7. Desarrollo backend y frontend de DashBoard de usuario 9](#_Toc177895274)

[5.8. Desarrollo backend y frontend del Lab 9](#_Toc177895275)

[5.9. Testing, log de tests y otros asapectos 9](#_Toc177895276)

[6. Trabajos relacionados 10](#_Toc177895277)

[7. Conclusiones y Líneas de trabajo futuras 11](#_Toc177895278)

[7.1. Conclusiones 11](#_Toc177895279)

[7.2. Líneas de trabajo futuras 11](#_Toc177895280)

[Bibliografía 12](#_Toc177895281)

Índice de figuras

[Figura 3.1: Imagen modelo 3](#_Toc177844215)

Índices de tablas

[Tabla 4.1 Tabla modelo 4](#_Toc177844212)

# Introducción

Texto introductorio

## Estructura de la memoria

La memoria sigue la siguiente estructura:

* Introducción: breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
* Objetivos del proyecto: exposición de los objetivos que persigue el proyecto.
* Conceptos teóricos: breve explicación de los conceptos teóricos clave para la comprensión de la solución propuesta.
* Técnicas y herramientas: listado de técnicas metodológicas y he rramientas utilizadas para gestión y desarrollo del proyecto.
* Aspectos relevantes del desarrollo: exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
* Trabajos relacionados: estado del arte en las aplicaciones y sitios web de bolsa y finanzas.
* Conclusiones y líneas de trabajo futuras: conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

* Plan del proyecto software: planificación temporal y estudio de viabilidad del proyecto.
* Especificación de requisitos del software: se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
* Especificación de diseño: se describe la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
* Manual del programador: recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
* Manual de usuario: guía de usuario para el correcto manejo de la aplicación

## Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

* Anexos: consultar la documentación técnica.
* Herramienta web: este proyecto está especialmente pensado para ser utilizado en un entorno local, pero se ha dispuesto una página web para que el usuario pueda hacer pruebas antes de instalarlo en su equipo. Visitar FAT: Financial Analysis Tool.
* Vídeo de presentación: ver presentación del TFG.
* Vídeo de demostración: ver demostración funcional.

Además, los siguientes recursos están accesibles a través de internet:

* Repositorio: visitar FAT. GitHub.
* Documentación del código: visitar FAT. Read The Docs.
* Logs de calidad de código: la calidad del código se ha medido de forma automatizada con pylint y GitHub actions. Los informes generados muestran sólo los aspectos mejorables. Es decir, aquello que no está en los logs es porque está con la calidad óptima. Descargar informe de pylint.
* Cobertura de código: La cobertura de los tests se ha medido de formaautomatizada con coverage y GitHub actions. Es posible descargar un informe HTML en formato .zip. Se descomprime en un directorio y se lanza desde index.html. Descargar informe de coverage.

# Objetivos del proyecto

Texto general

## Objetivos generales

El proyecto consistirá en la creación de una herramienta de software que ayude a automatizar el proceso de manejo de requisitos extraídos de documentos PDF en proyectos de ingeniería. A través de una interfaz sencilla, los documentos serán procesados para extraer la información, permitiendo al usuario gestionar, organizar y visualizar los requisitos, comparar versiones de documentos, y finalmente exportar esta información en formatos estándar como Excel o CSV. El sistema también incluirá funcionalidades adicionales para el seguimiento del cumplimiento de requisitos y la asignación automática de subsistemas.

Objetivos específicos:

• Automatización de la extracción de contenido

• Gestión y almacenamiento de los requisitos

• Interfaz de usuario intuitiva

• Comparación de versiones

• Exportación de datos

• Funcionalidad añadida: seguimiento del estado de los requisitos.

## Objetivos personales

# Conceptos teóricos

Textp general

## XXX

Icono

Descripción generada automáticamente

Figura .: Imagen modelo

# Técnicas y herramientas

Textp general

## Técnicas metodológicas

* Scrum

Scrum [109] es un marco de trabajo relativamente estructurado y con roles específicos dentro de la metodología Agile (roles principales: Product Owner, Scrum Master y desarrollador) . Se puede utilizar tanto para la gestión de proyectos como para el desarrollo de productos, especialmente en el despliegue de software.

Con Scrum los proyectos se dividen en iteraciones cortas llamadas sprints. Al final de cada sprint se debe presentar un producto mínimo viable y evaluar lo que se ha hecho bien y lo que se puede mejorar. Se ha optado por esta metodología, frente a otras como Waterfall [93] , porque ofrece una alta adaptabilidad y genera entrega temprana de valor, con productos viables y valorables por el usuario final desde las primeras fases.

* Test-Driven Development (TDD)

TDD [94] es una metodología de desarrollo de software que se enfoca en escribir una batería de tests automatizados antes de iniciar la implementación del código fuente del propio software. Posteriormente, se hace un proceso de refactorización para mejorar o solucionar los defectos encontrados. Mis conocimientos previos de Django y SQLite no me han permitido utilizar de forma integral esta metodología, pero sí que se ha seguido endiferentes etapas del desarrollo, mejorando notablemente la calidad del código final.

* Behavior-Driven Development (BDD)

BDD [86] es una metodología que se basa en el comportamiento del software y me ha resultado útil en aquellas fases del proyecto en las que no tenía una idea preconcebida del cómo trabajar con Django pero sí que conocía el resultado final esperado.

La ventaja de este enfoque es que las pruebas se escriben en un lenguaje natural y es sencillo extrapolarlas a un gestor de tareas con un sistema Kanban.

* Kanban

Kanban [100] es un método visual de gestión de proyectos a través de la utilización de un tablero, en el que se disponen una serie de tarjetas con las tareas pendientes, en curso o finalizadas. Esto permite crear un flujo de trabajo que prioriza aquellas tareas más urgentes o que aportan antes valor a un producto

## Patrones de diseño

## Control de versiones

Git y Mercurial son sistemas de control de versiones distribuidos (DVCS),

mientras que Subversion- o SVN- es centralizado (VCS).

Una de las ventajas de Git es que permite a cada desarrollador tener una

copia en local del repositorio completo y, aunque es menos eficiente para

proyectos muy grandes, es más sencillo de utilizar para proyectos pequeños.

Además, el sistema de ramificación de Git es más intuitivo y facilita la tarea

de los desarrolladores.

## Alojamiento de repositorio

Herramientas consideradas: GitHub [38], GitLab [40] y Gitea [37]. Herramienta elegida: GitHub. Me he decantado por GitHub porque ya lo conocía, porque se utiliza en algunas asignaturas del Grado de Ingeniería Informática y porque es muy popular, lo que facilita la resolución de problemas gracias a su mayor comunidad.48

Técnicas y herramientas

GitHub puede ofrecer menor control sobre proyectos grandes- Gitea y GitLab permiten auto hospedaje con la configuración que más nos interese-, pero en proyectos medios o pequeños es una herramienta práctica y sencilla de utilizar, con diferentes integraciones y que facilita el uso de flujos de trabajo CI/CD

## Gestión del proyecto

Herramientas consideradas: Zube [46], ZenHub [116], Trello [6] y Jira [5].

Herramienta elegida: Zube.

Zube es una plataforma de gestión de proyectos que se integra muy bien con GitHub. Además, permite la sincronización en tiempo real con el repositorio de referencia que se esté utilizando y ofrece una interfaz fácil de utilizar con posibilidad de seguimiento a través de burndonws , burnups y throughput del equipo de desarrollo o de los desarrolladores de forma individual.

Frente a las alternativas valoradas, Zube ha sido la más intuitiva, permitiendo hacer seguimiento y planificación del proyecto en pocos pasos.

## Comunicación

Herramientas consideradas: email, GitHub y Microsoft Teams [60]. Herramientas elegidas: todas las anteriores. La comunicación en tiempo real, con llamadas o vídeo llamadas a través de Teams, aporta soluciones rápidas por el continuo flujo de preguntas respuestas. Pero no siempre se pueden utilizar estos medios y es preferible hacer uso de email o de requests de GitHub. Además, recientemente, se ha dado la posibilidad de integrar MS Teams con GitHub [62] para enviar notificaciones a un grupo de trabajo

## Entorno de desarrollo integrado (IDE)

Herramientas consideradas: Spyder IDE [44], Visual Studio Code [61].

Herramienta elegida: Visual Studio Code.

A pesar de que ambos entornos tienen plugins de alta calidad, VS Code ofrece mayor personalización. Además, VS Code es integrable con GitHub de forma sencilla y ofrece aplicaciones de terceros que facilitan tanto la implementación de código como las labores de testeo y control de calidad. VS Code se ha utilizado en este trabajo para desarrollo de Django, como editor CSS y HTML, para JavaScript y para Markdown, así como medio de integracción con GitHub, entre otros

## Documentación de memoria

Herramientas consideradas: Latex y Word.

## Documentación del código

Herramientas consideradas: Sphinx [7], Read the Docs [80] y pdoc [67].

Herramientas elegidas: Sphinx y Read the Docs.

Sphinx es la herramienta más extendida en la comunidad Python para documentar código, es compatible con varios formatos y estilos de docstrings- PyDoc, Google o Numpy entre otros- y, además, puede generar documentación de manera automática a partir de los docstrings.

En este trabajo, se ha escogido el formato de Numpy para la documentación de código y el estilo de Read The Docs para las plantillas de la documentación HTML.

La documentación del código puede ser consultada en fat.readthedocs.io o en fat.rtfd.io

## Integración y despliegue continuos (CI/CD)

## Calidad y consistencia del código

## Cobertura de código

## Framework escritorio

Tkinter

## Sistema gestor de bases de datos

Herramientas consideradas: SQLite [77] y PotgreSQL [42].

Herramienta elegida: SQLite.

SQLite es una librería de código escrita en lenguaje C, que implementa un motor de bases de datos pequeño y rápido. Se califican a sí mismos como un sistema gestor de bases de datos ligero y multiplataforma. Tiene la ventaja de estar muy extendido y utilizan un único archivo en el sistema de almacenamiento, lo que favorece su distribución y uso.

Por su parte, PostgreSQL tiene opciones más avanzadas que SQLite y está pensado para soportar alta concurrencia y bases de datos grandes.

Pero SQLite está perfectamente integrado con Django y no requiere de configuración adicional como sí requeriría PostgreSQL. Además, SQLite es suficiente para las expectativas de este trabajo- la migración a PostgreSQL sería recomendable en caso de escalar el proyecto-

## Bibliotecas y librerías relevantes

Plumber extracción pdfs

Maker para los tokenes

pip install XlsxWriter

pip install pandas

pip install tk

pip install sqlalchemy

## Desarrollo web

## Otras herramientas

Mendeley [33] es una herramienta de gestión de referecnias bibliográficas y colaboración académica. Entre sus características cabe destacar la capacidad de organizar las referencias y exportarlas a un archivo que se puede utilizar desde Word

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Tabla . Tabla modelo

# Aspectos relevantes del desarrollo del proyecto

## Inicio del proyecto

## Metodologías

## Formación

## Obtención y procesamientos de datos

## División por aplicaciones

## Fuentes de noticias

## Desarrollo backend y frontend de DashBoard de usuario

## Desarrollo backend y frontend del Lab

## Testing, log de tests y otros asapectos

I

# Trabajos relacionados

# Conclusiones y Líneas de trabajo futuras

Textp general

## Conclusiones

## Líneas de trabajo futuras

Bibliografía