UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Ingeniería de Documentos aplicada al Proceso del Software

Trabajo Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

AUTOR/A: Hernandez Chinchilla, Susana Michelle

Tutor/a: Penadés Gramage, María Carmen

CURSO ACADÉMICO: 2023/2024

Agradecimientos

Quiero expresar mi sincero agradecimiento a **ValgrAI: Valencian Graduate School Artificial Intelligence**, por su apoyo financiero y por haberme dado la oportunidad de formar parte de su comunidad. Su compromiso con la educación y la investigación ha sido fundamental para mi crecimiento académico y profesional. A mi tutora, María Carmen Penadés Gramaje, por su guía y por todo lo que he aprendido de ella.

Resum

La documentació en el procés de programari és de vital importància, ja que proporciona una base estructurada per a la comunicació, manteniment i evolució dels projectes. Sense una documentació adequada, la comprensió i gestió del programari poden tornar-se caòtiques, afectant la qualitat i eficiència del desenvolupament.

En l'àmbit de l'Enginyeria de Documents aplicada al procés de programari, la generació de documentació a partir de repositoris o ferramentes de suport per a la gestió de projectes programari com Microsoft *Azure es presenta com un desafiament fonamental. Este Treball Final de Màster s'endinsa en l'exploració de com l'Enginyeria de Documents pot ser emprada per a extraure dades i informació d'estes aplicacions, amb l'objectiu de generar documentació d'alta qualitat i rellevància en el context del desenvolupament de programari.

En este treball s'ha investigat de quina manera és possible aprofitar les funcionalitats de plataformes com Microsoft *Azure *DevOps per a recopilar dades significatives que posteriorment seran transformats en documents pertinents per al procés de desenvolupament de programari. A més, s'ha analitzat l'estat de l'art quant a les pràctiques actuals d'Enginyeria de Documents en l'àmbit del programari, amb el propòsit d'identificar tendències, desafiaments i oportunitats per a la millora contínua.

Paraules clau: Documentació de Programari, Desenvolupament de Programari, Enginyeria de Documents, Generació de Documentació, Microsoft Azure DevOps, DITA

Resumen

La documentación en el proceso de software es de vital importancia, ya que proporciona una base estructurada para la comunicación, mantenimiento y evolución de los proyectos. Sin una documentación adecuada, la comprensión y gestión del software pueden volverse caóticas, afectando la calidad y eficiencia del desarrollo.

En el ámbito de la Ingeniería de Documentos aplicada al proceso de software, la generación de documentación a partir de repositorios o herramientas de soporte para la gestión de proyectos software como Microsoft Azure se presenta como un desafío fundamental. Este Trabajo Final de Máster se adentra en la exploración de cómo la Ingeniería de Documentos puede ser empleada para extraer datos e información de dichas aplicaciones, con el objetivo de generar documentación de alta calidad y relevancia en el contexto del desarrollo de software.

En este trabajo se ha investigado de qué manera es posible aprovechar las funcionalidades de plataformas como Microsoft Azure DevOps para recopilar datos significativos que posteriormente serán transformados en documentos pertinentes para el proceso de desarrollo de software. Además, se ha analizado el estado del arte en cuanto a las prácticas actuales de Ingeniería de Documentos en el ámbito del software, con el propósito de identificar tendencias, desafíos y oportunidades para la mejora continua.

Palabras clave: Documentación de Software, Desarrollo de Software, Ingeniería de Documentos, Generación de Documentación, Microsoft Azure DevOps, DITA

Abstract

Documentation in the software process is of vital importance, as it provides a structured basis for communication, maintenance, and evolution of projects. Without adequate documentation, understanding and managing software can become chaotic, affecting the quality and efficiency of development.

In the field of Document Engineering applied to the software process, generating documentation from repositories or support tools for software project management like Microsoft Azure presents a fundamental challenge. This Master's Thesis delves into exploring how Document Engineering can be employed to extract data and information from these applications to generate high-quality and relevant documentation in the context of software development.

This work investigates how it is possible to leverage the functionalities of platforms like Microsoft Azure DevOps to gather significant data that will later be transformed into pertinent documents for the software development process. Additionally, the state of the art regarding current Document Engineering practices in the software field has been analyzed, with the purpose of identifying trends, challenges, and opportunities for continuous improvement.

Keywords: Software Documentation, Software Development, Document Engineering, Documentation Generation, Microsoft Azure DevOps, DITA

Tabla de contenidos

1. Introducción **1**
1.1. Motivación 1
1.2. Antecedentes 1
1.3. Objetivos 3
1.4. Estructura del Documento 3

2. Preliminares **5**
2.1. Ingeniería del Software 5
2.1.1. Proceso de desarrollo del software 5
2.1.2. Microsoft Azure DevOps aplicado a la Ingeniería del Software . . . 8
2.2. Ingeniería de Documentos 13
2.2.1. Proceso de Ingeniería de Documentos: Model Matrix 13
2.2.2. DITA aplicado a la Ingeniería de Documentos 16

3. DITAOps Connect **21**
3.1. Azure DevOps y la Documentación Software 21
3.2. Automatizando la generación de documentos con DITA 22
3.3. Solución Propuesta 22

4. Familia de Documentos **27**
4.1. Definiendo la familia de documentos 27
4.1.1. Release Notes 29
4.1.2. Progress Report 29
4.1.3. DITA como soporte de la familia de documentos 30

5. Método de Estandarización **35**
5.1. Estandarización de los work items de Azure DevOps 35
5.1.1. Plantillas definidas para work items 35
5.2. Transformado los datos a DITA 38
5.2.1. Tecnologías Utilizadas 38
5.3. Diseño del proceso para la generación de tópicos DITA 40

6. Implementación **43**
6.1. DITAOps Connect Backend 43
6.1.1. Diseño de la API 43
6.1.2. Arquitectura de la API 45
6.1.3. Proceso de Transformación 45
6.1.4. Estructura de los archivos del proyecto backend 46
6.2. DITAOps Connect FrontEnd 46

7. Resultados **51**
7.1. Escenario 1: Release Notes 51
7.2. Escenario 2: Progress Report 53

8. Conclusiones y Trabajos Futuros **57**
8.1. Conclusiones 57
8.2. Trabajos Futuros 59

Bibliografía **63**

Anexos **65**
.1. Anexo I 65
.1.1. Estructuras XML Definidas de Componentes DITA 65
.1.1.1. Summary 65
.1.1.2. Overview 66
.1.1.3. Features 66
.1.1.4. Enhancements 68
.1.1.5. BugFixes 69
.1.1.6. DITA Map 70
.1.2. Estructuras XML de Ejemplos de Componentes DITA generados . 71
.1.2.1. Summary 71
.1.2.2. Overview 72
.1.2.3. Features 72
.1.2.4. Enhancements 74
.1.2.5. BugFixes 75
.1.2.6. DITA Map 77
.2. Anexo II 77
.2.1. Documentos Generados 77
.2.1.1. *Release Notes* 77
.2.1.2. *Progress Report* 91

Índice de figuras

2.1. Proyecto en *Azure DevOps* 8
2.2. *Work item* en *Azure DevOps* 10
2.3. Estructura en *Agile Process* (extraído de [36]) 10
2.4. Estructura en *Basic Process* (extraído de [36]) 11
2.5. Estructura en *Scrum Process* (extraído de [36]) 11
2.6. Estructura en *CMMI Process* (extraído de [36]) 11
2.7. Ciclo de vida *work items* (extraído de [36]) 12
2.8. *Model Matrix* 13
2.9. Metodología propuesta para la generación de documentación 19

3.1. Metodología habitual para la generación de documentación 23
3.2. Metodología propuesta para la generación de documentación 23
3.3. Arquitectura de DITAOps Connect 24

4.1. Diagrama de Clases para Release Notes y Progress Reports 27
4.2. Estructura de la Familia de Documentos 28
4.3. Estructura de los documentos *Release Notes* y *Progress Report* 29

5.1. Obtener un work item específico 39
5.2. Consultar work items con WIQL 39

6.1. Consola del Backend 45
6.2. Estructura del Backend 46
6.3. Interfaz Gráfica Frontend - *Home* 47
6.4. Interfaz Gráfica Frontend - *Organization* 48
6.5. Interfaz Gráfica Frontend - *Documentation* 48
6.6. Interfaz Gráfica Frontend - *Contact* 49
6.7. Interfaz Gráfica Frontend - *User Account* 49

7.1. Opciones seleccionadas para generar *Release Notes* 52
7.2. DITAOps Connect generando *Release Notes* 52
7.3. Documento de *Release Notes* generado 53
7.4. Opciones seleccionadas para generar *Progress Report* 54
7.5. DITAOps Connect generando *Progress Report* 54
7.6. Documento de *Progress Report* generado 55

Lista de Tablas

5.1. Plantilla *User Story* en *DevOps Connect* 36
5.2. Plantilla *Task* en *DevOps Connect* 36
5.3. Plantilla *Bugs* en *DevOps Connect* 37
5.4. Plantilla *Features* en *DevOps Connect* 37

Capítulo 1

Introducción

Este documento de Trabajo Fin de Máster (TFM) propone explorar las posibilidades que ofrece la Ingeniería de Documentos en la esfera de la Ingeniería del Software, destacando la importancia de una documentación precisa y eficaz en el ciclo de vida de un proyecto. A través de referencias relevantes y un análisis detallado del estado actual de la disciplina, se busca aportar conocimientos valiosos que contribuyan al avance y perfeccionamiento de las prácticas documentales en el contexto del desarrollo de software.

1.1. Motivación

La creciente complejidad de los proyectos de desarrollo de software y la necesidad de mantener una documentación actualizada y precisa han impulsado la investigación en el campo de la Ingeniería de Documentos aplicada a este proceso. Por ejemplo, en entornos ágiles donde los requisitos y funcionalidades evolucionan rápidamente, la capacidad de generar documentación de forma automatizada a partir de repositorios como *JIRA [1]* o *Microsoft Azure DevOps [2]* se vuelve crucial para mantener la coherencia y la transparencia en el desarrollo del software.

Además, la integración de herramientas como *Microsoft Azure DevOps [2]* en el flujo de trabajo de documentación ofrece oportunidades para optimizar la recopilación y organización de información relevante para el proyecto. Esta integración no solo agiliza el proceso de generación de documentación, sino que también facilita la colaboración entre equipos y mejora la comunicación dentro de la organización.

En este contexto, la motivación principal de este trabajo radica en explorar cómo la Ingeniería de Documentos puede potenciar la eficiencia y la calidad de la documentación en el proceso de desarrollo de software, aprovechando las funcionalidades de herramientas existentes y abordando los desafíos específicos que surgen en entornos dinámicos y colaborativos.

1.2. Antecedentes

En investigaciones similares, se han abordado temas relacionados con la Ingeniería de Documentos aplicada al proceso de software. Por ejemplo, el trabajo de *Godoy Sánchez* y *Danny Alexander* [3] exploró cómo la integración de una herramientas para la

documentación de requisitos puede utilizarse en el proceso de desarrollo de software para automatizar la generación de documentación. Asimismo, el estudio de *Felipe Ebert [4]* analizó el uso de repositorios de datos y otras herramientas utilzadas en el desarrollo de software para considerar comentarios importantes sobre los sistemas en la de documentación implementando una metodología para agregar esta información en comentarios de código fuente. Adicionalmente, con un alcance muy similar, *Michael Moser y Josef Pichler* [5] introdujeron un generador de documentación para extraer documentación automaticamente de los comentarios del código fuente de programas en C++. Estas investigaciones destacan la relevancia de aprovechar las herramientas y plataformas disponibles en el ámbito del desarrollo de software para optimizar el proceso de documentación y garantizar la trazabilidad de la información generada en estas plataformas.

El presente trabajo se centra en explorar la posible sinergia entre *Microsoft Azure DevOps [2]*, una plataforma creada para el desarrollo y gestión de proyectos de software, y la *Arquitectura para la Intercambiabilidad de Información Técnica (DITA) [6] [7]*, un estándar para la creación y gestión de documentación técnica, aplicada en *Document Product Lines (DPL) [8]*, una metodología que ofrece una guía para modelar los contenidos comunes y variables en familias de documentos, tratándolos como un conjunto de características.

La generación manual de documentos a partir de datos de *Azure DevOps* presenta desafíos en términos de eficiencia, precisión, consistencia y escalabilidad:

- Tiempo y esfuerzo: El proceso manual de generación de documentos consume mucho tiempo y esfuerzo de parte de los equipos. Esto puede resultar en una baja eficiencia y poca disponibilidad de los documentos.
- Errores humanos: La generación manual de los documentos aumenta la probabilidad de cometer errores, se puede llegar a omitir información que es relevante o provocar inconsistencias entre los documentos. Este tipo de errores afectan la calidad y precisión de la documentación.
- Inconsistencia en los documentos: La falta de coherencia en el formato, la estructura y el contenido de los documentos producidos puede dificultar la comprensión y el seguimiento de la información.
- Limitaciones de escalabilidad: A medida que el volumen de datos y la necesidad de generar documentos aumentan, el enfoque manual se vuelve menos viable. La generación manual no es escalable y puede resultar abrumadora a medida que crece la complejidad del proceso.
- Actualizaciones tardías: Al utilizar metodologías tradicionales para la generación de documentos, puede ocasionar que la distribución de información este desactualizada y obsoleta.

La solución que ofrece este TFM es automatizar este proceso utilizando herramientas adecuadas que ayuden a mejorar la calidad y la eficacia de la documentación generada cuidando aspectos importantes como la consistencia de la información utilizada. Esto permitirá la extracción eficiente de los datos de *Azure DevOps* y su posterior conversión a una estructura *DITA*, facilitando la generación automática de documentos técnicos precisos.

1.3. Objetivos

El objetivo general de este TFM es desarrollar un sistema de Ingeniería de Documentos que permita la generación automática de documentación coherente y actualizada a partir de los *dashboards* o *work items* de *Microsoft Azure DevOps*, con el fin de mejorar la eficiencia en la generación de documentos y la disponibilidad de estos.

Este objetivo general se desglosa en los siguientes objetivos específicos:

1. Analizar las necesidades de documentación en el contexto de los proyectos de desarrollo de software que utilizan *Azure DevOps*, identificando los elementos clave que deben ser documentados y las mejores prácticas en la industria.
2. Estudiar las funcionalidades y la estructura de datos de los *dashboards* y *work items* de *Azure DevOps* para determinar cómo se puede extraer información relevante para la generación de documentación.
3. Diseñar un modelo de documentos que se ajuste a las necesidades de documentación identificadas, incluyendo plantillas y estructuras que permitan la integración de datos dinámicos provenientes de *Azure DevOps*.
4. Desarrollar un prototipo de sistema de Ingeniería de Documentos para definir la estructura y configuraciones de los documentos con contenido dinámico obtenido de información de los *dashboards* y *work items* de *Azure DevOps* y generar documentos estandarizados.
5. Documentar el proceso de desarrollo, los resultados obtenidos y las lecciones aprendidas, con el fin de que sirva como guía para futuras implementaciones y mejoras del sistema.
6. Proponer recomendaciones y líneas futuras de investigación basadas en los resultados obtenidos, con el fin de mejorar y expandir el sistema de Ingeniería de Documentos para su aplicación en otros contextos y herramientas de gestión de proyectos de software.

1.4. Estructura del Documento

Se ha planteado en los apartados anteriores, la motivación y los objetivos que abarcara este trabajo y la problemática que se pretende atender con este trabajo.
Los siguientes capítulos se estructurarán de la siguiente manera:

- Capítulo 2 expone los preliminares, detalla sobre la Ingeniería del Software y la Ingeniería de Documentos y la plataforma y metodología principal de cada una respectivamente.
- Capítulo 3 aborda el uso de *Azure DevOps* en el proceso de la documentación del software, como se puede automatizar la generación de estos documentos utilizando *DITA* y la solución propuesta utilizando esta herramienta y la metodología *DITA*.
- Capítulo 4 se ocupa de la definición y estructura de la familia de documentos.
- Capítulo 5 describe el método de estandarización de la información para aprovechar al máximo el *API* de *Azure DevOps* y obtener el contenido para las es-

tructuras definidas en *DITA*.

- Capítulo 6 específica la implementación del desarrollo de la solución.
- Capítulo 7 expone los resultados obtenidos en los diferentes escenarios de uso.
- Capítulo 8 recopila las conclusiones y expone posible trabajos futuros para enriquecer el trabajo realizado.

Capítulo 2

Preliminares

En este capítulo se presenta una introducción al proceso de la Ingeniería del Software, destacando cómo se utiliza *Azure DevOps* como una herramienta esencial para apoyar las actividades del desarrollo de software. Además, se abordará la Ingeniería de Documentos, enfatizando el uso del estándar *DITA* (*Darwin Information Typing Architecture*) como un soporte fundamental para esta disciplina. El capítulo proporcionará una visión integral de cómo estas herramientas y metodologías se integran para optimizar tanto el desarrollo de software como la generación de documentación.

2.1. Ingeniería del Software

2.1.1. Proceso de desarrollo del software

La Ingeniería del Software es un enfoque sistemático y disciplinado para el desarrollo, operación y mantenimiento de software [9]. Este proceso abarca diversas etapas, cada una con métodos y prácticas específicas, que se utilizan para garantizar la calidad, eficiencia y éxito de los proyectos de software. El desarrollo del software puede ser un proceso complejo que requiere una planificación meticulosa, el uso de metodologías adecuadas y una documentación rigurosa ayuda a reducir esta complejidad. Las herramientas y plataformas modernas facilitan la colaboración, automatizan procesos y aseguran la calidad del producto final. Este enfoque sistemático y disciplinado es esencial para desarrollar software robusto y de alta calidad que satisfaga las necesidades del usuario y los requisitos de los negocios.

El proceso tradicional del desarrollo del software se puede sintetizar en los siguientes pasos:

1. Recolección de Requisitos:
   Consiste en la identificación y documentación de las necesidades del usuario y del sistema a desarrollar. En este paso se utilizan una serie de técnicas como entrevistas, encuestas, o análisis de documentos.
2. Análisis de Requisitos:
   Consiste en la definición clara y detallada de los requisitos y la creación de documentos de especificaciones de requisitos.
3. Diseño del Software:
   En este paso se define la arquitectura del sistema y se elabora un diseño deta-

llado de componentes. Se utilizan herramientas como diagramas UML (Unified Modeling Language) para definir los diagramas de clases y de secuencia [10].

4. Desarrollo o Implementación:
   Consiste en la codificación y programación del software. Uso de lenguajes de programación como que mejor conviene según el diseño y arquitectura del sistema.

5. Pruebas:
   Una vez finalizado el desarrollo o implementación se procede a realizar la verificación y validación del software. Existen varios tipos que se realizan como, *unit testing*, *integration testing*, *system testing*, *acceptance testing*.

6. Despliegue:
   Al concluir todos los pasos del ciclo de vida del software se procede con la implementación del sistema en un entorno de producción.

7. Mantenimiento:
   Aunque se haya concluido el trabajo de crear el sistema, no acaba ahí la labor ya que se continua el ciclo para agregar actualizaciones, mejoras y corrección de errores.

Cada uno de estos pasos debe estar documentado para garantizar el correcto mantenimiento y evolución del software, evitando así el parcheado sobre el código. Una documentación adecuada no solo facilita la comprensión y el seguimiento del proceso por parte de todos los involucrados, sino que también asegura la calidad y consistencia del producto final a lo largo del tiempo.

Metodologías de Desarrollo de Software
Existen dos tipos de metodologías principales utilizadas en el desarrollo del software:

1. Metodologías Tradicionales:
Existen diferentes metodologías tradicionales [11], algunas de las aproximaciones más conocidas son:

- Cascada (*Waterfall*): Con un enfoque secuencial donde cada fase debe completarse antes de pasar a la siguiente.
- RUP (*Rational Unified Process*): Una metodología iterativa que divide el proyecto en cuatro fases (inicio, elaboración, construcción y transición) y enfatiza la arquitectura del sistema y el desarrollo basado en componentes.
- CMMI (*Capability Maturity Model Integration*): Un modelo de mejora de procesos que proporciona a las organizaciones los elementos esenciales para mejorar su rendimiento en desarrollo y mantenimiento de software.

2. Metodologías Ágiles:
Existen diversas metodologías ágiles utilizadas en el desarrollo de software [12], siendo las más conocidas Scrum, Kanban y Extreme Programming (XP), entre otras.

- Scrum: Con un enfoque iterativo e incremental con roles específicos (*Scrum Master*, *Product Owner*) y reuniones recurrentes en una frecuencia determinada (*sprints*, *stand-ups*).
- Kanban: Metodología enfocada en la visualización del flujo de trabajo para mejorar la eficiencia y reducir el tiempo de ciclo.

- Extreme Programming (XP): Este enfoque se centra en prácticas técnicas sólidas y colaboración estrecha con el cliente.

Plataformas y Herramientas Utilizadas
Existe una amplia selección de herramientas y plataformas para diferentes actividades del desarrollo del software. Algunas de ellas se detallan a continuación

Control de Versiones
Control y seguimiento de los cambios realizados en el software.
Git: GitHub [13], GitLab [14], Bitbucket [15]

Integración y Entrega Continua (CI/CD)
Automatización de la construcción, prueba y despliegue del software.
Integraciones y servicios: Jenkins [16], Travis CI [17], CircleCI [18], Azure Pipelines [19]

Gestión de Proyectos
Herramientas para la planificación y seguimiento de tareas y sprints.
Plataformas: Jira [1], Trello [20], Asana [21], Azure Boards [22]

Entornos de Desarrollo Integrados (IDE)
Herramientas con funcionalidades específicas para la codificación y desarrollo de software.
Aplicaciones: Visual Studio [23], Visual Studio Code [24], IntelliJ IDEA [25], Eclipse [26], PyCharm [27]

Pruebas
Herramientas para pruebas automatizadas y manuales.
Selenium [28], JUnit [29], TestNG [30], Postman [31]

Documentación
Herramientas para crear, gestionar y publicar documentación técnica.
Confluence [32], SharePoint [33], DITA Open Toolkit [34]

Importancia de la Documentación del Software
Uno de los aspectos más importantes en el desarrollo del software, y la Ingeniería del Software en general, es la documentación. La documentación completa y bien estructurada ayuda en la comprensión del código y sus funcionalidades, permite conocer cómo se utiliza y agiliza el diseño de futuras mejoras y el mantenimiento. Además, proporciona los recursos para la capacitación de nuevos miembros del equipo y elimina la necesidad de depender de una persona o grupo de personas para conocer la funcionalidad de los sistemas. Generar documentación de software de forma manual es un proceso tedioso y propenso a errores, que consume mucho tiempo y puede resultar en información desactualizada o inconsistente. Este enfoque también dificulta la colaboración y la actualización eficiente de los documentos.

Una de las plataformas con muchas funcionalidades útiles es Azure DevOps Services de Microsoft [2]. Azure cuenta diferentes servicios que ayudan en las diversas actividades del desarrollo de software, algunos de ellos mencionados anteriormente. Azure DevOps también cuenta con una API [35] muy completa con muchas utilidades, una de ellas es la que explora este trabajo, la extracción de información, datos o contenido de la plataforma para utilizar en la generación de documentos. Considerando estas características se ha optado por enfocar este proyecto en la utilización de esta herramienta en concreto, en la siguiente sección se detallará más sobre esta plataforma y

su API.

2.1.2. Microsoft Azure DevOps aplicado a la Ingeniería del Software

Microsoft Azure DevOps es una plataforma integral de desarrollo y colaboración en la nube que proporciona una serie de herramientas y servicios para gestionar y respaldar el ciclo de vida completo de desarrollo de software. *Azure DevOps* integra funcionalidades esenciales como la planificación de proyectos, el control de versiones, la integración continua y la entrega continua (CI/CD), y la gestión de artefactos, facilitando así la colaboración eficiente entre equipos de desarrollo, operaciones y gestión de proyectos. En la Figura 2.1 se puede observar la página principal un proyecto en *Azure DevOps*.

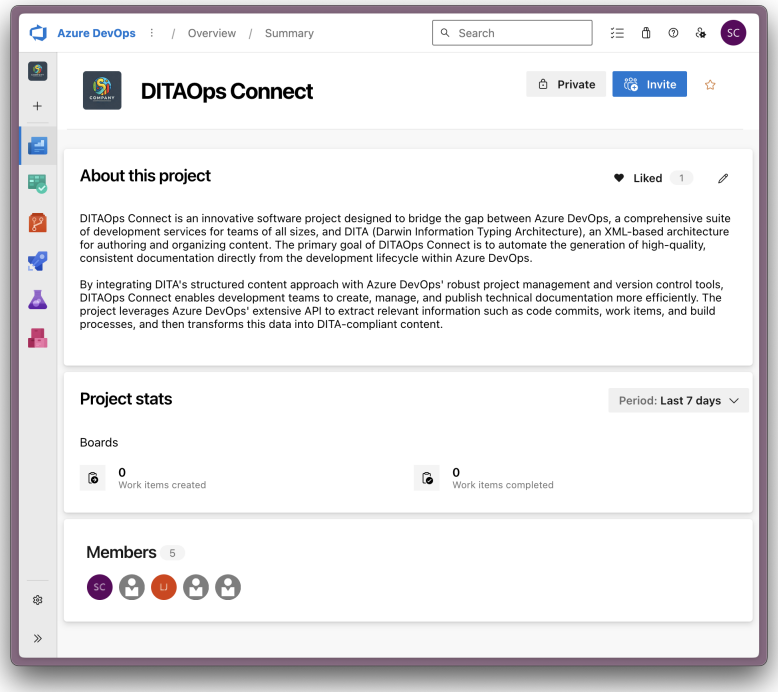

Figura 2.1: Proyecto en *Azure DevOps*

Cuando se trata de la generación de documentos para el proceso de desarrollo de software, Azure DevOps ofrece varias funcionalidades que pueden ser muy bien aprovechadas.

Funcionalidades

Azure Boards

Azure Boards es una herramienta que facilita la planificación y gestión de proyectos mediante metodologías ágiles. Proporciona tableros *Kanban*, *backlogs*, y paneles de trabajo adaptables que ayudan a rastrear tareas, errores de código (bugs) y características de manera visual y eficiente. Azure Boards permite a los equipos de desarrollo colaborar en la planificación, dar seguimiento, revisar y priorizar el trabajo y mantener la visibilidad del progreso del proyecto, mejorando la organización

y la productividad en el desarrollo de software. Uno de los elementos principales de los Azure Boards son los *work items* los cuales se pueden utilizar para rastrear tareas de documentación, y asegurar que toda la documentación necesaria se cree y se mantenga a lo largo del proceso de desarrollo. Este característica es crucial para este trabajo por lo que nos enfocaremos en ella mas adelante.

Azure Repos
Azure Repos es un servicio que proporciona repositorios de control de versiones para la gestión de código fuente. Ofrece dos tipos de repositorios: Git, un sistema de control de versiones distribuido ampliamente utilizado, y *Team Foundation Version Control (TFVC)*, un sistema centralizado de control de versiones. Azure Repos facilita la colaboración entre desarrolladores, permitiendo la revisión de código, la integración de cambios y la gestión eficiente de ramas, mejorando la calidad y la coherencia del código en proyectos de software.

Azure Pipelines
Azure Pipelines es un servicio que automatiza la integración continua (CI) y la entrega continua (CD) para la construcción, prueba y despliegue de aplicaciones. Compatible con cualquier lenguaje y plataforma. Azure Pipelines permite configurar y ejecutar *pipelines* que integran cambios de código de manera frecuente, asegurando que el software sea probado y esté listo para su despliegue en cualquier entorno.

Azure Artifacts
Azure Artifacts es un servicio que proporciona una solución para la gestión y el almacenamiento de paquetes de software, como NuGet, npm, Maven y Python. Permite a los equipos crear, alojar y compartir paquetes dentro de su organización. Azure Artifacts ayuda a gestionar dependencias de manera centralizada, asegurando que los paquetes utilizados en los proyectos sean consistentes y estén actualizados.

Azure Test Plans
Azure Test Plans es un servicio diseñado para gestionar y ejecutar pruebas manuales y automatizadas, garantizando la calidad del software en todas las etapas de desarrollo. Proporciona herramientas para planificar pruebas, rastrear resultados y reportar problemas, facilitando la creación de casos de prueba y la ejecución de pruebas exploratorias.

Al integrar estas funcionalidades de Azure DevOps en el proceso de desarrollo de software, los equipos pueden optimizar y automatizar la generación de documentación. Esto no solo garantiza que la documentación se mantenga actualizada, sino que también mejora la colaboración y comunicación entre los miembros del equipo al proporcionar una ubicación centralizada para toda la documentación relacionada con el proyecto. Es importante mencionar, que, aun teniendo a todas estas funcionalidades, habrá ciertos desafíos que tendremos que afrontar para la generación de esta documentación.

Work Items
Los *work items* o elementos de trabajo en *Azure DevOps* se utilizan para realizar el seguimiento de las diferentes características y requisitos que se están desarrollando en un proyecto, esto incluye tareas, bugs, historias de usuario y otras unidades de trabajo que ayudan a gestionar y organizar el progreso del desarrollo, asegurando que todas las actividades estén alineadas con los objetivos del proyecto y se cumplan dentro de los plazos establecidos. Los *work items* cuentan con una serie de campos

disponibles para la información del seguimiento. A cada elemento de trabajo se le asigna un identificador único dentro de la colección de proyectos. En la Figura 2.2 se muestra un *work item* en *Azure DevOps* Los tipos de *work items* disponibles difieren según el proceso utilizado cuando se crea el proyecto: Agile, Basic, Scrum o CMMI [36].

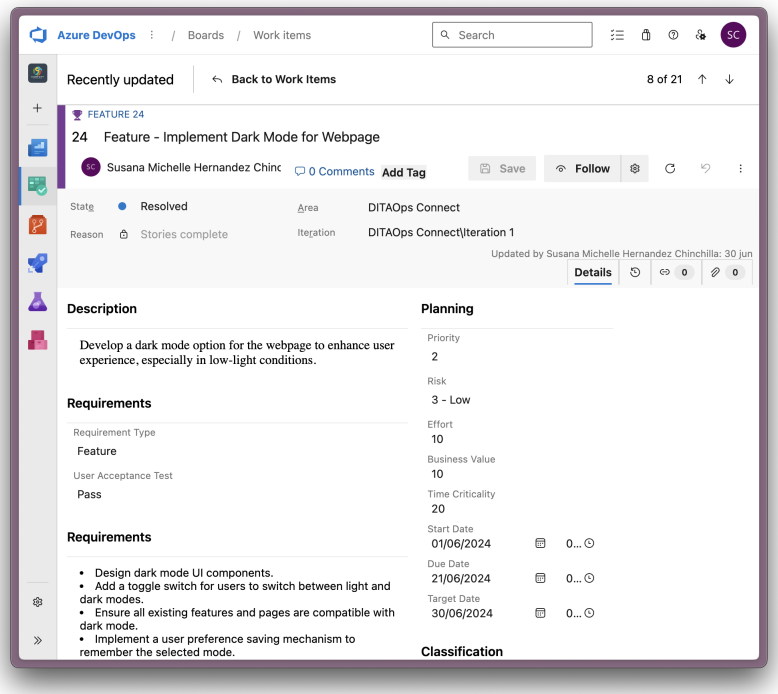

Figura 2.2: *Work item* en *Azure DevOps*

Las siguientes Figuras muestran los tipos de *work items* disponibles para los cuatro procesos predeterminados.

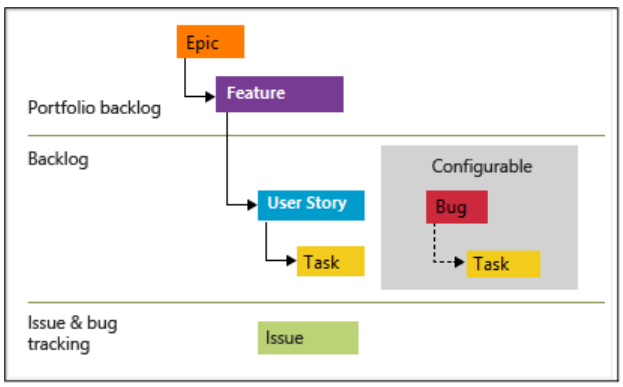

Figura 2.3: Estructura en *Agile Process* (extraído de [36])

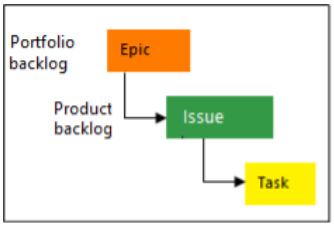

Figura 2.4: Estructura en *Basic Process* (extraído de [36])

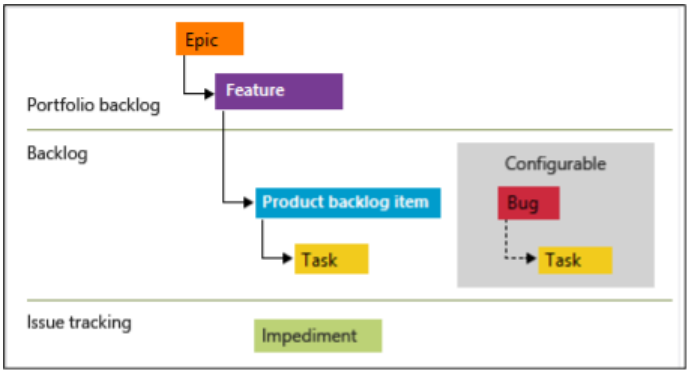

Figura 2.5: Estructura en *Scrum Process* (extraído de [36])

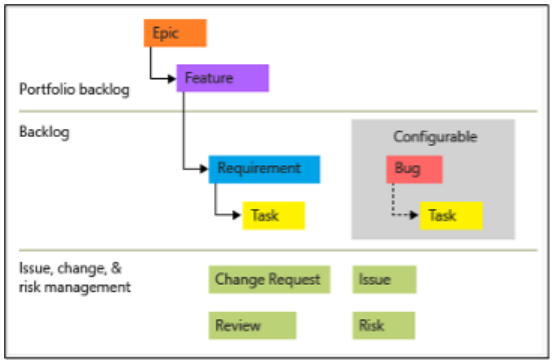

Figura 2.6: Estructura en *CMMI Process* (extraído de [36])

Los *work items* pendientes pueden denominarse:

1. *User Stories* (Agile) (Figura 2.3)
2. *Issues* (Basic) (Figura 2.4)
3. *Product Backlog Items* (Scrum) (Figura 2.5)
4. *Requirements* (CMMI) (Figura 2.6)

Los cuatro tipos son similares y describen la información para el cliente sobre el trabajo a realizar. Además, proporcionan campos para realizar un seguimiento ágil de la información sobre el trabajo.

En este trabajo, para la generación automática de documentación a partir de *work items* se ha optado por utilizar el tipo *Agile Process*.

Seguimiento de los work items en Agile Process

El siguiente diagrama (Figura 2.7) extraído de la página de *Microsoft Learn* [36], proporciona una imagen clara de cómo es el ciclo de vida de los *work items* de *Azure DevOps* en un proceso ágil. En el proceso *Agile*, una historia de usuario pasa por varios estados principales que describen su progreso. Los *work items* solamente pueden tener un estado a la vez y van transicionando entre uno y otro dependiendo de las actividades que se van realizando. Los cuatro estados principales que se definen para la historia de usuario son:

1. Nuevo (New): Este es el estado inicial de la historia de usuario. Aquí, la historia ha sido identificada y documentada, pero aún no se ha comenzado a trabajar en ella. Durante esta fase, la historia de usuario puede ser revisada y priorizada para ser abordada en próximas iteraciones.
2. Activo (Active): Una vez que se ha decidido trabajar en una historia de usuario, se mueve al estado .Activo". En esta etapa, los desarrolladores están trabajando activamente en la implementación de la historia de usuario. Esto puede incluir tareas como la codificación, el diseño, y la revisión de la funcionalidad que se está desarrollando.
3. Resuelto (Resolved): Cuando se completa la implementación inicial de la historia de usuario y ha sido completamente validada y probada, se mueve al estado -esuelto". Aquí, la historia de usuario ha pasado todas las revisiones necesarias y cumple con los criterios de aceptación definidos y está lista para ser entregada o desplegada en el entorno de producción. La historia de usuario en este estado está completamente terminada y no requiere más trabajo adicional.
4. Cerrado (Closed): Una historia de usuario se mueve al estado Çerradoçuando se decide que no procederá o se cancela. Esto puede suceder por diversas razones, como cambios en las prioridades del proyecto, descubrimiento de requisitos duplicados o irrelevantes, o porque la funcionalidad ya no es necesaria. Una vez en este estado, no se espera que se realice más trabajo en la historia de usuario.

Estos estados ayudan a los equipos Agile a organizar y gestionar el trabajo de manera eficiente, asegurando que las historias de usuario progresen de manera clara y estructurada desde su concepción hasta su finalización o cancelación.

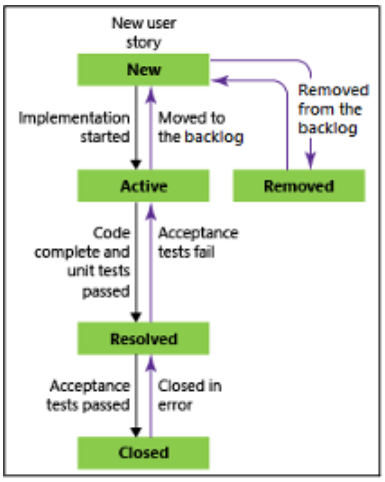

Figura 2.7: Ciclo de vida *work items* (extraído de [36])

2.2. Ingeniería de Documentos

2.2.1. Proceso de Ingeniería de Documentos: Model Matrix

La Ingeniería de Documentos se enfoca en la creación, gestión y mantenimiento de documentos técnicos y de otro tipo, esenciales para el soporte de procesos organizacionales y proyectos. Esta disciplina asegura que la documentación sea precisa, consistente y accesible, facilitando la comunicación y la colaboración en diversas fases de un proyecto. La Ingeniería de Documentos sintetiza ideas de diversos campos como análisis de sistemas, análisis y automatización de procesos de negocio y publicación electrónica. Este enfoque unifica diferentes modelos y perspectivas en torno al documento como elemento central.

La metodología de la Ingeniería de Documentos abarca técnicas de análisis y diseño que resultan en especificaciones precisas de la información requerida, documentos intercambiados y reglas de negocio. Esto facilita la creación de sistemas informáticos que automaticen o soporten estos procesos, siguiendo una metodología estructurada.

Model Matrix es una metodología utilizada en la gestión de documentos y contenidos. Esta técnica se centra en organizar, estructurar y gestionar los documentos de manera eficiente, facilitando su accesibilidad, reutilización y mantenimiento. En el ámbito de la Ingeniería del Software, esta metodología es utilizada para gestionar la complejidad y la interrelación de los documentos generados durante el desarrollo de un proyecto.

En la Ingeniería de Documentos, Model Matrix es un método que busca mejorar la calidad y la coherencia de los documentos producidos a lo largo del ciclo de vida de un proyecto de software. Su propósito principal es asegurar que todos los documentos relevantes estén alineados y que se mantenga una trazabilidad clara entre los diferentes artefactos del proyecto.

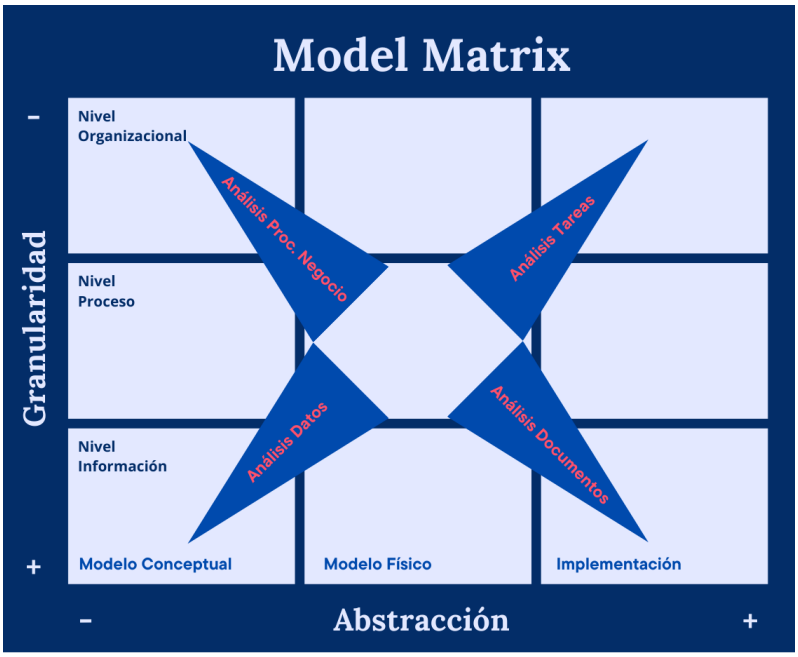

Figura 2.8: *Model Matrix*

La Figura 2.8 ilustra cómo diferentes tipos de análisis se distribuyen a lo largo de los niveles de granularidad y abstracción. Esto permite una comprensión estructurada y detallada de cómo los documentos y los datos se gestionan y se implementan en distintos niveles dentro de una organización. La metodología de *Model Matrix* se integra en este esquema para garantizar que todos los aspectos del análisis documental estén alineados y gestionados eficientemente, desde el nivel estratégico hasta la implementación detallada.

Fases del Proceso:

1. Análisis del Contexto de Uso: Se realiza un análisis del contenido que tendrán los documentos según el contexto de uso.

2. Análisis del Proceso de Negocio: Identificación de los procesos de negocio involucrados para determinar requisitos expresivos y aplicar patrones de diseño para definir estructuras de documentos.

3. Análisis de Documentos: Identificación de documentos representativos para extraer unidades de información atómicas.

4. Análisis de Componentes de Documentos: Identificación de componentes atómicos en los documentos, que consisten en unidades de información semántica, estructura sintáctica y formato de representación.

5. Ensamblar Componentes del Documento: Agrupación de componentes atómicos en unidades más complejas llamadas bloques constructivos, que representan unidades de información complejas.

6. Ensamblar Modelos de Documentos: Integración de componentes y relaciones para formar plantillas de documentos con estructura y contenido predeterminados.

7. Implementar Modelos de Procesos y Documentos: Desarrollo de una plataforma para generar documentos basados en plantillas, generalmente a través de asistentes basados en formularios.

Esta metodología permite gestionar documentos de manera eficiente, asegurando su adecuación a las necesidades de los procesos de negocio y la flexibilidad para adaptarse a diferentes tecnologías y contextos.

Herramientas de Autor y Publicación Utilizadas
En la Ingeniería de Documentos existen varias herramientas que se utilizan para las diferentes actividades de la creación y gestión de documentos complejos, asegurando que se mantengan estándares de calidad y consistencia en la documentación técnica.

Herramientas de Autor

1. Adobe FrameMaker
   Es una herramienta robusta de autoría que soporta la creación y gestión de documentos en múltiples formatos, incluidos *DITA* y *XML* [37].
   Su principales características son:

   - WYSIWYG (*What You See Is What You Get*): Permite a los usuarios ver el documento tal como aparecerá en su formato final mientras lo editan.

- Soporte para múltiples formatos: Incluye soporte para PDF, HTML5 responsivo, EPUB, Kindle, CHM, entre otros.
- Plantillas de estilo: Facilita la aplicación de estilos consistentes a través de plantillas CSS.
- Integración con DITA: *FrameMaker* facilita la creación de contenido estructurado y reutilizable basado en *DITA*, lo que es útil para la documentación técnica y compleja.

2. Oxygen XML Editor
   Es una de las herramientas más completas para la edición de XML, ampliamente utilizada para la creación de documentos basados en *DITA* y *DocBook* [38] [39].
   Su principales características son:
   - Editor visual: Proporciona una vista de autor visual que permite a los usuarios trabajar en un entorno *WYSIWYG*.
   - Soporte extenso para *DITA* y *DocBook*: Facilita la creación, edición y publicación de documentos en estos estándares.
   - Transformaciones y validaciones: Permite realizar transformaciones de documentos y validar la conformidad con los esquemas *DTD* o *XSD*.
   - Integración con sistemas de gestión de contenido: *Oxygen XML Editor* puede integrarse con sistemas como *CMS* para gestionar documentos a gran escala.
3. XMetaL
   Es una erramienta de autoría XML diseñada para crear y editar documentos basados en *DITA* [40].
   Su principales características son:
   - Edición estructurada: Ofrece una vista estructurada del documento que facilita la edición de contenido XML.
   - Soporte para *DITA*: Permite la creación de tópicos y mapas *DITA*, lo que es ideal para la documentación técnica.
   - Interfaz personalizable: Los usuarios pueden personalizar la interfaz y las funcionalidades para adaptarse a sus flujos de trabajo específicos.

Herramientas de Publicación

1. DITA Open Toolkit
   Es una plataforma de código abierto que proporciona herramientas para la transformación y publicación de documentos *DITA* en varios formatos [34].
   Su principales características son:
   - Transformaciones multiplataforma: Soporta la conversión de *DITA* a múltiples formatos de salida, incluyendo HTML, PDF, y ePub.
   - Extensible: Permite a los usuarios añadir *plugins* y personalizaciones para adaptar las transformaciones a sus necesidades específicas.
2. Adobe Experience Manager Guides
   Parte de la familia *Adobe Experience Manager*, esta plataforma soporta la crea-

ción y gestión de contenido técnico basado en *DITA* [41].
Su principales características son:

- Integración completa con *Adobe Experience Manager*: Facilita la gestión de contenido en un entorno centralizado.
- Publicación en múltiples canales: Soporta la publicación de contenido en diversos formatos y dispositivos.
- Herramientas colaborativas: Proporciona capacidades para la colaboración en equipo en la creación y gestión de contenido.

Las herrameintas de publicación suelen basarse en el uso de estándares de documentación, como es el caso de DITA [7] [6] [44] o de DocBook.

DocBook [39] es un estándar ampliamente utilizado para la creación de documentación técnica en XML, mantenido por OASIS para asegurar el cumplimiento con estándares internacionales. Sus principales características incluyen la estandarización y la disponibilidad de numerosas herramientas de conversión que permiten transformar los documentos a formatos como PDF, HTML y ePub.

DITA (Darwin Information Typing Architecture) [7] [6] [44] es un estándar XML especializado en la creación, administración y publicación de contenido estructurado y reutilizable, ideal para documentación técnica compleja. *DITA Open Toolkit* [34] es una plataforma de código abierto que facilita la transformación y publicación de contenidos DITA en múltiples formatos, como HTML, PDF y ePub. En la siguiente sección, se detallarán más sobre las capacidades de *DITA* y *DITA Open Toolkit*.

2.2.2. DITA aplicado a la Ingeniería de Documentos

Darwin Information Typing Architecture o *DITA* [7] es un estándar basado en *XML* desarrollado inicialmente por *IBM* y posteriormente adoptado por la *Organización para el Avance de Estándares de Información Estructurada (OASIS)* [6]. *DITA* se ha diseñado para la creación, gestión y publicación de documentación técnica, enfocándose en la modularidad y reutilización del contenido.

A continuación, algunos aspectos clave de *DITA*:

Principios de DITA
DITA organiza el contenido en unidades básicas llamadas "*tópicos*", cada uno abordando un tema específico y autosuficiente. Esto pretende facilitar la actualización y el mantenimiento de la documentación. Los tópicos pueden ser conceptos, tareas o referencias, permitiendo una estructura clara y consistente.

DITA permite reutilizar componentes de contenido en múltiples documentos, evitando la duplicación de información y asegurando la consistencia. Esto se realiza mediante la utilización de los *mapas DITA* que ayudan a estructurar y secuenciar los temas en documentos completos.

Además, *DITA* soporta la especialización para adaptarse a necesidades específicas y promueve la reutilización eficiente de contenido sin perder compatibilidad con las herramientas estándar. Se pueden crear nuevos tipos de documentos derivados de los tipos básicos. Al ser un estándar abierto, es compatible con diversas herramientas de edición y publicación, mejorando la eficiencia y calidad de la documentación técnica.

Los tópicos (*topic*) son una pieza importante en *DITA*. Los *topic* tienen un título y un contenido específico, lo suficientemente corto como para ser específica de un tema o la respuesta a una cuestión, pero lo suficientemente larga para tener sentido por sí misma y ser creada/editada como una unidad independiente. *DITA* define varios tipos de *topic* que se especializan en distintos propósitos. A continuación, se detalla cada uno de ellos:

1. *Topic*: El tipo de contenido más genérico en *DITA*. Es la base de la cual se derivan otros tipos más especializados.
2. *Concept*: Proporciona información para explicar un concepto, definir un término o describir por qué un usuario debe realizar una tarea. Es ideal para contenido que necesita una descripción general o teórica.
3. *Task*: Contiene información procedimental para realizar una tarea paso a paso. Es utilizado para guías de usuario, manuales de instrucciones y procedimientos operativos.
4. *Reference*: Contiene información detallada sobre ítems como comandos, mensajes de error, palabras reservadas en programación, etc. Se utiliza para contenido que requiere especificaciones detalladas o datos de referencia.

Un *topic* en *DITA* está compuesto de varias partes estructurales que organizan la información de manera coherente. Aquí se detalla la estructura general y los componentes clave:

1. *Prolog*: Contiene metadatos sobre el tópico, como el autor, la fecha de creación, y otros atributos relevantes.
2. *Title*: El título del *topic*, que debe ser claro y descriptivo.
3. *Body*: La parte principal del contenido. Dependiendo del tipo de *topic*, el cuerpo puede estar compuesto por diferentes elementos:
   - *Paragraphs* (`<p>`): Bloques de texto que presentan información.
   - *Lists* (`<ol>`, `<ul>`, `<dl>`): Listas ordenadas, no ordenadas y de definiciones para estructurar información en formato de lista.
   - *Sections*: Subdivisiones dentro del cuerpo para organizar contenido relacionado.
4. *Related-links*: Sección que contiene enlaces a otros tópicos o recursos relevantes.

A continuación se presenta un ejemplo simple de un *topic* en formato XML, mostrando su estructura básica:

```
<topic id="sample-topic">
  <title>Example of a DITA Topic</title>
  <prolog>
    <author>John Doe</author>
    <critdates>
      <created date="2024-07-07"/>
    </critdates>
  </prolog>
  <body>
```

```
    <p>This is a simple example of a DITA topic.</p>
    <section>
      <title>Section Title</title>
      <p>This is a paragraph within a section.</p>
      <ol>
        <li>First item in an ordered list</li>
        <li>Second item in an ordered list</li>
      </ol>
    </section>
  </body>
  <related-links>
    <link href="another-topic.dita">See also another topic</link>
  </related-links>
</topic>
```

Los *topics* en *DITA* tienen muchos beneficios como el contenido modular que se puede reutilizar y combinar en diferentes contextos, lo que mejora la eficiencia en la creación de contenido. La estructura de *DITA* permite la reutilización de *topics* en múltiples documentos, lo que reduce la redundancia y asegura la consistencia. Facilitan la actualización y el mantenimiento del contenido, ya que cada *topic* es independiente y puede ser actualizado sin afectar otros contenidos. Además, los *topics* pueden ser transformados y publicados en varios formatos, como HTML, PDF, ePub, y otros, usando herramientas de publicación como *DITA Open Toolkit*.

Los *topics* en *DITA* representan un enfoque estructurado y modular para la creación de contenido técnico y documentación. Al descomponer la información en unidades manejables y reutilizables, *DITA* facilita la gestión eficiente del contenido, asegurando su coherencia y calidad a lo largo del ciclo de vida de la documentación.

Los *ditamaps* en *DITA* son componentes clave que permiten la organización y estructura de la documentación técnica. Un *ditamap* es esencialmente un archivo XML que actúa como un contenedor y organizador de contenido en *DITA*, proporcionando una manera de agrupar y relacionar tópicos individuales en una estructura coherente y lógica.

Las funciones principales de los *ditamaps* son:

1. Organización del Contenido: Un *ditamap* permite agrupar tópicos en secciones, capítulos y otros niveles de jerarquía. Esto ayuda a los autores a estructurar la documentación de manera lógica y coherente.
2. Definición de Relaciones: Los *ditamaps* no solo organizan los temas, sino que también pueden definir relaciones entre ellos, como secuencias, dependencias y referencias cruzadas.
3. Facilitación de la Navegación: Al estructurar el contenido, los *ditamaps* facilitan la navegación para los usuarios finales, permitiendo que encuentren la información que necesitan de manera más eficiente.
4. Personalización del Contenido: Los *ditamaps* permiten la inclusión de diferentes versiones del contenido para distintos públicos o propósitos, lo que facilita la personalización y adaptación de la documentación.

La reutilización es uno de los principios fundamentales de *DITA* y los *ditamaps* son la base de la reutilización en la generación de documentos. Los *ditamaps* facilitan

la reutilización mediante el contenido modular dividiendo los temas en tópicos, los ditamaps permiten agrupar y reordenar estos tópicos según sea necesario, sin duplicar contenido. Los *ditampas* facilitan la referencia y uso múltiple de los tópicos ya que un solo archivo de tópico puede ser utilizado en diferentes contextos, manuales o publicaciones, lo que reduce la redundancia y el esfuerzo de mantenimiento. Con los *ditampas* existe la posibilidad de tener variantes de publicación ya que pueden incluir mecanismos como condicionales y filtrado, que permiten generar múltiples variantes de un mismo documento a partir de un conjunto común de temas. Esto es útil para crear versiones específicas para diferentes audiencias o plataformas. Por último los *ditamaps* garantizan un mantenimiento eficiente al tener el contenido organizado de manera modular y referenciado en múltiples *ditamaps*, cualquier actualización realizada en un tópico se refleja automáticamente en todas las instancias donde se usa. Esto hace que el mantenimiento sea más eficiente y consistente. En la Figura 2.9 se puede observar la autoría y gestión de contenido en *DITA*.

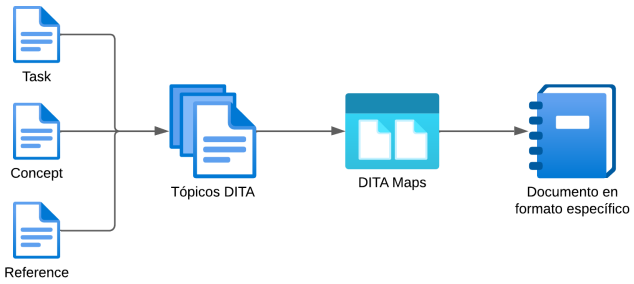

Figura 2.9: Metodología propuesta para la generación de documentación

A continuación se presenta un ejemplo simple de un *ditamap* en formato XML, mostrando su estructura básica:

```
<map>
  <title>User Guide</title>
  <topicref href="intro.dita"/>
  <topicref href="installation.dita"/>
  <topicref href="configuration.dita"/>
  <topicref href="usage.dita"/>
  <topicref href="troubleshooting.dita"/>
</map>
```

Beneficios de DITA

- Eficiencia: La modularidad y reutilización de contenido permiten una creación y mantenimiento más eficientes de la documentación.
- Consistencia: Garantiza la uniformidad en la terminología y el estilo a lo largo de los documentos.
- Escalabilidad: Facilita la gestión de grandes volúmenes de información técnica, haciendo el proceso escalable y sostenible.

Reutilización
En el marco de *DITA*, la reutilización de contenido se destaca como una de sus características más valiosas. Esta metodología permite que la información se divida en

componentes de contenido independientes llamados "*tópicos*", que pueden ser utilizados en diversos documentos sin necesidad de repetir el contenido. Por ejemplo, un tópico que detalla los pasos de instalación de un software puede ser incluido tanto en un manual de usuario como en una guía de referencia técnica, manteniendo la coherencia y precisión en ambos contextos. Esta estrategia no solo ahorra tiempo y esfuerzo en la creación y mantenimiento de la documentación, sino que también asegura que la terminología y el estilo sean uniformes en todos los documentos donde se aplique, lo que proporciona el valor de la consistencia en la documentación. Además, facilita la actualización de la información; al modificar un tópico reutilizado, todos los documentos que lo contienen se actualizan automáticamente, garantizando que la información sea siempre precisa y esté al día [44].

Capítulo 3

DITAOps Connect

DITAOps Connect es el nombre de la solución propuesta por este TFM a los problemas que se han expuesto en los capítulos anteriores consiste en demostrar la posibilidad de extraer contenido de *Azure DevOps* para generar documentación variable de forma automatizada utilizando el enfoque de la Ingeniería de Documentos, en concreto, *DITA* y la metodología *DPL*. Para esto se ha definido una estandarización para la estructura y configuración del contenido y se ha desarrollado un sitio web simple que procesa el contenido utilizando la estandarización y configuración definida para generar un documento PDF con la documentación del proyecto. Una consideración importante sobre la solución implementada es que se ha hecho en inglés, por una mejor integración y reutilización de la solución.

3.1. Azure DevOps y la Documentación Software

Azure DevOps es una plataforma integral para la gestión de proyectos de software que ofrece herramientas de colaboración, planificación y control de versiones, entre otras funcionalidades. Utilizar *Azure DevOps* para desarrollar un sistema de Ingeniería de Documentos implica aprovechar las capacidades de esta plataforma para mejorar la generación y actualización de documentación en proyectos de desarrollo de software. El uso de *Azure DevOps* en este TFM permitirá:

1. Contenido para la documentación en el contexto de los proyectos de desarrollo de software: *Azure DevOps* permite gestionar proyectos de desarrollo y observar cómo los equipos utilizan *work items* y *dashboards* para planificar, ejecutar y monitorear tareas.
2. Diseñar un modelo de documentos que se ajuste a las necesidades de documentación identificadas: Crear plantillas de documentos que incluyan las secciones y formatos necesarios para capturar la información relevante de los *dashboards* y *work items*. Definir estructuras dinámicas que permitan la integración de datos provenientes de *Azure DevOps*, asegurando que los documentos se actualicen automáticamente cuando cambien los datos en la plataforma.
3. Desarrollar un prototipo de sistema de Ingeniería de Documentos: Implementar un sistema que utilice las API de *Azure DevOps* para extraer información de *dashboards* y *work items* y generar documentos en formatos estándar (por ejemplo, PDF).

Utilizando *Azure DevOps* de esta manera, podremos obtener un sistema robusto y eficiente para la generación automática de documentación, mejorando así la eficiencia y la disponibilidad de documentos actualizados en proyectos de desarrollo de software.

3.2. Automatizando la generación de documentos con DITA

En el contexto del presente trabajo, se pretende integrar *DITA* con *Microsoft Azure DevOps* para mejorar la generación y mantenimiento de la documentación técnica en proyectos de desarrollo de software.
La sinergia entre estas dos tecnologías permitirá:

1. Extracción Eficiente de Datos: Utilizando las funcionalidades de *Azure DevOps*, se extraerán datos relevantes que se convertirán a una estructura *DITA*. Esto incluye la transformación de *work items*, *pipelines*, y otros elementos de *Azure DevOps* en módulos de contenido reutilizables.

2. Generación Automática de Documentación: La estructura modular de *DITA* permitirá la generación automática de documentación técnica precisa y actualizada, alineada con las mejores prácticas de la industria.

3. Mantenimiento Continuo: La capacidad de reutilización y especialización de *DITA* facilitará el mantenimiento continuo de la documentación, asegurando que se mantenga actualizada y relevante a lo largo del ciclo de vida del proyecto.

Integrar *DITA* en el flujo de trabajo de *Azure DevOps* no solo optimiza la generación de documentos, sino que también mejora la colaboración y comunicación entre los equipos de desarrollo. Al proporcionar una estructura coherente y centralizada para la documentación, se garantiza que toda la información técnica necesaria esté accesible y bien organizada, apoyando la eficiencia y calidad en el desarrollo de software. Este enfoque busca abordar los desafíos identificados en la generación manual de documentos, como el tiempo y esfuerzo requeridos, la propensión a errores humanos, la inconsistencia y la falta de escalabilidad [6] [7] [44].

3.3. Solución Propuesta

Como se ha mencionado en el capítulo anterior, la documentación es una fase importante en el proceso del desarrollo del software. La metodología habitual para la generación de documentación es de forma manual, esto resulta en un proceso tedioso, propenso a errores y que consume mucho tiempo. A continuación, se presenta un diagrama (Figura 3.1) de la metodología habitual para la generación de documentos dentro del proceso del desarrollo del software.

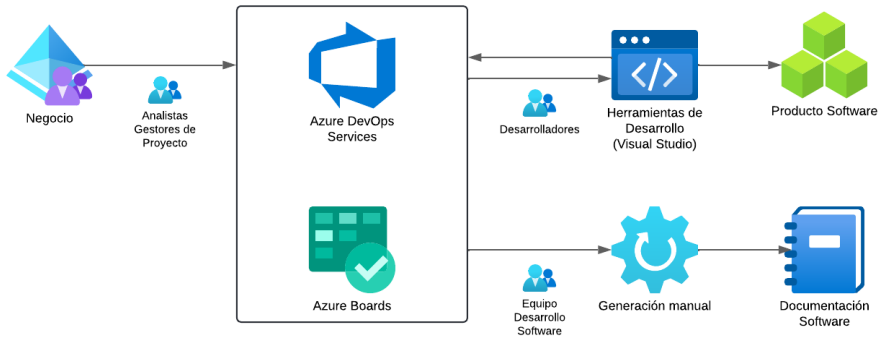

Figura 3.1: Metodología habitual para la generación de documentación

Tomando como punto de partida el diagrama de la metodología habitual (Figura 3.1), se ha elaborado un diagrama que ilustra gráficamente la solución propuesta para la automatización de la generación de documentación técnica a partir de *Azure DevOps* (Figura 3.2).

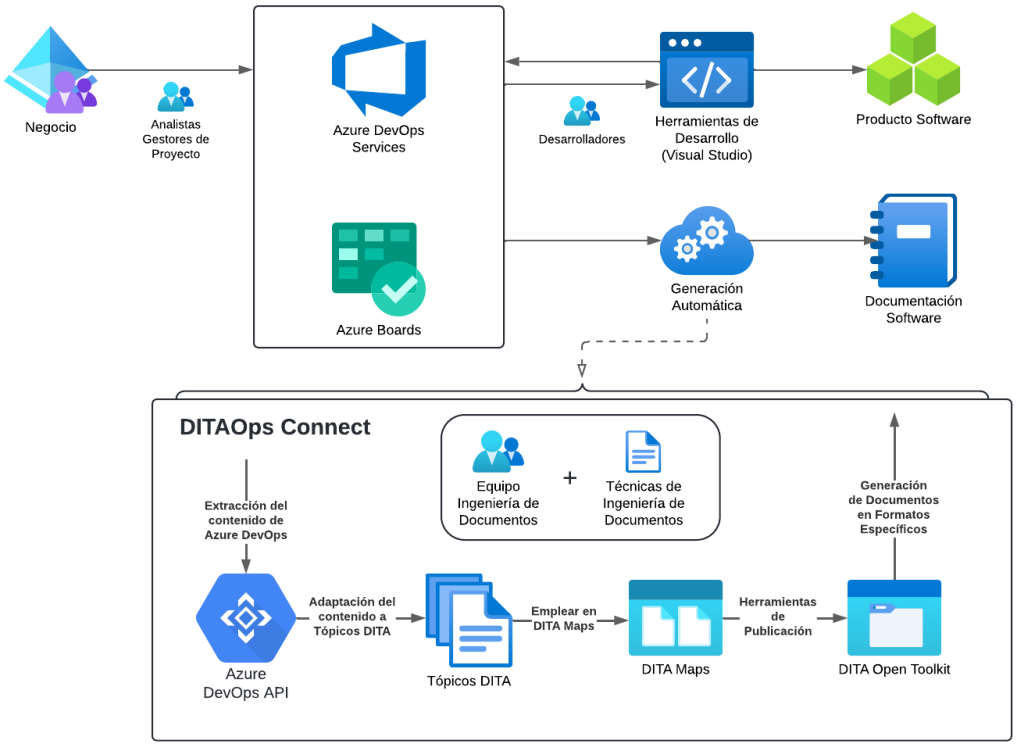

Figura 3.2: Metodología propuesta para la generación de documentación

De esta forma se provee una visión clara y detallada de los componentes principales y del flujo de datos, desde la extracción de contenido utilizando la API de *Azure DevOps* hasta la transformación y generación de documentos en formato *DITA* y su posterior conversión a PDF (u otros formatos) utilizando *Dita Open Tooklit*. La visualización del proceso permitirá una mejor comprensión de la arquitectura de la solución y de

cómo se integran las diferentes tecnologías y metodologías para alcanzar los objetivos planteados.

La arquitectura de la solución DITAOps Connect se puede observar en la Figura 3.3.

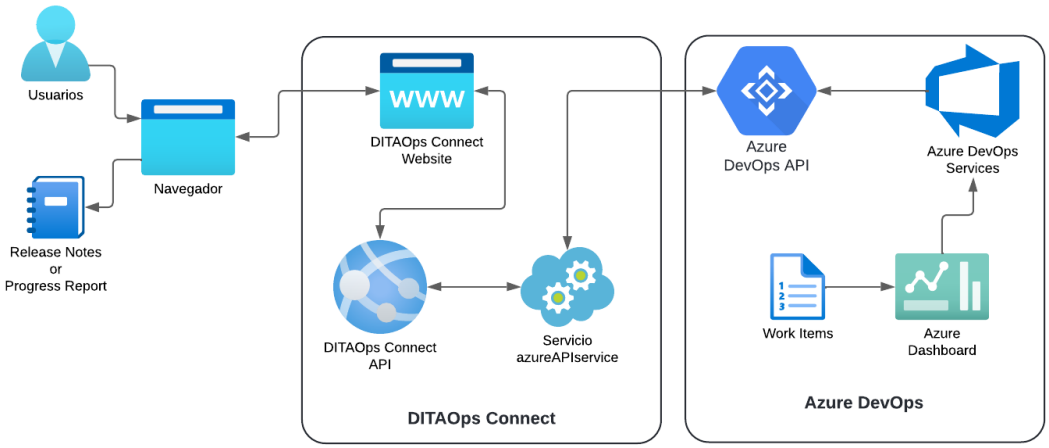

Figura 3.3: Arquitectura de DITAOps Connect

La arquitectura de DITAOps Connect, representa el diseñado propuesto para automatizar la generación de documentación técnica a partir de *Azure DevOps*. La arquitectura se divide en dos partes principales: DITAOps Connect y Azure DevOps.

DITAOps Connect

1. Usuarios y navegador: Los usuarios interactúan con el sistema a través de un navegador web el cual es el medio a través del cual los usuarios acceden al sitio web de DITAOps Connect.
2. DITAOps Connect Website: Este sitio web es la interfaz de usuario donde se configuran las opciones para generar la documentación. Permite a los usuarios seleccionar la organización, proyecto, tipo de documento a generar, y estado de los *work items* a considerar, que son necesarios para la generación de los documentos.
3. DITAOps Connect API:
   - La API de DITAOps Connect maneja las solicitudes del sitio web y se encarga de coordinar las operaciones necesarias para extraer y procesar los datos.
   - Esta API se comunica con el *Servicio azureAPIservice* para interactuar con la API de *Azure DevOps*.
4. Servicio azureAPIservice:
   - Este servicio es responsable de consumir la API de *Azure DevOps* para obtener los *work items* y otros datos relevantes de los *dashboards* de *Azure DevOps*.
   - Una vez obtenida la información, transforma los datos a *DITA* para generar los documentos.

Azure DevOps

1. Azure DevOps Services: Provee los servicios de *Azure DevOps*, incluyendo la gestión de proyectos, *work items*, *dashboards*, y otros datos.
2. Azure DevOps API: Una API que permite el acceso programático a los datos de *Azure DevOps*. El Servicio *azureAPIservice* utiliza esta API para extraer los datos necesarios.
3. Work Items y Azure Dashboard: Los *work items* y *dashboards* en *Azure DevOps* contienen la información que será extraída y transformada en documentación técnica.

Flujo de Trabajo

1. Interacción del Usuario: Los usuarios seleccionan las opciones y envían solicitudes desde el navegador a través del sitio web de DITAOps Connect.
2. Solicitud a la API: El sitio web envía las solicitudes a la DITAOps Connect API.
3. Extracción de Datos: La DITAOps Connect API coordina con el Servicio azureAPIservice para consumir la Azure DevOps API y obtener los *work items* y otros datos necesarios.
4. Transformación y Generación: El Servicio azureAPIservice transforma los datos extraídos a *DITA* y genera los documentos.
5. Entrega de Documentos: Los documentos generados, como *Release Notes* o *Progress Reports*, son devueltos al navegador, donde los usuarios pueden descargarlos o visualizarlos.

Puntos Clave

- Automatización: La arquitectura permite la generación automática de documentación a partir de datos en *Azure DevOps*, lo cual mejora la eficiencia y precisión de la documentación.
- Interacción Sencilla: Los usuarios pueden interactuar fácilmente con el sistema a través de un sitio web intuitivo.
- Transformación y Estandarización: Utiliza estándares como *DITA* para asegurar que los documentos generados sean consistentes y de alta calidad.
- Integración con *Azure DevOps*: Aprovecha las capacidades de *Azure DevOps* para obtener datos actualizados y relevantes directamente desde los *dashboards* y *work items*.

Esta arquitectura destaca por su capacidad de automatizar la generación de documentación técnica a partir de datos en *Azure DevOps*, utilizando un proceso estructurado y estandarizado que facilita la gestión de la información y mejora la eficiencia operativa.

Capítulo 4

Familia de Documentos

La definición de estructuras de documentos es crucial para asegurar una gestión organizada y coherente de la información a lo largo del ciclo de vida de un proyecto. La utilización del concepto de familia de documentos que introduce la metodología DPL [43] permite definir la estructura para que cada tipo de documento tenga un propósito claro y específico, facilitando la comunicación y colaboración entre los equipos involucrados. Además, ayuda a estandarizar la documentación, asegurando que todos los requisitos, diseños, pruebas, implementaciones y mantenimientos se registren y gestionen de manera sistemática.

4.1. Definiendo la familia de documentos

Diagrama de Clases
Usar un diagrama de clase para definir la familia de documentos es crucial debido a que proporciona una representación visual clara de la estructura y las relaciones entre los distintos documentos. Esto facilita la comprensión y organización de la información, asegurando que cada documento cumpla su propósito específico, mejorando la eficiencia y la coherencia en la gestión de documentos (Figura 4.1).

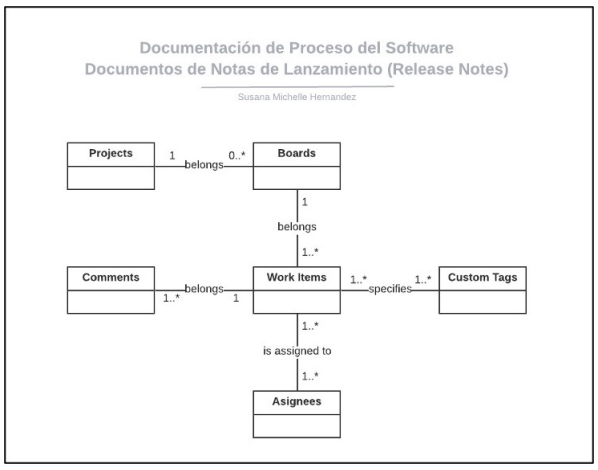

Figura 4.1: Diagrama de Clases para Release Notes y Progress Reports

Familia de Documentos
Considerando que el desarrollo de software posee una amplia gama de tópicos de documentación, se ha optado por enfocarse en las notas de publicación, o *Release Notes* y reporte del avance de las tareas, o *Progress Report*. Las *Release Notes* son un tipo crucial de documentación en el desarrollo de software que detalla los cambios, mejoras, corrección de errores y nuevas características en cada versión de una aplicación o sistema que se van publicando. Se ha optado por enfocar el presente trabajo en este tipo de documentación debido a que hay muchas oportunidades que se pueden aprovechar para mejorar la comunicación con los usuarios y mantener un registro claro y accesible de las actualizaciones del software.

La creación y mantenimiento de las *Release Notes* suele ser una tarea que consume mucho tiempo, ya que requiere la recopilación precisa y detallada de información de diferentes etapas y tareas realizadas en el desarrollo. Sin embargo, es evidente que este proceso puede automatizarse, lo que no solo ahorría tiempo, sino que también garantizaría la consistencia y precisión en la documentación. Esto resultaría en una mejor percepción y entendimiento de la documentación por parte de los usuarios y otras partes o entidades interesadas en el negocio.

Otro documento esencial por producir será el *Progress Report*. Este informe proporcionará una actualización detallada sobre los *work items*, incluyendo el tiempo invertido en su resolución y su comparativa en cuanto al tiempo estimado, facilitando el seguimiento del progreso del proyecto.

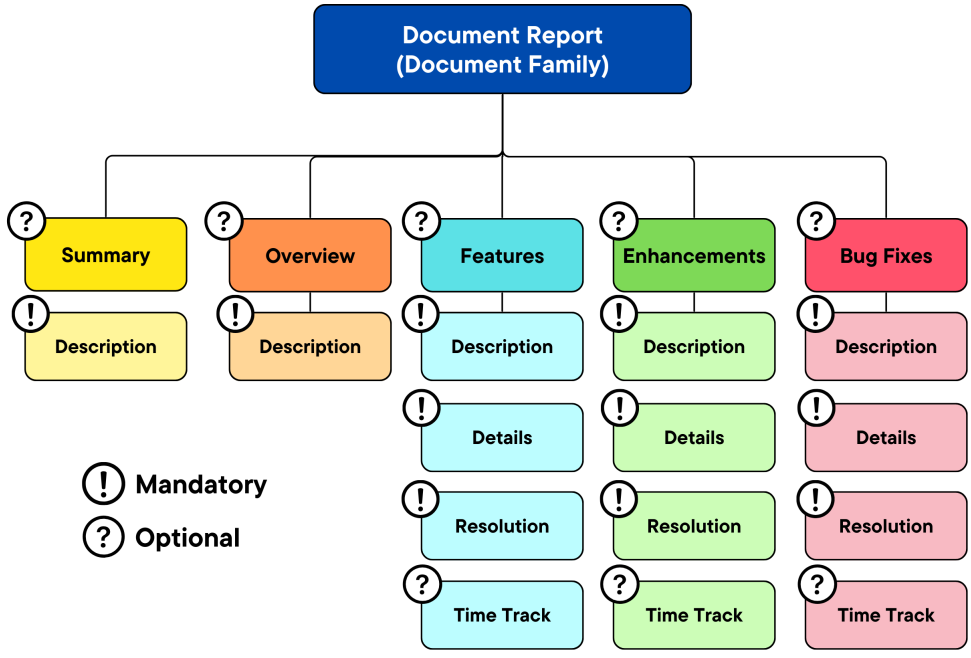

Figura 4.2: Estructura de la Familia de Documentos

El concepto de familia de documentos que introduce la metodología DPL [43] permite definir tópicos obligatorios y opcionales en la estructura para que cada tipo de documento tenga mayor variabilidad. Se ha definido una estructura para los *Document Reports*, como se muestra en la Figura 4.2, que usarán los *Progress Reports* y las *Release Notes*. Los documentos *Release Notes* utilizan únicamente los tópicos obliga-

torios definidos en la estructura y los documentos de *Progress Report* utilizan todos los tópicos opcionales aparte de los obligatorios tal y como se muestra en la Figura 4.3. Esta distinción permite variabilidad y la posibilidad de generar documentos distintos reutilizando información.

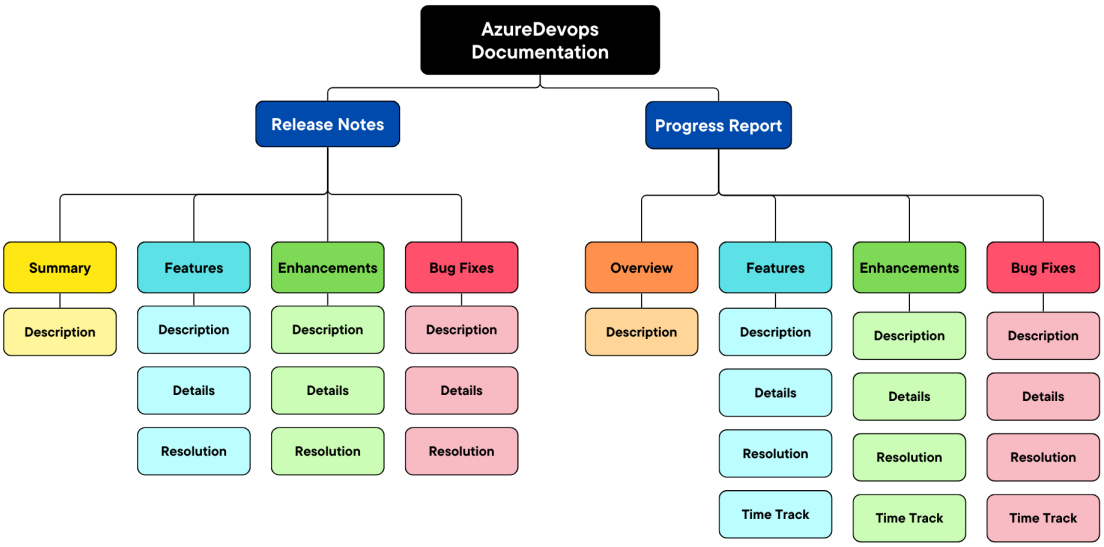

Figura 4.3: Estructura de los documentos *Release Notes* y *Progress Report*

A continuación, se detallan los tópicos principales para cada uno:

4.1.1. Release Notes

1. *Summary*: Este tópico ofrecerá un conteo general que incluirá el número total de *Features*, *Enhancements* y *Bugs* completados en el proyecto.
2. *Features*: Incluirá información detallada sobre los *work items* de tipo *Feature*. Se describirán las nuevas funcionalidades añadidas al proyecto, proporcionando una breve descripción de cada nueva funcionalidad que ha sido completada en el *Board* de *Azure DevOps*.
3. *Enhancements*: Esta sección estará dedicada a los *work items* de tipo *Tasks*. Aquí se enumerarán las mejoras y optimizaciones realizadas a desarrollos que han sido ya previamente implementados.
4. *Bug Fixes*: Se detallarán los *work items* de tipo *Bug* que han sido resueltos. Se incluirá una descripción de los problemas corregidos, el impacto de estos *bugs* en el sistema y detalles de su resolución.

4.1.2. Progress Report

1. *Overview*: Esta sección ofrecerá un conteo general de los *work items* en progreso o resueltos, proporcionando una visión cuantitativa de los elementos que están

actualmente en desarrollo o que han sido recientemente solucionados. Esto ayudará a comprender el volumen de trabajo pendiente y el avance del proyecto.

2. *Features*: En esta sección se incluirá información detallada sobre los *work items* de tipo *Features* que están en estado *Active* o *Resolved*. Se describirán las nuevas funcionalidades que están en desarrollo o que han sido completadas recientemente.

3. *Enhancements*: Esta sección estará dedicada a los *work items* de tipo Tasks en estado *Active* o *Resolved*. Aquí se enumerarán las mejoras y optimizaciones en curso o recientemente finalizadas, proporcionando una descripción de las tareas en desarrollo.

4. *Bug Fixes*: Se detallarán los *work items* de tipo *Bug* que están en estado *Active* o *Resolved*. Se incluirá una descripción de los problemas que están siendo abordados o que han sido resueltos recientemente, destacando el impacto de estos *bugs* en el sistema

Cada sección o tópico incluido en un *Progress Report*, brindara información sobre el tiempo que se ha invertido en cada uno de los *work ítems* con una comparativa del tiempo que se ha estimado inicialmente para su resolución.

4.1.3. DITA como soporte de la familia de documentos

Debido a que *DITA* es un estándar XML debemos establecer las estructuras que utilizaremos para la creación y gestión de la documentación, estructurado en tópicos los elementos de la familia de documentos. Definir una estructura, configuración y formato claros para la generación de documentos en *DITA* es esencial para mantener la consistencia, mejorar la eficiencia en la producción de contenido y asegurar que la información sea fácilmente accesible y comprensible para los usuarios finales

DITA Reference
La estructura XML de un DITA Reference Topic incluye elementos clave como `<reference>` para definir el tópico, `<title>` para el título del tópico, y `<refbody>` que contiene `<section>`, `<title>`, y `<p>` para organizar y presentar la información de manera clara. Esta estandarización facilita la creación, gestión y reutilización del contenido técnico, asegurando que la documentación sea consistente y fácilmente accesible.

La estructura base del *DITA reference* para la un tópico de tipo *Feature* es la siguiente [Anexo .1]:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="feature_report">
  <title>['fields']['System.Title']</title>
  <shortdesc>['fields']['System.Description']</shortdesc>
  <refbody>
    <section>
      <title>Requirement Type</title>
      <p>['fields']['Microsoft.VSTS.CMMI.RequirementType']</p>
    </section>
    <section>
      <title>Requirements</title>
```

```
      <p>['fields']['Custom.Requirements']</p>
    </section>
    <section>
      <title>Acceptance Criteria</title>
      <p>['fields']['Microsoft.VSTS.Common.AcceptanceCriteria']</p>
    </section>
    <section>
      <title>Priority</title>
      <p>['fields']['Microsoft.VSTS.Common.Priority']</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedDate']</
              stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedBy']['
              displayName']</stentry>
        </strow>
        <strow>
          <stentry>Reviewed By</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ReviewedBy']['
              displayName']</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.Resolution']</stentry>
        </strow>
      </simpletable>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.Effort']</p>
    </section>
    <section>
      <title>Original Estimate</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.OriginalEstimate']</p>
    </section>
    <section>
      <title>Completed Work</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.CompletedWork']</p>
    </section>
  </refbody>
</reference>
```

A continuación un ejemplo de como quedaría el *DITA reference* de *Features* después de la generación de un documento [Anexo .1]:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="feature_report">
  <title>Feature - Advanced Search Filters - Enhancing product search with
      filter options</title>
  <shortdesc>Design and implement a filtering system for the product search
       feature. Include filters for category, brand, price range, customer
      rating, and other relevant attributes. Ensure that the filters
      dynamically update the search results in real-time.   </shortdesc>
  <refbody>
    <section>
      <title>Requirement Type</title>
      <p>Functional</p>
    </section>
    <section>
      <title>Requirements</title>
      <p>FR-1: The system shall allow users to select one or more
          categories to filter the product search results. FR-2: The system
          shall dynamically update the search results to display only
          products that belong to the selected categories. FR-3: The system
          shall provide a list of brands for users to select from when
          filtering products. FR-4: The system shall update the search
          results to include only products from the selected brands. FR-5:
          The system shall allow users to specify a minimum and maximum
          price range to filter products. FR-6: The system shall display
          products within the specified price range in the search results.
          FR-7: The system shall enable users to filter products based on
          customer ratings (e.g., 1-5 stars). FR-8: The system shall update
          the search results to show products that have an average rating
          within the selected range.  </p>
    </section>
    <section>
      <title>Acceptance Criteria</title>
      <p>When a user selects categories, the search results shall display
          only products within those categories within 2 seconds. The system
           shall remember selected categories across page navigation. After
          selecting brands, the search results shall show only products from
           those brands within 2 seconds. Brand filter selections shall be
          maintained across browsing sessions if the "Remember my filters"
          option is enabled. Specifying a price range shall update the
          search results to show products within that range within 2 seconds
          . The system shall allow price range adjustments with a minimum
          increment of $1. Selecting a customer rating range shall update
          the search results to show products with ratings within the
          selected range within 2 seconds.    </p>
    </section>
    <section>
      <title>Priority</title>
      <p>2</p>
    </section>
    <section id="resolution_summary">
```

```
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>2024-06-28T20:53:11.6Z</stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>Susana Michelle Hernandez Chinchilla</stentry>
        </strow>
        <strow>
          <stentry>Reviewed By</stentry>
          <stentry>Susana Michelle Hernandez Chinchilla</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>Upon thorough testing and review, the development team
              confirms that all acceptance criteria for the Product Search
              Filtering System have been met. The system has been verified
              to perform according to the specified requirements, including
              real-time updates of search results based on user-selected
              filters for category, brand, price range, customer rating, and
               other relevant attributes. </stentry>
        </strow>
      </simpletable>
    </section>
  </refbody>
</reference>
```

DITA Map

El *DITA map* organiza los tópicos en una secuencia lógica y coherente, facilitando la navegación y reutilización del contenido. Se ha definido una estructura de *DITA map* para la familia de documentos, la cual se completará dinámicamente con información de los *work items* pertinentes. Además, se quitarán los componentes de documentos opcionales sin contenido, por ejemplo si un documento de *Release Notes*, no tiene *work items* de *Features* se quitaría esta sección del *DITA map*.

La estructura XML del *ditamap* incluye elementos como `<map>`, que actúa como contenedor principal, `<topicref>`, que referencia temas individuales, y `<title>`, que proporciona títulos para las secciones del mapa. Esta estructura permite la creación de una jerarquía de contenido, facilitando la navegación y el acceso a la información. Además, el *ditamap* soporta la inclusión de metadatos y enlaces cruzados, lo que mejora la gestión y reutilización del contenido técnico en grandes proyectos de documentación.

La estructura base del *DITA map* para la famiilia de documentos definida es la siguiente:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
```

```
<map>
    <title>[Document Family]</title>
    <topichead navtitle="Summary">
        <!-- Summary -->
    </topichead>
    <topichead navtitle="Overview">
        <!-- Overview -->
    </topichead>
    <topichead navtitle="Features">
        <!-- Features -->
    </topichead>
    <topichead navtitle="Enhancements">
        <!-- Enhancements -->
    </topichead>
    <topichead navtitle="Bug Fixes">
        <!-- Bug Fixes -->
    </topichead>
</map>
```

A continuación, un ejemplo de como quedaría el *DITA map* después de la generación de un documento [Anexo .1]:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
  <title>Release Notes</title>
  <topichead navtitle="Summary">
    <!-- Summary -->
    <topicref href="./topics/summary.dita"/>
  </topichead>
  <topichead navtitle="Features">
    <!-- Features -->
    <topicref href="./topics/features24.dita"/>
    <topicref href="./topics/features13.dita"/>
  </topichead>
  <topichead navtitle="Enhancements">
    <!-- Enhancements -->
    <topicref href="./topics/enhancement18.dita"/>
    <topicref href="./topics/enhancement17.dita"/>
    <topicref href="./topics/enhancement16.dita"/>
    <topicref href="./topics/enhancement15.dita"/>
    <topicref href="./topics/enhancement14.dita"/>
  </topichead>
  <topichead navtitle="Bug Fixes">
    <!-- Bug Fixes -->
    <topicref href="./topics/bugfixes23.dita"/>
    <topicref href="./topics/bugfixes22.dita"/>
    <topicref href="./topics/bugfixes21.dita"/>
    <topicref href="./topics/bugfixes20.dita"/>
  </topichead>
</map>
```

Capítulo 5

Método de Estandarización

Hemos explorado a detalle la plataforma *Microsoft Azure DevOps* y el estándar *DITA*, con este contexto en las siguientes secciones, se detallará la metodología de como se utilizarán los elementos y características de cada tecnología para desarrollar la solución. Además, se detallarán los tipos de consultas utilizadas, tales como la obtención de *work items* por tipo, y el estado de los mismos dependiendo de los parámetros introducidos por el usuario en la selección en el frontend . Además, se explicará cómo se construyen estas consultas, incluyendo ejemplos prácticos de código y las respuestas *JSON* recibidas desde la API. Esto permitirá comprender cómo se extraen los datos necesarios para transformarlos y estructurarlos en el formato *DITA*.

5.1. Estandarización de los work items de Azure DevOps

Reconociendo la importancia de la calidad y consistencia de la información en el proceso de desarrollo ágil, en este trabajo, se propone dedicar esfuerzos significativos a estructurar y estandarizar la manera en que se inserta la información en los *work items* del proceso ágil de *Microsoft Azure DevOps*. Esta labor es crucial para abordar el desafío de la inconsistencia de los datos que se extraen a través de la *API* [35], lo que podría comprometer la consistencia y confiabilidad de la documentación generada.

Mediante la implementación de estándares claros y la normalización de los campos de entrada, se busca asegurar que la información recopilada sea coherente y útil para la posterior generación de documentos, garantizando así que la documentación resultante sea precisa, completa y relevante para las necesidades del proyecto y del equipo de desarrollo.

5.1.1. Plantillas definidas para work items

Se proponen las siguientes plantillas para diferentes elementos de trabajo:

User Story
Las *User Stories* son descripciones breves y simples de una funcionalidad deseada del software escritas desde la perspectiva del usuario final. En un proyecto de desarrollo, se utilizan para capturar requisitos y expectativas del usuario, orientando al equipo sobre qué construir y por qué. Cada *User Story* incluye un título, una descripción y criterios de aceptación, que ayudan a definir cuándo una funcionalidad está completa

y cumple con las necesidades del usuario. Son esenciales en metodologías ágiles para priorizar el trabajo y mantener el enfoque en la entrega de valor al usuario. La estructura de la plantilla que se utilizará en la documentación generada para los *User Story* puede tener una estructura definida como la que se detalla en la Tabla 5.1.

Plantilla User Story

Title: *[User Story] [Feature/Component] - [Brief Description]*
Description:
As a *[user role]*: Describe the user role.
I want to *[goal]*: Clearly state the goal.
So that *[reason/benefit]*: Explain the benefit or reason.
Acceptance Criteria:
Bullet points listing the criteria for acceptance.
Priority: *[High/Medium/Low]*
Story Points: *[Numeric value]*
Tags: *[Relevant tags for categorization]*

Tabla 5.1: Plantilla *User Story* en *DevOps Connect*

Task

Las *Enhancements* en un proyecto de desarrollo de software son mejoras y nuevas capacidades a funcionalidades existentes que se agregan al sistema para satisfacer las necesidades y expectativas, ademas de mejorar la experiencia para los usuarios. En *Azure DevOps* se representan como *Tasks* e incluyen toda la información necesaria para la implementación de estas nuevas funcionalidades en los desarrollos de software. La estructura de la plantilla que se utilizará en la documentación generada para los *Tasks* o *Enhancements* puede tener una estructura definida como la que se detalla en la Tabla 5.2.

Plantilla Tasks/Enhancements

Title: *[Task] [Component/Feature] - [Brief Description]*
Description:
Detailed steps or actions required to complete the task.
Assigned To: *[Team Member]*
Due Date: *[Date]*
Priority: *[High/Medium/Low]*
Tags: *[Relevant tags for categorization]*
Resolved Date: *[YYYY-MM-DD]*
Resolution Summary:
    Resolution Date: *[YYYY-MM-DD*
    Resolved By: *[Name or of the person who resolved task]*
    Resolution: *[Any information related to the task resolution]*
Attachments: *[Links to any relevant attachments]*

Tabla 5.2: Plantilla *Task* en *DevOps Connect*

Bugs

Los *Bug Fixes* en un proyecto de desarrollo de software son correcciones de errores identificados en el sistema que afectan su funcionamiento y experiencia del usuario. Estas correcciones pueden incluir la solución de problemas varios. Cada corrección de errores se aborda para asegurar que el software funcione de manera correcta y eficiente, mejorando su estabilidad, rendimiento y fiabilidad. Los *Bug Fixes* son esenciales para mantener la calidad del software y garantizar una experiencia de usuario satisfactoria. La estructura de la plantilla que se utilizará se detalla a en la Tabla 5.3

Plantilla Bugs

Title: *[Bug] [Component/Feature] - [Brief Description]*
Description:
Steps to Reproduce: Numbered list of steps to reproduce the bug.
Severity: *[Critical/Major/Minor]*
Priority: *[High/Medium/Low]*
Resolution Summary:
    Root Cause: Summarize the findings on what caused the issue.
    Resolution Date/Time: *[YYYY-MM-DD] [hh:mm]*
    Resolved By: *[Name or Alias of the Resolver]*
    Reviewed By: *[Name or Alias of the Reviewer]*
    Resolved Reason: The issue has been resolved and confirmed.
    Workaround: A temporary solution has been implemented.

Tabla 5.3: Plantilla *Bugs* en *DevOps Connect*

Feature

Las *Features* en un proyecto de desarrollo de software son nuevas funcionalidades agregadas al sistema para mejorar su utilidad y satisfacer necesidades de los usuarios que no habian sido abordadas anteriormente. Cada nueva funcionalidad se diseña y desarrolla para añadir valor al software, optimizando su rendimiento, usabilidad y atractivo general. La estructura de la plantilla que se utilizará en la documentación generada para los *Features* puede tener una estructura definida como la que se detalla en la Tabla 5.4:

Plantilla Features

Title: *[Feature] [Component/Module] - [Brief Description]*
Description:
High-level overview of the feature.
Goals: What the feature aims to achieve.
Requirements: Detailed requirements of the feature.
Priority: [High/Medium/Low]
Tags: [Feature, Relevant tags for categorization]

Tabla 5.4: Plantilla *Features* en *DevOps Connect*

5.2. Transformado los datos a DITA

En esta sección se describe el proceso de transformación de los datos de *Azure DevOps* a *DITA*. Se detallan los pasos necesarios para convertir la información de proyectos y tareas de *Azure DevOps* en un formato modular y reutilizable, adecuado para *DITA*. Esta transformación facilita una gestión de contenido más eficiente y estructurada, mejorando la documentación y la colaboración entre equipos.

5.2.1. Tecnologías Utilizadas

Microsoft Azure DevOps API
La API de *Microsoft Azure DevOps* [35] es una herramienta poderosa que puede ser aprovechada para la generación automatizada de documentos técnicos en *DITA*. Esta API permite interactuar con los servicios de *Azure DevOps*, facilitando la extracción de datos y la transformación de estos en documentos estructurados y reutilizables.

WIQL - Work Item Query Languaje
WIQL (Work Item Query Language), es un lenguaje específico de *Azure DevOps* que permite realizar consultas estructuradas sobre los *work items*, facilitando la recuperación de información relevante para la generación automática de documentación [48].

Uso de la API para la generación de documentos en DITA
La API de *Azure DevOps* proporciona varios *endpoints* que son esenciales para la gestión de *work items*, los cuales pueden ser transformados en contenido *DITA*. A continuación, se describen algunos de los principales *endpoints* que han sido utilizados en este trabajo para este propósito.

- Obtener un work item específico (Figura 5.1)
  Method: `GET`
  Endpoint:
  https://dev.azure.com/organization/project/_apis/wit/workitems/id?api-version=7.0

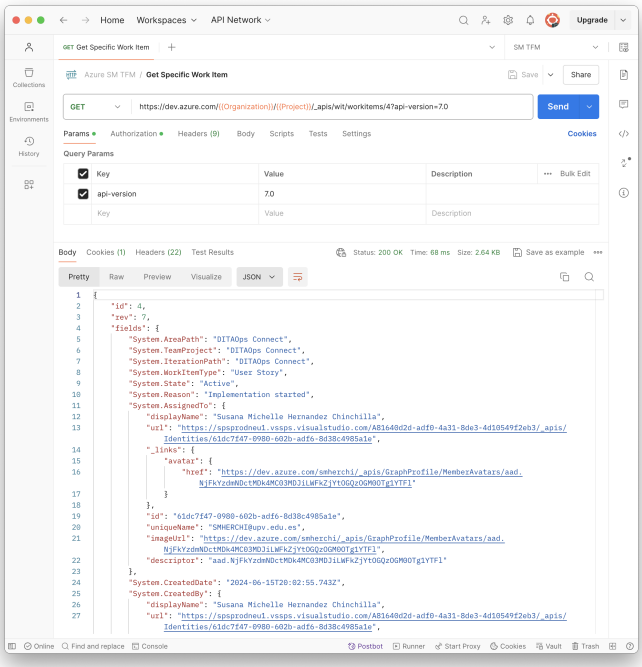

Figura 5.1: Obtener un work item específico

- Consultar work items con WIQL (Figura 5.2)
  Method: `POST`
  Endpoint: *https://dev.azure.com/organization/project/_apis/wit/wiql?api-version=7.0*

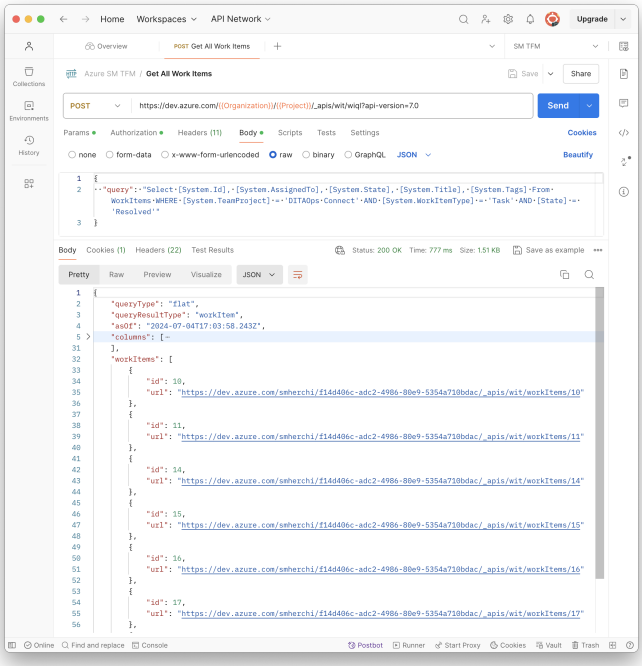

Figura 5.2: Consultar work items con WIQL

JSON a DITA
Se definirá una API específica que llevará a cabo la transformación de los resultados *JSON* obtenidos de la API de *Azure DevOps* a la estructura de *DITA* definida en el capítulo anterior. Esta API procesará los datos en formato *JSON*, extrayendo y reorganizando la información según las necesidades de *DITA*, lo que incluye la conversión de tareas, elementos de trabajo y descripciones en módulos de temas reutilizables y bien estructurados. La API garantizará que cada elemento se mapee correctamente a su correspondiente tipo de tópico en *DITA*, ya sean conceptos, tareas o referencias, facilitando así la creación de documentación técnica coherente y de alta calidad. Este proceso automatizado garantizará la eficiencia y precisión en la generación de documentos, asegurando una integración fluida entre *Azure DevOps* y *DITA*.

5.3. Diseño del proceso para la generación de tópicos DITA

Para cada tópico se realizará lo siguiente:

1. Obtener work items:
Se realizará una llamada a la API de Azure DevOps para obtener todos los *work items* según los parámetros proporcionados por el usuario. Se utilizará la petición para obtener la lista de *work items* mediante instrucciones *WIQL* [48], extrayendo los datos necesarios de los *work items* y organizándolos en el formato adecuado aceptado por *DITA*. Esto asegurará que las *Release Notes* generadas sean precisas, actualizadas y consistentes.

Cada tipo de elemento de *work item* se tendrá un WIQL definido.

WIQL - Features

```
{
    "query": "Select [System.Id], [System.AssignedTo], [System.State
       ], [System.Title], [System.Tags] From WorkItems WHERE [System
       .TeamProject] = 'projectName' AND [System.WorkItemType] = '
       Feature' AND [State] = 'STATUS'"
}
```

WIQL - Enhancements

```
{
    "query": "Select [System.Id], [System.AssignedTo], [System.State
       ], [System.Title], [System.Tags] From WorkItems WHERE [System
       .TeamProject] = 'projectName' AND [System.WorkItemType] = '
       Task' AND [State] = 'STATUS'"
}
```

WIQL - Bugs

```
{
    "query": "Select [System.Id], [System.AssignedTo], [System.State
       ], [System.Title], [System.Tags] From WorkItems WHERE [System
```

```
        .TeamProject] = '{{projectName}}' AND [System.WorkItemType] =
         'Bug' AND [State] = 'STATUS'"
}
```

2. Acceso a información completa del *work item*:
La petición inicial para obtener el listado únicamente devuelve el ID y la URL para cada *work item* (Figura 5.2). Es necesario acceder a cada URL individualmente para obtener la información completa del *work item*, esto implicará realizar una petición adicional para cada *work item* de manera iterativa.

3. Generación de Tópicos DITA:
Utiliando la información completa de cada *work item*, se generará un archivo `.dita` para cada *work item*, basado en la estructura previamente definida.

4. Agregar Tópicos al DITA Map:
Cada tópico *DITA* generado se agrega en un `<topicref>` al *DITA map* en la posición correspondiente según la estructura previamente definida.

Después de generar los tópicos y agregar sus referencias al *DITA map*:

5. Generación del documento PDF:
Una vez que el *DITA map* haya sido completado con todos los tópicos, se utilizará el *DITA Open Toolkit* [34] para generar el documento PDF a partir del *DITA map*.

6. Retorno del documento PDF:
La API devolverá el documento PDF como resultado de la petición, y el frontend se encargará de entregarlo al usuario final.

Capítulo 6

Implementación

La implementación desarrollada consiste en una API en *Python* que extrae contenido variable de *Azure DevOps* mediante su API, para generar documentación automatizada en formato PDF utilizando *DITA*. La API maneja la extracción de *work items*, los transforma en archivos `.dita` y un `.ditamap`, y luego genera un documento PDF. El frontend fue desarrollado en *Angular*, se utiliza para interactuar con esta API y permite a los usuarios seleccionar proyectos, configuraciones y ejecutar la generación de documentos. Este enfoque automatiza y estandariza la creación de documentación técnica, mejorando la eficiencia y consistencia del proceso.

6.1. DITAOps Connect Backend

Por su facilidad de uso se ha optado por utilizar *Python* como el lenguaje de programación esta API Rest. Para el desarrollo de la API, se utilizarán diversas herramientas y bibliotecas de Python.

La implementación realizada consiste en un backend simple en Python que actúa como una API RESTful, diseñada para gestionar la extracción y transformación de datos de *Azure DevOps* a la estructura DITA, generando documentos en formato PDF. La API consta de dos archivos principales: un archivo principal que maneja las peticiones a la API y un archivo de servicio que consume la API de *Azure DevOps* y se encarga de la generación de documentos en PDF mediante la utilización del *Dita Open Toolkit*.

6.1.1. Diseño de la API

La API contendrá una petición principal para generar los documentos en formato PDF, pero además también contendrá peticiones que funcionan como un proxy para la API de *Azure DevOps*. Se ha optado realizarlo de esta manera para poder alimentar el Frontend y tener los parámetros de forma dinámica para realizar la petición que genera el documento. Se ha enfocado en hacer el API lo más modular posible, por lo que los usuarios que la utilicen no necesariamente deben pertenecer a una misma organización en *Azure DevOps* ya cada usuario tendrá su propio *token* de la API de *Azure DevOps* almacenado en su perfil.

El API contiene cinco espacios de nombre (endpoints) para los diferentes elementos que maneja.

- Auth (`/api/v1/auth/`)
  Este *endpoint* contiene las peticiones de inicio y cierre de sesión.
- Users (`/api/v1/users/`)
  Este *endpoint* contiene las peticiones *CRUD* (Crear, Leer, Actualizar y Eliminar) para la gestión de usuarios.
- Projects (`/api/v1/projects/`)
  Este *endpoint* funciona como *proxy* para obtener los proyectos en *Azure DevOps* utilizando las peticiones de la API de *Azure DevOps*.
- Work Items (`/api/v1/workitems/`)
  Al igual que *Projects*, este *endpint* funciona como *proxy* para obtener los *work items* de un proyecto específico desde *Azure DevOps* mediante su API. Debido a que la petición para obtener la lista de *work items* directamente de la API de *Azure DevOps* no retorna toda la información completa para cada *work item* (Figura 5.2), este *endpoint* se encarga de incluir toda la información del *work item* para cada elemento de la lista.
- Documents (`/api/v1/documents/`)
  Este *endpoint* se encarga de generar los documentos en formato PDF dependiendo de los parámetros que se le envíen, considerando todas las especificaciones y configuraciones que se han expuesto a lo largo de este documento.

Además de los espacios de nombre mencionados, el API incluye características adicionales para mejorar la seguridad y el rendimiento. Se han implementado mecanismos de autenticación y autorización para asegurar que solo los usuarios autorizados accedan a los datos. Además, se ha optimizado para garantizar tiempos de respuesta rápidos y un manejo eficiente de las solicitudes.

A continuación, se describen las principales herramientas que se utilizarán para el desarrollo de la API:

- Python: Lenguaje de programación principal utilizado por su versatilidad, facilidad de uso y amplio catálogo de librerias.
- Swagger [45]: Herramienta de código abierto utilizada para diseñar, construir, probar, documentar y consumir servicios web *RESTful*. Swagger se utilizó para la creación de la documentación de la API y las pruebas durante el desarrollo.
- Azure DevOps Services REST API [35]: API de *Azure DevOps* utilizada para obtener información sobre los *work items*.
- Flask [46]: *Framework* micro para Python que permite la creación rápida de aplicaciones web. Flask se utilizará para desarrollar el servidor de la API, que manejará las solicitudes HTTP y las rutas de la aplicación.
- Flask_restx [46]: Extensión de Flask que facilita la construcción de servicios *RESTful*. Flask_restx se utilizará para estructurar y gestionar los *endpoints* y los *DTOs* de la API.
- Requests [47]: Biblioteca de Python para realizar solicitudes HTTP. Requests se utilizará para interactuar con la API de *Azure DevOps*.

6.1.2. Arquitectura de la API

Archivo Principal (*main.py*) Este archivo es el punto de entrada de la API RESTful. Utiliza frameworks como Flask para manejar las solicitudes HTTP. Las rutas definidas en este archivo permiten a los usuarios y al frontend de la solución interactuar con la API, enviar solicitudes para extraer datos de *Azure DevOps* y solicitar la generación de documentos.

Archivo de Servicio (*azureAPIservice.py*) Este archivo se encarga de la lógica principal del trabajo. Aquí se implementan las funciones que consumen la API de *Azure DevOps* para obtener el contenido de los *work items*. Una vez obtenidos los datos, se procesan y transforman en la estructura *DITA*. Esto incluye la creación de archivos `.dita` para cada *work item* y un archivo `.ditamap` que organiza estos tópicos de manera estructurada (Figura 6.1).

```
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 455-615-829
127.0.0.1 - - [04/Jul/2024 16:44:00] "OPTIONS /api/v1/documents/dita/smherchi?_pr
oject=DITAOps%20Connect&_status=Resolved&_documentFamily=Release%20Notes HTTP/1.1
" 200 -
https://dev.azure.com/smherchi/DITAOps Connect/_apis/wit/wiql?api-version=7.0
<Response [200]>
Building DITA topic for bugs
Building DITA topic for bugs
Building DITA topic for bugs
Building DITA topic for bugs
Building DITA topic for bugs
Building DITA topic for bugs
<Response [200]>
Building DITA topic for features
Building DITA topic for features
<Response [200]>
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for enhancements
Building DITA topic for summary
```

Figura 6.1: Consola del Backend

6.1.3. Proceso de Transformación

1. Extracción de Datos El servicio hace peticiones a la API de *Azure DevOps* utilizando peticiones HTTP. Se recupera la información detallada de los *work items*, incluyendo descripciones, estados, asignaciones, entre otros.

2. Transformación a DITA Los datos extraídos se estructuran en archivos XML `.dita`, que representan cada *work item* como un tópico independiente. Se genera un archivo `.ditamap` que actúa como índice, organizando y vinculando todos los tópicos `.dita`.

3. Generación de Documentos PDF Los archivos `.dita` y `.ditamap` se procesan usando *DITA Open Toolkit* para generar la documentación en formato PDF. El PDF resultante contiene todos los *work items* organizados y formateados según las especificaciones de *DITA* definidas en los capítulos anteriores.

6.1.4. Estructura de los archivos del proyecto backend

Los archivos generados se guardan en una carpeta con una estructura definida para asegurar que todos los elementos están correctamente organizados y son fácilmente accesibles por el servicio (Figura 6.2).
La estructura se detalla a continuación:

```
- main.py
/azure
    - azureAPIservice.py
    /dita
        /examples
        /formats
        /tfm
            - ReleaseNotes.ditamap
            /out
                - documentation.pdf
            /topics
                - topic1.dita
                - topic2.dita
```

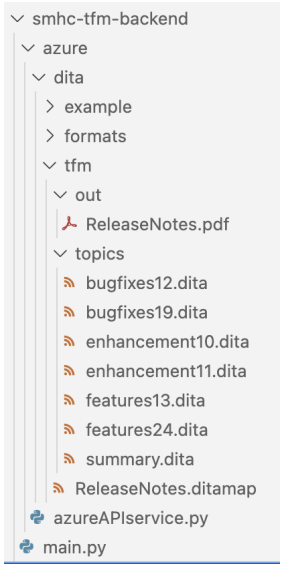

Figura 6.2: Estructura del Backend

6.2. DITAOps Connect FrontEnd

Adicional al API, se desarrollará un frontend para interactuar con el ella de manera intuitiva y eficiente. Este frontend permitirá a los usuarios acceder a las funcionalidades del API a través de una interfaz gráfica amigable, facilitando la generación de documentos sin la necesidad de herramientas complejas ni conocimientos técnicos avanzados.

Para el desarrollo de la interfaz de usuario, se utilizarán las siguientes herramientas:

- Angular [49]: es una plataforma y marco (framework) para construir aplicaciones web de una sola página con HTML y TypeScript.
- PrimeNG [50]: es una colección de componentes UI para Angular. Ofrece una amplia variedad de componentes como tablas de datos, entradas de formularios

y gráficos, con soporte para temas personalizables.

- PrimeFlex [51]: es una biblioteca de utilidades CSS de *PrimeTek*, diseñada para facilitar el diseño y el estilo en aplicaciones web. Proporciona clases CSS para crear rejillas (grids) responsivos, alinear elementos y gestionar el espaciado.

La interfaz diseñada para este proyecto (Figura 6.3), consiste en una página que permite al usuario indicar el tipo de documento que desea generar proporcionando los siguientes parámetros:

1. Organization: La organización en *Azure DevOps*
2. Project: El proyecto para el cual se desea generar la documentación
3. Document Family: El tipo de documento a generar, en este caso tenemos dos opciones, *Release notes* o *Progress Report*.
4. Status: El estado de los *work items* que se desean incluir en la documentación.

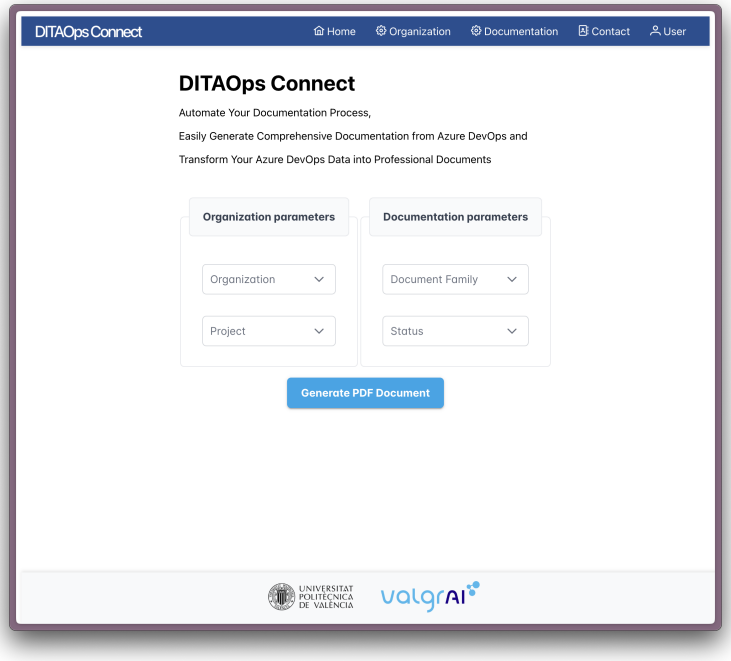

Figura 6.3: Interfaz Gráfica Frontend - *Home*

A parte de la página principal se realizaron páginas adicionales de las cuales algunas proporcionan a los usuarios una visión detallada y completa de la organización y los tipos de documentos que se pueden generar. Estas se muestran en las Figuras 6.4, 6.5, 6.6 y 6.6

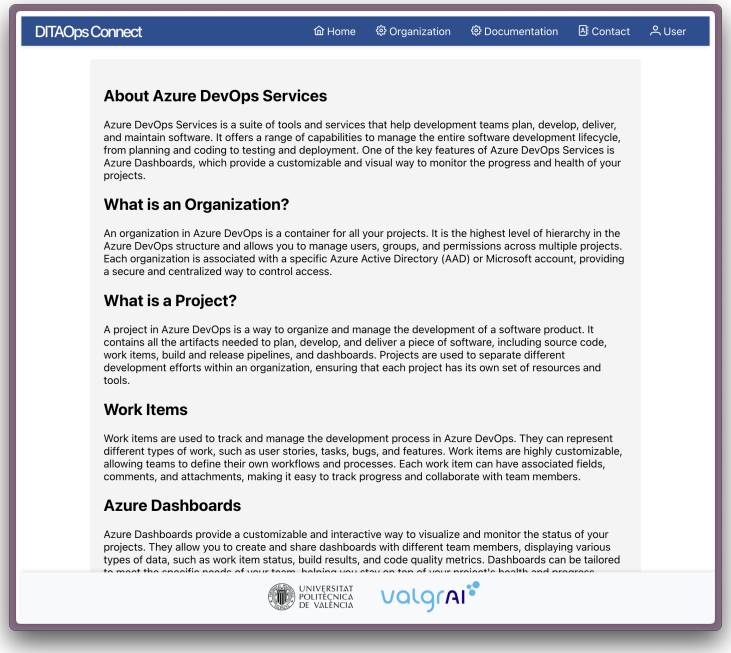

Figura 6.4: Interfaz Gráfica Frontend - *Organization*

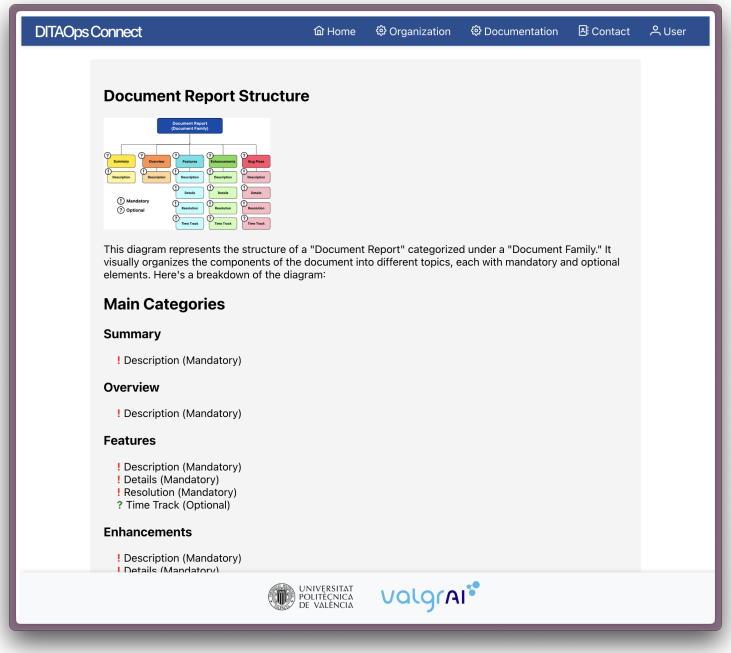

Figura 6.5: Interfaz Gráfica Frontend - *Documentation*

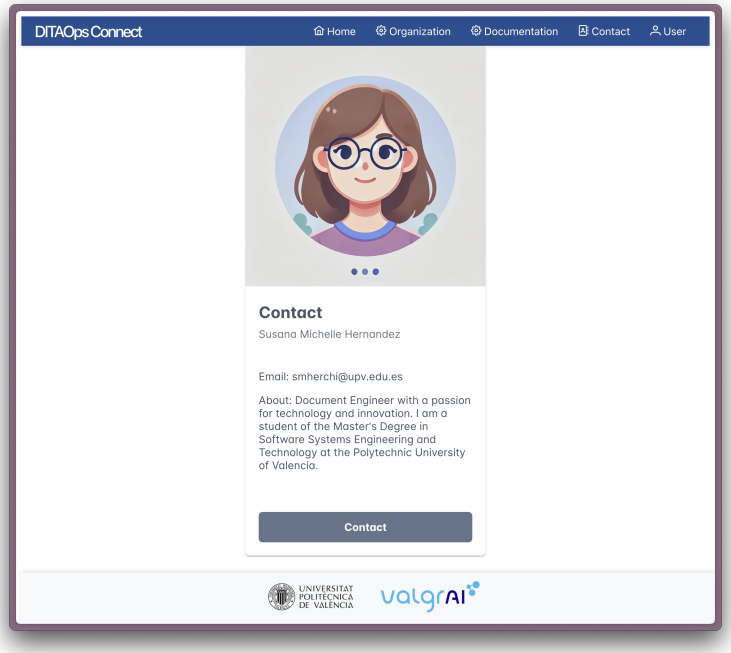

Figura 6.6: Interfaz Gráfica Frontend - *Contact*

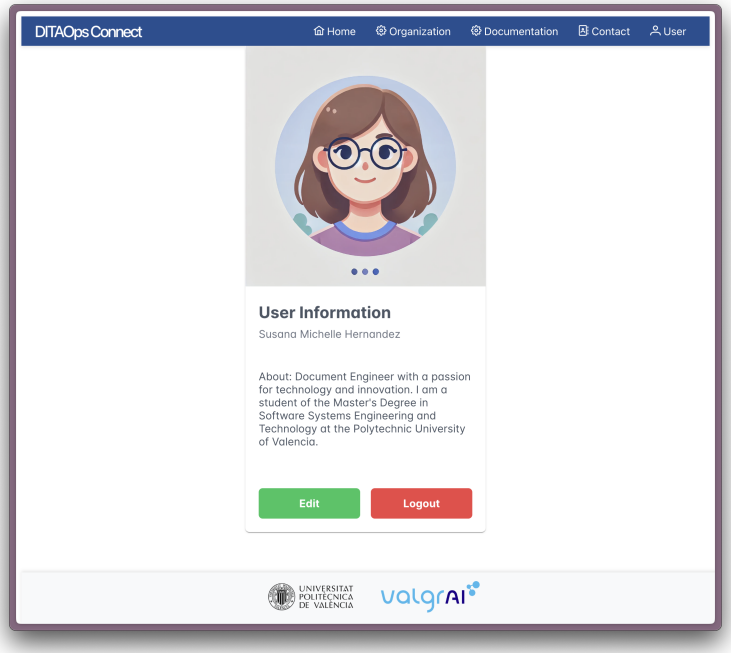

Figura 6.7: Interfaz Gráfica Frontend - *User Account*

Capítulo 7

Resultados

En este capítulo se presentan los resultados obtenidos en dos escenarios, donde se demuestra la efectividad de la solución para generar documentación técnica automatizada. En el ambos escenarios, se extrajeron y transformaron exitosamente los *work items* de un proyecto de desarrollo en *Azure DevOps*, generando un documento PDF consistente y actualizado que incluyen informes de progreso y notas de versión.

7.1. Escenario 1: Release Notes

El primer escenario consiste en generar el documento de *Release Notes*, por lo que se seleccionan las opciones de organización y proyecto en la sección *Organization parameters* y en la sección *Documentation parameter* en la opción *Document Family* se selecciona *Release Notes* y en la opción *Status* se debe seleccionar la opción *Resolved*. Tal y como se muestra en la Figura 7.1 .

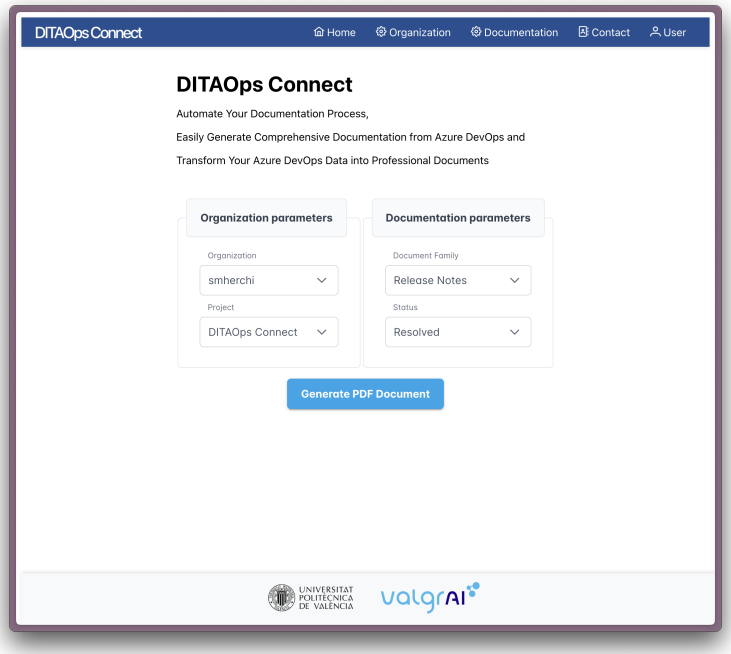

Figura 7.1: Opciones seleccionadas para generar *Release Notes*

La solución comenzará a realizar todos los pasos para generar el documento por lo que se mostrará un *spinner* en la pantalla, tal como se muestra en la Figura 7.2. Una vez termine se descargará automáticamente el documento.

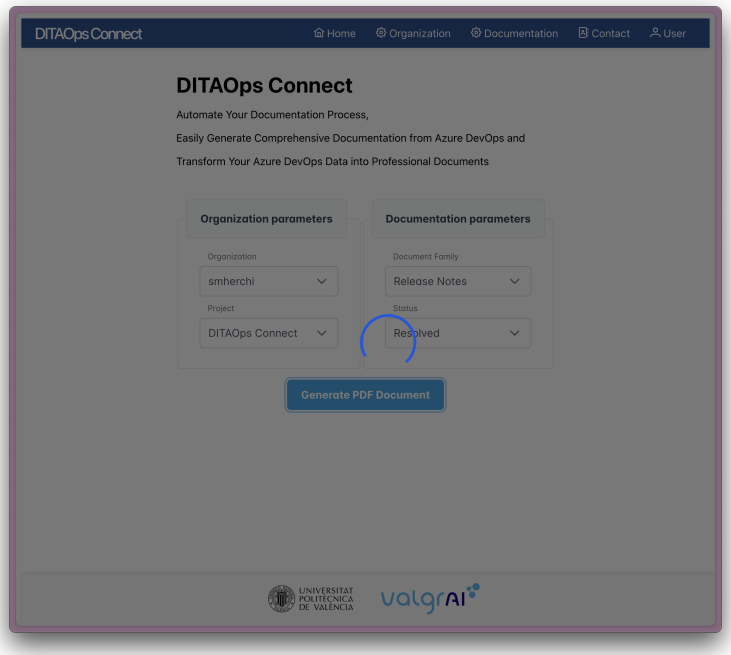

Figura 7.2: DITAOps Connect generando *Release Notes*

La solución proporciona un documento en PDF generado a partir de los parámetros establecidos por el usuario, como se muestra en la Figura 7.3 y se puede ver completo en el Anexo .2.1.1.

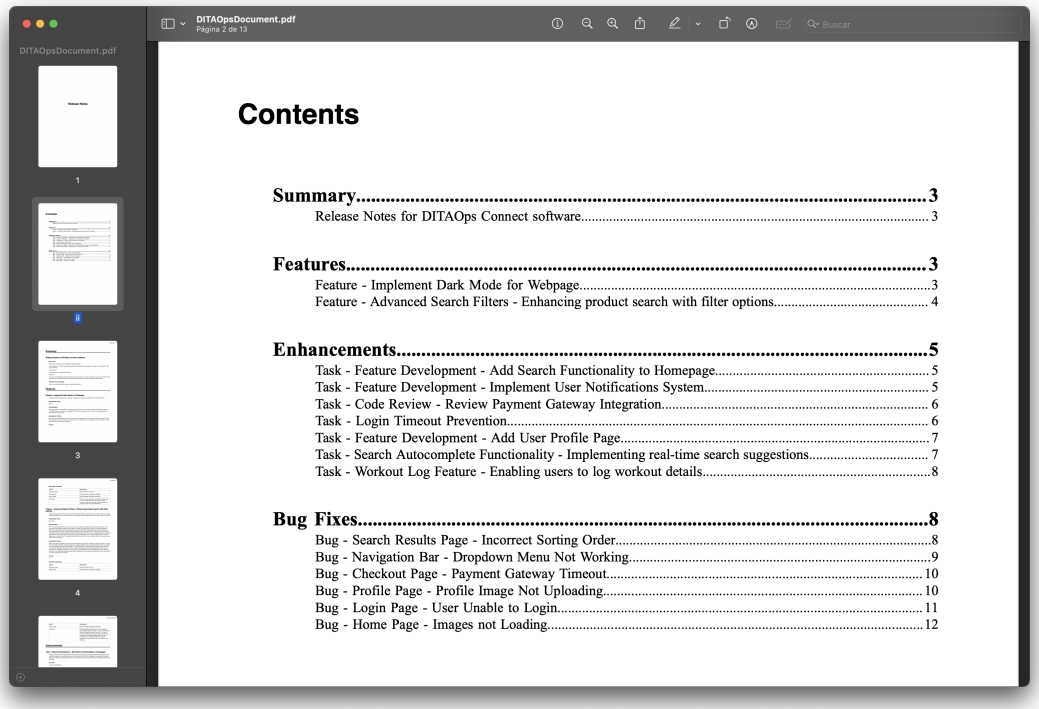

Figura 7.3: Documento de *Release Notes* generado

7.2. Escenario 2: Progress Report

El segundo escenario consiste en generar el documento de *Progress Report*, por lo que se seleccionan las opciones de organización y proyecto en la sección *Organization parameters* y en la sección *Documentation parameter* en la opción *Document Family* se selecciona *Progress Report* y en la opción *Status* se pueden seleccionar la opciones *Resolved* o *Active*. Tal y como se muestra en la Figura 7.4 .

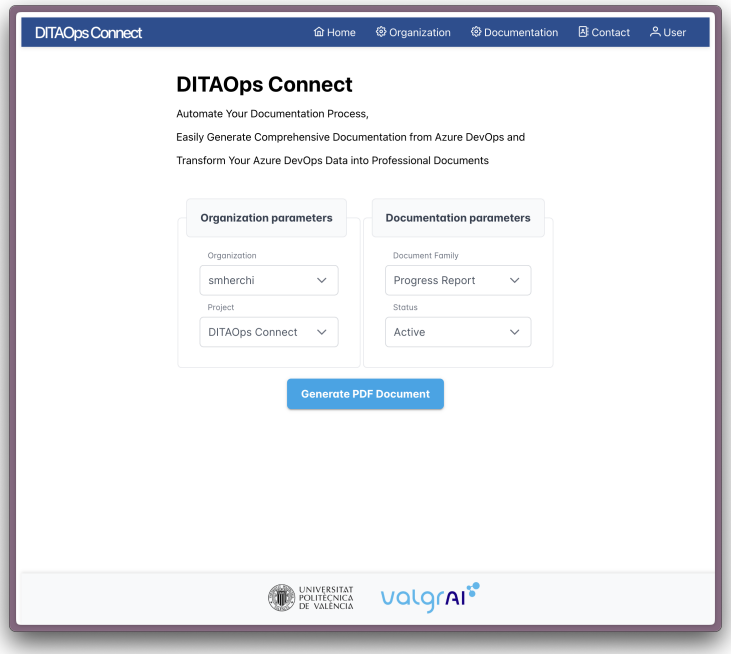

Figura 7.4: Opciones seleccionadas para generar *Progress Report*

La solución comenzará a realizar todos los pasos para generar el documento por lo que se mostrará un *spinner* en la pantalla, tal como se muestra en la Figura 7.5. Una vez termine se descargará automáticamente el documento.

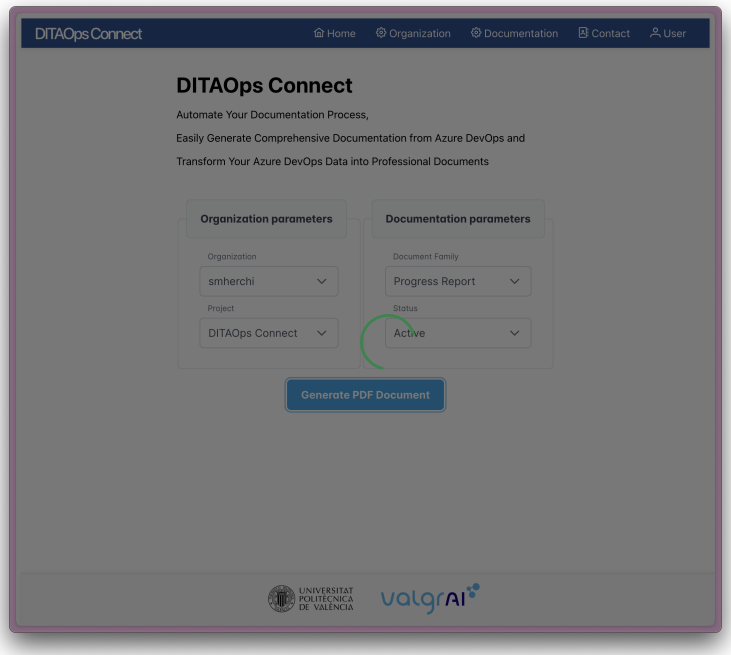

Figura 7.5: DITAOps Connect generando *Progress Report*

La solución proporciona un documento en PDF generado a partir de los parámetros establecidos por el usuario, como se muestra en la Figura 7.6 y se puede ver completo en el Anexo .2.1.2.

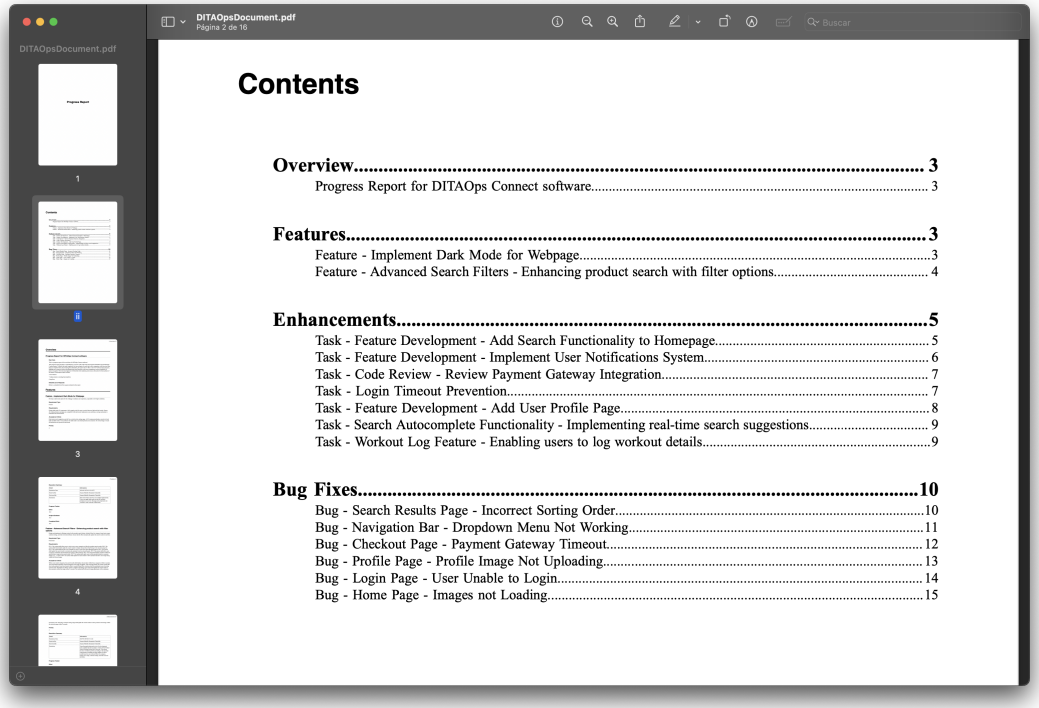

Figura 7.6: Documento de *Progress Report* generado

Capítulo 8

Conclusiones y Trabajos Futuros

A continuación, se presentan las conclusiones de este trabajo, y se proponen líneas futuras de investigación y desarrollo. Este capítulo ofrece una reflexión sobre los resultados obtenidos, evaluando el impacto y la efectividad de la solución implementada, y sugiere posibles mejoras y ampliaciones para continuar avanzando en la automatización y optimización de la generación de documentación técnica.

8.1. Conclusiones

Se han alcanzado los objetivos planteados en este TFM, logrando desarrollar un sistema de Ingeniería de Documentos que permite la generación automática de documentación coherente y actualizada a partir de los *dashboards* y *work items* de *Microsoft Azure DevOps*. Este sistema ha demostrado ser eficiente en la mejora del proceso de generación de documentos y en la disponibilidad de los mismos, facilitando la gestión de proyectos de desarrollo de software.

El análisis de las necesidades de documentación en proyectos que utilizan *Azure DevOps* permitió identificar los elementos clave que requieren documentación y las mejores prácticas de la industria. El estudio detallado de las funcionalidades y la estructura de datos de los *dashboards* y *work items* de *Azure DevOps* facilitó la identificación de información relevante que puede ser extraída para la generación de documentos.

Se diseñó un modelo de documentos adaptado a estas necesidades, integrando plantillas y estructuras que permiten la incorporación de datos dinámicos de *Azure DevOps*. Además, se desarrolló un prototipo funcional del sistema de Ingeniería de Documentos, que define y configura la estructura de los documentos con contenido dinámico, generando documentos estandarizados de alta calidad.

Este TFM no solo ha cumplido con los objetivos específicos planteados, sino que también ha sentado las bases para futuras innovaciones en el campo de la generación automática de documentación técnica, demostrando el potencial de la integración de herramientas avanzadas como *Azure DevOps* y *DITA* en la mejora de los procesos documentales en proyectos de software.

La adopción de *DITA* como marco para la gestión de la documentación técnica en proyectos de software ha demostrado ser una elección acertada y con buenos resultados,

a pesar de algunos desafíos iniciales. Al principio, la estructura de los documentos en *DITA* puede parecer confusa y restrictiva debido a que fuerza la estructura de contenido para cada tópico. Esta rigidez puede resultar un tanto restrictiva para los usuarios nuevos, quienes deben adaptarse a un enfoque más estructurado y categorizado de la documentación.

Sin embargo, una vez superada la curva de aprendizaje, los beneficios de *DITA* se vuelven muy valiosos. La estructura del contenido permite una reutilización eficiente, lo que significa que un solo tópico puede ser utilizado en múltiples documentos sin necesidad de replicar la información. Esto no solo ahorra tiempo y esfuerzo en la creación y mantenimiento de la documentación, sino que también garantiza una consistencia en todos los documentos generados.

Además, la capacidad de actualizar un tópico reutilizado en varios documentos asegura que cualquier cambio en la información se refleje automáticamente en todos los lugares donde se utiliza, manteniendo la documentación siempre precisa y actualizada, lo cual es muy valioso para cualquier tipo de negocio o proceso que hace uso del recurso de la documentación, sea del tipo que sea. Este aspecto es crucial en entornos de desarrollo de software donde los cambios son constantes y la exactitud de la información es vital.

Por otro lado, uno de los aspectos menos favorables de *DITA* es la limitada facilidad para personalizar el formato y la visualización de los documentos. La estructura rígida que impone *DITA* puede dificultar la adaptación de los documentos a estilos específicos o preferencias visuales particulares. Esta falta de flexibilidad en el diseño puede ser un inconveniente para organizaciones que valoran una presentación personalizada de su documentación técnica.

Además, es crucial destacar la importancia de estandarizar el proceso de llenado de información en los *work items* de *Azure DevOps*. La calidad y consistencia de la información en estos *work items* son vitales para garantizar que el contenido de la documentación generada sea preciso y relevante. Estandarizar el proceso de ingreso de datos ayuda a evitar la inconsistencia y los errores que podrían comprometer la integridad de la documentación final. Implementar estándares claros y normalizar los campos de entrada asegura que la información recopilada sea coherente y útil para la posterior generación de documentos, proporcionando así un recurso fiable y actualizado para el equipo de desarrollo y los interesados en el proyecto.

En resumen, aunque al principio *DITA* puede parecer complicado y algo restrictivo debido a su estructura específica, los beneficios a largo plazo, como la capacidad de reutilizar contenido, la consistencia en la documentación y la facilidad de actualización, son significativos. Implementar *DITA* en la gestión de documentación no solo mejora la eficiencia, sino también la precisión, dos elementos cruciales en el desarrollo de software. Es importante no dejar de lado las limitaciones de *DITA* en cuanto a la personalización del formato y la presentación visual. Es recomendable explorar soluciones adicionales que puedan suplir estas carencias y complementar la robustez de *DITA*. Asimismo, estandarizar el proceso de ingreso de información en los *work items* de *Azure DevOps* resulta fundamental para garantizar la calidad y confiabilidad de la documentación producida. Esta estandarización asegura que la información sea coherente y precisa, facilitando una documentación más efectiva y útil para todos los miembros del equipo de desarrollo.

8.2. Trabajos Futuros

El presente trabajo ofrece una sólida base para la generación de documentación técnica automatizada, pero tiene un gran potencial para ser ampliado y profundizado en futuros trabajos. Las oportunidades para futuras investigaciones y desarrollos incluyen:

- Explorar otros tipos de documentación relacionados con el desarrollo de software, como la documentación de pruebas, manuales de usuario, guías de instalación y documentación de la API, más allá del Swagger. Por ejemplo, incorporar módulos que automaticen la creación de manuales de usuario detallados o informes de pruebas puede mejorar significativamente la eficiencia y la calidad de la documentación.
- Expandir el enfoque para incluir una mayor variedad de familias de documentos. En lugar de enfocarse en los documentos técnicos básicos, se podría desarrollar plantillas y estructuras para documentos como guías de mejores prácticas o registros de decisiones técnicas. Esta ampliación facilitaría la gestión del conocimiento dentro de los equipos de desarrollo.
- Integración con herramientas adicionales aparte de Azure DevOps. Herramientas populares como JIRA, Zendesk, Confluence y GitHub también ofrecen valiosas fuentes de datos para la generación de documentación técnica. Al explorar estas plataformas, se podría proporcionar una solución más flexible y adaptable que se alinee con las diferentes preferencias y necesidades de los equipos de desarrollo y del negocio.
- Desarrollar un frontend más sofisticado y funcional. Un frontend mejorado podría permitir a los usuarios personalizar sus documentos de manera más eficiente, gestionar plantillas y obtener vistas previas en tiempo real de la documentación generada. Este desarrollo adicional haría que la herramienta sea más accesible y valiosa para los usuarios finales, facilitando una adopción más amplia y efectiva.

En resumen, las oportunidades para ampliar y mejorar este proyecto son bastante numerosas. Al ampliar el tipo de documentación, explorar la posibilidad de integración con otras herramientas y desarrollar un frontend más robusto, se puede crear una solución aún más robusta y versátil para la gestión de documentación técnica en el ámbito del desarrollo de software.

Bibliografía

[1] *Jira. https://www.atlassian.com/software/jira*

[2] *Azure DevOps Services, https://azure.microsoft.com/es-es/products/devops*

[3] *Godoy Sánchez, Danny Alexander. Generación Automática de Documentos de Requisitos en Proyectos de Software, 2010*

[4] *Felipe Ebert. From Transient Information to Persistent Documentation: Enhancing Software Documentation, 2020*

[5] *Michael Moser, Josef Pichler. POSITION PAPER: Documentation Generation from Annotated Source Code of Scientific Software, 2016*

[6] *OASIS. DITA 1.3 Specification. Retrieved from https://docs.oasis-open.org/dita/dita/v1.3/dita-v1.3-part1-overview.html, 2021*

[7] *Ann Rockley, Steve Manning and Charles Cooper. DITA 101: Fundamentals of DITA for Authors and Managers, 2009*

[8] *M. Carmen Penadés, José H. Canós, Marcos Borges, Manuel Llavador. Document product lines: Variability-driven document generation, 2010*

[9] *Roger S. Pressman. Ingeniería del software - Un enfoque práctico, 2010*

[10] *Grady Brooch, James Rumbaugh, Ivar Jacobson. El lenguaje unificado de modelado, 2006.*

[11] *Eloy Froufe Pérez. Principales metodologías de desarrollo software, 2023*

[12] *Alexandra Abuchar Porras. Metodologías ágiles para el desarrollo de software, 2023*

[13] *GitHub. https://github.com*

[14] *GitLab. https://gitlab.com*

[15] *Bitbucket. https://bitbucket.org/product*

[16] *Jenkins. https://www.jenkins.io*

[17] *Travis CI. https://www.travis-ci.com*

[18] *CircleCI. https://circleci.com*

[19] *Azure Pipelines. https://azure.microsoft.com/es-es/services/devops/pipelines/*

[20] *Trello. https://trello.com*

[21] *Asana. https://asana.com*

[22] *Azure Boards. https://azure.microsoft.com/es-es/services/devops/boards/*

[23] *Visual Studio. https://visualstudio.microsoft.com/es/*

[24] *Visual Studio Code. https://code.visualstudio.com*

[25] *IntelliJ IDEA. https://www.jetbrains.com/idea/*

[26] *Eclipse IDE. https://www.eclipse.org/ide/*

[27] *PyCharm. https://www.jetbrains.com/pycharm/*

[28] *Selenium. https://www.selenium.dev*

[29] *JUnit. https://junit.org/junit5/*

[30] *TestNG. https://testng.org/*

[31] *Postman. https://www.postman.com*

[32] *Confluence. https://www.atlassian.com/software/confluence*

[33] *SharePoint. https://www.microsoft.com/es-es/microsoft-365/sharepoint/collaboration*

[34] *DITA Open Toolkit 4.2, https://www.dita-ot.org/dev/*

[35] API rest versión 7.0 de Azure DevOps Services. https://learn.microsoft.com/es-es/rest/api/azure/devops/?view=azure-devops-rest-7.2&viewFallbackFrom=azure-devops-rest-7.0

[36] *M. Learn. About work items and work item types. Obtenido de https://learn.microsoft.com/en-us/azure/devops/boards/work-items/about-work-items?view=azure-devops&tabs=agile-process*

[37] *Adobe FrameMaker. https://www.adobe.com/es/products/framemaker.html*

[38] *Oxygen XML Editor. https://www.oxygenxml.com*

[39] *DocBook. https://docbook.org*

[40] *XMetaL. https://www.xmetal.com*

[41] *Adobe Experience Manager. https://www.adobe.com/es/marketing/experience-manager.html*

[42] Abel Gómez, M. Carmen Penadés, José H. Canós, Marcos Borges, Manuel Llavador. A framework for variable content document generation with multiple actors, 2014

[43] *Abel Gómez, M. Carmen Penadés, José H. Canós. Generación de Documentos con Contenido Variable en DPLfw. 2012*

[44] *M. Priestley. Introduction to DITA: A User Guide to the Darwin Information Typing Architecture. XML Press, 2018*

[45] *Swagger Documentation v3. https://swagger.io/docs/*

[46] *Flask Documentation v3. https://flask.palletsprojects.com/en/3.0.x/*

[47] *Requestes Python Library Documentation v2.32.3. https://requests.readthedocs.io/en/latest/*

[48] *Referencia de sintaxis del lenguaje de consultas de elemento de trabajo (WIQL). https://learn.microsoft.com/es-es/azure/devops/boards/queries/wiql-syntax?view=azure-devops*

[49] *Angular Documentation v18. https://angular.dev/overview*

[50] *PrimeNG Documentation v17.18.2 https://primeng.org/installation*

[51] *PrimeFlex Documentation v3.3.1. https://primeflex.org/installation*

[52] *Jaya Choudhury, B. Thushara. Software Documentation in a Globally Distributed Environment, 2014*

[53] *Md Athikul Islam, Rizbanul Hasan, Nasir U. Eisty. Documentation Practices in Agile Software Development: A Systematic Literature Review, 2023*

[54] *Lowe Wilsson. Automating and increasing efficiency of component documentation maintenance: A case study, 2022*

[55] *Robert J. Glushko, Tim McGrath. Document Engineering: Analyzing and Designing Documents for Business Informatics and Web Services, 2005*

[56] *Syntext Serna. https://www.syntext.com*

Anexos

.1. Anexo I

.1.1. Estructuras XML Definidas de Componentes DITA

.1.1.1. Summary

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="summary_report">
  <title>Release Notes for [project] software</title>
  <refbody>
    <section>
      <title>Dear User,</title>
      <p>Welcome to the latest release of [project] software!</p>
      <p>In this update, we have made significant improvements and
         addressed various issues to enhance your experience. This release
         includes:</p>
      <itemizedlist>
        <listitem>
          <p>[number of features] new features</p>
        </listitem>
        <listitem>
          <p>[number of enhancements] enhancements to existing
             functionalities</p>
        </listitem>
        <listitem>
          <p>[number of bugs] bug fixes</p>
        </listitem>
      </itemizedlist>
      <p>Our team has worked hard to ensure that our software continues to
         meet your needs and exceed your expectations. We appreciate your
         continued support and feedback, which will drive us to deliver the
          best possible product.</p>
    </section>
    <section>
      <title>Detailed List of Changes</title>
      <p>Below is a detailed list of the changes included in this release:<
         /p>
    </section>
  </refbody>
</reference>
```

.1.2. Overview

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="overview_report">
  <title>Progress Report for [project] software</title>
  <refbody>
    <section>
      <title>Dear User,</title>
      <p>This is a progress report of the work done for [project] software!
          </p>
      <p>This progress report provides a comprehensive overview of all work
           items and requests undertaken for the [project] project. It
          details the tasks completed, the time estimates for each task, and
           a comparison with the actual time taken to complete them. By
          presenting this information, the report aims to offer a clear view
           of the project progress, highlight any variances between the
          planned and actual timelines, and ensure transparency and
          accountability in project execution. This structured approach will
           facilitate better planning and resource allocation for future
          phases of the project.This progress report includes:</p>
      <itemizedlist>
        <listitem>
          <p>[number of features] new features</p>
        </listitem>
        <listitem>
          <p>[number of enhacnements] enhancements to existing
              functionalities</p>
        </listitem>
        <listitem>
          <p>[number of bugs] bug fixes</p>
        </listitem>
      </itemizedlist>
    </section>
    <section>
      <title>Detailed List of Requests</title>
      <p>Below is a detailed list of the requests included in this report:<
          /p>
    </section>
  </refbody>
</reference>
```

.1.3. Features

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="feature_report">
  <title>['fields']['System.Title']</title>
  <shortdesc>['fields']['System.Description']</shortdesc>
  <refbody>
    <section>
      <title>Requirement Type</title>
```

```
      <p>['fields']['Microsoft.VSTS.CMMI.RequirementType']</p>
    </section>
    <section>
      <title>Requirements</title>
      <p>['fields']['Custom.Requirements']</p>
    </section>
    <section>
      <title>Acceptance Criteria</title>
      <p>['fields']['Microsoft.VSTS.Common.AcceptanceCriteria']</p>
    </section>
    <section>
      <title>Priority</title>
      <p>['fields']['Microsoft.VSTS.Common.Priority']</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedDate']</
              stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedBy']['
              displayName']</stentry>
        </strow>
        <strow>
          <stentry>Reviewed By</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ReviewedBy']['
              displayName']</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.Resolution']</stentry>
        </strow>
      </simpletable>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.Effort']</p>
    </section>
    <section>
      <title>Original Estimate</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.OriginalEstimate']</p>
    </section>
    <section>
      <title>Completed Work</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.CompletedWork']</p>
```

```
    </section>
  </refbody>
</reference>
```

.1.1.4. Enhancements

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="enhancement_report">
  <title>['fields']['System.Title']</title>
  <shortdesc>['fields']['System.Description']</shortdesc>
  <refbody>
    <section>
      <title>Due Date</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.DueDate']</p>
    </section>
    <section>
      <title>Priority</title>
      <p>['fields']['Microsoft.VSTS.Common.Priority']</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedDate']</
              stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.ResolvedBy']['
              displayName']</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>['fields']['Microsoft.VSTS.Common.Resolution']</stentry>
        </strow>
      </simpletable>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.Effort']</p>
    </section>
    <section>
      <title>Original Estimate</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.OriginalEstimate']</p>
    </section>
```

```
    <section>
      <title>Completed Work</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.CompletedWork']</p>
    </section>
  </refbody>
</reference>
```

.1.1.5. BugFixes

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="bug_report">
  <title>['fields']['System.Title']</title>
  <shortdesc>['fields']['System.Description']</shortdesc>
  <refbody>
    <section id="reproduction_steps">
      <title>Steps to Reproduce</title>
      <p>['fields']['Microsoft.VSTS.TCM.ReproSteps']</p>
    </section>
    <section id="expected_result">
      <title>Expected Result</title>
      <p>['fields']['Custom.ExpectedResult']</p>
    </section>
    <section id="actual_result">
      <title>Actual Result</title>
      <p>['fields']['Custom.ActualResult']</p>
    </section>
    <section id="severity_priority">
      <title>Severity/Priority</title>
      <p>['fields']['Microsoft.VSTS.Common.Severity'] / ['fields']['
          Microsoft.VSTS.Common.Priority']</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <sectiondiv id="root_cause">
        <title>Root Cause</title>
        <p>['fields']['Microsoft.VSTS.CMMI.RootCause']</p>
      </sectiondiv>
      <sectiondiv id="resolution_details">
        <title>Resolution Details</title>
        <table>
          <title>Resolution Information</title>
          <tgroup cols="2">
            <colspec colname="c1" />
            <colspec colname="c2" />
            <thead>
              <row>
                <entry>Detail</entry>
                <entry>Information</entry>
              </row>
            </thead>
            <tbody>
              <row>
                <entry>Resolution Date/Time</entry>
```

```
                <entry>['fields']['Microsoft.VSTS.Common.ResolvedDate']</
                    entry>
              </row>
              <row>
                <entry>Resolution By</entry>
                <entry>['fields']['Microsoft.VSTS.Common.ResolvedBy']['
                    displayName']</entry>
              </row>
              <row>
                <entry>Reviewed By</entry>
                <entry>['fields']['Microsoft.VSTS.Common.ReviewedBy']['
                    displayName']</entry>
              </row>
              <row>
                <entry>Resolved Reason</entry>
                <entry>['fields']['Microsoft.VSTS.Common.ResolvedReason']</
                    entry>
              </row>
            </tbody>
          </tgroup>
        </table>
      </sectiondiv>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.Effort']</p>
    </section>
    <section>
      <title>Original Estimate</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.OriginalEstimate']</p>
    </section>
    <section>
      <title>Completed Work</title>
      <p>['fields']['Microsoft.VSTS.Scheduling.CompletedWork']</p>
    </section>
  </refbody>
</reference>
```

.1.1.6. DITA Map

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
    <title>[Document Family Type]</title>
    <topichead navtitle="Summary">
        <!-- Summary -->
        <!-- topicref added dynamically -->
    </topichead>
    <topichead navtitle="Overview">
        <!-- Overview -->
        <!-- topicref added dynamically -->
    </topichead>
```

```
    <topichead navtitle="Features">
        <!-- Features -->
        <!-- topicref added dynamically -->
    </topichead>
    <topichead navtitle="Enhancements">
        <!-- Enhancements -->
        <!-- topicref added dynamically -->
    </topichead>
    <topichead navtitle="Bug Fixes">
        <!-- Bug Fixes -->
        <!-- topicref added dynamically -->
    </topichead>
</map>
```

.1.2. Estructuras XML de Ejemplos de Componentes DITA generados

.1.2.1. Summary

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="summary_report">
  <title>Release Notes for DITAOps Connect software</title>
  <refbody>
    <section>
      <title>Dear User,</title>
      <p>Welcome to the latest release of DITAOps Connect software!</p>
      <p>In this update, we have made significant improvements and
         addressed various issues to enhance your experience. This release
         includes:</p>
      <itemizedlist>
        <listitem>
          <p>2 new features</p>
        </listitem>
        <listitem>
          <p>7 enhancements to existing functionalities</p>
        </listitem>
        <listitem>
          <p>6 bug fixes</p>
        </listitem>
      </itemizedlist>
      <p>Our team has worked hard to ensure that our software continues to
         meet your needs and exceed your expectations. We appreciate your
         continued support and feedback, which will drive us to deliver the
          best possible product.</p>
    </section>
    <section>
      <title>Detailed List of Changes</title>
      <p>Below is a detailed list of the changes included in this release:<
         /p>
    </section>
  </refbody>
</reference>
```

.1.2.2. Overview

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="overview_report">
  <title>Progress Report for DITAOps Connect software</title>
  <refbody>
    <section>
      <title>Dear User,</title>
      <p>This is a progress report of the work done for DITAOps Connect
          software!</p>
      <p>This progress report provides a comprehensive overview of all work
           items and requests undertaken for the DITAOps Connect project. It
           details the tasks completed, the time estimates for each task,
          and a comparison with the actual time taken to complete them. By
          presenting this information, the report aims to offer a clear view
           of the project progress, highlight any variances between the
          planned and actual timelines, and ensure transparency and
          accountability in project execution. This structured approach will
           facilitate better planning and resource allocation for future
          phases of the project.This progress report includes:</p>
      <itemizedlist>
        <listitem>
          <p>2 new features</p>
        </listitem>
        <listitem>
          <p>7 enhancements to existing functionalities</p>
        </listitem>
        <listitem>
          <p>6 bug fixes</p>
        </listitem>
      </itemizedlist>
    </section>
    <section>
      <title>Detailed List of Requests</title>
      <p>Below is a detailed list of the requests included in this report:<
          /p>
    </section>
  </refbody>
</reference>
```

.1.2.3. Features

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="feature_report">
  <title>Feature - Implement Dark Mode for Webpage</title>
  <shortdesc>Develop a dark mode option for the webpage to enhance user
      experience, especially in low-light conditions. </shortdesc>
  <refbody>
    <section>
      <title>Requirement Type</title>
```

```
      <p>Feature</p>
    </section>
    <section>
      <title>Requirements</title>
      <p>Design dark mode UI components. Add a toggle switch for users to
         switch between light and dark modes. Ensure all existing features
         and pages are compatible with dark mode. Implement a user
         preference saving mechanism to remember the selected mode.  </p>
    </section>
    <section>
      <title>Acceptance Criteria</title>
      <p>Dark mode can be toggled on and off via a switch on the settings
         page. All UI components display correctly in both light and dark
         modes. User preference for dark mode is saved and persists across
         sessions. No critical bugs or visual inconsistencies are present
         in dark mode.  </p>
    </section>
    <section>
      <title>Priority</title>
      <p>2</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>2024-06-30T18:23:23.027Z</stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>Susana Michelle Hernandez Chinchilla</stentry>
        </strow>
        <strow>
          <stentry>Reviewed By</stentry>
          <stentry>Susana Michelle Hernandez Chinchilla</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>Dark mode feature has been successfully implemented.
             Users can toggle dark mode on and off, and their preferences
             are saved. All pages and components are verified to work
             correctly in dark mode.  </stentry>
        </strow>
      </simpletable>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p>10.0</p>
    </section>
```

```
    <section>
      <title>Original Estimate</title>
      <p>10.0</p>
    </section>
    <section>
      <title>Completed Work</title>
      <p>10.0</p>
    </section>
  </refbody>
</reference>
```

.1.2.4. Enhancements

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="enhancement_report">
  <title>Task - Feature Development - Add User Profile Page</title>
  <shortdesc>Design the layout and functionality of the user profile page.
      Develop the frontend components for the profile page. Integrate the
      profile page with the backend API. Write unit and integration tests.
      Deploy the feature to the staging environment and test.  </shortdesc>
  <refbody>
    <section>
      <title>Due Date</title>
      <p>2024-07-06T00:00:00Z</p>
    </section>
    <section>
      <title>Priority</title>
      <p>2</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <simpletable>
        <sthead>
          <stentry>Detail</stentry>
          <stentry>Information</stentry>
        </sthead>
        <strow>
          <stentry>Resolution Date</stentry>
          <stentry>2024-06-30T17:50:33.083Z</stentry>
        </strow>
        <strow>
          <stentry>Resolved By</stentry>
          <stentry>Susana Michelle Hernandez Chinchilla</stentry>
        </strow>
        <strow>
          <stentry>Resolution</stentry>
          <stentry>The user profile page has been successfully implemented
              and tested. The feature is now live in the staging environment
              .  Profile page meets all design and functionality
              requirements. </stentry>
        </strow>
      </simpletable>
    </section>
```

```
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p />
    </section>
    <section>
      <title>Original Estimate</title>
      <p>10.0</p>
    </section>
    <section>
      <title>Completed Work</title>
      <p>10.0</p>
    </section>
  </refbody>
</reference>
```

.1.2.5. BugFixes

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE reference PUBLIC "-//OASIS//DTD DITA Reference//EN" "reference.
    dtd">
<reference id="bug_report">
  <title>Bug - Profile Page - Profile Image Not Uploading</title>
  <shortdesc>The profile image is not uploading. </shortdesc>
  <refbody>
    <section id="reproduction_steps">
      <title>Steps to Reproduce</title>
      <p>Navigate to the user profile page. Click on "Edit Profile."
         Attempt to upload a new profile image. Click "Save."  </p>
    </section>
    <section id="expected_result">
      <title>Expected Result</title>
      <p>The profile image should be successfully uploaded and displayed on
          the profile page. </p>
    </section>
    <section id="actual_result">
      <title>Actual Result</title>
      <p>The profile image fails to upload, and the user receives an error
         message "Image upload failed." </p>
    </section>
    <section id="severity_priority">
      <title>Severity/Priority</title>
      <p>2 - High / 2</p>
    </section>
    <section id="resolution_summary">
      <title>Resolution Summary</title>
      <sectiondiv id="root_cause">
        <title>Root Cause</title>
        <p>Communication Error</p>
      </sectiondiv>
      <sectiondiv id="resolution_details">
        <title>Resolution Details</title>
        <table>
```

```
          <title>Resolution Information</title>
          <tgroup cols="2">
            <colspec colname="c1" />
            <colspec colname="c2" />
            <thead>
              <row>
                <entry>Detail</entry>
                <entry>Information</entry>
              </row>
            </thead>
            <tbody>
              <row>
                <entry>Resolution Date/Time</entry>
                <entry>2024-06-30T18:06:42.02Z</entry>
              </row>
              <row>
                <entry>Resolution By</entry>
                <entry>Susana Michelle Hernandez Chinchilla</entry>
              </row>
              <row>
                <entry>Reviewed By</entry>
                <entry>Susana Michelle Hernandez Chinchilla</entry>
              </row>
              <row>
                <entry>Resolved Reason</entry>
                <entry>Fixed</entry>
              </row>
              <row>
                <entry>Resolution</entry>
                <entry>A misconfiguration in the server-side file upload
                    handler prevented the image from being saved correctly.
                    The issue has been resolved and confirmed through
                    testing.  </entry>
              </row>
            </tbody>
          </tgroup>
        </table>
      </sectiondiv>
    </section>
    <section>
      <title>Progress Tracker</title>
    </section>
    <section>
      <title>Effort</title>
      <p />
    </section>
    <section>
      <title>Original Estimate</title>
      <p />
    </section>
    <section>
      <title>Completed Work</title>
      <p />
    </section>
  </refbody>
</reference>
```

.1.2.6. DITA Map

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map>
  <title>Progress Report</title>
  <topichead navtitle="Overview">
    <!-- Overview -->
    <topicref href="./topics/overview.dita"/>
  </topichead>
  <topichead navtitle="Features">
    <!-- Features -->
    <topicref href="./topics/features24.dita"/>
    <topicref href="./topics/features13.dita"/>
  </topichead>
  <topichead navtitle="Enhancements">
    <!-- Enhancements -->
    <topicref href="./topics/enhancement18.dita"/>
    <topicref href="./topics/enhancement17.dita"/>
    <topicref href="./topics/enhancement16.dita"/>
    <topicref href="./topics/enhancement15.dita"/>
    <topicref href="./topics/enhancement14.dita"/>
    <topicref href="./topics/enhancement11.dita"/>
    <topicref href="./topics/enhancement10.dita"/>
  </topichead>
  <topichead navtitle="Bug Fixes">
    <!-- Bug Fixes -->
    <topicref href="./topics/bugfixes23.dita"/>
    <topicref href="./topics/bugfixes22.dita"/>
    <topicref href="./topics/bugfixes21.dita"/>
    <topicref href="./topics/bugfixes20.dita"/>
    <topicref href="./topics/bugfixes19.dita"/>
    <topicref href="./topics/bugfixes12.dita"/>
  </topichead>
</map>
```

.2. Anexo II

.2.1. Documentos Generados

.2.1.1. *Release Notes*

Release Notes

Contents

Summary **3**
Release Notes for DITAOps Connect software 3

Features **3**
Feature - Implement Dark Mode for Webpage 3
Feature - Advanced Search Filters - Enhancing product search with filter options 4

Enhancements **5**
Task - Feature Development - Add Search Functionality to Homepage 5
Task - Feature Development - Implement User Notifications System 5
Task - Code Review - Review Payment Gateway Integration 6
Task - Login Timeout Prevention 6
Task - Feature Development - Add User Profile Page 7
Task - Search Autocomplete Functionality - Implementing real-time search suggestions 7
Task - Workout Log Feature - Enabling users to log workout details 8

Bug Fixes **8**
Bug - Search Results Page - Incorrect Sorting Order 8
Bug - Navigation Bar - Dropdown Menu Not Working 9
Bug - Checkout Page - Payment Gateway Timeout 10
Bug - Profile Page - Profile Image Not Uploading 10
Bug - Login Page - User Unable to Login 11
Bug - Home Page - Images not Loading 12

Summary

Release Notes for DITAOps Connect software

Dear User,

Welcome to the latest release of DITAOps Connect software!

In this update, we've made significant improvements and addressed various issues to enhance your experience. This release includes:

2 new features

7 enhancements to existing functionalities

6 bug fixes

Our team has worked hard to ensure that our software continues to meet your needs and exceed your expectations. We appreciate your continued support and feedback, which will drive us to deliver the best possible product.

Detailed List of Changes

Below is a detailed list of the changes included in this release:

Features

Feature - Implement Dark Mode for Webpage

Develop a dark mode option for the webpage to enhance user experience, especially in low-light conditions.

Requirement Type

Feature

Requirements

Design dark mode UI components. Add a toggle switch for users to switch between light and dark modes. Ensure all existing features and pages are compatible with dark mode. Implement a user preference saving mechanism to remember the selected mode.

Acceptance Criteria

Dark mode can be toggled on and off via a switch on the settings page. All UI components display correctly in both light and dark modes. User preference for dark mode is saved and persists across sessions. No critical bugs or visual inconsistencies are present in dark mode.

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T18:23:23.027Z
Resolved By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolution	Dark mode feature has been successfully implemented. Users can toggle dark mode on and off, and their preferences are saved. All pages and components are verified to work correctly in dark mode.

Feature - Advanced Search Filters - Enhancing product search with filter options

Design and implement a filtering system for the product search feature. Include filters for category, brand, price range, customer rating, and other relevant attributes. Ensure that the filters dynamically update the search results in real-time.

Requirement Type

Functional

Requirements

FR-1: The system shall allow users to select one or more categories to filter the product search results. FR-2: The system shall dynamically update the search results to display only products that belong to the selected categories. FR-3: The system shall provide a list of brands for users to select from when filtering products. FR-4: The system shall update the search results to include only products from the selected brands. FR-5: The system shall allow users to specify a minimum and maximum price range to filter products. FR-6: The system shall display products within the specified price range in the search results. FR-7: The system shall enable users to filter products based on customer ratings (e.g., 1-5 stars). FR-8: The system shall update the search results to show products that have an average rating within the selected range.

Acceptance Criteria

When a user selects categories, the search results shall display only products within those categories within 2 seconds. The system shall remember selected categories across page navigation. After selecting brands, the search results shall show only products from those brands within 2 seconds. Brand filter selections shall be maintained across browsing sessions if the "Remember my filters" option is enabled. Specifying a price range shall update the search results to show products within that range within 2 seconds. The system shall allow price range adjustments with a minimum increment of $1. Selecting a customer rating range shall update the search results to show products with ratings within the selected range within 2 seconds.

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-28T20:53:11.6Z
Resolved By	Susana Michelle Hernandez Chinchilla

Detail	Information
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolution	Upon thorough testing and review, the development team confirms that all acceptance criteria for the Product Search Filtering System have been met. The system has been verified to perform according to the specified requirements, including real-time updates of search results based on user-selected filters for category, brand, price range, customer rating, and other relevant attributes.

Enhancements

Task - Feature Development - Add Search Functionality to Homepage

Design the search bar layout and determine its placement on the homepage. Develop the frontend components for the search bar. Integrate the search functionality with the backend to fetch results. Implement autocomplete and search suggestions. Write unit and integration tests for the search functionality. Deploy the feature to the staging environment for testing.

Due Date

2024-07-04T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T18:00:58.57Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	Search bar and autocomplete features are working correctly, and the search results are accurate.

Task - Feature Development - Implement User Notifications System

Design the notification system, including notification types and display methods. Develop backend logic to generate notifications based on user activities. Create frontend components to display notifications to users. Ensure real-time updates for notifications using WebSocket or similar technology. Write unit and integration tests for the notification system. Deploy the feature to the staging environment and conduct user testing.

Due Date

2024-06-30T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:59:08.443Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The user notifications system has been implemented, tested, and is operational in the staging environment. Notifications are being delivered in real-time, and user feedback indicates the system is user-friendly.

Task - Code Review - Review Payment Gateway Integration

Review the code for the newly integrated payment gateway. Ensure all security best practices are followed. Provide feedback and request changes if necessary. Approve the code for merging once all issues are addressed. Document the review process and findings.

Due Date

2024-07-29T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:55:58.327Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	Code review completed, and the payment gateway integration has been approved and merged.All identified issues were addressed, and security practices confirmed.

Task - Login Timeout Prevention

Investigate the root cause of the login timeout issue. Implement a solution to extend the session timeout duration. Test the fix in the development environment. Update any related documentation. Deploy the fix to the production environment.

Due Date

2024-07-04T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:53:39.127Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The login timeout control has been implemented by increasing the session timeout duration and optimizing session handling.

Task - Feature Development - Add User Profile Page

Design the layout and functionality of the user profile page. Develop the frontend components for the profile page. Integrate the profile page with the backend API. Write unit and integration tests. Deploy the feature to the staging environment and test.

Due Date

2024-07-06T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:50:33.083Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The user profile page has been successfully implemented and tested. The feature is now live in the staging environment. Profile page meets all design and functionality requirements.

Task - Search Autocomplete Functionality - Implementing real-time search suggestions

Develop the search autocomplete feature to provide real-time suggestions as users type. Ensure the autocomplete suggestions are relevant and based on popular search queries. Optimize the search algorithm to handle typos and suggest corrections.

Due Date

2024-11-14T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-29T11:02:55.613Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The search autocomplete functionality was developed, tested and approved by the user.

Task - Workout Log Feature - Enabling users to log workout details

Develop the workout log interface with input fields for type, duration, and intensity. Implement the ability to select from predefined workout types or enter custom workouts. Save workout data to the user's profile for history tracking.

Due Date

2024-08-22T00:00:00Z

Priority

3

Resolution Summary

Detail	Information
Resolution Date	2024-06-29T11:01:51.92Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The enhancement has been added in the new release.

Bug Fixes

Bug - Search Results Page - Incorrect Sorting Order

The search on the sublist in the Payments report is not sorting the results correctly.

Steps to Reproduce

Navigate to the search page. Perform a search with any keyword. Change the sorting order to "Price: Low to High."

Expected Result

Search results should be sorted with the lowest price items first.

Actual Result

Search results are not sorted correctly, with higher-priced items appearing before lower-priced ones.

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 1: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:18:01.18Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	An error in the sorting algorithm caused incorrect ordering of search results. The issue was resolved by correcting the sorting algorithm used for displaying search results.The issue has been resolved and confirmed through testing of various sorting options.

Bug - Navigation Bar - Dropdown Menu Not Working

The dropdown menu is not working.

Steps to Reproduce

Navigate to the homepage. Hover over the navigation bar to reveal dropdown menus. Click on any dropdown menu item.

Expected Result

The dropdown menu should expand, and clicking on an item should navigate to the corresponding page.

Actual Result

The dropdown menu does not expand, and clicking on an item does nothing.

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 2: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:15:24.143Z
Resolution By	Susana Michelle Hernandez Chinchilla

Detail	Information
Reviewed By	
Resolved Reason	Fixed
Resolution	The dropdown menu does not expand, and clicking on an item does nothing. The issue was resolved by fixing the JavaScript event handler for the dropdown menu. The issue has been resolved and confirmed through testing of the navigation bar.

Bug - Checkout Page - Payment Gateway Timeout

There is a timeout error showing up on the checkout page.

Steps to Reproduce

Add items to the cart. Proceed to the checkout page. Enter valid payment details. Click the "Pay Now" button.

Expected Result

Payment should be processed, and the user should receive a confirmation message.

Actual Result

The payment gateway times out, and the user receives an error message "Payment processing failed."

Severity/Priority

1 - Critical / 1

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 3: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:13:07.793Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	The issue was resolved by optimizing the API call to the payment gateway and increasing the timeout threshold.The issue has been resolved and confirmed through successful payment processing tests.

Bug - Profile Page - Profile Image Not Uploading

The profile image is not uploading.

Steps to Reproduce

Navigate to the user profile page. Click on "Edit Profile." Attempt to upload a new profile image. Click "Save."

Expected Result

The profile image should be successfully uploaded and displayed on the profile page.

Actual Result

The profile image fails to upload, and the user receives an error message "Image upload failed."

Severity/Priority

2 - High / 2

Resolution Summary

Root Cause

Communication Error

Resolution Details

Table 4: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:06:42.02Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	A misconfiguration in the server-side file upload handler prevented the image from being saved correctly.The issue has been resolved and confirmed through testing.

Bug - Login Page - User Unable to Login

User unable to login

Steps to Reproduce

Navigate to the login page. Enter valid credentials. Click the "Login" button.

Expected Result

User should be redirected to the dashboard upon entering valid credentials.

Actual Result

User remains on the login page and an error message "Invalid credentials" is displayed despite entering correct login details.

Severity/Priority

1 - Critical / 1

Resolution Summary

Root Cause

Specification Error

Resolution Details

Table 5: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:04:35.613Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	The issue was resolved by correcting the logic in the authentication service that was incorrectly validating user credentials.

Bug - Home Page - Images not Loading

The images from the homepage are not visible

Steps to Reproduce

Enter www.companyabc.com Scroll down to the gallery view Check the images on the photo grid

Expected Result

The images on the homepage of the website www.companyabc.com should be all visible

Actual Result

None of the images are visible in the homepage

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 6: Resolution Information

Detail	Information
Resolution Date/Time	
Resolution By	
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	

Detail	**Information**
Resolution	The tag was not closed. The tags were updated and closed.

.2.1.2. *Progress Report*

Progress Report

Contents

Overview **3**

Progress Report for DITAOps Connect software 3

Features **3**

Feature - Implement Dark Mode for Webpage 3

Feature - Advanced Search Filters - Enhancing product search with filter options 4

Enhancements **5**

Task - Feature Development - Add Search Functionality to Homepage 5

Task - Feature Development - Implement User Notifications System 6

Task - Code Review - Review Payment Gateway Integration 7

Task - Login Timeout Prevention 7

Task - Feature Development - Add User Profile Page 8

Task - Search Autocomplete Functionality - Implementing real-time search suggestions 9

Task - Workout Log Feature - Enabling users to log workout details 9

Bug Fixes **10**

Bug - Search Results Page - Incorrect Sorting Order 10

Bug - Navigation Bar - Dropdown Menu Not Working 11

Bug - Checkout Page - Payment Gateway Timeout 12

Bug - Profile Page - Profile Image Not Uploading 13

Bug - Login Page - User Unable to Login 14

Bug - Home Page - Images not Loading 15

Overview

Progress Report for DITAOps Connect software

Dear User,

This is a progress report of the work done for DITAOps Connect software!

This progress report provides a comprehensive overview of all work items and requests undertaken for the DITAOps Connect project. It details the tasks completed, the time estimates for each task, and a comparison with the actual time taken to complete them. By presenting this information, the report aims to offer a clear view of the project's progress, highlight any variances between the planned and actual timelines, and ensure transparency and accountability in project execution. This structured approach will facilitate better planning and resource allocation for future phases of the project.This progress report includes:

2 new features

7 enhancements to existing functionalities

6 bug fixes

Detailed List of Requests

Below is a detailed list of the requests included in this report:

Features

Feature - Implement Dark Mode for Webpage

Develop a dark mode option for the webpage to enhance user experience, especially in low-light conditions.

Requirement Type

Feature

Requirements

Design dark mode UI components. Add a toggle switch for users to switch between light and dark modes. Ensure all existing features and pages are compatible with dark mode. Implement a user preference saving mechanism to remember the selected mode.

Acceptance Criteria

Dark mode can be toggled on and off via a switch on the settings page. All UI components display correctly in both light and dark modes. User preference for dark mode is saved and persists across sessions. No critical bugs or visual inconsistencies are present in dark mode.

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T18:23:23.027Z
Resolved By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolution	Dark mode feature has been successfully implemented. Users can toggle dark mode on and off, and their preferences are saved. All pages and components are verified to work correctly in dark mode.

Progress Tracker

Effort

10.0

Original Estimate

10.0

Completed Work

10.0

Feature - Advanced Search Filters - Enhancing product search with filter options

Design and implement a filtering system for the product search feature. Include filters for category, brand, price range, customer rating, and other relevant attributes. Ensure that the filters dynamically update the search results in real-time.

Requirement Type

Functional

Requirements

FR-1: The system shall allow users to select one or more categories to filter the product search results. FR-2: The system shall dynamically update the search results to display only products that belong to the selected categories. FR-3: The system shall provide a list of brands for users to select from when filtering products. FR-4: The system shall update the search results to include only products from the selected brands. FR-5: The system shall allow users to specify a minimum and maximum price range to filter products. FR-6: The system shall display products within the specified price range in the search results. FR-7: The system shall enable users to filter products based on customer ratings (e.g., 1-5 stars). FR-8: The system shall update the search results to show products that have an average rating within the selected range.

Acceptance Criteria

When a user selects categories, the search results shall display only products within those categories within 2 seconds. The system shall remember selected categories across page navigation. After selecting brands, the search results shall show only products from those brands within 2 seconds. Brand filter selections shall be maintained across browsing sessions if the "Remember my filters" option is enabled. Specifying a price range shall update the search results to show products within that range within 2 seconds. The system shall allow price range adjustments with a minimum

increment of $1. Selecting a customer rating range shall update the search results to show products with ratings within the selected range within 2 seconds.

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-28T20:53:11.6Z
Resolved By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolution	Upon thorough testing and review, the development team confirms that all acceptance criteria for the Product Search Filtering System have been met. The system has been verified to perform according to the specified requirements, including real-time updates of search results based on user-selected filters for category, brand, price range, customer rating, and other relevant attributes.

Progress Tracker

Effort

Original Estimate

Completed Work

Enhancements

Task - Feature Development - Add Search Functionality to Homepage

Design the search bar layout and determine its placement on the homepage. Develop the frontend components for the search bar. Integrate the search functionality with the backend to fetch results. Implement autocomplete and search suggestions. Write unit and integration tests for the search functionality. Deploy the feature to the staging environment for testing.

Due Date

2024-07-04T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T18:00:58.57Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	Search bar and autocomplete features are working correctly, and the search results are accurate.

Progress Tracker

Effort

Original Estimate

Completed Work

Task - Feature Development - Implement User Notifications System

Design the notification system, including notification types and display methods. Develop backend logic to generate notifications based on user activities. Create frontend components to display notifications to users. Ensure real-time updates for notifications using WebSocket or similar technology. Write unit and integration tests for the notification system. Deploy the feature to the staging environment and conduct user testing.

Due Date

2024-06-30T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:59:08.443Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The user notifications system has been implemented, tested, and is operational in the staging environment. Notifications are being delivered in real-time, and user feedback indicates the system is user-friendly.

Progress Tracker

Effort

Original Estimate

20.0

Completed Work

18.0

Task - Code Review - Review Payment Gateway Integration

Review the code for the newly integrated payment gateway. Ensure all security best practices are followed. Provide feedback and request changes if necessary. Approve the code for merging once all issues are addressed. Document the review process and findings.

Due Date

2024-07-29T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:55:58.327Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	Code review completed, and the payment gateway integration has been approved and merged.All identified issues were addressed, and security practices confirmed.

Progress Tracker

Effort

Original Estimate

35.0

Completed Work

35.0

Task - Login Timeout Prevention

Investigate the root cause of the login timeout issue. Implement a solution to extend the session timeout duration. Test the fix in the development environment. Update any related documentation. Deploy the fix to the production environment.

Due Date

2024-07-04T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:53:39.127Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The login timeout control has been implemented by increasing the session timeout duration and optimizing session handling.

Progress Tracker

Effort

Original Estimate

15.0

Completed Work

15.0

Task - Feature Development - Add User Profile Page

Design the layout and functionality of the user profile page. Develop the frontend components for the profile page. Integrate the profile page with the backend API. Write unit and integration tests. Deploy the feature to the staging environment and test.

Due Date

2024-07-06T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-30T17:50:33.083Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The user profile page has been successfully implemented and tested. The feature is now live in the staging environment. Profile page meets all design and functionality requirements.

Progress Tracker

Effort

Original Estimate

10.0

Completed Work

10.0

Task - Search Autocomplete Functionality - Implementing real-time search suggestions

Develop the search autocomplete feature to provide real-time suggestions as users type. Ensure the autocomplete suggestions are relevant and based on popular search queries. Optimize the search algorithm to handle typos and suggest corrections.

Due Date

2024-11-14T00:00:00Z

Priority

2

Resolution Summary

Detail	Information
Resolution Date	2024-06-29T11:02:55.613Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The search autocomplete functionality was developed, tested and approved by the user.

Progress Tracker

Effort

Original Estimate

12.0

Completed Work

Task - Workout Log Feature - Enabling users to log workout details

Develop the workout log interface with input fields for type, duration, and intensity. Implement the ability to select from predefined workout types or enter custom workouts. Save workout data to the user's profile for history tracking.

Due Date

2024-08-22T00:00:00Z

Priority

3

Resolution Summary

Detail	Information
Resolution Date	2024-06-29T11:01:51.92Z
Resolved By	Susana Michelle Hernandez Chinchilla
Resolution	The enhancement has been added in the new release.

Progress Tracker

Effort

Original Estimate

15.0

Completed Work

Bug Fixes

Bug - Search Results Page - Incorrect Sorting Order

The search on the sublist in the Payments report is not sorting the results correctly.

Steps to Reproduce

Navigate to the search page. Perform a search with any keyword. Change the sorting order to "Price: Low to High."

Expected Result

Search results should be sorted with the lowest price items first.

Actual Result

Search results are not sorted correctly, with higher-priced items appearing before lower-priced ones.

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 1: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:18:01.18Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	An error in the sorting algorithm caused incorrect ordering of search results. The issue was resolved by correcting the sorting algorithm used for displaying search results.The issue has been resolved and confirmed through testing of various sorting options.

Progress Tracker

Effort

Original Estimate

Completed Work

Bug - Navigation Bar - Dropdown Menu Not Working

The dropdown menu is not working.

Steps to Reproduce

Navigate to the homepage. Hover over the navigation bar to reveal dropdown menus. Click on any dropdown menu item.

Expected Result

The dropdown menu should expand, and clicking on an item should navigate to the corresponding page.

Actual Result

The dropdown menu does not expand, and clicking on an item does nothing.

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 2: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:15:24.143Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	
Resolved Reason	Fixed
Resolution	The dropdown menu does not expand, and clicking on an item does nothing. The issue was resolved by fixing the JavaScript event handler for the dropdown menu. The issue has been resolved and confirmed through testing of the navigation bar.

Progress Tracker

Effort

Original Estimate

Completed Work

Bug - Checkout Page - Payment Gateway Timeout

There is a timeout error showing up on the checkout page.

Steps to Reproduce

Add items to the cart. Proceed to the checkout page. Enter valid payment details. Click the "Pay Now" button.

Expected Result

Payment should be processed, and the user should receive a confirmation message.

Actual Result

The payment gateway times out, and the user receives an error message "Payment processing failed."

Severity/Priority

1 - Critical / 1

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 3: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:13:07.793Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	The issue was resolved by optimizing the API call to the payment gateway and increasing the timeout threshold.The issue has been resolved and confirmed through successful payment processing tests.

Progress Tracker

Effort

Original Estimate

Completed Work

Bug - Profile Page - Profile Image Not Uploading

The profile image is not uploading.

Steps to Reproduce

Navigate to the user profile page. Click on "Edit Profile." Attempt to upload a new profile image. Click "Save."

Expected Result

The profile image should be successfully uploaded and displayed on the profile page.

Actual Result

The profile image fails to upload, and the user receives an error message "Image upload failed."

Severity/Priority

2 - High / 2

Resolution Summary

Root Cause

Communication Error

Resolution Details

Table 4: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:06:42.02Z

Detail	Information
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	Fixed
Resolution	A misconfiguration in the server-side file upload handler prevented the image from being saved correctly.The issue has been resolved and confirmed through testing.

Progress Tracker

Effort

Original Estimate

Completed Work

Bug - Login Page - User Unable to Login

User unable to login

Steps to Reproduce

Navigate to the login page. Enter valid credentials. Click the "Login" button.

Expected Result

User should be redirected to the dashboard upon entering valid credentials.

Actual Result

User remains on the login page and an error message "Invalid credentials" is displayed despite entering correct login details.

Severity/Priority

1 - Critical / 1

Resolution Summary

Root Cause

Specification Error

Resolution Details

Table 5: Resolution Information

Detail	Information
Resolution Date/Time	2024-06-30T18:04:35.613Z
Resolution By	Susana Michelle Hernandez Chinchilla
Reviewed By	Susana Michelle Hernandez Chinchilla

Detail	Information
Resolved Reason	Fixed
Resolution	The issue was resolved by correcting the logic in the authentication service that was incorrectly validating user credentials.

Progress Tracker

Effort

Original Estimate

Completed Work

Bug - Home Page - Images not Loading

The images from the homepage are not visible

Steps to Reproduce

Enter www.companyabc.com Scroll down to the gallery view Check the images on the photo grid

Expected Result

The images on the homepage of the website www.companyabc.com should be all visible

Actual Result

None of the images are visible in the homepage

Severity/Priority

3 - Medium / 2

Resolution Summary

Root Cause

Coding Error

Resolution Details

Table 6: Resolution Information

Detail	Information
Resolution Date/Time	
Resolution By	
Reviewed By	Susana Michelle Hernandez Chinchilla
Resolved Reason	
Resolution	The tag was not closed. The tags were updated and closed.

Progress Tracker

Effort

Original Estimate

Completed Work