UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Frontend para el desarrollo de proyectos de documentación colaborativa para productos de Digi

Autor/es

ÓSCAR LÁZARO MORENO

Director/es

María Vico Pascual Martínez Losa

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2019-20

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Frontend para el desarrollo de proyectos de documentación colaborativa para productos de Digi.

Realizado por:

Óscar Lázaro Moreno

Tutelado por:

María Vico Pascual Martínez-Losa

Logroño, Junio, 2020

Antes de comenzar quiero dar las gracias a todas las personas que me han apoyado durante la realización de este proyecto.

A mis tutores de Digi, Héctor González, Rubén Moral y Diego Escalona, y en especial a mi tutora de la Universidad, María Vico Pascual.

RESUMEN

La base de todo proyecto se fundamenta y se apoya sobre la documentación generada. De ahí, que gran parte del tiempo que se invierte en la elaboración de un proyecto, resida en la fase de documentación. Es por esto, que para ser más eficaz e invertir menos tiempo, Digi International necesita mejorar su framework, que facilite el trabajo de sus empleados.

El objetivo de este proyecto es desarrollar un plugin para un editor de texto, que facilite la creación y el desarrollo colaborativo de proyectos de documentación para los productos de Digi, realizando un frontend, que abstraiga a los usuarios de la complejidad del backend.

Palabras clave: documentación, facilitar, plugin, frontend.

ABSTRACT

The basis of every project is fundamental and is based on the documentation generated. Hence, a large part of the time spent on developing a project lies in the documentation phase. This is why, in order to be more effective and spend less time, Digi International needs to improve its framework which helps the work of its employees.

The objective of this project is to develop a plugin for a text editor, which helps the creation and collaborative development of documentation projects for Digi products, implementing a frontend which abstracts users from the complexity of the backend.

Keywords: documentation, helps, plugin, frontend

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Contexto

SOBRE DIGI

Digi International es una empresa **multinacional** norteamericana, con su sede principal en Hopkins, Minnesota. Es un proveedor global líder de productos y servicios de conectividad máquina a máquina (M2M) e Internet de las cosas (IoT) de misión crítica, en todo tipo de industrias, desde el transporte (flota, ferrocarril) hasta las ciudades inteligentes (agua y agua residuales, gestión de tráfico), pasando por la energía (monitorización, gestión de redes).

Una de sus principales características es que trabaja con clientes de distintos sectores, los cuales no tienen conocimientos de todas las tecnologías, esto implica tener que realizar una buena documentación de los productos, donde se detallen las características y su funcionamiento. Para ello, Digi dispone de un departamento de documentación, compuesto por especialistas que se encargan tanto de escribir documentación, como de supervisarla y corregirla antes de publicarla.

Antes, para realizar la documentación, se utilizaba un modelo técnico que precisaba de continua comunicación entre los desarrollares que escribían documentación y el departamento de especialistas en documentación, que la publicaban. Esto se debía a que la documentación tenía que ser escrita en una herramienta a la que los desarrolladores no tenían acceso. Cualquier cambio tenía que ser ejecutado por un miembro del departamento de documentación.

El proceso actual es muy ineficiente, y por este motivo, es necesario migrar el actual modelo de documentación a uno colaborativo usando AsciiDoc[1], en el que los propios desarrolladores son partícipes y actúan como editores. Con este cambio, se reduce la dependencia de un especialista del departamento de documentación. Para ello, la empresa ha desarrollado un backend, que integra este nuevo modelo.

Se entiende como proyecto de documentación, al conjunto de archivos que se obtienen al trabajar con el backend. Estos archivos son con los que luego se crea la documentación de los productos de la empresa.

En la siguiente figura, (Figura 1) se muestra el funcionamiento del backend, que será explicado a continuación.

[1] Lenguaje de marcas, usado en los editores de texto, muy útil para documentar.

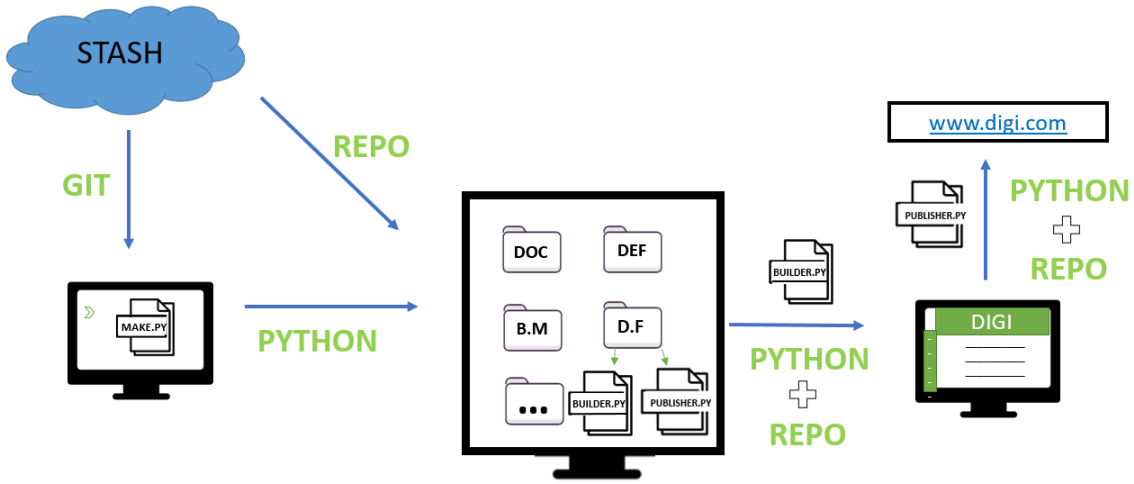


Figura 1. Estructura del backend.

Inicialmente desde Stash[2], a través de GIT, se clona en nuestro dispositivo local, un directorio que contiene un Script "make-project.py", el cual se usará más adelante.

En el siguiente paso, se ejecutará el Script, mencionado anteriormente, a través de Python, y obteniendo de Stash un repositorio llamado "Doc-framework", a través de Repo[3], se consigue crear en nuestro dispositivo local, un conjunto de carpetas estructuradas, lo que se entiende como proyecto de documentación.

Se usa repo, porque nos permite trabajar con todos los repositorios de Git a la vez, y además se ocupa de realizar las cargas al sistema de control de versiones. El uso de esta herramienta, obliga a combinar varios comandos, formando así, comandos complejos.

Estas carpetas representan los diferentes módulos que componen un proyecto de documentación. Dentro se encuentran los ficheros de texto con la propia documentación y los ficheros que configuran distintos parámetros del proyecto. Los ficheros con el texto de la documentación tienen que tener el formato (.AsciiDoc).

Cada módulo es independiente del otro, teniendo así cada uno una función distinta, lo que nos permite modificar por separado cada uno. Dichos módulos, pueden ser reutilizados en diferentes proyectos de documentación, con lo que se consigue escribir la documentación una sola vez y mostrarla en muchos proyectos sin duplicar las fuentes.

Cada proyecto de documentación contiene dos scripts de Python necesarios para proseguir con el proceso. A continuación se explica su función:

- "Builder.py": fichero encargado de construir el proyecto de documentación.
- "Publisher.py": fichero encargado para publicar el proyecto de documentación en la Web.

[2] Servidor de Bitbucket donde se almacenan todos los repositorios Git de los proyectos de la empresa.
[3] Herramienta de administración de repositorios creada por Google que se ejecuta en Git.

Tras haber creado todos los ficheros de documentación, y configurado los parámetros, se puede compilar y crear la documentación, ejecutando a través de Python y Repo el Script "Builder.py".

Dicho Script, nos ofrece distintos tipos de compilación, que cubren las diferentes necesidades de la empresa. Entre los principales tipos se encuentra:

- Local: Construir solo el espacio de trabajo local.
- Full: Construir todos los módulos con todas las versiones y todas las plataformas disponibles.
- Tag: Construir una versión etiquetada de la empresa.

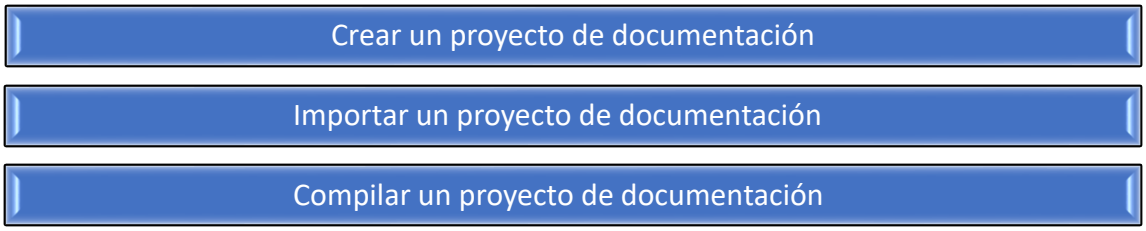
Una vez creada la documentación, se puede revisar, por si hay errores de gramática. Cuando ya está lista para publicarla, se ejecuta el Script "Publisher.py", a través de Python y Repo, y se publica en la página web de la empresa (www.digi.com).

Todo este proceso es muy costoso a la vez que complejo, por esto surge la necesidad de realizar este proyecto, que facilita el trabajo de documentación a los desarrolladores.

1.2. Objetivos

El objetivo principal del proyecto es facilitar la creación y el desarrollo colaborativo de proyectos de documentación para los productos de Digi, implantando un frontend que abstraiga a los usuarios de la complejidad que actualmente presenta el backend.

Así, se plantea como objetivos del TFG, el diseño e implementación de un frontend que permita:



Para que desde dicho frontend se puedan crear, importar y compilar proyectos de documentación, se plantea:

- Desarrollar un panel para crear proyectos de documentación.
- Desarrollar un panel para importar proyectos de documentación
- Crear una página de preferencias para personalizar un proyecto de documentación
- Añadir una acción para compilar un proyecto de documentación.
- Añadir una consola.
- Mostrar los resultados por pantalla.

1.3. Metodología

Para el desarrollo de este frontend, se utiliza una metodología ágil basada en **Scrum**. Dado que es un trabajo donde solo hay un desarrollador, solo se van seguir los principales principios de Scrum.

- Los roles usados son los siguientes:
  - **Product owner**: Pedro Pérez, responsable de Digi International Spain.

- **Scrum Master**: Diego Escalona, Héctor González y Rubén Moral, mis tutores de la empresa. (En cada Sprint, asumirá el rol una persona distinta, cambiando de forma rotativa.)
- **Desarrolladores**: Óscar Lázaro.

- Al igual que los demás equipos de la empresa, cada Sprint va a tener una duración de 2 semanas, comenzando y acabando los lunes. El tiempo en horas del que dispone cada Sprint es de 35h.
- Al finalizar un Sprint, y antes de comenzar el siguiente se realizará una reunión con los tutores de la empresa, en donde se validarán las tareas realizadas y se asignarán las nuevas tareas a realizar.
- No todos los sprints cuentan con fase de fase de Análisis, Diseño, Implementación y Pruebas, ya que no es necesario realizarlo en todas las tareas.

1.4. Tecnologías

ANTECEDENTES

A mediados de Noviembre, durante el periodo de prácticas, fui asignado a este proyecto, y desde entonces comencé a trabajar en él. Lo primero que tenía que hacer era decidir el editor de texto sobre el que trabajar, al efecto, realicé una investigación donde escogí 4 editores, que son Atom, AsciiDocFx, AsciiDoctor y Visual Studio Code.

Para ello, comparé sus distintas características y ventajas, que las he resumido en una tabla. (Tabla 1)

	Ascidocfx	**Asciidoctor**	**Atom**	**Visual Studio Code**
Permite añadir funcionalidad	SI	SI	SI	SI
Fácil instalación	SI	NO	SI	SI
Multiplataforma	SI	SI	SI	SI
Interfaz gráfica	SI	NO	SI	SI
Necesidad de instalar paquetes	NO	NO	SI	SI
Requisitos de instalación	NO	SI	NO	NO
Integración con Git	NO	NO	SI	SI

Tabla 1. Comparación de los Editores de Texto.

Tras la comparación de la (Tabla 1), me decanté por Atom y Visual Studio Code, que son los editores que más ventajas ofrecen de cara a este proyecto.

A partir de aquí, hasta el final del periodo de prácticas, aproximadamente durante unas 3 semanas, me dediqué a realizar distintas pruebas, en cuanto a su entorno gráfico, como a su funcionamiento.

Las pruebas sobre el entorno gráfico del editor, estaban orientadas a ver las posibilidades que ofrecen estos editores, a la hora de añadir funcionalidad a su interfaz gráfica, pero teniendo en cuenta aspectos importantes característicos de este proyecto, como:

- Tiene que seguir la misma estructura y forma de la interfaz gráfica del editor.
- El uso de esta interfaz, va a ser utilizada por desarrolladores de la empresa.
- Tiene que mostrar información sobre el estado del proceso en todo momento.
- La interfaz tiene que ser adaptada solo para una pantalla de ordenador, no para otro tipo de dispositivo con distintas dimensiones.

Al realizar estas pruebas, muchas decisiones sobre cómo diseñar la interfaz gráfica ya se han tomado antes de empezar a trabajar en el proyecto. Entre las principales decisiones se encuentran.

- Uso de paneles de cuadros de diálogo en vez de wizard. De esta forma se permite al usuario ver todas las opciones a la vez en la pantalla.
- Colocar la consola en la parte inferior de la pantalla. De esta manera ofrecemos al usuario la posibilidad de seguir trabajando con el editor a la vez que ve los resultados de la consola.
- Mostrar los errores y resultados, a través de un sistema de notificaciones propio del editor, para conseguir llamar la atención del usuario.

Sin embargo, la decisión sobre qué editor de texto escoger, no ha dado tiempo a tomarla durante el periodo de prácticas, teniendo así, que invertir las primeras horas del proyecto en comparar dichos editores y decantarme por uno.

COMPARATIVA DE TECNOLOGIAS

Tras hacer una investigación más profunda de estos dos editores, se ha creado la siguiente tabla (Tabla 2) para compararlos.

	Atom	**Visual Studio Code**
Simplicidad	SI	NO
Lenguajes	JavaScript y JSON	JavaScript y JSON
Documentación	SI	SI
Sistema de notificaciones propio	SI	SI
Foros de soporte	SI	NO
Limitación en el entorno gráfico	NO	SI
Paquetes	SI	SI

Tabla 2. Comparación de Atom con Visual Studio Code.

En esta tabla, se aprecia que Atom ofrece más posibilidades que Visual Studio Code, a la hora de trabajar con él. Este el motivo por el que me he decantado por Atom, y sobre el que se va a realizar el frontend.

Señalar que cuando se hace referencia a un plugin, en Atom significa lo mismo que un paquete o un complemento. A lo largo de esta memoria, nos referiremos a lo mismo usando cualquier término.

TECNOLOGIAS USADAS

En este apartado, se habla sobre las tecnologías y herramientas que se van a usar en el proyecto, las cuales pueden ser divididas en dos partes. Por un lado, las tecnologías que se van a usar para desarrollar el frontend. Y, por otro lado, las tecnologías que se usan para el desarrollo del backend, con las cuales no trabajo directamente, pero requieren conocerlas para poder realizar este proyecto.

Tecnologías para realizar el frontend:

- **Atom**: Editor de texto de código abierto, multiplataforma, construido por tecnologías web, sobre el que se realiza el plugin. Soporta distintos lenguajes de marcas, entre los cuales se encuentra AsciiDoc.
- **Jira**: Herramienta orientada a la administración de tareas de un proyecto. Es la que usa esta empresa, por lo que, también se usa en este proyecto.
- **Balsamiq**: Herramienta orientada para diseñar las maquetas de las interfaces. Es la que usa la empresa, por lo que, también se usa en este proyecto.
- **JavaScript**: Lenguaje de programación que, en este caso, sigue el paradigma de la orientación a objetos, usado normalmente en el desarrollo web. Es usado para construir todos los paquetes de Atom.
- **Git**: Software de control de versiones que se utiliza en este proyecto. Como apoyo se va a usar Git Extensions, que permite trabajar con interfaz gráfica. El repositorio remoto donde se almacena el código se encuentra en un servidor privado de la empresa, *Stash*.

Tecnologías usadas en el backend:

- **Python**: Lenguaje de programación de código abierto, orientado a objetos, muy usado en trabajos relacionados con la Inteligencia Artificial, Frameworks de pruebas y desarrollo web. Con este lenguaje se han desarrollado los distintos Scripts que gestionan el backend.
- **Repo**: Herramienta construida sobre Git, que ayuda a administrar sus repositorios. Realiza las cargas a los sistemas de control de revisiones y automatiza partes del flujo de trabajo de desarrollo. Se usa en el backend a la hora de crear la estructura del proyecto.

1.5. Planificación

En este apartado se va a precisar la planificación que se ha realizado antes del comienzo del proyecto. Consta de varios apartados como EDT, Diagrama de Gantt, Diagrama de hitos, y descomposición de tareas.

1.5.1. EDT

La estructura de descomposición de tareas (Figura 2), muestra una descomposición jerárquica del proyecto, orientada a los entregables y al trabajo a realizar. Se busca lograr los objetivos del proyecto a través de los entregables.

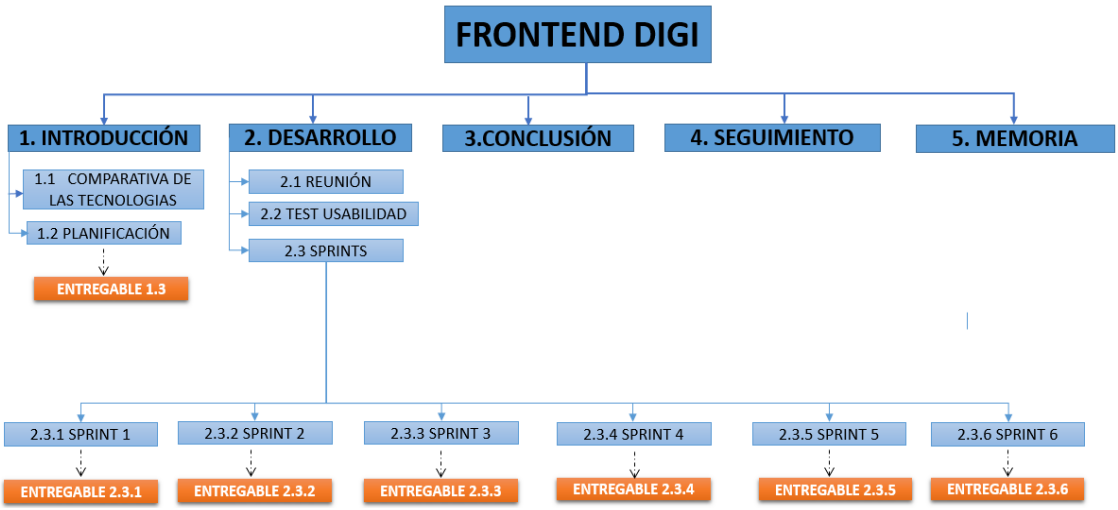


Figura 2. Estructura de descomposición de tareas del proyecto.

A continuación, se van a describir las tareas:

- **Comparativa de las tecnologías**: Comparación de las ventajas e inconvenientes de los dos editores de texto (Atom y Visual Studio Code).
- **Planificación**: Descomposición de las tareas a realizar durante el proyecto, asignándole el tiempo de estimación que llevará desarrollarla.
- **Reuniones**: Reuniones que se realizan al comenzar y finalizar el sprint donde se realiza un seguimiento del proyecto.
- **Sprints**: Etapa del proyecto donde se encuentran los 6 Sprints. A lo largo de ellos, se desarrollan la fase de Análisis, Diseño, Implementación y pruebas.
- **Conclusión**: Se muestran las lecciones aprendidas, al realizar este proyecto.
- **Seguimiento**: Control periódico de las actividades realizadas.
- **Memoria**: Redacción de este documento.

1.5.2. Diagrama de Gantt

El diagrama de Gantt, (Tabla 4), muestra una representación visual del avance del proyecto a lo largo de estos 5 meses. Nos ayuda a realizar un seguimiento de las tareas en el proyecto, así como de sus dependencias.

	Febrero				Marzo					Abril				Mayo				Junio		
	3	10	17	24	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15
Introducción																				
Comparativa de las Tecnologías																				
Planificación																				
Desarrollo																				
Sprint 1																				
Sprint 2																				
Sprint 3																				
Sprint 4																				
Sprint 5																				
Sprint 6																				
Reunión																				
Conclusión																				
Seguimiento																				
Menoria																				

Tabla 3. Diagrama de Gantt.

Señalar que el Sprint 4, aunque tenga una extensión de 3 semanas, tiene la misma duración que los demás Sprints, porque cuenta con varios días que son festivos (7 abril -15 abril) debido a la Semana Santa.

1.5.3. Diagrama de Hitos

Un hito es un evento importante dentro del proyecto. En este caso los eventos se corresponden con las entregas parciales al cliente. Hay 8, (tabla 4), correspondiendo 1 con la etapa de introducción y los siguientes 6 con el final de cada Sprint, y por último, la entrega final de toda la memoria.

La elaboración de este diagrama es necesaria para el correspondiente control y seguimiento del proyecto.

	Febrero				Marzo					Abril				Mayo				Junio		
	3	10	17	24	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15
Entregable 1.6				◆																
Sprint 1						◆														
Sprint 2								◆												
Sprint 3										◆										
Sprint 4													◆							
Sprint 5															◆					
Sprint 6																	◆			
Final																				◆

Tabla 4. Diagrama de hitos.

Las distintas entregas se realizarán los lunes de cada semana, salvo problema, en cuyo caso habrá de margen hasta el viernes.

1.5.4. Estimación de tiempos y descomposición de tareas

En este apartado se va a hablar de la descomposición de tareas realizada antes de la fase de desarrollo, con su respectiva estimación en horas.

En la siguiente tabla (Tabla 5) se muestran las tareas principales en que se divide el proyecto, con sus respectivas estimaciones.

TAREA	TIEMPO
Introducción	**20h**
Planificación	18h
Comparativa de las Tecnologías	2h
Desarrollo	**235h**
Sprints	215h
Reuniones	20h
Conclusión	**5h**
Seguimiento	**5h**
Memoria	**35h**
TOTAL	**300h**

Tabla 5. Estimación de tiempo de las tareas.

Destacar que la tarea llamada Sprints, con una duración de 215h, equivale a la suma de los 6 Sprints, más la tarea de pruebas. Es ahí donde se aborda el desarrollo de la herramienta en sí.

Para dividir el trabajo sobre los distintos Sprints se ha realizado un primer desglose de las tareas (Tabla 6).

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-1	Diseño de prototipo de la interfaz	15h
DOCFF-2	Creación de una consola, donde se ve el log de la ejecución de cada Script.	15h
DOCFF-3	Crear un proyecto de documentación	30h
DOCFF-4	Importar/clonar un proyecto de documentación existente	26h
DOCFF-5	Compilar un proyecto de documentación	35h
DOCFF-6	Publicar la documentación en la web	10h
DOCFF-7	Configurar un proyecto de documentación.	15h
DOCFF-8	Mostrar los posibles errores en la pantalla	10h
DOCFF-9	Añadir una nueva página a la documentación	15h
DOCFF-10	Eliminar una página de la documentación	10h
DOCFF-11	Renombrar una página de la documentación	12h
DOCFF-12	Añadir una nueva unidad de documentación	18h
DOCFF-13	Eliminar una nueva unidad de documentación	12h
DOCFF-14	Renombrar una nueva unidad de documentación	14h
TOTAL		**237h**

Tabla 6. Product backlog inicial.

El tiempo total estimado (237h), supera las 210h planificadas para desarrollar este proyecto, esto se debe a que se han añadido algunas tareas consideradas con poca relevancia, que se implementarán dependiendo del tiempo que se disponga al final de los últimos Sprints.

2. DESARROLLO

Antes de comenzar la fase de desarrollo, se realizó una reunión de planificación con los tutores. En ella se decidió eliminar, modificar y crear nuevas tareas, que junto a las ya creadas en la planificación componen el Product Backlog de la aplicación. También se estimaron de nuevo y se les asignaron distinta prioridad dependiendo de la importancia y necesidad de desarrollarlas.

La selección de las tareas en los distintos Sprints se ha hecho según su prioridad. Debido al alcance de algunas tareas y a su duración que es superior a la extensión del Sprint, al comienzo de cada Sprint, se ha decido desgranarlas en distintas subtareas.

Se adjunta la tabla con las tareas del product backlog final (Tabla 7).

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-0	Diagrama de casos de uso	1h
DOCFF-1.1	Esqueleto del proyecto	5h
DOCFF-1.2	Research: exportar plugin	2h
DOCFF-2	Consola de log	18h
DOCFF-3	Crear un proyecto de documentación	31h
DOCFF-4	Clonar/importar un proyecto de documentación	24h
DOCFF-5	Compilar un proyecto de documentación de forma local	11h
DOCFF-6	Lanzar el proyecto de documentación	8h
DOCFF-7	Configurar un proyecto de documentación	12h
DOCFF-8	Detectar los principales errores	4h
DOCFF-9	Añadir una nueva página a la documentación	14h
DOCFF-10	Eliminar una página de la documentación	10h
DOCFF-11	Renombrar una página de la documentación	12h
DOCFF-12	Configurar preferencias del plugin	12h
DOCFF-13	Clonar repositorio de Stash	9h
DOCFF-14	Notificaciones	6h
DOCFF-15	Barra de progreso	3h
DOCFF-16	Comprobar si existe un proyecto de documentación	3h
DOCFF-17	Añadir una comprobación de la compilación en el script de construcción	11h
DOCFF-18	Sincronizar el repositorio del Script	12h
DOCFF-19	Patrón Singleton	4h
DOCFF-20	Añadir a Atom la estructura del proyecto de documentación	4h
DOCFF-21	Abrir un cuadro de diálogo	2h
DOCFF-22	Tabbing[4]	3h
DOCFF-23	Mejorar el método de selección de proyectos	10h
DOCFF-24	Agregar entradas de menú contextual por proyecto en la vista de árbol del proyecto	6h
DOCFF-25	Sistema e integración de pruebas	5h
TOTAL		**242h**

Tabla 7. Product backlog.

[4] (Orden de tabulación), orden en el que los componentes de un wizard adquieren el foco cuando el usuario presiona la tecla Tab.

Nota: La unidad mínima de estimación es 1h. Por eso las tareas o subtareas estimadas con menos de 1 hora de desarrollo, estarán reflejadas con 1 hora.

Aunque el Análisis, Diseño e Implementación se recogen como subtareas y las pruebas no (excepto la tarea DOCFF-25), estas se realizarán a lo largo de cada Sprint. Las dividimos en dos partes:

- Pruebas funcionales: Se realizan durante la fase de desarrollo y su objetivo es probar las diferentes características de la aplicación por separado, en un entorno en que otros desarrollos de la herramienta no interfieran.
- Pruebas sistemáticas: Consiste en probar la integración de todas las partes desarrolladas para identificar problemas de funcionamiento conjunto.

Aunque se deberían realizar los dos tipos de pruebas, después de la implementación de cada tarea solo se han realizado las pruebas funcionales, debido al tiempo limitado del que disponemos. Las pruebas sistemáticas solo se desarrollarán en el último Sprint (tarea DOCFF-25), comprobando así el funcionamiento completo del frontend.

Para mostrar los resultados de estas pruebas, se hará uso de este diseño de plantilla.

ID	
SPRINT	
TAREA A LA QUE PERTENECE	
DESCRIPCION	
RESULTADO	
COMENTARIOS	

Aclarar que los diferentes fragmentos de código que se muestren a continuación, no corresponden con la versión final del código de la aplicación, ya que son ejemplos simplificados que muestran las funciones.

2.1. Sprint 1

Las tareas que se incluyen en este sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-0	Diagrama de casos de uso	1h
DOCFF-1.1	Esqueleto del proyecto	5h
DOCFF-1.2	Research: exportar plugin	2h
DOCFF-12	Configurar preferencias del plugin	12h
DOCFF-13	Clonar repositorio de Stash	9h
DOCFF-2	Consola de log	6h
TOTAL		**35h**

Tabla 8. Product backlog del Sprint 1.

Nota: Debido a su duración y al orden en el que deben llevarse a cabo, algunas tareas no podrán ser completadas en un único Sprint. En él, se les asignarán las horas que se les va a dedicar en ese Sprint, completándose la tarea en el siguiente.

Un ejemplo de ello es la tarea DOCFF-2, con una estimación de 18 horas (Tabla 7), pero que en este Sprint, solo se van a invertir 6 horas (Tabla 8). Las horas restantes se emplearán en el siguiente Sprint.

2.1.1. Tareas

- **Tarea DOCFF-0: Diagrama de casos de uso** (Completado)

  En esta tarea solo se desarrolla el diagrama de casos de uso. La especificación de cada caso de uso, se encuentra como subtarea, a lo largo de los Sprints.

  Solo se ha encontrado un rol, el de desarrollador. Se ha decidido llamarlo Usuario, porque representa a todos los desarrolladores y documentalístas de la empresa.

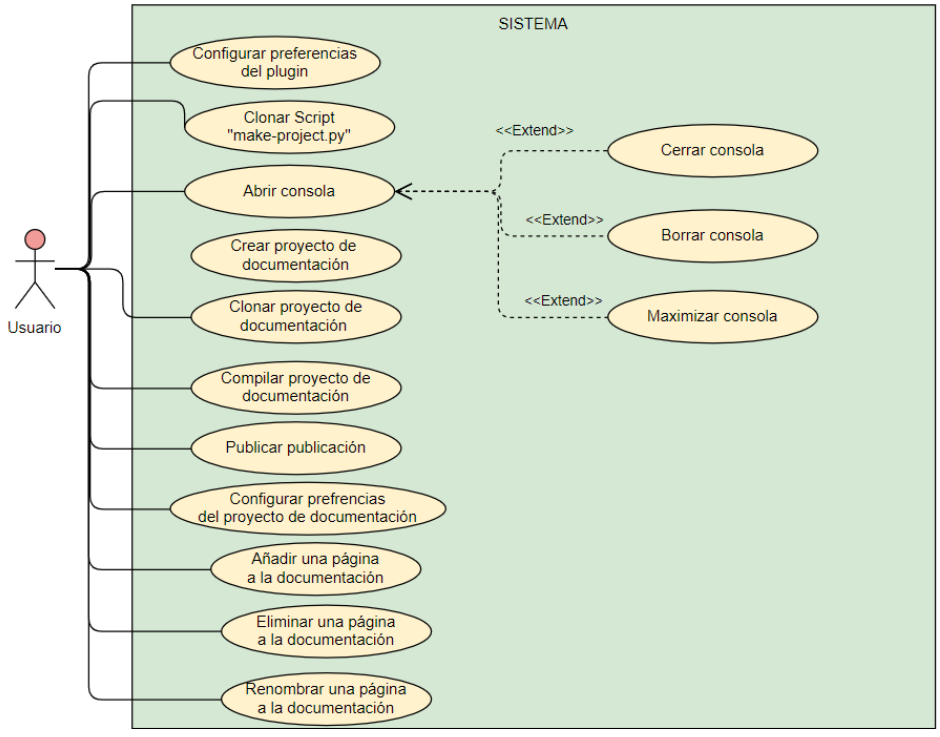


Figura 3. Diagrama de casos de uso.

- **Tarea DOCFF-0.1: Esqueleto del proyecto** (Completado)

  Es la primera tarea de la fase de implementación. Consiste en preparar el entorno de desarrollo y probarlo por primera vez. Esta tarea se divide en varias subtareas:

  - Crear un paquete en Atom, llamado DOCFF (*Documentation frontend framework*). Todos los paquetes tienen que seguir la siguiente estructura de carpetas.

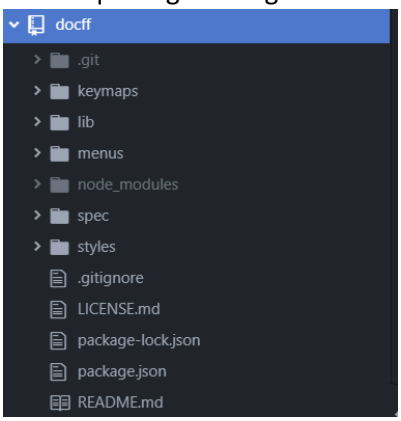


Figura 4. Estructura de un paquete en Atom.

  - Añadir funcionalidad al paquete, comprobando su funcionamiento.
  - Compilar y ejecutar el código, que consiste en recargar el editor de texto.
  - Crear un repositorio en Stash (servidor de la empresa), y realizar el primer commit.

- **Tarea DOCFF-0.2: Research** (Completado)

  Esta tarea consiste en buscar información sobre las distintas formas en que se puede exportar e importar un plugin de Atom. Se muestra un breve resumen.

  A la hora de exportar un paquete:

  - Creando un repositorio en Git, de esta forma se podría publicar el paquete en la página principal de Atom.
  - Comprimiendo la carpeta del plugin en un fichero `.zip´.

  Formas para importar un paquete:

  - Instalándolo, a través de Atom: File → Settings→ Install.
  - A través de la terminal de comandos: apm install name_package. (necesario instalar apm planner).
  - Descomprimiendo el paquete en la carpeta C:\Users\user_name\.atom\packages. (user_name corresponde con el nombre de la carpeta de usuario).
    Esta dirección local, es donde se almacenan los paquetes instalados de Atom.

- **Tarea DOCFF-12: Configurar preferencias del plugin** (Completado)

  Los requisitos que se han encontrado en esta tarea son los siguientes:

  Requisitos imprescindibles:

  - La aplicación permitirá seleccionar la ruta de instalación de las carpetas por defecto.
  - El usuario podrá seleccionar la ruta donde se ha instalado Git y Python.

  Esta tarea no solo es de implementación, sino que también hay que realizar una fase previa de análisis, por eso, se ha subdivido en tres subtareas:

  - Especificación del caso de uso: Configurar preferencias del plugin.
  - Comprobación de Git y Python.
  - Preferencias del plugin.

ANÁLISIS

Especificación del caso de Uso: Configurar preferencias del plugin

Caso de uso	**Configurar preferencias del plugin**
Descripción	El usuario configura las settings propias del plugin.
Actor	Usuario
Precondición	Las settings del producto sigan la guía de usabilidad y estilo de las settings de Atom.
Postcondición	Se habrán cambiado el valor de las variables de las settings del plugin.
Flujo básico	1. El usuario pulsa el botón *Settings* del menú. 2. El sistema muestra por pantalla un panel con el título y descripción de la settings, acompañado de un campo input de tipo texto. 3. El usuario rellena los distintos campos 4. Cuando el usuario deje de escribir en el campo, el sistema guarda el nuevo valor de la variable.
Flujo alternativo	
Anotaciones	

IMPLENTACIÓN

Preferencias del plugin

Se crean las preferencias del plugin, las cuales será necesario especificar para poder crear, clonar, compilar un proyecto de documentación.

Tras buscar varias posibilidades para acometer esta tarea, me he decantado por usar los métodos propios que ofrece el editor de texto para crear settings, aunque de esta forma se está limitado para diseñar la página, se cumplen todos los requisitos establecidos.

Comprobación de Git y Python

Se obtiene el path introducido por el usuario de las settings de las preferencias, y se comprueba que la dirección se corresponde con el ejecutable de Git o de Python en su caso. En un primer momento se comprobaba que existía la dirección obtenida y esta a su vez contenía un archivo ejecutable (formato .exe). Al ver que esta idea, era ineficiente y se podían producir falsos positivos, se decidió buscar nuevas soluciones.

La solución que se ha encontrado, ha sido ejecutar a través de una Shell (intérprete de comandos), la dirección obtenida junto con el comando '--version'. Para saber si el output[5] producido es el esperado, se ha comparado con una expresión regular. De esta forma se comprueba la instalación de Git y Python en el PC.

Para ejecutar el comando a través de una Shell, se ha creado el método *execute_command_simple,* que crea una promesa que llama la función *exec,* la cual ejecuta un comando a través de la terminal. Esta función se ha importado de la librería Node.js.

A continuación se adjunta una captura de las expresiones regulares que se han creado.

```
let patronGit = /[a-zA-Z]\s\d/;
let patronPython3 = /[a-zA-Z]\b\s3\.\d/;
let patronPython2 = /[a-zA-Z]\b\s2\.\d/;
```

[5] Resultado que se obtiene de ejecutar un comando por la terminal.

- **Tarea DOCFF-13: Clonar repositorio de Stash** (Completado)

  Los requisitos de esta tarea son los siguientes:

  Requisitos imprescindibles:

  - El usuario podrá descargarse el repositorio inicial de documentación.
  - El usuario será notificado cuando se haya descargado correctamente el proyecto inicial de documentación o cuando haya habido un error.

  Requisitos Secundarios:

  - Al usuario se le notificará si no tiene instalado Git.
  - Al usuario se le notificará si no tienen instalado Python.

  Esta tarea, también tiene fase de Análisis, y se he ha separado en dos subtareas:

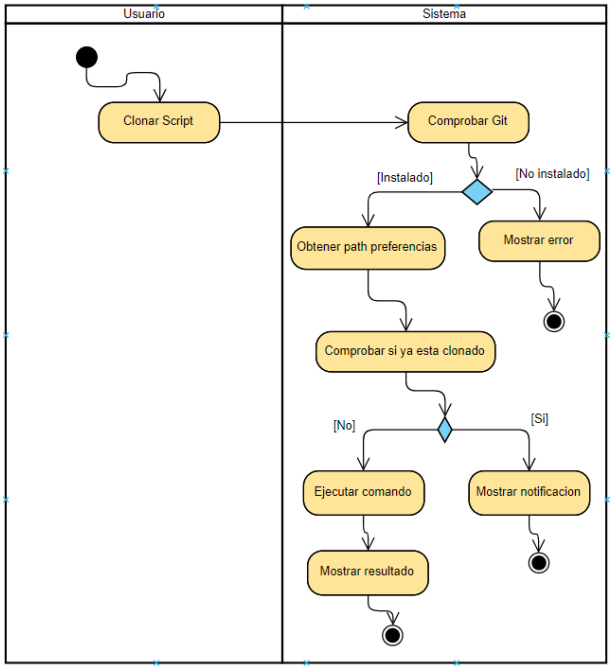
  - Especificación caso de uso: Clonar el Script 'make-project.py'.
  - Proceso de clonar un repositorio de Stash.

  ANÁLISIS

  Especificación del caso de uso: Clonar script "make –project.py"

  También se adjunta el diagrama de actividad para entender mejor el funcionamiento**.**

Caso de uso	**Clonar script "make-project.py"**
Descripción	El sistema descargará desde Stash (servidor de la empresa), el repositorio que contiene el Script "make-project.py", el cual se usará para crear o clonar un proyecto de documentación.
Actor	Usuario
Precondición	El usuario tiene que estar conectado a la red privada de la empresa. El usuario tiene que tener instalado Git en el PC.
Postcondición	En el path marcado en las preferencias del plugin, se encontrará una nueva carpeta con el Script.
Flujo básico	1. El usuario pulsa el botón *make-project.py* del menú. 2. El sistema comprueba que el equipo tiene instalado Git. 3. En caso de que Git esté instalado, el sistema obtiene la setting 'atom path' de las preferencias del plugin. 4. El sistema comprueba que en dicha ruta, no se encuentra el repositorio a clonar. 5. En el caso de que no exista, el sistema ejecuta en un proceso nuevo el comando para clonar el repositorio. 6. Se muestra por pantalla el resultado de dicha acción.
Flujo alternativo	- En el caso de que Git no esté instalado, se muestra un mensaje al usuario, avisando de que no tiene Git y no se ha instalado nada. - En el caso de que el repositorio ya haya sido clonado en la misma ruta, el sistema no hace nada.
Anotaciones	Se puede tener duplicado el repositorio en el mismo equipo, si están ubicados en diferentes direcciones.



IMPLEMENTACIÓN

Clonar repositorio de Stash

Esta tarea es imprescindible, ya que muchas de las siguientes tareas se basan en esta.

Consiste en clonar en nuestro dispositivo local, el repositorio para poder trabajar con un proyecto de documentación. Este repositorio contiene el Script 'make-project.py', mencionado anteriormente. Para ello se han usado algunos comandos de Git.

Durante el desarrollo, ha habido varios problemas de sincronización mientras se estaba ejecutando el proceso de clonación. Para solucionarlo se ha hecho uso de las funciones propias de JavaScript async(), y await(), que nos permiten trabajar con las promesas de una manera más cómoda. Este problema, ha afectado al tiempo en la estimación inicial de esta tarea.

- **Tarea DOCFF-2: Consola de log** (Incompleta)

Los requisitos que se cumplen para completar esta tarea son los siguientes:

Requisitos imprescindibles:

- El usuario podrá abrir una consola de log.
- El usuario podrá borrar los datos de la consola de log.
- El usuario podrá ver el comando que se está ejecutando.
- El usuario podrá cerrar la consola de log cuando quiera.

Requisitos Secundarios:

- La aplicación mostrará Tooltips[6], para ciertos botones.

Las subtareas en que se ha divido esta tarea son:

- Especificación de los casos de uso de la consola de log.
- Diseño del prototipo de la consola de log.
- Implementar consola de log.

[6] Pequeños recuadros de información que sirven de ayuda visual para el usuario, suelen aparecer cuando se pone el cursor encima de un botón.

- Mostrar Scrollbar.
- Borrar consola de log.

ANÁLISIS

Especificación de los casos de uso de la consola

Caso de uso	**Abrir consola**
Descripción	El sistema muestra por pantalla una consola de log.
Actor	Usuario
Precondición	
Postcondición	Se muestra la consola de log.
Flujo básico	1. El usuario pulsa el botón *new terminal.* 2. El sistema comprueba que no está abierta la consola. 3. En caso de no estar abierta, se abre la consola.
Flujo alternativo	
Anotaciones	

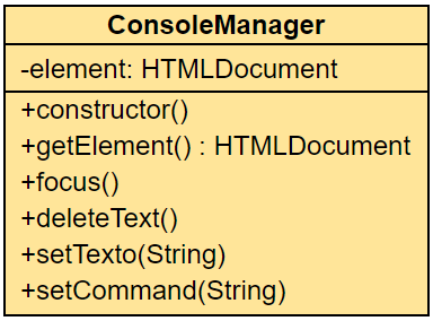
Caso de uso	**Borrar consola**
Descripción	El sistema borra los datos de la consola.
Actor	Usuario
Precondición	La consola tiene que estar abierta
Postcondición	La consola no tendrá nada de contenido para mostrar.
Flujo básico	1. El usuario pulsa el botón de borrar datos. 2. El sistema comprueba que hay datos mostrados por pantalla. 3. El sistema borra los datos de la consola.
Flujo alternativo	Si la consola no muestra ningún dato, el sistema no hace nada.
Anotaciones	

Caso de uso	**Cerrar consola**
Descripción	El sistema deja de mostrar la consola por pantalla.
Actor	Usuario
Precondición	La consola tiene que estar abierta.
Postcondición	La consola ha sido cerrada.
Flujo básico	1. El usuario pulsa el botón de cerrar consola. 2. El sistema minimiza la consola, y se deja de ver en la pantalla.
Flujo alternativo	
Anotaciones	En caso de que la consola este mostrando contenido y esta se cierre, los datos se guardan, y se seguirán viendo.

Caso de uso	**Maximizar consola**
Descripción	El sistema utiliza toda la pantalla para mostrar la consola de log.
Actor	Usuario
Precondición	La consola de log tiene que estar abierta.
Postcondición	La consola pasa a ocupar toda la pantalla.
Flujo básico	1. El usuario pulsa el botón de maximizar la consola. 2. El sistema aumenta el tamaño de la consola de log, ocupando así toda la pantalla.
Flujo alternativo	Si la consola de log, ya ocupa toda la pantalla, el sistema no hace nada.
Anotaciones	

DISEÑO

Para el desarrollo de esta tarea, se ha creado una nueva clase ***ConsoleManager.***
En ella se hace uso de la interfaz HTMLDocument. Esta interfaz, pertenece al API de JavaScript, y sirve para trabajar con los elementos HTML.



Diseño del prototipo de la consola de log

Se adjunta una foto del prototipo de la consola. Para su diseño, me he basado en la estructura y forma de otras consolas de log, que se han desarrollado en algunos paquetes de Atom.

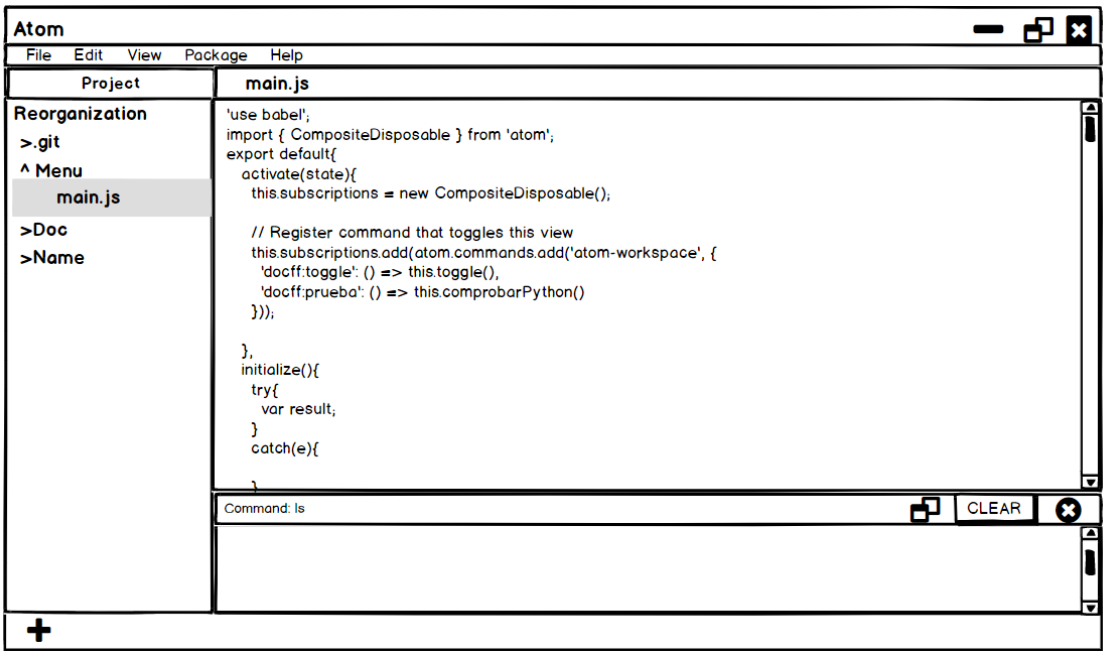


Figura 5. Diseño de la consola de log.

Como vemos en la imagen, se ha añadido un botón en la parte inferior de la izquierda, que pulsándolo abrirá la consola de log. En el momento que el cursor del ratón está por encima del botón, se mostrará un Tooltip, con el texto “new terminal”.

En cuanto a la consola, el comando se muestra en la parte superior de la izquierda, y los botones de izquierda a derecha tienen las siguientes funciones: Maximizar la consola, borrar los datos de la consola y cerrar la consola. El scrollbar, solo se mostrará en caso necesario.

Aclarar que la consola tiene el color negro, para seguir la estructura de la interfaz gráfica del editor.

2.1.2. Pruebas

ID	DOCFF 1.1
SPRINT	1
TAREA A LA QUE PERTENECE	Esqueleto del proyecto
DESCRIPCION	Comprobar que se ha añadido un botón a la interfaz gráfica, y que al hacer click, se muestra un panel vacío en medio de la pantalla.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 12
SPRINT	1
TAREA A LA QUE PERTENECE	Configurar preferencias del plugin
DESCRIPCION	Ir a las preferencias del plugin (botón *Settings*). Comprobar que las settings se pueden modificar y se guardan automáticamente por el sistema.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 13
SPRINT	1
TAREA A LA QUE PERTENECE	Clonar repositorio del proyecto inicial de documentación
DESCRIPCION	Pulsar el botón *make-project.py.* Comprobar que se añade en el path indicado una nueva carpeta llamada “doc-make-project”, con un fichero “make-project.py”.
RESULTADO	**COMPLETADO**
COMENTARIOS	

2.1.3. Revisión del Sprint

En este apartado se va a hacer una comparación de las horas estimadas vs las horas invertidas en la realización de las distintas tareas.

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 9)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-0	Diagrama de casos de uso	1 h	1 h	0%
DOCFF-1.1	Esqueleto del proyecto	5 h	4 h	**-20%**
DOCFF-1.2	Research: exportar plugin	2 h	2 h	0%
DOCFF-12	Configurar preferencias del plugin	12h	12h	0%
	Fase de Análisis	1h	1h	0%
	Comprobación de Git y Python	3h	5h	**+66,66%**
	Preferencias del plugin	8h	6h	**-25%**
DOCFF-13	Clonar repositorio de Stash	9h	11h	**+22,22%**
	Fase de Análisis	1h	1h	0%
	Proceso de clonar un repositorio de Stash	8h	10h	**+25%**
DOCFF-2	Consola de log	6h	5h	**INCOMPLETA**
	Fase de Análisis	2h	2h	0%
	Fase de Diseño	1h	1h	0%
	Implementación de la consola de log	3h	2h	**INCOMPLETA**
TOTAL		**35 h**	**35 h**	**+2.85%**

Tabla 9. Comparativa de los tiempos estimados y reales de las tareas del Sprint 1.

Al ver los resultados de la tabla, se ve que la estimación en horas con respecto a las tareas de la fase de análisis y diseño han sido acertadas, pero con respecto a la fase de implementación, se han producido desviaciones tanto a la alza como a la baja.

Como conclusión final de este Sprint, se puede decir que el resultado final ha sido positivo, porque para ser el primer Sprint, solo se ha producido una pequeña desviación de una hora.

2.2. Sprint 2

Las tareas que se han planificado para este Sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-2	Consola de log	13h
DOCFF-14	Notificaciones	6h
DOCFF-19	Patrón Singleton	4h
DOCFF-15	Barra de progreso	3h
DOCFF-3	Crear un proyecto de documentación	9h
TOTAL		**35h**

Tabla 10. Product backlog del Sprint 2.

La tarea DOCFF-2, solo consta de las subtareas pertenecientes a la fase de Implementación, ya que las otras subtareas se han desarrollado en el anterior Sprint. La estimación de esta tarea es de 13 horas porque ya se han invertido 5 horas en el anterior Sprint.

2.2.1. Tareas

- **Tarea DOCFF-2: Consola de log** (Completado)

  IMPLEMENTACIÓN

  Implementar consola de log

  Se ha desarrollado el código para construir la consola de log, diseñada anteriormente. Para ello se han utilizado algunos de los elementos básicos de HTML para representar datos.

  Durante la implementación, me he encontrado con un problema de diseño a la hora de añadir el botón *Clear*. En un principio, se pretendía añadir como icono representativo del botón una papelera, pero al ver que era demasiado pequeño, y al agrandarlo no quedaba gráficamente como se esperaba, se ha optado por usar este otro icono.

  A continuación se muestra una imagen del resultado de la consola de log.

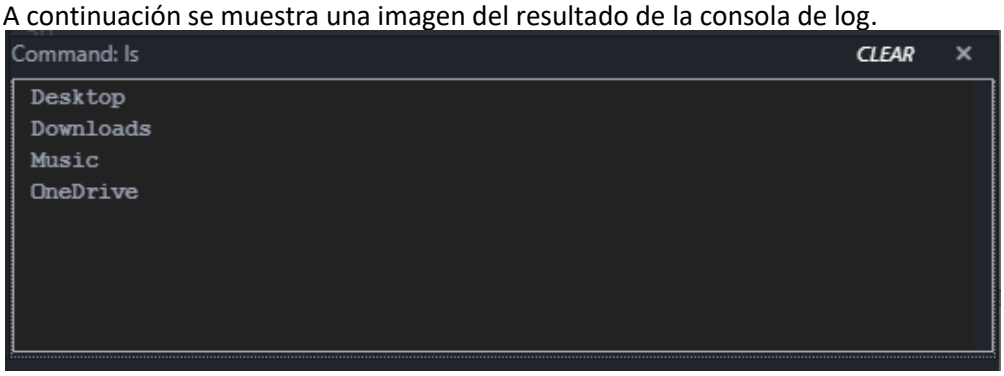


Figura 6. Consola de log.

  El botón para maximizar la consola no se ha implementado, ya que es una subtarea con poca prioridad, que todavía se encuentra en el backlog. Se realizará en los últimos Sprint, en caso de ir bien de tiempo.

  Mostrar Scrollbar

  Se desarrolla en la parte derecha de la consola de log una barra deslizable hacia arriba o hacia abajo. Con ella conseguimos acceder a la visualización de todos los datos de la consola. En caso de no ser necesaria, no aparecerá. Se ha implementado en la clase *ConsoleManager*, donde se crea la consola.

Borrar consola de log

Tarea muy sencilla, consiste en dejar la consola vacía de datos. Para ello se ha creado un evento para el botón *clear*.

- **Tarea DOCFF-14: Notificaciones** (Completado)

Los requisitos de esta tarea son los siguientes:

Requisitos imprescindibles:

- La aplicación mostrará las notificaciones de error de color rojo.
- La aplicación mostrará las notificaciones de éxito de color verde.
- La aplicación mostrará las notificaciones de información de color azul.

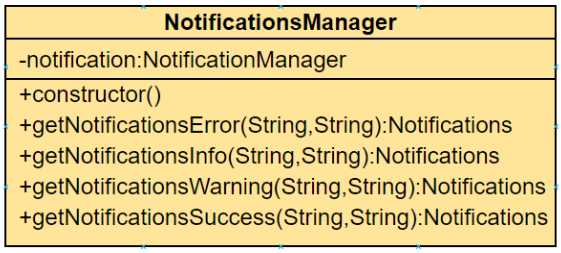
Requisitos Secundarios:

- El sistema podrá mostrar notificaciones de advertencia. En su caso serán de color amarillo.

DISEÑO

Para desarrollar esta tarea, se ha creado la clase ***NotificationsManager***. Para su desarrollo se hace uso de una clase propia de Atom, *NotificationManager*, la cual nos ofrece varios métodos para agregar notificaciones.

El objeto Notifications, que devuelven los métodos, corresponde con otra clase de Atom, la cual crea la notificación para el usuario y que contiene un mensaje y un tipo.

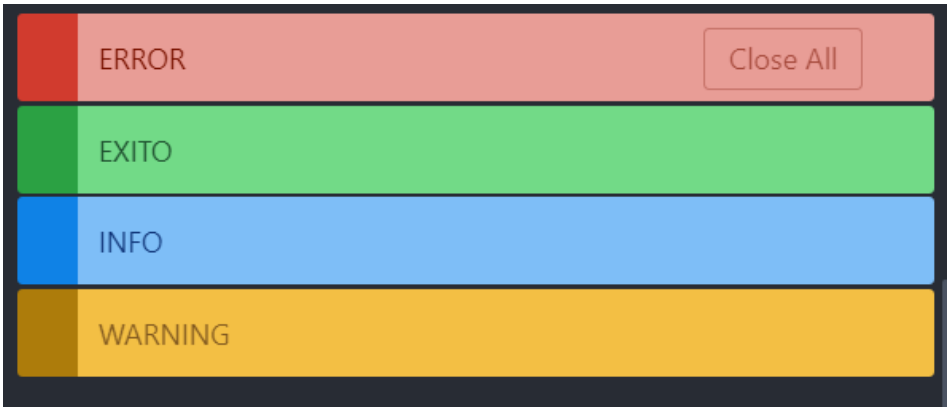


IMPLEMENTACIÓN

Notificación

En esta tarea se ha desarrollado un sistema de alerta para el usuario. Se muestran notificaciones de distintos colores, dependiendo de la función que tengan. De color rojo para mostrar un error, color azul para mostrar información, color amarillo para avisar de un Warning, y color verde para avisar que la acción se ha completado con éxito. Para realizar esta tarea se ha decido usar los métodos que ofrece Atom.

En la siguiente captura, se muestra el resultado de la prueba de esta tarea.



- **Tarea DOCFF-19: Patrón Singleton** (Completado)

  En esta tarea no se cumple ningún requisito, ya que es una tarea que surge por necesidad a partir de las dos tareas anteriores "Consola de log" y "Noficaciones", en las cuales para desarrollarlas, se ha creado una clase nueva. A ambas clases se puede acceder desde distintos puntos de la aplicación, lo que puede provocar acumulación de objetos. Para evitar este problema, se ha utilizado un patrón de diseño.

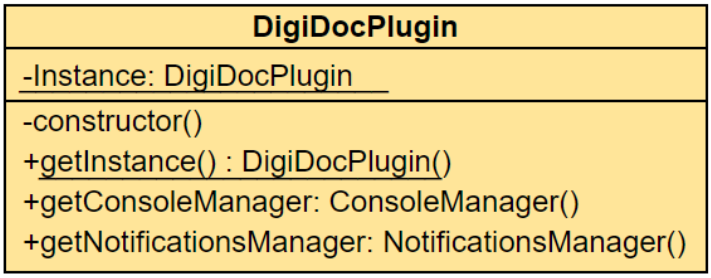
  Nos referimos a patrón de diseño, como una solución concreta en el desarrollo de software, haciendo el código reutilizable y más eficiente.

  En esta tarea se va a utilizar el patrón Singleton que consiste en crear una instancia de un objeto y solo una, para toda nuestra aplicación. Se proporciona un único acceso global que se almacena en nuestro objeto.

  Las características principales del uso de este patrón en esta aplicación son:

  - El constructor de esta clase es privado.
  - Se accede a esta clase mediante un método estático de esta clase.
  - No se debe instanciar si nunca fue utilizada.
  - Al estar internamente auto referenciado, el recolector de basura no actúa.

  Como resultado de esta tarea se ha obtenido la clase *DigiDocPlugin.*



- **Tarea DOCFF-15: Barra de progreso** (Completada)

  El requisito que se cumple en esta tarea es el siguiente:

  Requisitos imprescindibles:

  - El usuario verá en todo momento una barra de progreso, siempre que se esté ejecutando un proceso o subproceso en Atom.

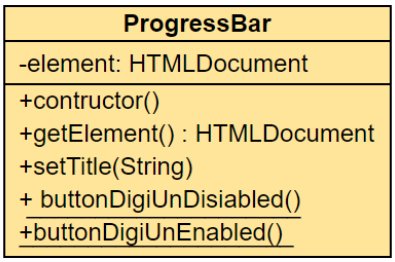
  Las subtareas en que se ha divido esta tarea son:

  - Diseño del prototipo.
  - Crear barra de progreso.

  DISEÑO

  Se ha creado la clase ***ProgressBar*****,** para implementar la barra de estado.

  Al igual que la clase *ConsoleLog*, puede ser llamada desde distintos puntos de la aplicación, por esto se ha añadido un nuevo método a la clase *DigiDocPlugin* (patrón singleton), que devuelva un objeto de esta clase.



Diseño del prototipo

Este diseño es muy básico, simplemente se proyecta un panel que bloquea la pantalla con un título y la barra de progreso. El titulo variará dependiendo del proceso que se esté ejecutando y la barra de progreso es fija (el valor siempre es constante), porque de momento no hay ninguna forma en el backend de saber el estado en el que se encuentra el proceso.

A continuación se muestra la imagen del boceto.

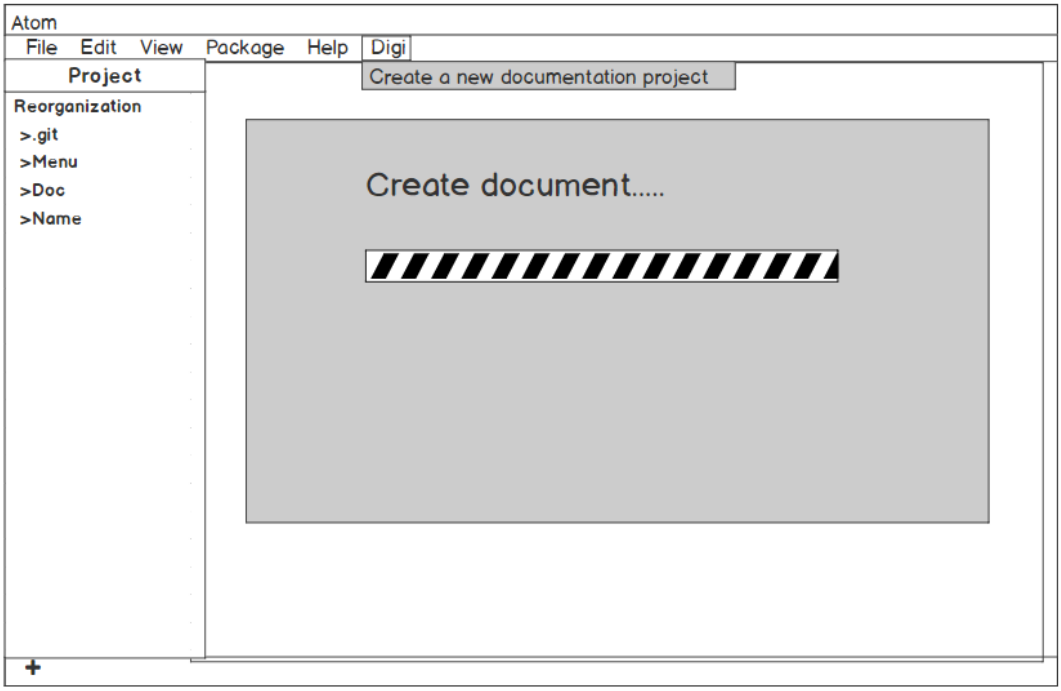


Figura 7. Diseño de la consola de progreso.

IMPLEMENTACIÓN

Crear barra de progreso

El objetivo de esta tarea es evitar la intervención del usuario, mientras se esté ejecutando un proceso, además de proporcionar información al usuario sobre que la tarea está en progreso. Para ello se bloqueará la pantalla a través de este panel y se deshabilitarán los botones del menú, creados en este plugin. En el momento que el proceso finalice, el panel desaparecerá y los botones del menú se volverán a habilitar.

Para implementarla se ha seguido el diseño del prototipo.

- **Tarea DOCFF-3: Crear un proyecto de documentación** (Incompleta)

Los requisitos que hay que cumplir para completar esta tarea son los siguientes:

Requisitos imprescindibles:

- El usuario podrá indicar los parámetros específicos del proyecto de documentación que quiera crear.
- El usuario podrá crear un nuevo proyecto de documentación, siempre que tenga el permiso de la empresa.
- El sistema bloqueará la pantalla mientras se esté ejecutando el proceso que crea el proyecto de documentación.
- El sistema mostrará el output por la pantalla.

Las subtareas en que se ha divido esta tarea son las siguientes:

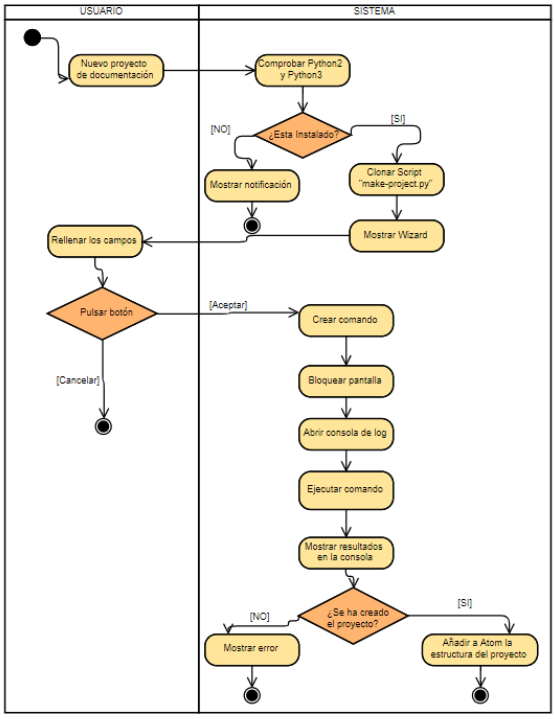
- Especificación del caso de uso: Crear un nuevo proyecto de documentación.
- Diseño del prototipo del wizard.
- Implementar el wizard.
- Ejecutar el proceso para crear un nuevo proyecto de documentación.
- Mostrar output en la consola.

ANALISIS

Especificación del Caso de Uso: Crear proyecto de documentación

A continuación se adjunta la tabla con la especificación del caso de uso. También se ha elaborado el diagrama de Actividad, para entender mejor la secuencia de pasos de este caso de uso.

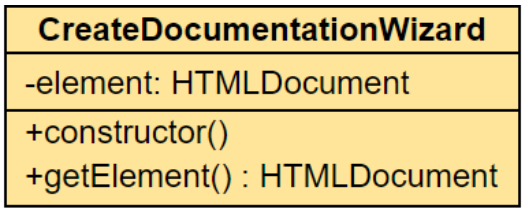
Caso de uso	**Crear proyecto de documentación**
Descripción	El sistema crea un nuevo proyecto de documentación, el cual es almacenado en Stash.
Actor	Usuario
Precondición	El usuario tiene que estar conectado a la red privada de la empresa. El usuario tiene que tener el permiso de la empresa, para crear un nuevo proyecto de documentación. Para ello debe disponer de una clave. El usuario tiene que tener instalado Git, Python y Repo en el
Postcondición	Se creará un nuevo proyecto de documentación en la dirección indicada por el usuario.
Flujo básico	1- El usuario pulsará el botón *Create a new documentation project* del menú. 2- El sistema comprueba que el equipo tiene instalado Python2 y Python3. 3- Se clona el Script 'Make-project.py. Caso de uso anterior "Clonar el script "make-project.py". 4- Se muestra el panel, con los parámetros específicos del proyecto. 5- El usuario rellena los campos y pulsa el botón *Aceptar*. 6- Se crea el comando a través de los datos introducidos en el panel. 7- El sistema abre la pantalla de log, y bloquea la pantalla con una barra de estado. 8- Se ejecuta el comando. 9- Se muestran los resultados en la consola. 10- En caso de haberse creado el proyecto de documentación se muestra la estructura del proyecto en el árbol de carpetas de Atom.
Flujo alternativo	- En el caso de no estar instalado alguna versión de Python, se muestra una notificación y se acaba el proceso. - En el caso de que el usuario pulse el botón *Cancelar*, el proceso se acaba. - En el caso de haber error, al ejecutar el comando, se muestra el error y se acaba el proceso.
Anotaciones	



DISEÑO

Para realizar esta tarea, se han creado varias clases:

- **CreateDocumentationWizard,** en la cual se desarrolla el código del wizard. Para su elaboración sea hecho uso de la interfaz HTMLDocument.

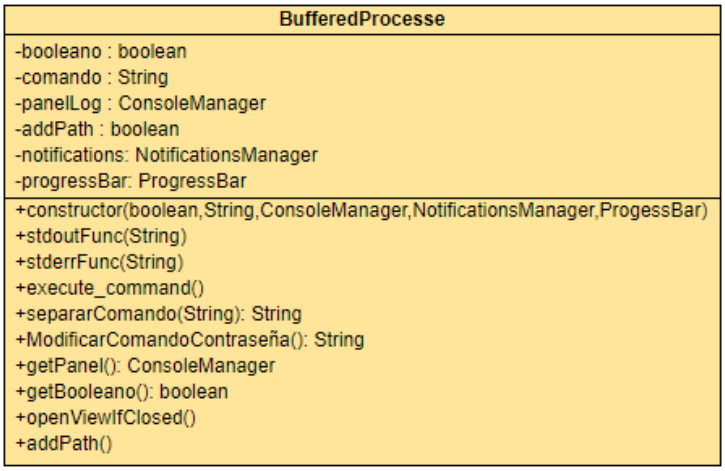


- **BufferedProcesse**: Esta clase ha surgido por necesidad a la hora de crear un proyecto nuevo de documentación. También se utilizará, a la hora de importar y compilar.

  Su función principal es ejecutar el comando creado con los parámetros del wizard, a través de la terminal de comandos. Para ello se hace uso de la clase de Atom *BufferedProcess*, que ofrece varios métodos que son útiles para trabajar con el output y el error de salida. Entre ellos están (args(),stdout(),stderr()).

  Por este motivo, al nombre de la clase se le añadido la última e, para diferenciarla de la propia de Atom.

  Con respecto a los métodos desarrollados, tienen la función esperada (execute_command() se ejecuta el comando, stderrFunc() trabaja con el mensaje de error, stdoutFun() trabaja con el output, modifcarComandoContraseña() se modifica el comando para no mostrar por pantalla el valor de la contraseña y mostrar en su defecto asteriscos).



Diseño del prototipado del wizard

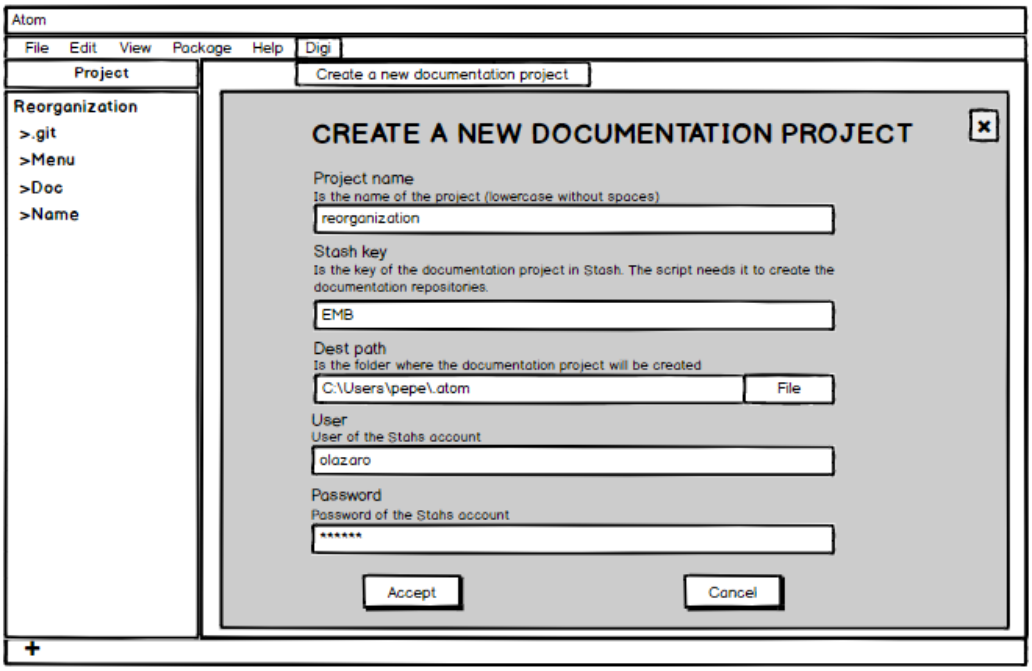


Figura 8. Diseño de la consola de progreso.

Los 5 campos de tipo texto que hay en el panel, tienen que ser rellenados por el usuario y corresponden con los parámetros que son necesarios para poder crear un proyecto de documentación. Son específicos para cada proyecto.

Con el botón *File*, se abre un cuadro de diálogo, para seleccionar una ruta de una carpeta y añadirla al campo.

Los dos botones de abajo, tienen la función esperada, el botón *Accept*, continúa con el proceso para crear un proyecto de documentación, y el botón *Cancel* cierra el panel. Aun cerrando el panel, los datos de los campos se siguen guardando. La funcionalidad de este botón es la misma que el botón *Close* de la parte de arriba.

IMPLEMENTACIÓN

Implementar wizard

En esta subtarea es donde se desarrolla el código del panel para crear un proyecto de documentación.

Durante la prueba de funcionamiento del wizard, Atom en los campos de tipo texto solo me permitía escribir, y no borrar. Tras buscar información, encontré que me faltaba

añadir en el atributo *class*, la propiedad “native-key-bindings”. Este problema me ha causado invertir más tiempo del esperado.

2.2.2. Pruebas

ID	DOCFF 2
SPRINT	2
TAREA A LA QUE PERTENECE	Consola de log
DESCRIPCION	Pulsar el botón de *new terminal*, situado en la parte inferior e izquierda de la pantalla y comprobar que se abre una nueva terminal. Otras comprobaciones: - Se muestra el scrollbar, cuando hay varias líneas. - El botón *clear* borra todo el contenido de la consola. - Se cierra la consola de log sin ningún problema.
RESULTADO	**COMPLETADO**
COMENTARIOS	En el caso de que la consola ya esté abierta, no se podrá abrir otra nueva consola.

ID	DOCFF 12
SPRINT	2
TAREA A LA QUE PERTENECE	Notificaciones
DESCRIPCION	Comprobar que el sistema muestra por pantalla los distintos tipos de notificaciones Éxito, Aviso, Error, Información.
RESULTADO	**COMPLETADO**
COMENTARIOS	En la explicación de esta tarea se adjunta una captura del resultado.

ID	DOCFF 15
SPRINT	2
TAREA A LA QUE PERTENECE	Barra de progreso
DESCRIPCION	Ejecutar un proceso (por ejemplo clonar un proyecto de documentación). Comprobar que se muestra un panel con una barra de progreso indeterminada y que bloquea toda la pantalla.
RESULTADO	**COMPLETADO**
COMENTARIOS	

2.2.3. Revisión del Sprint

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 11)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-2	Consola de log	13h	13h	0%
	Implementar consola de log	8h	10h	**+25%**
	Mostrar Scrollbar	3h	2h	**-33,33%**
	Borrar consola de log	2h	1h	**-50%**
DOCFF-14	Notificaciones	6h	4h	**-33.33%**
DOCFF-19	Patrón Singleton	4h	3h	**-25%**
DOCFF-15	Barra de progreso	3h	3h	0%
	Fase de Diseño	1h	1h	0%
	Crear barra de progreso	2h	2h	0%
DOCFF-2	Crear un proyecto de documentación	11h	12h	**+9.09%**
	Fase de Análisis	1h	1h	0%
	Fase de Diseño	1h	1h	0%
	Implementar el wizard	9h	10h	**+11,11%**
TOTAL		**37 h**	**35 h**	**-5,4%**

Tabla 11. Comparativa de los tiempos estimados y reales de las tareas del Sprint 2.

Como conclusión de esta tabla, podemos decir que ha sido un Sprint positivo. Las tareas que se han desarrollado no han sido muy complejas, y no se han encontrado muchos problemas a la hora de implementarlas, como resultado de ello se ha producido una pequeña desviación positiva (-5,4%).

2.3. Sprint 3

Las tareas que se han planificado para completar este Sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-3	Crear un proyecto de documentación	20h
DOCFF-21	Abrir un cuadro de diálogo	2h
DOCFF-20	Añadir a Atom la estructura del proyecto de documentación	4h
DOCFF-4	Clonar un proyecto de documentación	9h
TOTAL		**35h**

Tabla 12. Product backlog del Sprint 3.

2.3.1. Tareas

- **Tarea DOCFF-3: Crear un proyecto de documentación** (Completada)

IMPLEMENTACIÓN

Ejecutar el proceso para crear un nuevo proyecto de documentación

Esta tarea consiste en desarrollar un evento, cuando se pulsa el botón *Accept* del wizard para crear un proyecto de documentación. Se comprueba que ningún campo está vacío y se forma el comando a ejecutar.

Antes de ejecutar el comando (llama al backend), se bloquea la pantalla con la barra de progreso creada en la anterior tarea. Cuando acaba de ejecutarse el proceso, desaparece la barra de progreso.

A la hora de ejecutar el comando, me ha surgido un problema porque no podía utilizar el método creado en la tarea DOCFF-12 (*execute_command_simple),* ya que de esta forma hasta que no se acaba de ejecutar la promesa, no se puede mostrar el output.

Como se tiene que mostrar el output por la pantalla a medida que se va generando, he tenido que buscar una nueva forma de ejecutar comandos a través de una Shell. Para solucionarlo, se ha decidido hacer uso de la clase *BufferedProcess*, una clase propia de Atom que ejecuta procesos en segundo plano y ofrece dos métodos (*stdout*(), *stderr*()), que me permiten trabajar con el output una vez que se genera. El output que se genera, se va comparando con unos patrones que he desarrollado, y dependiendo del resultado se muestra la notificación de éxito, información o error.

Todo este desarrollo del código para ejecutar los comandos, se ha realizado en la clase BufferedProcesse (explicada en el apartado de diseño de esta misma tarea).

Mostrar output en la consola

Esta tarea es muy sencilla, consiste en escribir el output generado de ejecutar el comando en la consola de log. Para ello se ha implementado el método setTexto().

- **Tarea DOCFF-21: Abrir un cuadro de diálogo** (Completada)

El requisito que se cumple en esta tarea es el siguiente:

Requisitos imprescindibles:

- El usuario en los wizards, podrá seleccionar las direcciones de las carpetas, a través de un cuadro de diálogo.

IMPLEMENTACIÓN

Abrir un cuadro de diálogo

Esta tarea consiste en desarrollar un evento, para que cuando se pulse en el botón *File* del wizard (crear un proyecto de documentación), se abra un cuadro de diálogo, que

solo permita seleccionar la ruta de una carpeta. Esta ruta es guardada y se muestra directamente en el campo de texto correspondiente (campo destination path del wizard).

- **Tarea DOCFF-20: Añadir a Atom la estructura del proyecto de documentación** (Completada)

  El requisito que se cumple en esta tarea es el siguiente:

  Requisitos imprescindibles:

  - El sistema mostrará la estructura del proyecto generado, en el IDE.

  IMPLEMENTACIÓN

  Añadir a Atom la estructura del proyecto de documentación

  Una vez que el proceso ha terminado, se comprueba que se ha creado el proyecto de documentación, en ese caso se añade en el tree-view[7] de Atom el path del archivo creado.

  A continuación se muestra el resultado de esta tarea.

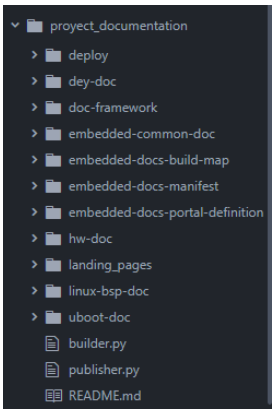


Figura 9. Estructura de un proyecto de documentación.

- **Tarea DOCFF-4: Clonar/importar un proyecto de documentación** (Completada)

  Los requisitos que hay que cumplir para completar esta tarea son los siguientes:

  Requisitos imprescindibles:

  - El usuario podrá indicar los parámetros específicos del proyecto de documentación que quiera clonar/importar.
  - El usuario podrá clonar/importar un proyecto de documentación.
  - El sistema bloqueará la pantalla mientras se esté ejecutado el proceso que clona el proyecto de documentación.
  - El sistema mostrará el output por la pantalla.

  Las subtareas en que se ha divido esta tarea son las siguientes:

  - Especificación del caso de uso: Clonar un nuevo proyecto de documentación.
  - Diseño del prototipo del wizard.
  - Implementar el wizard.
  - Ejecutar el proceso para importar proyecto de documentación.

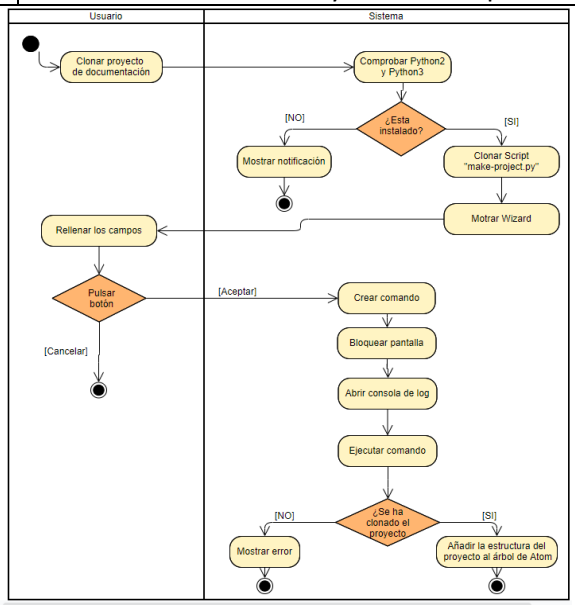
[7] (vista de árbol), Elemento gráfico que presenta una vista jerárquica de la carpeta.

ANÁLISIS

Especificación del caso de uso: Clonar proyecto de documentación

Esta tarea consta de la especificación del caso de uso y de un diagrama de actividad

Caso de uso	**Clonar proyecto de documentación**
Descripción	El sistema clona un proyecto de documentación, que exista en Stash.
Actor	Usuario
Precondición	El usuario tiene que estar conectado a la red privada de la empresa. El usuario tiene que tener instalado Git, Python y Repo en el PC.
Postcondición	Se clona un proyecto de documentación en la dirección indicada por el usuario.
Flujo básico	1- El usuario pulsa el botón *Clone a documentation project* del menú. 2- El sistema comprueba que el equipo tiene instalado Python2 y Python3. 3- Se clona el Script 'Make-project.py. Caso de uso anterior "Clonar el script "make-project.py". 4- Muestra panel, con los parámetros específicos del proyecto. 5- El usuario rellena los campos y pulsa el botón *Aceptar*. 6- Se crea el comando a través de los datos introducidos en el panel. 7- El sistema abre la pantalla de log, y bloquea la pantalla con una barra de estado. 8- Se ejecuta el comando. 9- Se muestran los resultados en la consola. 10- En caso de haberse clonado el proyecto de documentación se muestra la estructura del proyecto en el árbol de carpetas de Atom.
Flujo alternativo	- En el caso de no estar instalado alguna versión de Python, se muestra una notificación y se acaba el proceso. - En el caso de que el usuario pulse el botón *Cancelar*, el proceso se acaba. - En el caso de haber error, al ejecutar el comando, se muestra el error y se acaba el proceso.



DISEÑO

Para completar esta tarea, se ha creado la clase ***CloneDocumentationWizard***, donde se desarrolla el código para implementar el wizard para clonar un proyecto de documentación. Se hace uso de la interfaz HTMLDocument.

CloneDocumentationWizard
-element: HTMLDocument
+contructor() +getElement() : HTMLDocument

Diseñar el prototipo del wizard para clonar un proyecto de documentación

En esta tarea se ha diseñado un panel, con los parámetros específicos para importar de Stash un proyecto de documentación. Son 5, los tres primeros son campos vacíos, que el usuario estará obligado a rellenar si quiere seguir con el progreso, y los otros dos (Branch, y Manifest name), son campos que por defecto el sistema ya ha rellenado y que el usuario tendrá que modificar en casos muy excepcionales.

A continuación se adjunta una imagen del diseño. (Figura 10)

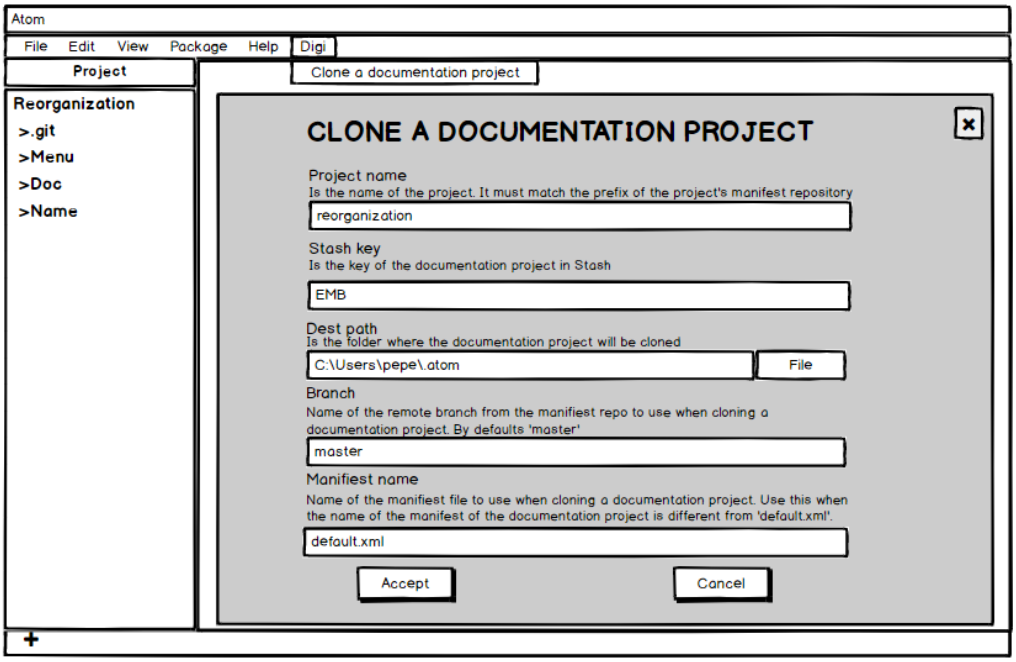


Figura 10. Diseño del wizard para clonar un proyecto de documentación.

Implementar wizard

Esta tarea consiste en implementar el código del panel con los parámetros específicos del proyecto de documentación que queremos clonar, el cual, se ha diseñado arriba.

Ejecutar el proceso para importar proyecto de documentación

Esta tarea es muy semejante a la de crear un proyecto de documentación. Se comprueba que los tres primeros campos no están vacíos y que el valor de los otros dos (Branch y Manifiest name) no se ha cambiado. Dependiendo de los valores obtenidos se crea el comando (hace uso del backend) y se ejecuta a través de la clase *BufferedProcesse*. La pantalla también se bloquea por el panel de la barra de progreso mientras se está ejecutando el proceso.

También se ha añadido nuevas comprobaciones en el método *stdoutFunc*(), (trabaja con el output generado del proceso), perteneciente a la clase *BufferedProcesse,* donde dependiendo del resultado del output se genera una notificación de error o éxito.

2.3.2. Pruebas

ID	DOCFF 3
SPRINT	3
TAREA A LA QUE PERTENECE	Crear un proyecto de documentación
DESCRIPCION	Pulsar el botón de Digi *Create*. Rellenar los datos del wizard. Comprobar que se bloquea la pantalla, se muestra el output en la consola de log y se crea un nuevo proyecto de documentación tanto en nuestro PC como en el servidor de Stash.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 21
SPRINT	3
TAREA A LA QUE PERTENECE	Abrir un cuadro de diálogo
DESCRIPCION	Pulsar el botón *File* del wizard. Comprobar que se abre un cuadro de diálogo que permite seleccionar la ruta de la carpeta. La ruta se tiene que añadir al campo de texto.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 20
SPRINT	3
TAREA A LA QUE PERTENECE	Añadir a Atom la estructura del proyecto de documentación
DESCRIPCION	Una vez creado o clonado en nuestro PC un proyecto de documentación. Comprobar cómo se añade a nuestro IDE el path del proyecto.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 4
SPRINT	3
TAREA A LA QUE PERTENECE	Clonar/importar un proyecto de documentación
DESCRIPCION	Pulsar el botón de Digi *Clone*. Rellenar los datos del wizard. Comprobar que se bloquea la pantalla, se muestra el output en la consola de log y se crea creado un nuevo proyecto de documentación en nuestro PC.
RESULTADO	**COMPLETADO**
COMENTARIOS	

2.3.3. Revisión del Sprint

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 13)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-3	Crear un proyecto de documentación	20h	16h	**-20%**
	Ejecutar el proceso para crear un nuevo proyecto de documentación.	17h	14h	**-17.7%**
	Mostrar output en la consola	3h	2h	**-33,33%**
DOCFF-21	Abrir un cuadro de diálogo	2h	2h	0%
DOCFF-20	Añadir a Atom la estructura del proyecto de documentación	4h	2h	**-50%**
DOCFF-4	Clonar/importar un proyecto de documentación.	24h	15h	**-37.5%**
	Fase de Análisis	1h	1h	0%
	Fase de Diseño	1h	1h	0%
	Implementar el wizard	8h	5h	**-37.5%**
	Ejecutar el proceso para importar un proyecto de documentación	14h	8h	**-42.85%**
TOTAL		**50 h**	**35 h**	**-30%**

Tabla 13. Comparativa de los tiempos estimados y reales de las tareas del Sprint 3.

Como vemos en la anterior tabla, se han terminado todas las tareas planificadas para este Sprint, y como ha sobrado tiempo, se ha avanzado en la siguiente tarea. Esto ha provocado que todas las tareas (excepto DOCFF-21), hayan tenido una desviación positiva, llegando a obtener una desviación total del 30%.

Esta desviación tan elevada de 15 horas, se ha debido a dos tareas en particular DOCFF-3 y DOCFF-4. De esta forma, se han podido añadir distintas subtareas que no estaban programadas para este Sprint, llegando incluso a completar la tarea DOCFF-2.

2.4. Sprint 4

Las tareas que se han planificado para completar este Sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-16	Comprobar si existe un proyecto de documentación	3h
DOCFF-7	Configurar un proyecto de documentación	12h
DOCFF-5	Compilar un proyecto de documentación de forma local	11h
DOCFF-22	Tabbing	3h
DOCFF-6	Lanzar el proyecto de documentación	6h
TOTAL		**35h**

Tabla 14. Product backlog del Sprint4.

2.4.1. Tareas

- **Tarea DOCFF-16: Comprobar si existe un proyecto de documentación** (Completada)

  El requisito que hay que cumplir para completar esta tarea es el siguiente:

  Requisitos imprescindibles:

  - El sistema detectará si en el working-tree de Atom, hay una ruta perteneciente a un proyecto de documentación.

  Requisitos secundarios:

  - El sistema deshabilitará la funcionalidad de algunos botones, en caso de no disponer de un proyecto de documentación.

  IMPLEMENTACIÓN

  Comprobar proyecto de documentación

  Esta tarea es esencial, ya que es necesaria para el resto de tareas que se realizarán a posteriori. Consiste en comprobar si las rutas del working-tree de Atom, corresponden con un proyecto de documentación.

  Tras una reunión con los tutores de la empresa, se ha decidido, que para que una carpeta corresponda con un proyecto de documentación tiene que cumplir dos requisitos. Debe contener los ficheros Schema.yml y Portal-config.yml y el otro es que tiene que tener algún repositorio que acabe en las siguientes terminaciones (-build-map,-doc,-definition,-manifiest).Estos requisitos se ampliarán en un futuro, en caso de que haya tiempo.

  Para la implementación de esta tarea se ha importado algún módulo de Atom.

- **Tarea DOCFF-7: Configurar un proyecto de documentación** (Completada)

  Los requisitos que hay que cumplir para completar esta tarea son los siguientes:

  Requisitos imprescindibles:

  - El usuario podrá modificar las preferencias de un proyecto de documentación, sin necesidad de tener que acceder al fichero.

  Requisitos secundarios:

  - El usuario podrá recuperar los valores por defecto de las preferencias de un proyecto de documentación.

  Las subtareas en que se ha divido esta tarea son las siguientes:

  - Especificación del caso de uso: Configurar preferencias del proyecto de documentación.

- Diseño del prototipo de la página.
- Implementar página de preferencias.
- Modificar fichero “Portal-config.yml”.

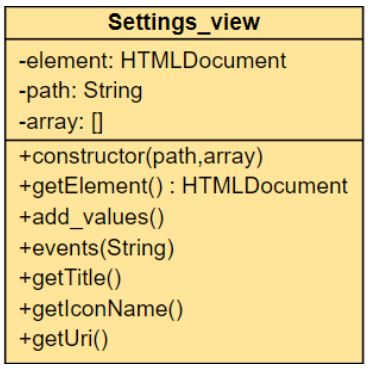
ANÁLISIS

Especificación del caso de uso: Configurar preferencias del proyecto de documentación.

Caso de uso	**Configurar preferencias del proyecto de documentación**
Descripción	El usuario configura las características propias del proyecto de documentación.
Actor	Usuario
Precondición	El usuario tiene que estar conectado a la red privada de la empresa. El usuario tiene que tener abierto un proyecto de documentación, en caso contrario el botón esta deshabilitado.
Postcondición	Se habrán cambiado las configuraciones propias del proyecto
Flujo básico	1- El usuario pulsa el botón *Project preferences* del menú. 2- El sistema muestra una página con todas las settings del proyecto (por defecto, ya tienen un valor asignado). 3- El usuario modifica el valor de las variables que quiera. 4- Si el usuario pulsa el botón *Save*, el sistema guarda el valor de las variables. 5- El sistema modifica el fichero *Portal-config.yml*, del proyecto de documentación, con los nuevos valores que el usuario ha introducido.
Flujo alternativo	- En el caso de que el usuario pulse el botón *Cancel*, el sistema no guarda ningún valor modificado. - En el caso de que el usuario pulse el botón *Default*, el sistema muestra los valores por defecto, cuando se crea un proyecto de documentación.
Anotaciones	

DISEÑO

Para completar esta tarea, se ha creado la clase ***Settings_view***, que incluye el desarrollo de la página para configurar las preferencias del proyecto de documentación.



Diseñar página de las preferencias de un proyecto de documentación

En esta tarea, se diseña el prototipo de una página, que muestra las 8 preferencias, que el usuario puede modificar, en un proyecto de documentación.

Tras haber realizado varios bocetos, a continuación se explica el proceso seguido desde la versión inicial, hasta la versión final.

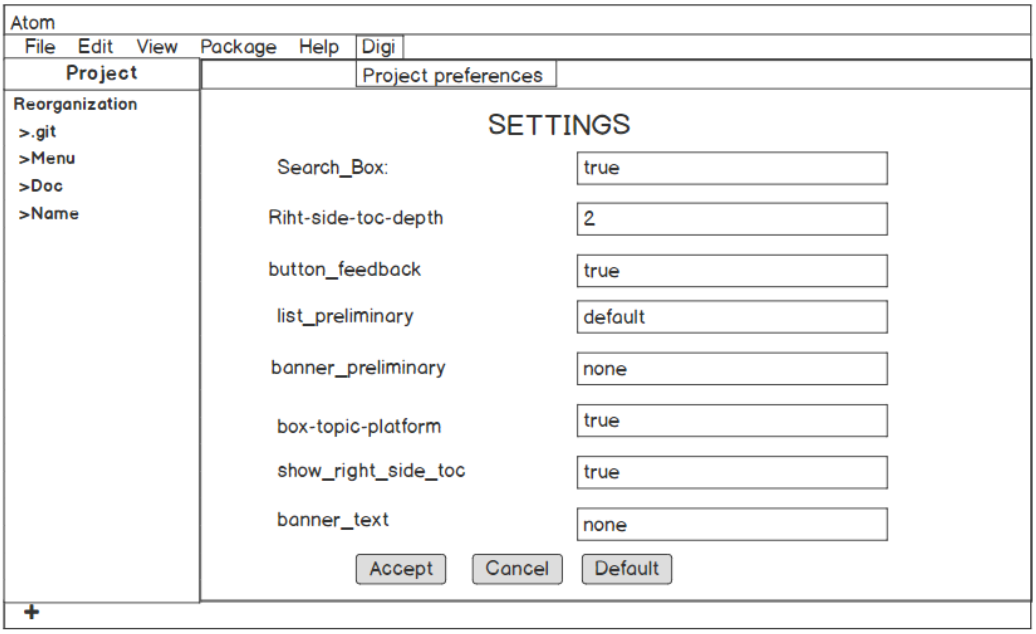


Figura 11. Versión inicial del prototipo de la página de preferencias.

En un primer momento, se pretendía diseñar una página sencilla, donde se pudieran mostrar todas las preferencias a la vez en la pantalla, sin necesidad de tener que usar el scrollbar. Todas las settings disponían de un campo de texto, con su valor asociado, en todos los casos de tipo String. También, se muestra un botón en el centro, con el que se guardan los nuevos datos introducidos por el usuario.

Tras realizar varias reuniones telemáticas con los tutores de la empresa, donde se presentaron las nuevas propuestas de diseño, nos decantamos como prototipo final, una página con un diseño más recargado, que facilita y simplifica el trabajo del usuario.

Entre los cambios realizados, se encuentran los siguientes:

- Dar un nombre representativo a la página, acompañado de un icono.
- Incluir una pequeña descripción de la página, a continuación del Título.
- Añadir en un nuevo párrafo el nombre del proyecto, al que corresponde las preferencias.
- Separar todas las preferencias, a través de una borra horizontal.
- Las settings que son un booleano, marcarlas a través de checbox, evitando así tener que escribir True o False al usuario.
- Añadir una pequeña descripción, a continuación de cada setting.
- Colocar los botones en los extremos de la pantalla, para seguir así, con la estructura de las páginas de Atom.
- Añadir un scrollbar, en la parte derecha de la pantalla.

A continuación se muestra el diseño, correspondiente a la versión final.

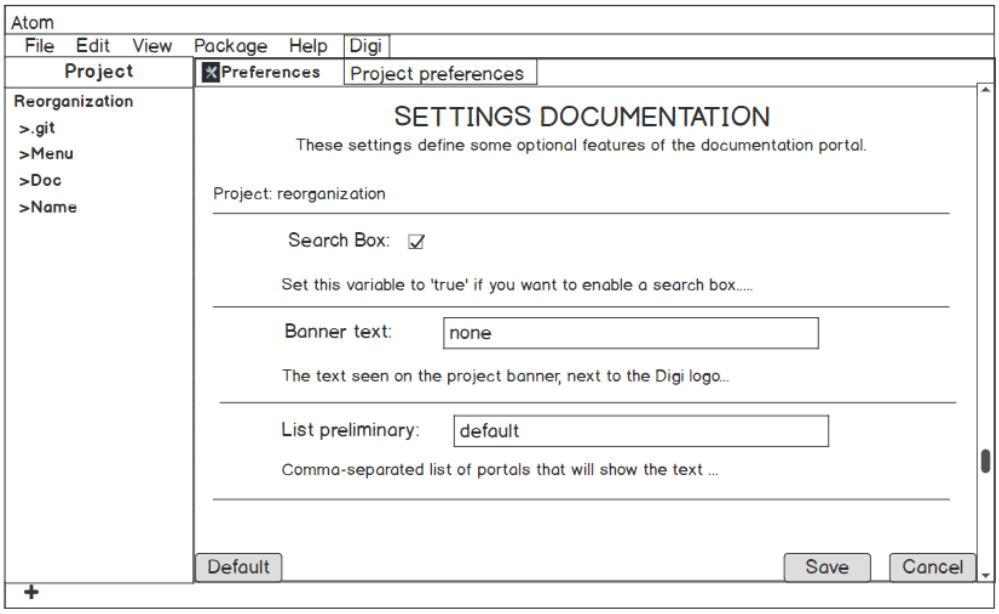


Figura 12. Versión final del prototipo de la página de preferencias.

IMPLEMENTACIÓN

Implementar página de preferencias

En esta tarea se desarrolla el código para crear una página, donde se pueda configurar las preferencias de un proyecto de documentación.

Este desarrollo se ha dividido en tres partes. En primer lugar, se ha desarrollado el código de la clase Settings_View, donde se crea la página de preferencias. Después se ha desarrollado un evento para cuando se pulsa el botón *Project Preferences* del menú de Digi. Y por último el desarrollo de los eventos correspondientes a los botones (*Default* y *Cancel*).

La dificultad de esta tarea, reside a la hora de crear en nueva página las preferencias de un proyecto de documentación. Para ello se ha hecho uso de las promesas, un objeto propio de JavaScript.

Modificar fichero "*Portal-config.yml*"

Esta tarea corresponde al momento en el que usuario pulsa el botón de *save*, en la página de las preferencias.

Se busca el fichero "*Portal-Config.yml*", en la carpeta correspondiente al proyecto de documentación y se modifican los antiguos valores de las settings, por los introducidos por el usuario.

Para su implementación, se ha utilizado el módulo "*Fs*", que nos facilita el proceso de lectura y escritura en un fichero.

Un problema que me he encontrado durante la implementación de esta tarea, ha sido a la hora de modificar el fichero y guardar todos los comentarios. En un principio intentaba parsear el código yml a JSON, pero no encontraba la forma de guardar los comentarios. Para solucionar este problema he incorporado otro módulo "*yaml*", que me permite trabajar con características del lenguaje YAML que no son directamente compatibles con los tipos de datos nativos de JavaScript, como los comentarios.

- **Tarea DOCFF-5: Compilar un proyecto de documentación de forma local** (Completada)

  Los requisitos que hay que cumplir para completar esta tarea son los siguientes:

  Requisitos imprescindibles:

  - El usuario podrá compilar un proyecto de documentación de forma local.

  Requisitos secundarios:

  - El usuario podrá parar la ejecución de un proceso de compilación, en el momento que quiera.
  - El usuario podrá ejecutar el proceso de compilación en un segundo plano.

  Las subtareas en que se ha divido esta tarea son las siguientes:

  - Especificación del caso de uso: Compilar un proyecto de documentación.
  - Ejecutar el proceso para compilar un proyecto de documentación de forma local.

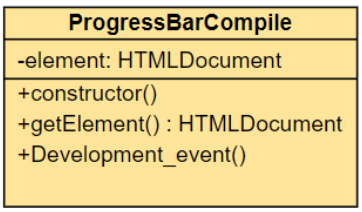
  ### ANÁLISIS

  Especificación del caso de uso: Compilar un proyecto de documentación

Caso de uso	**Compilar un proyecto de documentación**
Descripción	El sistema compila un proyecto de documentación, y crea una carpeta con los ficheros HTML, los cuáles, se podrán publicar en la página web de la empresa.
Actor	Usuario
Precondición	El usuario tiene que estar conectado a la red privada de la empresa. El usuario tiene que tener abierto un proyecto de documentación, en caso contrario él botón esta deshabilitado. El usuario tiene que tener instalado Git, Python y Repo en el PC.
Postcondición	Se crea una carpeta llamada *deploy* en el proyecto de documentación, con los ficheros html.
Flujo básico	1- El usuario pulsa el botón *Compile a documentation project* del menú de Digi. 2- El sistema ejecuta el comando para compilar el proyecto de documentación en forma local. 3- El sistema muestra un panel con una barra de progreso indeterminada, que bloquea la pantalla. 4- El sistema mostrará el output en la consola de log. 5- El usuario podrá ejecutar el proceso en segundo plano, pulsando el botón *Run in background* del panel bloqueante. 6- El usuario podrá parar el proceso de ejecución, pulsando el botón *Stop* del panel bloqueante.
Flujo alternativo	- En el caso de que el proceso se ejecute en segundo plano, se creará un nuevo botón en la terminal, el cual parará el proceso de ejecución.
Anotaciones	

  ### DISEÑO

  ProgressBarCompile: Esta clase ha sido creada para implementar el panel que se muestra mientras se está compilando un proyecto de documentación. Es un panel más complejo que el usado para el proceso de creación y clonación, ya que se han añadido dos botones. En el método *Development_event*, es donde se desarrolla el evento de los botones.



IMPLEMENTACIÓN

Ejecutar el proceso para compilar un proyecto de documentación de forma local

Esta tarea la he subdivido en dos partes a la hora de implementarla:

Por una parte, el desarrollo del evento cuando se pulsa el botón *Compile* del menú de Digi. Consiste en desplazar a través de la consola de comandos (CMD), el directorio actual de trabajo a la dirección del proyecto de documentación y ejecutar el comando que llama al backend, para compilar un proyecto de forma local.

Al desarrollar este código, me he encontrado con un problema, ya que al ejecutar en una sentencia los dos comandos a través del símbolo '*&*', la clase *BuferredProcesse* (ejecuta los comandos a través de una terminal de comandos) me daba error al no detectar como símbolo el signo "ampersand". Tras buscar información a través del código de otros paquetes de Atom, encontré la solución, que consiste en añadir un atributo opcional (*Shell: true*), al constructor de la clase BuferredProcess. Este problema me llevó invertir más tiempo del esperado porque en la documentación de Atom del propio constructor de esta clase, no aparece explicado dicho atributo.

La otra parte de la tarea, consiste en implementar un nuevo panel (clase *ProgressBarCompile*), que bloquea la pantalla mientras se está ejecutando el proceso.

Dicho panel consta de una barra de progreso indeterminada, y dos botones en su parte inferior. Uno de ellos, con nombre '*Run in Background*', que permite quitar el panel y ejecutar el proceso en segundo plano, y el otro '*STOP*', mata el proceso y termina con la compilación. A parte, en la clase *ConsoleManager* (donde se desarrolla el código de la consola de log), se ha añadido un nuevo botón oculto por defecto, con la misma funcionalidad que el botón *STOP*, que se hace visible cuando el proceso se está ejecutando en segundo plano.

La dificultad de esta parte no reside a la hora de ejecutar el comando en segundo plano, ya que a través de la clase BufferedProcess de Atom, se realiza automáticamente, sino en el desarrollo del botón *STOP* para matar el proceso.

Atom ofrece un método que mata el proceso a través de su PID, sin embargo con este método no me funcionaba porque el backend no crea solo un proceso, sino un proceso con varios subprocesos, a los que a través de esta forma no se tenían los permisos suficientes para matarlos. Para solucionar este problema, he tenido que invertir más tiempo del estimado y he encontrado dos posibilidades. La primera, consiste en modificar el Script del backend, para ofrecer la posibilidad de matar el proceso a través de un nuevo parámetro, y la siguiente que es la que he usado, la más rápida y eficiente, matar el proceso y los subprocesos a través de la terminal de comandos de forma forzosa.

Debido a estos dos problemas, se ha producido una gran desviación que está reflejada en el apartado 2.4.3 Revisión del Sprint.

- **Tarea DOCFF-22: Tabbing** (Completada)

El requisito que hay que cumplir para completar esta tarea es el siguiente:

Requisitos secundarios:

- El usuario podrá usar el tabulador para cambiar de campo en el wizard.

IMPLEMENTACIÓN

Tabbing

Esta tarea consiste en desarrollar un evento, para que cuando se pulse la tecla (tab) y se encuentre un panel habilitado se cambie el foco de un campo a otro.

Se ha hecho uso del fichero keymap que ofrece Atom, un fichero en formato JSON, que asocia el significado de pulsar una tecla o conjunto de teclas con un método.

2.4.2. Pruebas

ID	DOCFF 16
SPRINT	4
TAREA A LA QUE PERTENECE	Comprobar proyecto de documentación
DESCRIPCION	Añadir al working-tree de Atom, una ruta que corresponda con un proyecto de documentación. Comprobar cómo se habilitan los botones del menú de Digi. Después eliminarlo, y comprobar como los botones se deshabilitan.
RESULTADO	**COMPLETADO**

ID	DOCFF 7
SPRINT	4
TAREA A LA QUE PERTENECE	Configurar un proyecto de documentación
DESCRIPCION	Abrir la página de preferencias y cambiar el valor de alguna settings. Comprobar cómo se cambia el valor en el fichero "Portal-config.yml".
RESULTADO	**COMPLETADO**

ID	DOCFF 5
SPRINT	4
TAREA A LA QUE PERTENECE	Compilar un proyecto de documentación de forma local
DESCRIPCION	Pulsar el botón *Compile* del menú de Digi. Comprobar que se bloquea la pantalla mientras se está ejecutando el proceso. Al finalizar el proceso, comprobar que se ha compilado el proyecto (en la carpeta *deploy*, ver que se han creado ficheros HTML a partir de los ficheros adoc).
RESULTADO	**COMPLETADO**

ID	DOCFF 22
SPRINT	4
TAREA A LA QUE PERTENECE	Tabbing
DESCRIPCION	Seleccionar un campo de texto del wizard, y pulsar la tecla de tabulación, el sistema cambia el foco al siguiente campo de texto.
RESULTADO	**COMPLETADO**

2.4.3. Revisión del Sprint

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 15)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-16	Comprobar si existe un proyecto de documentación	3h	3h	0%
DOCFF-7	Configurar un proyecto de documentación	12h	14h	**+16.66%**
	Fase de Análisis	1h	1h	0%
	Fase de Diseño	1h	2h	**+100%**
	Implementar la página	7h	7h	0%
	Modificar fichero "Portal-config.yml"	3h	4h	**+33.3%**
DOCFF-5	Compilar un proyecto de documentación de forma local	11h	16h	**+45.45%**
	Fase de Análisis	1h	1h	0%
	Ejecutar el proceso para compilar un proyecto de documentación de forma local	10h	15h	**+50%**
DOCFF-22	Tabbing	3h	2h	**-33.33%**
TOTAL		**29 h**	**35 h**	**+20.68%**

Tabla 15. Comparativa de los tiempos estimados y reales de las tareas del Sprint 4.

Como conclusión de este Sprint, se puede decir que ha sido un Sprint poco productivo, llegándose a producir una desviación negativa del (20%). Este hecho ha provocado no poder empezar a desarrollar la tarea (DOCFF-6), estimada con 6h.

El motivo por el que no se han podido invertir las 35 horas estimadas, ha sido porque algunas tareas han sido más costosas de lo esperado, obligándome a invertir más tiempo. La principal desviación ha sido causada por la tarea DOCFF-5. A parte de los dos problemas comentados en la explicación de la tarea, me he encontrado con un problema a la hora de hacer la comprobación, ya que al ejecutar el comando "*Repo init*[8]", me daba error. Tras buscar información y pedir ayuda a los tutores de la empresa, se dio con la solución, que se debía a un desfase de versiones entre las herramientas repo y GPG[9].

Un dato curioso que nos encontramos al ver la tabla, corresponde con el diseño del prototipo de la tarea DOCFF-7, donde se ha producido una desviación del 100%. No tiene mucha importancia con respecto a la desviación final, pero nos indica, que en la fase de Diseño, en algunos bocetos, se pueden llegar a producir varias versiones.

[8] Comando de Repo, que sirve para instalar Repo en el directorio de trabajo. Se crea un directorio .repo/ con repositorios de Git para el código fuente de Repo y los archivos de manifiesto estándar.
[9] También conocido como GNU Privacy Guard, es una herramienta de cifrado de firmas digitales. Se usa en el backend para trabajar con el nombre de usuario y contraseña.

2.5. Sprint 5

Las tareas que se han planificado para completar este Sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-6	Lanzar el proyecto de documentación	8h
DOCFF-17	Añadir una comprobación de la compilación en el script de construcción	11h
DOCFF-18	Sincronizar el repositorio del Script	12h
DOCFF-8	Detectar los principales errores	4h
TOTAL		**35h**

Tabla 16. Product backlog del Sprint 5.

2.5.1. Tareas

- **Tarea DOCFF-6: Lanzar el proyecto de documentación** (Completada)

  El requisito que hay que cumplir para completar esta tarea es el siguiente:

  Requisitos imprescindibles:

  - El usuario podrá lanzar un proyecto de documentación, en el navegador por defecto, sin tener que subirlo a un servidor externo.
  - El usuario podrá finalizar el proceso de servir la documentación en el momento que quiera.

  IMPLEMENTACIÓN

  Lanzar el proyecto de documentación

  El objetivo de esta tarea es poder mostrar/abrir un proyecto de documentación que se haya compilado previamente, para poder ver el resultado del proyecto en un navegador web sin tener que cargarlo en un servidor externo.

  En primer lugar, se ha añadido una variable a las preferencias del plugin, permitiendo al usuario elegir si quiere servir o no la documentación.

  En segundo lugar, se ha modificado la tarea del anterior Sprint donde se compila un proyecto de documentación, modificando el comando, dependiendo de si quiere servir o no la documentación de forma local.

  Por último, se ha desarrollado un método que abre el navegador web por defecto, en la dirección en la que se sirve la documentación.

- **Tarea DOCFF-17: Añadir una comprobación de la compilación en el script de construcción** (Completada)

  El requisito que hay que cumplir para completar esta tarea es el siguiente:

  Requisitos secundarios:

  - El script no tendrá que compilar el proyecto de documentación, si no se ha producido ninguna modificación.

  IMPLEMENTACIÓN

  Añadir una comprobación de la compilación en el script de construcción

  Esta tarea no se realiza sobre el editor de texto Atom, si no sobre el Script del backend que se encarga de compilar un proyecto de documentación.

El script recorre en iteración todos los archivos fuente del proyecto de documentación y compara la fecha de modificación con la última fecha de modificación de la carpeta de implementación (carpeta *deploy* que se crea cuando se compila un proyecto de documentación).

Si alguno de los archivos de origen se ha modificado después de la última compilación, el script compila de nuevo el proyecto de documentación. En el caso contrario lo lanza directamente.

Este proceso se ha desarrollado solo en el caso de que el usuario aplica el parámetro 'server' (-s).

El lenguaje de programación que se ha utilizado en esta tarea no es JavaScript como en las demás tareas, si no Python.

- **Tarea DOCFF-18: Sincronizar el repositorio del Script** (Completada)

El requisito que hay que cumplir para completar esta tarea es el siguiente:

Requisito imprescindible:

- El IDE siempre tiene que tener la última versión del backend, para no tener que preocuparse de actualizarlo manualmente.

IMPLEMENTACIÓN

Sincronizar el repositorio del Script

El desarrollo del backend, con el que trabaja la empresa para documentar un proyecto, todavía no ha terminado. Esto quiere decir, que a medida que se usa este plugin, los desarrolladores de Digi, pueden ir actualizando los repositorios con los que estamos trabajando (Doc-project-maker, Doc-framework…).

Para evitar tener diferentes versiones de estos repositorios, se ha creado esta tarea, que comprueba la fecha de modificación del repositorio local con el repositorio remoto (Stash). En el caso de que se hubiera realizado alguna actualización, el sistema clonará automáticamente estos repositorios, en la ruta correspondiente.

- **Tarea DOCFF-8: Detectar los principales errores** (Completada)

El requisito que hay que cumplir para completar esta tarea es el siguiente:

Requisito imprescindible:

- El sistema detectará los principales errores que se produzcan.

IMPLEMENTACIÓN

Detectar los principales errores

Esta tarea surge a raíz de las distintas pruebas que se han realizado al comprobar las tareas. Se han ido apuntando los principales errores que se han encontrado al realizar las pruebas, y se les ha asociado un mensaje característico, que se le mostrará al usuario por medio de una notificación. En el caso de que se produzca un error que no se ha sido detectado en esta tarea, se muestra un mensaje general.

2.5.2. Pruebas

ID	DOCFF 6
SPRINT	5
TAREA A LA QUE PERTENECE	Lanzar el proyecto de documentación
DESCRIPCION	Pulsar el botón *Launch documentation project* del menú de Digi. Comprobar cómo se sirve la documentación en el navegador por defecto.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 17
SPRINT	5
TAREA A LA QUE PERTENECE	Añadir una comprobación de la compilación en el script de construcción
DESCRIPCION	Una vez, teniendo el proyecto de documentación ya compilado, ejecutar el Script de compilación del backend con el parámetro de servir la documentación activado. Comprobar en la consola de log, como no se compila de nuevo el proyecto y se sirve la documentación.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 18
SPRINT	5
TAREA A LA QUE PERTENECE	Sincronizar el repositorio del Script
DESCRIPCION	Realizar un cambio en el repositorio local o remoto del *doc-project-maker.* Comprobar como al iniciar el editor de texto Atom, se clona de nuevo dicho repositorio.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF 8
SPRINT	5
TAREA A LA QUE PERTENECE	Detectar los principales errores
DESCRIPCION	Intentar clonar un proyecto de documentación, sin rellenar todos los parámetros del wizard. Comprobar como el sistema muestra una notificación de error y no se cierra el panel.
RESULTADO	**COMPLETADO**
COMENTARIOS	

2.5.3. Revisión del Sprint.

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 17)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-6	Lanzar el proyecto de documentación	8h	8h	0%
DOCFF-17	Añadir una comprobación de la compilación en el script de construcción	11h	12h	**+9.09%**
DOCFF-18	Sincronizar el repositorio del Script	12h	11h	**-8.3%**
DOCFF-8	Detectar los principales errores	4h	4h	0%
TOTAL		**35 h**	**35 h**	0%

Tabla 17. Comparativa de los tiempos estimados y reales de las tareas del Sprint 5.

Como conclusión de la anterior tabla, se puede decir, que no se han producido grandes desviaciones de tiempo en las tareas desarrolladas, llegando incluso a coincidir el número total de horas estimadas con las invertidas.

2.6. Sprint 6

Las tareas que se han planificado para completar este Sprint son:

TAREA	DESCRIPCIÓN	ESTIMACIÓN
DOCFF-9	Añadir una nueva página a la documentación	14h
DOCFF-23	Mejorar el método de selección de proyectos	10h
DOCFF-24	Agregar entradas de menú contextual por proyecto en la vista de árbol del proyecto	6h
DOCFF-25	Sistema e integración de las pruebas	5h
TOTAL		**35h**

Tabla 18. Product backlog del Sprint 6.

2.6.1. Tareas

- **Tarea DOCFF-9 Añadir una nueva página a la documentación** (Completada)

  El requisito que hay que cumplir para completar esta tarea es el siguiente:

  Requisito imprescindible:

  - El script no tendrá que compilar el proyecto de documentación, si no se ha producido ninguna modificación.

  Las subtareas en que se ha divido esta tarea son las siguientes:

  - Especificación del caso de uso: Añadir una página a la documentación.
  - Diseño del prototipo del wizard.
  - Implementar el wizard.
  - Crear página y añadirla al fichero Schema.yml.

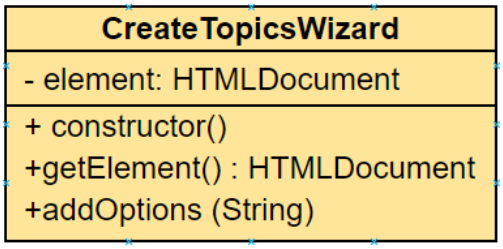
ANÁLISIS

Especificación del caso de uso: Añadir una página a la documentación.

Caso de uso	**Añadir una página a la documentación**
Descripción	El usuario añadirá una página al proyecto de documentación
Actor	Usuario
Precondición	El usuario tiene que tener abierto un proyecto de documentación, en caso contrario el botón esta deshabilitado.
Postcondición	Se crea un nuevo fichero <name>.adoc en el proyecto de documentación
Flujo básico	1- El usuario pulsa el botón *Create a topic* del menú. 2- El sistema muestra un panel al usuario, donde se pide el nombre del fichero, y la ubicación donde se va a guardar. 3- El usuario rellena los datos y pulsa el botón *Accept*. 4- El sistema comprueba que los campos no están vacíos. 5- Se crea un fichero (.adoc), en la ruta especificacada por el usuario, con dicho nombre. 6- Se añade la referencia a esta página en el fichero *Schema.yml* del proyecto de documentación seleccionado.
Flujo alternativo	
Anotaciones	

DISEÑO

Para completar esta tarea, se ha creado la clase ***CreateTopicsWizard***, que incluye el desarrollo del panel para añadir un Topic al proyecto de documentación.



Diseño del prototipo del wizard

En esta tarea, se diseña wizard para añadir una página al proyecto de documentación. El panel consta de 4 campos, uno de tipo Select donde el usuario pueda seleccionar una opción sin necesidad de escribir, y los otros 3 de tipo texto.

En el primer parámetro, se selecciona la ubicación donde se va a guardar el fichero creado. Entre las opciones, aparecen todos los repositorios (-doc) que se encuentran en el proyecto de documentación seleccionado.

El campo Slug/Name, corresponde con el nombre del fichero que se quiere crear. Los otros dos campos (Title y Title short), son opcionales y corresponden con el nombre que aparece en el menú principal de la documentación.

A continuación se adjunta una captura del diseño realizado.

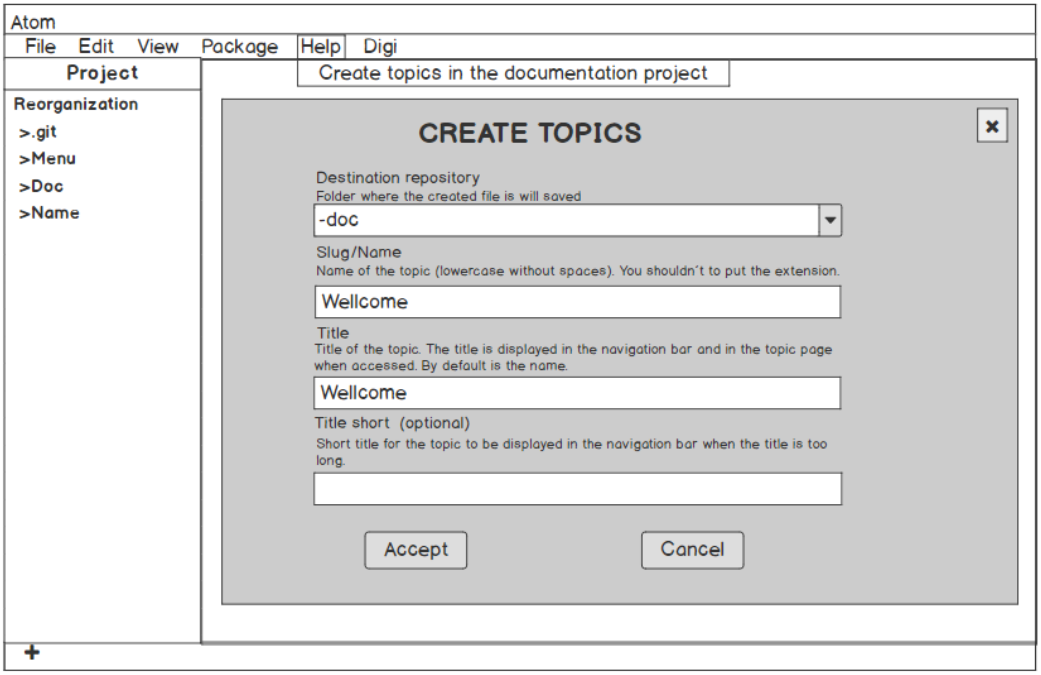


Figura 13. Diseño del prototipo del panel para crear un topic.

IMPLEMENTACIÓN

Implementación del panel

En esta tarea se implementa el código para crear el panel diseñado anteriormente. Su desarrollo se ha realizado en la clase *CreateTopicsWizard*.

Para añadir las opciones del campo Select, se ha desarrollado el método *addOptions(String),* que recorre todas las carpetas del proyecto de documentación y

busca si se corresponden con un repositorio (-doc). Los requisitos establecidos para esta búsqueda son los siguientes:

- El nombre del repositorio tiene que acabar en (–doc).
- Dicho repositorio tiene que contener dos carpetas (data y content).
- La carpeta data tiene que contener, como mínimo un fichero que acabe en la terminación (schema.yml).

Crear la página y añadirla al fichero Schema.yml

En esta tarea, en primer lugar, se crea un nuevo fichero vacío (.adoc), con el nombre introducido por el usuario y posteriormente se abre. Seguido en el fichero *Schema.yml* del proyecto de documentación seleccionado, se añade una referencia a dicho fichero.

En el fichero *Schema.yml*, es donde se guardan todas las referencias de todos los ficheros (.adoc) de la carpeta (–doc) donde nos encontramos. A través de estas referencias, los Scripts del backend, crean los ficheros HTML, que forman la documentación.

- **Tarea DOCFF-23: Mejorar el método de selección de proyectos** (Completada)

Hasta ahora, todas las opciones, solo habían supuesto el caso que solo había un proyecto de documentación abierto en Atom. Si el usuario tuviera dos proyectos abiertos, solo se trabajaba con el primero, para solucionar este problema y poder trabajar con ambos proyectos a la vez, surge esta tarea.

El objetivo es facilitar al usuario la manera de seleccionar las acciones que quiera aplicar al proyecto que haya elegido.

Inicialmente, si solo se abre un proyecto de documentación, todas las acciones deben aplicarse a ese proyecto. Si hay varios proyectos de documentación disponibles en el working-tree, para cada acción de la barra del menú principal debe aparecer una ventana emergente para seleccionar el proyecto de destino al que se aplicará la acción.

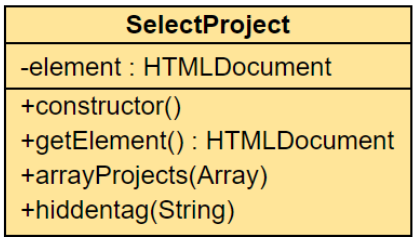
El requisito que se ha encontrado en esta tarea es:

Requisito imprescindible:

- El usuario podrá trabajar con varios proyectos de documentación abiertos a la vez en el entorno de Atom.

DISEÑO

Para desarrollar esta tarea, se ha creado la clase ***SelectProject***, que incluye la implementación del panel para seleccionar un proyecto de documentación.



IMPLEMENTACIÓN

Para el desarrollo de esta tarea, se ha implementado un panel, donde el usuario pueda seleccionar el proyecto de documentación sobre el que quiera trabajar. El panel simplemente cuenta con un campo Select, con los proyectos de documentación abiertos en el tree-view.

Este panel, solo se mostrará en caso de haber más de un proyecto abierto y se pulse algunos de los botones que requieran trabajar con un proyecto de documentación como son (Preferencias, Compilar un proyecto, Lanzar un proyecto, Añadir una página).

Para completar esta tarea, se ha tenido que modificar la forma de obtener el proyecto de documentación, en los eventos correspondientes a estos botones. Sin embargo, en el desarrollo del método para Añadir una nueva página, se producía un fallo de sincronización a la hora de mostrar los dos paneles. Para solucionarlo, se ha decidido modificar el wizard diseñado en la figura 13 (Panel para crear un topic), añadiendo un nuevo campo Select, que permita seleccionar el proyecto de documentación sobre el que trabajar. Este campo no se muestra si solo hay un proyecto de documentación abierto.

Debido a este problema, se ha producido una pequeña desviación con respecto al tiempo estimado.

A continuación se muestra una captura del nuevo panel que se ha desarrollado.

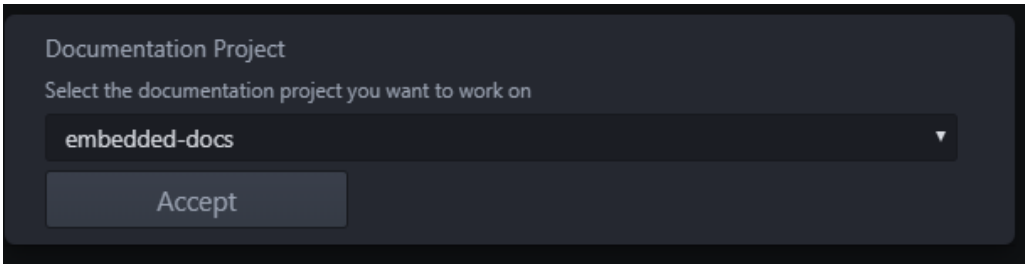


Figura 14. Panel para seleccionar un proyecto de documentación.

- **Tarea DOCFF-24: Agregar entradas de menú contextual por proyecto en la vista de árbol del proyecto** (Completada)

Esta tarea significa que las opciones de compilación, ejecución, preferencias y agregar topics deben estar disponibles al hacer click con el botón derecho en el tree-view. Este proceso omite la ventana emergente para seleccionar un proyecto, porque el proyecto de destino es aquel en el que se hace click con el botón derecho ocurrido.

El requisito que se ha encontrado en esta tarea, es el siguiente:

Requisito secundario:

- El usuario podrá seleccionar las opciones características de un proyecto de documentación desde el tree-view.

IMPLEMENTACIÓN

Agregar entradas de menú contextual, en la vista de árbol del proyecto

Esta tarea, consiste en añadir en la vista de árbol del proyecto, los 4 botones (Preferencias, Compilar un proyecto, Lanzar un proyecto, Añadir una página), con los que se puede trabajar en un proyecto de documentación. Se les ha asociado el mismo evento que a los botones del menú de Digi, del panel superior.

- **Tarea DOCFF-25: Sistema e integración de pruebas** (Completada)

Esta tarea consiste en realizar unas pruebas sistemáticas, donde se comprueba el funcionamiento de las principales tareas desarrolladas (Crear, importar, compilar un proyecto de documentación), así como la integración de estas, no afecta al comportamiento del producto.

ID	DOCFF 25
SPRINT	6
TAREA A LA QUE PERTENECE	Sistema e integración de pruebas
DESCRIPCION	Importar un proyecto de documentación desde Stash. Configurar las preferencias del proyecto. Compilar el proyecto de documentación Servir de forma local, el proyecto.
RESULTADO	**COMPLETADO** Si se proporciona una ruta de destino válida, el nombre y clave de un proyecto de documentación existente, se clona el proyecto. Se puede modificar los valores por defectos de las preferencias. Se permite compilar un proceso de compilación de forma local. Se puede servir localmente el proyecto de documentación y comprobar los cambios realizados en las settings del proyecto.

ID	DOCFF 25
SPRINT	6
TAREA A LA QUE PERTENECE	Sistema e integración de pruebas
DESCRIPCION	Crear un nuevo proyecto de documentación en Stash. Teniendo los dos proyectos abiertos en Atom a la vez: - Configurar las preferencias del proyecto. - Añadir una página/topic al proyecto de documentación - Compilar y servir localmente, el proyecto de documentación.
RESULTADO	**COMPLETADO** Si se proporciona una ruta de destino válida, y se tiene la clave del proyecto, acreditada por la empresa, se crea el proyecto de documentación. Con el anterior proyecto importado y el creado ahora, abiertos en Atom, comprobar que: Se puede modificar las preferencias por defecto del nuevo proyecto creado ahora. Se puede añadir una página al proyecto de documentación importado en la prueba anterior. Se pueden compilar y servir los dos proyectos de documentación de forma separa, comprobando los cambios realizados respectivamente.

Durante la realización de la primera prueba, se produjo un error, ya que tras importar bien el proyecto de documentación y añadirlo al working-tree, a la hora de configurar las preferencias, no se detectaba bien la ruta del proyecto. Tras depurar varias veces el código, se encontró el fallo, que se debía a la mala programación de un método if. Tras corregir el error, la prueba funcionó correctamente.

Este problema fue lo que ocasionó tener que invertir más horas de las estimadas.

2.6.2. Pruebas

ID	DOCFF-9
SPRINT	6
TAREA A LA QUE PERTENECE	Añadir una nueva página a la documentación
DESCRIPCION	Pulsar el botón *Create a topic* del menú de Digi, y rellenar los datos. Comprobar cómo se abre un fichero vacío nuevo con el formato (.adoc), guardado en la dirección indicada en el panel. Abrir el fichero *Schema.yml* y comprobar que se ha añadido una referencia del nuevo fichero que se ha creado.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF-23
SPRINT	6
TAREA A LA QUE PERTENECE	Mejorar el método de selección de proyectos
DESCRIPCION	Abrir dos proyectos de documentación distintos en Atom, y pulsar cualquier botón del menú Digi, que requiera tener un proyecto abierto. (Por ejemplo el botón *Compile*). Comprobar cómo se muestra un panel, con un campo Select, que nos permite seleccionar un proyecto de documentación y a continuación se ejecuta el proceso de compilación del proyecto.
RESULTADO	**COMPLETADO**
COMENTARIOS	

ID	DOCFF-24
SPRINT	6
TAREA A LA QUE PERTENECE	Agregar entradas de menú contextual, en la vista árbol del proyecto
DESCRIPCION	Pulsar el botón derecho, sobre un proyecto de documentación. Comprobar que se muestran los 4 botones del menú Digi, del panel superior y que al pulsarlos, tienen el comportamiento esperado.
RESULTADO	**COMPLETADO**
COMENTARIOS	

2.6.3. Revisión del Sprint.

En la siguiente tabla se muestran las tareas realizadas, junto al tiempo estimado, tiempo invertido y su posible desviación. (Tabla 19)

TAREA	DESCRIPCIÓN	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
DOCFF-9	Añadir una nueva página a la documentación	14h	14h	0%
	Fase de Análisis	1h	1h	0%
	Fase de Diseño	1h	1h	0%
	Implementación del wizard	6h	7h	+16.66%
	Crear la página y añadirla al fichero Schema.yml	6h	5h	-16.66%
DOCFF-23	Mejorar el método de selección de proyectos	10h	14h	+40%
DOCFF-24	Agregar entradas de menú contextual por proyecto en la vista de árbol del proyecto	6h	6h	0%
DOCFF-25	Sistema e integración de las pruebas	5h	6h	+25%
TOTAL		**35 h**	**40 h**	**+14.28%**

Tabla 19. Comparativa de los tiempos estimados y reales de las tareas del Sprint 6.

En este Sprint, al ser el último, se ha querido completar todas las tareas que se han programado desde un principio. Este ha sido el motivo, por el que se han invertido más horas de las estipuladas en la planificación.

La desviación producida (5 horas), corresponde en gran parte a la tarea DOCFF23, el motivo de esto, se debe al problema que he tenido al modificar el evento correspondiente al botón *'Preferences'* del menú Digi, del panel superior. En él, en vez de mostrar la página de preferencias, se mostraba una página en blanco. Para solucionarlo, en el evento del botón *Accept*, del nuevo panel, se ha creado una nueva promesa, para abrir la página de preferencias.

2.7. Seguimiento

En este apartado se va hacer una revisión de todas las tareas planificadas en el apartado 1.5 así como las diferencias entre los períodos de realización.

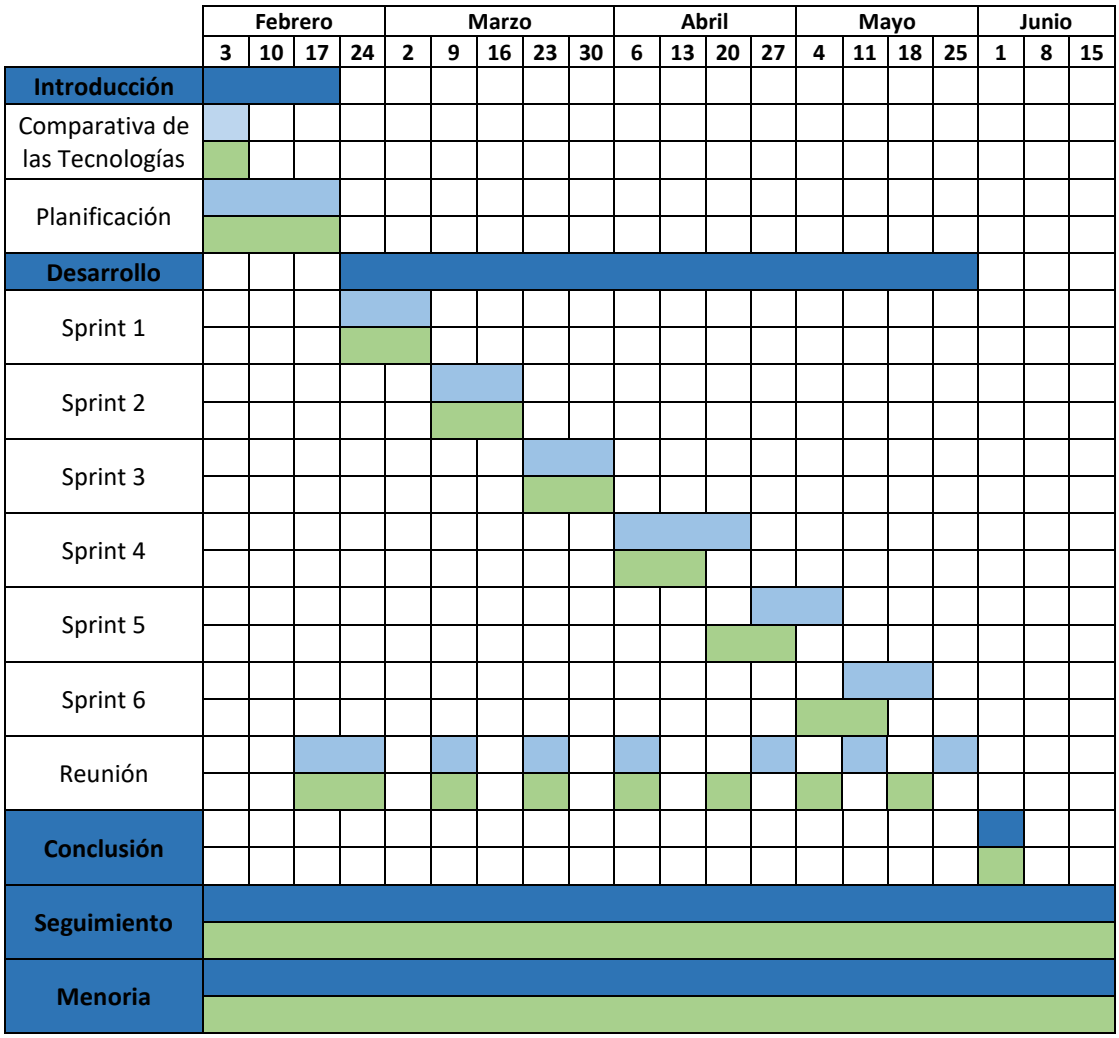


Tabla 210. Comparativa de los periodos estimados (azul) y reales (verde) de cada tarea.

TAREA	TIEMPO ESTIMADO	TIEMPO INVERTIDO	DESVIACIÓN
Introducción	**20h**	**20h**	**0%**
Planificación	18h	18h	0%
Comparativa de las Tecnologías	2h	2h	0%
Desarrollo	**235h**	**250h**	**+6.38%**
Sprints	215h	230h	+6.98%
Reuniones	20h	20h	0%
Conclusión	**5h**	**5h**	0%
Seguimiento	**5h**	**5h**	0%
Memoria	**35h**	**45h**	**+28.57**
TOTAL	**300h**	**325h**	**+8.3%**

Tabla 201. Comparativa de los tiempos estimados y reales de las tareas.

El Sprint 4, en vez de tener una duración de 3 semanas como estaba previsto en la planificación, por las fiestas de Semana Santa, tiene una duración de 2 semanas. Esta decisión ha sido tomada en una reunión con los tutores de la empresa, como consecuencia de continuar con el proyecto a distancia, en vez de en la empresa. Esto ha afectado a los demás Sprints, al comenzar una semana antes de lo previsto.

Con respecto a los tiempos invertidos, se puede ver que se ha creado una desviación en la elaboración de esta memoria. Pero la mayor desviación que se ha producido se corresponde con la tarea de Desarrollo. Aunque no viene indicado en la (Tabla 21), una pequeña parte de la desviación corresponde con el Sprint 6, que tiene una duración superior a la esperada (5 horas), por otro lado, las otras 10 horas restantes, se corresponden con las pruebas realizadas a las tareas que se han desarrollado en este proyecto.

Por último, en la (Figura 14), se refleja una comparativa entre las tareas creadas en JIRA y el momento en el que se han completado. Los puntos de color naranja, son las tareas que se han ido creando y los puntos de color morado, las tareas que se han resuelto. Se ve que la línea de color morado, no llega a coincidir con la de color naranja, eso se debe a que no se han acabado todas las tareas del Product Backlog (Tabla 7), como son la tarea (DOCFF-10 y DOCFF-11).

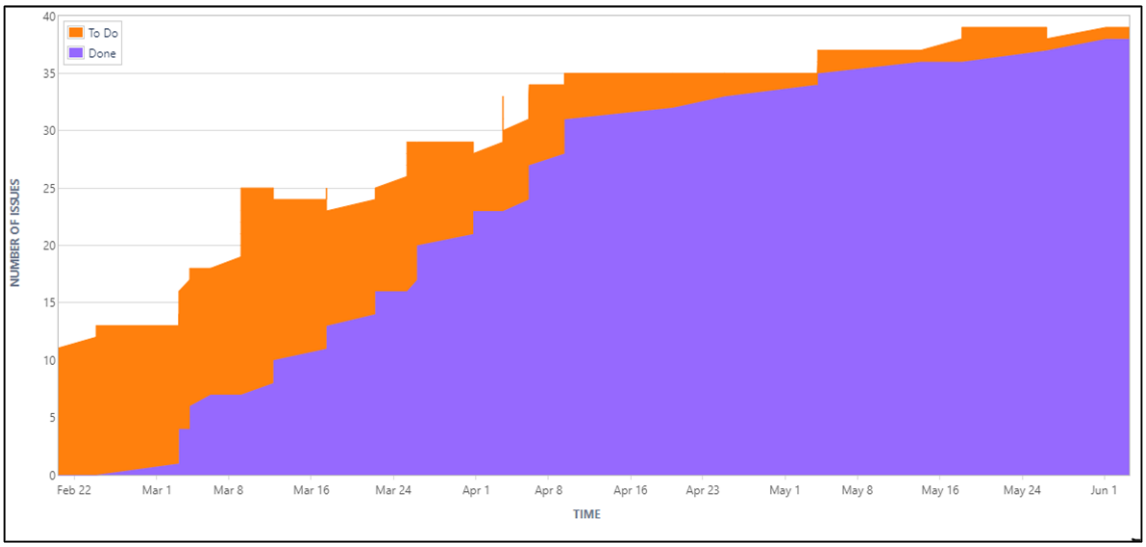


Figura 15. Seguimiento de las tareas durante el proyecto.

3. CONCLUSION

En este apartado se exponen las conclusiones obtenidas de este proyecto, así como, los conocimientos adquiridos, la evaluación de objetivos y las futuras mejoras del proyecto.

3.1. Evaluación de objetivos

En el apartado 1.2, se han especificado una serie de objetivos. Tras finalizar el proyecto, hay que comprobar el cumplimiento de estos, para verificar que se ha completado de forma exitosa.

Respecto a los principales objetivos, se ha podido comprobar, que el sistema permite a los desarrolladores de Digi, Crear, Importar y Compilar un proyecto de documentación, sin necesidad de tener cocimientos del backend. Esto significa, que estos objetivos se han cumplido correctamente.

Haciendo referencia a los objetivos generales, se han cumplido con éxito ya que en el desarrollo del paquete, se despliega una consola de log, donde de se muestran los resultados por pantalla. En cuanto a las funciones que permite el plugin, se han desarrollado paneles que simplifican el trabajo de los usuarios, al igual que una página que permite personalizar las preferencias de un proyecto de documentación.

Comentar que esta aplicación ya ha empezado a ser utilizada por los desarrolladores de esta oficina, en los nuevos proyectos que han empezado a desarrollar. Cuando se terminen de actualizar la documentación de los proyectos en los que están trabajando actualmente, se empezará a utilizar por los desarrolladores de la empresa.

3.2. Conocimientos adquiridos

El haber tenido que realizar un proyecto real en una empresa, y más en una internacional como Digi, me ha servido como una experiencia enriquecedora y positiva, ya que te encuentras de primera mano problemas que frecuentan en proyectos reales. Haciendo referencia a las lecciones aprendidas, he profundizado en conocimientos de lenguajes de programación como Python y JavaScript, también en herramientas de trabajo como (Repo, Ruby, Git).

En cuanto a las competencias transversales, se han adquirido importantes habilidades a la hora de trabajar en una empresa, como puede ser la toma de decisiones de forma autónoma, participar en reuniones, intentar resolver los problemas sin ayuda de un supervisor y el trabajo en equipo en los momentos que he colaborado, con alguna persona de la empresa.

Por último, destacar que esta empresa como multinacional que es, requiere el manejo del inglés en ciertos momentos. Aunque para este proyecto de forma oral no ha hecho falta, de forma escrita sí, para realizar toda la documentación del proyecto. Además todos los nombres para los métodos y variables, se han tenido que fijar en inglés. Por todo esto, en cierto modo, se ha adquirido soltura en el manejo de este idioma.

3.3. Futuras mejoras

Hasta este punto, como se ha dicho anteriormente, los objetivos del proyecto ya se han cumplido, pero esto no indica que la aplicación quede cerrada por completo, sino que se propone una serie de mejoras para un futuro.

- Modificar el backend, para que el usuario conozca en todo momento el estado en el que se encuentra el proceso de creación/compilación de un proyecto de documentación. Para ello, el Script debería devolver un porcentaje que indique el tiempo aproximado que le queda al proceso para acabar.
- Añadir tres botones que permitan al usuario crear, eliminar y renombrar una nueva unidad en el proyecto de documentación.
- Cambiar la configuración del backend, para actualizar en todo momento el repositorio común a todos los proyectos de documentación.

4. BIBLIOGRAFÍA

- Atom: Página principal del editor de texto. Documentación y código de otros paquetes.
  https://atom.io/
- Jira: Herramienta de la empresa para administrar y organizar el proyecto.
  https://jira.digi.com/
- Balsamiq: Herramienta usada para el diseño de los prototipos de las interfaces.
  https://balsamiq.com/
- Visual paradigm: Herramienta donde se han dibujado los diagrama de Actividad, Clase y Casos de uso.
  https://online.visual-paradigm.com/
- Stack overflow: Página web de preguntas y respuestas para programadores, usada para resolver las dudas.
  https://stackoverflow.com/
- Javascript: Tutorial de javascript.
  https://developer.mozilla.org/es/docs/Web/JavaScript