Valencia (España), 1 de Noviembre de 2023

Al Profesor de la Asignatura

Proyecto de Ingeniería de Software

Grado en Ingeniería Informática

Universidad Internacional de Valencia

Dr. Horacio Kuna

De mi mayor consideración:

Me dirijo a Ud. a efectos de presentar la propuesta para el desarrollo de un producto software como tema para la cátedra “Proyecto de Ingeniería de Software” correspondiente al Grado en Ingeniería Informática de la Universidad Internacional de Valencia.

El producto software a desarrollar se denomina: **Herramienta de extracción automática de información de documentos PDF**

Sin otro particular, y quedando a la espera de la evaluación de la propuesta, me despido atte.

_____

Daniel González Cerviño

Índice de contenidos

Introducción y objetivos

Introducción

Se pretende desarrollar una herramienta de software bajo el nombre "Herramienta de Extracción Automática de Información de Documentos PDF".

Esta herramienta tiene el propósito de convertir documentos en formato PDF en respuestas estructuradas, extrayendo información relevante del contenido.

El objetivo principal de este proyecto es desarrollar una arquitectura de software que permite fácilmente ser extendida en el futuro.

Para ello, este sistema contendrá a su vez otros subsistemas diseñados para ser fácilmente extensibles, lo que permitirá registrar nuevos componentes y mejorar la funcionalidad según las necesidades.

En cuanto a la metodología, se seguirán principios avanzados de ingeniería del software, como una arquitectura limpia, desarrollo dirigido por pruebas (TDD), y la contenerización de servicios mediante tecnologías como Docker. Además, se aplicarán metodologías ágiles como Extreme Programming (XP) y Kanban para una gestión eficiente.

Antecedentes

Esta propuesta surge de la necesidad en mi trabajo actual en un fondo de inversión https://strongholdam.com/, donde se requiere analizar documentación de manera manual.

Un sistema de estas características permitirá automatizar una gran parte de este trabajo, permitiendo a los empleados centrarse en realizar aquellas tareas para las que realmente aportan valor.

Además, considero presentar este proyecto, o una versión similar, como mi Trabajo de Fin de Grado (TFG).

Objetivos

El objetivo fundamental de este proyecto es la creación de una sólida base de software que permita la extracción automática de información a partir de documentos PDF.

Esta base debe ser diseñada de manera que sea altamente extensible, lo que facilitará la incorporación de nuevos componentes en el futuro.

Ejemplos de casos de uso

A continuación veremos algunos ejemplos de casos de uso que podrían darse.

Caso de uso 1: Contrato de compraventa

Un usuario sube un documento PDF con un contrato de compraventa, el sistema lo categoriza como contrato de compraventa y recupera información la información más importante: las partes, vendedor y comprador, nombres, apellidos y documentos de identificación, el bien que está siendo vendido y el precio. Luego la estructura para poder ser utilizada por otros sistemas.

Veamos un ejemplo de una respuesta:

- Categoría: contrato de compra venta
- Partes
    - Vendedor
        - María González, 12345678A
        - Juan Pérez, 87654321B
    - Comprador
        - Laura Martínez, 54321678C
- Objeto: Patinete
- Precio: 1234,56
- Fecha: 28/10/2023

Caso de uso 2: Contrato de trabajo

Un usuario sube un documento PDF con un contrato de trabajo, el sistema lo categoriza como contrato de trabajo y recupera información la información más importante: las partes, empleador y empleado, nombres, apellidos y documentos de identificación, puesto de trabajo, salario anual, y fecha de inicio. Luego la

estructura para poder ser utilizada por otros sistemas.

Veamos un ejemplo de una respuesta:

- Categoría: contrato de trabajado
- Partes
    - Empleador
        - FashionFusion Boutique, Q6789012R
    - Comprador
        - Ana López, 65432187E
- Puesto: Analista de Datos
- Salario: 30.000,00
- Fecha: 28/10/2023

Descargo de responsabilidad

A pesar de que esta propuesta no cumple con todos los requisitos formales de la actividad, considero que aborda aspectos avanzados de ingeniería de software y proporcionará una valiosa experiencia en el desarrollo de proyectos de código abierto.

Alcance y limitaciones

El alcance de este proyecto se limita a la creación de una sólida base de software con la capacidad de extenderse por futuros implementadores con facilidad para la extracción automática de información a partir de documentos PDF.

En esta etapa inicial, se diseñará una versión sencilla de cada uno de los subsistemas.

Estas versiones, aunque funcionales, serán insuficientes para su utilización en un entorno real.

Especificación de módulos

Funcionales

1. Módulo generador de texto
2. Módulo de lectura automática de texto
3. Módulo de la interfaz

No funcionales

1. Contenerización
2. Sistema de logs / auditoría

Descripción de los módulos

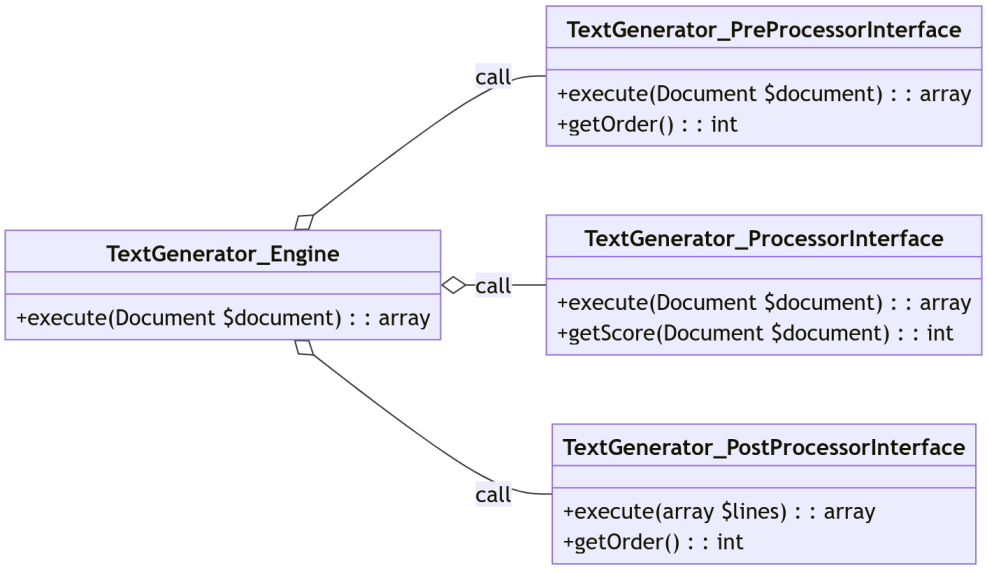
Funcionales

Generador de texto

Este componente tiene como finalidad recibir documentos en formato PDF y transformarlos en documentos de texto. Dado que existen diversas casuísticas en este proceso, se ha diseñado para ser altamente extensible, lo que permite registrar nuevas piezas de software en él mediante configuración.

Arquitectura

La arquitectura principal de este componente se ilustra en la siguiente figura.



Fase de pre procesado

Durante esta etapa, se invocarán todos los "preprocesadores" registrados en un orden específico, y la salida de cada uno alimentará al siguiente.

Fase de procesado

En esta fase, se ejecutará un único procesador de entre los registrados. El sistema llamará a cada uno de los procesadores y les proporcionará el documento. Cada procesador devolverá una puntuación que oscila entre 0 y 100, indicando la idoneidad de su uso. La siguiente tabla ilustra cómo se podría seleccionar el procesador más adecuado según la puntuación

puntuación	**Documento escaneado desde un documento físico**	**Documento generado desde microsoft word**
Procesador basado en un Tesseract	50	50
Procesador basado en pdftotext	0	100

Tesseract es una utilidad que convierte imágenes a texto, mientras que pdftotext extrae texto de un documento binario. Cada uno será adecuado o no según el tipo de documento.

Es importante destacar que se pueden agregar tantos procesadores como sea necesario.

Procesador basado en tesseract

Se va a implementar un procesador de texto basado en tesseract OCR https://es.wikipedia.org/wiki/Tesseract_OCR. una herramienta de línea de comandos que permite obtener el texto de una imagen.

Este procesador está indicado para convertir en texto documentos físicos escaneados.

Procesador basado en pdf2text

Se va a implementar un procesador de pdf a texto basado en pdf2text https://en.wikipedia.org/wiki/Pdftotext una herramienta de línea de comandos que permite obtener el texto plano de un documento pdf.

Este procesador está indicado para convertir en texto documentos generados

digitalmente.

Fase de post procesado

Finalmente, en esta etapa, se invocarán todos los "post procesadores" registrados en un orden específico, y la salida de cada uno alimentará al siguiente.

Ejemplos de post procesado pueden incluir la corrección de errores de codificación y la eliminación de caracteres HTML, entre otros.

Entradas y salidas

La entrada de este módulo es la ruta de un documento PDF que ya se encuentra dentro del sistema de archivos.

Si estuviéramos utilizando la interfaz de línea de comandos, la llamada a este sistema podría ser así:

```
$ php bin/console generate:text /tmp/my_temporal_document.pdf > output.txt
```

La salida de este comando sería el contenido en texto plano del texto extraido, si en nuestro ejemplo se tratara de un contrato de compraventa, podría ser la siguiente.

```
$ cat output.txt

CONTRATO DE TRABAJO Entre: Empleador: Nombre de la Empresa:
[Nombredelaempresa] Domicilio de la Empresa: [Direcciónde laempresa] Número de
Identificación Fiscal de la Empresa: [Númerodeidentificaciónfiscal] (en adelante, el
"Empleador") Empleado : Nombre del Empleado: [Nombrecompletodelempleado]
Domicilio del Empleado:
[...]
```

Registros de log

Este módulo registrará los logs de las entradas y salidas de cada subcomponente.

Los registros tendrán formato JSON, con la siguiente estructura

Campo	Valor

date_time	fecha y hora del registro.
owner	componente del sistema que ha generado el registro.
message	mensaje principal del registro.
input	entrada que recibió el sub sistema.
output	salida que generó el sub sistema.
level	nivel del registro según se determina en el RFC rfc2119 https://datatracker.ietf.org/doc/html/rfc2119.

A continuación un ejemplo de un conjunto de registros generado por este sistema.

```
$ cat log/dev.log

{"date_time" : "Wed, 01 Nov 2023 11:01:25 +0100", "owner": "main", "message": "process initialized", "input": "/tmp/my_temporal_document.pdf", "level": "debug"}

{"date_time" : "Wed, 01 Nov 2023 11:01:25 +0100", "owner": "pre proccessor opacity", "message": "pre processor executed", "input": "/tmp/my_temporal_document.pdf", "output":"/tmp/my_temporal_document.pdf_0001", "level": "debug"}

{"date_time" : "Wed, 01 Nov 2023 11:01:26 +0100", "owner": "proccessor pdf2 text", "message": "processor executed", "input": "/tmp/my_temporal_document.pdf_0001", "output": "<h1>CONTRATO DE TRABAJO</h1> <p>Entre:</p><ul>  <li>Empleador: Nombre de la Empresa: [Nombredelaempresa]</li> <li>Domicilio de la Empresa: [Direcciónde laempresa]</li><li> Número de Identificación Fiscal de la Empresa: [Númerodeidentificaciónfiscal]</li> </ul><p>(en adelante, el "Empleador")</p> <ul><li>Empleado : Nombre del Empleado: [Nombrecompletodelempleado] </li><li>Domicilio del Empleado:[...]</il></ul>", "level": "debug"}

{"date_time" : "Wed, 01 Nov 2023 11:01:27 +0100", "owner": "post processor remove html", "message": "post processor executed", "input": "<h1>CONTRATO DE TRABAJO</h1> <p>Entre:</p><ul>  <li>Empleador: Nombre de la Empresa: [Nombredelaempresa]</li> <li>Domicilio de la Empresa: [Direcciónde laempresa]</li><li> Número de Identificación Fiscal de la Empresa: [Númerodeidentificaciónfiscal]</li> </ul><p>(en adelante, el "Empleador")</p> <ul><li>Empleado : Nombre del Empleado: [Nombrecompletodelempleado] </li><li>Domicilio del Empleado:[...]</il></ul>", "output": "CONTRATO DE TRABAJO Entre: Empleador: Nombre de la Empresa: [Nombredelaempresa] Domicilio de la Empresa: [Direcciónde laempresa] Número de Identificación Fiscal de la Empresa: [Númerodeidentificaciónfiscal] (en adelante, el "Empleador") Empleado : Nombre del Empleado: [Nombrecompletodelempleado] Domicilio del Empleado:[...]", "level": "debug"}
```

Los registros se almacenarán inicialmente en un fichero de texto, pero este

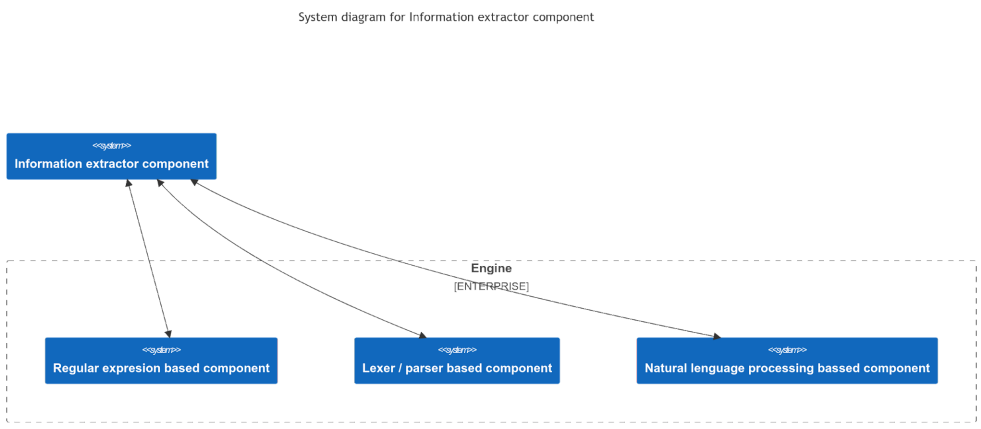
formato está listo para ser almacenado en un sistema que permita el análisis avanzado de registros de logs, como puede ser elastich search https://www.elastic.co/

Lectura de texto

El núcleo esencial de este proyecto reside en el módulo de lectura automática de texto. Este componente debe tener la capacidad de comprender el contenido de un documento, categorizar según los diversos modelos definidos y extraer la información en una estructura previamente establecida.

Los detalles específicos del diseño y las características de este módulo se precisarán en mayor profundidad durante su desarrollo.

El sistema está pensado para que diferentes tecnologías puedan ser utilizadas, a continuación algunas herramientas que podrían ser utilizadas según el tipo de documento.



Documentos con una estructura fija conocida

Se pueden utilizar expresiones regulares para la extracción de información cuando se conoce detalladamente la estructura del documento.

Por ejemplo, al analizar un modelo de entidad con un formato constante, podríamos aplicar expresiones regulares para identificar el nombre del vendedor entre las palabras "Nombre y apellidos" y "Documento de identificación".

Documentos con una estructura conocida

Si bien la estructura puede variar, pero se tiene un cierto conocimiento de cómo se presenta la información, una opción efectiva es utilizar tecnologías de análisis sintáctico de texto, como Spacy (https://spacy.io/).

Mediante esta tecnología, podríamos buscar la frase que contiene la expresión "en adelante el vendedor" y extraer el sujeto de dicha frase.

Documentos con una estructura desconocida

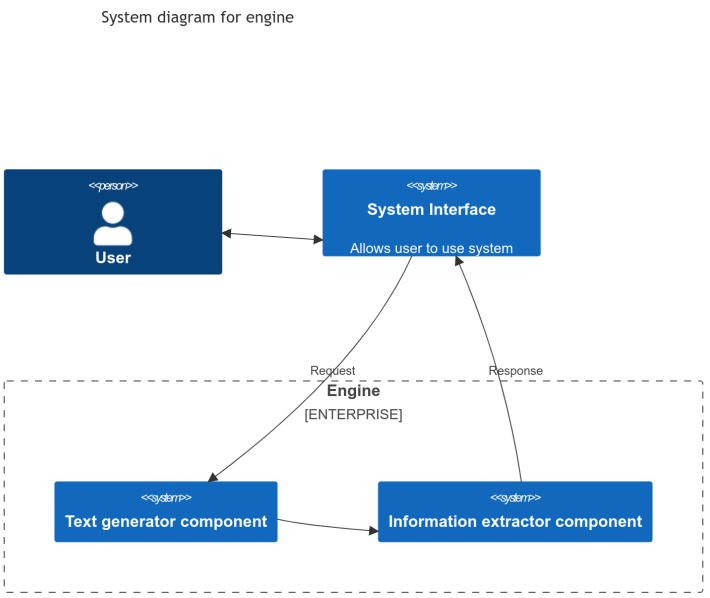
Para casos en los que la estructura puede variar significativamente o incluso ser desconocida, se requerirá un modelo de lenguaje natural tipo transformers (https://github.com/huggingface/transformers).

Con esta tecnología, simplemente proporcionamos el texto completo y luego formulamos preguntas directas, como "¿Cómo se llama el vendedor?", para obtener la información deseada.

Este enfoque es particularmente útil cuando se enfrenta a documentos con estructuras variables y complejas.

Interfaz

El sistema debe estar diseñado para poder ser integrado en un sistema más complejo mediante una interfaz.



Algunos ejemplos de interfaces pueden ser:

- Interfaz de línea de comandos
- Sitio web
- API

Para este proyecto proponemos desarrollar una interfaz de línea de comandos, por ser sencilla de programar, y utilizar mediante scripts de línea de comandos.

Interfaz de línea de comandos

Una interfaz de línea de comandos es una interfaz muy sencilla que puede ser implementada muy fácilmente, de esta forma se pone el foco en el desarrollo de la tecnología y no en el desarrollo de la interfaz de la tecnología.

Además una interfaz de línea de comandos, permite ser combinada fácilmente con otros programas de línea de comandos en linux, veamos algunos ejemplos.

Guardar la salida en un documento de texto

```
$ php bin/console generate:text /tmp/my_temporal_document.pdf > output.txt
```

Procesar varias imágenes, que han sido escaneadas por separado.

```
$ convert folder/*.png /tmp/my_temporal_document.pdf && php bin/console generate:text /tmp/my_temporal_document.pdf > output.txt
```

Buscar una palabra clave en la salida.

```
$ php bin/console generate:text | grep keyword
```

Todas estas tareas y otras más complejas pueden ser automatizadas mediante scripts de línea de comandos, o mediante utilidades como make https://www.gnu.org/software/make/manual/make.html, muy populares en proyectos open source.

Gestión de usuarios

En una interfaz de línea de comandos no se realiza ningún tipo de gestión de usuarios.

El objetivo principal de esta tecnología es convertir documentos PDF en información estructurada, realizar una gestión de usuarios supone perder de vista el verdadero objeto de este proyecto. Por lo tanto queda fuera del alcance de este proyecto.

Interfaz web sencilla

Para optimizar la experiencia del usuario, se planifica la creación de una interfaz web intuitiva y fácil de usar, tomando inspiración de la plataforma ilovepdf (https://www.ilovepdf.com/).

En el diseño de esta interfaz web, se ha decidido prescindir de la gestión de usuarios para simplificar el proceso. En lugar de ello, se implementará una función básica que permitirá a los usuarios simplemente arrastrar documentos a una casilla designada. Esta acción activará el sistema de análisis, que examinará el contenido del documento y ofrecerá una respuesta inmediata.

Esta decisión de diseño se basa en la premisa de poner un énfasis máximo en el desarrollo de la tecnología en sí misma, en lugar de orientarse hacia la creación

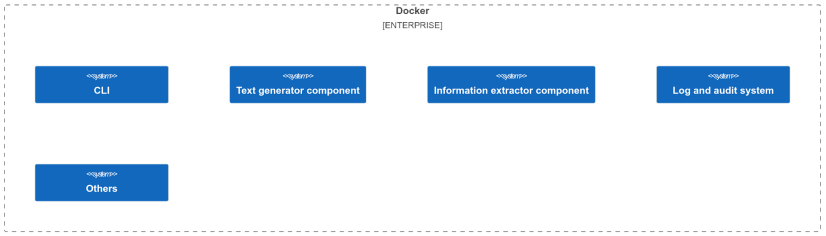
de un sistema que sea capaz de utilizar la tecnología que estamos proponiendo.

No funcionales

Contenerización

Los contenedores Docker se han convertido en una tecnología fundamental en el mundo del desarrollo de aplicaciones debido a su capacidad para encapsular aplicaciones y sus dependencias en entornos aislados y portátiles.

El sistema emplea esta tecnología para orquestar todos los servicios necesarios.



Sistema de logs / auditoría

Utilizar un sistema de logs proporciona una visión detallada de la actividad de la aplicación, lo que resulta crucial para la detección y resolución de problemas, tanto durante el desarrollo como en la producción.

Estos registros permiten rastrear eventos, identificar errores, y entender el comportamiento de la aplicación, lo que agiliza la depuración y mejora la calidad del software.

Cobertura de test

Todos los módulos deben tener una amplia cobertura de pruebas automáticas.

Entorno tecnológico

Arquitectura y diseño

Arquitectura limpia

Se va a implementar una arquitectura limpia tal y como aparecen descritas en Martin, R. C. (2017). Clean Architecture: A Craftsman 's Guide to Software Structure and Design. Prentice Hall, junto con las directrices que aparecen mencionadas en Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

Una arquitectura limpia es un enfoque de diseño de software que promueve la organización y separación de componentes para facilitar el mantenimiento y escalabilidad.

Ofrece ventajas como una estructura modular, facilidad para realizar pruebas unitarias, y adaptabilidad a cambios sin afectar otras partes del sistema.

Desarrollo dirigido por pruebas

Como se describe en Beck, K. (2002). Test Driven Development: By Example. Addison-Wesley.

El Desarrollo dirigido por Pruebas (TDD) es una metodología de desarrollo de software que implica escribir pruebas automatizadas antes de escribir el código de la funcionalidad.

Ofrece ventajas como la detección temprana de errores, mayor calidad del código y una mejor comprensión de los requisitos del software.

Contenerización

Para la implementación del grupo de contenedores se siguen las indicaciones de Nickoloff, J. (2016). Docker in Action. Manning Publications.

La contenerización es una tecnología que permite empaquetar aplicaciones y sus dependencias en entornos aislados y portátiles.

Ofrece ventajas como la portabilidad, escalabilidad, y la gestión eficiente de recursos para implementar aplicaciones de manera consistente en diferentes entornos..

Lenguajes de programación y frameworks

Como lenguaje de programación vamos a utilizar PHP. PHP es un lenguaje de programación ampliamente utilizado para el desarrollo web de código abierto y con una gran comunidad.

Como framework de desarrollo vamos a utilizar Symfony. Symfony es un framework muy popular en el desarrollo de aplicaciones web en PHP. Ofrece una estructura organizada, componentes reutilizables y una comunidad muy activa.

Motor de base de datos

El motor de base de datos apropiado no se ha identificado todavía. Puede que se vaya a usar uno o varios.

En una arquitectura limpia, esto se considera un 'detalle de implementación', y el sistema debe ser adaptable para permitir cambios de uno a otro con facilidad.

Entorno metodológico

Se ha optado por un stack metodológico ligero. En equipos pequeños o con un solo desarrollador, se prefieren menos procesos formales para enfocarse en la entrega de valor a corto plazo en lugar de seguir un plan rigurosamente definido.

Extreme programing

Cómo se define en Agile Manifesto. (2001). Manifesto for Agile Software Development. http://agilemanifesto.org/ y en la forma en la que se aplica al desarrollo de software en Hunt, A., & Thomas, D. (2019). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley.

Elegimos esta metodología, por su simplicidad, la retroalimentación constante y por poner por encima la entrega de software que funciona sobre la ejecución de

un plan establecido.

Kanban

Cómo se define en Anderson, D. J. (2010). Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press.

Kanban es un sistema visual de gestión y organización que se utiliza para mejorar la eficiencia y el flujo de trabajo.

Ofrece ventajas como la visibilidad en tiempo real de tareas, reducción de cuellos de botella y la capacidad de adaptarse a cambios de manera ágil.

Para un proyecto como este, con una única persona, utilizaremos las siguientes columnas:

- Backlog: las tareas en esta columna no están necesariamente completamente definidas.
- TODO: las tareas en esta columna están claramente definidas y próximas a hacer. Podría ser asimilable a un sprint backlog. Se establece un límite máximo de 5 tareas en esta columna.
- WIP: las tareas en esta columna son en las que se están trabajando en el momento actual. Se establece un límite máximo de 1 tarea en esta columna.
- DONE: las tareas en esta columna son las que han sido terminadas.

Sprints

Los sprints son un concepto clave en la metodología ágil de desarrollo de software, habitualmente usados en marcos como Scrum.

Se trata de un periodo de tiempo predefinido, generalmente de una a cuatro semanas, durante el cual un equipo de desarrollo se compromete a completar un conjunto específico de tareas o funcionalidades.

Por lo general, al inicio de cada sprint se definen las tareas a realizar, y al término del mismo se lleva a cabo una demostración y una revisión para evaluar el progreso del proyecto.

Tipo de proyecto

Este proyecto no tiene un cliente final y se utilizará en mi trabajo de fin de grado. El código se compartirá como un proyecto de código abierto, permitiendo que la comunidad lo copie, modifique y mantenga.

Módulos a programar

Se han definido las siguientes tareas principales de este proyecto, no se trata de un análisis detallado, si no un análisis superficial que permita definir los grandes bloques.

Cada una de estas actividades será dividida en subtareas una vez que comience su ejecución según sea necesario.

Cod.	Nombre y descripción	Horas	desarrolla?
001	Análisis y definición inicial de requisitos En esta actividad se realiza un análisis y una definición inicial del sistema completo, dejando paso al análisis detallado de cada componente para más adelante.	10.0	**SI**
002	Diseño de la arquitectura del sistema de generación de texto. En esta actividad se diseñarán todas las clases e interfaces que intervienen en este componente, así como sus entradas y salidas. Depende de 001	6.0	**SI**
003	Diseño de la arquitectura del sistema de lectura de texto. En esta actividad se diseñarán todas las clases e interfaces que intervienen en este componente, así como sus entradas y salidas. Depende de 001	8.0	**SI**
004	Diseño e implementación del sistema de pruebas automáticas.	4.0	**SI**

	En esta actividad se realizará la configuración de los procesos de pruebas automáticas. Depende de 001		
005	Diseño e implementación del sistema basado en contenedores. En esta actividad se desarrolla una estructura de contenedores y la comunicación entre ellos de forma que pueda ejecutarse el proyecto sin la necesidad de instalar todas las utilidades requeridas. Depende de 001	10.0	**NO**
006	Diseño e implementación del sistema de registro de logs. En esta actividad se desarrolla un sistema de registro de logs, que permita la depuración de errores, así como la obtención de información estadística de uso. Depende de 001	4.0	**NO**
007	Definición de un conjunto de datos de prueba. En esta actividad se desarrollará un conjunto de datos de pruebas con algún tipo de documento, generando manualmente la salida esperada. Depende de 001	20.0	**SI**
008	Implementación de un proceso automático de verificación En esta actividad se ejecutará el conjunto de datos de prueba sobre el sistema y se obtendrá como resultado un grado de precisión sobre el resultado esperado. Depende de 007	8.0	**SI**
009	Implementación de un procesador de texto basado en pdf2text. En esta actividad se desarrollará un procesador de texto basado en pdf2text. Depende de 002	12.0	**SI**

010	Implementación de un procesador de texto basado en tesseract. En esta actividad se desarrollará un procesador de texto basado en tesseract. Depende de 002	16.0	**NO**
011	Investigación sobre el estado del arte de las herramientas de lectura automática de texto. En esta actividad se realizará una investigación sobre las herramientas existentes en este campo, priorizando herramientas open source sobre herramientas con licencia privativa. Depende de 001.	40.0	**SI**
012	Implementación de un módulo de lectura automática de texto. Como resultado de la actividad anterior se realizará la implementación con al menos una de las herramientas investigadas. Depende de 010.	40.0	**SI**
013	Elaboración de la documentación requerida para la asignatura.	10.0	**SI**
014	Publicación del proyecto con una licencia Open Source Como resultado de esta actividad se realizará los pasos necesarios para ser publicados como un proyecto open source, selección de la licencia, elaboración de la documentación, manuales de instalación, ejecución, ejecución de las pruebas, guía de estilo, guía de cómo contribuir, etc.	16.0	**NO**
015	Desarrollo de una interfaz de línea de comandos Como resultado de esta actividad se podrá interactuar con el sistema mediante llamadas en la línea de comandos	16.0	**SI**
016	Desarrollo de una interfaz web sencilla Como resultado de esta actividad se podrá interactuar con el sistema mediante una página web, en la que se podrán arrastrar documentos y	24.0	**SI**

	ver el resultado del análisis.		
	Total	**244**	

Estrategia de validación

Definición del conjunto de datos de prueba

Se definirá un conjunto de datos de prueba. Para ello se determinará algún tipo de documentos, por ejemplo facturas.

Se elaborará manualmente una base de datos con algunas decenas o cientos de documentos y el resultado esperado de cada una de las facturas, por ejemplo:

- Documento: f9fa42b2cee441c3c06f5ffea8dd1e4ed11b1f58.pdf
- Categoría: factura
- Partes
    - Emisor
        - SwiftLogistics Group, C9876543D
    - Receptor
        - Carlos Rodríguez, 98765432D
- Fecha: 28/10/2023
- Importe: 30.000,00

Proceso de verificación sobre el conjunto de datos de prueba

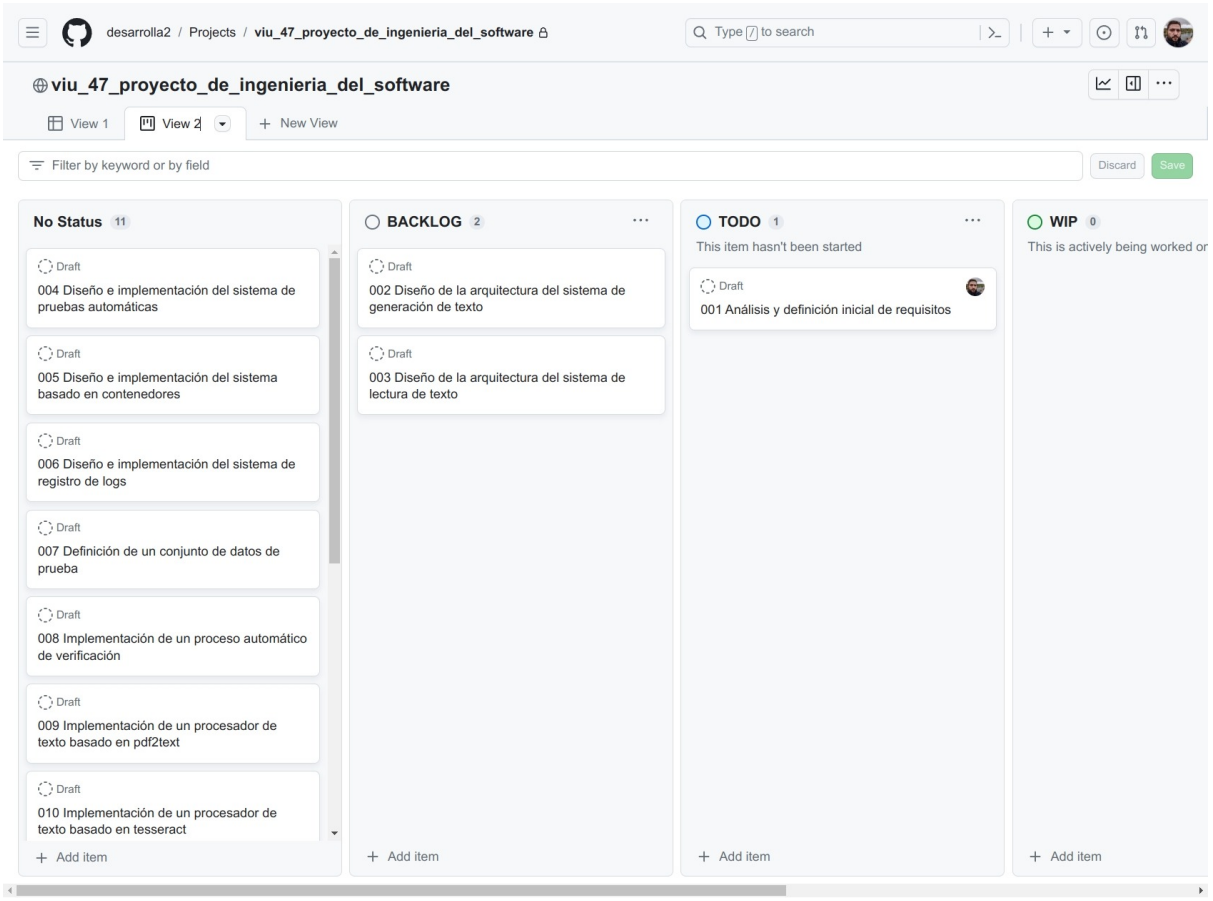
Se definirá un proceso que comparará el resultado del conjunto de datos de prueba realizado manualmente contra el conjunto de datos de prueba que produzca el sistema.

Como resultado de este proceso, se obtendrá un grado de precisión que permitirá evaluar cómo de preciso es el sistema al completo.

Planificación de actividades

A pesar de que se ha definido una metodología ágil donde prima la obtención de software funcionando, sobre el seguimiento de un plan minuciosamente definido, se han definido las siguientes actividades principales.

El seguimiento del proyecto se realizará a través de la herramienta de github projects en la siguiente URL https://github.com/users/desarrolla2/projects/1/views/2.



Sprints y entregables

Si bien en una metodología ágil debe primar la realización de un trabajo que aporte el máximo valor en cada momento, se ha diseñado el siguiente plan de trabajo que deberá ser reevaluado cada semana y adaptado a los avances realizados.

Al principio de cada sprint se realizará un análisis del trabajo a realizar, profundizando en la definición de cada una de las tareas.

Al final de cada sprint, se hará una reevaluación del calendario de trabajo en función de si se están cumpliendo los objetivos.

Sprint #0, Inicio 26/10/23, Final 09/11/23

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
001	Análisis y definición inicial de requisitos	10.0
002	Diseño de la arquitectura del sistema de generación de texto	6.0
	Total	**16.0**

Sprint #1, Inicio 10/11/23, Final 23/11/23

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
003	Diseño de la arquitectura del sistema de lectura de texto.	8.0
004	Diseño e implementación del sistema de pruebas automáticas.	4.0
007	Definición de un conjunto de datos de prueba.	10.0
	Total	**22.0**

Sprint #2, Inicio 24/11/23, Final 07/12/23

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
008	Implementación de un proceso automático de verificación.	10.0
009	Implementación de un procesador de texto basado en pdf2text.	12.0
	Total	**22.0**

Sprint #3, Inicio 08/12/23, Final 22/12/23

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
011	Investigación sobre el estado del arte de las herramientas de lectura automática de texto	40.0

	Total	**40.0**

Sprint #4, Inicio 22/12/23, Final 11/01/24

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
012	Implementación de un módulo de lectura automática de texto	40.0
	Total	**40.0**

Sprint #5, Inicio 12/01/24, Final 25/01/24

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
015	Desarrollo de una interfaz de línea de comandos	16.0
	Total	**16.0**

Sprint #6, Inicio 26/01/24, Final de la asignatura

Se han definido las siguientes actividades a realizar.

Cod.	Nombre y descripción	Horas
013	Elaboración de la documentación requerida para la asignatura	10.0
016	Desarrollo de una interfaz web sencilla	24.0
	Total	**34.0**

Dedicación

Con el fin de garantizar el progreso adecuado de este proyecto, se han planificado entregas cada dos semanas.

Se estima que el esfuerzo requerido oscila entre 16 y 40 horas por entrega, equivalente a un rango semanal de 8 a 20 horas.

Aunque esta carga de trabajo es significativa, vale la pena señalar que puedo

aprovechar todo este trabajo para mi proyecto de fin de grado, lo que lo convierte en una oportunidad valiosa para mi desarrollo académico y profesional.

Anexo I: Comentarios del profesor

Primera revisión

En este bloque repasamos los comentarios realizados por el profesor en la revisión del día 26/10/2023 .

Presentación en general

> Ok, correcto. Sería bueno contar con más información acerca de para qué se utilizarían los resultados de las operaciones a realizar con la aplicación a fin de determinar otros aspectos que pudiera ser relevante incorporar.

He ampliado la información en el apartado de introducción y objetivos indicando antecedentes de motivación del proyecto y algunos ejemplos de casos de uso, indicando las entradas y las salidas.

Requisitos funcionales

> Se deben detallar con más profundidad los procesos involucrados para tener una visión completa de qué se implementaría. También las salidas a generar y de qué manera serían presentadas al usuario final o almacenadas.

He ampliado la información que aparece en el apartado de descripción de los módulos, bloque de requisitos funcionales relacionado con el sistema que convierte PDF en texto.

Se ha detallado cada una de las fases que aparecen, así como se han indicado las entradas y salidas que se obtienen de este componente.

Me dejó pendiente desarrollar el bloque de requisitos relacionado con la lectura automática del texto para la última versión de este informe.

Requisitos no funcionales

> Igualmente, se requiere mayor detalle en algunos aspectos como la interfaz gráfica (no limitar a que va a ser por línea de comandos) y qué tipos de registros se estarían generando como logs.

He ampliado la información que aparece en el apartado de descripción de los

módulos, bloque de interfaz de línea de comandos. He indicado ejemplos de entrada y salida, así como la posibilidad de combinar el uso de este componente con otras herramientas existentes en un sistema linux.

He ampliado la información que aparece en el apartado de descripción de los módulos, bloque de requisitos funcionales relacionado con el sistema que convierte PDF en texto, indicando ejemplos del resultado de una ejecución, en el sistema de registro de logs.

> Se va a implementar un control de usuarios? Un histórico de procesamientos de archivos realizados por un usuario? Existiría algún tipo de jerarquía entre usuarios?

Este es un proyecto en el que se desarrolla una tecnología, quedando fuera de su alcance la gestión de usuarios.

Sin embargo he ampliado la información que aparece en el apartado de descripción de los módulos, bloque de interfaz de línea de comandos, indicando cómo podría implementarse esta gestión de usuarios para la interfaz de línea de comandos utilizando herramientas disponibles en un sistema operativo linux.

Estrategia de validación / verificación

> Completar este item.

He desarrollado el apartado de estrategia de validación indicando que se va a elaborar manualmente un conjunto de datos de prueba. También se va a implementar un proceso automático, que permite ejecutar todos los documentos del conjunto de datos de prueba y compararlos con el resultado esperado. De esta forma se obtiene un grado de precisión del sistema.

Procesos automáticos

> Se tienen que detallar, con las adaptaciones que correspondan.

Como comento en el apartado anterior he desarrollado el apartado de estrategia de validación indicando que se va a implementar un proceso automático, que permite ejecutar todos los documentos del conjunto de datos de prueba y compararlos con el resultado esperado.

Planificación

> Completar este ítem. Además de definir, una vez aplicados los cambios en los puntos anteriores, el nivel de detalle de la implementación a realizar para la cátedra.

He desarrollado el apartado de planificación de actividades indicando las actividades que han sido detectadas en este momento así como un diagrama de gantt, con un plan inicial de trabajo.

Metodología

> Si bien con Kanban no se establece un esquema de iteraciones, se tienen que definir hitos de revisión / control mediante los cuales se realizarán las presentaciones a la cátedra.

He desarrollado el apartado de planificación de actividades indicando las actividades que han sido detectadas en este momento así como un diagrama de gantt, con un plan inicial de trabajo.

Segunda revisión

En este bloque repasamos los comentarios realizados por el profesor en la revisión del día 04/11/2023 .

Requisitos funcionales

> Requisitos funcionales Resta tener la descripción de uno de los módulos "lectura automática de texto". Se ve que se destina carga horaria para su investigación e implementación, sin embargo no se tiene una descripción del mismo inicialmente. Si fuera a ser algo como text-to-speech se debería delimitar con qué recursos podría ser implementado.

He extendido la información presente en la sección de descripción de los módulos, específicamente en el bloque de requisitos funcionales relacionado con el sistema de lectura automática de texto.

Sin embargo, para proporcionar una descripción más detallada, se requerirá una investigación más exhaustiva sobre el estado actual de las tecnologías de lectura

automática de texto. Este análisis incluirá la identificación de tecnologías existentes, su disponibilidad para uso gratuito, entre otros aspectos relevantes.

Con el objetivo de abordar esta investigación de manera efectiva, hemos planificado el Sprint #3, que se llevará a cabo desde el 08/12/23 hasta el 22/12/23. Durante este período, asignaremos una carga horaria sustancial para llevar a cabo esta investigación de manera integral y detallada.

Después de esa investigación se definirán con precisión los recursos necesarios para su implementación.

> Las categorías de documentos, definen la estructura de la salida a generar según lo que se ve. ¿Estos datos dónde van a estar almacenados? Sería un módulo a agregar a modo de configuración de la generación que, al no quedar únicamente definido en el código, permitiría la extensión del producto a otros tipos de documentos.

He extendido la información presente en la sección de descripción de los módulos, específicamente en el bloque de requisitos funcionales relacionado con el sistema de lectura automática de texto.

Se pueden implementar diferentes subsistemas basados en diferentes tecnologías, cada uno de ellos, tendrá que definir cómo se configura para permitir su extensión.

> Se sugiere contar al menos básicamente con una interfaz gráfica tipo web o similar, aún cuando el producto posteriormente pueda ser integrado como parte de otro sistema que no la requiera. Como modelo podría ser una versión simplificada de ilovepdf.com, más cuando la salida a generar pareciera ser en HTML.

He extendido la información presente en la sección de descripción de los módulos, específicamente en el bloque de requisitos funcionales relacionado con el sistema de lectura automática de texto.