

Universidad de Murcia
Facultad de Informática

GRADO EN INGENIERÍA INFORMÁTICA

**Diseño e implementación de una herramienta
bioinformática para la gestión y procesamiento de datos
dentro de la terapia celular del trasplante de médula ósea**

Trabajo Fin de Grado realizado por:
Jose María Belmonte Martínez

Bajo la dirección de:
Dr. D. José Manuel García Carrasco

Julio, 2022

Declaración firmada sobre la originalidad del trabajo

D. Jose María Belmonte Martínez con DNI 48701180Z, estudiante de la titulación de **Grado en Ingeniería Informática** de la Universidad de Murcia y autor del TF titulado "Diseño e implementación de una herramienta bioinformática para la gestión y procesamiento de datos dentro de la terapia celular del trasplante de médula ósea".

De acuerdo con el Reglamento por el que se regulan los Trabajos Fin de Grado y de Fin de Máster en la Universidad de Murcia (aprobado C. de Gob. 30-04-2015, modificado 22-04-2016 y 28-09-2018), así como la normativa interna para la oferta, asignación, elaboración y defensa de los Trabajos Fin de Grado y Fin de Máster de las titulaciones impartidas en la Facultad de Informática de la Universidad de Murcia (aprobada en Junta de Facultad 27-11-2015)

DECLARO:

Que el Trabajo Fin de Grado presentado para su evaluación es original y de elaboración personal. Todas las fuentes utilizadas han sido debidamente citadas. Así mismo, declara que no incumple ningún contrato de confidencialidad, ni viola ningún derecho de propiedad intelectual e industrial

Murcia, a 4 de Julio de 2022

A handwritten signature in black ink, appearing to read 'J. Belmonte', with a long horizontal stroke extending to the right.

Fdo.: Jose María Belmonte Martínez.
Autor del TF

Índice

Índice de figuras.....	6
Resumen.....	7
Extended Abstract.....	8
1. Introducción.....	12
1.1. Motivación	13
2. Estado del arte.....	14
2.1. Trabajos relacionados.....	14
2.2. Desarrollos realizados	15
2.2.1. Proceso médico.....	15
2.2.2. Entorno de desarrollo previo.....	15
3. Análisis de objetivos y metodología	16
3.1. Objetivos principales	16
3.2. Metodología	19
4. Diseño y resolución del trabajo realizado	20
4.1. Conexión de un Front-end y un Back-end	20
4.2. Extracción de datos de los informes	20
4.2.1. Archivos TXT	20
4.2.2. Archivos PDF.....	21
4.3. Desarrollo de la aplicación web	27
4.4. Conexión con la base de datos	34
4.5. Escritura de las plantillas Excel	36
4.6. Contenedores Docker.....	37
4.7. Resultado: Prueba de concepto	38
5. Conclusiones y vías futuras.....	46
Bibliografía.....	47

Índice de figuras

Figura 1. Distintos casos de fases en la terapia celular	15
Figura 2. Etapas generales en la terapia celular	16
Figura 3. Tablas de la base de datos para las etiquetas	21
Figura 4. Archivo PDF Tipo 1	22
Figura 5. Entrada multilínea PDF Tipo 1	22
Figura 6. Resultado de una entrada multilínea 1	22
Figura 7. Tiempo de lectura y extracción de datos en un archivo PDF Tipo 1	23
Figura 8. Primera página de un archivo PDF Tipo 2	23
Figura 9. Última página de un archivo PDF Tipo 2	24
Figura 10. Entrada multilínea PDF Tipo 2	25
Figura 11. Resultado de una entrada multilínea 2	25
Figura 12. Entrada multilínea con información omitida	25
Figura 13. Tiempo de lectura y extracción de información en un archivo PDF Tipo 2	25
Figura 14. Archivo PDF Tipo 3	26
Figura 15. Tiempo de lectura y extracción de información en un archivo PDF Tipo 3	26
Figura 16. Componente file_uploader de Streamlit	27
Figura 17. Fase inicial de la búsqueda de archivos en el sistema	28
Figura 18. Resultado de la búsqueda de archivos en el sistema	28
Figura 19. Botones de carga y borrado de archivos	29
Figura 20. Introducir Código de donación	31
Figura 21. Caso de búsqueda de un campo no existente	32
Figura 22. Caso de búsqueda de un campo existente	33
Figura 23. Ejemplo de introducir una búsqueda sin formatear	33
Figura 24. Tablas creadas en la base de datos	35
Figura 25. Ejemplo de una tabla de un fichero	35
Figura 26. Ejemplo de una tabla de etiquetas	35
Figura 27. Selección de la carpeta raíz	36
Figura 28. Botones para la creación de los informes	36
Figura 29. Dockerfile Front-end	37
Figura 30. Docker-compose.yaml	37
Figura 31. Página Inicio	38
Figura 32. Página Registrar donación	39
Figura 33. Sección Evaluación donante para un trasplante alogénico	39
Figura 34. Sección Aféresis	40
Figura 35. Sección Selección CD34+	41
Figura 36. Sección Descongelación y Lavado	42
Figura 37. Tiempo de lectura	43
Figura 38. Botón Crear Informe	43
Figura 39. Mensajes de error	43
Figura 40. Barra lateral	43
Figura 41. Página Visualizar DB	44
Figura 42. Página Buscar datos	45

Índice de tablas

Tabla 1. Parámetros del método buttons	30
Tabla 2. Ejemplo de formato del nombre de las columnas	34

Resumen

Durante el análisis de la información derivada de la terapia celular se genera un gran número de datos que son recogidos en los ficheros resultantes de los distintos procedimientos llevados a cabo en cada una de sus fases. Esto dificulta la obtención de información y conclusiones de forma sencilla y precisa en el sector sanitario. El objetivo de este trabajo es desarrollar una *herramienta bioinformática* que recoja y gestione dichos datos, unifique el procedimiento de la manipulación de éstos automatizando esta tarea para el personal sanitario.

Con el propósito de conocer los pasos a realizar en la terapia celular, así como llevar a cabo el desarrollo y comprobación de la herramienta, el *Servicio de Hematología del Hospital Universitario Virgen de la Arrixaca* proporcionó los informes y datos clínicos necesarios.

La aplicación con la cual se registraron los distintos informes y su posterior extracción de datos ha sido desarrollada en *Python*, ya que permite el uso de distintas librerías de lectura de archivos *PDF*, el manejo de datos organizados en tablas y la incorporación del entorno de trabajo *Streamlit*.

Los datos obtenidos fueron almacenados en un sistema *Back-end* usando *SQLite* para que la información obtenida quede disponible en una base de datos relacional, permitiendo así su uso en aplicaciones futuras en el ámbito de la *Inteligencia Artificial* y *Machine Learning*.

Con el fin de hacer la aplicación portable realizaremos su despliegue dentro de contenedores de *software* mediante *Docker*, garantizando así su abstracción y virtualización siendo operativa en cualquier equipo del hospital.

Extended Abstract

Currently, the large diversity of medical procedures, research in new drugs and the increase of population are factors driving the collapse of the conventional tools in the area of clinical studies [2]. The vast volume of calculations and data processing made by a computer has considerably increased. Thus, advancing in studies that improve the digitalization of society with the integration of bioinformatic tools which minimize the manual data processing by sanitary personal.

The data processing derived from cell therapy generates a vast amount of data which are stored in different. This hinders obtaining accurate information and conclusions in a simple way within the sanitary sector. This study aims the development and implementation of a bioinformatics tool which resolves a medical problem, that is, the processing of information derived from cell therapy of bone marrow transfer.

There exists a problem of heterogeneity when a vast volume of clinical data must be exploited. This work focuses on the case of blood cells. The main trouble is due to the lack of classification of the whole pool of hematopoietic stem cells (HSCs) as well as the new discovery of new types in the last years [1]. This has led problems of nomenclature despite the great importance of hierarchy and cell heterogeneity of the data used in this study.

With the aim of knowing the different steps in cell therapy, proceed with the development and checking of the tool, the Haematology Service of the Hospital-University Virgen de la Arrixaca provided the necessary documents and clinical data.

A key aspect to consider is the nature of the information since it involves private clinical data which must be treated with reliability and respecting their confidentiality. Besides, no error in the collection and manipulation of the data must be done since it could alter the study and determine either the success or failure in the treatment of a patient.

The prototype created is composed of the connection between a Front-end and a Back-end system, using the open-source app framework Streamlit for the creation of the web application and the relational database SQLite3. For the collection, management and extraction of the data from the cell therapy results, different algorithms have been created using the Python programming language. Finally, has been incorporated Docker for the deployment of the application within software containers.

A first step focuses in accurately knowing the medical process, in which the different steps in cell therapy are classified and the different study-cases considered in the development of the tool are generated.

Firstly, several meeting with the sanitary personal of the *cell and tissue therapy Lab* were stablished with the aim of fist-hand knowing the whole procedure of transplant transferring within cell therapy field. Furthermore, the methods employed in the field were examined and questions arised were studied.

There is large diversity of documents and steps-ways within the cell therapy. Even, the different procedures to make in each step can vary in the resultant amount and content of filed produced.

For instance, the initial step where the assessment of the patient is done is crucial. When an allogenic transplant is planned (i.e., stem cells derive from a person different to the patient), a study in both patient and donor will be done to guarantee the accurate match between them. However, in the case of an autologous transplant (i.e., stem cells derive from the own patient) only the first study is needed.

Another example would rely on the documents obtained during the initial step of Donor assessment in which multiple studies can exit and only the most recent is useful. Besides, in the step of *Pre-Apheresis*, there can exist either one or several counters of blood stem cell obtained along the days and thus, registering all the information will be fundamental to follow their evolution.

The following is a summary explanation of the different phases involved in the clinical process of cell therapy:

- Donor assessment: The donor is evaluated in order to study the suitability of the transplant between the donor and the recipient patient.
- Apheresis: Collection of hematopoietic progenitors by counting cells in the peripheral blood (PB) of the donor. Several controls may be necessary. Both pre- and post-harvest count are made in order to check which cell levels are maintained during the collection. Optionally, intermediate controls can be performed in the middle of the process. At the end, the apheresis bag of the final product is counted, and a microbiological control is performed.
- Selection/Depletion: Specific cells are selected from the processed product [7] for cell counting and microbiological control of the Post-Sepax [8] allowing automated processing of the blood with significant recovery results. Both the counting and control of a positive and a negative (or depleted) fraction are performed. The focus on either one or the other in case will be done depending on either a selection or a depletion procedure.
- Washing: Prior to this step, both in the apheresis phase and in the selection phase, the product may have been frozen and preserved until the moment of its washing and infusion in the recipient patient. A freezing of the rest of the product as the last step after infusion into the patient can be found.
- Infusion: Finally, the recipient is injected with the final product. A new transplantation may be necessary in the future to restart the operation.

Once the complete process has been studied and the medical problem to be solved has been analyzed, the different steps to be followed in the methodological development flow of this project are described below:

Front-end and Back-end connection

Set up and connect a simplified Front-end with a Back-end using the framework Django [6] in Python and the SQLite3 [9] relational database, making it accessible from external computers by making public the localhost of our server using the Ngrok tool [14].

Data extraction from the medical reports

Implementation of the algorithms for reading and extracting the different reports resulting from cell therapy, distinguishing and classifying the type of file to be processed.

The creation of these methods will be in a Python class using different string manipulation libraries or PDF file reading libraries, such as Fitz [\[10\]](#) and Tabula [\[11\]](#).

Finally, the information is collected in two joint lists with the key-value format.

Web application development

Build a simple and intuitive environment to be used by health staff based on a client-server architecture.

The MVC (Model-View-Controller) structure consists of the division of an application in these three modules: the View is the module in charge of displaying the view with which the user interacts, the Model manages the information in a database and the Controller manages the communications between the View and the Model.

We will use the open-source framework Streamlit [\[12\]](#), showing the content by pages incorporating the multipage functionality of its latest version.

The functionality included in its pages is the following:

- Home: Presentation page that will initially appear when running the application. In this page, information about the present work, its author and the functionality of the application are shown.
- Register donation: Main page where the type of transplant can be selected, browsing and reading all the reports to register a donation and finally create an Excel report that collects relevant data.
- Visualize database: Functionality that allows the visualization of the content of the database.
- Search data: Functionality which purposes is to search on the tables of either a patient or all of them within a specific field.

Connection to the database

Construction of the SQL functions necessary for the creation and initialization of tables, to add new columns according to the type of data read and finally to write to the database the information extracted from the medical reports collected by the main page of the application.

In addition, some other methods will be incorporated to complement the functionality of the web application, such as obtaining all the tables for its visualization or searching for information by columns in the database.

Web App optimization

After having a complete version of the web application, the content of the main page will be divided to avoid visual and performance overload by reducing its components.

A new component of self-development will be included. This, in addition to re-use the code of the main page, will extend the functionality of those already included in the project by replacing the Streamlit *file_upload* buttons by others that implement the Tkinter interface [13].

Writing Excel templates

Using the Excel templates provided by the Haematology Service, a new functionality is added to the application to compile from the reports collected in the database the most relevant information of each stage of cell therapy to be collected in a final report.

Docker Containers

The application will be deployed within software containers using Docker [3] for abstraction and virtualization in any operating system, thus ensuring the correct operation of the tool in any hospital computer, since making the app portable it is a requirement.

Result: Proof of concept

Finally, a prototype or proof of concept of this system is obtained as a result with a complete functional environment.

The integration of different computer technologies in the sanitary sciences is considered fundamental, thus preventing the sanitary staff from the processing of large amounts of data and avoiding possible human errors. In addition, tools such as the one presented in this work, help to bring about a change in the digitization of medical documents, which were collected in physical filing cabinets so far.

A plausible future implementation would be the application of Machine Learning and Artificial Intelligence techniques on the data extracted by the system by examining the parameters that can determine the success of a transplant. This would be possible through the use of a relational database that stores the information collected in tables.

Conventional techniques carried out in hospitals for the manual processing of these reports are considered slow and delicate tasks, while delegating this job to an automated system can be performed the task of collecting, reading and recording in a database more than a dozen reports in a few seconds and being able to search on them efficiently.

1. Introducción

Este proyecto toma como objetivo el diseño e implementación de una herramienta bioinformática que resuelva un problema médico real como es la gestión de la información dentro de la terapia celular del trasplante de médula ósea, procedimiento que puede arrojar un gran número de datos difíciles de manipular y gestionar por el personal sanitario.

Un aspecto a tener en cuenta la naturaleza de la información a administrar, pues se trata de datos clínicos privados que han de ser tratados con fiabilidad y respetando su confidencialidad.

Surge un problema de heterogeneidad ante la explotación de grandes volúmenes de datos clínicos. Este trabajo se ha centrado en el caso de las células sanguíneas. El mayor inconveniente se debe a una falta de clasificación del conjunto de las células madre hematopoyéticas (HSCs) y a que se está produciendo a lo largo de los años la aparición de nuevos tipos [1], generando problemas de nomenclatura cuando es de gran importancia la organización jerárquica y la heterogeneidad celular que poseen los datos a tratar en este proyecto.

Inicialmente, la línea de este trabajo fin de grado tenía como título *"How to Build a Simple Machine Learning Web App in Python"*, donde se estudiaron algunos componentes que han sido incluidos en el actual proyecto tras recibir la propuesta de ampliar la línea de trabajo y desarrollar un prototipo real bajo la codirección del Dr. D. Miguel Blanquer Blanquer, del Laboratorio de Terapia Celular y Tisular del Servicio de Hematología del Hospital Universitario Virgen de la Arrixaca.

Previamente, fue necesario realizar un estudio del procedimiento a llevar a cabo en la terapia celular, de la actual gestión de su información y de los objetivos a alcanzar para cubrir las necesidades actuales.

El prototipo creado se compone de la conexión entre un sistema Front-end y Back-end, usando el *framework* de código abierto Streamlit para la creación de la aplicación web y la base de datos relacional SQLite3. Para la recogida, gestión y extracción de datos de los informes resultados de la terapia celular se han creado distintos algoritmos usando el lenguaje de programación Python. Finalmente, se ha incorporado la herramienta Docker para el despliegue de la aplicación dentro de contenedores de *software*.

1.1. Motivación

En la actualidad, la diversidad de procedimientos médicos, la investigación de nuevos fármacos o el aumento de la población son factores que están provocando un colapso de las herramientas convencionales en el ámbito de estudios clínicos [2].

Es por este motivo que el sector salud se encuentra en proceso de la digitalización de documentos que hasta ahora eran almacenados en archivadores físicos.

Sin embargo, el volumen de cálculo y análisis de datos a ser procesado por un ordenador se ha visto incrementado considerablemente, por lo que es vital avanzar en estudios que mejoren esta digitalización de la sociedad con la integración de tecnologías o herramientas bioinformáticas que minimicen la gestión manual de datos por parte del personal sanitario.

Es importante recordar que en esta área se trabaja con información sensible, pues no es tolerable ningún tipo de error en la recogida o manipulación de estos datos ya que podría alterar el estudio y ser la diferencia entre un éxito o fracaso en el tratamiento de un paciente.

Para llevar a cabo el desarrollo de este proyecto deberemos trabajar en distintos campos de la informática ya estudiados durante el grado en una situación real, como es el uso y manejo de base de datos, además de aprender nuevos conceptos como son el desarrollo de una app levantando un servidor, la programación en *Python* descubriendo nuevas interfaces o librerías útiles para nuestro propósito o la virtualización en contenedores *Docker* [3].

En definitiva, la línea de este trabajo me pareció una propuesta interesante y además he encontrado mis aportaciones muy valoradas, pudiendo colaborar con el sector sanitario desde la informática empleando distintas áreas de ésta.

2. Estado del arte

2.1. Trabajos relacionados

Mencionamos en este apartado dos trabajos previos realizados dentro del Máster en Bioinformática de la Universidad de Murcia, bajo la dirección del Dr. José Manuel García Carrasco y el Dr. Miguel Blanquer.

El primero está titulado "*Diseño e implementación de un sistema informático de recogida y gestión de datos para la terapia celular del trasplante de médula ósea*", realizado por Francisco Herrera González [4].

El segundo ha sido realizado por Elena de la Cruz Martínez, sin título provisional y pendiente de ser defendido en septiembre del curso 2021/22 [5].

El desarrollo del presente proyecto ha partido de ambos trabajos, teniendo en cuenta los diferentes estudios, análisis y componentes que forman parte del sistema propuesto en el primero.

A su vez, el segundo proyecto toma como objetivo sistematizar el proceso de la obtención y gestión de los informes que intervienen en las distintas etapas de la terapia celular, tratando con el personal sanitario involucrado.

Con la finalidad de poder crear y calibrar la herramienta bioinformática que venimos a presentar, ha sido fundamental haber partido de este proceso médico ya resuelto, pudiendo así tener recogidos en un único directorio los informes involucrados en cada donación de la terapia celular, siguiendo cierto orden o nomenclatura y listos para ser usados en este trabajo.

2.2. Desarrollos realizados

2.2.1. Proceso médico

Como ya adelantábamos, partimos de la formalización del proceso médico, donde además de llevar a cabo la recogida de casos de uso reales, se clasifican las diferentes etapas por las que transcurre la terapia celular, dando a lugar a los distintos casos que se contemplarán durante la creación de este sistema. En la siguiente figura podemos ver unos ejemplos de estas fases que intervienen en un trasplante de médula ósea.

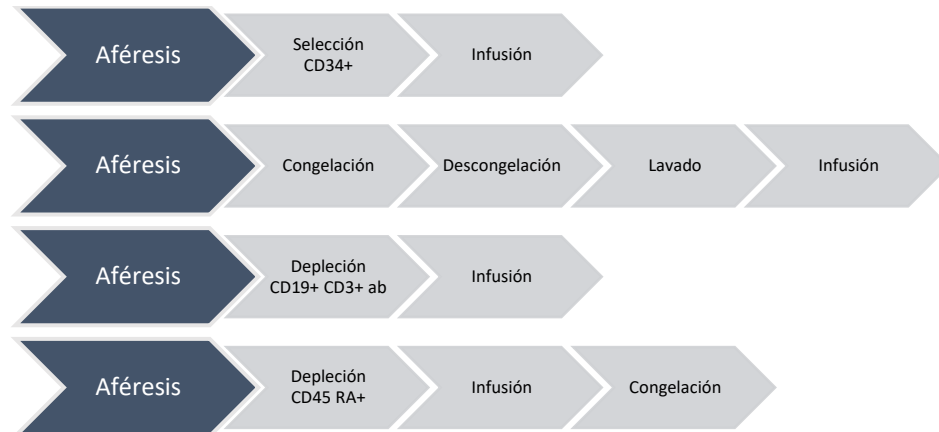


Figura 1. Distintos casos de fases en la terapia celular

2.2.2. Entorno de desarrollo previo

Entre los componentes propuestos en los trabajos sobre los que partimos, encontramos como lenguaje de programación escogido *Python* debido a su integración con el resto de partes del sistema como es la conexión con la base de datos o el manejo y lectura de ficheros, ya que es un lenguaje de programación multiparadigma que permite programación funcional y orientada a objetos, además que poder funcionar bajo cualquier sistema operativo

Para el framework o entorno de trabajo fue empleado previamente *Django* [6] por su manejo en tareas como la conexión con una base de datos, pero dificultando otras como la creación de vistas debido al uso de código *HTML* y *CSS*, además de ser necesario la implementación de ciertas funcionalidades complementarias.

La propuesta anterior para la arquitectura de la base de datos fue usar un modelo no relacional tras un estudio previo de ventajas e inconvenientes de las diferentes alternativas, finalmente usando *MongoDB* debido a su carácter de base de datos orientada a documentos.

3. Análisis de objetivos y metodología

3.1. Objetivos principales

Estudio previo del problema médico a resolver

Con la propuesta de ampliar la línea de trabajo, inicialmente se realizaron unas reuniones de manera presencial con el personal sanitario del *Laboratorio de terapia celular y tisular* con el objetivo de conocer de primera mano cómo es el procedimiento completo de una donación y un trasplante dentro de la terapia celular, además de analizar la metodología actual de trabajo en esta área estudiando las cuestiones a resolver.

Existe una alta diversidad de documentos físicos y de etapas dentro de la terapia celular e incluso los procedimientos a realizar en cada etapa también pueden variar la cantidad y contenido de los ficheros resultantes.

Por ejemplo, en la etapa inicial donde se realiza una *evaluación del donante*, en caso de que se trate de un *trasplante alogénico* (las células madre provienen de una persona distinta al paciente) será necesario un estudio tanto del paciente receptor como del donante sano, para garantizar la adecuada interacción entre ambos. Sin embargo, si se tratase de un *trasplante autólogo* (las células madre se obtienen del propio paciente), únicamente es necesario el primer estudio.

Otro ejemplo sería en los informes obtenidos durante esta etapa inicial *Evaluación donante*, donde pueden haberse realizado distintos estudios de los cuales únicamente interesa el más reciente. En cambio, en la etapa de *Pre-Aféresis* se pueden encontrar uno o varios contajes de células de sangre periférica obtenidos en distintos días y será necesario registrar todos estos informes para así marcar su evolución.

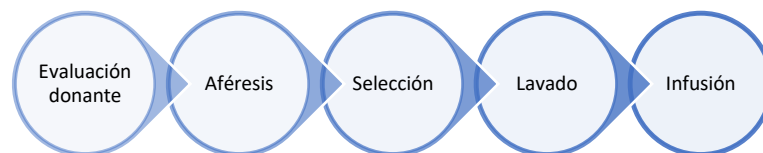


Figura 2. Etapas generales en la terapia celular

A continuación, se explica de manera resumida las distintas fases que intervienen en el proceso clínico de terapia celular:

Evaluación Donante

Se somete a evaluación la persona donante con el fin de estudiar la idoneidad del trasplante entre éste y el paciente receptor.

Aféresis

Recolección de progenitores hematopoyéticos mediante un conteo de células en la sangre periférica (SP) del donante. Pueden ser necesarios varios controles.

Para comprobar que los niveles celulares se mantienen durante la recolección es realizado un conteo previo y otro posterior. Opcionalmente, se puede realizar unos controles intermedios a mitad del proceso.

Finalmente se realiza un conteo de la bolsa de aféresis del producto final y un control microbiológico.

Selección/Depleción

Son seleccionadas unas células específicas en el producto procesado [7] para un conteo de células y control microbiológico del *Post-Sepax* [8] permitiendo el procesamiento automatizado de la sangre con unos resultados de recuperación significativos. Se realiza conteo y control de una fracción positiva y otra negativa (o deplecionada) y nos centraremos en una u otra en caso de que estemos realizando una selección o una depleción.

Lavado

Previamente a este paso puede haberse producido tanto en la fase de *Aféresis* como en la *Selección* una *Congelación* del producto y haberse conservado hasta el momento de su *Lavado e Infusión* en el paciente receptor. También podemos encontrar una congelación del resto del producto como último paso tras la infusión en el paciente.

Infusión

Finalmente, el receptor es inyectado con el producto final. Puede ser necesario un nuevo trasplante en el futuro que reinicie la operación.

Una vez estudiado el proceso completo y analizado el problema médico a resolver, nos planteamos unos objetivos para seguir el flujo de desarrollo metodológico de este proyecto.

Objetivo 1: Conexión de un Front-end y un Back-end

Levantar y conectar un *Front-end* y un *Back-end* simplificados utilizando el marco de trabajo *Django* en *Python* y la base de datos relacional *SQLite3* [9], que sea accesible desde equipos externos.

Objetivo 2: Extracción de datos de los informes

Implementación de los algoritmos necesarios para la lectura y extracción de los distintos informes resultantes de la terapia celular, distinguiendo y clasificando el tipo de archivo a tratar. La creación de estos métodos será en una clase *Python* incorporando diferentes librerías de manipulación de cadenas *string* o de lectura de archivos *PDF*, como *Fitz* [10] y *Tabula* [11]. Ordenar la información recogida en dos listas conjuntas con el formato clave-valor.

Objetivo 3: Desarrollo de la aplicación web

Elaboración de un entorno simple e intuitivo a ser utilizado por el personal sanitario mediante el marco de trabajo Streamlit [12], ofreciendo las siguientes opciones: registrar una nueva donación, examinar las donaciones existentes y buscar información dentro de los registros anteriores. La aplicación debe ser capaz de examinar los distintos informes en la sección *Registrar donación* y conectarse correctamente con la base de datos.

Objetivo 4: Conexión con la base de datos

Construcción de las funciones en lenguaje *SQL* necesarias para la creación e inicialización de tablas, para añadir columnas nuevas según el tipo de datos leído y finalmente para escribir en la base de datos la información extraída de los informes médicos recogidos por la página principal de la aplicación. Además, se incorporarán los métodos necesarios para complementar la funcionalidad de la aplicación web, como es la obtención de todas las tablas para su posterior visualización o la búsqueda de información por columnas en la base de datos.

Objetivo 5: Optimización de la aplicación web

Tras tener una versión completa de la aplicación web funcional e integrada con la base de datos, estudiaremos el rendimiento de su funcionamiento buscando las mejoras posibles. Se estudiará la división por secciones en la página para el registro de donaciones, evitando la sobrecarga visual y de rendimiento reduciendo componentes. Además, nos plantearemos la posibilidad de incorporar un nuevo componente de elaboración propia que además de lograr la reutilización de código en la página principal, amplíen la funcionalidad de los ya incluidos en el proyecto sustituyendo los botones para la subida de archivos por defecto Streamlit por otros que implementen la interfaz *Tkinter* [13].

Objetivo 6: Escritura de las plantillas Excel

Empleando las plantillas *Excel* proporcionadas por el *Servicio de Hematología*, se añade a la aplicación la funcionalidad necesaria para recopilar de los informes leídos y registrados en la base de datos la información más relevante de cada etapa de la terapia celular para ser recogidos en un informe final.

Objetivo 7: Contenedores Docker

Se llevará a cabo el despliegue de la aplicación dentro de contenedores de *software* mediante *Docker* para la abstracción y virtualización en cualquier sistema operativo, garantizando así el correcto funcionamiento de la herramienta en cualquier equipo del hospital.

Resultado: Prueba de concepto

Finalmente, añadiendo las incorporaciones de cada objetivo al desarrollo previo, se obtiene como resultado un prototipo o prueba de concepto de este sistema con todo un entorno funcional.

3.2. Metodología

Tras establecer los pasos a seguir, explicamos algunos conceptos de la *Ingeniería de Software* y metodología sobre cómo se van a cumplimentar estos objetivos.

Para el desarrollo de la aplicación resultante, nos basaremos en una arquitectura *cliente-servidor*. Dentro del servidor, levantaremos un *Front-end* usando el *framework Django* o *Streamlit*, realizando previamente un ciclo de aprendizaje de ambos entornos de trabajo y el diseño *modelo-vista-controlador* utilizando los lenguajes de programación *Python* y *HTML*.

La estructura *MVC* (Modelo-Vista-Controlador) consiste en la división de una aplicación en estos tres módulos: la Vista es el módulo encargado de mostrar la vista con el que el usuario interactúa, el Modelo se encarga de la gestión de la información en una base de datos y el Controlador es el módulo encargado de gestionar las comunicaciones que existen entre la Vista y el Modelo.

Levantaremos un *Back-end* usando la base de datos relacional *SQLite3* y el programa *DB Browser for SQLite* que proporciona una interfaz sencilla con la que comprobar las tablas creadas y que la información introducida queda correctamente registrada. En caso de usar *Django*, podremos hacer público y accesible el localhost de nuestro servidor usando la herramienta *Ngrok* [14].

Debido a los requisitos altos de seguridad que tiene el manejo de ficheros con información sensible, la aplicación web en lugar de "cargar" estos informes únicamente serán examinados en el equipo para así marcarles su localización a los algoritmos de extracción de datos, que se ejecutarán dentro de los propios equipos del hospital.

El desarrollo de la aplicación ha sido llevado a cabo en una máquina virtual de Linux Ubuntu 22.04 LTS. Sin embargo, es una obligación hacer esta aplicación portable y funcional independientemente del equipo o sistema operativo sobre el que se ejecute, por lo que no existen unos requisitos mínimos. Esto es conseguido incluyendo en la imagen de un contenedor *Docker* todas las librerías necesarias con las versiones empleadas en la elaboración del proyecto.

4. Diseño y resolución del trabajo realizado

Como ya vimos en la sección anterior, antes de comenzar la creación de esta herramienta fueron necesarias unas visitas físicas al hospital para, además de comprender el procedimiento completo de una donación y un trasplante de médula ósea en la de la terapia celular, estudiar con el personal sanitario del *Servicio de Hematología* los distintos informes clínicos con los que vamos a tratar.

Una vez acabada esta fase de análisis del problema médico a resolver y tenemos una idea de las características o requisitos a incorporar en nuestro sistema, elaboramos unos pasos para su guiar su desarrollo y realizamos una etapa intermedia para el aprendizaje de las distintas tecnologías a utilizar.

En esta sección trataremos de explicar en detalle la resolución de cada objetivo con las decisiones tomadas durante su elaboración o los problemas que hayan podido surgir.

4.1. Conexión de un Front-end y un Back-end

Durante la etapa previa de análisis de requisitos, a modo de prueba se elaboró una página principal simplificada usando *Django* con unos pocos campos de texto para rellenar y que fuera capaz de conectarse a una base de datos y registrar esa información.

Esto se pudo mostrar en una reunión usando la herramienta *Ngrok* a través de *Django* para conectarnos desde el hospital a la máquina sobre la que se ejecutaba el proyecto.

4.2. Extracción de datos de los informes

Una vez obtuvimos acceso a casos reales de donaciones y a sus informes clínicos, pudimos analizar los distintos archivos que el programa espera recibir desde la página principal, encontrando archivos con extensión *TXT* y distinguiendo varios tipos de *PDF*.

4.2.1. Archivos TXT

En caso de que estemos examinando un archivo con extensión *TXT* se tratará o bien de un código de donación que usaremos como identificador en las tablas de la base de datos o se tratará de una etiqueta, la cual prepararemos para que sí sea registrada en una tabla específica.

Código de donación

El código de donación son 13 dígitos formado por el ID del centro donde se han tomado las muestras, el año de la toma y el número del proceso. Por ejemplo: E0052 21 100235.

Para el Hospital Universitario Virgen de la Arrixaca, el código del centro será E0052.

Los primeros tres dígitos del número del proceso indican si se trata de un *trasplante alogénico* (100) o *autólogo* (021) mientras que los últimos dígitos indican el orden cronológico dentro de cada tipo de trasplante.

Etiquetas

Las etiquetas son formadas por un código de 8 dígitos y nos indican también si se trata de un *trasplante alogénico* o *autólogo* y el número de bolsas del producto. Por ejemplo: S1869400.

El antepenúltimo dígito (4) nos indica que nos encontramos con un *trasplante alogénico*, siendo 1 si se tratase de un *trasplante autólogo*. Los dos últimos dígitos marcan el número de bolsas del producto, encontrando únicamente una en este caso (00). Si existieran varias, se indicarían mediante la siguiente nomenclatura: A0, B0, C0...

Cultivo Microbiológico Frac Pos			
	Codigo_Donacion	Fecha	Hemocultivo
0	E005222100000	17/03/2021	Hemocultivo Negativo a los 5 días de incubación.

Etiqueta Infusión		
	Codigo_Donacion	Etiqueta_0
0	E005222100000	S2847400

Etiquetas Congelación				
	Codigo_Donacion	Etiqueta_0	Etiqueta_1	Etiqueta_2
0	E005222021000	S25281A0	S25281B0	S25281C0

Figura 3. Tablas de la base de datos para las etiquetas

Como vemos en la anterior figura, el archivo *TXT* que contiene el código de donación es usado como identificador de todas las tablas creadas en la base de datos junto a la fecha que dicte cada informe, mientras que el resto de archivos *TXT* leídos que contienen códigos de etiquetas son registrados en su correspondiente tabla, sin este campo *Fecha*.


4.2.2. Archivos PDF

La extracción de datos de los distintos informes clínicos en formato *PDF* mediante la elaboración de distintos algoritmos ha sido una tarea muy importante dentro de este proyecto. Es por ello por lo que vamos a explicar con detalle la resolución de este objetivo repasando tanto las decisiones tomadas como sus motivos.

PDF Tipo 1

Encontraremos este tipo de archivos en la fase de *Evaluación del donante*, pues tanto para el informe *Estudio del donante sano* como para el informe *Estudio del pre-trasplante* existen datos demográficos, hemograma, coagulación, bioquímica y demás información en un formato formulario por líneas, simplificando así la extracción de los campos de éste.

En la siguiente figura, donde se ha borrado la información personal del paciente, podemos echar un vistazo al aspecto de estos archivos:

Donante sano PH - HEM					
	Paciente		Sexo		Cama
	NHC		Fecha Nac.		
	NSS		Servicio	Hematología	
	DNI		Ámbito	HOSPITALIZACION	
	Alertas				

ÁREA DE SALUD 1
Murcia-centro
Arrixaca

Fecha de la toma: 26-oct-2018 11:41

Glucosa: 101
Urea: 28
Creatinina: 0.88
Ác. Úrico: 5.4
Proteínas totales: 7.3
Albumina: 4.3
Calcio: 10.0
Bilirrubina total: 0.17
Colesterol: 184
AST/GOT: 14
ALT/GPT: 14
Fosfatasa alcalina: 89

Figura 4. Archivo PDF Tipo 1

El modo de proceder será leyendo por líneas el contenido del fichero desde que el comienzo del estudio (línea Fecha de la toma). Tomemos como ejemplo la información de la siguiente línea: *Reticulocitos: 1.43*. En la dupla clave-valor registraríamos como clave la parte izquierda de la cadena separada por el símbolo ":" y la parte derecha como valor.

Exploración Analíticas: Hemograma: Leucocitos 8820 (4360 N, 3210 L, 790 M). Hb 11.2 g/dl. Hto: 35.7%, VCM: 83 fL. Plaquetas: 433000. Vit B12 538. Cuantificación Igs: IgG 1220, IgA 356, IgM 168.

Figura 5. Entrada multilínea PDF Tipo 1

En caso de que encontremos una entrada multilínea como en la figura anterior, sería detectada la parte faltante (Plaquetas..., Vit..., Cuantificación Igs...) e introducida en su campo correspondiente, quedando correctamente recogida toda la información en su correspondiente columna. Podemos comprobarlo en la siguiente imagen:

Estudio Donante Sano

is	Exploracion_Analiticas
o do	Hemograma: Leucocitos 8820 (4360 N, 3210 L, 790 M). Hb 11.2 g/dl. Hto: 35.7%, VCM: 83 fL. Plaquetas: 433000. Vit B12 538 Cuantificación Igs: IgG 1220, IgA 356, IgM 168.

Figura 6. Resultado de una entrada multilínea 1

Puede darse el caso de que encontremos más de un estudio en un mismo fichero.

La forma de proceder en este caso será conservar únicamente el estudio más reciente actualizando los valores registrados anteriormente e introduciendo las columnas nuevas.

Para este tipo de fichero leeremos la información usando la librería *Fitz*, pues no es necesario el uso de la librería *Tabula* al no encontrar los datos distribuidos en un formato por filas y columnas.

Es por este motivo que el tiempo de extracción de datos en estos archivos no es más que unas milésimas de segundo.

2. Estudio donante sano 5431181 formulario.pdf leído correctamente en 0.017 segundos.

3. Estudio pre-trasplante 5416380 formulario.pdf leído correctamente en 0.015 segundos.


Figura 7. Tiempo de lectura y extracción de datos en un archivo PDF Tipo 1

PDF Tipo 2

Este es el tipo de archivo que encontraremos con más frecuencia en todo el proceso, pues es con los que se trabaja en todas las etapas de la terapia celular a excepción de la primera. Emplearemos aquí la librería *Tabula* con ayuda de otros programas adicionales.


Además, el método de recogida de datos en este tipo de archivos será el más complejo de implementar por el complejo tratamiento de la información obtenida y por contar con un mayor número de variantes a considerar, dependiendo del número de páginas o secciones de cada uno.

Primero, echemos de nuevo un vistazo a este tipo de archivos para ver su forma de distribuir la información a obtener. De nuevo, han sido borrados los datos personales del paciente.



Área I
Murcia Oeste
Asistencia

SERVICIOS DE ANÁLISIS CLÍNICOS,
HEMATOLOGÍA, INMUNOLOGÍA, MEDICINA
NUCLEAR Y ALERGIA



Servicio
Murciano
de Salud

Nº Petición:
Apellidos:
Edad:
Nº TSI:
Nº de Historia:
Médico:
Diagnóstico:

C.I.P. Autonómico:
Nombre:
Fecha de Nacimiento:
Nº S.S.:
Centro de Extracción:
Cama:
Servicio:
Destino:

Fecha Solicitud:
Observaciones:
HEMATIMETRÍA (sangre total)
Resultados validados por:

Fecha Recepción:

Fecha Envío:

Hemograma

Serie Roja

Hematíes
Hemoglobina
Hematocrito
Volumen corpuscular medio
Hemoglobina corpuscular media
Concentración hemoglobina corp. media
Ancho de distribución eritrocitaria (CV)

4.5
12.2
37.7
84.2
27.2
32.4
14.6

x10⁶/uL
g/dL
%
fL
pg/célula
g/dL
%

4.0 - 5.2
12.0 - 16.0
36.0 - 46.0
80.0 - 100.0
26.0 - 34.0
31.0 - 36.0
11.5 - 14.5

Eritroblastos
Eritroblastos %

** 0.10
* 0.20

x10³/uL
/100WBC

0.00 - 0.03
0.00 - 0.05

Serie Plaquetar

Plaquetas
Volumen plaquetario medio
Ancho de distribución plaquetar

278.0
10.7
12.1

x10³/uL
fL
%

150.0 - 350.0
6.4 - 11.0
9.0 - 17.0

Serie Blanca

Leucocitos
Neutrófilos
Linfocitos
Monocitos
Eosinófilos
Basófilos
Neutrófilos inmaduros
Cociente granul. inm/ granul.t
Cociente Recuento Neutrófilos/ Linfocitos

** 66.51
** 56.42
3.61
** 5.47
0.85
0.16
* 4.740
0.084
* 15.63

x10³/uL
x10³/uL (84.90 %)
x10³/uL (5.40 %)
x10³/uL (8.20 %)
x10³/uL (1.30 %)
x10³/uL (0.20 %)
x10³/uL 7.10 %

4.50 - 11.00
1.80 - 7.70
1.00 - 4.00
0.00 - 0.80
0.00 - 0.50
0.00 - 0.20
0.000 - 0.100
0.000 - 0.160
0.10 - 3.13

Figura 8. Primera página de un archivo PDF Tipo 2

**SERVICIOS DE ANÁLISIS CLÍNICOS, HEMATOLOGÍA, INMUNOLOGÍA,
MEDICINA NUCLEAR Y ALERGIA**

Paciente:
Número:

CITOMETRÍA - INMUNOFENOTIPO (S. INMUNOLOGÍA HCUVA)

Resultados validados por:

Nº de citometría		
Tipo de muestra	Sangre periférica	
Progenitores hematopoyéticos (CD34, CD133, etc.)	Ver descripción de la muestra	
Referencia progenitores hepatopoyéticos (CD34)	CONTROL PHSP	
Panel de inmunofenotipo	CD34 Truecount	
Factor de Dilucion CD34	1	
LEUCOSviabiles (%)	99.85	%
LINFOCITOS (%)	6.45	%
MONOCITOS (%)	5.71	%
NEUTROFILOS (%)	87.55	%
CD34+ (%)	0.163	%
LEUCOSviabiles (cel/ml x10 ⁶)	65.626	Millon/ml
	65626.45	Cels/μL
Linfocitos (Millon/ml)	4.2299	Millon/ml
	4229.89	Cels/μL
Monocitos (Millon/ml)	3.7473	Millon/ml
	3747.33	Cels/μL
Neutrofilos (Millon/ml)	57.4589	Millon/ml
	57458.88	Cels/μL
CD34+ (Millon/ml)	0.107	Millon/ml
	106.77	Cels/μL

Servicios de Análisis Clínicos y Hematología acreditados por ENAC según la norma UNE-EN ISO 15189 (nº579/LE1193). El listado de pruebas acreditadas se puede consultar en la Intranet del hospital

Servicio de Inmunología (Laboratorio Regional de Histocompatibilidad e Inmunogenética) acreditado por la European Federation for Immunogenetics (EFI-09-ES-009.991).

Murcia, a 22 de marzo de 2021

Figura 9. Última página de un archivo PDF Tipo 2

El mayor problema del uso del método tabula es que resulta ineficaz recogiendo y distribuyendo en columnas los datos si en la página se encuentra información que no respeta este formato tabla, como es el caso al final de la última página o en la primera, con la sección inicial para los datos del paciente.

La forma de solventar esto es restringiendo por áreas la zona donde extraer la información, indicando el límite superior, inferior, izquierdo y derecho de dicha área.

El límite izquierdo y derecho de esta área irá en todos los casos de cero al ancho de la página, mientras que el límite superior dependerá de si estamos tratando la primera página u otra, por el encabezado que incorpora cada una, o de si estamos extrayendo la información por secciones dentro de un informe.

Para encontrar el píxel del límite superior de esta área he usado el programa *SumatraPDF*, donde podemos buscar las coordenadas del encabezado de la página de un *PDF* en píxeles.

Para encontrar el comienzo o final de una sección e indicar este límite superior e inferior hemos empleado de nuevo la librería *Fitz* con su método *search_for*, que nos devolverá las coordenadas de un rectángulo que encuadra la posición donde se encuentre la palabra introducida como parámetro.

Igual que realizábamos en el apartado anterior, en caso de encontrar información multilínea se detectará y se introducirá en su correspondiente campo.

Fenotipo eritrocitario (Rh+Kell)	c (-) C (+) E (-) e (+) K (-)
----------------------------------	---

Figura 10. Entrada multilínea PDF Tipo 2

Contaje Control Pre Aféresis	
	Fenotipo_eritrocitario_RhposKell
0	c (-) C (+) E (-) e (+) K (-)

Figura 11. Resultado de una entrada multilínea 2

En cambio, si la información viene en multilínea por tratarse de un cambio de unidades como vemos en la siguiente figura, omitiremos esta información. Detectaremos este caso si el valor de la línea extra a añadir es igual al valor de la línea anterior multiplicado por 10^3 .

Linfocitos (Millon/ml)	4.2299 4229.89	Millon/ml Cels/ μ L
Monocitos (Millon/ml)	3.7473 3747.33	Millon/ml Cels/ μ L

Figura 12. Entrada multilínea con información omitida

El tiempo de ejecución de la lectura y extracción de datos de estos archivos *PDF* Tipo 2 empleando la librería *Tabula* es bastante más elevado que el que encontrábamos en el método anterior, yendo desde los 2 hasta los 10 segundos por archivo.

1. Contaje Control Pre-aféresis (22-02-21).pdf leído correctamente en 9.6 segundos.
1. Contaje Control Pre-aféresis (23-02-21).pdf leído correctamente en 3.8 segundos.
1. Contaje Control Pre-aféresis (25-02-21).pdf leído correctamente en 4.0 segundos.
2. 1ºTubo SP (26-02-21).pdf leído correctamente en 4.0 segundos.
3. 2ºTubo SP (26-02-21).pdf leído correctamente en 4.1 segundos.
5. Contaje P.Final BolsaAferesis (26-02-21).pdf leído correctamente en 2.1 segundos.
Finalizado! Tiempo: 27 segundos.

Figura 13. Tiempo de lectura y extracción de información en un archivo PDF Tipo 2

PDF Tipo 3

En los informes donde se incluya información sobre un *cultivo microbiológico* únicamente leeremos en el espacio desde donde comienza la información que buscamos hasta la línea de separación. Podemos verlo en la siguiente figura, donde queda marcado en rojo el área de interés.

INFORME DEL SERVICIO DE MICROBIOLOGIA

DPLICADO

Nombre: _____ N° Historia: _____ N° Petición: _____
 Fecha Nacimiento: _____ Edad: _____ Sexo: _____ Fec. Registro: _____ Ref. externa: _____
 N° Seg. social: _____ CIP: _____ Procedencia: _____
 Médico: _____ Servicio: _____
 Observaciones: _____ Cama: _____ F.Extracción: _____
 C. Extracción: _____

Tipo muestra: **Otras Muestras (Sala Blanca)** Validación: _____

BACTERIOLOGIA

Test esterilidad (SB)
 Otras Muestras (Sala Blanca)

Hemocultivo TRG

Negativo a los 7 días de incubación.

Validado por: _____

Figura 14. Archivo PDF Tipo 3

El tiempo de ejecución de este tipo de *PDF* al igual que pasaba con los archivos del primer tipo, es algo insignificante. Esto es debido a que no utilizamos el método *tabula* con un área, sino que leeremos con *Fitz* el fichero por líneas como ya realizábamos con los de tipo 1, indicando el principio y el final del espacio donde leer los resultados del cultivo.

5. CultivoMicro P.Final BolsaAferesis.pdf leído correctamente en 0.017 segundos.
2. CultivoMicro Post-Sepax.pdf leído correctamente en 0.0073 segundos.
3. CultivoMicro Frac.Deplec..pdf leído correctamente en 0.017 segundos.
4. CultivoMicro Frac. Positiva.pdf leído correctamente en 0.012 segundos.

Figura 15. Tiempo de lectura y extracción de información en un archivo PDF Tipo 3

4.3. Desarrollo de la aplicación web

Sección en la que explicaremos el desarrollo de la aplicación web, desde la ya mencionada página de prueba simplificada empleando *Django* para probar la conexión con la base de datos hasta el cambio por el nuevo entorno de trabajo de código abierto *Streamlit* y la posterior optimización de la aplicación, realizando una división del contenido por páginas y secciones desplegables, además de sustituir componentes originales de *Streamlit* por otros de elaboración propia incorporando *Tkinter*.

Con el cambio de framework a *Streamlit*, el cual ya se estudió en la línea original de este trabajo, además de poder incorporar nuevas tecnologías pudimos ahorrar en el proyecto la programación en código *HTML* y *CSS*.

Un inconveniente del cambio es que en *Django* se realizaba de forma muy sencilla la conexión con la base de datos mientras que empleando esta alternativa habría que programar ciertas funciones en código *SQL* para la creación de tablas y escribir o consultar información.

En este punto donde conocíamos las fases que intervienen en la terapia celular y los ficheros que resultan de cada proceso, creamos una página principal para examinar y registrar dichos ficheros con *Streamlit* incorporando la funcionalidad más reciente de su última versión (1.10.0 Release date: June 2, 2022) como es el soporte para aplicaciones con multipáginas.

En una primera versión se usó el componente *file_uploader* de *Streamlit* para la carga de ficheros:

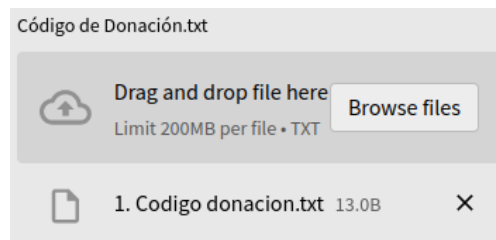


Figura 16. Componente *file_uploader* de *Streamlit*

El problema de este método es que devuelve una subclase de *BytesIO* del tipo *file-like* que se puede utilizar como argumento en cualquier método que espere un archivo, pero nuestras funciones de lectura de *PDF* necesitan recibir como parámetro la ubicación o *path* del archivo, cosa que no podemos obtener del fichero examinado por un asunto de seguridad y privacidad en navegadores web. Lo que sí podemos obtener es el nombre del archivo examinado y el espacio que ocupa en el disco en bytes.

Se encontró una solución provisional usando unas funciones de *Python* capaces de localizar los archivos en el sistema que coincidan con el nombre pasado como parámetro y comprobando además que los ficheros encontrados ocupen el mismo espacio en disco que el que habremos examinado.

Sin embargo, seguía habiendo un inconveniente y es que podíamos encontrar varios ficheros distintos al que buscamos por coincidir en el mismo nombre.

Veamos un ejemplo en la siguiente imagen.

```
0 : "1.Codigo donacion.txt"
1 :
"/home/josebelmonte/Escritorio/pdfs/autologo/021012/1. Evaluación
donante/1. Codigo donacion.txt"
2 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100234/1. Evaluacion
donante/1. Codigo donacion.txt"
3 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100236/1. Evaluacion
donante/1. Codigo donacion.txt"
4 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100233/1. Evaluacion
donante/1. Codigo donacion.txt"

1 : [
0 : "2. Estudio_DonanteSano_882100_formulario.pdf"
1 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100236/1. Evaluacion
donante/2. Estudio_DonanteSano_882100_formulario.pdf"
```

Figura 17. Fase inicial de la búsqueda de archivos en el sistema

En este ejemplo estaríamos examinando y buscando dos archivos: *1. Código donación.txt* y *2.Estudio_DonanteSano_882100_formulario.pdf* del proceso número 100236.

Como vemos en las primeras 4 entradas, en el sistema se han encontrado cuatro archivos con un nombre similar al que buscamos mientras que para el estudio del donante sano se ha encontrado únicamente uno por tener un nombre más distintivo. ¿Cómo sabemos entonces cuál de los cuatro códigos de donación encontrados es el que nos interesa?

Realizábamos entonces un conteo en cada ruta del archivo “duplicado” buscando el que mejor coincide con la ruta del archivo con una única coincidencia (Estudio Donante Sano).

De este modo, como vemos en la siguiente figura, nos quedamos con las rutas únicas de los archivos que hemos examinado y buscado en el sistema.

```
0 : "1. Codigo donacion.txt"
1 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100236/1. Evaluacion
donante/1. Codigo donacion.txt"

1 : [
0 : "2. Estudio_DonanteSano_882100_formulario.pdf"
1 :
"/home/josebelmonte/Escritorio/pdfs/alogenico/100236/1. Evaluacion
donante/2. Estudio_DonanteSano_882100_formulario.pdf"
```

Figura 18. Resultado de la búsqueda de archivos en el sistema

Una forma de aumentar la fiabilidad de este método fue indicarle a la función encargada de buscar el nombre en el sistema una ruta base sobre la que comenzar la búsqueda, con la intención de indicar aquí el directorio donde se recogen nuestros informes.

Fue entonces cuando encontramos en un foro de *Streamlit* donde se discutía este asunto que descubrimos *Tkinter*, una interfaz de *Python* para examinar directorios y obtener una cadena de texto con su ruta con el método *askdirectory* y así poder indicar como parámetro este directorio base donde se iban a encontrar los archivos.

Aprovechamos para comentar un punto importante de la funcionalidad de la página principal como es la persistencia de datos en el estado de la sesión, pues cada vez que realicemos una acción la página se estaría recargando y necesitamos no perder valores ya cargados, como era este directorio base.

Usando esta nueva interfaz *Tkinter* comprobamos que del mismo modo que podemos examinar con una ventana un directorio es posible realizar la misma acción para examinar archivos y recibir así la ruta completa en la que se encuentra el fichero con el método *askopenfilename* (y *askopenfilenames* para seleccionar múltiples archivos).

Encontrado esto, se descartó la metodología explicada anteriormente que empleaba los botones de subida de archivos de *Streamlit* y aquella posterior búsqueda por nombre en el sistema para pasar a emplear únicamente botones vacías de *Streamlit* que activaran esta nueva funcionalidad con *Tkinter*.

Una ventaja de la carga de ficheros de *Streamlit* frente a *Tkinter* es que sí es persistente con los ficheros mientras que usando *Tkinter* perdíamos el archivo cargado al realizar cualquier otra acción en la página, por lo que se creó una función auxiliar que imitara el comportamiento de este componente de *Streamlit* con funcionalidades aumentadas, empleando *Tkinter* y conservando en la sesión el archivo leído, pudiendo indicar mediante parámetros la posibilidad de cargar uno o varios archivos o si mostrar o no un mensaje de confirmación, entre otros.

Posteriormente se sumó la opción de eliminar el archivo ya examinado, borrándolo tanto de la lista de archivos como de la sesión.

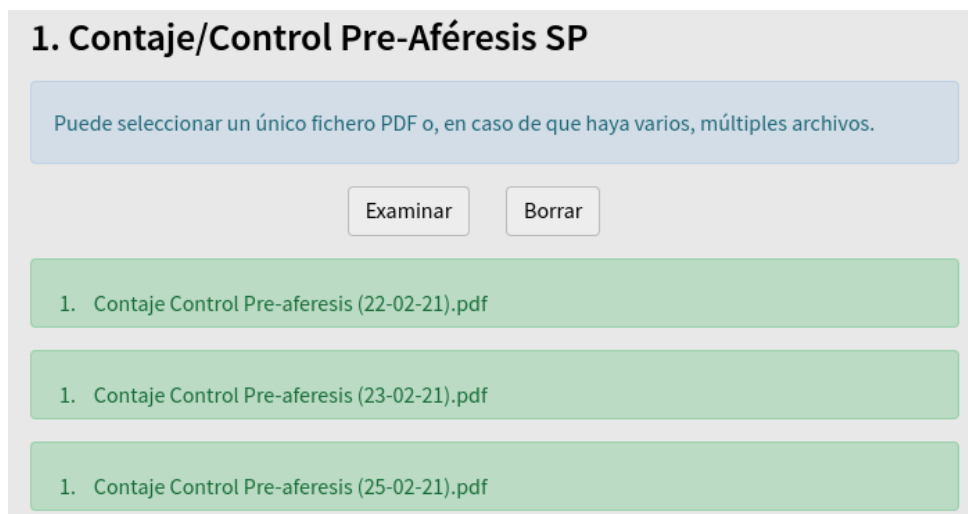


Figura 19. Botones de carga y borrado de archivos

Para desplegar en la página múltiples botones es necesario que éstos reciban en su declaración una *key* única, controlaremos esto con una variable global que pondrá así un identificador diferente a cada botón.

Con la intención de simplificar el código que podía ser algo abrumador ante la necesidad de incluir tantos botones para la carga de los múltiples archivos, incluimos en esta función el código relativo a la creación de dichos botones que era muy repetido, como el título del proceso, la posición sobre la que ubicar los botones, etc.

Para acabar, vamos a ver este nuevo método para incluir botones de cargar y borrado de archivos con sus distintos parámetros de entrada que nos puede aportar distintas funcionalidades en una sola línea de código:

Function signature	
buttons(root, files, key, file_type, name_table, window_title = None, pos = None, pos_delete = None, ext = "pdf", multiple = False, write_title = True, write = True, info = None, etiqueta = None)	
Parameters	
root	Creación única del <i>Tkinter</i> .
files (list)	Lista que contiene las rutas de los archivos leídos, el tipo de archivo y el nombre de la tabla.
key (str)	Identificador con el que se registrará el archivo en el estado de la sesión. <i>Nota: no confundir con el key único necesario en la creación de cada botón de Streamlit.</i>
file_type (int)	Tipo del archivo leído. 0: Etiquetas. 1: Archivos <i>PDF</i> Tipo 1. 2: Archivos <i>PDF</i> Tipo 2. 3: Archivos <i>PDF</i> Tipo 3. -1: Reservado para el código de donación. Este último no se registrará en una tabla de la base de datos al igual que una etiqueta (sí se registrará como identificador de cada tabla).
name_table (str)	Nombre de la table de la base de datos donde se va a guardar su información.
window_title (str)	Será el título de la ventana que aparece para examinar un archivo en el equipo. También escribirá el nombre de la sección encima del botón si <i>write_title</i> es <i>True</i> .
pos	Posición del botón <i>Examinar</i> . Si por defecto es <i>None</i> , lo colocará automáticamente en el centro-izquierda de la pantalla.
pos_delete	Posición del botón <i>Examinar</i> . Si por defecto es <i>None</i> , lo colocará automáticamente en el centro-derecha de la pantalla.
ext (str)	Extensión del archive a examinar. Por defecto, buscará archivos en formato <i>PDF</i> .
multiple (bool)	Si es <i>True</i> , permite seleccionar varios archivos a la vez, en cuyo caso devolvería una lista de archivos.
write_title (bool)	Permite escribir el título de la sección sobre los botones. El motivo para no realizar esta acción en todos los casos es que queremos poner dos pares de botones lado a lado bajo una única sección. (Ejemplo: <i>Contaje 1er Tubo</i> , <i>Contaje 2º Tubo</i>)
write (bool)	Si es <i>True</i> , mostrará un mensaje de confirmación con el título del nombre leído. El motivo para no realizar esta acción en todos los casos es para poder tratar de forma diferente a las etiquetas.
info (str)	Si no es <i>None</i> , mostrará bajo el título de la sección y sobre los botones un mensaje informativo. (Ejemplo, mostrar que es posible la subida múltiple de archivos).
etiqueta (str)	Si no es <i>None</i> , mostrará un mensaje de confirmación de lectura de la etiqueta similar al producido con el parámetro <i>write</i> pero incluyendo en el mensaje la etiqueta contenida en el archivo.

Tabla 1. Parámetros del método buttons

A continuación, explicaremos las páginas disponibles en la aplicación y su funcionalidad.

Inicio

Página de presentación que aparecerá inicialmente al ejecutar la aplicación. En ella mostramos información sobre el presente *TFG*, su autor y las funcionalidades de la aplicación.

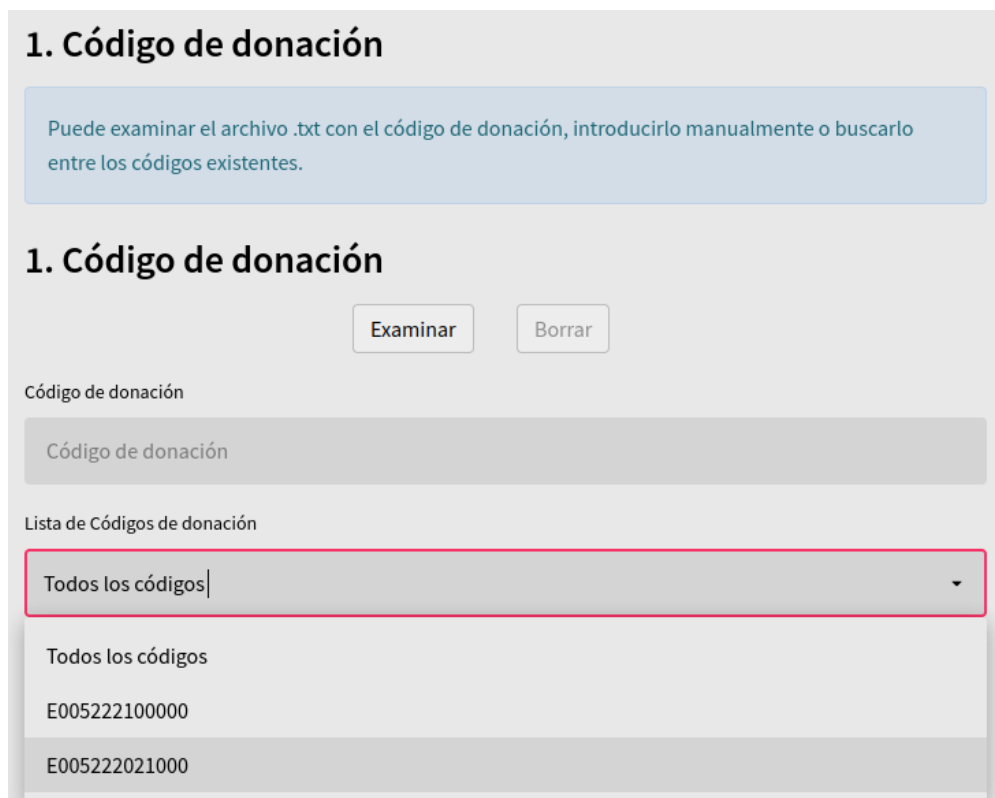
Registrar donación

Página principal donde podremos seleccionar el tipo de trasplante, examinar y leer todos los informes necesarios para registrar una donación y finalmente crear un informe en *Excel* que recoja los datos más relevantes.

Visualizar DB

Funcionalidad que permite la visualización desde la página el contenido de la base de datos. Podemos seleccionar un código de donación examinando un fichero, introduciéndolo manualmente o seleccionándolo de los existentes en el sistema.

La información, que se mostrará como un *DataFrame* en un panel desplegable, es obtenida de la base de datos con las funciones que mencionaremos para la consulta y obtención de datos en la siguiente sección.



1. Código de donación

Puede examinar el archivo .txt con el código de donación, introducirlo manualmente o buscarlo entre los códigos existentes.

1. Código de donación

Examinar Borrar

Código de donación

Código de donación

Lista de Códigos de donación

Todos los códigos

Todos los códigos

E005222100000

E005222021000

Figura 20. Introducir Código de donación

Buscar datos

Funcionalidad cuyo propósito es buscar en las tablas de un paciente (o de todos) los valores de un campo específico. La selección de la donación a buscar es la ya utilizada en el apartado anterior para la visualización de las tablas existentes.

En caso de que la cadena introducida no coincida exactamente con el nombre de ninguna columna, usaremos la función *SequenceMatcher*, que buscará los títulos de las columnas de las tablas de la base de datos que coincidan en cierto rango con la cadena introducida.

Introduce campo a buscar

Leuco

Buscar

No existe la columna: Leuco

Búsqueda relacionada: ['Leucocitos']

Datos encontrados

	Código Donación	Fecha	Tabla	Tipo	Búsqueda
0	E005222021000	22/02/21	B1_Contaje_Control_Pre_Aféresis	Leucocitos	9.52
1	E005222021000	23/02/21	B1_Contaje_Control_Pre_Aféresis	Leucocitos	10.56
2	E005222021000	25/02/21	B1_Contaje_Control_Pre_Aféresis	Leucocitos	10.36
3	E005222021000	26/02/21	B2_Contaje_1er_Tubo	Leucocitos	10.99
4	E005222021000	26/02/21	B3_Contaje_2o_Tubo	Leucocitos	13.02

Figura 21. Caso de búsqueda de un campo no existente

En la figura superior podemos ver que, si buscamos el campo *Leuco* la página nos indicará que no existe ninguna columna que coincida con ese nombre, pero seguidamente buscará coincidencias y nos lo mostrará en un *DataBase* con el *Código Donación*, la *Fecha* y la *Tabla* del informe en el que se encuentra el campo encontrado, con sus datos.

El funcionamiento de este umbral de *SequenceMatcher* consiste en que comenzará siendo un valor elevado que iterando disminuirá su valor hasta que aparezca la primera coincidencia, asegurándonos así de que se casi en todos los casos encontrará información relacionada con la que introduzcamos.

En caso de que sí encuentres un campo que coincida exactamente con la cadena introducida, nos mostrará los resultados obtenidos y adicionalmente repetiremos el mismo método con *SequenceMatcher* para buscar si hubiera más campos con una alta coincidencia. De este modo, si buscamos por ejemplo *Reticulocitos* nos desplegará esa información y seguidamente encontrará *Reticulocitos %*.

Datos encontrados

	Código Donación	Fecha	Tabla	Tipo	Búsqueda
0	E005222100000	26-oct-2018	A2_Estudio_Donante_Sano	Reticulocitos	61500
1	E005222100000	04-dic-2018	A3_Estudio_Pre_Transplante	Reticulocitos	75300
2	E005222021000	12-feb-2021	A3_Estudio_Pre_Transplante	Reticulocitos	124000

Búsqueda relacionada: ['Reticulocitos_porc']

Datos encontrados

	Código Donación	Fecha	Tabla	Tipo	Búsqueda
0	E005222100000	26-oct-2018	A2_Estudio_Donante_Sano	Reticulocitos_porc	1.43
1	E005222100000	04-dic-2018	A3_Estudio_Pre_Transplante	Reticulocitos_porc	2.65
2	E005222021000	12-feb-2021	A3_Estudio_Pre_Transplante	Reticulocitos_porc	2.18

Figura 22. Caso de búsqueda de un campo existente

Cabe mencionar que a pesar de que los títulos de las tablas sufren el formateo anteriormente comentado que realiza una eliminación de símbolos especiales, no es necesario que empleemos esa nomenclatura en el campo de búsqueda, pues se aplicará automáticamente el mismo formato a nuestra cadena.

Introduce campo a buscar

CD4+CD45RA+CD45RO+ (%)

Datos encontrados

	Código Donación	Fecha	Tabla	Tipo	Búsqueda
0	E005222100000	16/03/21	B4_Contaje_Controles_Inte	CD4posCD45RAposCD45ROpos_porc	2.06
1	E005222100000	16/03/21	B51_Contaje_Producto_Fin	CD4posCD45RAposCD45ROpos_porc	3.02
2	E005222021000	26/02/21	B51_Contaje_Producto_Fin	CD4posCD45RAposCD45ROpos_porc	<NA>
3	E005222100000	16/03/21	C21_Contaje_Post_Sepax	CD4posCD45RAposCD45ROpos_porc	1.35
4	E005222100000	16/03/21	C31_Contaje_Frac_Deplec	CD4posCD45RAposCD45ROpos_porc	1.62
5	E005222100000	16/03/21	C41_Contaje_Frac_Pos	CD4posCD45RAposCD45ROpos_porc	0

Figura 23. Ejemplo de introducir una búsqueda sin formatear

4.4. Conexión con la base de datos

Para la creación, inserción y consulta de información en la base de datos, usaremos algunas cuantas funciones recogidas en la clase *db_functions.py*, que previamente se encargará de abrir la conexión.

Una vez hemos examinado nuestros archivos en la página principal y pulsado el botón *Registrar donación*, comenzará la lectura y extracción de información de dichos ficheros para su posterior escritura en la base de datos.

Como detallamos en el apartado anterior, en el momento de registrar un fichero estamos guardando también el nombre de la tabla donde se almacenará su información y el tipo de archivo: -1 para los códigos de donación, 0 para las etiquetas y en el rango 1-3 para los *PDF*.

Usando como parámetro el nombre, crearemos la tabla con el método *create_table()*, que creará una tabla vacía con dos columnas, *Codigo_Donacion* y *Fecha*.

Hemos repetido en algunas ocasiones que en las tablas usaremos como identificador el código de donación y, sin embargo, registraremos también junto a él la fecha de creación del archivo a tratar debido a que puede darse que de un mismo proceso de la terapia celular se encuentren varios informes tomados en distintos días y sea necesario el registro de todos ellos.

Alternativamente, usaremos otro método alternativo al anterior, omitiendo el campo *Fecha*. Esto es así para el registro de etiquetas, pues únicamente tendremos uno o varios códigos en un archivo *TXT* sin fecha alguna.

Una vez creadas las tablas vacías, se registrarán inicialmente los campos *Codigo_Donacion* y *Fecha* de cada una (o únicamente *Codigo_Donacion*), pues estos valores se obtienen de otros métodos auxiliares para leer campos específicos y no de los métodos generales para la extracción completa de la información.

Luego, una vez tenemos los datos recogidos en dos listas *type_info* y *data_info*, procedemos a crear la nueva columna si ésta no existía previamente y a introducir los datos.

Puesto que los nombres de las columnas no pueden contener caracteres especiales realizamos una conversión a base de eliminación de tildes, son cambiados los espacios por '_' y reemplazados algunos símbolos por cadenas significativas. Veamos algunos ejemplos:

Original	Nombre de la columna
CD34+ (Millón/ml)	CD34pos_Millon_ml
CD3+ (%)	CD3pos_porc
CD4+CD45RA+CD45RO+ (%)	CD4posCD45RAposCD45ROpos_porc
Linfocitos (L)	Linfocitos_L
FEV1/CVF	FEV1_CVF
Fenotipo erotrocitario (Rh+Kell)	Fenotipo_erotrocitario_RhposKell
Concentración hemoglobina corp. media	Concentracion_hemoglobina_corp_media

Tabla 2. Ejemplo de formato del nombre de las columnas

En caso de que intentemos introducir una columna que ya exista, el modo de proceder será comprobar si su valor es el mismo al que tratamos de introducir, tratándose de un duplicado que podemos omitir o, en caso de que sea un valor diferente, añadiremos una '_' al final del título de la columna y registraremos la nueva información.

Inicialmente, esta acción se realizaba en código *SQL* dentro de las funciones del manejo de la base de datos, pero fue movido a la clase *pdf_read.py*, pues es más sencillo el manejo de cadenas directamente en el momento de extracción y formateo de la información.

Para el resto de funcionalidades de consulta de datos en la aplicación web, contamos con funciones que nos permiten obtener los nombres de las tablas existentes en la base de datos, los nombres de las columnas de una tabla y obtener sus datos.

Puede darse el caso que estemos buscando una tabla o una columna que no existe o aún no ha sido creada, por eso es importante realizar cierta búsqueda con un *SELECT* dentro de un bloque *try-except* de *Python* para evitar una parada de la ejecución del programa.



Figura 24. Tablas creadas en la base de datos

	Codigo_Donacion	Fecha	Hematies	Hemoglobina	Hematocrito	Volumen_corpuscular_medio	Hemoglobina_corpuscular_media
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	E005222100000	15/03/21	4.5	12.2	37.7	84.2	27.2
2	E005222021000	22/02/21	4.6	12.6	36.5	79.0	27.3
3	E005222021000	23/02/21	4.8	13.1	38.1	79.5	27.3
4	E005222021000	25/02/21	4.8	13.3	38.3	79.1	27.5

Figura 25. Ejemplo de una tabla de un fichero

	Codigo_Donacion	Etiqueta_0	Etiqueta_1
	Filtro	Filtro	Filtro
1	E005222021000	S17461A0	S17461B0

Figura 26. Ejemplo de una tabla de etiquetas

4.5. Escritura de las plantillas Excel

Tras un visionado de los informes en *Excel* que podemos encontrar al final de cada etapa de la terapia celular, recogiendo los datos más relevantes de los informes del proceso, incorporamos la funcionalidad para realizar automáticamente presionando un botón la consulta de la base de datos y su posterior escritura en las plantillas para cada fase proporcionadas por el *Servicio de Hematología*.

Llevamos a cabo esta tarea incorporando *openpyxl* [15], una librería de *Python* para la lectura y escritura de archivos *Excel* *xlsx/xlsm*.

El nombre del informe creado será *Informe_Proceso_CodigoDonacion.xlsx* y por defecto se creará en la misma ubicación del proyecto donde se encuentra la base de datos.

Como explicamos en la sección de *Desarrollo de la aplicación web*, en la página *Registrar donación* de nuestra *app* inicialmente podremos seleccionar un directorio con el método *askdirectory* del mismo modo que empleamos *askopenfilename* para examinar los distintos archivos. Es una acción opcional, pero esto hará que, si indicamos el directorio donde se recogen los informes para la donación que queremos registrar, automáticamente abrirá el buscador en esa ubicación cada vez que queramos examinar un archivo, agilizando el proceso.

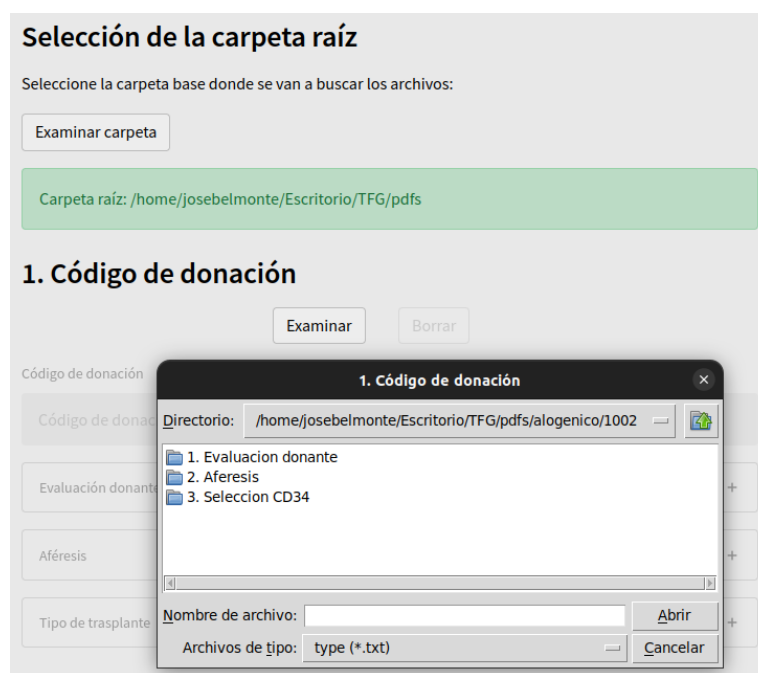


Figura 27. Selección de la carpeta raíz

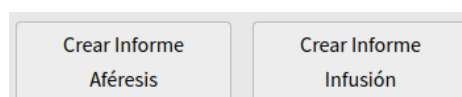


Figura 28. Botones para la creación de los informes

Otro motivo para incluir esta funcionalidad opcional consiste en que si hemos indicado en la página esta localización donde se recogen los informes para el caso a tratar, será en este directorio donde se encontrarán los informes *Excel* creados.

4.6. Contenedores Docker

Docker es una herramienta de código abierto diseñada para crear, ejecutar y desplegar aplicaciones dentro de contenedores de *software*, permitiendo a los desarrolladores empaquetar su aplicación con todas las librerías o dependencias que necesite y desplegarlo comprimido en un único paquete. Además, en esta sección veremos y usaremos *Docker-compose*, que es a su vez una herramienta para la definición y ejecución de aplicaciones en múltiples contenedores *Docker*. Emplearemos un fichero *YAML* para realizar la configuración de los diferentes servicios de la aplicación que finalmente podremos crearlos y lanzarlos con un único comando.

En primer lugar, usaremos la herramienta *Pipenv* [16] que se encarga de crear y manejar un entorno virtual con el que podremos agregar paquetes desde un archivo *Pipfile* y generar el *Pipfile.lock* que necesitamos para construir el fichero *requirements.txt*, el cual contiene estas librerías y dependencias necesarias para nuestro *Front-end*.

Una vez tenemos los requisitos listos, creamos un *Dockerfile* que construiremos con el siguiente comando: `$ sudo Docker Build -t app:v1.0 .`

```
1 FROM python:3.10
2 WORKDIR /app
3 COPY requirements.txt ./requirements.txt
4 RUN pip install -r requirements.txt
5 EXPOSE 8501
6 COPY . /app
7 ENTRYPOINT ["streamlit", "run"]
8 CMD ["app.py"]
```

Figura 29. Dockerfile Front-end

En cambio, el *Back-end* no requiere la construcción de su propio *Dockerfile*, pues parte de una imagen *Docker* ya construida, que invocaremos desde nuestro *Docker-compose* a continuación. Finalmente, se elabora un archivo *YALM* para la construcción de un *Docker-compose* que conecte dos imágenes, una para el *Front-end* mediante *Streamlit* y otra para el *Back-end* mediante *SQLite3*, conectando el primero al puerto que utiliza *Streamlit* 8501 y marcando su dependencia con la segunda imagen *Docker*.

```
1 version: "3.7"
2
3 services:
4   streamlit:
5     image: myapp:v1.0
6     container_name: streamlit_container
7     ports:
8       - 8501:8501
9     volumes:
10      - ./:/app/
11      - ./db:/db/
12     depends_on:
13       - sqlite3
14   sqlite3:
15     image: nouchka/sqlite3:latest
16     container_name: sqlite3_container
17     stdin_open: true
18     tty: true
19     volumes:
20      - ./db:/db/
```

Figura 30. Docker-compose.yaml

4.7. Resultado: Prueba de concepto

Tras el desarrollo de los apartados previos, ofrezco como resultado una prueba de concepto que entrega un entorno funcional, compuesto de un *Front-end* con un entorno sencillo e intuitivo para su manejo por el personal sanitario, un *Back-end* con una base de datos relacional para almacenar la información registrada y distintas funcionalidades como los algoritmos de extracción de datos en los informes o la escritura de los informes *Excel* finales, entre otros. Además, estos componentes quedan recogidos en contenedores *Docker* haciendo la aplicación portable, pues es uno de los requisitos de este proyecto.

El proyecto se encuentra disponible en *GitHub* alojado en el siguiente repositorio, con unos casos completos de ejemplo registrados en una base de datos:

<https://github.com/JBelmontte/TFG.git>

(Nota: los códigos de donación de los casos introducidos en la base de datos son ficticios)

En esta sección expongo unas imágenes de la aplicación en funcionamiento y mostraré el registro de una donación completa.



Figura 31. Página Inicio

Registrar donación

Selección de la carpeta raíz

Seleccione la carpeta base donde se van a buscar los archivos:

Examinar carpeta

Carpeta raíz: /home/josebelmonte/Escritorio/TFG/pdfs/alogenico

1. Código de donación

Examinar Borrar

Código de donación

E005222100000

Evaluación donante +

Aféresis +

Tipo de trasplante +

Registrar donación

Crear Informe Aféresis Crear Informe Infusión

Figura 32. Página Registrar donación

Evaluación donante

2. Estudio pre-trasplante

Tipo de trasplante

☒ Alogénico
☐ Autólogo

Estudio donante sano.pdf Estudio pre-trasplante.pdf

Examinar Borrar Examinar Borrar

2. Estudio donante sano 5431181 formulario.pdf

3. Estudio pre-trasplante 5416380 formulario.pdf

Figura 33. Sección Evaluación donante para un trasplante alogénico

Aféresis

1. Contaje/Control Pre-Aféresis SP

Puede seleccionar un único fichero PDF o, en caso de que haya varios, múltiples archivos.

Examinar
Borrar

1. Contaje Control Pre-Aferesis SP.pdf

2 y 3. Contajes Tubos SP Control Pre-Aféresis

Contaje 1er Tubo SP.pdf
Contaje 2o Tubo SP.pdf

Examinar
Borrar
Examinar
Borrar

2. Contaje 1ºTubo SP.pdf

3. Contaje 2ºTubo SP.pdf

☒ ¿Se han realizado Contajes de controles intermedios de la bolsa de aféresis?

4. Contaje controles intermedios de la bolsa de aféresis

Examinar
Borrar

4. Contaje MitadProceso BolsaAferesis.pdf

5.1. Contaje Bolsa Aféresis - Producto Final

Examinar
Borrar

5. Contaje P.Final BolsaAferesis.pdf

5.2. Cultivo Microbiológico - Producto Final

Examinar
Borrar

5. CultivoMicro P.Final BolsaAferesis.pdf

5.3. Etiqueta Aféresis

Examinar
Borrar

Etiqueta Aféresis: S1860400

☐ Congelación

Figura 34. Sección Aféresis

Tipo de trasplante

Tipo

Selección CD34+

1. Etiqueta producto inicial

ExaminarBorrar

Etiqueta producto inicial: S2600400

2.1. Contaje Post-Sepax

ExaminarBorrar

2. Contajes Post-Sepax.pdf

2.2. Cultivo Microbiológico Post-Sepax

ExaminarBorrar

2. CultivoMicro Post-Sepax.pdf

3.1. Contaje Fracción Deplecionada

ExaminarBorrar

3. Contajes Frac.Deplec.pdf

3.2. Cultivo Microbiológico Fracción Deplecionada

ExaminarBorrar

3. CultivoMicro Frac.Deplec..pdf

4.1. Contaje Fracción Positiva

ExaminarBorrar

4. Contajes FRAC. POSITIVA.pdf

4.2. Cultivo Microbiológico Fracción Positiva

ExaminarBorrar

4. CultivoMicro Frac. Positiva.pdf

5. Etiqueta Infusión

ExaminarBorrar

Etiqueta Infusión: S2847400

Figura 35. Sección Selección CD34+

Tipo de trasplante

Tipo

Descongelación y Lavado

1.1. Contaje Control NUNC

ExaminarBorrar

1. MuestraDescongelacion NUNC (08-03-21).pdf

1.2. Cultivo Microbiológico NUNC

ExaminarBorrar

1. CultivoMicro_NUNC.pdf

2. Descongelación

Puede seleccionar un único fichero PDF o, en caso de que haya varios, múltiples archivos.

ExaminarBorrar

2. Descong_A0_SinLav (30-03-21).pdf

2. Descong_B0_SinLav (30-03-21).pdf

3.1. Lavado

Puede seleccionar un único fichero PDF o, en caso de que haya varios, múltiples archivos.

ExaminarBorrar

3. Descong_A0_Lav (30-03-21).pdf

3. Descong_B0_Lav (30-03-21).pdf

3.2. Etiquetas de Descongelación y Lavado

ExaminarBorrar

Etiqueta/s Descongelación y Lavado: S17461A0 S17461B0

Figura 36. Sección Descongelación y Lavado

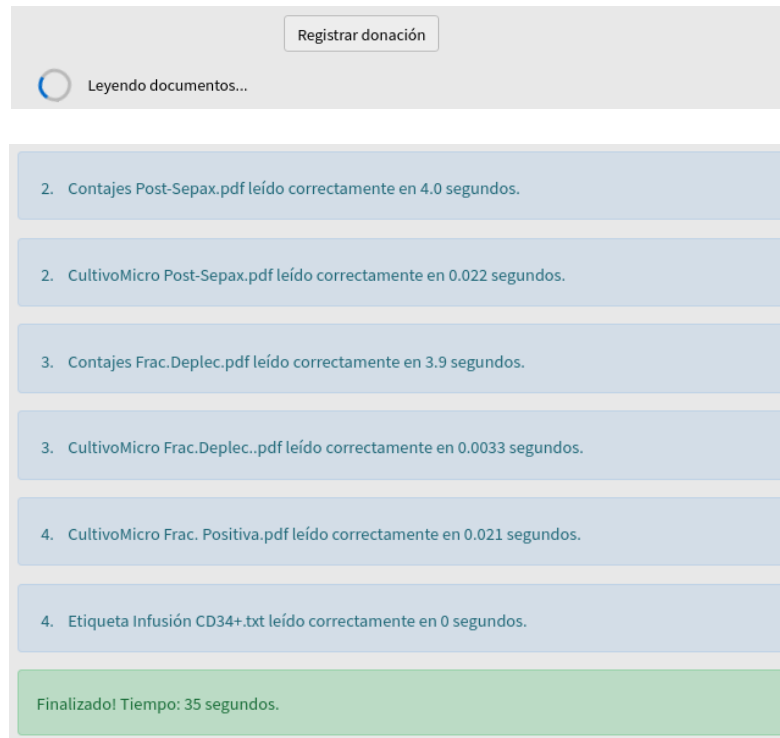


Figura 37. Tiempo de lectura

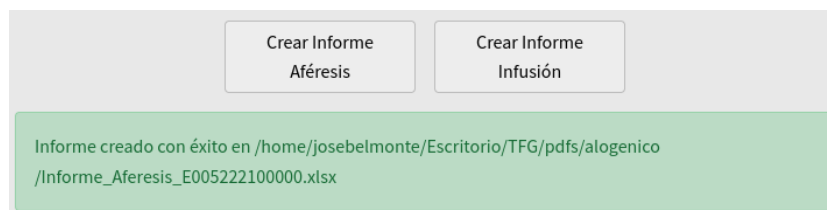


Figura 38. Botón Crear Informe



Figura 39. Mensajes de error

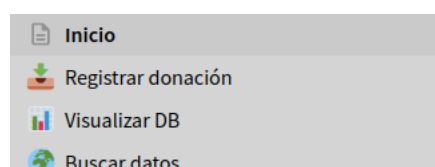


Figura 40. Barra lateral

Visualizar DB

1. Código de donación

Puede examinar el archivo .txt con el código de donación, introducirlo manualmente o buscarlo entre los códigos existentes.

1. Código de donación

Examinar

Borrar

Código de donación

Código de donación

Lista de Códigos de donación

Todos los códigos

Estudio Donante Sano

	Codigo_Donacion	Fecha	Motivo_de_Ingreso	Parentesco
0	E005222100000	26-oct-2018	Evaluación de donante de progenitores hematopoyéticos.	Hermana

Estudio Pre Transplante

Contaje Control Pre Aféresis

	Codigo_Donacion	Fecha	Hematies	Hemoglobina	Hematocrito	Volumen_corpuscular_medio
0	E005222100000	15/03/21	4.5	12.2	37.7	84.2
1	E005222021000	22/02/21	4.6	12.6	36.5	79.0
2	E005222021000	23/02/21	4.8	13.1	38.1	79.5
3	E005222021000	25/02/21	4.8	13.3	38.3	79.1

Contaje 1er Tubo

Contaje 2o Tubo

Figura 41. Página Visualizar DB

Buscar datos

1. Código de donación

Puede examinar el archivo .txt con el código de donación, introducirlo manualmente o buscarlo entre los códigos existentes.

1. Código de donación

Examinar

Borrar

Código de donación

Código de donación

Lista de Códigos de donación

Todos los códigos

Introduce campo a buscar

Reticulocitos

Buscar

Datos encontrados

	Código Donación	Fecha	Tabla	Tipo	Búsqueda
0	E005222100000	26-oct-2018	A2_Estudio_Donante_Sano	Reticulocitos	61500
1	E005222100000	04-dic-2018	A3_Estudio_Pre_Transplante	Reticulocitos	75300
2	E005222021000	12-feb-2021	A3_Estudio_Pre_Transplante	Reticulocitos	124000

Búsqueda relacionada: ['Reticulocitos_porc']

Datos encontrados

Figura 42. Página Buscar datos

5. Conclusiones y vías futuras

Tras haber realizado un testeo de la aplicación para todas las donaciones recogidas en los datos clínicos proporcionados por el *Servicio de Hematología del Hospital Virgen de la Arrixaca*, podemos decir que el *sistema bioinformático* desarrollado tiene un funcionamiento satisfactorio. Las técnicas convencionales llevadas a cabo en los hospitales para el procesamiento manual de estos informes es una tarea lenta y delicada, mientras que delegando esta tarea en manos de este sistema automatizado se puede realizar la tarea de recoger, leer y registrar en una base de datos más de una docena de informes en unos pocos segundos, además de poder consultarlos o realizar búsquedas en ellos de forma eficiente.

La medicina es una ciencia que ha siempre ha existido, al contrario de la informática. Por este motivo se considera necesaria la integración de distintas tecnologías informáticas en las ciencias de la salud, abstrayendo al personal sanitario del procesamiento de grandes cantidades de datos y evitando así posibles errores humanos. Además, con herramientas como la presentada aportamos una ayuda al cambio por la digitalización de los documentos médicos, que hasta ahora eran recogidos en archivadores físicos.

Las posibles vías futuras, además de dar soporte y mantener actualizada la aplicación web sufriendo necesidades futuras, serían:

Aplicar técnicas de seguridad más elevadas con métodos de encriptación de datos utilizando algoritmos que codifiquen la información y solo el personal autorizado pueda acceder a ella.

También se podría contar con un almacenamiento remoto de documentos que cumpla las cláusulas de legalidad, pues recordemos que se trata de informes clínicos.

Aplicar técnicas de *Machine Learning* e *Inteligencia Artificial* sobre los datos extraídos por el sistema, estudiando los parámetros que puedan determinar el que un trasplante resulte de manera satisfactoria. Esto sería posible por el uso de una base de datos relacional que almacena la información recogida en tablas.

Como conclusión quiero resaltar lo aprendido en el desarrollo total de este proyecto, como es la creación de un *Front-end* usando el joven *framework Streamlit*, la conexión con un *Back-end* manejando bases de datos relacionales, la elaboración propia de algoritmos o componentes con un propósito específico, el uso del lenguaje de programación *Python* con sus múltiples librerías e interfaces y el despliegue de aplicaciones dentro de contenedores *Docker*.

Bibliografía

- [1] North Japan Hematology Study Group. "High metabolic heterogeneity on baseline 18FDG-PET/CT scan as a poor prognostic factor for newly diagnosed diffuse large B-cell lymphoma." *Blood advances* vol. 4.10, pp. 2286-2296, 2020.
- [2] OME Cleveland Clinic Orthopaedics. "Implementing a Scientifically Valid, Cost-Effective, and Scalable Data Collection System at Point of Care: The Cleveland Clinic OME Cohort." *The Journal of bone and joint surgery. American volume* vol. 101.5, pp. 458-464, 2019.
- [3] Docker. Available in <https://www.docker.com/>. [Online; accessed April 2022]
- [4] Herrera González, Francisco. "Diseño e implementación de un sistema informático de recogida y gestión de datos para la terapia celular del trasplante de médula ósea." Trabajo Fin de Máster. Máster en Bioinformática, Universidad de Murcia, Julio 2021.
- [5] de la Cruz Martínez, Elena. Trabajo Fin de Máster pendiente de ser defendido. Máster en Bioinformática, Universidad de Murcia, Previsto para septiembre de 2022.
- [6] Django. Available in <https://docs.djangoproject.com/en/4.0/>. [Online; accessed April 2022]
- [7] Kapoor, Saketh et al. "CD34 cells in somatic, regenerative and cancer stem cells: Developmental biology, cell therapy, and omics big data perspective." *Journal of cellular biochemistry* vol. 121.5-6, pp. 3058-3069, 2020.
- [8] Huvarová, Lucie et al. "Washing transplants with Sepax 2 reduces the incidence of side effects associated with autologous transplantation and increases patients' comfort." *Transfusion* vol. 61.8, pp. 2430-2438, 2021.
- [9] SQLite3. Available in <https://docs.python.org/es/3/library/sqlite3.html>. [Online; accessed April 2022]
- [10] Fitz. Available in <https://pymupdf.readthedocs.io/en/latest/module.html>. [Online; accessed April 2022]
- [11] Tabula. Available in <https://tabula-py.readthedocs.io/en/latest/>. [Online; accessed April 2022]
- [12] Streamlit. Available in <https://streamlit.io/>. [Online; accessed April 2022]
- [13] Tkinter. Available in <https://docs.python.org/es/3/library/tkinter.html>. [Online; accessed April 2022]
- [14] Ngrok. Available in <https://ngrok.com/>. [Online; accessed April 2022]
- [15] Openpyxl. Available in <https://openpyxl.readthedocs.io/en/stable/>. [Online; accessed April 2022]
- [16] Pipenv. Available in <https://pipenv-es.readthedocs.io/es/latest/>. [Online; accessed April 2022]