“BookNet Manager”: Gestión remota de inventario para librerías con publicación automática en páginas de venta online

Trabajo de Fin de Máster

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Agosto de 2021

Autor

Víctor Ramos Sánchez

Tutores

Pablo Chamoso Santos

Roberto Carlos Casado Vara

Autorización para la presentación del trabajo

D. Pablo Chamoso Santos y D. Roberto Carlos Casado Vara, personal del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICAN:

Que el trabajo titulado “BookNetManager: Gestión remota de inventario para librerías con publicación automática en páginas de venta online” ha sido realizado por D. Víctor Ramos Sánchez, con DNI 71095070T y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Máster de la Titulación Máster Universitario en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 3 de septiembre de 2021

D. Pablo Chamoso Santos
Dpto. Informática y Automática
Universidad de salamanca

D. Roberto Carlos Casado Vara
Dpto. Informática y Automática
Universidad de salamanca

CASADO VARA ROBERTO CARLOS - 45686237A

Firmado digitalmente por CASADO VARA ROBERTO CARLOS - 45686237A
Fecha: 2021.09.03 17:57:43 +02'00'

Resumen

Este proyecto se centra en la gestión remota de una base de datos de una librería, permitiendo además reflejar los cambios que se realizan en su inventario en páginas de venta de libros como Abebooks o Todocolección. Por ello, el uso del programa al que hace referencia este proyecto estará dirigido exclusivamente a trabajadores de librerías, personas que no han de poseer conocimientos avanzados sobre tecnología.

Se ha desarrollado como una aplicación de escritorio escrita en C#, utilizando el entorno de desarrollo Visual Studio para la implementación del código y diseño de vistas, en concreto, se trata de un proyecto del tipo Windows Forms.

El principal objetivo de esta aplicación es que el inventario de una librería pueda ser gestionado desde cualquier parte, permitiendo visualizar, ordenar, buscar, añadir, modificar o borrar los libros de su inventario. Para ello ha sido necesaria la implementación de un servidor remoto en el que se alojarán los datos del programa. Este servidor utiliza Flask Python para proveer de una API a la aplicación de escritorio que manejarán los trabajadores.

Además, será necesario que el programa se comunique con las APIs de Abebooks y Todocolección para enviar los cambios que se realizan sobre el inventario a estas páginas webs, sin que el trabajador deba preocuparse de realizar esta acción.

Debido a que se trata de una primera versión de la aplicación en la que solo se posee el inventario de una librería, se ha omitido el paso de autenticación inicial, por lo que el programa se conectará automáticamente a la base de datos de la librería Mundus Libri, librería que ha apoyado en el desarrollo de este proyecto durante todo su ciclo de vida.

Por ello, el proceso de implementación del software ha sufrido numerosos cambios durante su desarrollo, atendiendo a los cambios propuestos por los trabajadores de Mundus Libri.

El funcionamiento del sistema se basa en una ventana principal en la que el trabajador puede ver su inventario y realizar acciones sobre él, como por ejemplo ordenarlo según el atributo deseado o realizar búsquedas. Para las labores de edición o modificación se dispondrá de una ventana auxiliar a modo de formulario en la que rellenar los datos de los libros antes de que sean enviados a la red.

El sistema también permitirá exportar e importar todo su inventario mediante un archivo, permitiendo así la migración del sistema a otro servidor si fuera necesario.

Para organizar el trabajo se ha seguido el Proceso Unificado, ya que, durante todo el ciclo de vida del proyecto, se irán realizando iteraciones hasta la obtención de hitos, pasando desde la obtención de requisitos principal, hasta la implementación, la fase de pruebas y la obtención del producto final.

Palabras clave

Libro, librerías, API, Abebooks, Todocolección, inventario, remoto, comunicación, migración.

Abstract

This project focuses on the remote management of a bookshop's database, allowing the changes made to its inventory to be reflected on book sales websites such as Abebooks or Todocolección. For this reason, the use of the program referred to in this project will be aimed exclusively at bookshop workers, people who do not need to have advanced knowledge of technology.

It has been developed as a desktop application written in C#, using the Visual Studio development environment for the implementation of the code and design of views, more precisely, it is a Windows Forms type project.

The main objective of this application is that the inventory of a library can be managed from anywhere, allowing to visualize, order, search, add, modify or delete the books of its inventory. To achieve this, it has been necessary to implement a remote server where the program's data will be hosted. This server uses Flask Python to provide an API to the desktop application that will be used by the workers.

In addition, it will be necessary for the program to communicate with the APIs of Abebooks and Todocolección to send the changes made to the inventory to these websites, without the worker having to worry about carrying out this action.

Since this is a first version of the application in which only the inventory of one library is available, the initial authentication step has been omitted, so the program will automatically connect to the Mundus Libri database, a library that has supported the development of this project throughout its life cycle.

For this reason, the software implementation process has suffered numerous changes during its development, in response to the changes proposed by the Mundus Libri staff.

The operation of the system is based on a main window in which the worker can see his inventory and carry out actions on it, such as sorting it according to the desired attribute or searching. For editing or modification tasks, there will be an auxiliary window in the with a form for fill in the data of the books before they are sent to the network.

The system will also allow to export and import the inventory using a file, allowing the system to be migrated to another server if necessary.

The Unified Process has been followed to organize the work, as iterations will be carried out throughout the life cycle of the project until milestones are reached, from the main requirements gathering, to the implementation, the testing phase and the final product.

Keywords

Book, bookstores, API, Abebooks, Todocolección, inventory, remote, communication, migration.

Índice

Tabla de ilustraciones

Tabla de tablas

1. Introducción

En este documento se detalla toda la información relevante del proceso de desarrollo del proyecto utilizado para la obtención de la aplicación propuesta por el Trabajo de Fin de Máster titulado "BookNet Manager: gestión remota de inventario para librerías con publicación automática en páginas de venta online" perteneciente a la titulación Máster Universitario en Ingeniería Informática.

Este documento y la aplicación que se detalla en él han sido desarrollados por Víctor Ramos Sánchez, durante el curso 2020-2021, y ha contado con la ayuda de sus tutores asignados, Pablo Chamoso Santos y Roberto Carlos Casado Vara, profesionales del Departamento de Informática y Automática de la Universidad de Salamanca.

Hoy en día la informática y la tecnología han conquistado la gran mayoría de sectores, debido a la comodidad que aportan estas herramientas a la hora de realizar distintos trabajos, o el aumento de ventas que supone vender productos de forma online. Por ello, muchas empresas han comenzado a informatizar sus recursos permitiendo así una mayor agilidad y eficiencia a la hora de llevar a cabo sus tareas.

En concreto, las librerías de todo el mundo se han visto ayudadas por la tecnología debido a la capacidad de almacenar información sobre su inventario y realizar tareas como búsquedas de manera mucho más rápida a como se haría sin estas tecnologías.

Por otro lado, el interés por poder realizar tareas sin la necesidad de estar físicamente en la empresa del trabajador ha crecido exponencialmente gracias a Internet. Esto se traduce en la creación de sistemas remotos que permitan conectarse y trabajar desde cualquier lugar.

Aun así, se sigue buscando que la mayoría de las tareas a realizar por el trabajador se realicen de forma automática, sin que el trabajador deba preocuparse por ellas.

Por ello, y por la importancia de vender productos en la red hoy en día, la principal innovación de este proyecto es que se busca que el trabajador únicamente deba preocuparse de modificar y gestionar su inventario, pudiendo realizar esta tarea de forma remota desde cualquier parte del mundo y realizando una publicación automática en páginas de venta online como Abebooks o Todocolección.

Para detallar proceso seguido en el desarrollo de la aplicación, se ha dividido el documento en las siguientes secciones:

- **Estudio de tecnologías similares:** Se estudiarán otras plataformas existentes en la actualidad de forma que puedan observarse similitudes que comparten con este proyecto.
- **Objetivos del proyecto:** Se detallarán los objetivos funcionales y no funcionales que se esperan del producto final.
- **Técnicas y herramientas:** Se mostrarán las diferentes técnicas metodológicas y herramientas utilizadas a lo largo de todo el ciclo de vida del proyecto.

- **Aspectos relevantes en el desarrollo del proyecto:** Se mostrará la información más importante sobre el proyecto, especialmente respecto a su ciclo de vida y a su funcionamiento.
- **Conclusiones y líneas futuras de trabajo:** Para finalizar, se concluirá con una revisión del cumplimiento de los objetivos del proyecto y un resumen de las futuras líneas de trabajo que podrá seguir el proyecto para obtener un producto final aún más completo.

Además, como complemento a la información detallada en este documento, se han realizado los siguientes seis anexos adjuntos:

- Anexo I: Planificación temporal del proyecto
- Anexo II: Especificación de requisitos software
- Anexo III: Análisis del sistema
- Anexo IV: Diseño del sistema
- Anexo V: Documentación técnica y despliegue
- Anexo VI: Manual de usuario

Gracias a estos anexos, se podrá consultar la información que aparece en las distintas secciones de este documento.

2. Estudio de tecnologías similares

Para el correcto desarrollo del proyecto y agilizar algunos aspectos del mismo, se estudiaron tecnologías similares a los mecanismos que va a realizar el programa en aplicaciones que ya existen en la actualidad. En concreto, se estudió el programa HomeBase, aplicación propia de Abebooks que permite gestionar mediante una aplicación de escritorio el inventario de dicha web.

2.1. Aplicaciones existentes

A continuación, se detallarán las aplicaciones existentes en la actualidad que utilizan tecnologías similares a las que se utilizarán en el desarrollo de este proyecto.

2.1.1. HomeBase

HomeBase es un programa gestor de inventario para la página de venta online Abebooks, permite realizar cambios y subirlos directamente su página web.

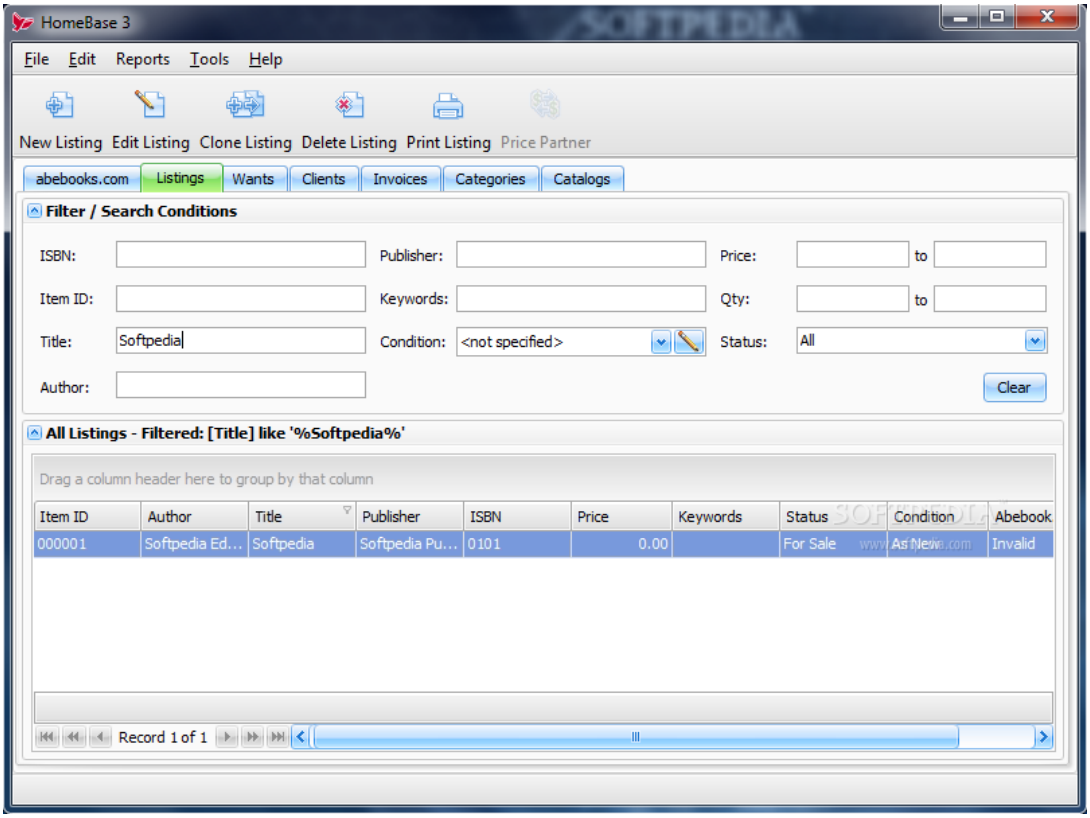

Ilustración 1: Interfaz de HomeBase 3

El uso de HomeBase se dedica exclusivamente a Abebooks, además de gestionar la base de datos que posee una librería en esta página web, permite ajustar precios, buscar u ordenar el inventario.

Este programa utiliza una ventana auxiliar a modo de formulario para rellenar los datos de los libros que se van a añadir o modificar.

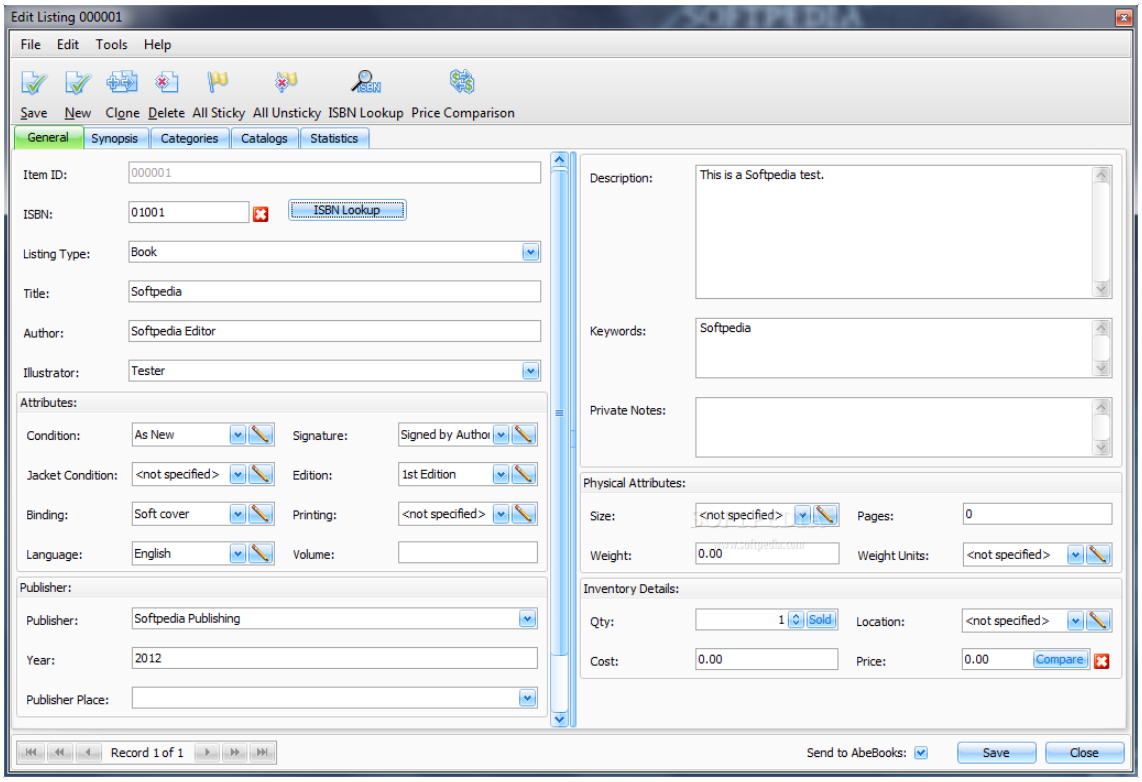

Ilustración 2: Ventana formulario de HomeBase 3

Además, este software permite al usuario exportar o importar el inventario mediante un archivo en un formato propio.

2.2. Características similares

Esta aplicación tiene características muy similares a la aplicación que se desarrolla en este proyecto, ya que la librería Mundus Libri utilizaba HomeBase anteriormente y su deseo en cuanto a BookNet Manager era que se tratara de un programa similar que utilice una base de datos propia y se comunique también con Todocolección, en lugar de solo con Abebooks.

2.2.1. Respecto a la visualización de inventario

Ambas aplicaciones poseen una ventana principal con una tabla en la que visualizar el contenido del inventario, sobre la que es posible realizar búsquedas u ordenar en función a sus atributos todo el inventario.

También posee una barra de herramientas superior mediante la cual será posible realizar otras tareas como importar/exportar, o abrir una ventana formulario para añadir o modificar un libro de la tabla, de la misma forma que ha sido implementado en BookNet Manager.

2.2.2. Respecto a la publicación de inventario

HomeBase publica automáticamente las acciones realizadas por el usuario en su página web, actualizando así el inventario del usuario en Abebooks. Este funcionamiento se ve reflejado en BookNet Manager, enviando esta información a Abebooks y Todocolección a través de sus APIs.

3. Objetivos del proyecto

A continuación, se detallarán los objetivos del proyecto, estos objetivos se encuentran detallados en el ANEXO II del proyecto.

El objetivo principal es el de crear una aplicación capaz **de gestionar de forma remota una base de datos permitiendo que estos cambios se vean reflejados de forma automática en páginas de venta online**, aportando velocidad y facilidad de uso a sus usuarios.

3.1. Objetivos funcionales

Los objetivos funcionales del proyecto son los objetivos que se cumplen con el funcionamiento que se espera de la aplicación.

A continuación, se muestran los distintos objetivos funcionales identificados.

- **Permitir una gestión remota de inventario:** Para cumplir con el objetivo principal de la aplicación será necesario que esté provista de una base de datos remota en la que pueda realizar cambios desde cualquier lugar.
- **Realizar un envío automático de información a la red:** Para cumplir con el otro objetivo principal de este proyecto, el sistema deberá enviar de forma automática la información sobre los cambios realizados a otros sistemas como Todocolección o Abebooks.
- **Permitir realizar migraciones de inventario entre servidores:** Debido a la posible avería del servidor remoto u otros posibles accidentes, se deberá proveer de un sistema capaz de realizar una migración de los datos, exportando e importando desde la aplicación desarrollada.
- **Permitir realizar búsquedas y filtrado de datos:** Para una mayor comodidad de uso, se implementarán sistemas de búsqueda y ordenación del inventario visualizado por el usuario.

Con el cumplimiento de todos los requisitos listados, el producto final será tan completo como se espera, permitiendo al usuario gestionar su base de datos de manera remota y pudiendo olvidarse de la publicación de estos cambios en páginas de venta online.

3.2. Objetivos no funcionales

Además de los requisitos funcionales, el programa deberá cumplir otros objetivos referentes a la forma en que trabaja.

A continuación, se detallan estos objetivos:

- **Integridad de los datos:** Los datos que maneja la aplicación deben ser correctos, mostrando un inventario válido y que no pierda su integridad a la hora de realizar modificaciones o enviar información a través de la red.

- **Velocidad de envío de datos:** La velocidad con la que el sistema envía datos a través de la red es un factor importante en la eficiencia de esta aplicación, por lo que se intentará agilizar en la medida de lo posible estas operaciones.
- **Facilidad de uso:** El sistema deberá proveer una interfaz intuitiva, así como un flujo de trabajo cómodo, de forma que los usuarios no necesiten un gran entrenamiento para sacar todo el partido a la herramienta.

Con el cumplimiento de estos objetivos, el funcionamiento de la aplicación será el que se espera, los datos manejados son correctos y los usuarios de la aplicación podrán realizar sus tareas de una forma rápida y cómoda.

4. Técnicas y herramientas

A continuación, se detallarán las herramientas y técnicas empleadas para el desarrollo del proyecto.

4.1. Técnicas

Las siguientes técnicas metodológicas han sido utilizadas para el desarrollo del proyecto.

4.1.1. Programación orientada a objetos

Para el desarrollo de todas las funcionalidades de la aplicación se ha elegido como lenguaje de programación C#, este lenguaje está basado en programación orientada a objetos, permitiendo así realizar un código basado en clases (objetos) con diferentes módulos y permitiendo la comunicación entre los objetos que componen el sistema.

4.1.2. Comunicación mediante APIs

Para la correcta comunicación de la aplicación con diferentes sistemas, se ha llevado a cabo una comunicación con ellos mediante sus APIs.

Se realizan llamadas HTTP a los distintos sistemas, utilizando las credenciales propias de cada uno y realizando el envío y la recepción de información mediante formatos JSON y XML.

4.1.3. Control de versiones

Para llevar un correcto control sobre los cambios que se han ido realizando en el código fuente del proyecto, se utilizará la técnica de control de versiones. En concreto mediante GitHub, de esta forma el desarrollador podrá obtener el proyecto con los últimos cambios realizados en cualquier momento y crear distintas ramas.

Estas ramas permiten al desarrollador crear una rama derivada de la principal, realizar cambios en ella y, en el momento que se acepten esos cambios, fusionarlos con la rama de la que derivó.

4.1.4. Estimación de costes basada en puntos de casos de uso

Para estimar el coste en duración del proyecto, se ha utilizado la técnica basada en los puntos de casos de uso del proyecto.

Esta técnica consiste en:

- Identificar la complejidad de los casos de uso una vez definidos en función al número de transacciones que el usuario realiza con el sistema para completarlos.
- Determinar la influencia de los factores técnicos
- Determinar la influencia de los factores de entorno

Todo esto nos permitirá utilizar una herramienta para determinar el coste en horas-persona del proyecto. Esta estimación ha sido detalla en el ANEXO I del proyecto.

4.2. Herramientas

Las siguientes herramientas se han utilizado para la realización del producto funcional junto a la documentación que lo acompaña.

4.2.1. Visual Studio (Windows Forms)

Este entorno de desarrollo integrado (IDE) ha sido el elegido para la realización de todo el código fuente del proyecto.

Utiliza un sistema de diseño de interfaces cómodo, que permite diseñar de manera sencilla toda la interfaz del programa y separar de manera automática las vistas de sus controladores, siguiendo un patrón de diseño Modelo-Vista-Controlador.

Además, proporcionará un editor de código con distintas herramientas en el que resulta sencillo utilizar bibliotecas propias y de terceros con el objetivo de obtener todo lo necesario para la ejecución funcional del proyecto.

Mediante este IDE se realizará también el instalador de la aplicación de forma que resulte sencillo desplegar el programa en otras máquinas.

4.2.2. Apache server

Apache es un servidor web al que es posible acceder mediante el protocolo HTTP, permite servir páginas web completas, así como otros recursos.

Se ha utilizado para servir las imágenes subidas mediante FTP desde la aplicación, gracias a este servidor puede generarse una URL a la que accedan los sistemas de Abebooks y Todocolección para descargar las imágenes correspondientes a los libros e insertarlas en sus sistemas.

4.2.3. Flask

Flask es un framework escrito en Python que permite crear aplicaciones web con un mínimo número de líneas de código.

Se ha utilizado para crear la API del servidor con la que se comunica la aplicación para obtener y realizar cambios en su inventario remoto.

4.2.4. GitHub Desktop

GitHub es una herramienta utilizada para el control de versiones y la colaboración de diferentes trabajadores en proyectos software.

Proporciona un sistema de ramas mediante el que el usuario puede crear una rama derivada de la rama principal del software y realizar cambios en ella sin afectar a la principal, de esta forma, permite trabajar sobre determinados cambios sin perder la versión anterior del programa y, una vez que se aceptan esos cambios, fusionar la rama derivada con la rama principal.

Ha sido utilizado durante el desarrollo del proyecto para controlar los cambios que se han ido realizando en el código fuente y para disminuir la probabilidad de una pérdida de avances en el desarrollo.

4.2.5. Visual Paradigm

Para la creación de todos los diagramas UML realizados en la documentación del proyecto, se ha utilizado Visual Paradigm, este software proporciona todas las herramientas utilizadas durante el desarrollo de los diagramas de este proyecto.

4.2.6. EZEstimate

EZEstimate es utilizada para estimar el coste de un proyecto software en desarrollo, utiliza el cálculo de coste en función de puntos de caso de uso (UCP), interpretando la complejidad de los casos de uso del proyecto, definidos en el ANEXO II, y los factores técnicos y de entorno, los cuales se definen en el ANEXO I.

Este software se ha utilizado para proporcionar una estimación del coste en horas-persona para este proyecto.

4.2.7. FileZilla

FileZilla es una aplicación FTP libre y de código abierto, permite conectarse a un servidor de forma remota y realizar descargas o envíos de archivos mediante los protocolos FTP, SFTP y FTP sobre SSL/TLS.

Se ha utilizado para gestiona archivos en el servidor remoto, como por ejemplo el código de la API que reside en él.

4.2.8. Microsoft Office Word

Este editor de documentos desarrollado por Microsoft será la principal herramienta a la hora de realizar la documentación del proyecto.

Con él, se aplicará el formato definido por las normas de estilo apropiadas para un documento técnico.

4.2.9. Microsoft Office Project

Este software proporcionado por Microsoft es un programa dedicado a la planificación temporal de proyectos, es utilizado para la planificación de este proyecto y su uso se detalla en el ANEXO I.

5. Aspectos relevantes en el desarrollo del proyecto

Durante el desarrollo del proyecto, se ha utilizado la metodología marcada por el Proceso Unificado, con la que el ciclo de vida del proyecto será dividido en diferentes etapas desde la etapa de planificación y elicitación de requisitos hasta la etapa de pruebas finales.

La información de las etapas del proyecto está ampliamente detallada en el ANEXO I.

A continuación, se detallarán los aspectos más relevantes del ciclo de vida del proyecto y de la implementación del producto final.

5.1. Ciclo de vida

Para la planificación temporal del ciclo de vida del proyecto se ha seguido el Proceso Unificado, dividiendo el proyecto en tareas y subtareas pertenecientes a iteraciones que terminan con la consecución de un hito, estas a su vez estarán divididas en diferentes etapas.

Tras la planificación temporal del proyecto, se obtuvo un tiempo de trabajo de aproximadamente 5 meses, terminando el 30 de junio de 2021.

A continuación, se detallan los aspectos más importantes de cada etapa de esta planificación.

5.1.1. Etapa de planificación y definición de requisitos del proyecto

Esta primera etapa da comienzo con la realización de reuniones con los trabajadores de Mundus Libri y los tutores.

Estas reuniones tienen como fin establecer una estimación de la duración del proyecto, definir los objetivos que deberá cumplir el software una vez completado y comenzar la obtención de los requisitos funcionales y no funcionales del mismo.

5.1.1.1. Planificación temporal

En cuanto a la planificación temporal del proyecto, se mostrarán las tareas identificadas junto a su duración para cada etapa que seguirá el ciclo de vida del proyecto según el Proceso Unificado, entre ellas se encontrarán tareas periódicas, estas tareas se realizan un número de veces a la semana siempre en el mismo día de cada semana.

Las tareas de la etapa de inicio son las siguientes:

Tabla 1: Lista de tareas de la etapa de inicio

Tareas de la etapa de inicio	Iteración	Duración
Reunión con Mundus Libri	1	1 día
Reunión con los tutores	1	1 día
Obtención de requisitos	1	1 día
Reunión con los tutores	2	1 día
Elaboración del calendario de trabajo	2	1 día
Refinamiento de los requisitos	2	1 día

En la etapa de elaboración se identificaron las siguientes tareas:

Tabla 2: Lista de tareas de la etapa de elaboración

Tareas de la etapa de elaboración	Iteración	Duración
Reuniones semanales Mundus Libri (tarea periódica)	1	1 día
Reuniones semanales con los tutores (tarea periódica)	1	1 día
Organización de paquetes	1	1 día
Diagramas de casos de uso	1	1 día
Modelo de dominio	1	1 día
Clases de análisis	1	1 día
Refinamiento de casos de uso	2	1 día
Refinamiento de modelo de dominio y clases de análisis	2	1 día
Diagramas de secuencia y comunicación	2	1 día
Clases de diseño	2	1 día
Reuniones semanales Mundus Libri (tarea periódica)	3	6 días
Reuniones semanales con los tutores (tarea periódica)	3	6 días
Refinamiento de requisitos y casos de uso	3	1 día
Refinamiento de diagramas de secuencia y comunicación	3	1 día
Vista de arquitectura del modelo de análisis	3	1 día
Refinamiento de clases de diseño	3	1 día
Comienzo de implementación de métodos	3	8 días

Una vez completada la etapa de elaboración, se dio paso a la etapa de construcción, más centrada en el diseño e implementación.

Las tareas que la componen son:

Tabla 3: Lista de tareas de la etapa de construcción

Tareas de la etapa de construcción	Iteración	Duración
Reuniones semanales Mundus Libri (tarea periódica)	1	16 días
Reuniones semanales con los tutores (tarea periódica)	1	16 días
Refinamiento de vista arquitectura	1	1 día
Diagramas de estado y actividad	1	1 día
Implementación de métodos	1	15 días
Comienzo de implementación de interfaz gráfica	1	5 días
Pruebas de métodos (tarea periódica)	1	15 días
Reuniones semanales Mundus Libri (tarea periódica)	2	16 días
Reuniones semanales con los tutores (tarea periódica)	2	16 días
Modelo de despliegue	2	1 día
Implementación de métodos	2	10 días
Implementación de interfaz gráfica	2	20 días
Pruebas de métodos (tarea periódica)	2	19 días
Pruebas de interfaz gráfica (tarea periódica)	2	19 días
Reuniones semanales Mundus Libri (tarea periódica)	3	26 días
Reuniones semanales con los tutores (tarea periódica)	3	26 días
Implementación métodos con interfaz gráfica	3	15 días
Comienzo de implementación de servidor remoto	3	15 días
Pruebas métodos con interfaz gráfica (tarea periódica)	3	16 días
Reuniones semanales Mundus Libri (tarea periódica)	4	11 días
Reuniones semanales con los tutores (tarea periódica)	4	11 días
Implementación de servidor remoto	4	10 días
Refinamiento de interfaz gráfica de usuario	4	5 días

Obtención de versión beta	4	5 días
Pruebas servidor remoto (tarea periódica)	4	10 días
Pruebas beta (tarea periódica)	4	5 días

Una vez obtenida una primera versión inicial del producto, se da paso a la etapa de transición, en ella cual se realizarán las últimas pruebas y mejoras del producto final.

Las tareas que componen la etapa de transición son las siguientes:

Tabla 4: Lista de tareas de la etapa de transición

Tareas de la etapa de transición	Iteración	Duración
Comprobación de objetivos	1	1 día
Últimas mejoras y versión final	1	3 días
Últimas pruebas (tarea periódica)	1	3 días

Con la obtención de todas las tareas indicadas y las dependencias existentes entre ellas, se realizó la planificación temporal del proyecto completa.

Estableciendo la fecha de inicio en el 1 de febrero de 2021, se estima gracias este método que la finalización del proyecto tendrá lugar el 30 de junio de 2021.

A continuación, se muestra el diagrama de Gantt del proyecto para cada etapa, se mostrará en varias páginas una misma etapa en caso de ser demasiado grande.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**5 días?**	**lun 01/02/21**	**vie 05/02/21**		
Iteración 1	**3 días?**	**lun 01/02/21**	**mié 03/02/2**		
Modelado de negocio	**2 días?**	**lun 01/02/21**	**mar 02/02/2**		
Reunión con MundusLibri	1 día?	lun 01/02/21	lun 01/02/21		Víctor Ramos Sa
Reunión con los tutores	1 día?	mar 02/02/2	mar 02/02/2	4	Víctor Ramos Sa
Requisitos	**1 día?**	**mié 03/02/2**	**mié 03/02/2**		
Obtención de requisitos	1 día?	mié 03/02/2	mié 03/02/2	5	Víctor Ramos Sa
Fin iteracción 1	0 días	mié 03/02/2	mié 03/02/2	7	
Iteración 2	**2 días?**	**jue 04/02/21**	**vie 05/02/21**	**2**	
Modelado de negocio	**1 día?**	**jue 04/02/21**	**jue 04/02/21**		
Reunión con los tutores	1 día?	jue 04/02/21	jue 04/02/21		Víctor Ramos Sa
Elaboración de calendario de proyecto	1 día?	jue 04/02/21	jue 04/02/21		Víctor Ramos Sa
Requisitos	**1 día?**	**vie 05/02/21**	**vie 05/02/21**		
Refinamiento de requisitos	1 día?	vie 05/02/21	vie 05/02/21	11	Víctor Ramos Sa
Fin iteración 2	0 días	vie 05/02/21	vie 05/02/21	14	

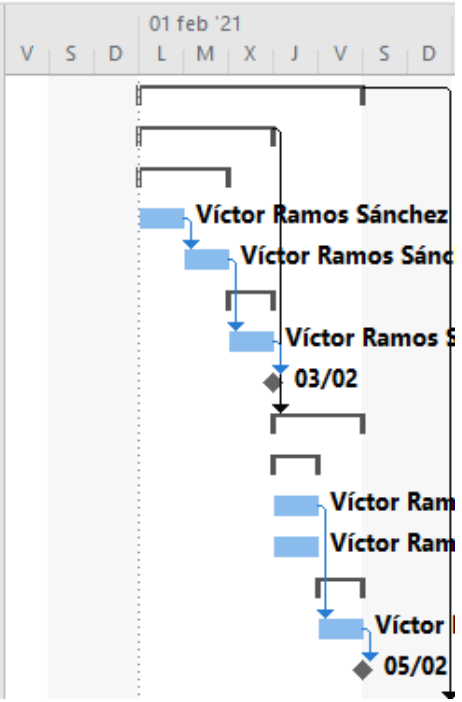

Ilustración 3: Diagrama de Gantt de la etapa de inicio

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**5 días?**	**lun 01/02/21**	**vie 05/02/21**		
Elaboración	**15 días?**	**lun 08/02/21**	**vie 26/02/21**	**1**	
Iteración 1	**2 días?**	**lun 08/02/21**	**mar 09/02/2**		
Modelado de negocio	**2 días**	**lun 08/02/21**	**mar 09/02/2**		
Reuniones semanales MundusLibri	**1 día**	**lun 08/02/21**	**lun 08/02/21**		
Reuniones semanales con los tutores	**1 día**	**mar 09/02/2**	**mar 09/02/2**	**20**	
Requisitos	**1 día?**	**lun 08/02/21**	**lun 08/02/21**		
Organización de paquetes	1 día?	lun 08/02/21	lun 08/02/21		Víctor Ramos Sa
Diagramas CU	1 día?	lun 08/02/21	lun 08/02/21		Víctor Ramos Sa
Análisis	**1 día?**	**mar 09/02/2**	**mar 09/02/2**	**23**	
Modelo de dominio	1 día?	mar 09/02/2	mar 09/02/2		Víctor Ramos Sa
Clases de análisis	1 día?	mar 09/02/2	mar 09/02/2		Víctor Ramos Sa
Fin iteración 1	0 días	mar 09/02/2	mar 09/02/2	26	
Iteración 2	**3 días?**	**mié 10/02/2**	**vie 12/02/21**	**17**	
Requisitos	**1 día?**	**mié 10/02/2**	**mié 10/02/2**		
Refinamiento CU	1 día?	mié 10/02/2	mié 10/02/2		Víctor Ramos Sa
Análisis	**2 días?**	**mié 10/02/2**	**jue 11/02/21**		
Refinamiento modelo de dominio y clases de análisis	1 día?	mié 10/02/21	mié 10/02/21		Víctor Ramos Sánchez[50%]
Diagramas de secuencia y comunicación	1 día?	jue 11/02/21	jue 11/02/21	32;34	Víctor Ramos Sánchez
Diseño	**1 día?**	**vie 12/02/21**	**vie 12/02/21**		
Clases de diseño	1 día?	vie 12/02/21	vie 12/02/21	35	Víctor Ramos Sa
Fin iteración 2	0 días	vie 12/02/21	vie 12/02/21	37	

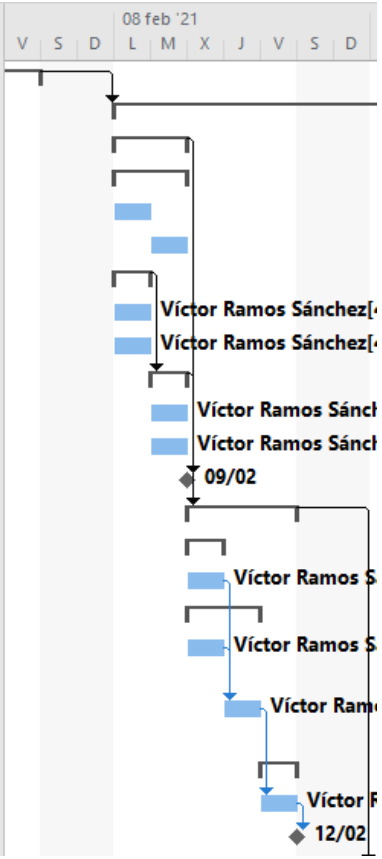

Ilustración 4: Diagrama de Gantt de la etapa de elaboración, iteraciones 1 y 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**5 días?**	**lun 01/02/21**	**vie 05/02/21**		
Elaboración	**15 días?**	**lun 08/02/21**	**vie 26/02/21**	**1**	
Iteración 1	**2 días?**	**lun 08/02/21**	**mar 09/02/2**		
Iteración 2	**3 días?**	**mié 10/02/2**	**vie 12/02/21**	**17**	
Iteración 3	**10 días?**	**lun 15/02/21**	**vie 26/02/21**	**30**	
Modelado de negocio	**7 días**	**lun 15/02/21**	**mar 23/02/2**		
Reuniones semanales MundusLibri	**6 días**	**lun 15/02/21**	**lun 22/02/21**		
Reuniones semanales con los tutores	**6 días**	**mar 16/02/2**	**mar 23/02/2**		
Requisitos	**1 día?**	**lun 15/02/21**	**lun 15/02/21**		
Refinamiento requisitos y CU	1 día?	lun 15/02/21	lun 15/02/21		Víctor Ramos Sa
Análisis	**2 días?**	**lun 15/02/21**	**mar 16/02/2**		
Refinamiento de diagramas de secuencia y comunicación	1 día?	lun 15/02/21	lun 15/02/21		Víctor Ramos Sánchez[40%]
Vista de arquitectura del modelo de análisis	1 día?	mar 16/02/21	mar 16/02/21	50	Víctor Ramos Sánchez[40%]
Diseño	**1 día?**	**mar 16/02/2**	**mar 16/02/2**		
Refinamiento de clases de diseño	1 día?	mar 16/02/21	mar 16/02/21	50	Víctor Ramos Sánchez[40%]
Implementación	**8 días**	**mié 17/02/2**	**vie 26/02/21**		
Comienzo implementación de métodos	8 días	mié 17/02/21	vie 26/02/21	53	Víctor Ramos Sánchez
Fin iteración 3	0 días	vie 26/02/21	vie 26/02/21	55	

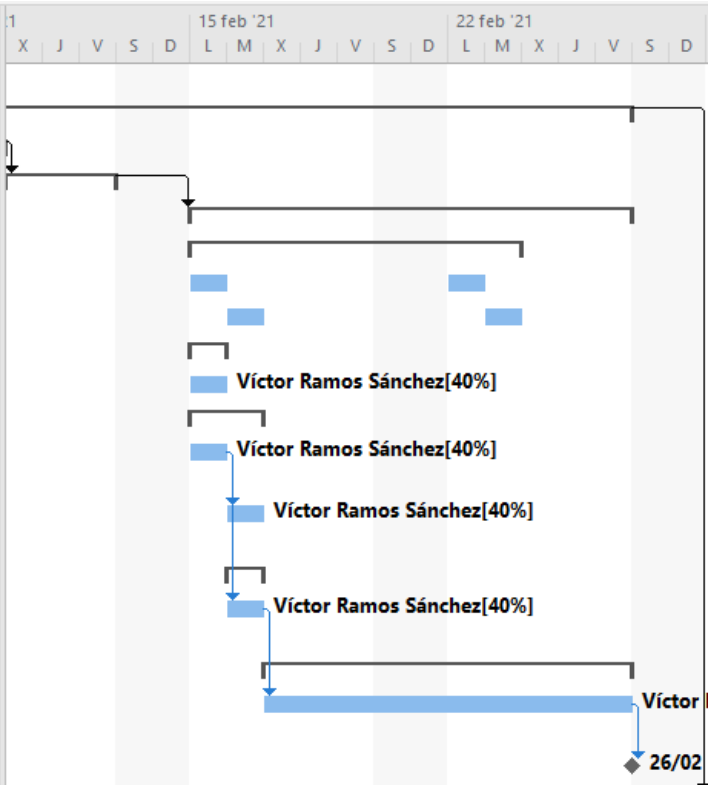

Ilustración 5: Diagrama de Gantt de la etapa de elaboración, iteración 3

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**5 días?**	**lun 01/02/21**	**vie 05/02/21**		
Elaboración	**15 días?**	**lun 08/02/21**	**vie 26/02/21**	**1**	
Construcción	**85 días?**	**lun 01/03/21**	**vie 25/06/21**	**16**	
Iteración 1	**20 días?**	**lun 01/03/21**	**vie 26/03/21**		
Modelado de negocio	**17 días**	**lun 01/03/21**	**mar 23/03/2**		
Reuniones semanales MundusLibri	**16 días**	**lun 01/03/21**	**lun 22/03/21**		
Reuniones semanales con los tutores	**16 días**	**mar 02/03/2**	**mar 23/03/2**		
Análisis	**1 día?**	**lun 01/03/21**	**lun 01/03/21**		
Refinamiento vista de arquitectura	1 día?	lun 01/03/21	lun 01/03/21		Víctor Ramos Sánchez[10%]
Diseño	**1 día?**	**lun 01/03/21**	**lun 01/03/21**		
Diagramas de estado y actividad	1 día?	lun 01/03/21	lun 01/03/21		Víctor Ramos Sa
Implementación	**20 días**	**lun 01/03/21**	**vie 26/03/21**		
Implementación de métodos	15 días	lun 01/03/21	vie 19/03/21		Víctor Ramos Sa
Comienzo implementación GUI	5 días	lun 22/03/21	vie 26/03/21	75	Víctor Ramos Sa
Pruebas	**16 días**	**lun 01/03/21**	**lun 22/03/21**		
Pruebas métodos	**15 días**	**lun 01/03/21**	**vie 19/03/21**		
Fin iteración 1	0 días	vie 26/03/21	vie 26/03/21	76	
Iteración 2	**20 días?**	**lun 29/03/21**	**vie 23/04/21**	**58**	
Modelado de negocio	**17 días**	**lun 29/03/21**	**mar 20/04/2**		
Reuniones semanales MundusLibri	**16 días**	**lun 29/03/21**	**lun 19/04/21**		
Reuniones semanales con los tutores	**16 días**	**mar 30/03/2**	**mar 20/04/2**		
Diseño	**1 día?**	**vie 23/04/21**	**vie 23/04/21**		
Modelo de despliegue	1 día?	vie 23/04/21	vie 23/04/21		Víctor Ramos Sa
Implementación	**20 días**	**lun 29/03/21**	**vie 23/04/21**		
Implementación de métodos	10 días	lun 29/03/21	vie 09/04/21		Víctor Ramos Sa
Implementación GUI	20 días	lun 29/03/21	vie 23/04/21		Víctor Ramos Sa
Pruebas	**20 días**	**lun 29/03/21**	**vie 23/04/21**		
Pruebas métodos	**19 días**	**lun 29/03/21**	**jue 22/04/21**		
Pruebas GUI	**19 días**	**mar 30/03/2**	**vie 23/04/21**		
Fin iteración 2	0 días	vie 23/04/21	vie 23/04/21	99;102	

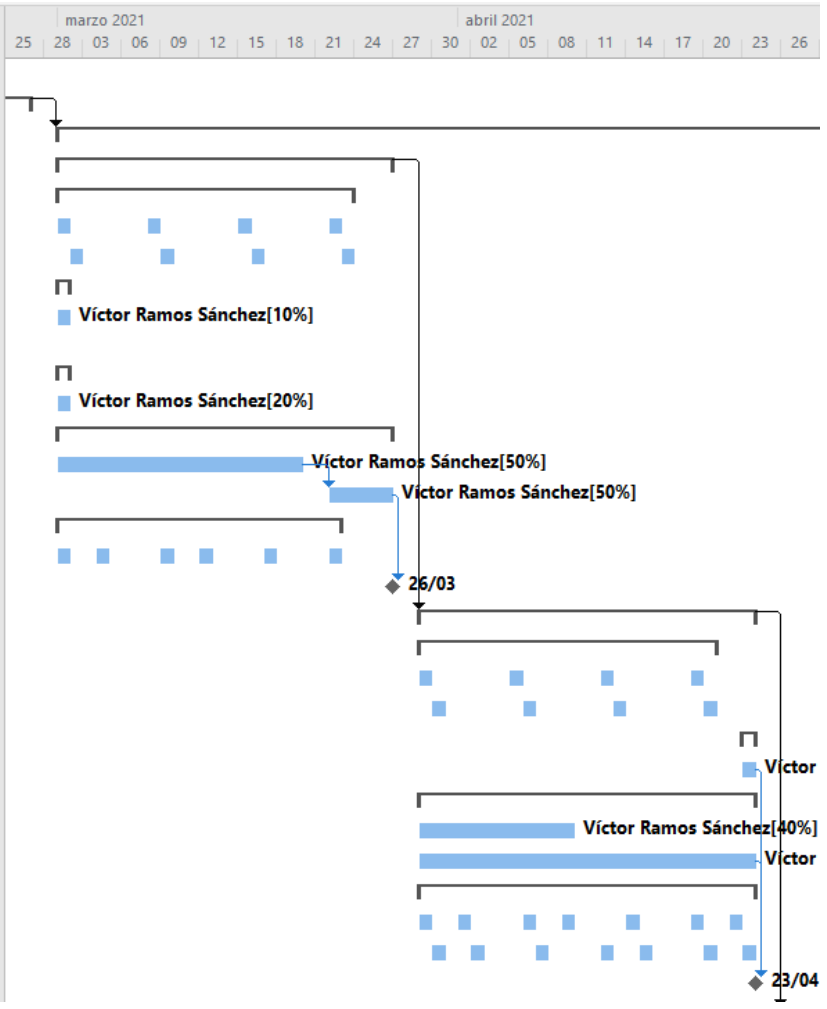

Ilustración 6: Diagrama de Gantt de la etapa de construcción, iteraciones 1 y 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**5 días?**	**lun 01/02/21**	**vie 05/02/21**		
Elaboración	**15 días?**	**lun 08/02/21**	**vie 26/02/21**	**1**	
Construcción	**85 días?**	**lun 01/03/21**	**vie 25/06/21**	**16**	
Iteración 1	**20 días?**	**lun 01/03/21**	**vie 26/03/21**		
Iteración 2	**20 días?**	**lun 29/03/21**	**vie 23/04/21**	**58**	
Iteración 3	**30 días**	**lun 26/04/21**	**vie 04/06/21**	**86**	
Modelado de negocio	**27 días**	**lun 26/04/21**	**mar 01/06/2**		
Reuniones semanales MundusLibri	**26 días**	**lun 26/04/21**	**lun 31/05/21**		
Reuniones semanales con los tutores	**26 días**	**mar 27/04/2**	**mar 01/06/2**		
Implementación	**30 días**	**lun 26/04/21**	**vie 04/06/21**		
Implementación métodos con GUI	15 días	lun 26/04/21	vie 14/05/21		Víctor Ramos Sánchez[75%]
Comienzo implementación servidor remoto	15 días	lun 17/05/21	vie 04/06/21	138	Víctor Ramos Sánchez[75%]
Pruebas	**16 días**	**lun 26/04/21**	**lun 17/05/21**		
Pruebas métodos con GUI	**16 días**	**lun 26/04/21**	**lun 17/05/21**		
Fin iteración 3	0 días	vie 04/06/21	vie 04/06/21	139	
Iteración 4	**15 días**	**lun 07/06/21**	**vie 25/06/21**	**121**	
Modelado de negocio	**12 días**	**lun 07/06/21**	**mar 22/06/2**		
Reuniones semanales MundusLibri	**11 días**	**lun 07/06/21**	**lun 21/06/21**		
Reuniones semanales con los tutores	**11 días**	**mar 08/06/2**	**mar 22/06/2**		
Implementación	**15 días**	**lun 07/06/21**	**vie 25/06/21**		
Implentación servidor remoto	10 días	lun 07/06/21	vie 18/06/21		Víctor Ramos Sá
Refinamiento interfaz	5 días	lun 07/06/21	vie 11/06/21		Víctor Ramos Sá
Obtención de versión beta	5 días	lun 21/06/21	vie 25/06/21	160	Víctor Ramos Sá
Pruebas	**15 días**	**lun 07/06/21**	**vie 25/06/21**		
Pruebas servidor remoto	**10 días**	**lun 07/06/21**	**vie 18/06/21**		
Pruebas beta	**5 días**	**lun 21/06/21**	**vie 25/06/21**		
Fin iteración 4	0 días	vie 25/06/21	vie 25/06/21	162	

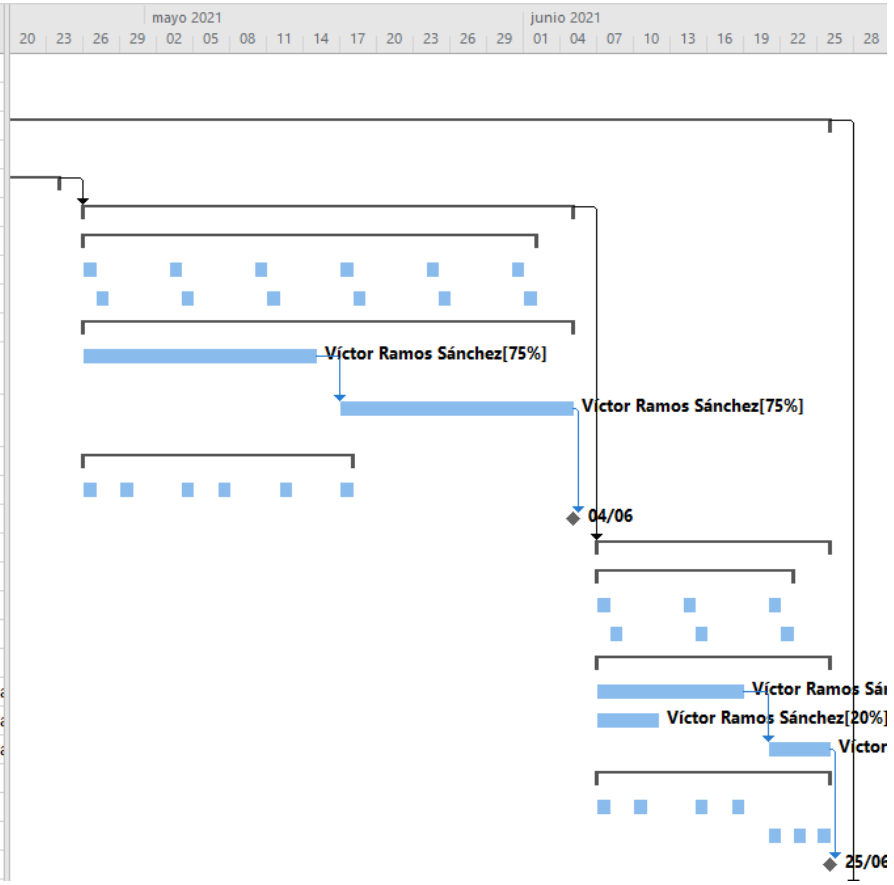

Ilustración 7: Diagrama de Gantt de la etapa de construcción, iteraciones 3 y 4

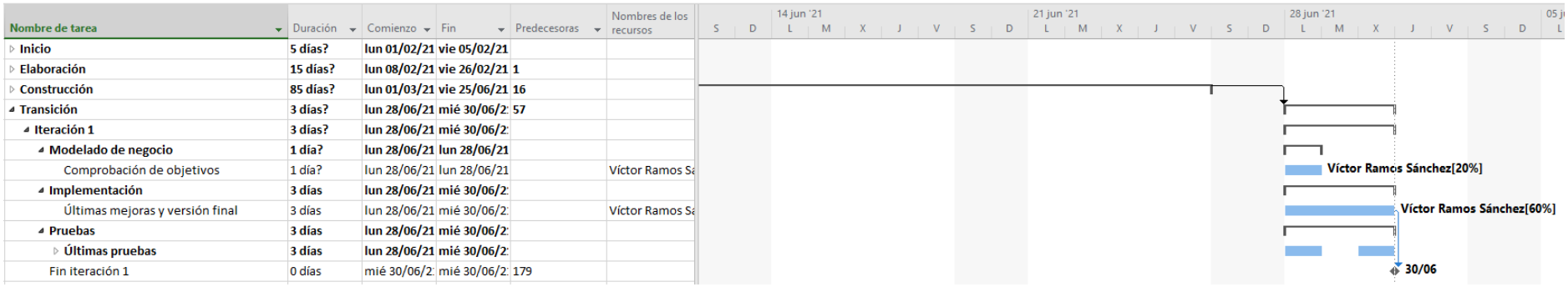

Ilustración 8: Diagrama de Gantt de la etapa de transición

5.1.2. Definición de requisitos

Se determinaron los siguientes objetivos para el proyecto, estos objetivos se encuentran detallados en el apartado dos de este documento "Objetivos del proyecto".

- Gestionar cambios en inventario de forma remota y reflejar cambios en webs de venta de libros.
- Inventario remoto.
- Subir información de inventario a la red.
- Migración de inventario.
- Búsqueda y filtrado.

En cuanto a la obtención de los requisitos de información, dado que la aplicación utiliza un servidor para almacenar sus datos, deberá almacenar información persistente sobre sus libros, así como otra información persistente en el tiempo que tendrá un tiempo de vida de todo el ciclo de vida del proyecto.

En concreto, los requisitos de información obtenidos son los siguientes:

- Información de importación de inventario.
- Información de volcado de inventario.
- Información de formato de archivo.
- Información de autenticación.
- Información de envío de datos.
- Información de búsqueda.
- Información de ordenación.
- Información de libro.
- Información de idiomas.
- Información de secciones.
- Información de estanterías.
- Información de cubiertas.

El aspecto más importante de estas reuniones fue el de obtener los requisitos funcionales del proyecto para después, en la siguiente etapa definirlos en forma de casos de uso, dado que este es el marco de trabajo que seguirá el Proceso Unificado.

La información acerca de los requisitos se encuentra ampliada en el ANEXO II.

5.1.3. Etapa de análisis

Tras obtener los requisitos del proyecto y realizar una estimación temporal del mismo, se da por cerrada la etapa de inicio, dando comienzo a la etapa de elaboración, en esta etapa se realizará el análisis del sistema.

Lo primero que se debió realizar en esta etapa fue terminar los diagramas de casos de uso del sistema. La información de casos de uso y sus diagramas correspondientes se encuentran ampliados en el ANEXO II.

Puede destacarse en las primeras iteraciones de esta etapa la división del sistema organizado en paquetes y la realización del modelo de dominio del proyecto, en el que se detallan las clases participantes en él y sus relaciones.

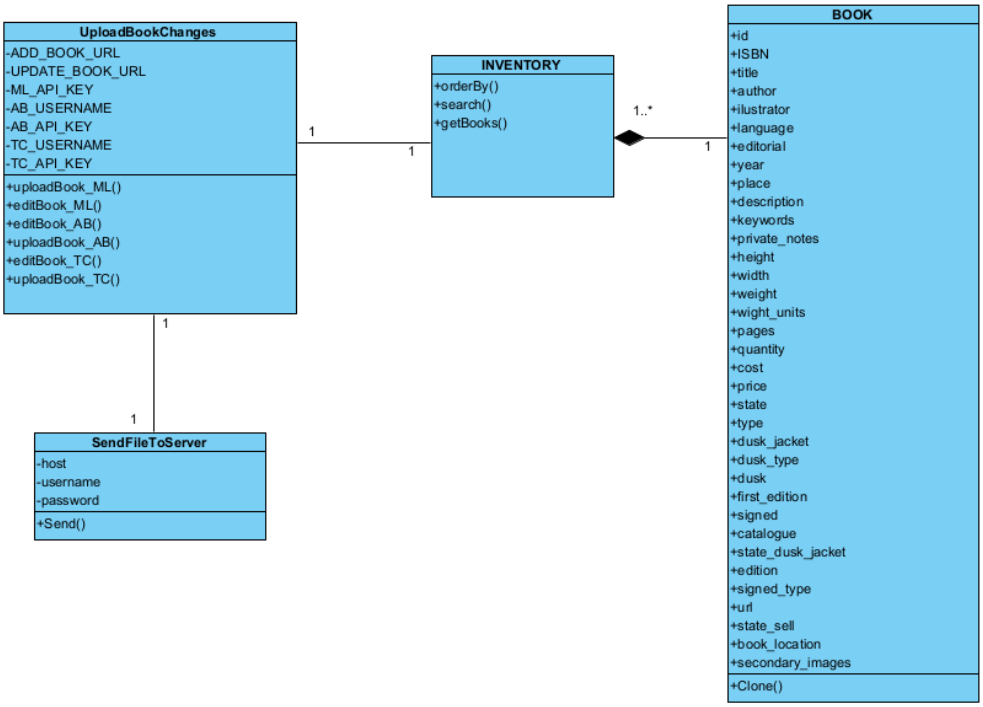

Ilustración 9: Diagrama de clases del modelo de dominio del sistema

En cuanto a la división en paquetes del sistema, puede observarse su estructura en el siguiente diagrama, formado por los paquetes conexión remota, visualización de inventario, migración de inventario y gestión remota de inventario.

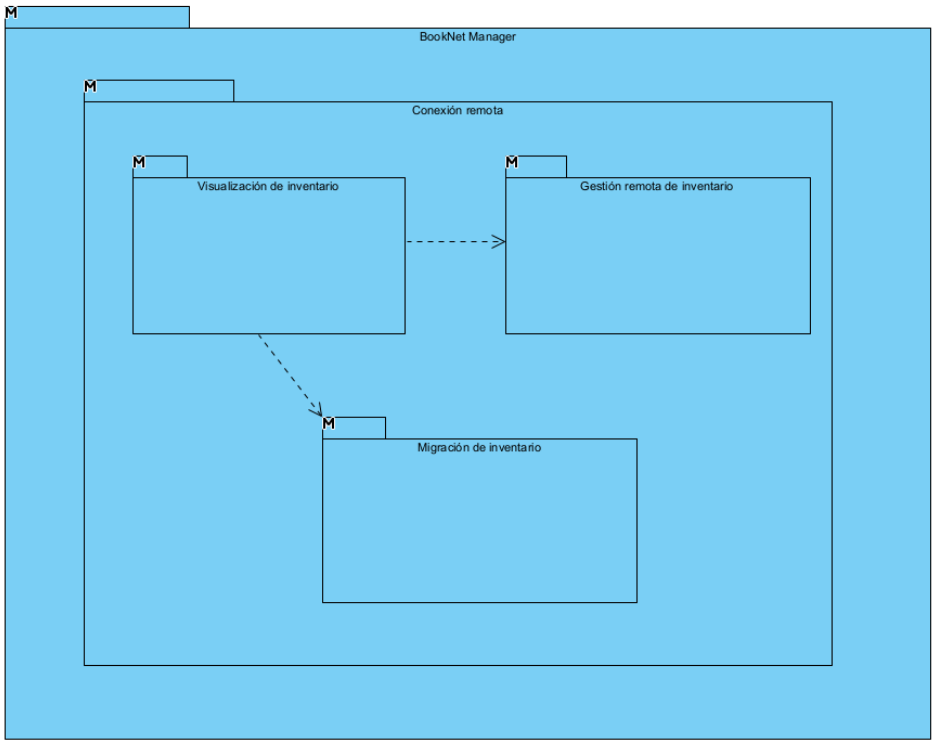

Ilustración 10: Diagrama de paquetes del sistema

Como puede observarse, todos los paquetes se encuentran dentro del paquete de conexión remota, ya que el sistema no realizará ninguna función sin antes haber realizado una conexión con el servidor remoto.

Un aspecto muy importante de esta etapa del ciclo de vida es la realización de los casos de uso mediante diagramas de secuencia, ya que primero se realizaron sin tener en cuenta el patrón de diseño que se utilizará, para después poder refinarlo en función a los elementos que participarán en ellos tras aplicar un patrón de diseño.

Por ello en este documento únicamente se mostrará un diagrama de secuencia general, para más adelante mostrar los diagramas de secuencia refinados. Los diagramas de secuencia iniciales pueden encontrarse en el ANEXO III de este documento.

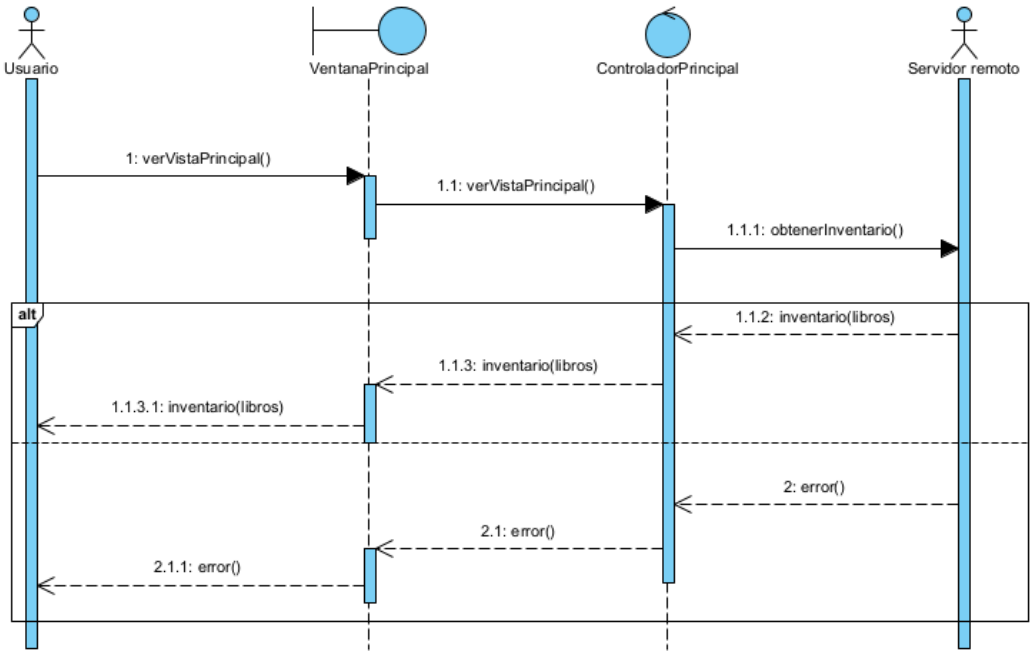

Ilustración 11: Diagrama de secuencia inicial del CU-0003 "Visualizar inventario"

5.1.4. Etapa de diseño

Una vez realizado el análisis del sistema, da comienzo el diseño del mismo.

El diseño elegido para la arquitectura del sistema ha sido el patrón Modelo-Vista-Controlador (MVC), este patrón se basa en tres elementos principales:

- **Modelo:** En esta capa se realizan los cambios de los datos, es el código referente a la actualización y acceso a información.
- **Vista:** Es lo que el usuario ve, todo el código que genera las ventanas con las que el usuario podrá interactuar.
- **Controlador:** Es la capa encargada de comunicar la vista con el modelo, contiene el código capaz de transmitir una petición del usuario al modelo.

El patrón MVC comienza a utilizarse al realizar la primera vista arquitectónica del modelo de diseño, dado que se definieron ya posibles interfaces, los posibles controladores que utilizará el sistema y los modelos participantes en él.

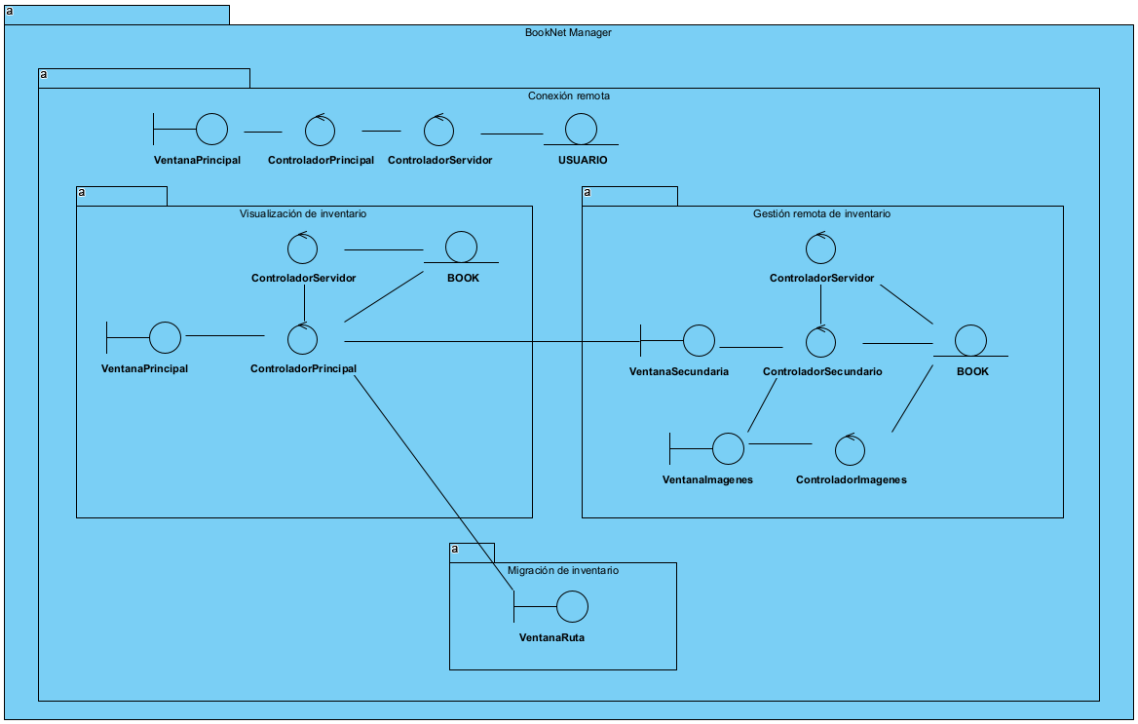

Ilustración 12: Diagrama de la vista arquitectónica del modelo de diseño

Como puede observase, el paquete de conexión remota realizará una comunicación con el servidor para acceder a la base de datos de usuarios.

Por otro lado, puede verse que el paquete de visualización de inventario estará formado por la ventana principal, cuyo controlador será el encargado de dar acceso al resto de paquetes iniciando sus interfaces.

Una vez realizado el análisis del sistema, se continúa el diseño del sistema mediante la utilización del patrón MVC, para ello se realizaron las clases de diseño permitiendo así dividir todo el código en las diferentes capas de este patrón.

Las clases de diseño de la capa vista, controlador y modelo vienen detalladas en el ANEXO IV, por lo que, en lugar de mostrarlas en este documento, se mostrará únicamente los elementos de la capa vista. Más adelante se mostrará el diagrama de la vista arquitectónica del modelo de diseño refinado, ya que en él aparecen todas las clases de diseño y sus relaciones.

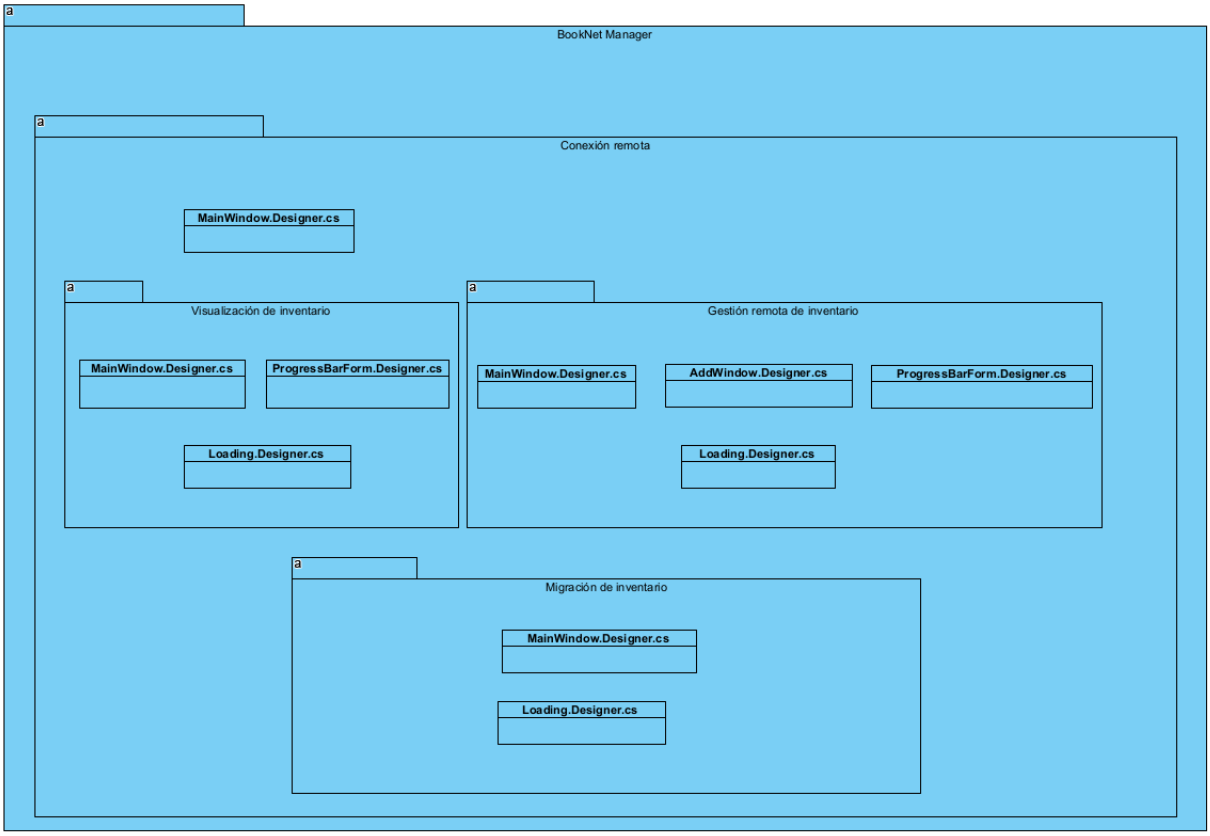

Ilustración 13: Diagrama de clases de diseño de la capa "vista" del sistema

En cuanto a la vista, cabe destacar que el entorno de desarrollo utilizado separa automáticamente la vista del controlador creando un archivo con formato C# para cada capa, los archivos referentes a las vistas terminarán con ".Designer.cs".

Un aspecto también muy importante de esta etapa es que, tras haber realizado las clases de diseño, se comienzan a implementar los primeros algoritmos de los métodos del sistema.

Tras realizar las clases de diseño, se refina la vista de arquitectura del modelo de diseño, este diagrama muestra los elementos pertenecientes a las capas vista, controlador y modelo y cómo se relacionan entre ellos.

Se muestra el diagrama en las siguientes dos páginas debido a su tamaño. Primero se muestra el paquete de visualización de inventario y a continuación, el paquete de gestión remota de inventario.

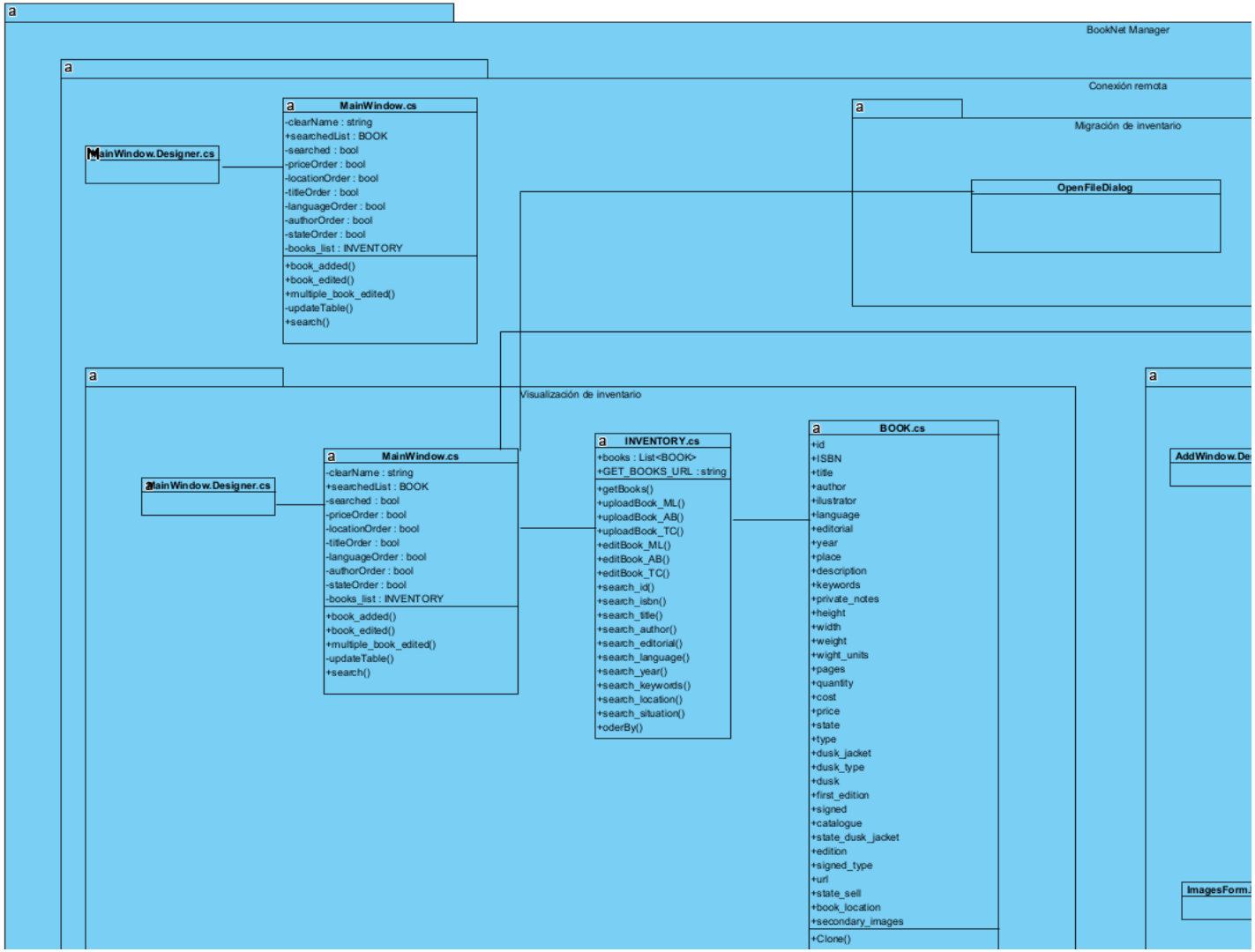

Ilustración 14: Diagrama de la vista arquitectónica del modelo de diseño, visualización de inventario

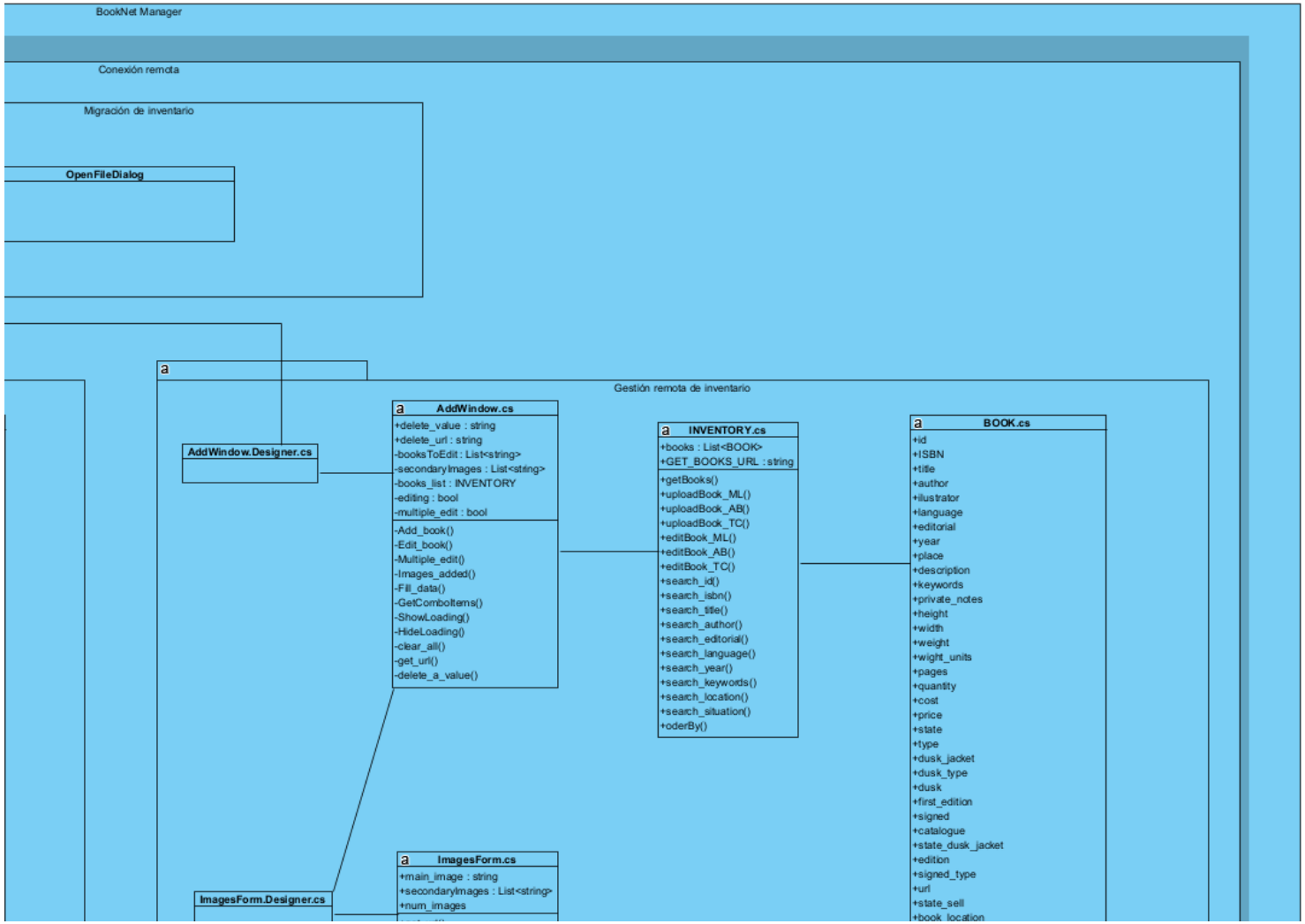

Ilustración 15: Diagrama de la vista arquitectónica del modelo de diseño, gestión remota de inventario

En esta etapa, se refinaron los diagramas de secuencia mediante la aplicación del patrón de diseño elegido.

A continuación, se mostrarán algunos de los diagramas de secuencia finales, el resto pueden encontrarse en el ANEXO IV.

-CU-0001: Conectar con servidor remoto.

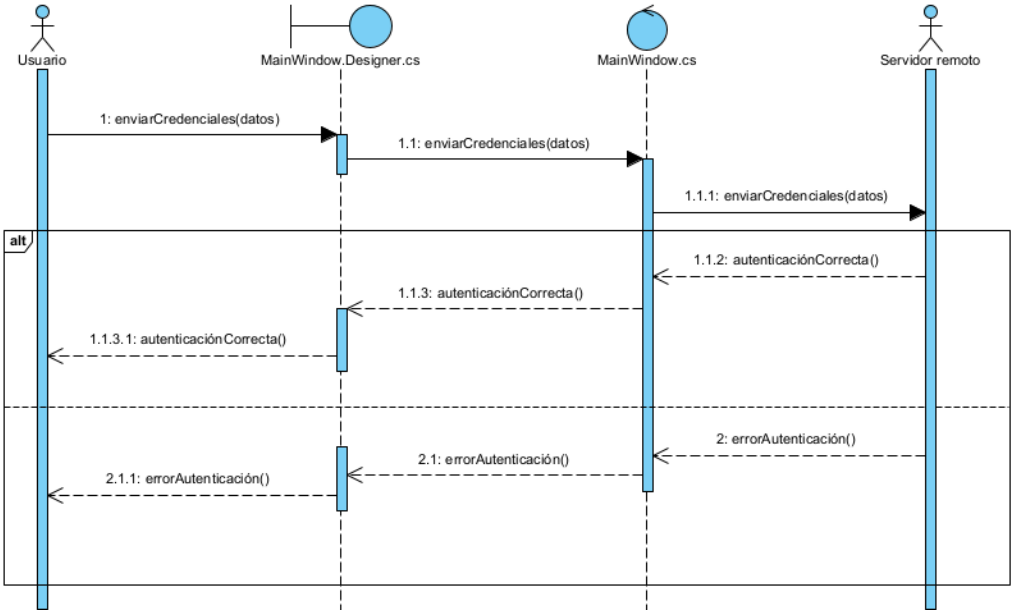

Ilustración 16: Diagrama de secuencia refinado del CU-0001 "Conectar con servidor remoto"

-CU-0004: Actualizar inventario.

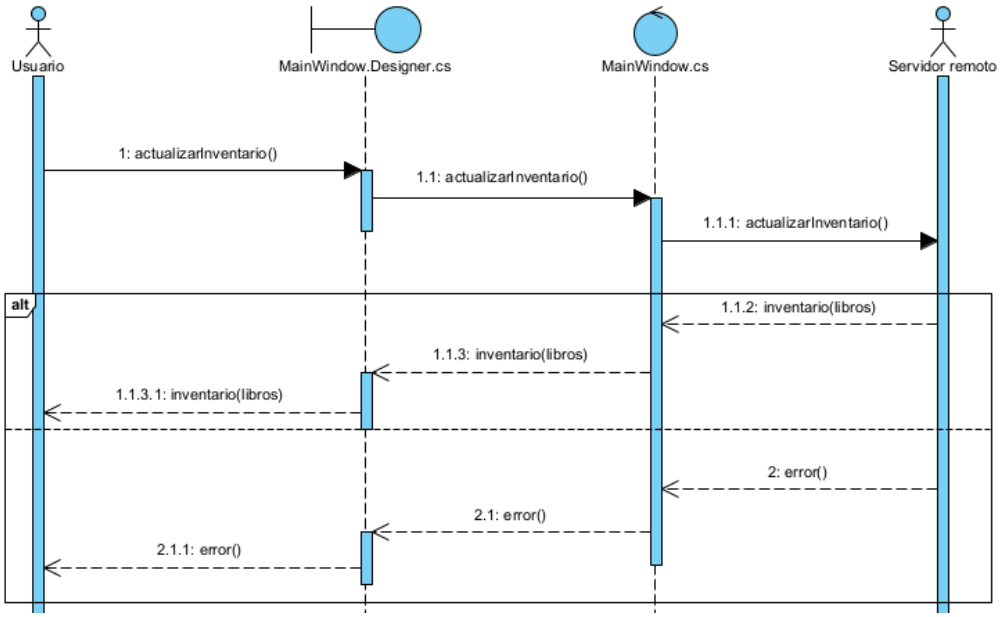

Ilustración 17: Diagrama de secuencia refinado del CU-0004 "Actualizar inventario"

-CU-0011: Añadir libro.

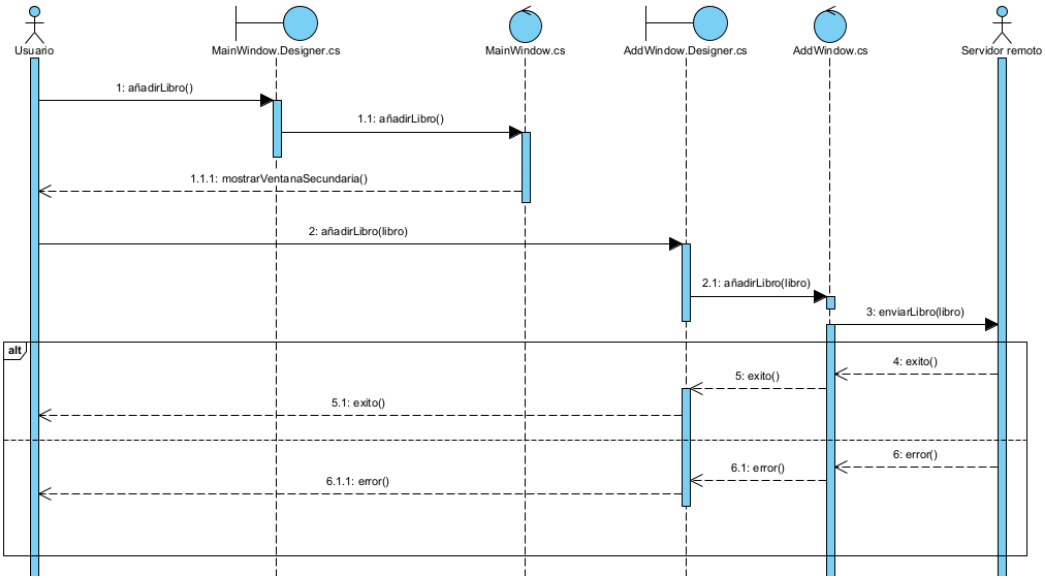

Ilustración 18: Diagrama de secuencia refinado del CU-0011 "Añadir libro"

-CU-0013: Añadir imágenes.

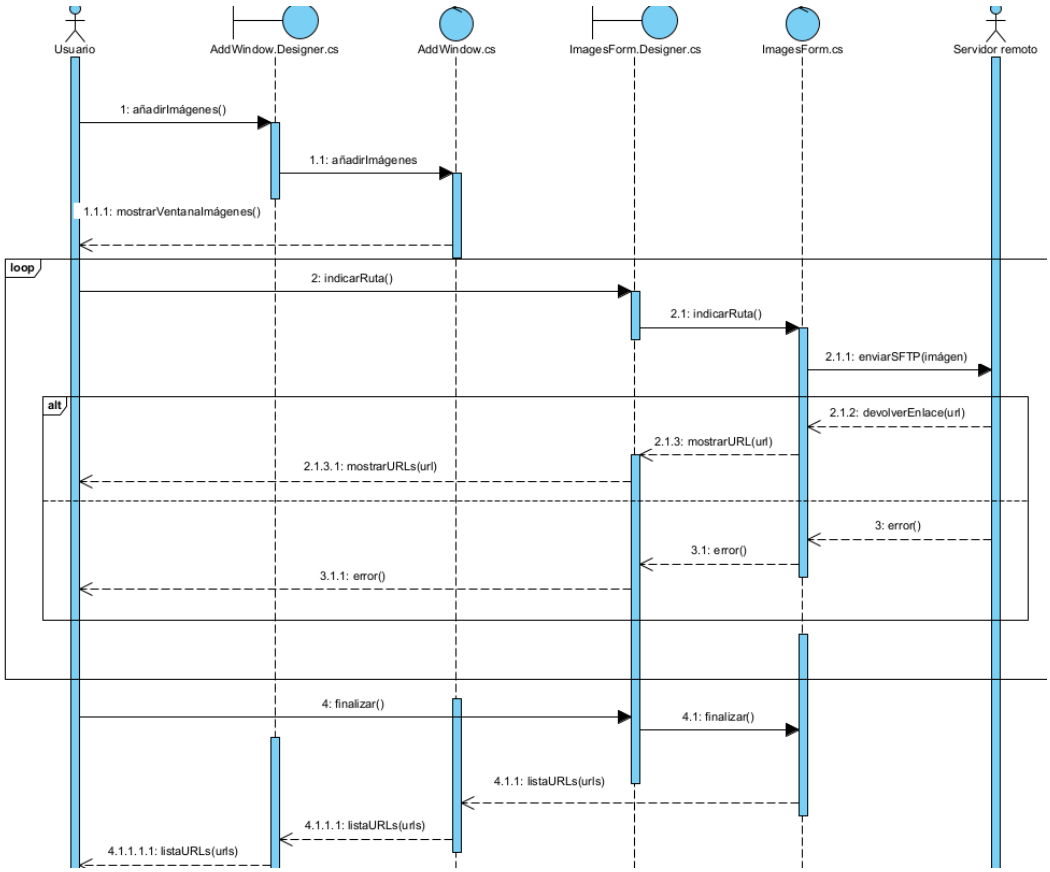

Ilustración 19: Diagrama de secuencia refinado del CU-0013 "Añadir imágenes"

Con la vista de arquitectura y los diagramas de secuencia refinados, se comenzó a diseñar las interfaces de cada paquete y a implementar sus controladores.

Además, durante esta etapa se realizan pruebas periódicas a los métodos implementados y a la interfaz, terminando con una implementación de los métodos utilizando como entrada de datos la información aportada por la interfaz gráfica, por lo que se comprueba que los controladores se comunican correctamente con las vistas y el modelo del sistema.

Finalmente, un aspecto a destacar muy importante de esta etapa es la obtención de una versión inicial del programa, sobre la que poder realizar pruebas tratando de generar una primera versión estable del producto.

5.1.5. Etapa de desarrollo

La etapa de desarrollo del sistema tiene lugar a la vez que se realiza el diseño, dado que los refinamientos de los elementos de diseño del sistema influirán en el modo en que se implementan sus métodos.

Se comenzó realizando una versión local, sin la utilización de un servidor remoto en el que almacenar el inventario. Más adelante se incorporó este servidor comenzando a implementar los objetivos reales de este proyecto.

En las siguientes páginas se detallará la información sobre cómo se implementaron las funcionalidades del proyecto en su versión remota.

5.1.5.1. Visualización de datos remotos

Tras realizar pruebas de visualización del inventario utilizando una base de datos local, se comenzó a implementar la obtención de los libros desde una base de datos remota.

Para ello, tras crear la base de datos y sus tablas en el servidor remoto, se implementó una API mediante la utilización de Flask. A continuación, puede verse el endpoint al que accede la aplicación para obtener el inventario.

```python
@app.route('/get/books')
def get_sentiment():
    config = {
        'user': '[redacted]',
        'password': '[redacted]',
        'host': '127.0.0.1',
        'database': 'munduslibri_db',
        'auth_plugin':'mysql_native_password',
        'raise_on_warnings': True
    }

    try:
        cnx = mysql.connector.connect(**config)
        cursor = cnx.cursor()

        query = ("SELECT * FROM BOOK")
        cursor.execute(query)

        row_headers = [x[0] for x in cursor.description]  # this will extract row headers
        rv = cursor.fetchall()
        json_data = []
        for result in rv:
            json_data.append(dict(zip(row_headers, result)))
        cursor.close()
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print("Something is wrong with your user name or password")
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            print("Database does not exist")
        else:
            print(err)
    else:
        cnx.close()
        return json.dumps(json_data)
```

Ilustración 20: Código del servidor remoto, obtener libros

El código del cliente para obtener el inventario es el siguiente:

```
public void getBooks()
{
    String url = GET_BOOKS_URL;
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    request.Method = "GET";
    request.Headers.Add("api-key", "[redacted]");
    HttpWebResponse response;
    response = (HttpWebResponse)request.GetResponse();
    if (response.StatusCode == HttpStatusCode.OK)
    {
        Stream response_stream = response.GetResponseStream();
        String book_rows = new StreamReader(response_stream).ReadToEnd();
        books = JsonConvert.DeserializeObject<List<BOOK>>(book_rows);


    }
}
```

Ilustración 21: Código de la aplicación, obtener libros

Durante la visualización del inventario, la aplicación permite al usuario realizar búsquedas y ordenar mediante un atributo deseado.

El código capaz de realizar esta acción se muestra a continuación:

```
public List<BOOK> search_isbn(String ISBN)
{
    var searchedList = books.Where(x => x.ISBN.StartsWith(ISBN)).ToList();
    return searchedList;
}
```

Ilustración 22: Código de la aplicación, búsqueda

```
public List<BOOK> orderBy(String name)
{
    var propertyInfo = typeof(BOOK).GetProperty(name);
    List<BOOK> newList = books.OrderBy(o => propertyInfo.GetValue(o, null)).ToList();
    return newList;
}
```

Ilustración 23: Código de la aplicación, ordenación

5.1.5.2. Exportado e importado de datos

Para realizar las tareas de importado y exportado del conjunto de libros se ha tenido en cuenta el formato de archivo que utiliza HomeBase, ya que permite exportar su inventario a un archivo y este archivo se ha utilizado para realizar el primer importado de datos al servidor.

El formato que utiliza HomeBase es el siguiente:

- **Carácter delimitador**: ~
- **Atributos**: Book ID, ISBN, Listing Type, Title, Author, Illustrator, Keywords, Book Condition, Jacket Condition, Binding Type, Language, Signature Type, Edition, Printing, Volume, Publisher, Publisher Year, Publisher Place, Size, Weight, Weight Units, Page Count, Inventory Location, Quantity, Status, Price, Cost, Description, Synopsis, Private Notes, Categories, Catalogs.

En cuanto al formato propio de la aplicación, diseñado a partir de las peticiones de Mundus Libri, es el siguiente:

- **Carácter delimitador**: ~
- **Atributos**: id, ISBN, title, autor, ilustrator, language, editorial, year, place, description, keywords, private_notes, height, width, weight, weight_units, pages, quantity, cost, Price, state, type, dusk_jacket, dusk_type, dusk, first_edition, signed, catalogue, state_dusk_jacket, edition, signed_type, url, state_sell, book_location, secondary_images.

Esto provoca que haya que realizar una adaptación o conversión de los datos entre ambos formatos.

Se muestra el código capaz de importar desde un conjunto proveniente de HomeBase, ya que deberá adaptar estos datos al formato propio de la aplicación para comenzar a enviar los libros al servidor remoto. Este método no se muestra entero ya que es demasiado largo al tener un gran número de atributos.

Adaptación de los datos:

```
private void migrate_books_old()
{
        int i = 1;
        string line;

        StreamReader file = new StreamReader(migration_path, System.Text.Encoding.UTF8);
        while ((line = file.ReadLine()) != null)
        {

            if (i == 1)
            {
                i++;
                setProgressSafe(i, "OK");
                continue;
            }
            var splitted = line.Split('~');
            String info = "migrando libro: " + splitted[0] + ".....";
            setInfoSafe(info);
            for (int j = 0; j < splitted.Length; j++)
            {
                if (splitted[j] == "")
                {
                    splitted[j] = null;
                }
            }


            String ID = splitted[0];
            String ISBN = splitted[1];
            String Type = splitted[2];
            String Title = splitted[3];
            String Author = splitted[4];
            String Illustrator = splitted[5];
            String Keywords = splitted[6];
            String State = splitted[7];
            String State_dusk_jacket = splitted[8];
            String Type_dusk_jacket = splitted[9];
            String Language = splitted[10];
            String Signature_type = splitted[11];
            String Edition = splitted[12];
            String Printing = splitted[13];
            String Volume = splitted[14];
```

Ilustración 24: Código de la aplicación, migración

Una vez obtenidos los datos, se crea el objeto libro y se envía a la base de datos:

```
        BOOK newBook = new BOOK
        {
            id = ID,
            ISBN = ISBN,
            title = Title,
            author = Author,
            ilustrator = Illustrator,
            language = Language,
            editorial = Editorial,
            year = Year,
            place = Place,
            description = Description,
            keywords = Keywords,
            private_notes = Private_notes,
            heigth = Size_height,
            width = Size_width,
            weight = Weight,
            weight_units = Weight_units,
            pages = Pages,
            quantity = Quantity,
            cost = Cost,
            price = Price,
            state = State,
            type = Type,
            dusk = Type_dusk_jacket,
            catalogue = Catalogue,
            state_dusk_jacket = State_dusk_jacket,
            edition = Edition,
            signed_type = Signature_type,
            state_sell = State_sell,
            book_location = Inventory_location
        };

    var book_json = JsonConvert.SerializeObject(newBook);
    Console.WriteLine(book_json);

    var result = UploadBookChanges.Post_request_json(ADD_BOOK_URL, book_json);
    if (result != null)
        setProgressSafe(i, "OK");
    i++;
```

Ilustración 25: Código de la aplicación, migración 2

5.1.5.3. Envío de operaciones a otros sistemas

Para realizar el envío de la información acerca de operaciones de alta, baja o modificación en la base de datos se utilizan llamadas HTTP a las distintas APIs.

Tanto para añadir como para modificar elementos de la base de datos, se utilizará una ventana auxiliar a modo de formulario que recogerá toda la información insertada por el usuario.

Una vez insertada toda la información por parte del usuario o elegidos los libros a eliminar, el sistema enviará dicha información al servidor remoto, así como a los sistemas de Abebooks y Todocolección.

Para ello, hará uso de la clase estática **UploadBookChanges.cs**, que contendrá todos los métodos necesarios para realizar las llamadas a los distintos sistemas.

A continuación, se muestra el código referente al envío de datos al servidor remoto mediante peticiones POST:

```
public static void uploadBook_ML(BOOK added_book)
{
    try
    {
        var book_json = JsonConvert.SerializeObject(added_book);
        Console.WriteLine(book_json);
        var result = Post_request_json(ADD_BOOK_URL, book_json);
    }
    catch
    {
        MessageBox.Show("El libro no ha podido añadirse al servidor de la librería, contacta con el administrador vramos@usal.es",
    }
}
```

Ilustración 26: Código de la aplicación, envío a Mundus Libri

```
public static string Post_request_json(String url, String json)
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    request.ContentType = "application/json";
    request.Method = "POST";
    request.Headers.Add("api-key", "[redacted]");

    using (var streamWriter = new StreamWriter(request.GetRequestStream()))
    {
        streamWriter.Write(json);
    }

    HttpWebResponse response;
    response = (HttpWebResponse)request.GetResponse();
    if (response.StatusCode == HttpStatusCode.OK)
    {
        Stream response_stream = response.GetResponseStream();
        String response_str = new StreamReader(response_stream).ReadToEnd();
        return response_str;
    }
    return null;

}
```

Ilustración 27: Código de la aplicación, petición POST

En el servidor remoto, se obtiene esta información y se almacena en la base de datos.

```
@app.route('/add/book', methods=['POST'])
def add_book():
    config = {
        'user': '[redacted]',
        'password': '[redacted]',
        'host': '127.0.0.1',
        'database': 'munduslibri_db',
        'auth_plugin':'mysql_native_password',
        'raise_on_warnings': True
    }

    try:
        cnx = mysql.connector.connect(**config)
        cursor = cnx.cursor()
        book = request.json


        for key in book:
            if key == 'dusk_jacket' or key == 'first_edition' or key == 'signed':
                if book[key] == None:
                    book[key] = '0'
                elif book[key] == True:
                    book[key] = '1'
                elif book[key] == False:
                    book[key] = '0'
            else:
                if book[key] == None:
                    book[key] = ''
                else:
                    value = book[key].replace('"', '\\"')
                    value = value.replace("'", "\\'")
                    book[key] = value


        print(book)
        query = ('INSERT INTO BOOK (id,ISBN,title,author,ilustrator,language,editorial,year,place,description,key
        print(query)
        cursor.execute(query)
        cnx.commit()
        ok_code = {
            'code': 200,
            'error': 'BookMan API is working properly'
        }
        return jsonify(ok_code)
```

Ilustración 28: Código del servidor remoto, inserción de libro

El método para añadir o modificar en Todocolección utiliza el mismo sistema, por ello y debido a las numerosas comprobaciones que debe realizar para asegurar mandar la información con el formato correcto solo se muestra en este documento el inicio del método encargado de añadir un libro.

```
public static void uploadBook_TC(BOOK added_book, int id_section, int book_condition, List<S
{
    String url = "https://www.todocoleccion.net/app/tcolapi/1.0/products";

    String description = make_desc(added_book);

    string price = added_book.price.ToString();
    price = price.Replace(',', '.');

    String title = added_book.title;
    title = title.Replace("\"", "''");
    if (title.Length > 100)
    {
        while (title.Length > 100)
        {
            List<String> first = new List<String>();
            String[] splitted = title.Split(' ');
            int counter = 0;
            foreach (String txt in splitted)
            {
                if (counter != splitted.Length - 1)
                {
                    first.Add(txt);
                }
                counter++;

            }
            title = String.Join(" ", first);

        }
        description = "Título: " + added_book.title + "\n - Decripción: " + description;

    }
```

Ilustración 29: Código de la aplicación, envío a Todocolección

En cuanto a Abebooks, el formato con el que se debe enviar la información es del tipo XML (Extensible Markup Language), por lo que el método encargado de esta funcionalidad utiliza una forma distinta de construir los datos para la llamada a su API.

```
String xml_add = "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n" +
                 "<inventoryUpdateRequest version=\"1.0\">\n" +
                     "<action name=\"bookupdate\">\n" +
                         " <username>[redacted]</username>\n" +
                         " <password>[redacted]</password>\n" +
                     "</action>\n" +
                 "<AbebookList>\n" +
                     " <Abebook>\n" +
                         " <transactionType>add</transactionType>\n" +
                         " <vendorBookID>" + newBOOK.id + "</vendorBookID>\n" +
                         " <author>" + newBOOK.author + "</author>\n" +
                         "<title>" + newBOOK.title + "</title>\n" +
                         " <publisher>" + newBOOK.editorial + "</publisher>\n" +
                         " <subject>" + newBOOK.keywords + "</subject>\n" +
                         " <price currency=\"EUR\">" + price + "</price>\n" +
                         " <dustJacket>" + newBOOK.dusk_jacket + "</dustJacket>\n" +
                         " <binding type=\"" + newBOOK.dusk_type + "\">" + newBOOK.dusk + "</binding>\n
                         " <firstEdition>" + newBOOK.first_edition + "</firstEdition>\n" +
                         " <signed>" + newBOOK.signed + "</signed>\n" +
                         " <booksellerCatalogue>" + newBOOK.catalogue + "</booksellerCatalogue>\n" +
                         " <description>" + newBOOK.description + "\n\n Páginas: " + newBOOK.pages + ".
                         " <bookCondition>" + newBOOK.state + "</bookCondition>\n" +
                         " <size>" + height + "\" x " + widht + "\"</size>\n" +
                         " <jacketCondition>" + newBOOK.state_dusk_jacket + "</jacketCondition>\n" +
                         " <bookType>Libro</bookType>\n" +
                         " <isbn>" + newBOOK.ISBN + "</isbn>\n" +
                         " <publishPlace>" + newBOOK.place + "</publishPlace>\n" +
                         " <publishYear>" + newBOOK.year + "</publishYear>\n" +
                         " <edition>" + newBOOK.edition + "</edition>\n" +
                         " <inscriptionType>" + newBOOK.signed_type + "</inscriptionType>\n" +
                         " <quantity amount=\"" + newBOOK.quantity + "\"></quantity>\n";
```

Ilustración 30: Código de la aplicación, XML para Abebooks

5.1.5.4. Ejecución mediante hilos

Debido a que este proyecto trata de una aplicación de escritorio, se encontraron problemas a la hora de realizar tareas costosas.

Al realizar dichas tareas, la interfaz del programa queda bloqueada debido a que el mismo hilo que se encarga de visualizarla y recibir sus eventos es el que se encarga de dicha tarea.

Para que el uso del programa sea cómodo, se han ejecutado estas tareas en hilos asíncronos de forma que la ejecución del código se haga en un hilo distinto al que creó la interfaz, evitando así la apariencia de bloqueo.

A continuación, se muestra un fragmento de código referente a esta mejora utilizando las clases **Thread** y **ThreadStart** en la migración de inventario:

```
if (old)
{
    this.migrationThread = new Thread(new ThreadStart(this.migrate_books_old));
    this.migrationThread.IsBackground = true;
    this.migrationThread.Start();
}
else
{
    this.migrationThread = new Thread(new ThreadStart(this.migrate_books));
    this.migrationThread.IsBackground = true;
    this.migrationThread.Start();
}
```

Ilustración 31: Código de la aplicación, ejecución mediante hilos

5.1.6. Etapa de pruebas del sistema

Con la versión inicial construida, se da comienzo a la etapa de transición, en esta etapa se comprobará que el sistema ha cumplido sus objetivos definidos en la etapa de inicio y se realizarán los cambios convenientes si fueran necesarios.

En esta etapa, se decidió que sería conveniente añadir una pantalla de carga, de modo que se conozca cuándo la aplicación está realizando una tarea costosa o la comunicación con otros sistemas sin que parezca que la aplicación ha quedado bloqueada. Por lo que se añadió esta pantalla realizando así las últimas mejoras acordadas.

Una vez realizados estos cambios, esta etapa finaliza con la versión final del sistema, realizando el despliegue de éste utilizando Visual Studio para la generación de un instalador que permita al usuario desplegar el sistema en su máquina.

La información referente al despliegue del sistema se encuentra detallada en el ANEXO V.

5.2. Descripción funcional de la herramienta

Durante el desarrollo del proyecto, se han tenido en cuenta diversos aspectos relevantes en cuanto cómo funciona el sistema. A continuación, se detallarán esos aspectos empezando por los aspectos generales y continuando con las distintas fases de la implementación.

5.2.1. Aspectos relevantes generales

Un aspecto relevante a tener en cuenta es que la aplicación permite al usuario realizar todas las tareas de forma remota gracias a su servidor, sin importar el lugar en el que se encuentre. Esto cumple con uno de los objetivos principales del proyecto.

Por otro lado, puesto que mucha parte del programa se ha refinado mediante reuniones con los trabajadores de Mundus Libri, la aplicación se ha adaptado a las necesidades de la librería añadiendo funcionalidades como borrar libros directamente de Abebooks o exportar únicamente los IDs de los libros al realizar un volcado a un archivo.

Como ejemplo se muestra este componente de la barra de herramientas superior que permite borrar libros de Abebooks de una forma muy rápida indicando únicamente su ID.

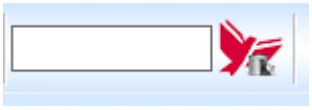

Ilustración 32: Interfaz de la aplicación, borrar libros de Abebooks

5.2.2. Visualización de inventario

En cuanto a las funciones de ordenación y búsqueda durante la visualización del inventario por parte del usuario, se he permitido fusionar estas funcionalidades para obtener resultados más precisos.

De esta forma, puede realizarse una búsqueda en la base de datos y sobre el resultado de dicha búsqueda ordenar sus elementos en función de un atributo.

En la siguiente imagen se muestra el resultado de la búsqueda por título “Carlismo” ordenado por título.

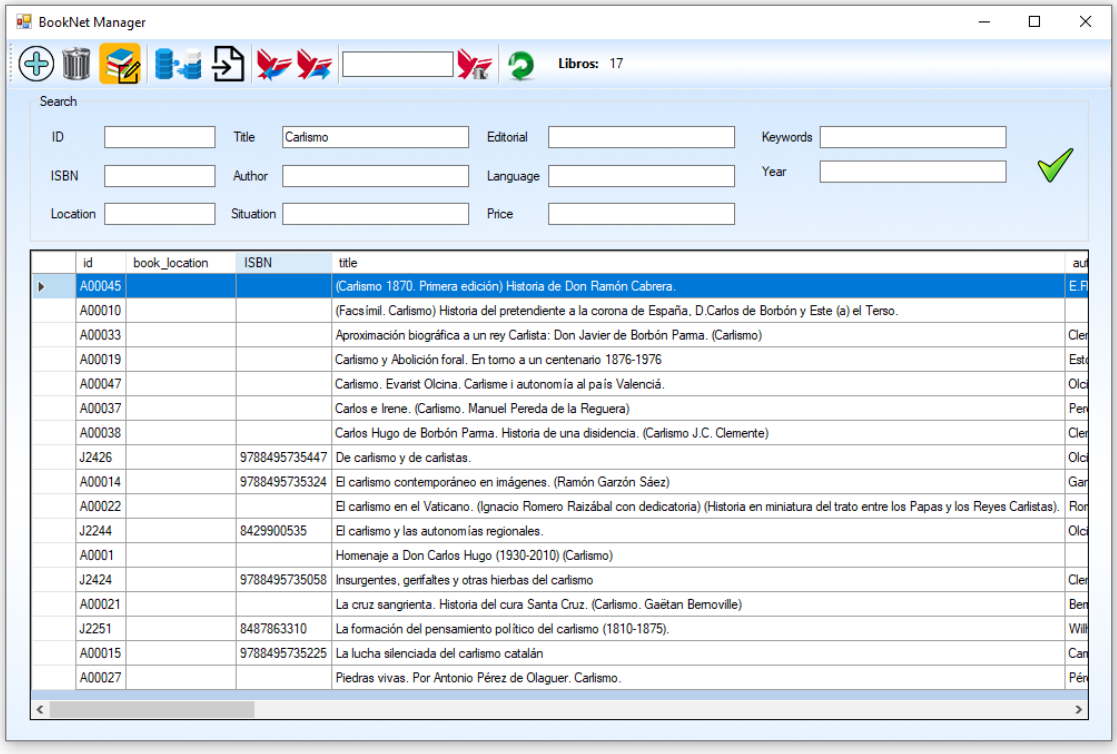

Ilustración 33: Interfaz de la aplicación, resultado de búsqueda

5.2.3. Gestión remota de inventario

Para realizar las tareas de añadido o modificación se utiliza una ventana auxiliar a modo de formulario.

Esta ventana se encarga de recoger la información escrita por el usuario para su posterior envío a los diferentes sistemas.

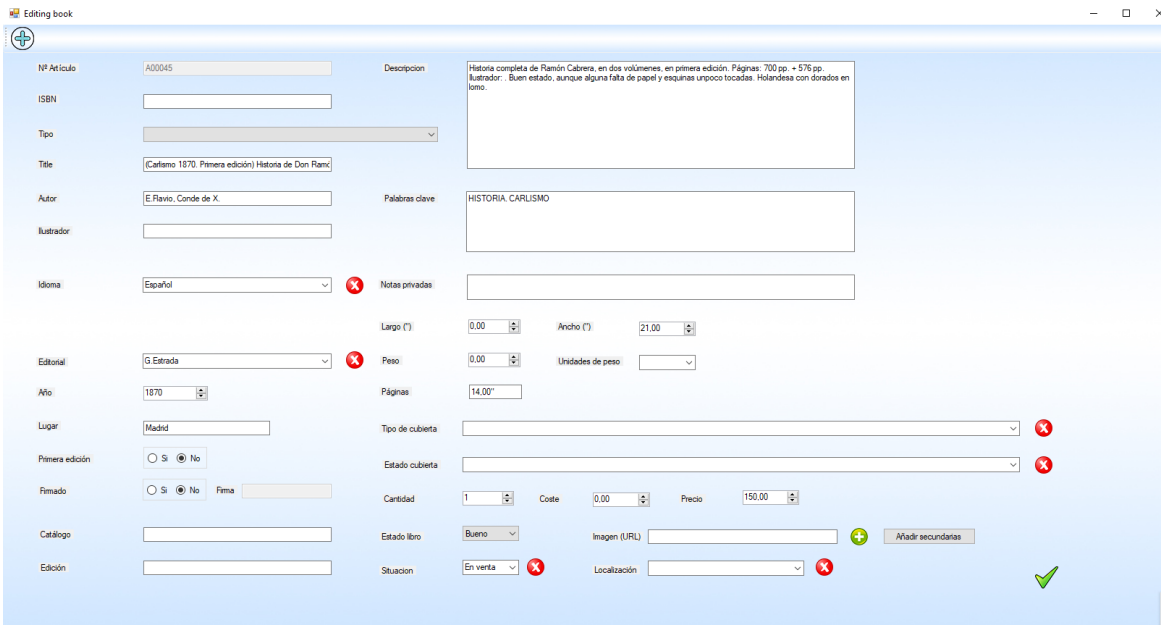

Ilustración 34: Interfaz de la aplicación, formulario de libros

Esta ventana cuenta con un botón encargado de limpiar todos los controles del formulario para, de esta forma, poder añadir más de un libro sin cerrar la ventana.

El envío de información a los distintos sistemas se realizará de forma automática y la aplicación únicamente informará al usuario de si las operaciones han tenido éxito o no.

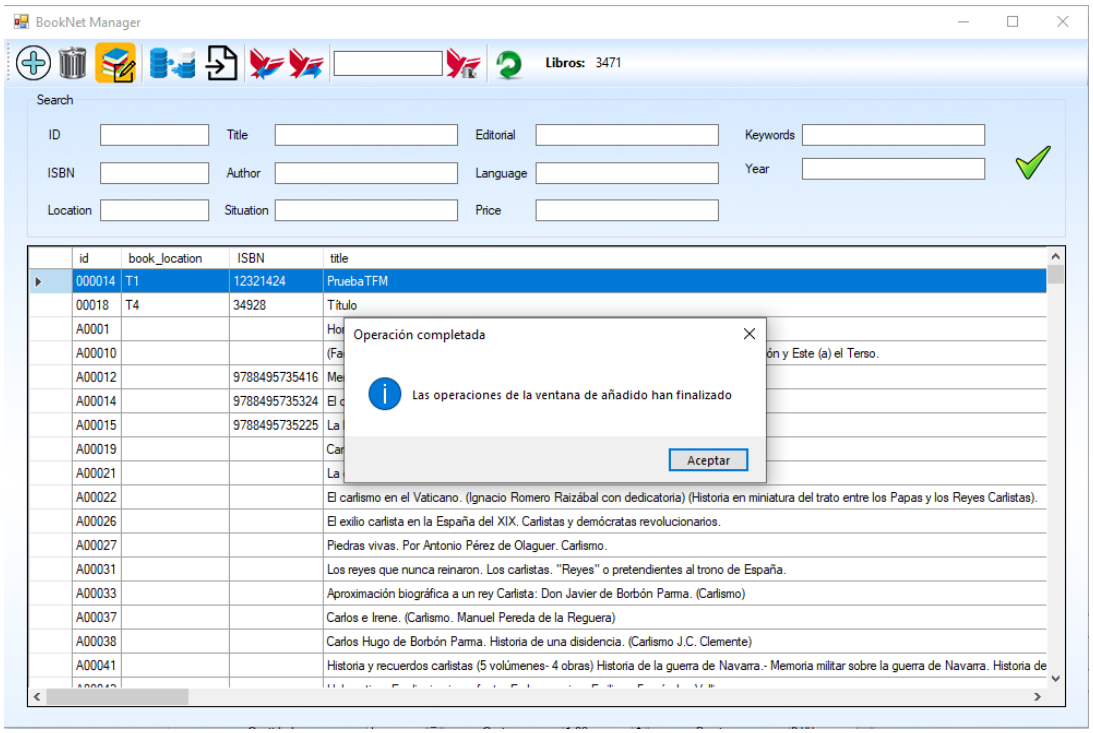

Ilustración 35: Interfaz de la aplicación, mensaje de confirmación

Otro aspecto relevante del proyecto es la inclusión de una pantalla de carga a petición de los trabajadores de Mundus Libri.

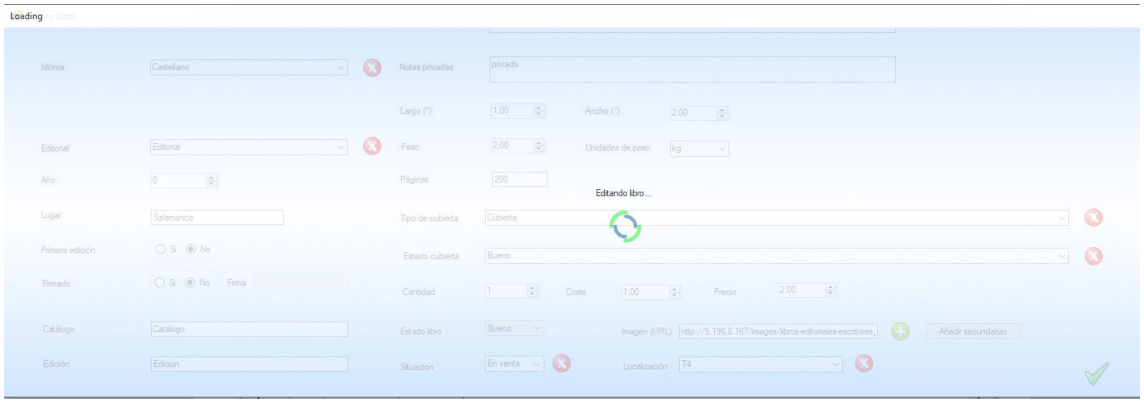

Ilustración 36: Interfaz de la aplicación, pantalla de carga

5.2.3.1. Añadido de imágenes

Para poder añadir imágenes a los libros se ha creado otra ventana auxiliar encargada esta función.

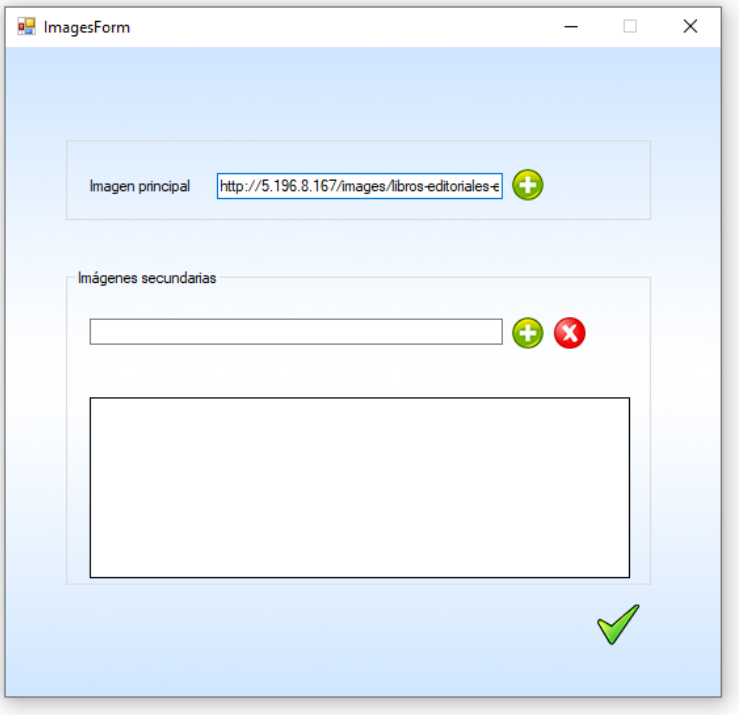

Ilustración 37: Interfaz de la aplicación, añadido de imágenes

El usuario deberá elegir la imagen en su sistema de archivos y la aplicación la enviará al servidor remoto para obtener una URL gracias a su servidor Apache.

Esta ventana también contará con una pantalla de carga para esperar a que el envío FTP sea realizado.

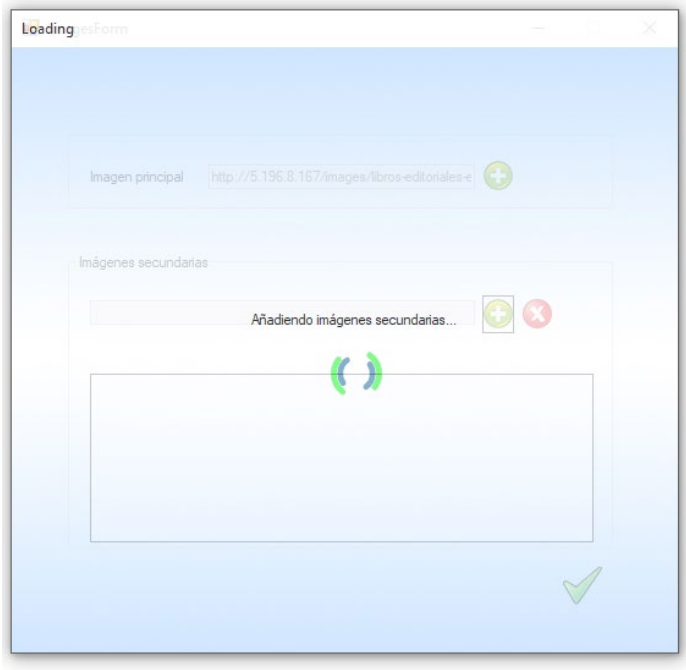

Ilustración 38: Interfaz de la aplicación, añadido de imágenes con pantalla de carga

5.2.4. Migración de inventario

En cuanto a la migración de inventario, el sistema leerá o escribirá uno a uno los libros elegidos.

Esto provoca que una simple pantalla de carga no sea suficiente dado que algunas tareas como el importado pueden extenderse en el tiempo si el conjunto posee muchos libros.

Para solucionar este problema se ha implementado una pantalla de carga informativa que contiene una barra de progreso e información sobre lo que está haciendo el programa actualmente.

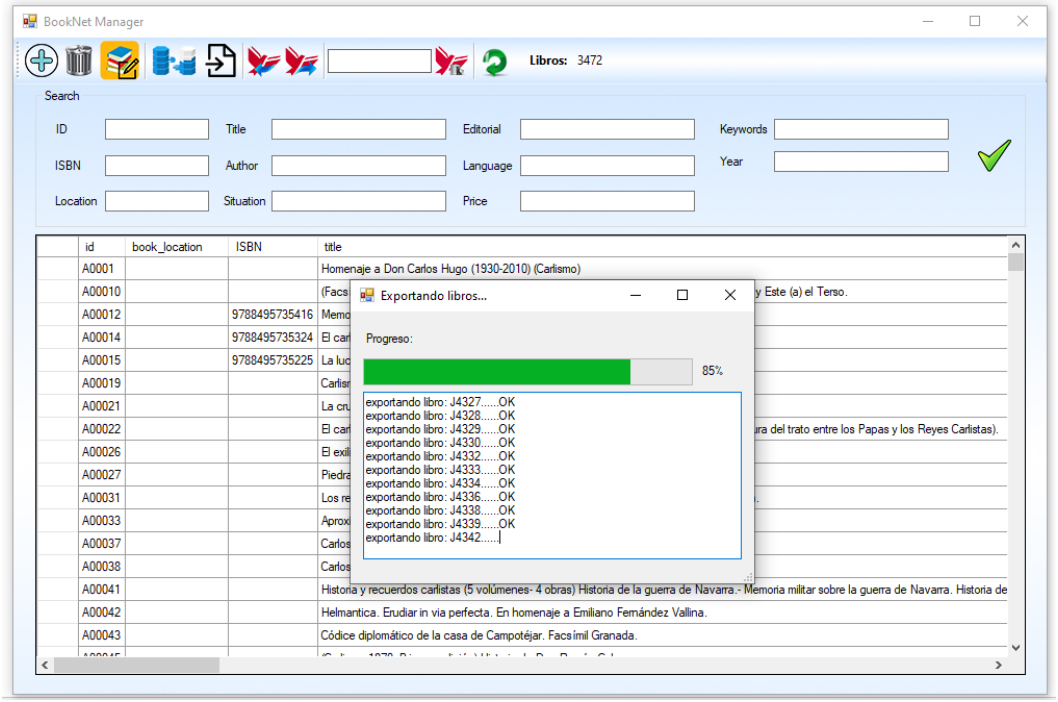

Ilustración 39: Interfaz de la aplicación, exportando libros

6. Conclusiones y líneas futuras de trabajo

Para finalizar este documento, se analizará el resultado obtenido tras la implementación total del sistema, se comprobará que los requisitos obtenidos en la fase de análisis se han cumplido y, además, se realizará una descripción de las posibles líneas de trabajo futuras que el proyecto podrá seguir con el objetivo de obtener un producto final aún más completo.

6.1. Conclusiones

Tras la finalización de este proyecto y la documentación que lo acompaña, se puede concluir que se han cumplido con los objetivos definidos en los requisitos funcionales, ya que el programa es capaz de gestionar una base de datos de forma remota y los cambios que se realizan en ella pueden verse reflejados en páginas de venta online.

En las siguientes imágenes puede verse el resultado de añadir un nuevo libro de prueba a la base de datos en las páginas de Abebooks y Todocolección.

Mundus Libri:

	id	book_location	ISBN	title
▶	000010	T1	144213213213	PruebaTFM

Ilustración 40: Resultado de añadir un libro, Mundus Libri

Todocoleccion:

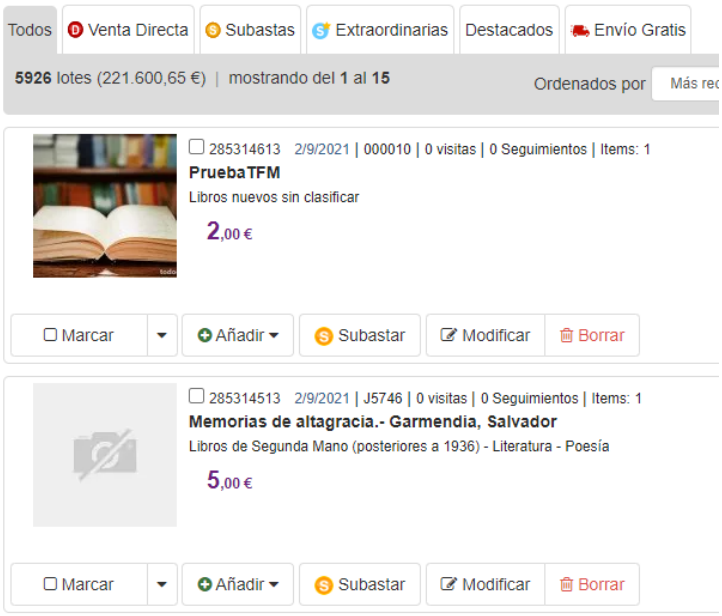

Ilustración 41: Resultado de añadir un libro, Todocolección

Abebooks:

Ilustración 42: Resultado de añadir un libro, Abebooks

Por otro lado, el sistema cumple con los objetivos de ser capaz de migrar su inventario mediante el importado y el volcado de datos y es capaz de realizar búsquedas y ordenar sus datos.

En la siguiente imagen se muestra el resultado de un volcado de datos utilizando el formato propio de la aplicación.

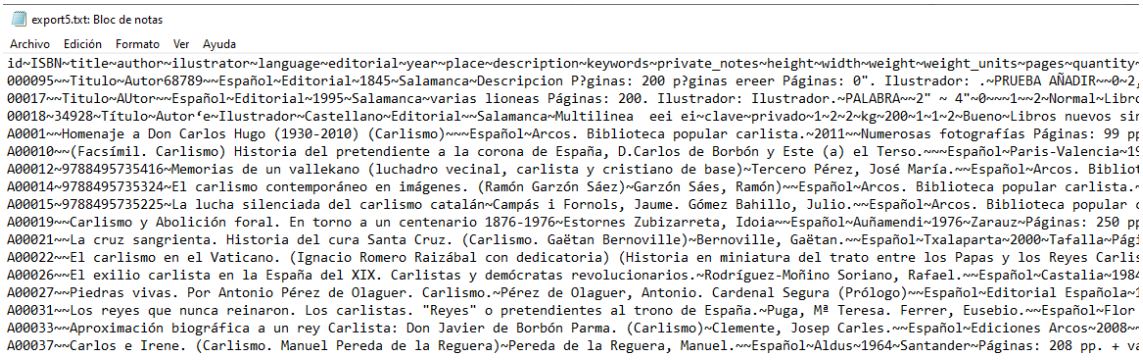

Ilustración 43: Resultado de exportado de datos

Este mismo archivo generado es válido para realizar el importado en caso de un cambio de servidor u otro problema.

Aun así, se han detallado algunas líneas de trabajo futuras que harán de este software un programa más completo para sus usuarios.

6.2. Líneas de trabajo futuras

Se han identificado cuatro líneas principales en las que se seguirá trabajando para ofrecer un producto final todavía más completo de cara a satisfacer las máximas necesidades posibles que los usuarios puedan tener.

6.2.1. Autenticación de usuario

Debido a que este proyecto se ha realizado para una sola librería, se ha omitido la autenticación mediante usuario contraseña para el acceso a la base de datos del usuario.

En un futuro, se pretende llevar este software a más librerías del territorio nacional por lo que será necesario implementar mecanismos de autenticación.

6.2.2. Autocompletado de campos

Sería de gran ayuda a la agilidad y rapidez de uso del programa que algunos campos como "idioma" o "localización" guardaran su información en base de datos.

De esta forma, cuando el usuario inserte, por ejemplo, un idioma nuevo, será añadido a la base de datos permitiendo que en la próxima ejecución del programa no sea necesario escribirlo y aparezca como una opción predeterminada.

6.2.3. Incrementar la seguridad

Aunque se han implementado medidas de seguridad en el servidor remoto como la utilización de una clave API, algunos aspectos de seguridad deberán ser mejorados en un futuro.

Las llamadas al servidor remoto deberán efectuarse utilizando el protocolo HTTPS, de forma que la información enviada sea cifrada evitando así posibles robos de información o problemas mayores.

6.2.4. Despliegue en otras plataformas

Debido a que se pretende llevar esta aplicación a otras librerías, deberá realizarse el esfuerzo necesario para implementar este proyecto en otras plataformas como Linux o MacOs.

7. Referencias

[1] Moreno García, M. Apuntes de Gestión de Proyectos, INSO I, INSO II (2019). Universidad de Salamanca.

[2] Antigüedades, Arte, LIBROS y Coleccionismo. (n.d.). Retrieved from https://www.todocoleccion.net/. Último acceso el 2 de septiembre de 2021.

[3] Search for books, fine art and collectibles. (n.d.). Retrieved from https://www.abebooks.com/. Último acceso el 2 de septiembre de 2021.

[4] Welcome to flask. (n.d.). Retrieved from https://flask.palletsprojects.com/en/2.0.x/. Último acceso el 2 de septiembre de 2021.

[5] Microsoft. (n.d.). Retrieved from https://support.microsoft.com/es-es/office/v%c3%addeo-usar-orientaci%c3%b3n-horizontal-y-vertical-en-el-mismo-documento-ddd80cb6-c9ae-4493-ba75-c663074031a0?ui=es-es&rs=es-es&ad=es. Último acceso el 2 de septiembre de 2021.

[6] Microsoft. (n.d.). Retrieved from https://support.microsoft.com/es-es/office/ajustar-texto-alrededor-de-una-imagen-en-word-bdbbe1fe-c089-4b5c-b85c-43997da64a12?ui=es-es&rs=es-es&ad=es. Último acceso el 2 de septiembre de 2021.