Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Sistema de Vectorizado y Procesamiento de Imágenes Especializada en Digitalización de Trazos en Papel

---

Alumno: Ismael Plaza García

Tutores: José Vicente Álvarez Bravo
Luis María Fuentes García

Sistema de Vectorizado y Procesamiento de Imágenes Especializada en Digitalización de Trazos en Papel

Ismael Plaza García

Índice general

Lista de figuras VII

Lista de tablas IX

Lista de ecuaciones XI

Lista de código XIII

Resumen XIX

Summary XXI

I Memoria del Proyecto 1

1. Descripción del proyecto 3
- 1.1. Introducción . . . 3
- 1.2. Objetivos del trabajo . . . 4
- 1.3. Entorno de aplicación . . . 5
  - 1.3.1. Enfoque con nuevo Hardware. Las tabletas gráficas. . . . 5
  - 1.3.2. Los diferentes softwares . . . 9

2. Metodología 13
- 2.1. Proceso de desarrollo . . . 13
- 2.2. Herramientas utilizadas . . . 17
- 2.3. Arquitectura . . . 19
- 2.4. Diseño de interfaz . . . 22

3. Planificación 25
- 3.1. Estimación del esfuerzo . . . 25
- 3.2. Planificación temporal . . . 29
- 3.3. Presupuesto económico . . . 31
  - 3.3.1. Hardware y software . . . 31
  - 3.3.2. Recursos humanos . . . 32

3.3.3. Presupuesto total 33

II Documentación técnica 35

4. Análisis 37
4.1. Requisitos 37
4.2. Atributos de calidad 45

5. Diseño 47
5.1. Diseño de datos 47
5.2. Diagrama de clases 49
5.3. Diagramas de secuencia 52
5.4. Diagrama de estado 54

6. Implementación 57
6.1. Implementación de la vectorización. VTracer por Vision Cortex 58
6.2. Implementación de la corrección de errores. 65
6.3. Implementación de subclases. 71

7. Pruebas 73
7.1. Pruebas unitarias 73
7.2. Pruebas de integración 75
7.3. Pruebas de sistema 76
7.4. Pruebas de validación 77

III Conclusiones y mejoras 79

8. Conclusiones 81
8.1. Reflexiones personales 81

9. Trabajo a futuro 83

IV Manuales y otros Apéndices 85

A. Manuales 87
A.1. Manual de Instalación 88
A.2. Installation Manual 90
A.3. Manual de Usuario 91
A.4. User Manual 109

Índice de términos 127

Índice de abreviaciones **129**

Bibliografía **131**

Índice de figuras

1.1. Ejemplo de Tableta Digitalizadora. Wacom Intuos Medium Bluetooth . . . 6
1.2. Ejemplo de Monitor interactivo (Mirror display). HUION Kamvas 13 . . . 7
1.3. Ejemplo de Portátil táctil. LENOVO con su serie de portátiles YOGA, YOGA PRO . . . 8
1.4. Ejemplo de Lápiz Digital. Wacom Pen 4K . . . 9
1.5. Pantalla principal de Illustrator nada más empezar un nuevo proyecto . . . 10
1.6. Pantalla principal de Inkscape nada más empezar un nuevo proyecto . . . 11
1.7. Búsqueda rápida en Google sobre Webs de Vectorización . . . 11

2.1. Representación gráfica de la metodología seguida . . . 14
2.2. Tablero Kanban físico . . . 15
2.3. Tarjeta del Kanban físico . . . 17
2.4. Diagrama visual de la arquitectura física. . . . 20
2.5. Diagrama visual de la arquitectura lógica fusionada con el patrón MVC. . . . 21
2.6. Paleta de colores utilizados en la aplicación. . . . 22
2.7. Logotipo de la aplicación. . . . 23
2.8. Cursor que se puede utilizar en la aplicación. . . . 23
2.9. Ilustración de un estado primitivo de la estructura de la aplicación en Qt Designer. . . . 24

3.1. Diagrama de Gantt con sus diferentes actividades. . . . 30
3.2. Diagrama de Gantt con sus diferentes actividades redimensionado para fácil lectura en la documentación. . . . 30

5.1. Diagrama de clases correspondiente a la ventana principal con ConfigDialog. 50
5.2. Diagrama de clases correspondiente a la generación del historial. . . . 51
5.3. Diagrama de clases con la gestión de los projectos. . . . 51
5.4. Diagrama de clases con los objetos subclaseados de la UI utilizados en la ventana principal. . . . 52
5.5. Diagrama de secuencia de la corrección de imagen. . . . 53
5.6. Diagrama de secuencia de la vectorización de imagen. . . . 54
5.7. Diagrama de estados de una imagen en la aplicación. . . . 55

7.1. Captura de la salida por consola del test. . . . 75

7.2. Captura de proyectos realizados durante las pruebas de sistema. 76
7.3. Captura de historiales guardados durante las pruebas de sistema. 77
7.4. Captura del interior de un historial tras una prueba de sistema. 77

A.1. Página de descarga de Github. 88
A.2. Download Github page. 90
A.3. Paso de bienvenida de la aplicación. 91
A.4. Primer paso de la aplicación. 92
A.5. Primer paso de la aplicación. Cambio de paso. 92
A.6. Primer paso de la aplicación. Uso de proyecto guardado. 93
A.7. Primer paso de la aplicación. Reuso de proyecto. 93
A.8. Guardado del proyecto. 94
A.9. Segundo paso de la aplicación. Rotar la imagen. 95
A.10.Segundo paso de la aplicación. Recortar la imagen. 95
A.11.En cualquier paso de la aplicación se pueden resetear los cambios realizados. 96
A.12.Segundo paso de la aplicación. Cambio de paso. 96
A.13.Tercer paso de la aplicación. Añadir color a la lista correspondiente. 97
A.14.Tercer paso de la aplicación. Editar o eliminar el color seleccionado. 97
A.15.Tercer paso de la aplicación. Conversión de colores. 98
A.16.Tercer paso de la aplicación. Binarización automática. 98
A.17.Tercer paso de la aplicación. Inversión de colores. 99
A.18.Tercer paso de la aplicación. Cambio de paso. 99
A.19.Cuarto paso de la aplicación. Corrección superficial. 100
A.20.Cuarto paso de la aplicación. Esqueletizacion y dilatación. 100
A.21.Cuarto paso de la aplicación. Cambio de paso. 101
A.22.Quinto paso de la aplicación. Vectorizado. 101
A.23.Sexto paso de la aplicación. Borrado de trazos. 102
A.24.Sexto paso de la aplicación. Restaurar trazo. 102
A.25.Sexto paso de la aplicación. Finalizar proyecto. 103
A.26.Configuración de la aplicación. 103
A.27.Configuración de la aplicación. Cursor. 104
A.28.Configuración de la aplicación. Tamaño del cursor. 105
A.29.Configuración de la aplicación. Tema de colores. 105
A.30.Configuración de la aplicación. Uso del historial. 106
A.31.Configuración de la aplicación. Expiración del historial. 106
A.32.Configuración de la aplicación. Ruta del historial. 107
A.33.Configuración de la aplicación. Ajustes. 107
A.34.Configuración de la aplicación. Botones de ventana. 108
A.35.Welcome screen of the application. 109
A.36.First step of the application. 110
A.37.First step of the application. Transition to the next step. 110
A.38.First step of the application. Using a saved project. 111
A.39.First step of the application. Reusing a project. 111

A.40.Saving the project. 112
A.41.Step two of the application. Rotate the image. 113
A.42.Step two of the application. Crop the image. 113
A.43.Changes made in the actual step of the application can be reset. 114
A.44.Step two of the application. Step transition. 114
A.45.Step three of the application. Add colour to the corresponding list. 115
A.46.Step three of the application. Edit or delete selected colour. 115
A.47.Step three of the application. Colour conversion. 116
A.48.Step three of the application. Automatic binarization. 116
A.49.Step three of the application. Colour inversion. 117
A.50.Step three of the application. Step transition. 117
A.51.Fourth step of the application. Surface correction. 118
A.52.Fourth step of the application. Skeletonization and dilation. 118
A.53.Fourth step of the application. Step transition. 119
A.54.Fifth step of the application. Vectorization. 119
A.55.Sixth step of the application. Deleting paths. 120
A.56.Sixth step of the application. Restoring a path. 120
A.57.Sixth step of the application. Finish project. 121
A.58.Application configuration. 121
A.59.Application configuration. Cursor. 122
A.60.Application configuration. Cursor size. 122
A.61.Application configuration. Colour theme. 123
A.62.Application configuration. Log use. 123
A.63.Application configuration. Log expiration. 124
A.64.Application configuration. Log path. 124
A.65.Application configuration. Settings. 125
A.66.Application configuration. Window buttons. 126

Índice de tablas

3.1. Casos de Uso del TFG con sus pesos asociados. 26
3.2. Reparto de pesos y valores para el ajuste por TCF. 27
3.3. Reparto de pesos y valores para el ajuste por EF. 28
3.4. Presupuesto del Hardware y del Software utilizado. 32
3.5. Presupuesto de los recursos humanos utilizados. 32
3.6. Presupuesto total estimado de todos los recursos utilizados. 33

4.1. CU01. Cargar datos de proyecto. 39
4.2. CU02. Guardar proyecto. 39
4.3. CU03. Girar imagen. 40
4.4. CU04. Cortar imagen. 40
4.5. CU05. Arreglar colores de la imagen. 41
4.6. CU06. Arreglar colores de la imagen. 42
4.7. CU07. Vectorizar imagen. 42
4.8. CU08. Personalizar imagen vectorizada. 43
4.9. CU09. Exportar imagen final. 43
4.10. CU10. Personalizar experiencia. 44
4.11. CU11. Seguridad de historial. 44
4.12. CU12. Configuración exportable. 44

Lista de ecuaciones

3.1 Ecuación de Cálculo de los UUCP

3.2 Ecuación de Cálculo del TCF

3.3 Ecuación de Cálculo del ajuste por EF

3.4 Ecuación de Cálculo del UCP

3.5 Ecuación de Cálculo del Esfuerzo

Lista de código

3 VectorizarT.pro - Búsqueda de Python Embebido.

4 PythonManager.h y PythonManager.cpp - Funciones del Singleton.

4 PythonManager.h y PythonManager.cpp - Conexión con Python.

5 VTracer.h y VTracer.cpp - Uso de Python para acceder y usar VTracer.

6.1 MainWindow.cpp - Uso de VTracer para realizar la vectorización.

6.2 Python Script - Uso de OpenCV para realizar cierres/erosiones y aperturas.

6.2 Python Script - Uso de OpenCV para realizar esqueletización y dilatación.

6.2 MainWindow.cpp - Uso de OpenCV a través de QProcess.

7.1 Test.pro - Librería QtTest.

7.1 testclass.h/testclass.cpp - Clase de test.

Por el avance y disfrute,

¡Por Yoshistonpguorld y todos sus IntegrAntes!

Agradecimientos

Muchas gracias a todas las personas que me han ayudado a llegar a donde estoy. Tanto buenos como malos momentos me han forjado hasta el día de hoy, muchas gracias.

Gracias a los profesores que me enseñaron todo lo que sé.
Con mención especial a mis tutores José Vicente y Luis María por su expléndido tutelaje durante este trabajo fin de grado y maravillosas asignaturas.

A mi familia por todo el apoyo y paciencia durante estos años.
Abuelos, ¡¡el último trabajo de la carrera #^^#!!

A mis amigos por soportar mis rarezas y por toda la ayuda, gracias a vosotros he aguantado todo este tiempo. Habéis hecho mi vida llevadera y feliz.
Gracias al Ceni0, gracias a los Pipiolos y gracias a todos los Shitpost Crusaders.
Un especial y caluroso abrazo a Pablo; entré por y gracias a ti, te dedico parte de esta salida, muchas gracias *amijo*.

To my princess, who gave me all the energy I needed and the best reason to continue <3. Work with you by my side was a pleasure and I'll always remember this days when you support me this much to continue working. For you, my lovely princess, thanks you so much, your smile filled my batteries in seconds.

ฉันรัก
คุณ

Resumen

Muchas personas tienen problemas pasando del formato físico al digital, pues los programas actuales se centran demasiado en el público amplio y ofrecen múltiples herramientas contenidas. VectorizarT, la aplicación realizada y documentada en este Trabajo Fin de Grado, se centra en estos usuarios para poder darles una herramienta sencilla y eficaz para digitalizar sus dibujos físicos.

Para ello, se han empleado OpenCV y VTracer a través de Python para el procesamiento de imagen y la vectorización, respectivamente, y Qt Creator en C++ y todas sus herramientas de compilación para el diseño y programación de la Interfaz gráfica de usuario. Todo este proceso ha seguido las reglas metodológicas combinadas del Modelo en V y Kanban, documentándolo usando Visual Studio Code con LaTeX.

El resultado tras todo el desarrollo ha sido una aplicación bastante prometedora con unas vectorizaciones rápidas, limpias y eficientes facilitando una experiencia de digitalización más accesible y satisfactoria que otras herramientas más complejas. Esta aplicación aporta a la comunidad de artistas digitales novatos una nueva manera de iniciarse en el medio a través de los propios dibujos hechos a mano sin que su calidad original se vea diluida.

Palabras claves: Vectorización, digitalización, Qt Creator, Kanban, modelo en V, aplicación Windows.

Summary

Many people have difficulties transitioning from physical to digital format, as current programs focus too much on a broad audience and offer multiple built-in tools. VectorizarT, the application developed and documented in this Final Degree Project, focuses on these users to provide them with a simple and effective tool to digitize their physical drawings.

To achieve this, OpenCV and VTracer have been used through Python for image processing and vectorization, respectively, and Qt Creator in C++, along with all its compilations tools, for designing and programming the graphical user interface. The entire process followed a combined methodological approach based on the V-Model and Kanban, and was documented using Visual Studio Code with LaTeX.

The result of the whole development process is a highly promising application, offering fast, clean, and efficient vectorizations that provide a more accessible and satisfying digitization experience compared to other, more complex tools. This application offers the community of novice digital artists a new way to enter the environment through their own hand-drawn work, without compromising the original quality.

Keywords: Vectorization, digitization, Qt Creator, Kanban, V-model, Windows application.

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

1.1. Introducción

A través de esta Memoria de Trabajo de Fin de Grado (abreviado a *TFG* en el resto del documento) se **documentará** todo el **procedimiento** seguido en la creación de una **aplicación de escritorio** de **procesamiento y vectorización de imágenes**, especializada en las imágenes resultantes de escaneos de trazados realizados a mano de manera física.

El objetivo de este TFG es el **cubrir una necesidad** que personas de varias generaciones pasadas y algunas actuales sufren: todas las piezas gráficas y visuales actuales son necesarias en formato digital y estas personas han aprendido a dibujar en un entorno físico, pero no en uno virtual; la aplicación facilitará a estos perfiles de usuario esta transición mediante una herramienta sencilla, accesible y eficaz.

El problema principal para estas personas es ese **salto digital** producido por la era de la informática donde todas **las ilustraciones son usadas en el ambiente tecnológico** y pocas veces, en comparación, en físico (y aún en los casos puramente físicos, como carteles publicitarios, tienen que pasar por una imprenta que dificulta aún más el trabajo al usar formatos de imagen extremadamente específicos y tamaños aún más específicos[1]). Entre logotipos, diseños, bocetos, arte, etcétera; todo acaba siendo necesario por una razón u otra de manera digital.

Las barreras que encuentran estas personas incluyen interfaces complejas, curvas de aprendizaje elevadas, modelos de pago restrictivos y una falta de enfoque en la usabilidad como puente entre el mundo físico y el digital. En este contexto, **VectorizarT** nace como una propuesta alternativa que pretende reducir esa brecha y servir de herramienta de iniciación al entorno digital para artistas y creativos que parten del dibujo manual. El diseño de la aplicación busca ser lo más intuitivo posible, permitiendo que usuarios con perfiles muy distintos puedan adaptarse a su uso sin necesidad de conocimientos técnicos avanzados.

[1]Información proporcionada por uno de los asesores. *Ocupación: Diseñador de Juegos de Mesa.*

Siguiendo también la filosofía de compartir y propagar el conocimiento a la vez que la confianza con el usuario, cualquier persona podrá **acceder al código del programa y al programa** compilado (*Open Source*), evitando de esta manera cualquier barrera de entrada y ayudando a crear un ambiente cooperativo.

Durante el desarrollo del proyecto se han utilizado tecnologías como **Python**, con bibliotecas como **OpenCV** y **VTracer**, para el procesamiento y vectorizado de imágenes, así como **Qt Creator** en **C++** para la creación de la interfaz gráfica. Además, se ha adoptado una metodología mixta basada en el **Modelo en V** y **Kanban**, lo que ha permitido una planificación estructurada y un desarrollo ágil y flexible.

Por último, para una lectura más ordenada y agradable, esta memoria de TFG se estructurará a través de los siguientes apartados:

1. **Memoria del proyecto**. Empezaremos el capítulo con la descripción del proyecto, tendrá detallada la metodología utilizada en el desarrollo de la aplicación y explicará la planificación seguida.

   En este capítulo se verá la **motivación** principal del proyecto, cómo ha sido el **desarrollo** de la aplicación y cómo se ha trabajado.

2. **Documentación técnica del TFG**. Continuaremos con el análisis y diseño de la aplicación, junto con su implementación y pruebas realizadas durante toda la vida del TFG.

   En este capítulo se seguirá el **proceso de creación** de la aplicación partiendo del problema a resolver.

3. **Manuales de uso de la aplicación**. Explicará de manera secuencial las acciones que se pueden realizar en el programa, tanto en su instalación en un nuevo ordenador como su uso.

   En este capítulo se **guiará** a todo posible usuario en la **instalación y uso** del programa.

4. **Conclusiones**. Finalizaremos con una retrospección del trabajo realizado y miraremos al futuro para dar una solución al problema principal desde el enfoque más definitivo y pulido.

   En este capítulo se explicará el **trabajo futuro** que tendrá la aplicación junto con las **conclusiones personales** una vez acabado el TFG.

1.2. Objetivos del trabajo

El **objetivo principal** del desarrollo de este TFG es el de hacer **uso de todo el conocimiento adquirido** a través de estos años de estudio para realizar una **aplicación completamente funcional** que sea útil a nivel personal y profesional de **uso diario**.

A mayores, se tienen objetivos más específicos que ayudan a conseguir este objetivo principal:

- Desarrollar una **aplicación que ayude a un público específico** que está pasando por el mismo problema que yo: necesidad de una **digitalización de imagen fácil y comprensible con una calidad excelente, accesible a nivel económico y con seguridad en un código fiable y accesible**.
- Construir la aplicación de manera **estructuralmente correcta** y haciendo uso de **tecnologías actuales y adaptadas** a las necesidades de la aplicación.
- Realizar una **investigación previa** y **estudio de mercado riguroso** para focalizar aún más en el resultado necesario.
- Conseguir el diseño **más dinámico y a la vez lineal** para que todos los tipos de usuarios puedan hacer uso de la aplicación a su ritmo.
- **Documentar** todo el **código** y **mantenerlo** de la manera **más ordenada posible para una mejor comprensión externa**.
- **Traducir la aplicación y el código** a un idioma más globalizado, como es el inglés, para ensanchar el grupo de usuarios potenciales.
- **Aprender en mayor profundidad** la programación en una herramienta vista durante los estudios del Grado para extraer todo su potencial.
- **Maquetar** toda la aplicación para que **los usuarios puedan estar cómodos**.

1.3. Entorno de aplicación

Existen muchos enfoques diferentes para nuestro problema a solucionar; de **hardware** y de **software**, llegando hace poco para este último las Inteligencias Artificiales (de aquí en adelante abreviadas como IAs) generativas de imágenes. Enfoques que, en su gran mayoría, no ayudan inmediatamente teniendo que aprender de cero un programa inmenso e incluso añadiendo más problemas aún. Estudiaremos en este apartado estos problemas añadidos con sus respectivos enfoques.

1.3.1. Enfoque con nuevo Hardware. Las tabletas gráficas.

Para empezar en el mundo digital, mucha gente opta por el planteamiento del nuevo hardware con el cambio de paradigma al resultar más familiar, pero viene con todas las consecuencias de un cambio absoluto.

Entre sus **ventajas** está su **aproximación física al usar lápices**, pero la apariencia externa es todo lo que tienen en común y tienen barreras de entrada demasiado grandes como para tomarlos como una posibilidad nada más empezar a digitalizar.

Tabletas Gráficas

[1] El producto más usado por los usuarios finales son las tabletas gráficas. Hacen el reclamo a nuevos usuarios intentando verse como la manera más fácil para pasarse al digital pues, al contrario del ratón, son **movimientos más conocidos, agradables y ergonómicos**.

El componente principal y **ventaja** de todas las tabletas gráficas es su **superficie de dibujo**, donde se realizan los trazos ya sea por pulsos electromagnéticos provenientes de un lápiz o por presión. La velocidad del trazo, presión, ángulo y posición del lápiz hará que cambie el trazo al momento y se consigue con ello un **estilo propio con el tiempo**, **pero esta cantidad de tiempo es justo su mayor desventaja** para nuestro objetivo. Puede resultar tedioso dependiendo de la habilidad, paciencia y software con el que se acompañe el hardware, y la **curva de dificultad inicial resulta abrumadora** pues la gran mayoría de técnicas utilizadas en el dibujo tradicional no funcionan y hay que desaprender para aprender de nuevo.

Otra **ventaja** de estos productos son sus **botones de rápida acción**, muchos de ellos configurables, que permiten hacer **comandos de instrucciones de manera inmediata**. Estas configuraciones son capaces desde copiar, cortar y pegar hasta cambiar de capa[2] en el programa que se está utilizando o incluso crear nuevas.

Dividiéndolas por su diferencia más notable tenemos 2 tipos:

Tabletas digitalizadoras El usuario realiza los trazos en una **superficie completamente opaca y rugosa** y estos se transmiten al ordenador/pantalla al que está conectada 1.1. Es una ventaja al ser capaz de ver todos los trazos sin que la mano los tape, pero esta **coordinación** necesita aún **más entrenamiento**, por lo que a corto plazo se convierte en una **desventaja añadida**.

Figura 1.1: Ejemplo de Tableta Digitalizadora. Wacom Intuos Medium Bluetooth

Los precios las convierten en las **tabletas más asequibles** rondando los

[2]Las capas en el dibujo digital separan partes del dibujo para poder trabajar más libremente y que luego se vean juntas y superpuestas de la manera deseada en la composición final.

50€ los modelos con menor tamaño y llegando a los 350€ los modelos más profesionales y grandes.

Monitores interactivos y Pen-Displays Tabletas gráficas que tienen **incorporada una pantalla táctil** en la que se realizan los **trazos** y movimientos que se verán **plasmados directa e inmediatamente**. En estos modelos, el dibujo se realiza de la forma habitual: **viendo en la misma plataforma de dibujo lo que estás haciendo**. Resulta una ventaja práctica si se está empezando al resultar más visual.

En esta misma categoría, existen dos **tipos de pantallas**: los monitores que **reflejan directamente** lo que se está viendo en la **pantalla a la que están conectados** 1.2 o monitores que funcionan como una **pantalla adicional al dispositivo** en la que puedes ver y dibujar diferentes contenidos.

Figura 1.2: Ejemplo de Monitor interactivo (Mirror display). HUION Kamvas 13

Como **mayor desventaja**, el **precio** de estos modelos comienzan desde los 300€ en los modelos más convencionales y llegan hasta los 1100€ en los modelos más profesionales y con mejores prestaciones.

Productos sustitutivos

Las tabletas *lowcost*, portátiles con pantalla táctil e incluso los *iPads* son productos sustitutivos que **cubren la misma necesidad que las tabletas gráficas** y, aunque a priori lo pareciera, los problemas que tienen no son exactamente los mismos.

Tabletas *Lowcost* Este tipo de tabletas, muchas de ellas a raíz de una **reconceptualización de tablets táctiles comunes**, suelen ser **productos toscos y poco amigables**. No tienen botones adicionales, su interfaz no está preparada para ser utilizada con buenas herramientas de dibujo digital y su conexión con otros sistemas es realmente tedioso. Además, incluso los modelos mejores acondicionados, tienen un **tiempo de respuesta a las herramientas digitales demasiado larga** y muchas veces no llega a registrar algunos trazos.

Pórtatiles táctiles 1.3 Son una **opción equilibrada** para los usuarios que llevan un tiempo en el dibujo digital, pero que no quieren llegar a profesionalizarlo. Estos portátiles vienen provistos de una pantalla que registra los trazos realizados por los lápices digitales conectados al portátil y piel (las **pantallas son parecidas a las que tienen los móviles**).

Figura 1.3: Ejemplo de Portátil táctil. LENOVO con su serie de portátiles YOGA, YOGA PRO

[2] Tienen la ventaja de ser **completamente portátiles e independientes** al ser un ordenador, pero tener que procesar el propio ordenador y los trazos puede resultar en una **ralentización de todo el proceso de dibujo y digitalización**. Otra característica de estos portátiles es que suelen tener la posibilidad de **doblarse por completo** para ser controlados en forma de tableta, pero encuentra sus desventajas al poder presionar sin querer las teclas del teclado.

iPads [3] Son tabletas capaces de reconocer y capturar trazos en su superficie **sin necesidad de un ordenador al que mandar la señal**. Son, al igual que los portátiles táctiles, una **opción equilibrada** a mitad de camino, pero **con aún más limitaciones** propias de la compañía *Apple*. Entre sus ventajas se encuentra su **comodidad**, su **facilidad de trazo** y **sencillez de control**, pero desgraciadamente por su precio no merece la pena para este propósito al poder comprar algo mucho más completo por el mismo precio o incluso menor; sus precios suelen empezar en los 1000€ y sus complementos, como los lapices digitales, pueden alcanzar los 200€.

Además de las características y desventajas de cada propio dispositivo, este nuevo hardware viene con la **necesidad de productos añadidos** como son los propios **lápices digitales** 1.4 (algunos son muy duraderos, pero otros necesitarán incluso cambios de punta adicionales), **guantes** para que la mano no cree anomalías en los trazos o irregularidades

por el movimiento de la pantalla interior del programa e incluso **soportes** para ayudar a estabilizar el trazo.

Figura 1.4: Ejemplo de Lápiz Digital. Wacom Pen 4K

La razón principal por la que el nuevo hardware **no soluciona el problema** es por la gran **barrera de entrada** que es comprar un dispositivo y aprender a controlarlo **sin haber experimentado** aún todas las bondades que permite **el dibujo digital** tales como la facilidad a los cambios y borrado sin imperfecciones añadidas, pintado y texturizado rápido, experimentación en el estilo sencillo e instantáneo, etcétera.

*"Cuando el aprendizaje del hardware es demasiado obtuso lleva a la persona a dejar su avance y acaba quemado ... sin contar el aprendizaje del software que hay que hacer simultáneamente con el del hardware, muchas personas no tienen ni siquiera el tiempo suficiente para seguir las lecciones ... les pasó a muchos de mis compañeros al final del primer año al no ver resultados y lo dejaron. ... Eso sin tener en cuenta que ninguno de los que estábamos en clase habríamos podido siquiera empezar el curso sin que *institución en la que estudió* nos cediera buenas tabletas gráficas en las que estudiar y trabajar."* [3]

1.3.2. Los diferentes softwares

Dado que el hardware es un complemento que por si solo no soluciona el problema y que necesita el software para poder trabajar, ¿cuál es la **razón de no usar un software existente** para nuestro problema? La respuesta inmediata es: **la complejidad de los programas**; pero los problemas que presentan individualmente también son dignos de mención.

Illustrator [4] El programa 1.5 **más completo y enfocado** en el dibujo y tratamiento de imágenes vectoriales de la **Suite de Adobe**. Su **principal problema** es el **precio**. Individualmente el programa cuesta 27€ al mes, aunque también permiten pagar por toda la Suite por 68€ al mes, lo que resulta más "económico" si se va a usar más de un programa; como sería el caso si se van a hacer animaciones con After Effects (que también funciona con vectores) o realizar pequeños bocetos como imágenes rasterizadas intermedias con Photoshop. Un **problema añadido** a este modo de pagar software como servicio, **no posees el software que compras** pues funciona a través de la suscripción mensual.

Otro **gran problema** que presenta es su **uso de recursos de memoria RAM y tarjeta gráfica**. Al ser un programa tan completo y gigante a nivel de opciones, el simple hecho de abrir el programa puede ser imposible para ordenadores comunes, pero además a la hora de renderizar[4] cualquiera de esas herramientas puede detener

[3] Testimonio de un asesor. *Ocupación: Artista e ilustrador digital.*

[4] La renderización es el proceso por el que pasa una imagen cuando se le hace cualquier cambio para que este sea efectivo en el programa. Para que esa imagen renderizada se pueda usar fuera del programa hay que exportar la imagen.

por completo el ordenador durante horas y acabar el procedimiento con fallos.

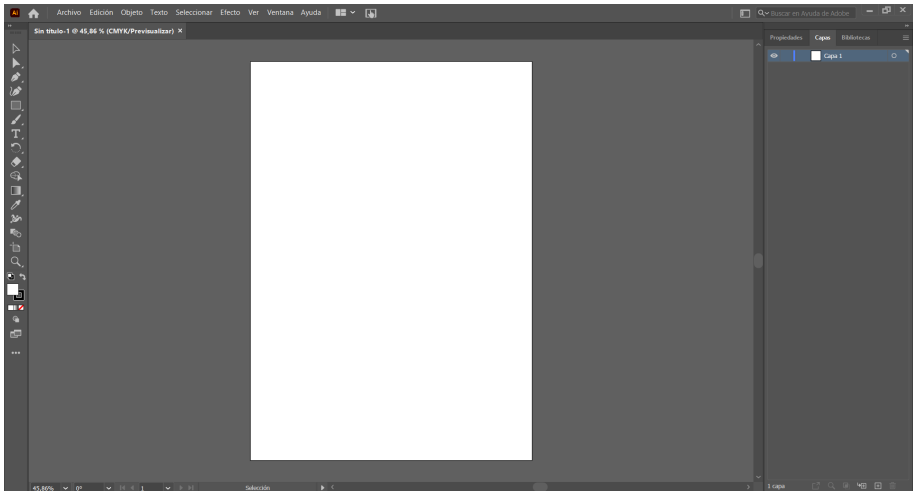

Figura 1.5: Pantalla principal de Illustrator nada más empezar un nuevo proyecto

Inkscape [5] Programa 1.6 para dibujo vectorial que trabaja a **nivel de potencia y profesionalidad** como Illustrator, pero su **mayor ventaja** es su tipo de **licencia y precio**. Su licencia es Open Source y de uso completamente gratis.

Su **desvenja principal**, compartida con su contraparte de pago, es su **inmensidad y complejidad**, pero en este caso su uso y repartición de recursos no es un problema pues los procesos no aparecen en memoria hasta que se van a utilizar.

Inkscape podría ser la solución al problema que intentamos solucionar, pero falla en el punto más importante y más crítico: **la herramienta que puede ayudar a los usuarios objetivos está escondida** entre muchas otras y, cuando consigues ejecutarla, **resulta complicada de seguir** aún sabiendo lo que hace cada parámetro.

Cambios de posición entre versiones, cambios de nombre e incluso cambios de procedimientos internos hace que estas **funcionalidades acaben siendo vagas** y, aunque completas, **no llegan a mostrar todo su potencial** al no saber enfocarse con el público objetivo que usará esa característica.

Webs "gratuitas" + IAs de vectorización automática Con estas webs ya **no hay complejidad ni aprendizaje real** para el usuario pues simplemente se sube una imagen y se descarga el vector que la web cree que es el correcto (algunas también permiten corregirlo, pero correcciones superficiales).

Aún así, el problema principal con estas webs es la **honestidad** que tienen y confianza que hay que depositar en ellas. Las cookies, las políticas de uso y todos los derechos que se les ceden automáticamente solo por hacer uso de su conversor a vectores son, en muchas ocasiones, contratos invisibles que se aceptan sin poder desistir

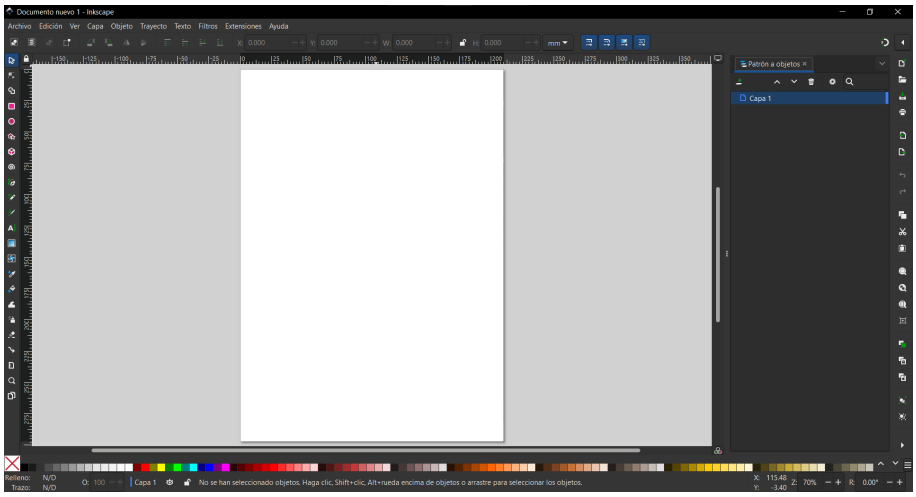

Figura 1.6: Pantalla principal de Inkscape nada más empezar un nuevo proyecto

en el que la web puede llegar a **robar el contenido** creado para posteriormente venderlo o utilizarlo para crear "nuevo" contenido con IAs.

Las IAs en este contexto funcionan de la misma manera, pero el vector correspondiente suele ser solo una **aproximación** en la que **aparecen y desaparecen artefactos** de manera incontrolable para el usuario.

Figura 1.7: Búsqueda rápida en Google sobre Webs de Vectorización

IAs generativas En el caso de las IAs generativas, **no existen ventajas reales** más allá de la inmediatez de hacer una idea realidad pasando por la simple escritura de la misma, pero **no hay aprendizaje posible**. El usuario no ha realizado la digitalización de su arte, ni tampoco podrá aprender lo que le puede permitir hacerlo.

Además, por como se han conseguido la mayoría de estas IAs, se generaría un juicio moral para el usuario por el uso ilícito de otras obras.

En la resolución de nuestro problema, las IAs son solo una **sustitución y agravación del problema**.

En conclusión, el **programa descrito en este TFG** seguirá de cerca los pasos andados por Inkscape haciendo un **buen software**, de **uso libre y abierto a cualquier**

vistazo o incluso mejora por parte de los usuarios, pero **simplificando el proceso** para que todo tipo de usuarios sean capaces de usarlo. De este modo, se conseguirá arreglar el problema de complejidad y los usuarios tendrán el control total de su vectorización.

Capítulo 2

Metodología

Este TFG se realizará haciendo uso de una **metodología híbrida personalizada** y adaptada al tipo de proyecto y al marco de trabajo unipersonal. Para ello se apoyará en herramientas especializadas para conseguir el mejor resultado posible.

2.1. Proceso de desarrollo

La metodología utilizada es una **hibridación** entre la metodología tradicional del **Modelo Cascada en V** y la metodología ágil **Kanban** 2.1.

Se utilizará esta mezcla específica de metodologías para poder aprovechar la calidad, precisión y fiabilidad del Modelo en V al ser estricto en las pruebas y la agilidad y flexibilidad provista por la metodología Kanban con su representación visual de las tareas añadiendo posible adaptabilidad.

[6]La metodología del **Modelo en V**, variante del modelo Cascada, es una metodología **tradicional**: se basa en **enfoques secuenciales** y **estructurados** para el desarrollo de software donde cada fase del proyecto debe completarse antes de pasar a la siguiente.

Su principal y más diferenciada característica es su **enfoque en la verificación y validación de cada fase** del desarrollo a través de pruebas específicas. Estas pruebas están estructuradas en el tiempo de desarrollo de manera inversa a las fases que valida lo que forma en una representación visual una V; el lado izquierdo representa las fases de estudio y diseño de la solución, la parte inferior está representada por la fase de programación donde es implementada y el lado derecho representa las fases de verificación y validación con sus respectivas pruebas 2.1.

Kanban es una metodología que utiliza un **sistema visual** para **optimizar el flujo de trabajo**, mejorar la productividad y garantizar que las tareas se completen eficientemente.

Sus elementos principales son el **tablero** y las **tarjetas Kanban**: un tablero visual que representa el flujo de trabajo dividido en columnas que indican el estado de las tareas

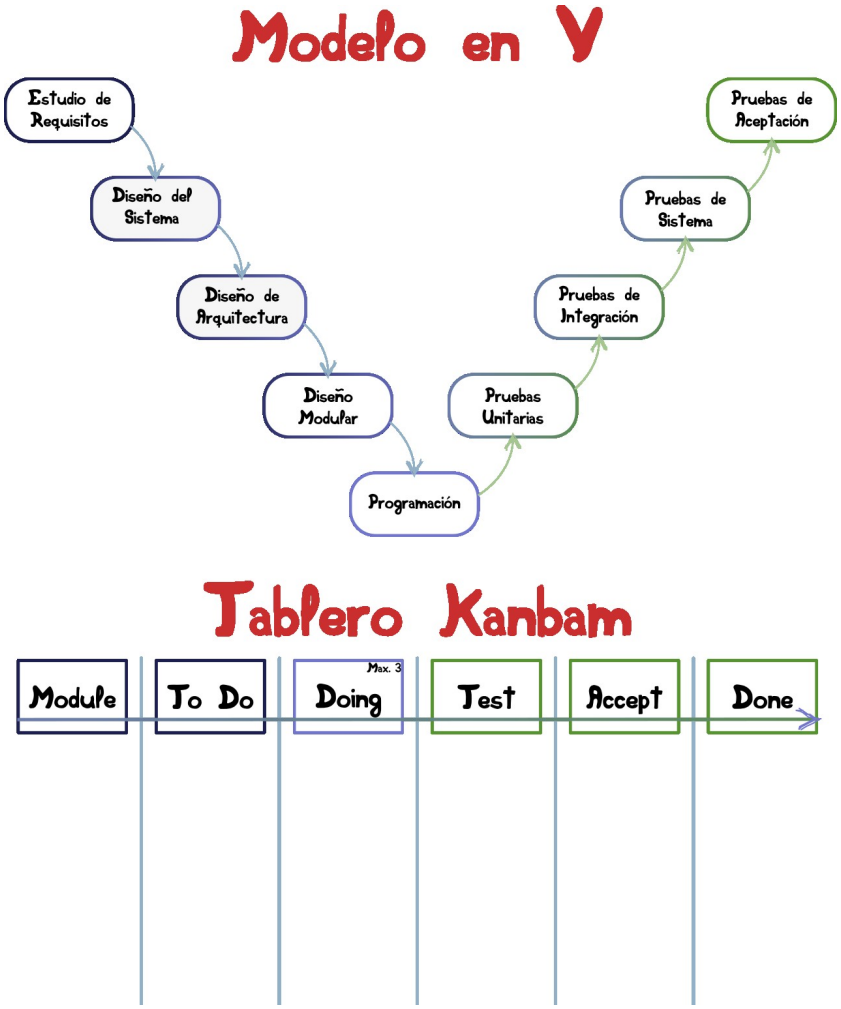

Figura 2.1: Representación gráfica de la metodología seguida

y unas tarjetas que representan cada tarea y que incluye información para detallarla lo máximo posible.

Como principales características destacan los **límites de trabajos en proceso** (en una columna no puede haber un número de tareas mayor al especificado para evitar sobrecargas y poder mantener el foco), el **flujo de trabajo continuo** (no hay ciclos de trabajo predefinidos) y la **mejora continua** donde se ajustan las prácticas usadas a lo largo del trabajo para maximizar la eficiencia y la productividad 2.1.

Teniendo en cuenta las características individuales de estas metodologías, se preparó la **metodología híbrida** que se utilizará en el proceso de desarrollo de este TFG. En esta metodología híbrida, las **fases del Modelo en V se mantienen**, pero el **flujo de trabajo** de las fases se gestiona con un **tablero Kanban**:

1. **Definición de requisitos**.

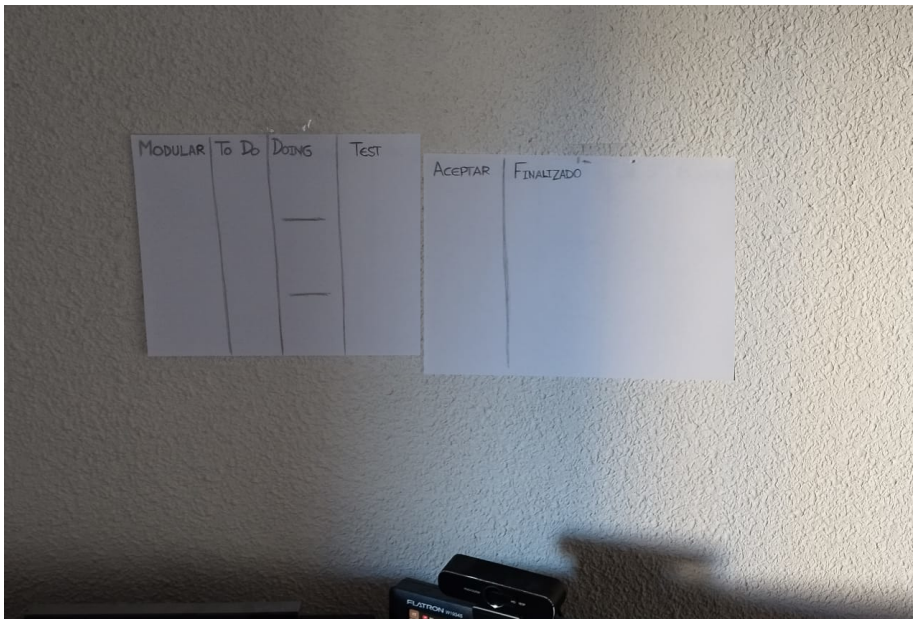

Figura 2.2: Tablero Kanban físico

En esta fase se estudiarán y definirán completamente los requisitos creando una **tarjeta específica e individual para cada requisito**.

Cuando la tarjeta esté lista, se añadirá a la columna del tablero Kanban "*Modular*" 2.2.

2. **Diseño del sistema, de la arquitectura y diseño modular**.

   A partir de las tarjetas creadas con los requisitos, se **modularán las tarjetas correspondientes del sistema y de la arquitectura** representando cada componente o módulo que va a ser implementado. Cada una de estas tarjetas tendrán como parte de su descripción la referencia numérica de la tarjeta del requisito correspondiente.

   Cuando la tarjeta del requisito haya sido modulada por completo, se moverá fuera del tablero; las tarjetas correspondientes a esos módulos se pondrán en el tablero en la columna "*To Do*" (cuya traducción es "*Por Hacer*") 2.2.

3. **Implementación**.

   Usando las tarjetas de módulos creadas, se **implementarán en código siguiendo el flujo de trabajo** del tablero moviendo las tarjetas a la columna "*Doing*" ("*En progreso*") 2.2. Se ayudará a mantener el foco en estas tareas teniendo un **máximo de 3 tarjetas** a la vez en esta columna.

   Una vez se implemente el módulo, se moverá la tarjeta a la columna de "*Test*" a la espera de que todos los módulos sean implementados por completo.

4. **Verificación**.

Se probarán las distintas partes del sistema realizando **pruebas unitarias a cada módulo** implementado, **pruebas de integración** para comprobar que la **comunicación entre los módulos** funciona correctamente y **pruebas del sistema** comprobando que todo funciona conjuntamente para conseguir el **objetivo funcional**.

Según vaya siendo comprobado cada módulo y sistema, se moverán fuera del tablero las tarjetas correspondientes y se pondrá en la columna "*Accept*" (o "*Aceptar*") 2.2 el requisito referenciado por esas tarjetas. En caso de no pasar estos los tests, se deberán retroceder todas las tarjetas necesarias a la columna "*Doing*" (o "*To Do*" en caso de sobrepasar las 3 máximas) para realizar todos las correcciones pertinentes.

5. **Validación**.

   Se **comprobarán todos los requisitos iniciales** realizando **pruebas de aceptación**.

   Cuando un requisito supere las pruebas pasará a la columna "*Done*" ("*Finalizado*") 2.2. En caso de no pasar las pruebas, se deberá realizar una nueva tarjeta o modificar una anterior para solucionar el error y ser colocada en la columna "*Doing*" o "*To Do*" según corresponda por la capacidad de la columna.

- Las **características** aportadas por **Kanban**, a parte de la **visualización del progreso del trabajo**, son los **límites de las tareas** para un mayor equilibrio y una **gestión continua** que identificará y resolverá cuellos de botella en tiempo real y adaptará las prioridades de las tareas sin interrumpir la estructura general del Modelo en V.

- Además, con este enfoque híbrido, también mantendrá la **documentación exhaustiva** que aporta la estructura del **Modelo en V** donde al principio de cada fase se irá completando la documentación formal, lo que complementa al enfoque ágil y visual de Kanban de sus tarjetas.

- En el caso de las **tarjetas Kanban** de esta metodología híbrida tendrán la siguiente información 2.3:

  - **Una etiqueta identificativa**. Las tarjetas de requisitos tendrán una estructura **RXX** siendo las X números del 01 al 99 y las tarjetas moduladas una estructura de **MXXX** siendo las X números del 001 al 999.
  - **Un título descriptivo**. Título de extensión descriptiva que dé una **información superficial, pero específica** de la tarjeta. Se utilizará la documentación exhaustiva para completar más información haciendo uso de la referencia en el documento escrito.
  - **Una lista de etiquetas de referencia**. Grupo de **etiquetas hijo** en el caso de una tarjeta requisito o grupo de **etiquetas padre** en el caso de las tarjetas módulo (un mismo modulo puede ser necesario para cubrir dos requisitos).

- **Una estimación de dificultad numérica**. **Estimación numérica** del 0 al 5 en expresada en numeración quinaria (un sistema de marcas de conteo) para poder ir añadiendo **dificultad** a lo largo de la implementación en caso de ser necesario. Con este número se ayudará al número límite de tarjetas simultáneas para poder trabajar en distintas dificultades al mismo tiempo.
- **Una estimación de necesidad**. **Estimación numérica** parecida a la dificultad. En este caso, el número nos **expresará la prioridad** que se debe de dar a la tarjeta. Esta necesidad se dará a las tarjetas hijo y estas tarjetas podrán sumar necesidades de tarjetas padre hasta llegar a la necesidad 5.

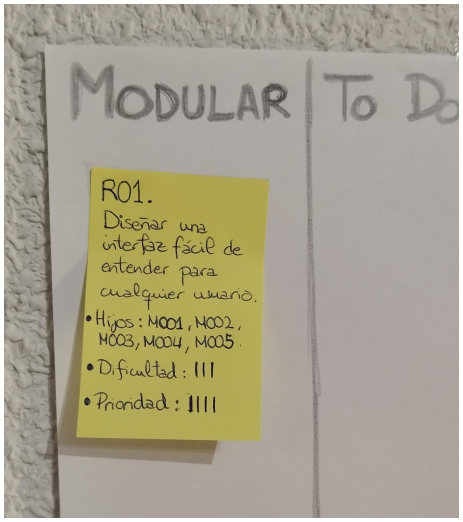

Figura 2.3: Tarjeta del Kanban físico

- Por último, el **control de calidad sólido** provisto por el Modelo en V se verá **potenciado y ayudado** de la **adaptabilidad** provista por Kanban y su posibilidad de **ajustar tareas y prioridades** llegando a retroceder de fase las tareas si estas necesitan arreglos.

2.2. Herramientas utilizadas

Para la realización de este TFG se han utilizado muchas herramientas, aplicaciones, sistemas y lenguajes de programación. Están listadas junto con la razón por la que se utilizaron y, en algunos casos, con la justificación frente a otros sustitutivos.

- Como **SO** (Sistema Operativo) se eligió **Windows 10** por facilidad, recursos y familiaridad. Se barajó la posibilidad de hacer una instalación completa de una distribución de Linux para tener un mayor control de los recursos o incluso realizar todo el proyecto dentro de una máquina virtual, pero se acabó decantando Windows 10 como SO para evitar complicaciones a mayores y para aprovechar la maestría en cuanto a resolución de errores adquirida durante años.

- Para el ***estado del arte*** se utilizaron las herramientas de **Adobe Photoshop, Adobe Illustrator** e Inkscape. A mayores, **Inkscape** se ha utilizado a lo largo de todo el TFG para **realizar** todas las **imágenes**; las dos excepciones son la imagen del producto que se trabajó con **Penpot**[1], herramienta open-source que funciona nativamente en navegador, y el diagrama de Gantt que se utilizó **Microsoft Excel** para agilizar el orden de los bloques. Por otra parte, para la **edición de las imágenes** se utilizó **GIMP**[2], otra herramienta open-source especializada en edición de gráficos rasterizados (imágenes formadas por matrices de píxeles organizados en cuadrícula en vez de usar vectores).

- **Visual Studio Code** junto con la extensión de **LaTeX Workshop** (que permite una integración inmediata del lenguaje *LaTeX* junto con una construcción del archivo casi inmediata) fueron utilizados para realizar toda la **documentación y memoria**. Su alternativa habría sido hacer uso de Google Docs, Word Microsoft Office o Libre Office, pero por problemas pasados con maquetaciones extrañas de esos programas se decidió realizar la documentación con un enfoque más programático para poder ofrecer más automatización, reproducibilidad, flexibilidad y escalabilidad.

- El **navegador** elegido para la búsqueda de información fue **Brave** con su integración de **Tor** pues aúna en una misma aplicación las características y personalización de Chromium[3] (projecto de navegador web de código abierto apoyado por Google usado en Google Chrome) y la seguridad provista por la enrutación de Tor[4]. A mayores, cuenta con bloqueador de anuncios y bloqueador de sitios patrocinados completamente integrados que ayudan a una búsqueda en la red más eficiente y eficaz.

- La **aplicación** documentada en este TFG se ha programado en su totalidad en **Qt Creator** haciendo uso del lenguaje de **Qt** en **C++**. La elección del lenguaje de Qt fue por curiosidad y ganas de aprender todo lo posible después de haber programado superficialmente anteriormente a través de asignaturas optativas de la carrera. La elección del entorno de programación vino condicionado por la elección del lenguaje pues Qt Creator es el entorno mejor preparado para la escritura de código en este lenguaje. [7][8]Las dos opciones que se barajaron fueron: hacer uso de Qt en Python usando Visual Studio Code o usar Qt Creator y C++; después de una investigación a través de usuarios experimentados se eligió Qt Creator con C++ para poder tener **disciplina** a la hora del **uso de la memoria**, una mejor **preparación de aplicación**, una **mayor organización** y la posibilidad de usar la **herramienta Qt Designer** a la hora de programar la interfaz gráfica (herramienta disponible de manera inmediata e integrada en Qt Creator).

[1] `https://www.penpot.app`
[2] `https://www.gimp.org`
[3] `https://www.chromium.org/Home`
[4] `https://www.torproject.org`

Qt Creator permite elegir **contructor** entre dos posibles: CMake y QMake. Aún siendo CMake el más recomendado, a través de unas primeras pruebas iniciales de conceptos se eligió **QMake** al estar más consolidado y al ser más estable. CMake dio problemas en estas pruebas con requisitos básicos necesarios para la aplicación como la modificación superficial de los objetos básicos de Qt (promocionar objetos para añadir modificaciones daba errores dependiendo de las versiones de Qt de manera aleatoria).

Para realizar las vectorizaciones se hará uso de **VTracer** como libreria adicional (habiendo compilado el programa como librería para Python usando **Rust** y **Python** embebido en la aplicación). VTracer es un software Open Source construido en Rust que permite convertir las imágenes rasterizadas en imágenes vectoriales. Rust es un lenguaje de programación gratuito y de código abierto de alto rendimiento y seguro diseñado para evitar errores de memoria sin necesidad de un recolector de basura. Además, aprovechando que se utiliza Python embebido, se utilizó **OpenCV** como librería en Python para realizar los operaciones morfológicas de limpieza de imágenes.

- Se hizo uso de **GitHub** y su herramienta **GitHub Desktop** para hacer **copias de seguridad** periódicas durante el progreso de la aplicación. También se hicieron uso de **Google Drive**, **Whatsapp + Whatsapp Web** y **OneDrive** para **compartir archivos** entre dispositivos. Además, Whatsapp junto con **Instagram Messenger** y **Discord** se usaron para comunicarse con **asesores y usuarios interesados** durante la toma de requisitos y **testers** durante las pruebas de aceptación.

- Para la representación gráfica del **Tablero y las Tarjetas Kanban** se acabó utilizando una **perspectiva analógica**. La opción digital estudiada fue la aplicación web Trello, pero no resultó lo suficientemente dinámica ni satisfactoria como la opción física del concepto. Además, poder tenerlo a la vista de manera instantánea sin necesidad de cambiar la pantalla fue la razón que más peso supuso en la decisión.

2.3. Arquitectura

Al tratarse de una aplicación de escritorio, la **arquitectura física** 2.4 que se seguirá en el desarrollo será **simple** y tendrá todos sus componentes en el cliente. Todos estos componentes se comportarán de manera conjunta en el **ejecutable comunicándose con librerías internamente** evitando un requerimiento de programas adicionales.

La única **conexión al entorno externo** de la aplicación será la **lectura de la imagen a tratar** y la **escritura de la imagen resultante**. De manera completamente ***opcional***, se permitirán lecturas y escrituras de *perfiles de configuración* de la aplicación y lecturas y escrituras de *archivos de sesiones de guardado*.

Para la **arquitectura lógica** se seguirá el **patrón de diseño MVC** (siglas correspondientes a Modelo-Vista-Controlador) a la hora de la programación donde separaremos

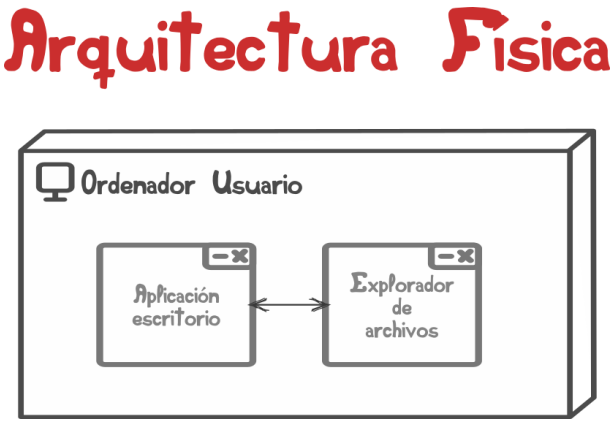

Figura 2.4: Diagrama visual de la arquitectura física.

el código para mayor comprensión, orden y control 2.5.

[9] Este patrón de diseño se caracteriza por enfatizar en una **separación entre la lógica de negocio y su visualización** para proporcionar una mejor división del trabajo y una mejora sustancial a su mantenimiento. Las tres partes que se separan son:

1. **Modelo** Mantendrá toda la **lógica de negocios** fácilmente visible y distinguible.

   Define la estructura interna que seguirá la aplicación y qué datos contiene junto con su tipo.

   Por ejemplo, la aplicación utilizará un booleano llamado "imagenRecortada".

2. **Vista** Tendrá todo el **diseño**, presentación y embellecimiento de la aplicación.

   Define qué objetos visuales se presentarán, cuándo lo harán, cómo aparecerán y dónde.

   Por ejemplo, la aplicación tendrá un botón situado en la esquina superior derecha con un texto explicativo cuando se mantenga el ratón por encima.

3. **Controlador** Manejará los **datos** entrantes y salientes de la aplicación y **enrutará los comandos** entre los modelos y las vistas.

   Contiene la programación que actualiza el valor de los datos del modelo y actualiza la vista. A mayores, también controla todos los archivos externos en lectura y escritura y todos los recursos necesarios del programa.

   Por ejemplo, la aplicación tendrá una función que se ejecutará cuando en la vista se haga clic en un botón específico. Esta función cambiará el valor de "imagenRecortada" a "true" que actualizará la vista como sea necesario.

Se seguirá esta arquitectura apoyado por la **estructura** programativa **de ficheros construida por Qt Creator**. Esta estructura divide los ficheros por carpetas y especificando diferentes extensiones de archivo:

1. Una carpeta llamada *"Headers"* (*cabeceras*) donde se muestran archivos con extensión ".h".

   En estos archivos dividiremos la parte del **Modelo** de la aplicación.

2. La carpeta de los *"Forms"* (que viene, en este contexto, de la palabra inglesa *formas*) con archivos de extensión ".ui".

   Estos ficheros contendrán el diseño de la **Vista**.

3. Y el directorio *"Sources"* (*fuentes*) que enseña los archivos con extensión ".cpp".

   Estos archivos tendrán todas las funciones necesarias del **Controlador**.

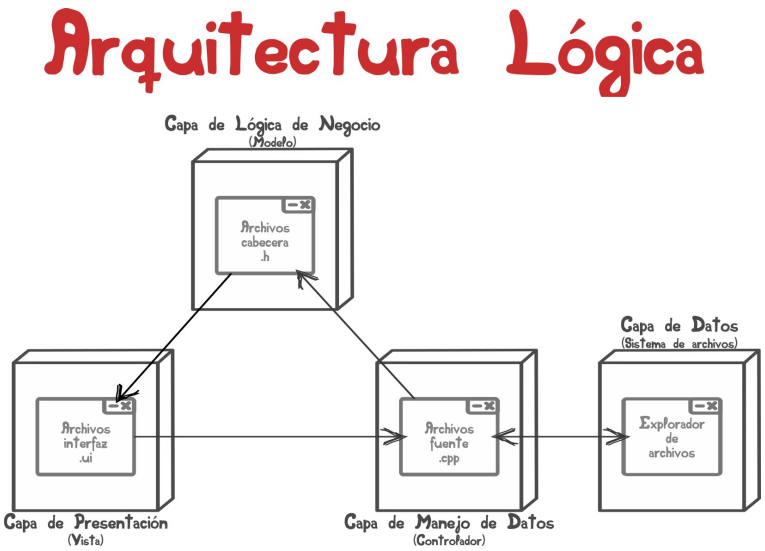

Figura 2.5: Diagrama visual de la arquitectura lógica fusionada con el patrón MVC.

En cuanto a **paradigma de programación**, el código se escribirá siguiendo una **programación orientada a objetos** (POO).

El paradigma POO es un modelo de desarrollo que se basa en **organizar el código con atributos y funciones**. Permite el **encapsulamiento** y la **abstracción** para proteger información interna de los datos frente a lecturas externas y, también, añade posibilidad de **herencias** de clases para la reutilización de código junto con la posibilida del **polimorfismo** (tener el mismo método, pero con diferentes comportamientos dependiendo de la clase).

Qt potencia estas características con sus clases base *QObject* y *QWidget* para la creación de componentes personalizados y amplia sus capacidades con el *Meta-Object Compiler* que gestiona señales y ranuras. La **gestión de señales y ranuras** es un mecanismo que permite la comunicación entre objetos sin necesidad de dependencias directas.

Además el código seguirá el paradigma derivado del POO de **Component-Based Development** (CBD) o Desarrollo Basado en Componentes.

En el CBD se divide el desarrollo en componentes reutilizables e independientes donde cada componente puede desarrollarse y probarse de manera individual.

El uso de esta variante del paradigma de POO se compenetra a la perfección con la metodología utilizada durante todo el procedimiento y ayudará a que el desarrollo avance de una manera más orgánica, sin pausa y ordenada.

2.4. Diseño de interfaz

Para el diseño de interfaz se tendrá en mente durante todo el proceso las siguientes **dos ideas claves** del proyecto: el usuario tendrá el **completo control en la personalización** de la aplicación cuando lo desee y **cualquier usuario**, sin depender del nivel de profesionalidad del usuario, **podrá manejar sin dificultad la aplicación** de manera rápida, simple y controlando su experiencia.

Con estas ideas definidas, se diseñará la aplicación haciendo uso de la herramienta **Qt Designer** para mayor agilidad visual de la UI (User Interface, Interfaz de Usuario) final de la siguiente forma:

1. **Tendrá un diseño global predeterminado cómodo y con personalidad**.

   El diseño básico mantendrá una **paleta de colores** para ayudar a la personalidad de la aplicación 2.6. De esta manera, un usuario principiante sabrá la **cohesión de la aplicación y sus límites**.

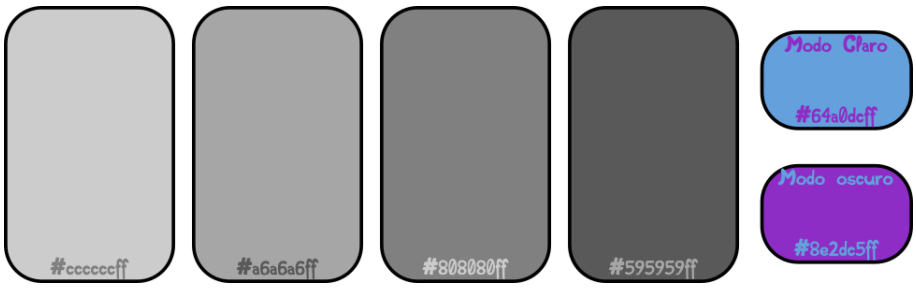

Figura 2.6: Paleta de colores utilizados en la aplicación.

   A mayores, se diseñó un **logotipo fácilmente identificativo** y familiar 2.7 para ayudar a su diferenciación y que se usará como **cursor dentro de la aplicación** 2.8 para que los usuarios inexpertos tengan aún más claros los límites físicos de la UI. Este logotipo se diseñó con la forma de la inicial del nombre de la aplicación Vectorizart (V) añadiendo pequeñas modificaciones para asemejarse a la punta de una pluma estilográfica. Para su uso como cursor se girará el logotipo para que sea mucho más fácil de hacer clic con él al tener la misma forma que un cursor predeterminado de sistema operativo.

Figura 2.7: Logotipo de la aplicación.

Figura 2.8: Cursor que se puede utilizar en la aplicación.

Toda la UI **estará en inglés**. La elección del inglés como idioma viene dado por la comodidad y generalidad que da el idioma al estar más globalizado y al estar mucho más difundido en Internet.

2. **Estructuralmente se seguirá un enfoque lineal completo**.

   Para estructurar la aplicación se seguirá una adaptación **guiada por pasos** 2.9, como los programas de instalación. Este enfoque ayudará a todos los usuarios a entender de manera guiada todos los apartados del programa, todos los parámetros necesarios y todo lo que pueden llegar a realizar usando el programa. Además, se les ayudará visualmente a entender el porcentaje de completación del proyecto al poder ver los pasos pasados y futuros con un menú no interactuables (pero explicativos) de pestañas.

3. **Personalizaciones y configuraciones**.

   De manera paralela a todos los nuevos usuarios y personas poco acostumbradas al mundo digital, se propondrán configuraciones y personalizaciones:

   - Posibilidad de **cambios de temas** entre claro y oscuro a placer.

     El **cursor** también será **intercambiable** entre otros colores predeterminados y podrá elegir mantener el cursor predeterminado del SO.

   - A través de las configuraciones, el usuario podrá elegir **mantener un sistema de log** para mayor rapidez y fluidez a la hora de pedir ayuda externa.

     Del mismo modo, podrá **exportar e importar configuraciones** de otros usuarios. Esto ayudará a los usuarios a mantener grupos de trabajo con la

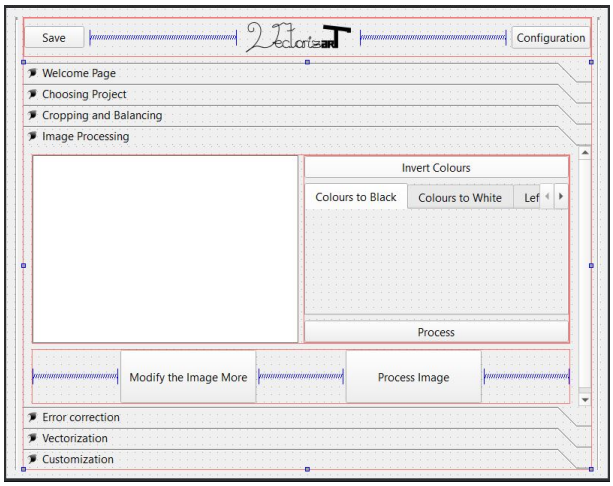

Figura 2.9: Ilustración de un estado primitivo de la estructura de la aplicación en Qt Designer.

misma cohesión o distintos dispositivos con la misma personalización.

Capítulo 3

Planificación

En este capítulo se explicarán las cuestiones relativas a la planificación de todo el TFG siguiendo con la metodología híbrida explicada en el capítulo anterior.

Para una mejor y más detallada organización y explicación, en la sección de planificación temporal se tendrá en cuenta **todo lo relativo al TFG**, siendo incluidas tareas como la realización de esta documentación y **no solo su programación**.

3.1. Estimación del esfuerzo

Se hará uso del método de estimación por **Puntos de Casos de Uso** (UCP por sus siglas inglesas, "Use Case Point") pues permite calcular el tamaño del software, y con ello estimar el esfuerzo, basándose en el análisis de casos de uso. Estos casos de uso son fácilmente extrapolables a tarjetas Kanban y viceversa, por lo que haciendo uso de esta estimación haremos congeniar la metodología y la planificación de manera perfecta.

[10] Este método proporciona una estimación del esfuerzo requerido para el desarrollo de un proyecto de software basado en la complejidad de sus casos de uso y sus factores técnicos y ambientales. Esta estimación mediante UCP sigue el siguiente proceso para su cálculo final:

1. **Cálculo de los Puntos de Casos de Uso sin ajustar** (Unadjusted Use Case Points o UUCP)

   Este cálculo es el resultado de la **suma** de los pesos por el tipo de actor y su número de transacciones de todos los casos de uso 3.1.

   - El **peso de los actores** [UAW - Unadjusted Actor Weight] se reparten como: **Actor Simple** (Peso 1. Cuando otro sistema interactúa con el caso de uso, como en el uso de librerías o APIs), Actor Medio (Peso 2. Cuando otro sistema interactúa con el caso de uso a través de un protocolo de Internet, en esta aplicación no hay ningún Actor Medio) y **Actor Complejo** (Peso 3. Cuando es una persona la que interactúa usando la interfaz, cualquier modificación de la imagen rasterizada tendrá este actor).

- El **peso de las transacciones** [UUCW - Unadjusted Use Case Weight], interacciones completas dentro de un caso de uso, se reparten como: **Simple** (Peso 5. Menos de 3 transacciones), **Medio** (Peso 10. De 4 a 7 interacciones) y **Complejo** (Peso 15. Más de 7 transacciones).

(Las casos de uso y su diagrama están explicados en el apartado correspondiente de la Documentación técnica)

Caso de Uso	Peso de Actores	Peso de Transacciones
Abrir proyecto	Complejo: 3	Simple: 5
Abrir imagen	Complejo: 3	Simple: 5
Guardar proyecto	Complejo: 3	Simple: 5
Girar imagen	Complejo: 3	Medio: 10
Cortar imagen	Complejo: 3	Medio: 10
Arreglar colores de la imagen	Complejo: 3	Complejo: 15
Arreglar errores de la imagen	Complejo: 3	Complejo: 15
Vectorizar imagen	Simple: 1	Medio: 10
Personalizar imagen vectorizada	Complejo: 3	Complejo: 15
Exportar imagen	Complejo: 3	Medio: 10
Configurar programa	Complejo: 3	Complejo: 15

Cuadro 3.1: Casos de Uso del TFG con sus pesos asociados.

$$UUCP = \sum UAW + \sum UUCW \tag{3.1}$$

$$UUCP = 31 + 115$$

$$UUCP = 146$$

2. **Ajuste por factores de complejidad técnica** (acortado como TCF por Technical Complexity Factors)

   Se evalúan los **factores técnicos** que pueden influir en la complejidad del sistema y se valoran en una escala del 0 (irrelevante) al 5 (esencial).

   Para saber el peso que se debe dar a cada factor, se seguirá la tabla 3.2, en esta tabla también está recogido el valor que se le ha correspondido a cada factor para este TFG:

Factor Técnico	Peso	Valor	Resultado (valor x peso)
Distribución del sistema en múltiples plataformas	2	1	2
Rendimiento crítico	1	5	5
Procesamiento interno complejo	1	5	5
Código reutilizable	1	5	5
Instalación fácil	0'5	5	2'5
Fácil utilización del sistema	0'5	5	2'5
Múltiples interfaces externas	1	5	5
Interfaces de usuario complejas	1	0	0
Alta seguridad	1	2*	2
Acceso directo a terceros	1	0	0
Disponibilidad total	1	0*	0
Altos volúmenes de datos	1	2	2
Formación especial para los usuarios	1	0	0

Cuadro 3.2: Reparto de pesos y valores para el ajuste por TCF.
* Al tratarse de una aplicación de escritorio cerrada, la seguridad y la disponibilidad depende completamente del usuario. Igualmente, se añadirá un sistema de log por seguridad.

Se calcula el TCF completo a través de la siguiente fórmula 3.2:

$$TCF = 0{,}6 + (0{,}01 * \sum(valor * peso)) \qquad (3.2)$$

$$TCF = 0{,}6 + (0{,}01 * 29)$$

$$TCF = 0{,}89$$

3. **Ajuste por factores ambientales** (Environmental Factors, EF)

   Para este cálculo se calificarán del 0 (sin experiencia o malo) al 5 (experto o muy bueno) **factores relacionados con la experiencia y habilidades** del equipo de desarrollo. En la tabla 3.3 están los pesos asociados para estos factores junto con los valores que aplicarán en este TFG.

Factor Ambiental	Peso	Valor	Resultado (valor x peso)
Experiencia en el lenguaje de programación	1'5	3	4'5
Experiencia en la aplicación del sistema	0'5	4	2
Experiencia en la metodología usada	1	3	3
Motivación	1	5	5
Estable, sin rotación de personal	1	5	5
Experiencia en las herramientas CASE	0'5	4	2
Experiencia en las plataformas de desarrollo	2	3	6
Tiempo de entrega estricta	-1	1	-1
Usuarios familiarizados con el sistema	1	0	0
Distribuido en diferentes ubicaciones	-1	0	0

Cuadro 3.3: Reparto de pesos y valores para el ajuste por EF.

Se calcula el ajuste EF final con la fórmula 3.3:

$$EF = 1{,}4 + (-0{,}03 * \sum(valor * peso)) \tag{3.3}$$

$$EF = 1{,}4 + (-0{,}03 * 26{,}5)$$

$$EF = 0{,}605$$

4. **Cálculo de los UCP ajustados** (UCP).

   Se obtiene el cálculo final con la multiplicación directa de los elementos calculados en los anteriores pasos entre ellos.

$$UCP = UUCP * TCF * EF \tag{3.4}$$

$$UCP = 146 * 0{,}89 * 0{,}605$$

$$UCP = 78{,}6137$$

Además, se puede calcular la cantidad en horas por persona para tener más claro todo el tiempo necesario basándose en el **Factor de Productividad** (PF, Productivity Factor).

Este PF viene determinado por el **nivel que tenga el equipo**: Muy experimentado - 10 horas/UCP, Nivel intermedio - 15 horas/UCP y Baja experiencia - 20 horas/UCP. Cuando no se tiene un valor específico, como es el caso al ser completamente desconocido, se toma 20 horas/UCP como referencia.

El cálculo de horas se realiza multiplicando los UCP calculados por el PF del equipo:

$$Esfuerzo = UCP * PF \tag{3.5}$$

$$Esfuerzo = 78{,}6137 * 20$$

$$Esfuerzo = 1572{,}274\ horas$$

3.2. Planificación temporal

Para poder mantener la planificación del TFG ordenada y visual se hará uso de un **diagrama de Gantt**. Este nos ayudará a tener el alcance total del proyecto desde el principio ayudado por el esfuerzo calculado en el anterior apartado (para la actividad "programación").

[11] El diagrama de Gantt es una herramienta de gestión de proyectos en el que **visualizar** las actividades de un proyecto **a lo largo de un gráfico temporal**. Ayudará a mantener un **seguimiento del proceso completo** y a **gestionar las dependencias** de las actividades apoyándose en el orden a seguir por la metodología del Modelo en V.

Una planificación temporal completamente estricta supone un problema de compatibilidad con la metodología Kanban por lo que en el diagrama de Gantt utilizado para este TFG no se especificará cuando se programará cada Caso de Uso/Tarjeta Kanban. Al haber hecho esta distinción, podemos aprovechar el esfuerzo en horas calculado y poner una actividad que cubra esa cantidad de horas específica.

Debido a la inconsistencia que pueden causar las fechas específicas por contratiempos, el diagrama usará en su eje horizontal **tiempo en días de trabajo "ininterrumpido"**. De este modo, el espacio que separa cada línea vertical será equivalente a 24 horas de trabajo que pueden realizarse en distintos días. De manera completamente paralela, las **sesiones de trabajo** en las actividades serán siempre de un **mínimo de 2 horas** y, sin tener en cuenta pequeñas pausas de descanso, no se dejará el trabajo hasta **terminar secciones fácilmente separables** para evitar "olvidar el hilo" durante las horas de trabajo en la actividad 3.1.

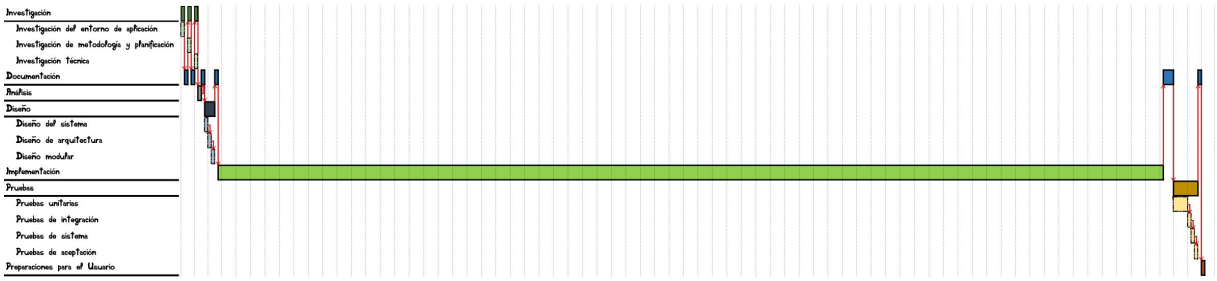

Figura 3.1: Diagrama de Gantt con sus diferentes actividades.

Como resultado de una actividad de "Implementación" tan larga, resulta complicado visualizar el conjunto. Durante el desarrollo, se hará uso de este primer diagrama, pero para que sea fácilmente visible se ha realizado una modificación en la perspectiva para que quepa en la página de la documentación 3.2:

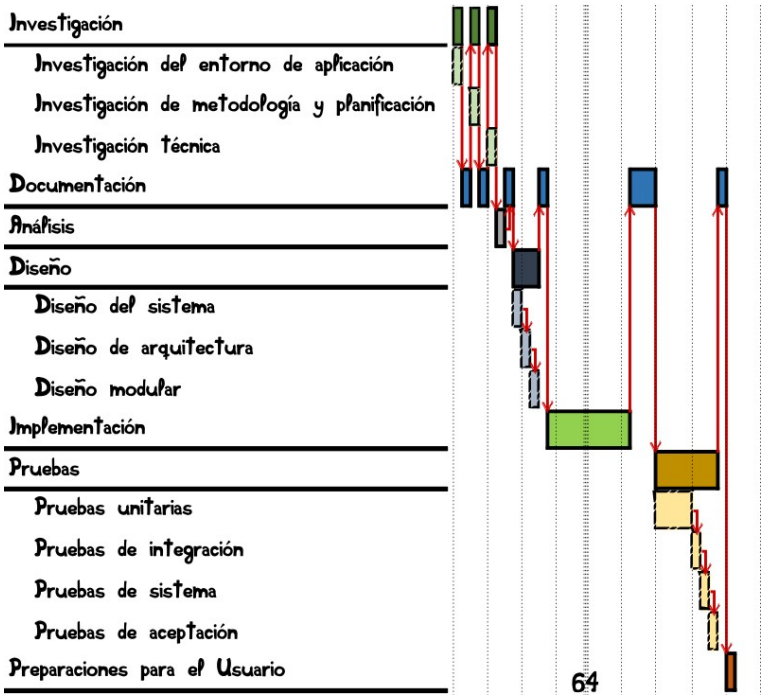

Figura 3.2: Diagrama de Gantt con sus diferentes actividades redimensionado para fácil lectura en la documentación.

La planificación realizada resulta en un total de **1710 horas** repartidas en:

- 3 bloques de 6 horas (**18** en total) para la "**Investigación**" separados entre ellos por bloques de "**Documentación**" para plasmar lo investigado en esta memoria (2 bloques de 6 horas cada uno; **12** horas en total).

  Esta investigación es para averiguar el entorno de la aplicación con sus productos sustitutivos, aprender sobre la mejor metodología y planificación en compatibilidad

entre ellas y con la aplicación, y, por último, investigar sobre la mejor tecnología a utilizar y sobre el entorno en el que se programará durante la implementación.

- 1 bloque de **6** horas para el "**Análisis**" de requisitos de la aplicación con su consiguiente bloque de **6** horas de "**Documentación**".

- 3 bloques de "**Diseño**" (**18** horas) seguidos de su "**Documentación**" (**6**) donde se diseñará el sistema, la arquitectura y la modularidad de los requisitos analizados y transcritos como tarjetas Kanban.

- 1 bloque de las estimadas **1572** horas de "**Implementación**" y su respectivo bloque de "**Documentación**" de **16** horas correspondientes a comentar todo el código.

- 1 bloque de "**Pruebas**" separado en 4 (en los que se cuentan los posibles errores encontrados y su tiempo de solución): pruebas unitarias de 24 horas de duración, pruebas de integración (6 horas), pruebas de sistema de 6 horas y pruebas de aceptación con usuarios potenciales de otras 6 horas (un total de **42** horas de pruebas). Estas pruebas estarán seguidas del último bloque de "**Documentación**" de **6** horas.

- Por último, se hará 1 bloque de **6** horas para las "**Preparaciones para el usuario**" donde se realizará el ejecutable de la aplicación y los manuales de usuario en los que se explicarán las bases del programa.

3.3. Presupuesto económico

Para realizar una estimación del precio que supondría este proyecto se tendrá en cuenta un **periodo ficticio** en el que todas las horas trabajadas en el proyecto supondrán un **horario de trabajo completo de 8 horas por día, 5 días a la semana**. En este proyecto se tiene a solo una persona haciendo todos los roles, pero en estas mismas suposiciones tendremos a un **equipo de personas** que harán cada uno su **posición de trabajo** teniendo cada uno su sueldo durante el tiempo que tardarían en hacer estas actividades.

Aun así, sí que se tendrá en cuenta las **herramientas reales, hardware y software utilizados** durante el proyecto.

3.3.1. Hardware y software

Valoración del hardware y software utilizado en el proyecto 3.4.

Producto	**Precio**	**Total**
Hardware. Portatil LENOVO Yoga 7i 2-in-1*	0’0808 €/h	138’09 €
Hardware. Carga del Portatil**	0’000648 €/h	1’11 €
Hardware. Conexión a Internet móvil***	0’265 €/GB	25’175 €
Hardware. Tablero y Tarjetas Kanban	5 €	5 €
Software. Sistema Operativo. Windows 10	Incluido en Portatil	0 €
Software. Copias de Seguridad. GitHub.	Gratis	0 €
Software. Pruebas del Estado del Arte	Pruebas gratuitas	0 €
Software. Inkscape, GIMP y Penpot	Open Source	0 €
Software. Microsoft Excel	Incluido en Portatil	0 €
Software. LaTeX en Visual Studio Code	Gratis	0 €
Software. Brave y Tor	Gratis	0 €
Software. Qt Creator y librerías****	Open Source	0 €
Total		**169’38 €**

Cuadro 3.4: Presupuesto del Hardware y del Software utilizado.

* Precio de mercado de 1179’01€. Vida útil promedio de 5 años.
** Batería de 4535mAh y 15’36V. 0’0696576kWh. Con 14 horas de trabajo en alto rendimiento por carga. Consumo por hora de 0’00497 kWh/h.
*** Precio de 7’95 € por 30 GB. Se calcula que la instalación de los diferentes softwares y las copias de seguridad, junto con las búsquedas de Internet, han llegado a los 95 GB.
**** Al estar realizando un programa Open Source, la licencia es gratuita. Además, todas las librerías y APIs utilizadas son también Open Source. La construcción de las librerías también fueron Open Source en todo su proceso.

3.3.2. Recursos humanos

[12] Presupuesto previsto por el uso de recursos humanos 3.5.

Trabajo	**Precio por hora**	**Total**
Junior Project Manager	23’42 €/h	1545’72 €
Analista	17’45 €/h	314’1 €
Diseñador software	16’83 €/h	908’82 €
Desarrollador de Software	4’87 €/h	7655’64 €
Tester	14’38 €/h	603’96 €
Total		**11028’24 €**

Cuadro 3.5: Presupuesto de los recursos humanos utilizados.

1. Al trabajo de **Project Manager** se le han impuesto las horas de trabajo de investigación del entorno y de metodología y planificación junto con la preparación para el usuario y todas las horas relativas a la documentación. El total de horas son: 66.

2. **Analista**. Su trabajo serán las horas de análisis e investigación técnica y las pruebas de aceptación. 18 horas.

3. El **diseñador** hará las horas dedicadas al diseño del sistema, arquitectura y modular junto con las horas de pruebas de sistema, de integración y unitarias. Un total de 54 horas.

4. Las horas computadas al **desarrollador** han sido solo las de implementación pues todas las relativas a las pruebas se realizan bajo el puesto de tester. 1572 horas estimadas por Casos de Uso.

5. **Tester**: Repitiendo horas con las pruebas del resto de puestos al ser necesario para arreglar los errores. 42 horas en total.

3.3.3. Presupuesto total

Teniendo en cuenta los cálculos realizados en las secciones anteriores, la estimación a la que se ha llegado resultaría en un coste de 11.197'62 €. 3.6

Presupuesto	**Precio**
Hardware y Software	169'38 €
Recursos humanos	11028'24 €
Total	**11197'62 €**

Cuadro 3.6: Presupuesto total estimado de todos los recursos utilizados.

Parte II

Documentación técnica

Capítulo 4

Análisis

4.1. Requisitos

En este apartado se recogen los **requisitos funcionales** (RF) y no funcionales (RnF) que definen el **comportamiento de la aplicación desarrollada**, así como los distintos **casos de uso derivados** de los requisitos que expanden y guían su diseño y funcionalidad.

Los requisitos y casos de uso aquí expuestos corresponden al refinamiento hecho en todas sus interacciones durante su vida en el tablero Kanban:

RF01 - Cargar datos de proyecto. El sistema debe permitir al usuario introducir un nombre de proyecto, una carpeta de guardado y una imagen.

RF02 - Guardar proyecto. El sistema debe permitir al usuario guardar el proyecto en su equipo.

RF03 - Cargar proyecto. El sistema debe permitir al usuario hacer uso de un proyecto guardado.

RF04 - Girar imagen. El sistema debe permitir al usuario girar la imagen.

RF05 - Cortar imagen. El sistema debe permitir al usuario cortar la imagen.

RF06 - Arreglar colores de la imagen. El sistema debe permitir al usuario cambiar a placer los colores de la imagen a blanco y negro.

RF07 - Arreglar errores de la imagen. El sistema debe permitir al usuario arreglar aquellas imperfecciones que tenga la imagen usando los scripts de OpenCV.

RF08 - Vectorizar imagen. El sistema debe vectorizar la imagen modificada usando VTracer.

RF09 - Personalizar imagen vectorizada. El sistema debe permitir al usuario eliminar los trazos de la imagen vectorizada.

RF10 - Exportar imagen personalizada. El sistema debe exportar la imagen vectorizada personalizada.

RF11 - Personalizar experiencia. El sistema debe permitir al usuario personalizar la apariencia del programa.

RF12 - Seguridad de historial. El sistema debe mantener un historial de las sesiones del programa a elección del usuario.

RF13 - Configuración exportable. El sistema debe permitir exportar e importar las configuraciones del programa.

RnF01 - Sistema Operativo. La aplicación debe funcionar en sistemas Windows.

RnF02 - Idioma. La interfaz debe estar en inglés.

RnF03 - Open Source. El código fuente de la aplicación estará disponible bajo licencia libre de uso y modificación.

RnF04 - Programación. El código se realizará en una combinación de Qt en C++ y Python.

RnF05 - Tiempo de respuesta. Los procesos del programa deben completarse de manera rápida para imágenes estándar.

RnF06 - Interfaz de Usuario. El diseño de interfaz será claro, personalizable e interactivo.

Nombre e ID del CU	CU01. Cargar datos de proyecto.
Actor	Usuario
Descripción	El usuario carga los datos en el programa para comenzar con un proyecto.
Precondiciones	PRE01. El usuario ha iniciado el programa y un nuevo proyecto.
Postcondiciones	POST01. El proyecto puede empezar.
Flujo Normal	FN01. El usuario introduce el nombre del proyecto. FN02. El usuario introduce la carpeta del proyecto a través del selector de carpetas. FN03. El usuario introduce la imagen a través del selector de archivos. FN04. El usuario pasa al paso siguiente.
Flujo Alternativo 1	FA101. El usuario pasa al paso anterior.
Flujo Alternativo 2	FA201. El usuario puede no introducir el nombre del proyecto ni seleccionar la carpeta del proyecto.
Flujo Alternativo 3	FA301. El usuario puede abrir un proyecto guardado anteriormente y carga los datos automáticamente.

Flujo Alternativo 3.1	FA301.1. El usuario puede reusar un proyecto abierto para usarlo con otra imagen, otro nombre u otra carpeta.
Excepciones	E01. La carpeta seleccionada no es válida. E02. La imagen introducida no es válida. E03. No se ha introducido una imagen.
Prioridad	Alta

Cuadro 4.1: CU01. Cargar datos de proyecto.

Nombre e ID del CU	CU02. Guardar proyecto.
Actor	Sistema
Descripción	El sistema guarda los datos del proyecto.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen.
Postcondiciones	POST01. Un archivo .vectorizarT se guarda en la carpeta del proyecto.
Flujo Normal	FN01. El usuario comienza el guardado. FN02. El sistema genera el JSON con los datos del proyecto. FN03. El sistema guarda el JSON generado con la extensión del programa.
Flujo Alternativo 1	FA101. El usuario no puso nombre para el proyecto y el sistema pondrá un nombre predeterminado.
Flujo Alternativo 2	FA201. El usuario no seleccionó una carpeta para el proyecto y se abre un selector para guardar el archivo.
Excepciones	E01. Cuando no hay carpeta de proyecto y no se selecciona ninguna carpeta, no se realiza ningún guardado.
Prioridad	Media

Cuadro 4.2: CU02. Guardar proyecto.

Nombre e ID del CU	CU03. Girar imagen.
Actor	Usuario
Descripción	El usuario gira la imagen.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen.
Postcondiciones	POST01. La imagen estará girada los grados especificados.

Flujo Normal	FN01. El usuario hace clic en los botones para girar la imagen. FN02. El sistema gira la imagen con los grados seleccionados. FN03. El sistema visualiza la imagen girada en el visor.
Flujo Alternativo 1	FA101. El usuario no gira la imagen.
Prioridad	Media

Cuadro 4.3: CU03. Girar imagen.

Nombre e ID del CU	CU04. Cortar imagen.
Actor	Usuario
Descripción	El usuario corta la imagen.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen.
Postcondiciones	POST01. Imagen recortada.
Flujo Normal	FN01. El usuario realiza un rectángulo sobre la imagen haciendo clic y arrastrando. FN02. El sistema usa ese rectángulo para hacer el recorte en la imagen. FN03. El sistema visualiza el recorte como imagen en el visor.
Flujo Alternativo 1	FA101. El usuario no recorta la imagen.
Excepciones	E01. El usuario gira la imagen tras recortarla (esto resetea la imagen).
Prioridad	Media

Cuadro 4.4: CU04. Cortar imagen.

Nombre e ID del CU	CU05. Arreglar colores de la imagen.
Actor	Usuario
Descripción	El usuario cambia los colores de la imagen a blanco y negro.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen. PRE02. Haber completado el giro y recorte de la imagen.
Postcondiciones	POST01. Imagen binarizada.

Flujo Normal	FN01. El usuario hace clic izquierdo en los colores que quiere cambiar al negro. FN02. El usuario hace clic derecho en los colores que quiere cambiar al blanco. FN03. El usuario selecciona los umbrales para los colores. FN04. El usuario comienza el cambio de colores. FN05. El sistema realiza los cambios de colores según los seleccionados. FN06. El sistema muestra la imagen con los cambios de color en el visor.
Flujo Alternativo 1	FA101. El usuario usa la binarización automática transformando los colores a blanco y negro según parámetros predeterminados.
Flujo Alternativo 2	FA201. El usuario puede eliminar colores seleccionados.
Flujo Alternativo 3	FA301. El usuario hace clic en un color seleccionado para cambiar el color.
Excepciones	E01. El usuario no realiza ningún cambio de color.
Prioridad	Alta

Cuadro 4.5: CU05. Arreglar colores de la imagen.

Nombre e ID del CU	CU06. Arreglar colores de la imagen.
Actor	Usuario
Descripción	El usuario elimina errores visuales de la imagen.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen. PRE02. Haber completado el giro y recorte de la imagen. PRE03. Haber Completado el cambio de colores de la imagen.
Postcondiciones	POST01. Imagen sin errores visuales.
Flujo Normal	FN01. El usuario elige los tamaños de errores (ya sea para borrar o rellenar huecos). FN02. El sistema realiza la corrección en la imagen realizando erosiones para eliminar errores y aperturas para rellenar huecos. FN03. El sistema muestra la imagen arreglada en el visor. FN04. El usuario elige una dilatación para la esqueletización. FN05. El sistema realiza una esqueletización de la imagen y realiza la dilatación especificada. FN06. El sistema muestra la imagen dilatada tras la esqueletización en el visor.
Flujo Alternativo 1	FA101. El usuario no realiza ninguna corrección.
Flujo Alternativo 2	FA201. El usuario no realiza la esqueletización tras la corrección.
Excepciones	E01. Python no se encuentra en el directorio asignado. E02. Python no encuentra el módulo de OpenCV.

Prioridad	Alta

Cuadro 4.6: CU06. Arreglar colores de la imagen.

Nombre e ID del CU	CU07. Vectorizar imagen.
Actor	Sistema
Descripción	El sistema realiza la vectorización de la imagen.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen. PRE02. Haber completado el giro y recorte de la imagen. PRE03. Haber completada el cambio de colores de la imagen. PRE04. Haber completado la corrección de la imagen.
Postcondiciones	POST01. Imagen vectorizada guardada en el sistema.
Flujo Normal	FN01. El usuario activa la vectorización. FN02. El usuario selecciona dónde quiere guardar la imagen. FN03. El sistema realiza la vectorización de la imagen. FN04. El sistema guarda la imagen vectorizada. FN05. El sistema muestra la imagen vectorizada.
Excepciones	E01. Carpeta seleccionada por el usuario no válida. E02. Python no se encuentra en el directorio asignado. E03. Python no encuentra el módulo VTracer en el directorio asignado.
Prioridad	Alta

Cuadro 4.7: CU07. Vectorizar imagen.

Nombre e ID del CU	CU08. Personalizar imagen vectorizada.
Actor	Usuario
Descripción	El usuario personaliza la imagen eliminando los trazos que quiera.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen. PRE02. Haber completado el giro y recorte de la imagen. PRE03. Haber completada el cambio de colores de la imagen. PRE04. Haber completado la corrección de la imagen. PRE05. Haber realizado la vectorización.
Postcondiciones	POST01. Imagen personalizada exportable.
Flujo Normal	FN01. El usuario elimina los trazos de la imagen no deseados.
Flujo Alternativo 1	FA101. El usuario devuelve el último trazo eliminado a la imagen.

Prioridad	Baja

Cuadro 4.8: CU08. Personalizar imagen vectorizada.

Nombre e ID del CU	CU09. Exportar imagen final.
Actor	Usuario
Descripción	El usuario finalizará el proyecto exportando la imagen.
Precondiciones	PRE01. Haber empezado el proyecto y tener seleccionada una imagen. PRE02. Haber completado el giro y recorte de la imagen. PRE03. Haber completada el cambio de colores de la imagen. PRE04. Haber completado la corrección de la imagen. PRE05. Haber realizado la vectorización. PRE06. Haber completado la personalización de la imagen.
Postcondiciones	POST01. El programa se cerrará.
Flujo Normal	FN01. El usuario finaliza el proyecto. FN02. El usuario selecciona la carpeta de destino de la imagen personalizada. FN03. El sistema prepara la nueva imagen vectorizada. FN04. El sistema exporta la imagen al lugar indicado. FN05. El sistema cierra el programa.
Excepciones	E01. Carpeta seleccionada no válida para la exportación.
Prioridad	Alta

Cuadro 4.9: CU09. Exportar imagen final.

Nombre e ID del CU	CU10. Personalizar experiencia.
Actor	Usuario
Descripción	El usuario seleccionará la personalización que tendrá el programa en tema y cursor.
Postcondiciones	POST01. Tema elegido y cursor elegido visibles.
Flujo Normal	FN01. El usuario hace clic sobre la configuración específica de tema y cursor. FN02. El usuario guarda la configuración. FN03. El sistema cambia el tema y cursor al elegido por el usuario.
Flujo Alternativo 1	FA101. El usuario no guarda y no se cambia nada.
Prioridad	Baja

Cuadro 4.10: CU10. Personalizar experiencia.

Nombre e ID del CU	CU11. Seguridad de historial.
Actor	Sistema
Descripción	El sistema mantendrá un historial de acciones importantes realizadas.
Precondiciones	PRE01. El usuario tiene que querer tener este historial activo.
Postcondiciones	POST01. El historial se guarda en el directorio de historial.
Flujo Normal	FN01. El sistema genera un nuevo fichero de historial. FN02. El sistema va guardando nuevas líneas según van sucediendo en el programa. FN03. El sistema cierra el historial.
Excepciones	E01. El usuario cancela la generación de historiales.
Prioridad	Baja

Cuadro 4.11: CU11. Seguridad de historial.

Nombre e ID del CU	CU12. Configuración exportable.
Actor	Sistema
Descripción	El sistema exporta e importa las configuraciones.
Precondiciones	PRE01. Tener una configuración exportada para la importación.
Postcondiciones	POST01. Configuración exportada o importada.
Flujo Normal	FN01. El usuario selecciona la exportación. FN02. El usuario selecciona la carpeta donde exportar el archivo. FN03. El sistema genera el JSON que exportará con la información de la configuración. FN04. El sistema exporta el fichero.
Flujo Alternativo 1	FA101. El usuario selecciona la importación. FA102. El usuario selecciona el archivo que importar. FA103. El sistema lee el archivo. FA104. El sistema importa los datos a la configuración.
Excepciones	E01. El archivo no es válido.
Prioridad	Baja.

Cuadro 4.12: CU12. Configuración exportable.

4.2. Atributos de calidad

Traducidos a partir de los **requisitos no funcionales**, en el desarrollo de la aplicación se siguieron los siguientes atributos de calidad:

Compatibilidad. La aplicación está diseñada específicamente para **sistemas Windows**, asegurando su **correcto funcionamiento** en este entorno gracias a la facilidad de *testeo* en muchos dispositivos con diferentes configuraciones.

Seguridad. El sistema puede generar **historiales** que permiten detectar errores, trazabilidad de uso y posibles accesos inadecuados. Ayuda a **garantizar la integridad** y el seguimiento del comportamiento de la aplicación.

Robustez. La aplicación está diseñada para **manejar entradas inesperadas y errores sin fallos críticos**. Se busca un comportamiento estable incluso ante condiciones no ideales (archivos corruptos, errores de formateo, etc.).

Usabilidad. Interfaz clara, **comprensible** y fácil de usar tras varios usos. Con un idioma de mayor accesibilidad y estándarización como es el inglés y con mejoras de la **experiencia del usuario de interactividad** y personalización.

Mantenibilidad. El **acceso libre al código fuente** permite modificaciones, correcciones y mejoras a lo largo del tiempo, y facilita la colaboración y revisión por parte de la comunidad.

Portabilidad. Uso de **tecnologías ampliamente soportadas** como Qt, C++ y Python que permiten una adaptación rápida a otros entornos en el futuro.

Eficiencia y Rendimiento. Procesamiento rápido y eficaz en imágenes estándar con optimización de recursos computacionales.

Personalización. Permite adaptar ciertos elementos de la interfaz al gusto o necesidad del usuario.

Capítulo 5

Diseño

En este capítulo se detallarán los aspectos más importantes del diseño de la aplicación exponiendo la arquitectura conceptual de la solución desarrollada. Este diseño será la guía para comprender cómo se organiza la lógica de datos y su interacción.

5.1. Diseño de datos

La persistencia de datos en la aplicación no se realiza a través de una base de datos, sino apoyándose en el **sistema de archivos del usuario**; un proyecto se guardará siempre que el usuario lo desee y la configuración del programa se guarda como claves del sistema. De este modo, un modelo de entidad-relación pierde el sentido en nuestra aplicación, pero, sin embargo, un **diagrama de clases** realiza sus funciones a la perfección dando toda la **información necesaria** para mostrar las colecciones de datos que se usan en la aplicación.

Las dos **colecciones de datos persistentes** son el proyecto en uso y la configuración del usuario:

Configuración de usuario: Colección de datos utilizada a través del objeto Qt QSettings. Este objeto permite guardar información en las claves del sistema operativo a través de una especificación inicial de "Empresa" y "Programa" que determinan el espacio de uso y protegen la modificación del resto de claves. Su uso es muy parecido al de una **lista de vectores de dos cadenas de caracteres claves-valor**. Aunque su guardado se efectúa como QStrings (cadenas de caracteres en Qt) en el uso interno de la aplicación las claves corresponden a un **enumerado** con las diferentes posibilidades de clave. La siguiente lista enumera las posibilidades de clave y su valor asociado en QSettings.

- **CURSOR_PATH**. El valor será un **QString** que corresponde a la ruta de archivos donde está guardado el cursor a utilizar.
- **CURSOR_SIZE**. El valor será una posibilidad del **enumerado** de tamaño ("SMALL", "MEDIUM", "LARGE").

- **THEME**. Valor QString con el nombre del tema **QStyleFactory** correspondiente (QStyleFactory es una colección de temas).
- **THEME_DARK**. **Booleano** correspondiente a la elección de tener un tema oscuro o claro.
- **FILEPATH_LAST_USED**. Ruta de carpeta en **QString** que se visitó usando el programa por última vez.
- **FILEPATH_TEMP**. Ruta a la carpeta temporal utilizada por el programa, su valor se usa como **QString**.
- **LOG_FILEPATH**. Ruta a la carpeta donde se guardarán los historiales del programa. Valor en **QString**.
- **LOG_USE**. Uso del historial aceptado por el usuario o no, se usa como **booleano**.
- **LOG_EXPIRE_TIME**. **Número** de unidades de tiempo en las que expirarán los historiales creados.
- **LOG_EXPIRE_TIME_UNIT**. **Enumerado** de unidades de tiempo. Sus posibilidades son: "Seconds", "Minutes", "Hours", "Days", "Weeks", "Months", "Years".
- **LOG_MAX_NUMBER**. Máxima cantidad de historiales que pueden existir representados en **número**.
- **NONE**. Valor por defecto utilizado para comprobar errores, no se llega a crear como clave nunca.

Archivo de proyecto: Esta colección de datos está compuesta por dos QHash (clase Qt que implementa una tabla hash que almacena pares clave-valor). Ambos QHash se comportan como QStrings a la hora de ser exportados y guardados como archivo, pero en la aplicación se comportan de la siguiente manera:

Configuración de proyecto: En la aplicación, este QHash tiene pares clave-valor de Enumerado-QString. Las claves son:

- **PROJECT_NAME**. Nombre del proyecto.
- **PROJECT_FILEPATH**. Ruta del proyecto donde se localizarán todas las imágenes temporales guardadas.
- **IMAGE_FILEPATH**. Ruta de la imagen utilizada en el proyecto.
- **IMAGE_CROPPED_FILEPATH**. Imagen recortada y girada.
- **COLOURS_PICKED**. Número de colores seleccionados.
- **IMAGE_BINARIZE_FILEPATH**. Ruta de la imagen binarizada (los colores están en blanco y negro).
- **IMAGE_TO_CORRECT_FILEPATH**. Ruta de la imagen previa a la corrección de errores.

- **CLOSING_CORRECTION_VALUE**. Valor de la erosión que sufrirá la imagen.
- **OPENING_CORRECTION_VALUE**. Valor de la apertura que tendrá la imagen.
- **SKELET_CORRECTION_VALUE**. Valor de la dilatación que afectará a la imagen tras la dilatación.
- **IMAGE_CORRECTED_FILEPATH**. Ruta de la imagen corregida.
- **NONE**. Valor por defecto usado como reconocimiento de problemas.

Colores seleccionados: Este QHash funciona con claves-valor de dos QString. Generando por cada selección de color dos claves-valor:

- **COLOUR_PICKED_[X]_[Y]**. Siendo "X" el color al que debe cambiar (blanco o negro) e "Y" el número correspondiente a la cantidad de colores seleccionados. Su valor es el **nombre del color** seleccionado.
- **COLOUR_PICKED_[X]_[Y]_THRESHOLD**. Compartiendo clave y construcción del anterior, pero añadiendo "umbral" para designar el **valor utilizado** para los umbrales de ese color.

5.2. Diagrama de clases

El diagrama de clases final de la aplicación resulta imposible de mostrar en su totalidad, pero es posible mostrarlo enseñando las relaciones que suceden alrededor de la ventana principal individualmente. Además, para mantener las cosas más simples, no se mostrarán las relaciones con elementos bases de Qt (como son las herencias de subclases o usos dentro de las clases).

Comencemos con el diagrama de clases correspondiente a la **ventana de configuración** del programa 5.1. En él se pueden ver:

- La clase y ventana principal **"MainWindow"** que mantiene la aplicación viva. Es donde se realizan todos los pasos y modificaciones de la imagen.
- La clase **"ConfigDialog"** que corresponde a la ventana de configuración. Todos los cambios de QSettings se realizarán en esta ventana.
- La subclase de QSpinBox, **"CustomQSpinBox"**, para bloquear visualmente los QSpinBox.
- La clase **"SettingsHandler"** se ocupa de manejar QSettings (teniendo atributos enumerados).
- La clase **"FileManager"** gestiona las aperturas y escrituras de ficheros.

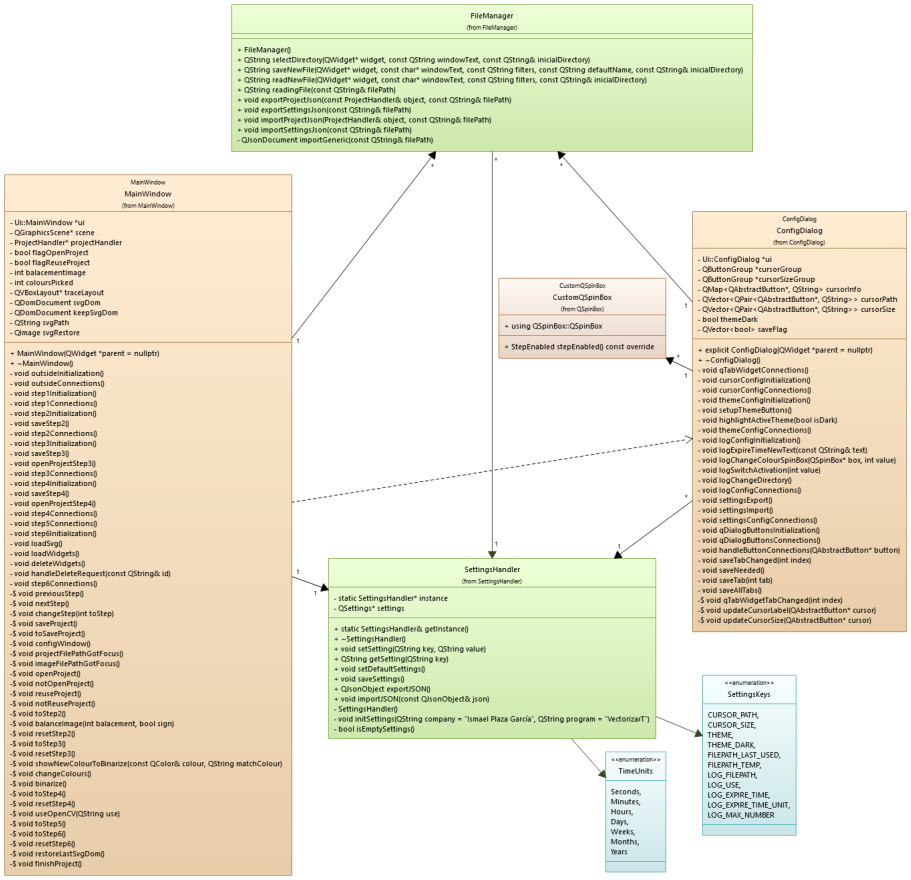

Figura 5.1: Diagrama de clases correspondiente a la ventana principal con ConfigDialog.

El siguiente diagrama de clases corresponde a la **generación del historial** 5.2. En este diagrama obviaremos todas las relaciones que llaman a la generación (dejando la relación del MainWindow como referencia) de una nueva línea del historial pues esas relaciones nos llevarían al diagrama global de la aplicación (este diagrama corresponde a las relaciones del historial cuando ya se ha solicitado una nueva línea).

- **"MainWindow"** como generalización del resto de llamadas.
- **"SettingsHandler"**.
- **"FileManager"**.
- Clase **"SecurityManager"** que maneja todas las llamadas recibidas.
- Clase **"LogManager"**, gestor de los historiales a crear y creados.

Como principal y más importante diagrama de clases, está el diagrama de gestión de los propios projectos 5.3.

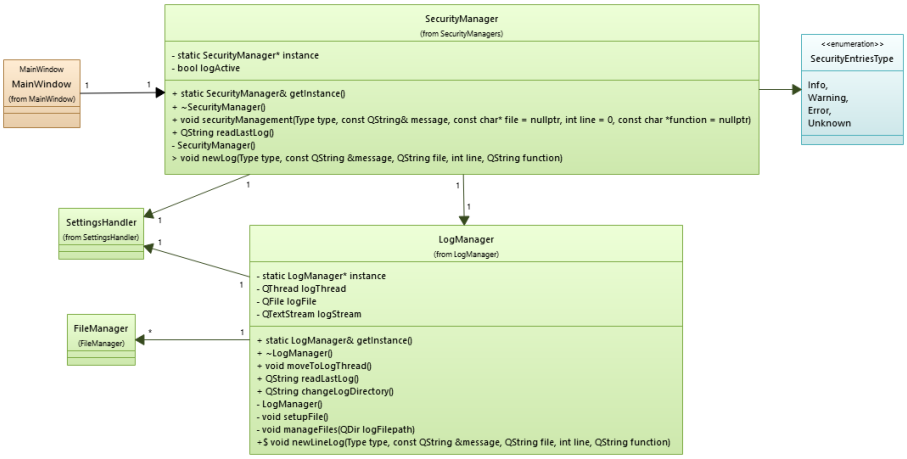

Figura 5.2: Diagrama de clases correspondiente a la generación del historial.

- **"MainWindow"** como punto central de todas las gestiones.
- Clase **"ProjectHandler"** que maneja todos los datos de los que se compone un proyecto de la aplicación.
- Con las clases **"PythonManager"** y **"VTracer"** que, respectivamente, generan el puente necesario para poder hacer un uso completo de Python y usan ese puente para realizar la vectorización de la imagen.

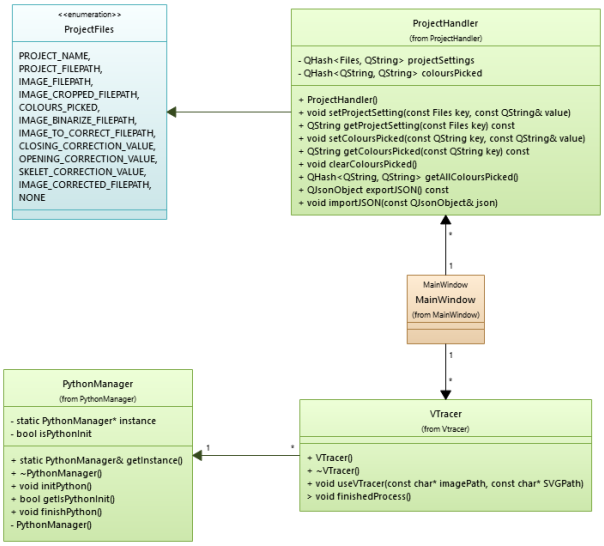

Figura 5.3: Diagrama de clases con la gestión de los projectos.

Por último, el diagrama de clases compuesto por todas las subclases de objetos de UI utilizados en la ventana principal que extienden las posibilidades de los objetos de Qt base para un acabado más interactivo para el usuario 5.4.

- **"MainWindow"** donde se usaron estas subclases.
- Subclase de colección de objetos **"CustomQWidget_ColoursToBinarize"** que contiene los objetos necesarios para cambiar el color seleccionado, proponer un umbral o eliminar el objeto en el paso de binarización de imagen.
- Subclase de colección de objetos **"CustomQWidget_SvgPaths"**. Contiene los objetos necesarios para personalizar la imagen vectorizada y poder eliminar los distintos vectores extra.
- **"CustomQToolBox_StepByStep"**. Subclase que permite la personalización de la herramienta contenedora de los pasos de la aplicación para una mejor experiencia de usuario.
- Subclase **"CustomQGraphicsView_ClickablePic"**, subclase de los visores de imágenes para poder manipular la imagen pudiendo tener distintas funciones dependiendo del "tipo"de imagen.
- La subclase de un visor de imagen, **"CustomQLabel_Images"**, sin interacción pero manteniendo de manera consistente la relación de aspecto de la imagen.
- **"CustomQLineEdit_newFocus"** como subclase de una línea de texto editable que avisa al programa cuando tiene el focus (por ejemplo, cuando se hace clic en ella).

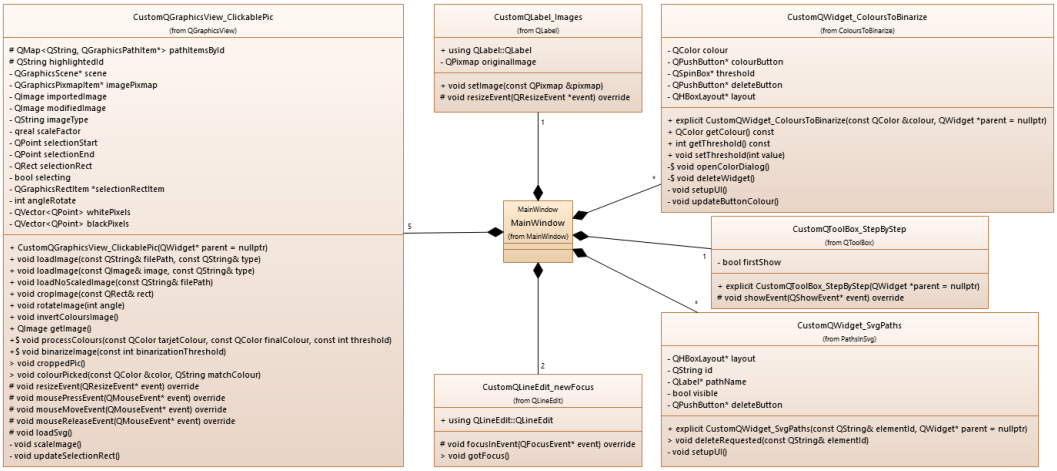

Figura 5.4: Diagrama de clases con los objetos subclaseados de la UI utilizados en la ventana principal.

5.3. Diagramas de secuencia

A la hora de realizar los diagramas de secuencia se han considerado como importantes las actividades de corrección de imagen y de vectorización al ser los procesos donde la trazabilidad se vuelve interesante.

Comenzamos con el diagrama de secuencia de corrección de imagen donde tenemos la interacción entre el usuario, la UI de la ventana principal, el controlador de las acciones y el modelo de datos 5.5.

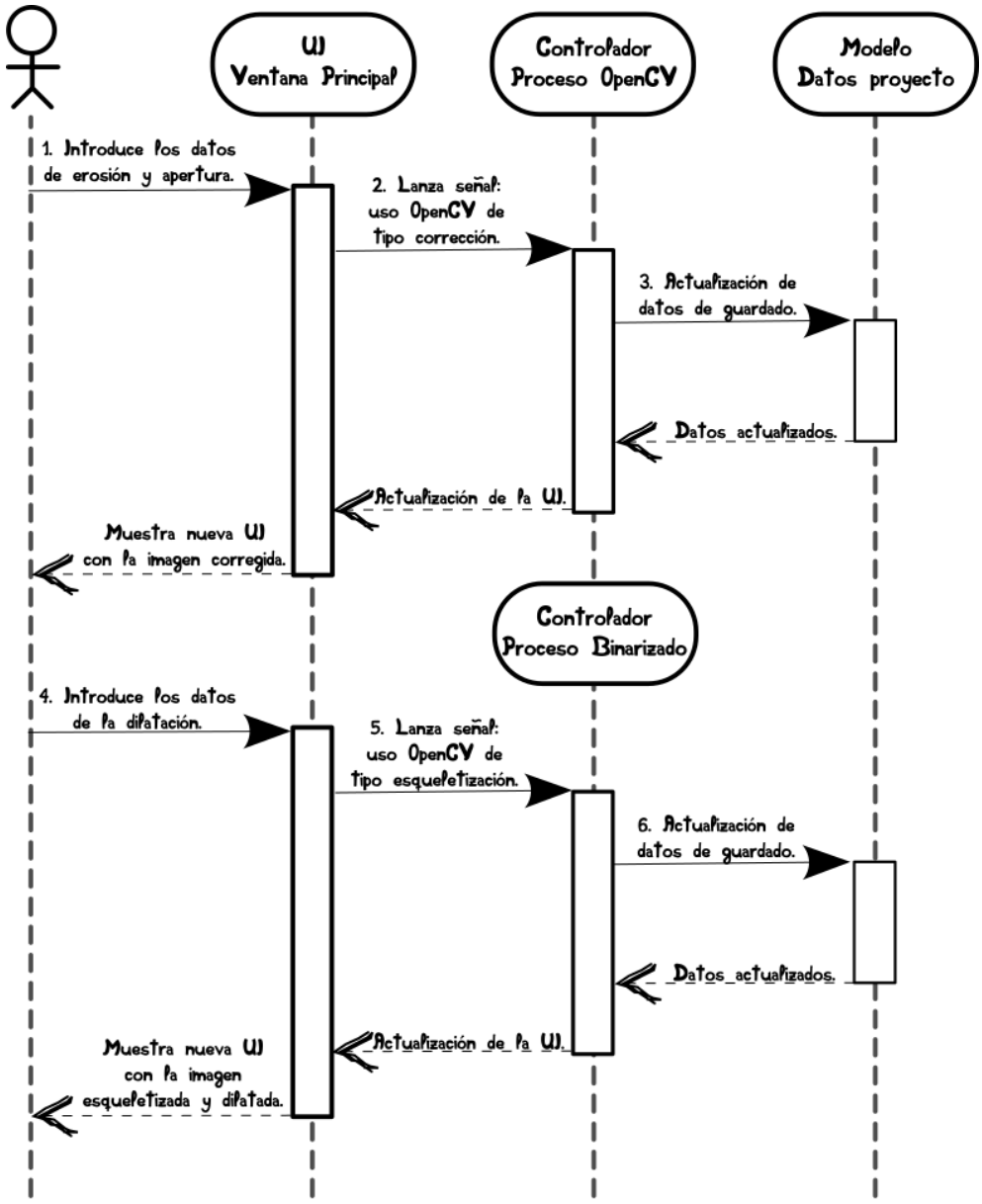

Figura 5.5: Diagrama de secuencia de la corrección de imagen.

El segundo y último diagrama de secuencia es el del proceso de vectorización.

Después de vectorizar la imagen, la aplicación cambia de paso a la personalización de la imagen vectorial y esto genera un segundo proceso que actualizará la UI con nuevos objetos 5.6.

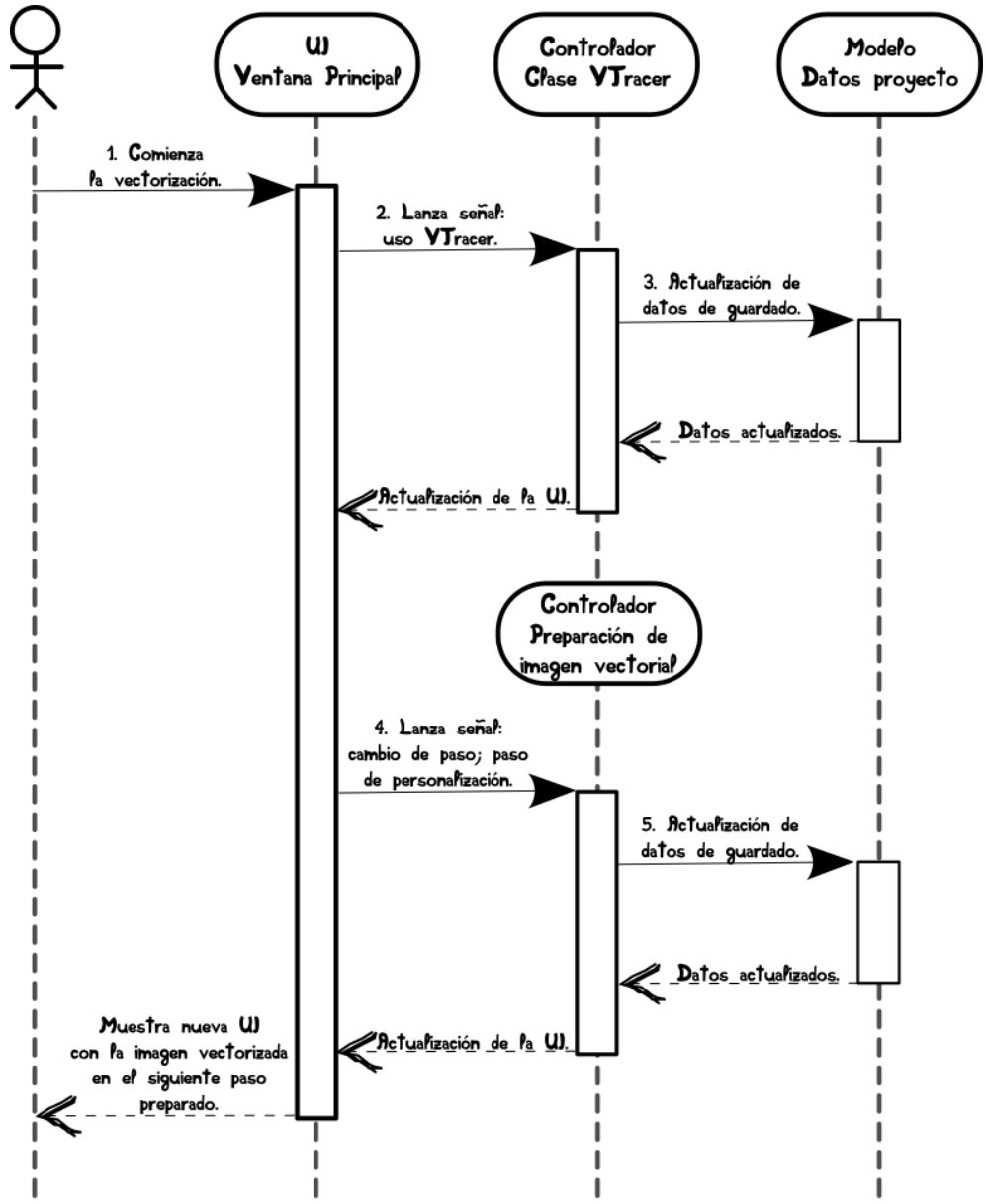

Figura 5.6: Diagrama de secuencia de la vectorización de imagen.

5.4. Diagrama de estado

Con la diferencia entre los pasos y los procesos de los mismos, podemos seguir todo el progreso que tiene la imagen con el siguiente diagrama de estados 5.7:

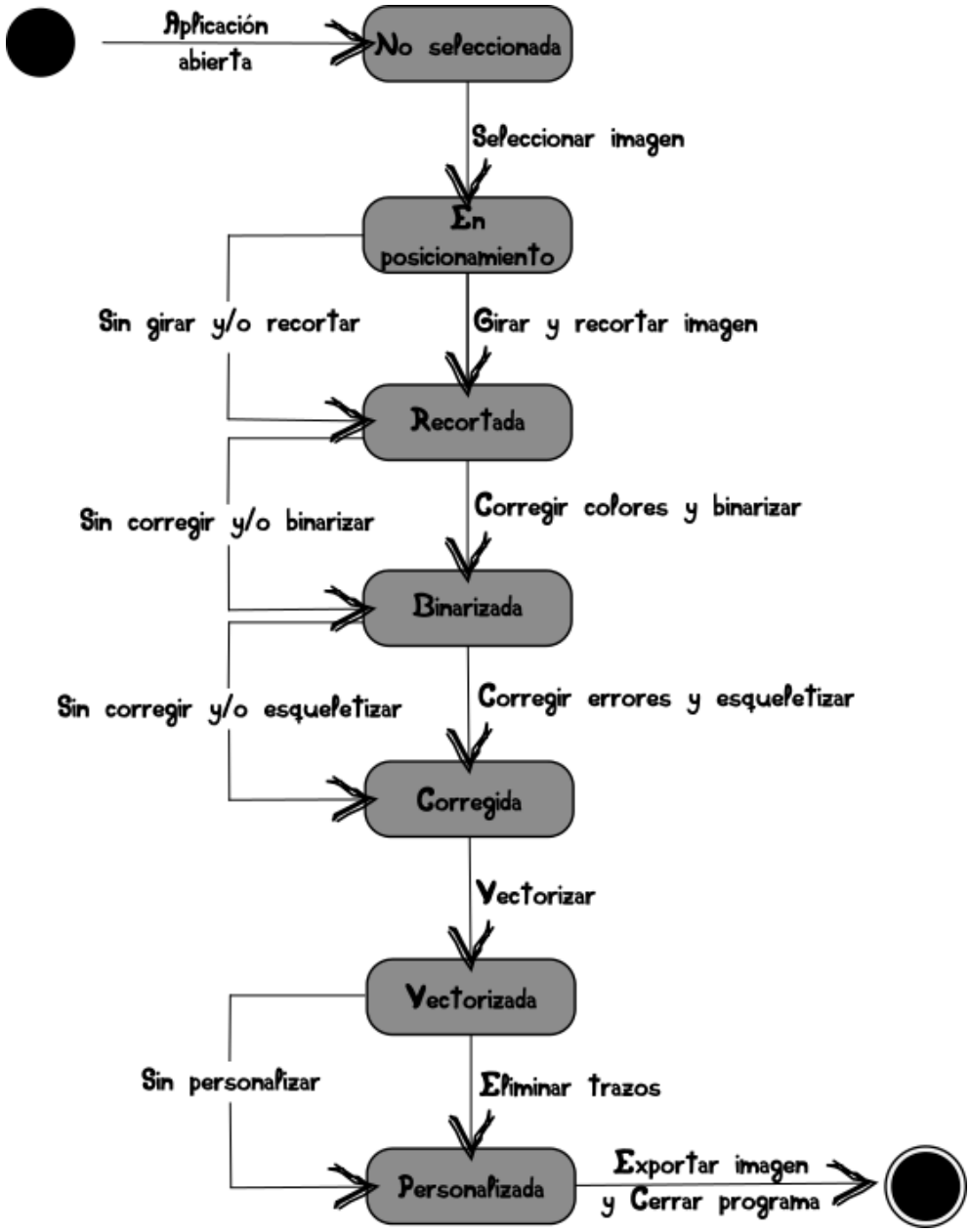

Figura 5.7: Diagrama de estados de una imagen en la aplicación.

Capítulo 6

Implementación

Este capítulo tiene como objetivo detallar cómo se ha llevado a cabo la implementación de la aplicación descrita en los capítulos anteriores, transformando el diseño lógico en una solución funcional.

La implementación se ha realizado siguiendo los principios del paradigma de **Programación Orientada a Objetos** y del **Desarrollo Basado en Componentes** (CBD), lo que ha permitido una clara separación de responsabilidades, una mejor escalabilidad del proyecto y un profundo uso de objetos y clases de Qt (incluso pudiendo subclasear para mayor extensión de posibilidades). A su vez, se ha respetado la estructura propuesta por el **patrón Modelo-Vista-Controlador**.

Esta fusión de paradigma y patrón ha llevado a producir un código **dividido en distintas clases** (haciendo objetos de ellas cuando se usan) con las que el usuario se comunica a través de la interfaz; **cada clase está separada en distintos archivos** cabecera y fuente creando el puente entre modelo y controlador con **inclusiones** (en el archivo del controlador se incluye el archivo del modelo para hacer las modificiones desde el controlador como si formasen parte del mismo archivo).

La comunicación entre vista y controlador se realiza en Qt con el sistema **signal/slot** donde las **señales** lanzadas por los objetos de interfaz están conectadas a **receptores** que nos permiten desarrollar componentes individuales siguiendo fácilmente la organización de código del CBD.

A continuación, se explicarán más **detalladamente** algunos procesos de **implementación** seguidos para partes notables de la aplicación detallando su funcionalidad, su integración con el resto del sistema y las decisiones técnicas tomadas durante el desarrollo.

6.1. Implementación de la vectorización. VTracer por Vision Cortex

A la hora de la implementación de la vectorización, se presentaron varias posibilidades: realizar un método de vectorización propio o utilizar otras aplicaciones existentes para conseguir un resultado más optimizado.

Al principio se optó por realizar una vectorización propia a través de Octave, pero toda la casuística derivada crearía grandes problemas de optimización para llegar a un resultado mediocre al no tener los conocimientos matemáticos necesarios[1].

Como opción ganadora se optó por **utilizar una aplicación externa** para conseguir unos resultados aceptables. De las aplicaciones que se barajaron, la lista se redujo a dos candidatos:

1. **Potrace**. Aplicación y algoritmo de vectorización usado por el programa Inkscape.

2. **VTracer**. [13]Aplicación y algoritmo de vectorización usado por aplicaciones propietarias de los mismos desarrolladores, Vision Cortex, que utilizan **valores de simplificación** para realizar la vectorización.

Después de hacer comparativas entre estas dos opciones Open Source, se concluyó tanto por calidad, posibilidades y optimización que se utilizaría VTracer. Su **calidad y posibilidades** gracias a la simplificación ayudan a que imágenes (incluso a color) puedan ser vectorizadas con diferentes características, rango de colores y estilos, y su **optimización** con un algoritmo de ajuste 0(n) (frente al $0(n^2)$ de Potrace) ayuda a tiempos de renderización de imágenes más bajos.

Su implementación supuso la necesidad de realizar una **instalación de Rust y Cargo** (su herramienta de gestión de proyectos), junto con la descarga del Github del proyecto de VTracer. Estas instalaciones y descargas se hicieron para **entender de una manera más profunda las llamadas que se necesitan** desde C++. Con el código al descubierto se intentó realizar una librería que permitiera el uso de la aplicación de manera directa con C++, pero al no tener experiencia con Rust se dejó esta idea para una futura posible actualización de mejora en rendimiento y reducción del peso del programa.

Añadida a estas instalaciones transitorias de investigación y aprendizaje se instaló **Python** para, haciendo uso de su librería de conexión con C++, realizar un **uso completo y hermético** en código de la aplicación VTracer. De manera paralela, toda la implementación de código se hizo con la idea principal de **embeber Python junto con la aplicación** para evitar al usuario descargas y aplicaciones extra; un código o programa embebido es aquel que es integrado dentro de otro sistema para que funcione como parte de él sin que el usuario lo note.

Para que pueda ser utilizado embebido con VTracer en esta aplicación se hizo lo siguiente:

[1]En el apartado de "Trabajo a futuro" hay más información de este opción descartada.

1. **Python embebido con la mejor versión para VTracer**. Se copiaron todos los archivos del Python instalado a una carpeta dentro del proyecto durante la programación y pruebas. A la hora de la compilación y creación del ejecutable distribuido se copió esta misma carpeta para hacer uso de rutas relativas a la carpeta del ejecutable. [2]La versión elegida de Python es la 3.12 por compatibilidad completa con VTracer.

   [3]Para tener VTracer en Python se descargó del repositorio los archivos compilados y se pegaron en su carpeta de librerias: Python/Lib/site-packages. (Se intentó realizar su instalación a través de "pip" en Python, pero daban errores).

2. **Copia del archivo .dll de Python**. Junto con el ejecutable de la aplicación y la carpeta de Python se copiará el archivo de librería de Python para Windows (archivo .dll), la ruta es necesaria que sea esta pues genera problemas si no la encuentra directamente.

3. **Búsqueda de los documentos de compilación de Python en tiempo de ejecución**. Se usa para que la ejecución de Python sea correcta pues necesita que todas las librerias y los import de Python (archivos cabecera para usar Python en C y C++) se encuentren con las rutas relativas. Esto se consigue en el constructor del ejecutable (archivo .pro) indicando estas dos rutas (también relativas) que llevan a sus carpetas del Python embebido.

Listing 6.1: VectorizarT.pro - Búsqueda de Python Embebido.

```
// VectorizarT.pro
INCLUDEPATH += $$PWD/Python/include
LIBS += -L$$PWD/Python/libs -lpython312
```

4. **Programación del archivo para usar Python**. Para conectar el código C++ con la librería de Python se programó un controlador singleton (clase que sólo permite la instancia de un objeto). En C++ este tipo de clase no existe de manera explícita por lo que se simula haciendo uso de una función que devuelve su instancia y privatiza la creación de nuevas instancias.

Listing 6.2: PythonManager.h y PythonManager.cpp - Funciones del Singleton.

```
// PythonManager.h
#include <Python.h>

#ifndef PYTHONMANAGER_H
#define PYTHONMANAGER_H

class PythonManager {
```

[2]https://www.python.org/downloads/release/python-3120/
[3]https://pypi.org/project/vtracer/#vtracer-0.6.11-cp312-none-win_amd64.whl

```
8       public:
9           static PythonManager& getInstance();
10
11          ~PythonManager();
12      private:
13          PythonManager();
14
15          static PythonManager* instance;
16  // ...
17  }
18
19  // PythonManager.cpp
20  PythonManager* PythonManager::instance = nullptr;
21
22  PythonManager::PythonManager():isPythonInit(false){}
23
24  PythonManager& PythonManager::getInstance() {
25      if (!instance)
26          instance = new PythonManager();
27
28      return *instance;
29  }
30
31  PythonManager::~PythonManager() {
32      finishPython();
33
34      delete instance;
35  }
36  // ...
```

Python embebido no debe ser inicializado más de una vez[4] durante toda su vida en la aplicación, por lo que tener su clase instanciada una sola vez asegurará a que no se produzcan errores de reinicio del módulo.

El resto de la clase creará la conexión con Python:

Listing 6.3: PythonManager.h y PythonManager.cpp - Conexión con Python.

```
1   // PythonManager.h
2   // ...
3   public:
4       void initPython();
5
```

[4][14]Reiniciar Python embebido no es seguro ni recomendable porque algunos módulos pueden no limpiarse correctamente.

```
6       bool getIsPythonInit() {
7           return isPythonInit;
8       }
9
10      void finishPython();
11
12  private:
13      bool isPythonInit;
14
15  // PythonManager.cpp
16  // ...
17  void PythonManager::initPython() {
18      if (!Py_IsInitialized()) {
19          const char* pythonHome = "/Python";
20          _putenv_s("PYTHONHOME", pythonHome);
21          _putenv_s("PYTHONPATH",
22              std::string(pythonHome)
23                  .append("/Lib").c_str());
24
25          Py_Initialize();
26      }
27
28      if (Py_IsInitialized()) {
29          isPythonInit = true;
30      }
31  }
32
33  void PythonManager::finishPython() {
34      if (isPythonInit) {
35          Py_Finalize();
36
37          isPythonInit = false;
38      }
39  }
```

5. **Programación del archivo para usar VTracer**. Para hacer uso de VTracer se utiliza una clase específica donde se utiliza la conexión de la clase PythonManager. Para un uso responsable de la memoria, todos los elementos PyObject utilizados son eliminados en cuanto cumplen su función.

Listing 6.4: VTracer.h y VTracer.cpp - Uso de Python para acceder y usar VTracer.

```
1  // VTracer.h
2  class VTracer {
```

```
3  public:
4      VTracer();
5
6      ~VTracer();
7
8      void useVTracer();
9  };
10
11 // VTracer.cpp
12 #include "vtracer.h"
13
14 VTracer::VTracer() {}
15
16 VTracer::~VTracer() {}
17
18 void VTracer::useVTracer() {
19     if(!PythonManager::getInstance().getIsPythonInit()){
20         PythonManager::getInstance().initPython();
21     }
22
23     PyObject* moduleName =
24         PyUnicode_FromString("vtracer");
25     PyObject* module = PyImport_Import(moduleName);
26     Py_DECREF(moduleName);
27
28     if (module) {
29         PyObject* func =
30             PyObject_GetAttrString(module,
31                 "convert_image_to_svg_py");
32         if (func && PyCallable_Check(func)) {
33             PyObject* inPath =
34                 PyUnicode_FromString("IMAGEN EDITADA");
35             PyObject* outPath =
36                 PyUnicode_FromString("RUTA ELEGIDA");
37
38             PyObject* args =
39                 PyTuple_Pack(2, inPath, outPath);
40             PyObject* result =
41                 PyObject_CallObject(func, args);
42
43             Py_XDECREF(args);
44             Py_XDECREF(inPath);
45             Py_XDECREF(outPath);
```

```
46
47              if (result) {
48                  Py_DECREF(result);
49              } else {
50                  PyErr_Print();
51              }
52
53              Py_DECREF(func);
54          } else {
55              PyErr_Print();
56          }
57
58          Py_DECREF(module);
59      } else {
60      PyErr_Print();
61      }
62  }
```

Después de la preparación completa de las clases que hacen el proceso, el uso de la clase como objeto y la conexión con la UI está en el controlador de la ventana principal:

Listing 6.5: MainWindow.cpp - Uso de VTracer para realizar la vectorización.

```
1  //MainWindow.cpp
2  //...
3  void MainWindow::step5Connections() {
4      //Conexiones con la UI.
5      connect(ui->s5_previousStepButton, &QPushButton::clicked,
6          this, &MainWindow::previousStep);
7      connect(ui->s5_nextStepButton, &QPushButton::clicked,
8          this, &MainWindow::toStep6);
9  }
10 void MainWindow::toStep6() {
11     //Comprobaciones previas.
12     QString SVGPath;
13     if (projectHandler->getProjectSetting(
14             Files::PROJECT_FILEPATH).isEmpty())
15         SVGPath = FileManager().saveNewFile(this,
16             "Save vectorized image...",
17             "Vectorize images (*.svg);;All files (*)",
18             projectHandler->
19                 getProjectSetting(Files::PROJECT_NAME)
20                 .isEmpty() ? "newProject.svg" :
21                 projectHandler->getProjectSetting(
```

```
                    Files::PROJECT_NAME) + ".svg");
    else
        SVGPath = FileManager().saveNewFile(this,
            "Save vectorized image...",
            "Vectorize images (*.svg);;All files (*)",
            projectHandler->
                getProjectSetting(Files::PROJECT_NAME)
                .isEmpty() ? "newProject.svg"
                : projectHandler->
                    getProjectSetting(Files::PROJECT_NAME)
                    + ".svg",
            projectHandler->
                getProjectSetting(Files::PROJECT_FILEPATH));
    if (SVGPath.isEmpty())
        return;

    QString imageFilePath;
    if (projectHandler->
        getProjectSetting(Files::PROJECT_FILEPATH).isEmpty())
    {
        imageFilePath =
            SettingsHandler::getInstance().
                getSetting(SettingsKeys::
                    keyToString(Key::FILEPATH_TEMP)) +
                    "/s5_toVectorize.png";
    } else {
        imageFilePath = projectHandler->
            getProjectSetting(Files::PROJECT_FILEPATH) +
            "/edited_step_images/s5_toVectorize.png";
    }
    QImage correctedImage = ui->s5_graphicsView->getImage();
    correctedImage.save(imageFilePath);

    //Comienzo del proceso y creacion del objeto.
    SecurityManager::sc_INFO("Starting vectorizing process.");
    VTracer* vtracer = new VTracer();

    vtracer->useVTracer(imageFilePath.toUtf8(),
        SVGPath.toUtf8());

    delete vtracer;

    //...
```

```
}
//...
```

6.2. Implementación de la corrección de errores.

Dada la naturaleza y el objetivo principal del TFG de aprender mucho más de Qt y tener una visión más profunda de su uso, la implementación de la corrección de errores se realizó de una manera diferente a la vectorización.

Aún tener una base parecida a la vectorización por usar Python, para la corrección se utilizó, a modo de investigación del medio, un objeto de Qt que permite realizar procesos de otros programas y lenguajes. El objeto provisto por Qt es **QProcess** con el que realizar procesos externos de la aplicación es posible de forma casi nativa y sin diferencias en tiempo de respuesta frente a usar una librería (al menos para el uso realizado en la aplicación).

Como desventaja en el uso de este objeto se encontró la poca división del código que se puso hacer en este caso, pero resulta una herramienta muy interesante y completa.

Para el uso correcto de este objeto Qt se separaron los scripts Python que acompañan al programa junto al .dll. Esto sería un problema en otras filosofías, pero al realizar un programa Open Source que el usuario pueda ver (y modificar con las herramientas adecuadas) los scripts no resulta en ningún problema añadido.

Script de corrección con cierre/erosión y apertura de los trazos de una imagen.

A partir de los datos para el tamaño de los núcleos que tendrán el cierre y la apertura se usará una imagen (invertida en colores previamente) para aplicarse la corrección.

Listing 6.6: Python Script - Uso de OpenCV para realizar cierres/erosiones y aperturas.

```
import cv2
import sys
import os
import numpy as np

#Paso de argumentos
originalPath = sys.argv[1]
originalFolder = os.path.dirname(originalPath)
newImagePath = os.path.join(originalFolder,
    f"s4_afterCorrection.png")

image = cv2.imread(originalPath, cv2.IMREAD_GRAYSCALE)

#Inversion de colores
imageInverted = cv2.bitwise_not(image)
```

```
16
17 #Cierre
18 closeKernelSize = int(sys.argv[2])
19 if closeKernelSize % 2 == 0:
20     closeKernelSize += 1
21 kernelClose = cv2.getStructuringElement(cv2.MORPH_RECT,
22     (closeKernelSize, closeKernelSize))
23 morphClose = cv2.morphologyEx(imageInverted,
24     cv2.MORPH_CLOSE, kernelClose)
25
26 #Apertura
27 openKernelSize = int(sys.argv[3])
28 if openKernelSize % 2 == 0:
29     openKernelSize += 1
30 kernelOpen = cv2.getStructuringElement(cv2.MORPH_RECT,
31     (openKernelSize, openKernelSize))
32 morphOpen = cv2.morphologyEx(morphClose,
33     cv2.MORPH_OPEN, kernelOpen)
34
35 finalImage = cv2.bitwise_not(morphOpen)
36
37 #Guardado de imagen
38 cv2.imwrite(newImagePath, finalImage)
```

Script de esqueletización y dilatación de los trazos de la imagen.

Con el tamaño para el núcleo que tendrá la dilatación se aplica a una imagen corregida previamente la esqueletización y después la dilatación.

Para la esqueletización se realiza una función recursiva que va erosionando poco a poco el trazo hasta que solo queda el "esqueleto" para encontrar el trazo base sin grosor.

Listing 6.7: Python Script - Uso de OpenCV para realizar esqueletización y dilatación.

```
1 import cv2
2 import sys
3 import os
4 import numpy as np
5
6 #Paso de argumentos
7 originalPath = sys.argv[1]
8 originalFolder = os.path.dirname(originalPath)
9 newImagePath = os.path.join(originalFolder,
10     f"s4_afterCorrection.png")
11
```

```
image = cv2.imread(originalPath, cv2.IMREAD_GRAYSCALE)

imageInverted = cv2.bitwise_not(image)

#Esqueletizacion (que usa recursividad)
def skeletotization(binaryImage):
    img = binaryImage.copy()
    img = img // 255
    img = img.astype(np.uint8)

    kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
        (3, 3))
    skeleton = np.zeros(img.shape, np.uint8)

    while True:
        erosion = cv2.erode(img, kernel)
        open = cv2.morphologyEx(erosion,
            cv2.MORPH_OPEN, kernel)
        temp = cv2.subtract(erosion, open)
        skeleton = cv2.bitwise_or(skeleton, temp)
        img = erosion.copy()

        if cv2.countNonZero(img) == 0:
            break

    return skeleton * 255

#Uso de la funcion recursiva
skeleton = skeletotization(imageInverted)

#Dilatacion
kernelSize = int(sys.argv[2])
if kernelSize % 2 == 0:
    kernelSize += 1
kernel_dil = cv2.getStructuringElement(cv2.MORPH_RECT,
    (kernelSize, kernelSize))
dilated = cv2.dilate(skeleton, kernel_dil)

finalImage = cv2.bitwise_not(dilated)

cv2.imwrite(newImagePath, finalImage)
```

Por último, el **código del controlador del proceso**.

Código donde el QProcess se crea, se configura (tanto el proceso como su entorno) y se utiliza. En este proceso, además, se pudo hacer uso de un objeto extra como es el QProgressDialog donde se muestra una pequeña ventana del progreso del proceso. Con su propia señal y sensor se puede detectar el inicio y el final del proceso para mantener al usuario informado.

Listing 6.8: MainWindow.cpp - Uso de OpenCV a través de QProcess.

```
//Comprobaciones previas
SecurityManager::sc_INFO("Opening OpenCV with the "
    + use + " correction.");
QString imageFilePath;
if (projectHandler->
    getProjectSetting(Files::PROJECT_FILEPATH).isEmpty())
{
    imageFilePath = SettingsHandler::getInstance()
        .getSetting(SettingsKeys::
            keyToString(Key::FILEPATH_TEMP)) +
        "/s4_beforeCorrection.png";
} else {
    imageFilePath = projectHandler
        ->getProjectSetting(Files::
            IMAGE_TO_CORRECT_FILEPATH);
    if (imageFilePath.isEmpty()) {
        imageFilePath = projectHandler
            ->getProjectSetting(Files::PROJECT_FILEPATH) +
                "/edited_step_images
                /s4_beforeCorrection.png";
        projectHandler
            ->setProjectSetting(Files::
                IMAGE_TO_CORRECT_FILEPATH,
                imageFilePath);
    }
}
QFileInfo info(imageFilePath);
QDir dir(info.absolutePath());
if (!dir.exists()) {
    QDir().mkpath(info.absolutePath());
}
QImage correctedImage = ui->s4_graphicsView->getImage();
correctedImage.save(imageFilePath);

//Preparacion de QProgressDialog
```

```
QProgressDialog* progress = new QProgressDialog(
        "Executing VectorizarT " + use +
        " script...", "Cancel", 0, 0, this);
progress->setWindowModality(Qt::ApplicationModal);
progress->setCancelButton(nullptr);
progress->setMinimumDuration(0);
progress->show();

QString appDir = QCoreApplication::applicationDirPath();
QString pythonExe = QDir(appDir)
    .absoluteFilePath("Python/python.exe");
QProcess* process = new QProcess(this);

//Configuracion de entorno.
QProcessEnvironment env = QProcessEnvironment::
    systemEnvironment();
env.insert("PYTHONHOME", appDir + "/Python");
env.insert("PYTHONPATH", appDir + "/Python/Lib");
env.insert("PATH", appDir + "/Python/libs;" +
    env.value("PATH"));

//Configuracion de QProcess.
process->setProcessEnvironment(env);
process->setProgram(pythonExe);

//Configuracion del uso elegido
QString code;
if (use == "superficial") {
    ui->s4_skeletonizationButton->setEnabled(true);
    code = QDir(appDir)
        .absoluteFilePath("scripts/openCV
            /superficialCorrection.py");
    process->setArguments(QStringList() <<
        code << imageFilePath << ui->s4_sB_erosion->text() <<
        ui->s4_sB_morphOpening->text());
}
if (use == "skeletotization") {
    code = QDir(appDir)
        .absoluteFilePath("scripts/openCV
            /skeletotizationCorrection.py");
    process->setArguments(QStringList() <<
        code << imageFilePath <<
        ui->s4_sB_skelDilation->text());
```

```
}

//Conexion signal/slot
connect(process, &QProcess::finished, this,
    [=](int exitCode, QProcess::ExitStatus)
{
    progress->close();
    QByteArray stdErr = process->readAllStandardError();

    if (!stdErr.isEmpty()) SecurityManager::
        sc_ERROR("Errors detected while using OpenCV: " +
            stdErr);;
    process->deleteLater();
    progress->deleteLater();
    toSaveProject();

    QString afterImage;
    if (projectHandler
        ->getProjectSetting(Files::PROJECT_FILEPATH)
            .isEmpty())
    {
        afterImage = SettingsHandler::getInstance()
            .getSetting(SettingsKeys::
                keyToString(Key::FILEPATH_TEMP)) +
            "/s4_afterCorrection.png";
    } else {
        afterImage = projectHandler
            ->getProjectSetting(Files::PROJECT_FILEPATH) +
            "/edited_step_images/s4_afterCorrection.png";
    }
    ui->s4_graphicsView->loadImage(
        afterImage,
        "correctionImage");

    SecurityManager::sc_INFO("Finished OpenCV process.");
});

//Comienzo del proceso
process->start();
```

6.3. Implementación de subclases.

El uso de las subclases resulta muy útil y muy fácil con el constructor QMake:

- Se crea un objeto de la clase base y se coloca en la UI.
- Se genera una posibilidad de promoción para todos los objetos de la clase base indicando nombre y archivo cabecera de la subclase.
- Se promociona el objeto configurado a la subclase personalizada y usará los nuevos valores y funciones.

Para acabar con la sección de la implementación expandiré más la implementación de las subclases de objetos de Qt realizadas con la razón principal de su implementación.

CustomQGraphicsView_ClickablePic. La subclase que más complicaciones y necesidades extra ha sido el visor de imágenes interactivo.

Qt consta del objeto **QGraphicsView** que permite mostrar imágenes en un visor.

Las personalizaciones realizadas aplican una resolución y visión que respeta constantemente la relación de aspecto de la imagen, permite devolver la imagen contenida en el visor, permite mostrar todo tipo de imágenes.

Como parte de esta personalización, se añadieron las funcionalidades de recorte y recogida del color en cuanto a señales; y permitió seguir con el paradigma de programación pudiendo crear nuevas funciones para cambiar los colores de la imagen y girarla a través de receptores.

CustomQLabel_Images. Subclase de **QLabel**, visor de contenido no interactuable (generalmente solo texto o pequeños iconos).

Para esta subclase se busca poder usar el visor para enseñar imágenes que no necesitan interacción, pero que respete la relación de aspecto de cualquier imagen sin necesidad de buscar tamaños 100 % proporcionales a la hora de colocarla en la UI.

CustomQLineEdit_newFocus. Clase base **QLineEdit**. Permite al usuario introducir un texto editable.

Con la personalización se busca saber con exactitud cuando el usuario está intentando introducir el texto. Esta interacción la separamos para poder usarla con distintos propósitos por lo que solo emite una señal cuando el QLineEdit entra en foco y en el emisor es donde se realiza la acción.

CustomQSpinBox. Un **QSpinBox** es un selector de números enteros.

La pequeña personalización devuelve, dependiendo del rango de números utilizado, si ya no permite un número más bajo o más alto del ya seleccionado.

CustomQToolBox_StepByStep. Objeto promocionado que permite a la aplicación tener pasos claramente definidos.

La clase base, **QToolBox**, es un objeto contenedor que permite, a través de distintas pestañas, mantener otros objetos en su interior.

Su personalización permite mantener activa y mostrada la pestaña correspondiente al paso del programa actual siempre y sin ninguna posibilidad de cambio externa a los botones de cambio de paso.

QWidgets personalizados. QWidget es la clase base y padre de todas las clases Qt que tienen una representación en la UI.

En esta aplicación se han creado 2 subclases nuevas con base QWidget:

CustomQWidget_ColoursToBinarize. Colección de objetos compuesta por un botón cuya interacción permite el cambio del color contenido, un QSpinBox con el umbral del color seleccionado y otro botón cuya interacción permite el borrado del QWidget.

Esta colección de objetos se personalizó para poder crear de manera rápida e interactiva la lista de colores seleccionados por el usuario en el paso de corrección de colores. Permite tener un modelo y controlador más aislado de la pantalla principal y una conexión más sencilla.

CustomQWidget_SvgPaths. Compuesta por un botón y un QLabel con texto, esta **colección** permite mostrar de manera visual un trazo de una imagen vectorial. Un bucle donde se generan todos los trazos nos lleva a una lista de trazos visualmente seleccionables.

El botón tendrá la función de emitir una señal al ser pulsado especificando el trazo que quiere eliminarse de la imagen.

Capítulo 7

Pruebas

Este capítulo explica las pruebas que se han realizado para demostrar el funcionamiento del sistema.

Al haber seguido la metodología híbrida de modelo en V y Kanban, en la parte del proceso de verificación y validación se realizan **pruebas unitarias, de integración, de sistema y de aceptación**.

El grupo de pruebas realizadas permitieron atajar y arreglar los errores según se iban encontrando para acabar con la robusted esperada en la aplicación.

7.1. Pruebas unitarias

Las **pruebas unitarias** verifican el funcionamiento aislado de una unidad mínima de código. Siguiendo el tablero Kanban, estas unidades mínimas de código son las tarjetas individuales moduladas por lo que en cuanto se termina la programación y la tarjeta se mueve a la columna "Test".

Para estas pruebas unitarias se usó la librería de **QtTest** de manera temporal; se eliminó antes de hacer el ejecutable portable para no meter librerías innecesarias para el usuario.

Para un uso correcto de QtTest se realiza un constructor extra de la aplicación añadiendo la librería necesaria y dejando fuera del projecto al archivo "main.cpp":

Listing 7.1: VectorizarT.pro - Librería QtTest

```
1 QT += testlib
2 //Previene la instalacion de ejecutable para el test.
3 CONFIG += no_testcase_installs
```

También se crea la clase base que hará el test; una clase con una estructura básica y común que funciona como ejecutable individual. Esta clase contiene la sentencia QTEST_MAIN(TestClass) que sustituye al main.cpp para se ejecute el test con todos los sensores privados de la clase.

En esta clase se pueden definir dos sensores privados que permiten añadir trazabilidad al test pues siempre se ejecutan en un orden específico. Estos dos sensores son initTestCase(), que se ejecutará el primero, y cleanupTestCase(), que se ejecutará el último.

Como ejemplo de uso, aquí muestro la prueba previa que se hizo:

Listing 7.2: testclass.h/testclass.cpp - Clase de test.

```
//TestClass.h
#ifndef TESTCLASS_H
#define TESTCLASS_H

#include <QObject>

class TestClass : public QObject {
    Q_OBJECT
private:
    bool tryCode();

private slots:
    void initTestCase();
    void cleanupTestCase();
    void test();
    void testFalse();
};

#endif // TESTCLASS_H

//TestClass.cpp
#include <testclass.h>
#include <QtTest>
#include <QObject>


bool TestClass::tryCode() {
    return true;
}

void TestClass::initTestCase() {
    qDebug("Init test");
}

void TestClass::cleanupTestCase() {
    qDebug("Finished test");
}
```

```
38
39 void TestClass::test() {
40     QVERIFY(tryCode());
41     QCOMPARE(1, 1);
42 }
43
44 void TestClass::testFalse() {
45     QVERIFY(false);
46     QCOMPARE(1, 2);
47 }
48
49 QTEST_MAIN(TestClass);
```

Su salida de consola 7.1:

```
********* Start testing of TestClass *********
Config: Using QtTest library 6.6.3, Qt 6.6.3 (x86_64-little_endian-llp64 shared (dynamic) release build; by GCC 11.2.0), windows 10
QDEBUG : TestClass::initTestCase() Init test
PASS   : TestClass::initTestCase()
PASS   : TestClass::test()
FAIL!  : TestClass::testFalse() 'false' returned FALSE. ()
..\..\testclass.cpp(24) : failure location
QDEBUG : TestClass::cleanupTestCase() Finished test
PASS   : TestClass::cleanupTestCase()
Totals: 3 passed, 1 failed, 0 skipped, 0 blacklisted, 7ms
********* Finished testing of TestClass *********
```

Figura 7.1: Captura de la salida por consola del test.

Usando QVERIFY(tryCode()) y usando tryCode() (en un sensor sencillo "test()") para llamar a los módulos se comprobaron todos los módulos individualmente devolviendo verdadero o falso dependiendo de si la llamada resultaba en éxito o si resultaba en error.

Todas estas pruebas unitarias se iban haciendo poco a poco según se iban completando y pasando a la columna "Test"; la realización de poder quitar del tablero una tarjeta de manera definitiva resultaba en una gran motivación.

7.2. Pruebas de integración

Las **pruebas de integración** validan que múltiples módulos o componentes trabajan correctamente juntos.

Para estas pruebas se volvió a utilizar la herramienta de QtTest, pero con la funcionalidad completa. De este modo se conseguía comprobar el conjunto trabajando para el fin.

A mayores, se utilizó de una manera mucho más completa la herramienta de **qDebug()**. Un objeto de Qt que permite lanzar un mensaje específico a la consola de salida.

Estos dos elementos de Qt se hicieron uso siguiendo una consecución de **pruebas de caja negra** (que es en lo que se basa QtTest) con **pruebas de caja blanca**. Las pruebas

de caja negra se centran únicamente en la **funcionalidad externa**, sin importar cómo esté implementado; sólo fijándose si los atributos necesarios para comenzar la funcionalidad resultan en el desenlace esperado. Sin embargo, las pruebas de caja blanca analizan el **interior del código fuente** permitiendo detectar los errores exactos en la lógica interna.

QDebug se utilizó cuando QtTest avisaba del **resultado anómalo** de las pruebas. Se aplicaba entonces qDebug en todos los puntos diferentes y conflictivos de la funcionalidad y permitía seguir el camino trazado por la ejecución y detectar de manera inmediata los fallos.

Estas pruebas se expandieron con qDebug especialmente para las funcionalidades de corrección de errores y vectorización pues había muchos puntos de la funcionalidad completa que necesitaban pruebas exhaustivas.

Las pruebas de integración se realizaron en cuanto todos los módulos correspondientes a una tarjeta de requisito/caso de uso estaban programados, probados y funcionales individualmente. De este modo permitía mantener el foco de atención de una manera mucho más acertada en el resto de funcionalidades al saber que el trabajo pasado funcionaba a la perfección.

7.3. Pruebas de sistema

Las **pruebas de sistema** se realizan sobre la aplicación completa, en su entorno real o simulado. Su objetivo es comprobar que el sistema cumple con todos los requisitos funcionales y no funcionales.

En estas pruebas todo el software funciona como un único bloque.

Para la realización de estas comprobaciones se estuvo probando el programa con distintos pequeños trazos realizados en un papel a mano alzada y con "mala calidad de escaneado" para tener una visión completa la funcionalidad 7.2.

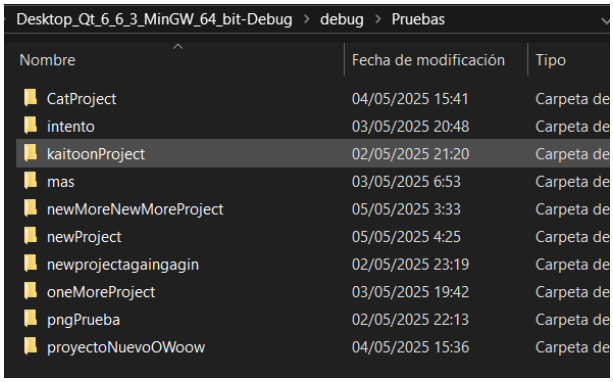

Figura 7.2: Captura de proyectos realizados durante las pruebas de sistema.

De manera paralela, se utilizó la propia herramienta implementada por seguridad del historial para poder seguir el trazado y poder tener una prueba mixta de caja negra y blanca al mismo tiempo.

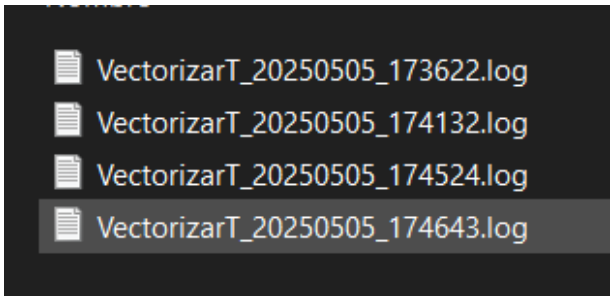

Figura 7.3: Captura de historiales guardados durante las pruebas de sistema.

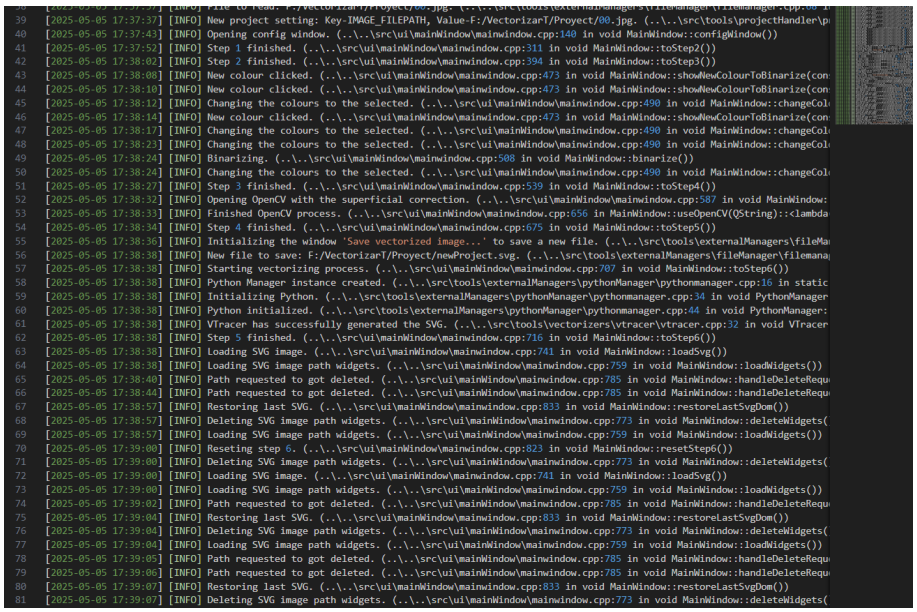

Figura 7.4: Captura del interior de un historial tras una prueba de sistema.

7.4. Pruebas de validación

Las **pruebas de aceptación** son realizadas por usuarios finales para confirmar que el producto cumple con las necesidades y con los objetivos del proyecto.

En estas últimas pruebas se dejó a varios usuarios probar el programa y, tras recoger las opiniones, se llegó a la decisión posible de realizar un manual de usuario completo para ayudar con el primer uso.

Los usuarios encontraron muy cómodo el sistema tras enseñarles su uso y acababan contentos con el resultado.

El resto de comentarios se explorarán en los apartados de *Conclusiones* y en *Trabajo a futuro*.

Parte III

Conclusiones y mejoras

Capítulo 8

Conclusiones

Tras el desarrollo completo de este Trabajo Fin de Grado, puedo afirmar con satisfacción que **todos los objetivos planteados inicialmente han sido cumplidos** de forma íntegra. La aplicación final es plenamente funcional, cumple con los requisitos tanto funcionales como no funcionales definidos, y representa fielmente la idea original que se buscaba alcanzar.

Este proyecto ha supuesto un verdadero reto superado, especialmente en lo referente a la adquisición de conocimientos profundos en el desarrollo con Qt y en la programación de aplicaciones de escritorio. Al inicio del trabajo, el entorno de Qt y el desarrollo en C++ para escritorio suponían un terreno poco explorado a esta escala, pero gracias a una intensa dedicación y a una metodología organizada, se ha conseguido no solo dominar estas herramientas, sino también aplicarlas con criterio para resolver problemas reales.

El proceso ha estado acompañado de un **aprendizaje continuo**, tanto en las fases de investigación como durante el desarrollo práctico. El contraste entre teoría y práctica ha sido especialmente enriquecedor, permitiéndome adquirir una visión más completa de todo el ciclo del desarrollo de software, desde el análisis hasta las pruebas y la documentación final. Además, gracias al aprendizaje de LaTeX, la documentación nunca más será la pesadilla que era con maquetaciones fantasma.

8.1. Reflexiones personales

Más allá de lo académico, este trabajo tiene un componente emocional y personal muy fuerte. Desde el primer año del Grado soñaba con poder desarrollar una herramienta como esta: útil, concreta y que, además, pudiera utilizar en mi día a día. Poder haber hecho realidad esa idea inicial, a través de mi propio esfuerzo y aprendizaje, es una fuente de orgullo personal que me acompañará para siempre. Saber que **usaré esta aplicación de manera habitual** le otorga aún más valor al trabajo realizado y confirma que ha sido un proyecto verdaderamente significativo en mi trayectoria formativa.

Capítulo 9

Trabajo a futuro

Debido a la ambición personal con este proyecto y la posibilidad escalable del proyecto gracias a la metodología de trabajo adoptada y el paradigma elegido, sería muy posible y sencillo continuar implementando módulos para mejorar y expandir la aplicación.

Teniendo en cuenta los comentarios recibidos tras las pruebas de aceptación, mejoras que por falta de tiempo y conocimiento no se pudieron realizar, expansión de contenidos, y actualización de elementos, algunos de los trabajos a futuro posibles son:

- **Guía de ayuda y tutorial en vivo.** Realizar a través de Qt, un tutorial superpuesto, que permita ser saltado si el usuario lo decide, a la interfaz de la aplicación para ayudar al nuevo usuario a utilizar la interfaz, los datos y la necesidad de cada uno para cada parte del proceso. Como resumen: hacer el programa aún más explicativo.

  También tendría botones de ayuda explicativos y maneras de hacer que la guía interactiva en vivo se ejecutase a placer del usuario.

- **Un módulo nuevo de idiomas.** Realizar un módulo de idiomas para que cualquier persona pueda realizar una traducción, cargarla, distribuirla... Como prueba de su uso se realizaría paralelamente una traducción al español del programa para comprobaciones.

- **Nuevas formas de vectorización.** Añadir nuevos módulos que realicen vectorizaciones y explotar todas las posibilidades de los mismos para que el usuario pueda decidir el algoritmo a utilizar. Entre los nuevos módulos se realizaría uno personal utilizando el código descartado en Octave con el que se intentaba este fin.

- **Mayor personalización.** Añadir nuevas posibilidades para que el usuario pueda tener más personalización: nuevos temas, distintos colores de temas, dar la posibilidad al usuario de añadir imágenes que utilizar como cursor, iconos dentro de la aplicación, etc.

- **Menor número de parámetros por defecto.** Paralelo a la guía de ayuda, se le permitiría al usuario elegir más valores que hacen que el proceso varíe. Esta mejora va de la mano con la siguiente...

- **Mejor fluidez y mejora en los ajustes.** Meter muchas mejoras acabaría hundiendo al programa en el mismo pozo en el que se encuentran otros grandes programas.

  Con la mejora de fluidez permitiría a los usuarios más experimentados saltarse ciertos pasos. El programa, con la mejora de los parámetros, podría aceptar un proyecto antiguo para que automáticamente y sin tener que elegirlo en cada ejecución, tenga los valores por defecto.

- **Multiplataforma.** Realizar compilaciones compatibles con otros sistemas operativos como Linux o incluso dispositivos Android.

  Esto resulta complicado en el programa actual pues, aunque Qt permite construir para ellos, el uso de Python embebido nos ata al sistema operativo. Con esta mejora también vendría la necesidad de una compilación sin dependencias de Python usando todo desde Qt. Esto supondría traducir todo el código al Qt Python o añadir las dependencias base a Qt C++ de OpenCV y de VTracer, siendo este último el problema al tener que programar y compilar la biblioteca de VTracer usando Rust primero.

Parte IV

Manuales y otros Apéndices

Apéndice A

Manuales

Dados los comentarios recibidos durante las pruebas de aceptación se han preparado manuales de instalación y de uso para los usuarios.

Estos manuales se realizan en español e inglés para documentar correctamente este TFG y para informar al general de los usuarios.

A.1. Manual de Instalación

La aplicación **VectorizarT** funciona en todos los dispositivos Windows y no tiene prerrequisitos mínimos. Es posible que el tiempo de generación y limpieza de imágenes tarde más de lo normal en dispositivos de bajas especificaciones o imágenes grandes, pero su funcionamiento está garantizado.

Para la instalación del programa será necesario tener en el ordenador cualquier programa que permita descomprimir archivos ".zip". Mi recomendación personal es 7-Zip[1], programa gratuito y open source que permite comprimir y descomprimir en muchos formatos (.zip incluido).

Para descargar la aplicación se debe acceder al proyecto compilado en Github[2]. En la página, se accede a la descarga del ZIP a través del botón verde de "<>Code" A.1.

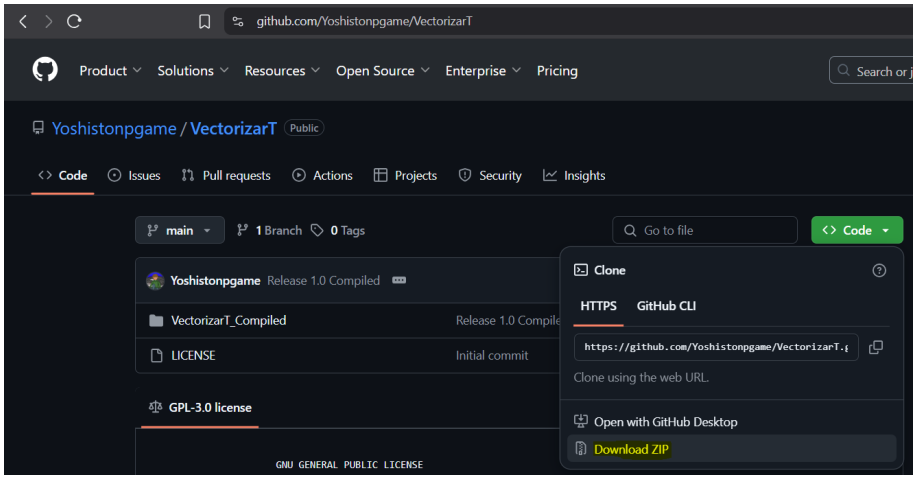

Figura A.1: Página de descarga de Github.

Con el archivo .zip descargado y colocado en la carpeta deseada, se **extraen sus componentes** usando la extensión de 7-zip (o cualquier otro programa) con el clic derecho sobre el archivo.

Todos los archivos que se encuentran en esa carpeta pertenecen al programa y deben mantenerse, cualquier movimiento o borrado de archivos internos puede producir un malfuncionamiento (la carpeta completa puede moverse a cualquier parte siempre y cuando la aplicación no esté en ejecución).

La aplicación estaría **descargada y funcional** en este punto. La compilación realizada mantiene completamente portable el programa sin necesidad de tener instalado Qt, Python o ninguna de sus librerías.

[1] `www.7-zip.org`
[2] `www.github.com/Yoshistonpgame/VectorizarT`

Adicionalmente, se recomienda general un **acceso directo al ejecutable** para poder acceder al programa sin necesidad de entrar a la carpeta base.

A.2. Installation Manual

The application **VectorizarT** works on all Windows devices and has no minimum prerequisites. The image generation and cleanup process may take longer than usual on low-spec devices or with large iamges, but its functionality is guaranteed.

To install the program, you will need any software on the computer that can extract ".zip" files. My personal recommendation is 7-Zip[3], a free and open-source program that can compress and extract in many formats (including .zip).

To download the application, go to the compiled project on Github[4]. On the page, download the ZIP file using the green "<>Code" button.

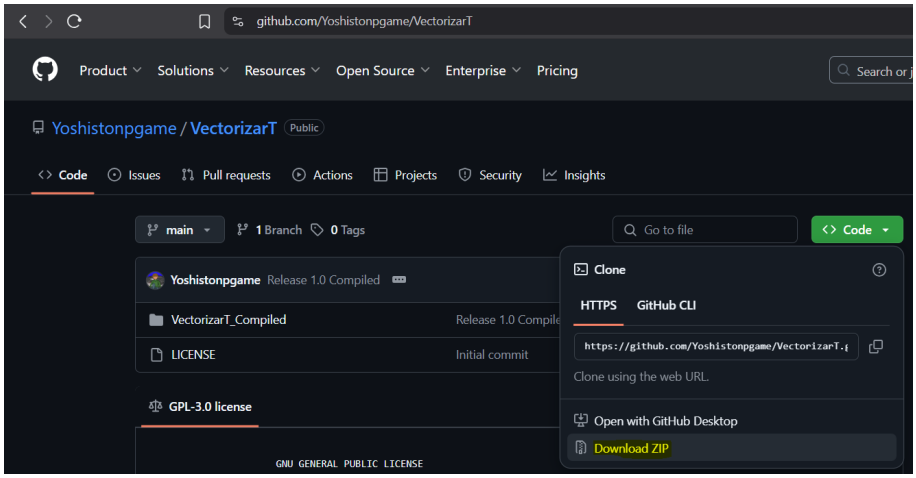

Figura A.2: Download Github page.

Once the .zip file is downloaded and placed in the desired folder, **extract its contents** using the 7-zip extension (or any other program) by right-clicking on the file.

All files in that folder belong to the program and must be kept. Moving or deleting internal files may cause the program to malfunction (the whole folder can be moved anywhere as long as the application is not running).

At this point, the application is **downloaded and functional**. The compiled version is fully portable and does not require Qt, Python or any of their libraries to be installed.

Additionally, it is recommended to create a **shortcut to the executable** for easier access to the program without needing to enter the base folder.

[3] `www.7-zip.org`
[4] `www.github.com/Yoshistonpgame/VectorizarT`

A.3. Manual de Usuario

Comenzamos la aplicación usando el enlace directo o haciendo doble clic en el archivo ejecutable de la aplicación "VectorizarT.exe".

1. En la primera pantalla nos encontramos el paso de bienvenida. Empezaremos el proceso haciendo clic en el botón "Get Started".

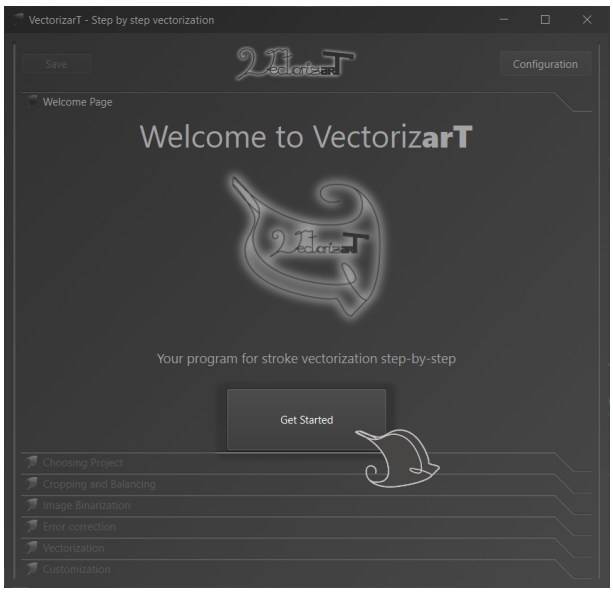

Figura A.3: Paso de bienvenida de la aplicación.

2. En el primer paso nos encontramos la configuración del proyecto inicial. Podremos elegir el nombre y la ruta del proyecto, y la imagen que queremos procesar.

   En este paso solo es obligatorio elegir la imagen. El nombre del proyecto y la ruta del mismo solo son datos necesarios a la hora de guardar el proyecto.

   A la hora de elegir ruta de proyecto se abrirá una ventana que permite seleccionar la carpeta que se quiere utilizar de directorio. Una ventana parecida se abrirá para seleccionar la imagen, sólo las imágenes válidas serán mostradas en esta visión del sistema de archivos.

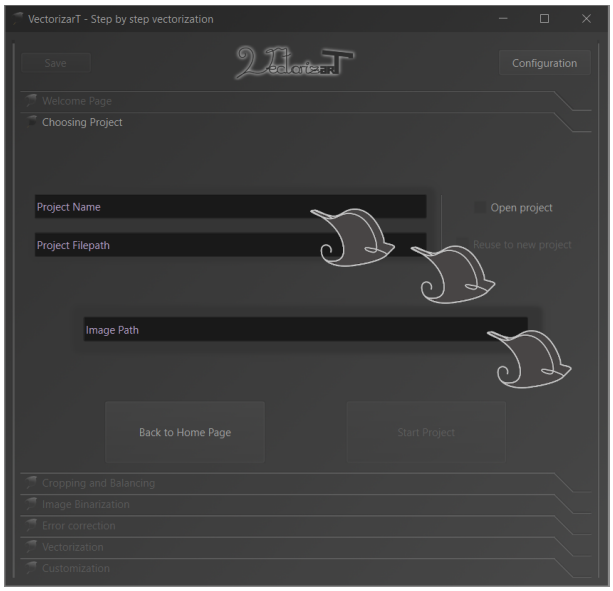

Figura A.4: Primer paso de la aplicación.

Con la imagen válida seleccionada, el botón se activa para pasar al siguiente paso.

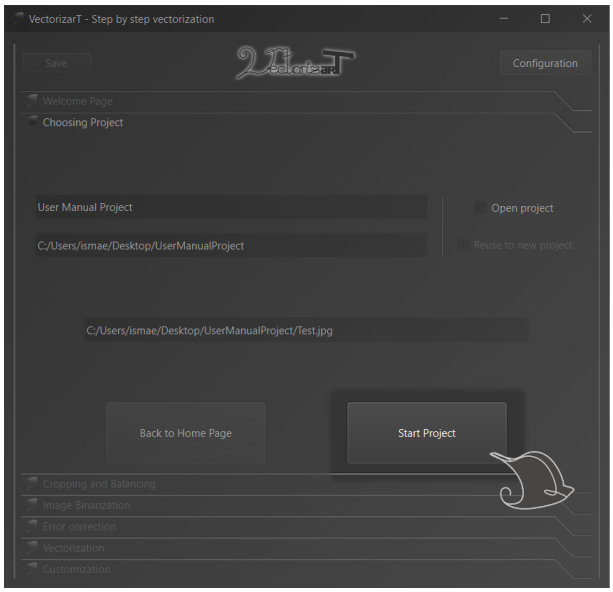

Figura A.5: Primer paso de la aplicación. Cambio de paso.

En este primer paso nos encontramos con la posibilidad de "Abrir un proyecto", esta caja de validación nos permite elegir un proyecto guardado con una ventana auxiliar para continuar con su proceso.

La carga de un archivo válido ".vectorizart" cargará los datos necesarios para continuar al siguiente paso (esta carga no permite seleccionar una nueva imagen, nombre

de proyecto o ruta).

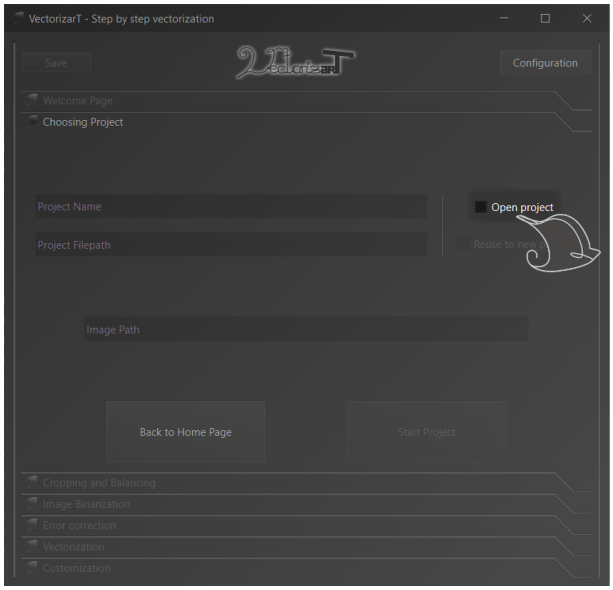

Figura A.6: Primer paso de la aplicación. Uso de proyecto guardado.

De la misma manera podremos "Reusar un proyecto" con la nueva caja de validación activa que nos permitirá usar los valores guardados en el proyecto antiguo para generar un nuevo proceso (activará la posibilidad de cambio de configuraciones).

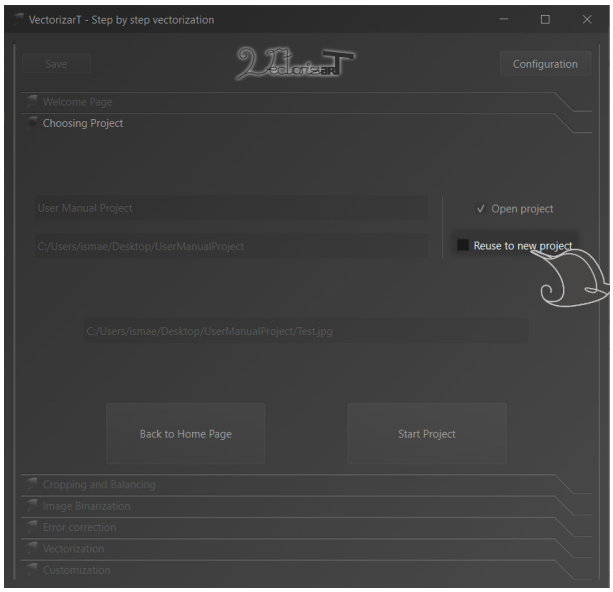

Figura A.7: Primer paso de la aplicación. Reuso de proyecto.

3. Al pasar al segundo paso, el botón de guardado empezará a estar habilitado. Este

botón permite guardar el proyecto siempre que está activo (si está deshabilitado significa que el archivo se ha guardado y no hay cambios nuevos).

Si en el primer paso se han seleccionado nombre de proyecto y ruta, el botón simplemente guarda el archivo.

Si no se especificó ruta, aparece una ventana auxiliar para seleccionar el lugar donde se guardará el archivo del proyecto. El nombre del archivo y la ruta de guardado se reutilizarán para nombre y ruta de proyecto respectivamente.

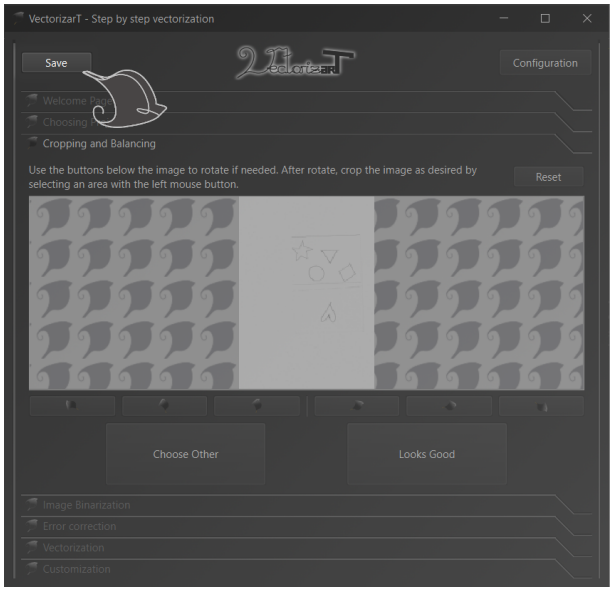

Figura A.8: Guardado del proyecto.

4. En el segundo paso podremos girar la imagen seleccionada a través de los seis botones situados debajo del visor de la imagen.

   Podremos girar la imagen 25^{o}, 5^{o} y 1^{o} a la izquierda y 1^{o}, 5^{o} y 25^{o} a la derecha, con los respectivos botones situados de izquierda a derecha.

Figura A.9: Segundo paso de la aplicación. Rotar la imagen.

Dentro del visor de la imagen el usuario puede recortar la imagen seleccionando la zona exacta que quiere vectorizar. El recorte se realizará tras soltar el clic del ratón.

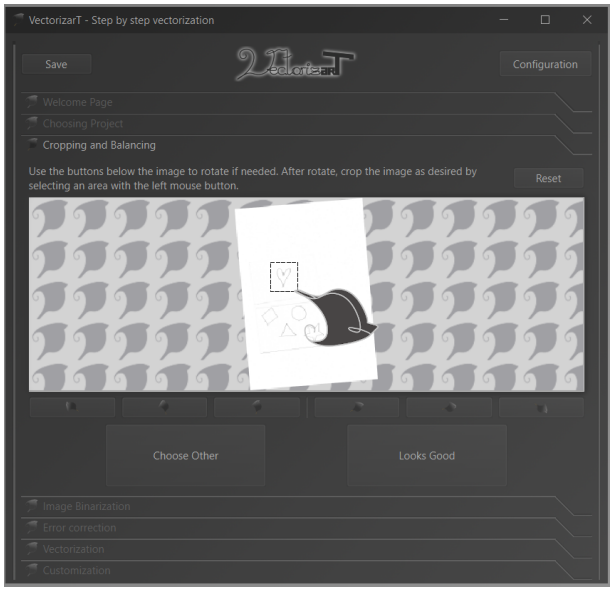

Figura A.10: Segundo paso de la aplicación. Recortar la imagen.

Si se quiere devolver la imagen al original, basta con usar el botón de "Reset" para revertir todos los cambios realizados en el paso.

Figura A.11: En cualquier paso de la aplicación se pueden resetear los cambios realizados.

Cuando el usuario tenga la imagen deseada, pasará al siguiente paso usando el botón "Looks good".

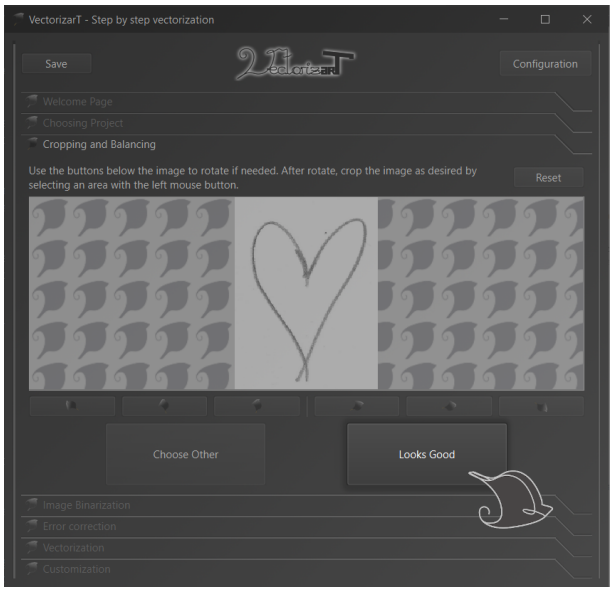

Figura A.12: Segundo paso de la aplicación. Cambio de paso.

5. En el tercer paso se encuentra la corrección de colores.

   Haciendo clic izquierdo en el visor de la imagen añadirá el color seleccionado a la lista de colores a convertir en "negro". Haciendo clic derecho en el visor añadirá el color seleccionado a la lista de colores a convertir en "blanco".

Figura A.13: Tercer paso de la aplicación. Añadir color a la lista correspondiente.

Cada vez que se hace clic en un color se genera en la lista correspondiente un botón que permite cambiar el color a través de una ventana auxiliar, un selector de números para poner el umbral deseado del color (colores adyacentes al color seleccionado que serán tratados como el seleccionado) y un botón que permite eliminar el color de la lista.

Figura A.14: Tercer paso de la aplicación. Editar o eliminar el color seleccionado.

Con los colores deseados seleccionados se puede realizar la conversión para transformar los colores usando el botón de “Convert colours”.

Figura A.15: Tercer paso de la aplicación. Conversión de colores.

También se puede realizar este cambio de colores a través de una binarización de umbral predeterminada que transforma los colores cercanos al blanco en "blanco" y todo el resto en "negro" haciendo clic en "Binarize the image".

Lo recomendable es hacer la conversión y luego aplicar la binarización para cubrir todos los colores con seguridad.

Figura A.16: Tercer paso de la aplicación. Binarización automática.

Además, se provee de una funcionalidad de inversión de colores. Esto puede ayudar

a los usuarios a seleccionar los colores o vectorizar imágenes de dibujos trazados con tiza blanca en fondo negro.

Figura A.17: Tercer paso de la aplicación. Inversión de colores.

Cuando la imagen esté preparada se continuará al siguiente paso con el botón "Processed image".

Figura A.18: Tercer paso de la aplicación. Cambio de paso.

6. El cuarto paso corresponde a la correcciones de errores.

El usuario puede modificar los tamaños de los errores a borrar o a rellenar a través de los editores de números.

Con "Surface correction" ejecutará la corrección de erosión y apertura.

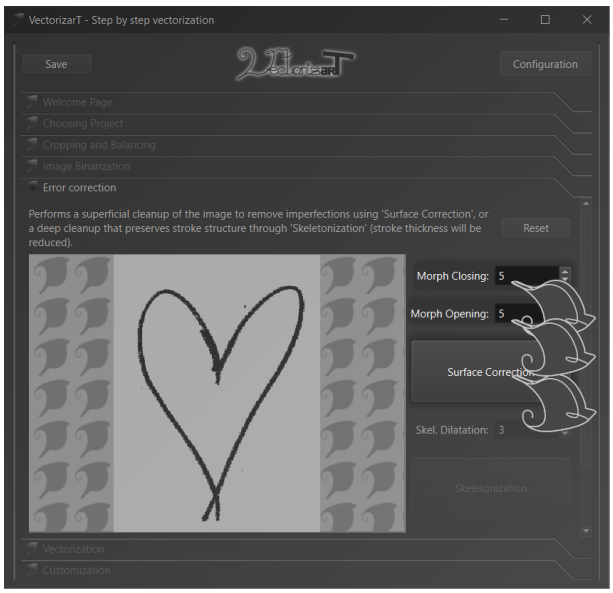

Figura A.19: Cuarto paso de la aplicación. Corrección superficial.

Tras la corrección superficial, el usuario puede hacer una esqueletización y una dilatación si lo que quiere es mantener las direcciones de los trazos y no el grosor. La dilatación puede personalizarla con el selector.

Figura A.20: Cuarto paso de la aplicación. Esqueletizacion y dilatación.

Cuando la corrección sea del agrado del usuario hacer clic en el botón de "Correction done" pasará la imagen al siguiente paso.

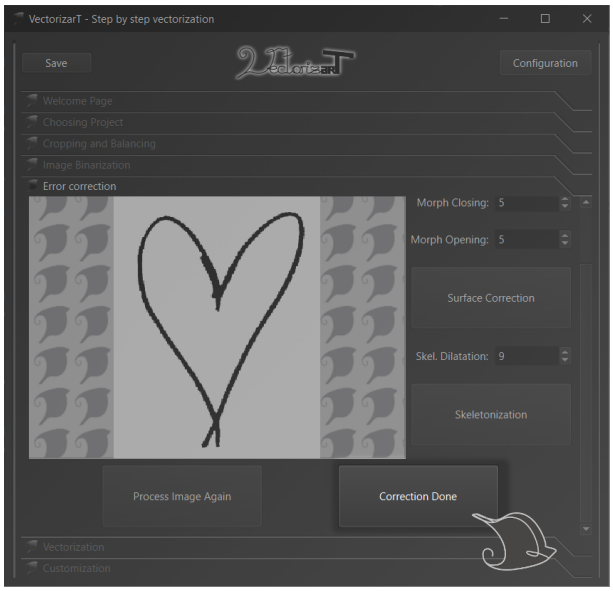

Figura A.21: Cuarto paso de la aplicación. Cambio de paso.

7. El quinto paso realiza la vectorización. Haciendo clic en el botón de "Vectorize" aparecerá una ventana auxiliar para indicar el nombre y ruta del archivo vectorizado resultante.

   Tras terminarse el proceso de vectorizar se pasará al siguiente paso (la imagen vectorizada estará disponible en la ruta especificada).

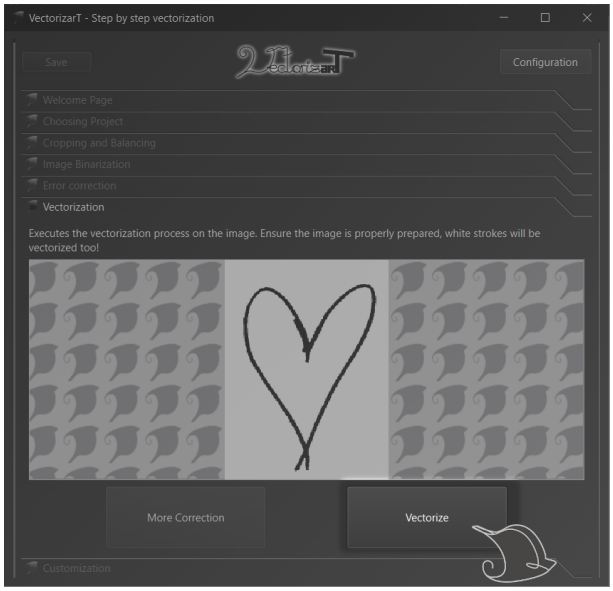

Figura A.22: Quinto paso de la aplicación. Vectorizado.

8. El último paso deja al usuario realizar una personalización del archivo vectorizado permitiendo eliminar trazos.

   A través del listado de trazos de la imagen, el usuario puede elegir el trazo que eliminar haciendo clic en el botón de eliminar.

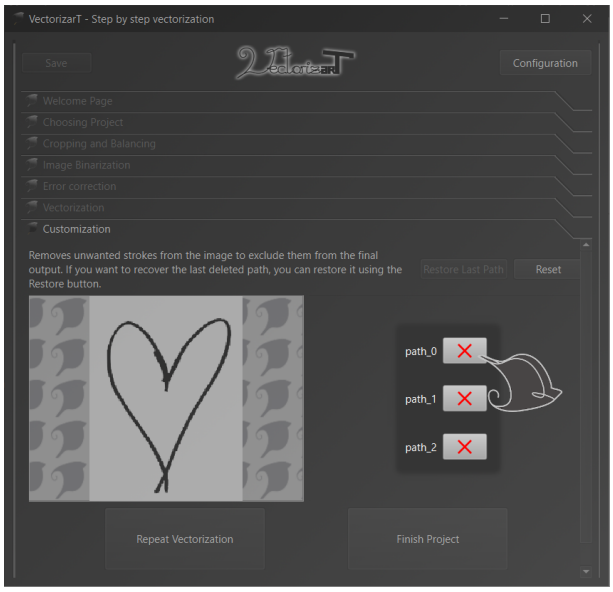

Figura A.23: Sexto paso de la aplicación. Borrado de trazos.

Si el trazo recientemente eliminado se quiere recuperar, se podrá restaurar la anterior modificación realizada haciendo clic en el botón "Restore last path".

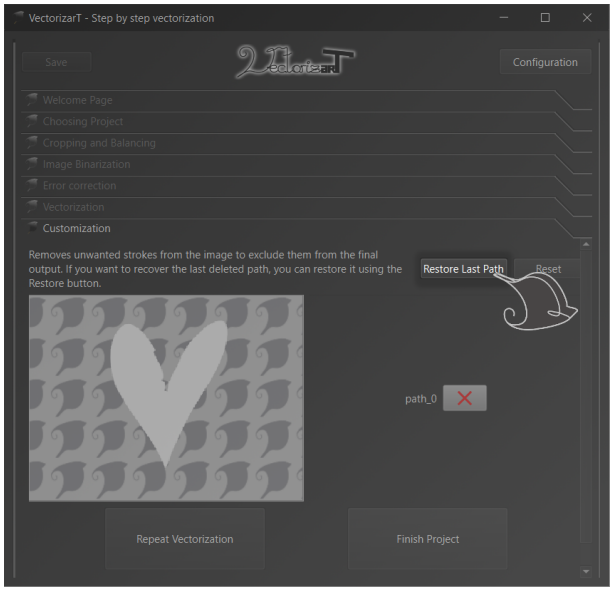

Figura A.24: Sexto paso de la aplicación. Restaurar trazo.

A la hora de terminar el proyecto, cuando se hace clic en el botón de "Finish project", permite al usuario guardar la imagen vectorizada personalizada con otra ventana auxiliar.

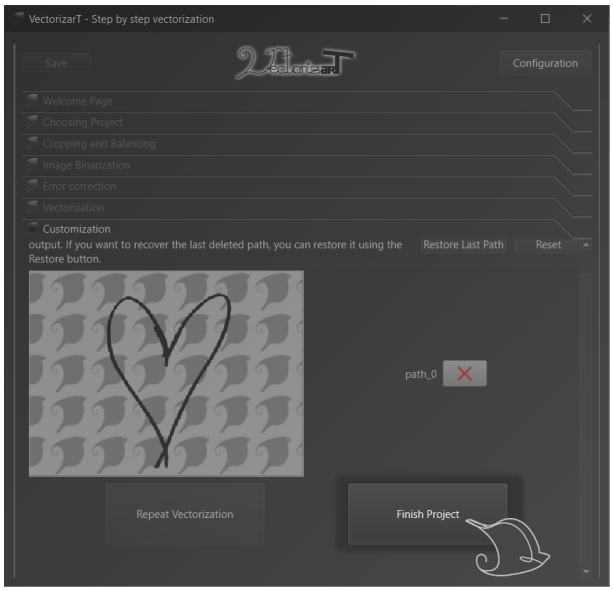

Figura A.25: Sexto paso de la aplicación. Finalizar proyecto.

Además de los pasos propios del proyecto, la aplicación cuenta con configuración personalizable accesible en cualquier momento desde el botón de "Configuration".

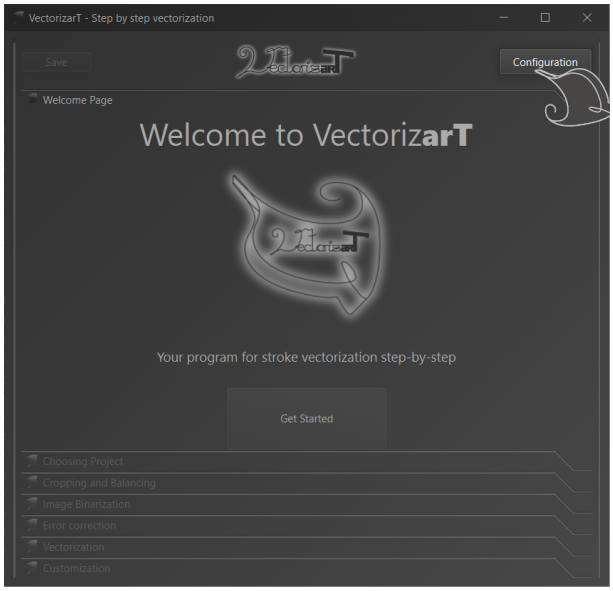

Figura A.26: Configuración de la aplicación.

Esta nueva ventana contiene la configuración del cursor, del tema, del historial y de ajustes.

1. La primera pestaña corresponde a la configuración del cursor donde se le permite al usuario elegir un cursor especial para la aplicación.

   Estos cursores tienen una variedad de colores y también se le permite eliminar cualquier cursor y elegir el predeterminado por el sistema operativo.

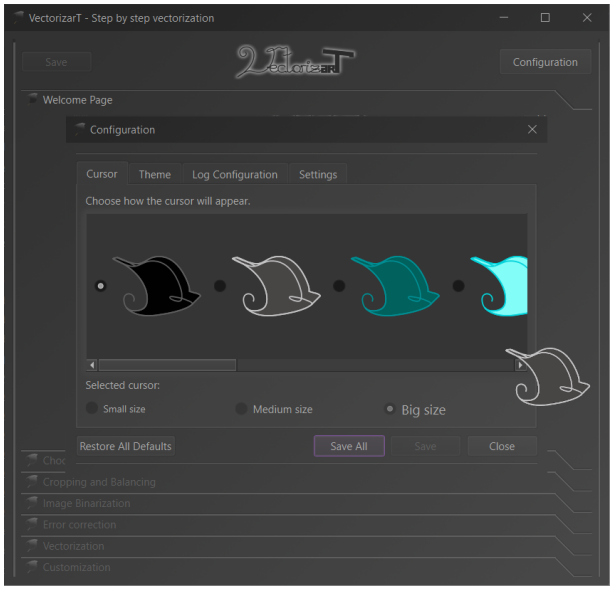

Figura A.27: Configuración de la aplicación. Cursor.

También se permite al usuario elegir el tamaño del cursor.

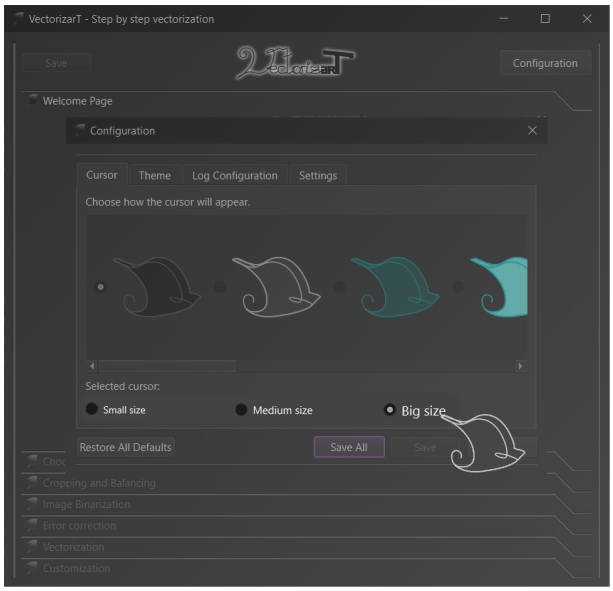

Figura A.28: Configuración de la aplicación. Tamaño del cursor.

2. En la pestaña del tema se puede seleccionar entre dos temas disponibles: tema oscuro o tema claro.

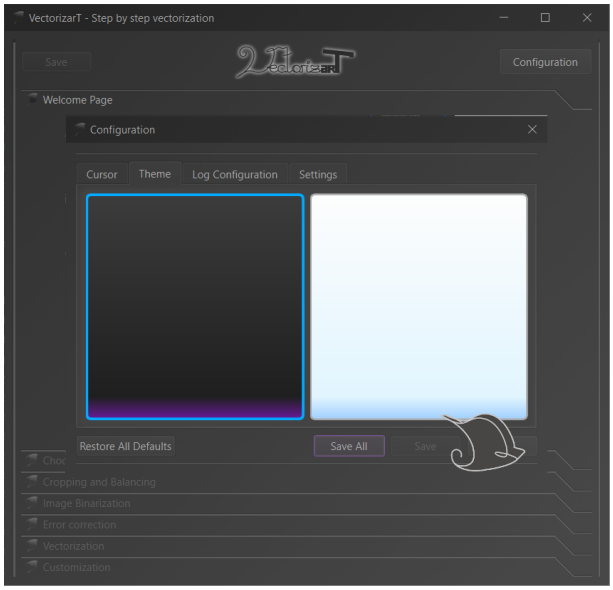

Figura A.29: Configuración de la aplicación. Tema de colores.

3. En la tercera pestaña se encuentran las configuraciones del historial.

   Su primera configuración corresponde al uso del historial, si se acepta el uso o no.

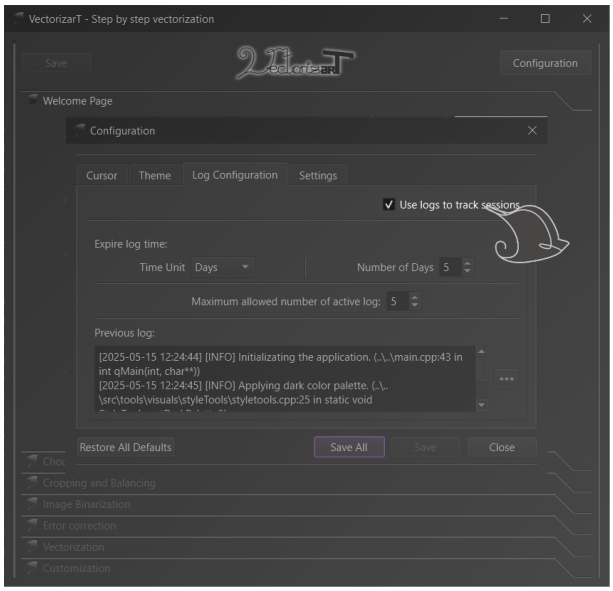

Figura A.30: Configuración de la aplicación. Uso del historial.

Su siguiente configuración es sobre el tiempo de expiración que tendrán los historiales y el número máximo posible de historiales.

El tiempo de expiración tiene distintas unidades asignadas posibles.

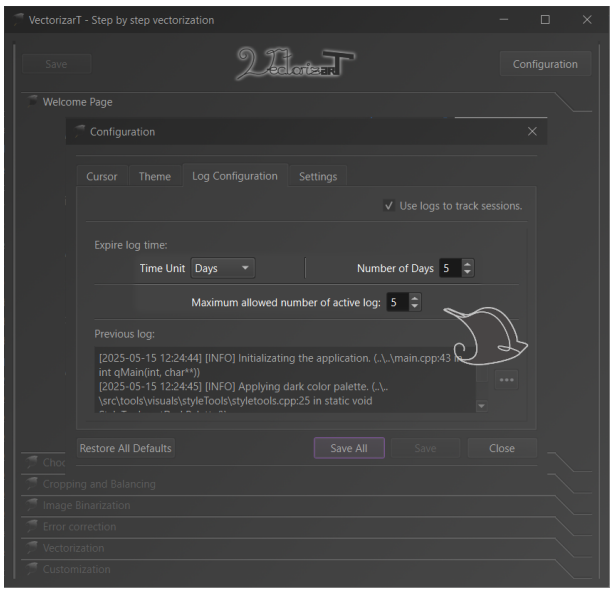

Figura A.31: Configuración de la aplicación. Expiración del historial.

También muestra el anterior historial generado y un botón que permite cambiar el lugar donde los historiales se generarán.

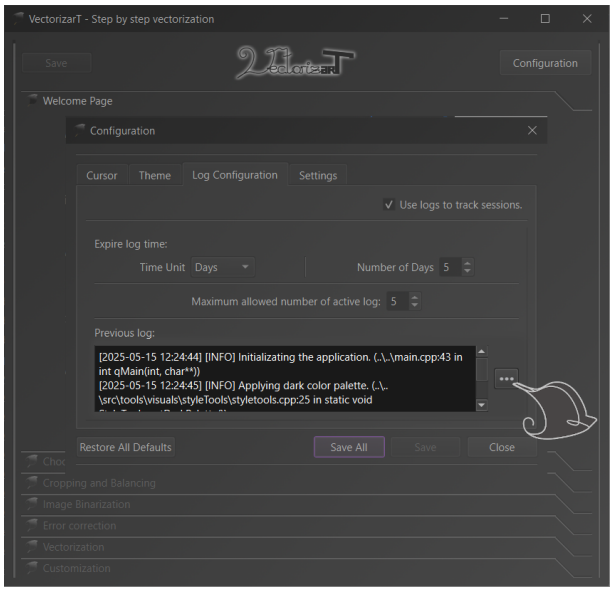

Figura A.32: Configuración de la aplicación. Ruta del historial.

4. Por último, la pestaña de los ajustes. En esta pestaña se podrán exportar e importar los ajustes de otros usuarios (o de otro dispositivo).

   En ambos se nos abrirá una ventana auxiliar que nos permitirá seleccionar donde guardar el archivo y qué archivo importar respectivamente.

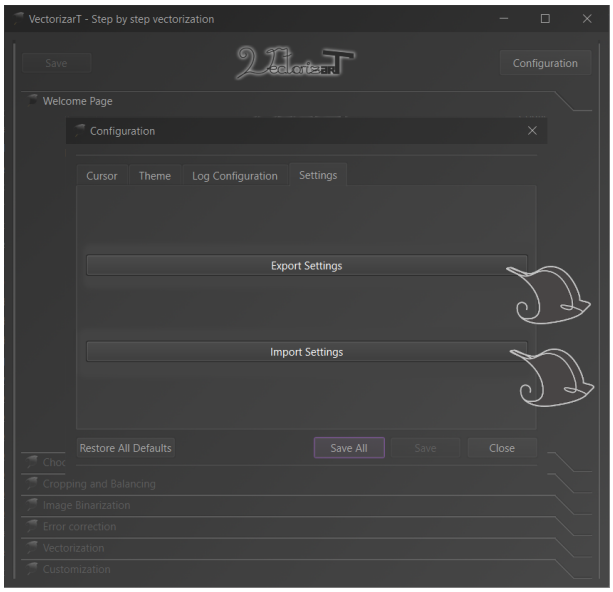

Figura A.33: Configuración de la aplicación. Ajustes.

Para hacer realizar los cambios se utilizarán los botones de la parte inferior de la ventana.

El botón "Restore all defaults" cerrará la ventana y pondrá todos los datos por defecto.

El botón de "Save all" guardará de manera secuencial todas las pestañas de configuración y cerrará la ventana.

El botón de "Save" guardará la pestaña actual. Sólo estará activa cuando haya algún cambio que guardar.

El botón "Cancel" que cerrará la ventana cancelando todos los cambios no guardados.

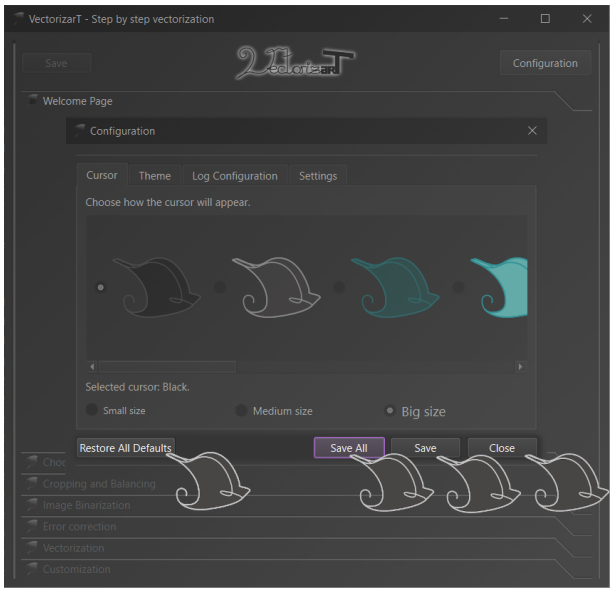

Figura A.34: Configuración de la aplicación. Botones de ventana.

A.4. User Manual

Start the application by using the shortcut or by double-clicking the executable file "VectorizarT.exe".

1. On the first screen, you'll see the welcome step. We begin by clicking the button "Get Started" to start the process.

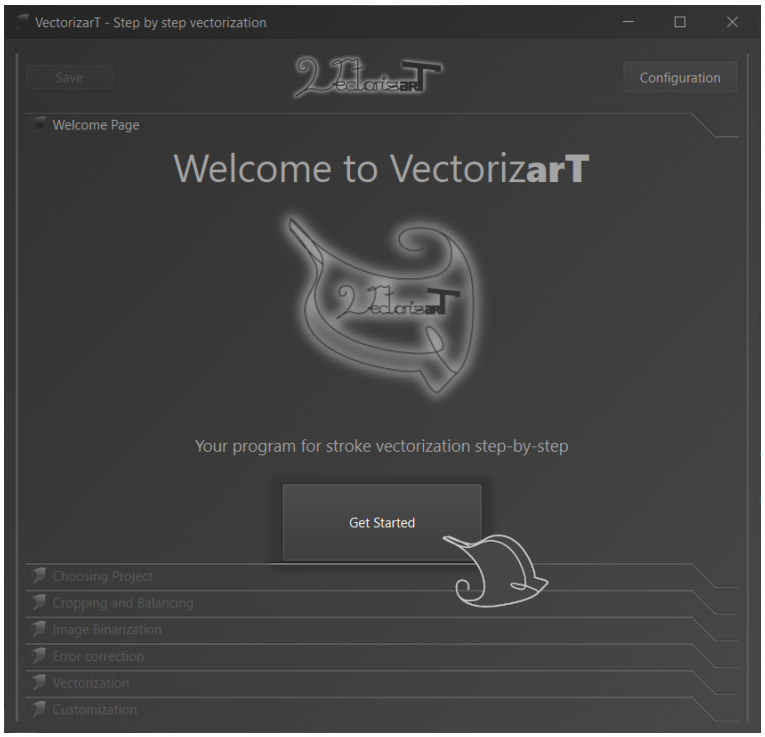

Figura A.35: Welcome screen of the application.

2. In the first step, we are presented with the initial project setup. Here, we can choose the project name, its save location, and the image we want to process.ç

   In this step, only the image selection is mandatory. The project name and its path are only necessary when saving the project.

   When selecting the project directory, a window will open allowing the user to choose the desired folder. A similar window will open to select the image; only valid image files will be shown in this file system view.

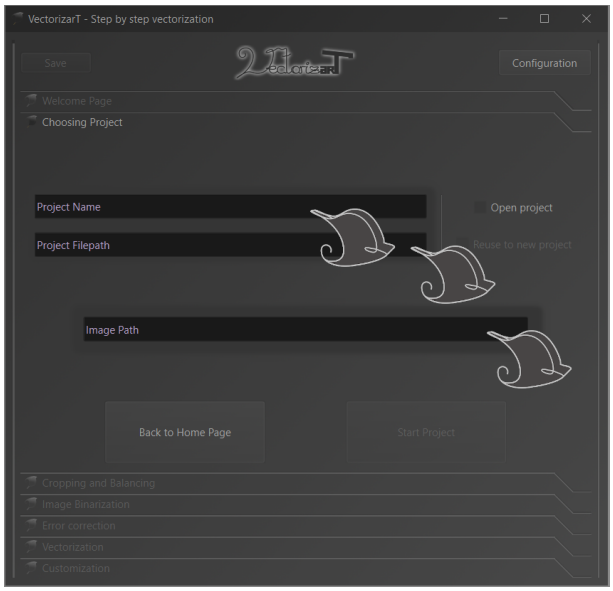

Figura A.36: First step of the application.

Once a valid image is selected, the button is enabled to procced to the next step.

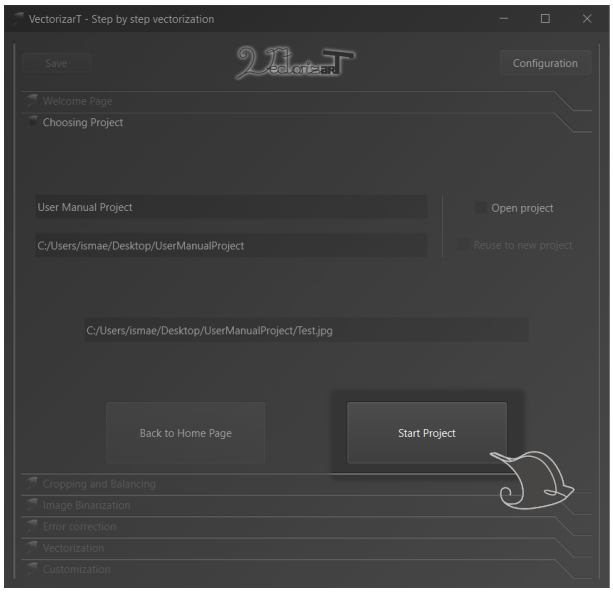

Figura A.37: First step of the application. Transition to the next step.

In this initial step, we also have the option to "Open project". This checkbox allows us to select a previously saved project via a helper window and continue from where we left off.

Loading a valid ".vectorizart" file will load the necessary data to continue to the next step (this process does not allow selecting a new image, project name, or path).

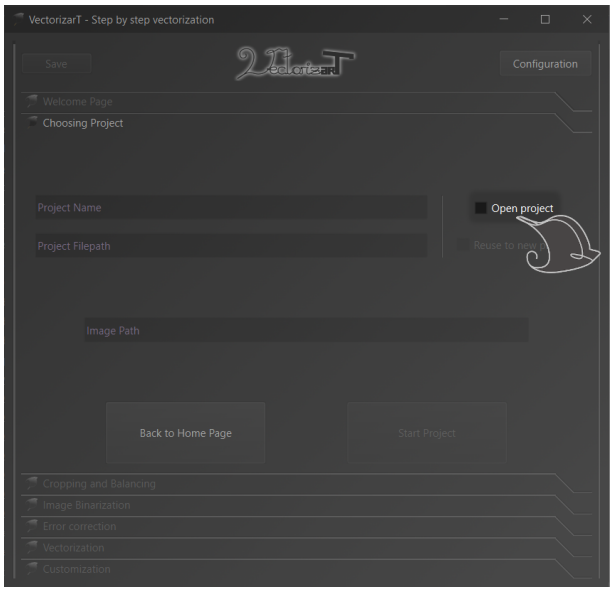

Figura A.38: First step of the application. Using a saved project.

Likewise, we can now "Reuse project" after the checkbox became enabled, which will allow us to use the previously saved project values to generate a new process (this enables the option to change the configurations).

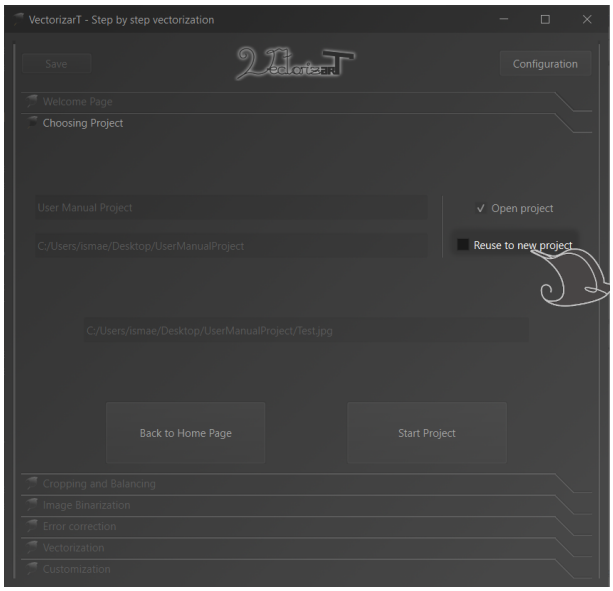

Figura A.39: First step of the application. Reusing a project.

3. When moving to the second step, the save button becomes enabled. This button allows the user to save the project whenever it is active (if it's disabled, it means the file has already been saved and there are no new changes).

If a project name and path were selected in the first step, the button will simply save the file.

If no path was specified, a helper window will appear to let the user choose where to save the project file. The selected file name and save path will then be reused as the project name and directory, respectively.

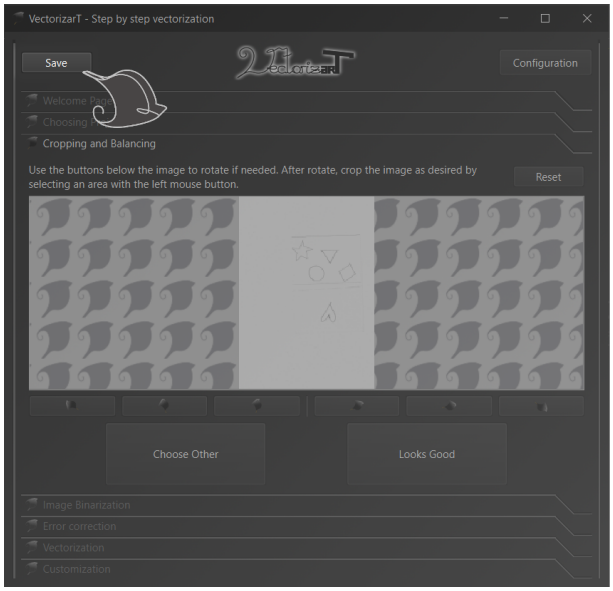

Figura A.40: Saving the project.

4. In the second step, the selected image can be rotated using the six buttons located below the image view.

   The user can rotate the image 25^{o}, 5^{o} and 1^{o} to the left, and 1^{o}, 5^{o} and 25^{o} to the right, using the corresponding buttons from left to right.

Figura A.41: Step two of the application. Rotate the image.

Inside the iamge viewer, the user can crop the image by selecting the exact are they want to vectorize. The crop will be applied after releasing the mouse click.

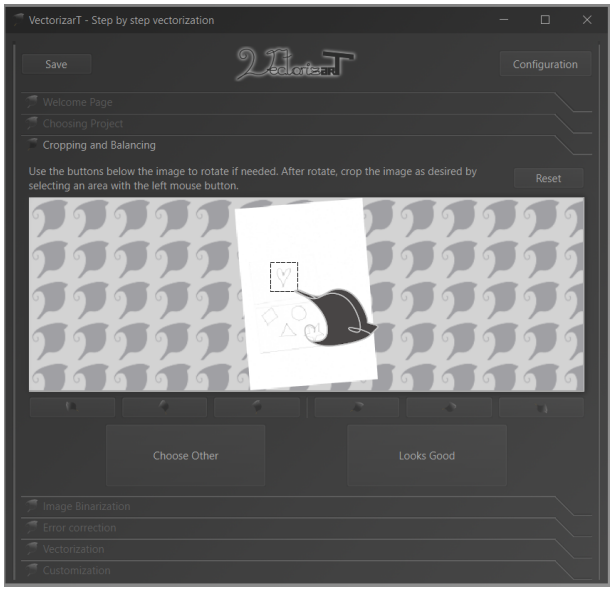

Figura A.42: Step two of the application. Crop the image.

If you want to restore the image to its original state, simply use the “Reset” button to undo all changes made during the step.

Figura A.43: Changes made in the actual step of the application can be reset.

Once the user is satisfied with the image, they can proceed to the next step by clicking the "Looks good" button.

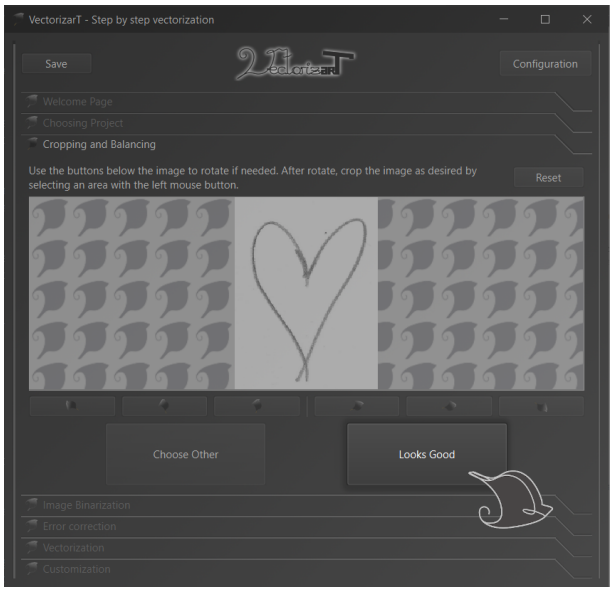

Figura A.44: Step two of the application. Step transition.

5. In the third step, you will find the colour correction process.

   Left-clicking on the image viewer adds the selected colour to the list of colours to be converted to "black". Right-clicking adds the selected colour to the list of colours to be converted to "white".

Figura A.45: Step three of the application. Add colour to the corresponding list.

Each time you click on a colour, a button is added to the corresponding list that allows you to modify the colour using an auxiliary window, a number selector to adjust the threshold value (to include adjacent colours similar to the selected one), and a button that allows you to remove the colour from the list.

Figura A.46: Step three of the application. Edit or delete selected colour.

Once the desired colours are selected, you can perform the conversion by clicking the “Convert colours” button.

Figura A.47: Step three of the application. Colour conversion.

Alternatively, you can apply a default threshold-based binarization by clicking "Binarize the image", which transforms colours close to white into white and everything else into black.

It is recommended to first use colour conversion and then apply binarization to ensure all colours are properly processed.

Figura A.48: Step three of the application. Automatic binarization.

Additionally, a colour inversion feature is provided. This is useful for selecting cer-

tain colours or being able to vectorize drawing made with white chalk on a black background.

Figura A.49: Step three of the application. Colour inversion.

When the image is ready, click the "Processed image" button to continue to the next step.

Figura A.50: Step three of the application. Step transition.

6. The fourth step corresponds to error correction.

The user can adjust the sizes for removing or filling in errors using the numeric input fields.

By clicking "Surface correction", the program will perform erosion and opening corrections.

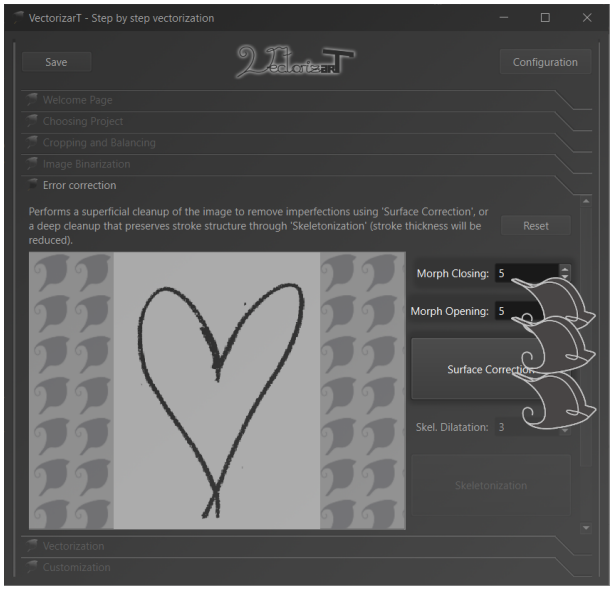

Figura A.51: Fourth step of the application. Surface correction.

After the surface correction, the user can apply skeletonization and dilation, which are useful if the goal is to preserve the direction of the strokes rather their thickness. The amount of dilation can be customizaed with the corresponding selector.

Figura A.52: Fourth step of the application. Skeletonization and dilation.

When the correction results are satisfactory, clicking the "Correction done" button will move the image to the next step.

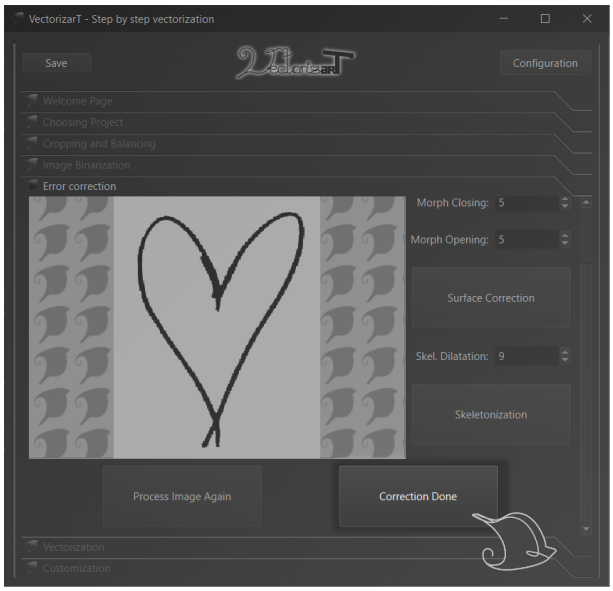

Figura A.53: Fourth step of the application. Step transition.

7. The fifth step performs the vectorization. By clicking the "Vectorize" button, a dialog window will open where the user can specify the name and path of the resulting vector file.

   Once the vectorization process is completed, the application will automatically move to the next step (the vectorized image will be saved in the specified location).

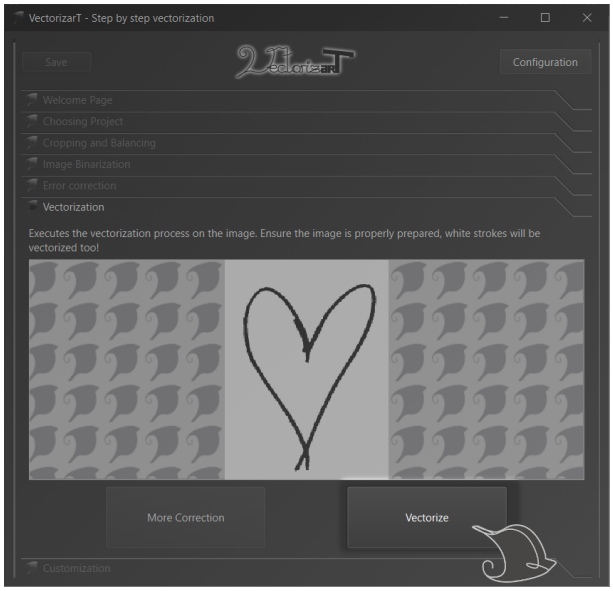

Figura A.54: Fifth step of the application. Vectorization.

8. The final step allows the user to customize the vectorized file by enabling the deletion of individual paths.

   Using the list of image paths, the user can choose which path to delete by clicking the delete button.

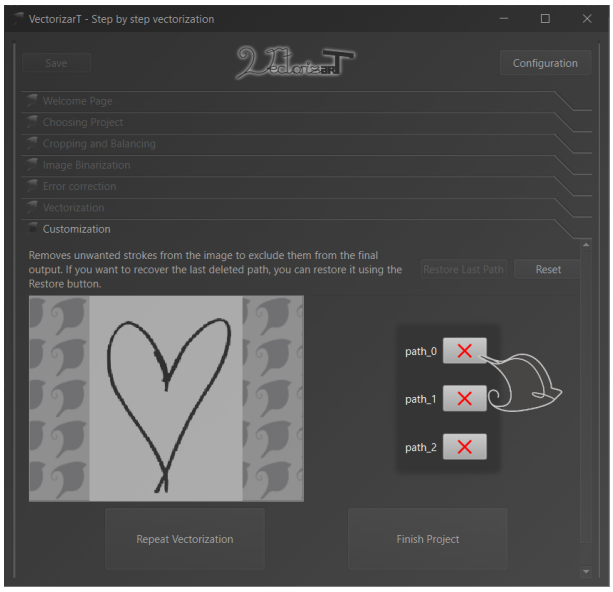

Figura A.55: Sixth step of the application. Deleting paths.

If a recently deleted path needs to be restored, the previous modification can be undone by clicking the "Restore last path" button.

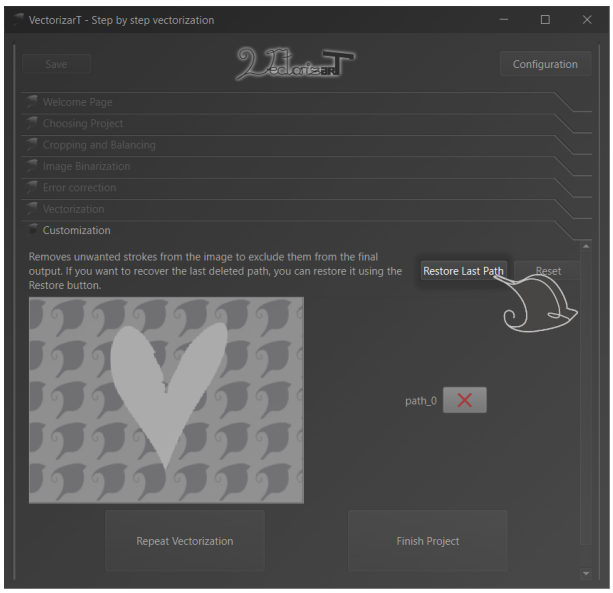

Figura A.56: Sixth step of the application. Restoring a path.

To finalize the project, clicking on the "Finish project" button opens a dialog window that allows the user to save the customized vectorized image.

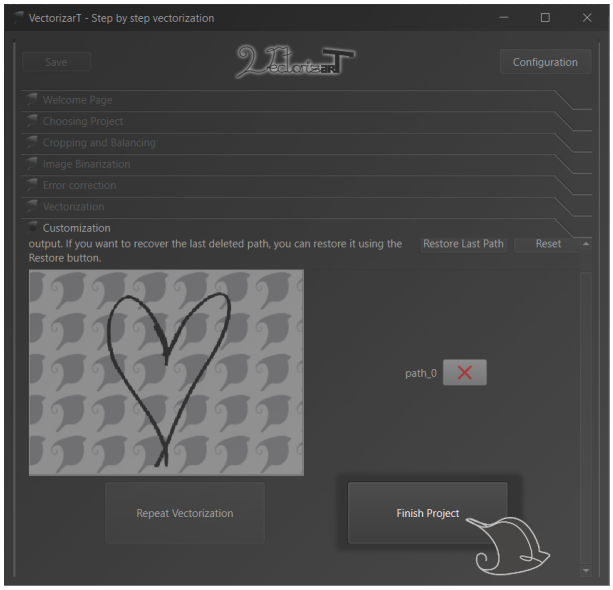

Figura A.57: Sixth step of the application. Finish project.

In addition to the core project steps, tha application includes customizable settings accessible at any time via the "Configuration" button.

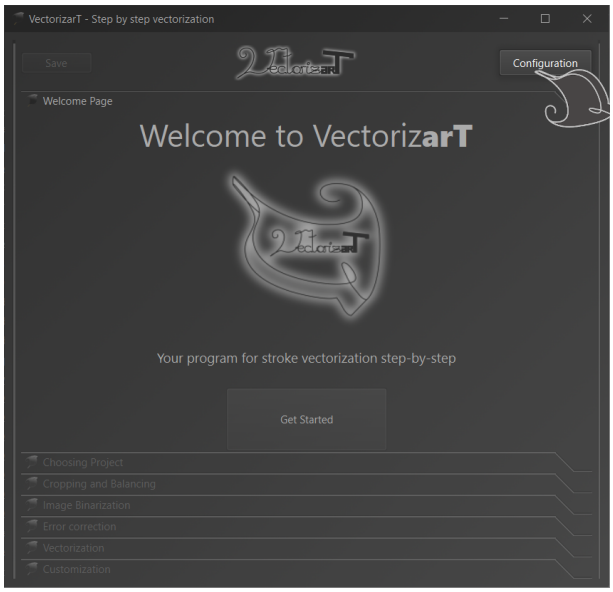

Figura A.58: Application configuration.

This new window contains settings for cursor, theme, log history, and settings.

1. The first tab corresponds to cursor settings, where users can choose a special cursor for the application.

   These cursors come in various colors, and users can also select the default system cursor.

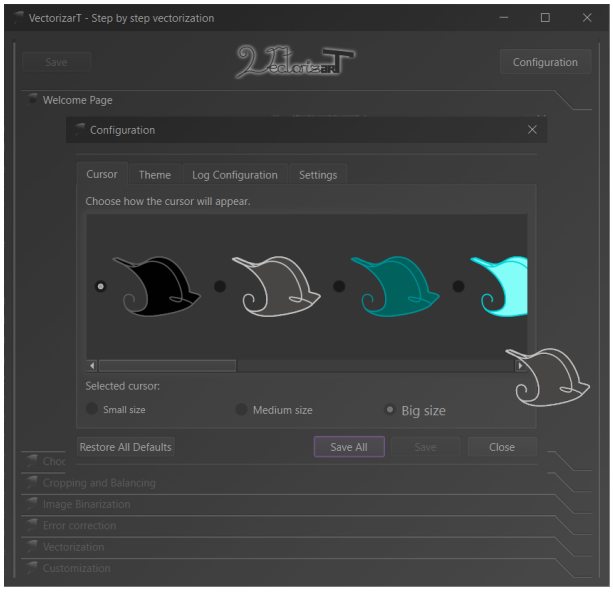

Figura A.59: Application configuration. Cursor.

Users can also select the cursor size.

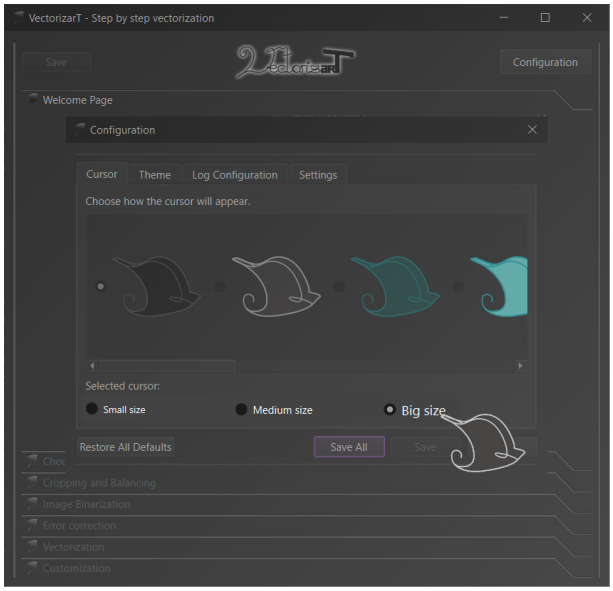

Figura A.60: Application configuration. Cursor size.

2. In the theme tab, users can choose between two available themes: dark theme or light theme.

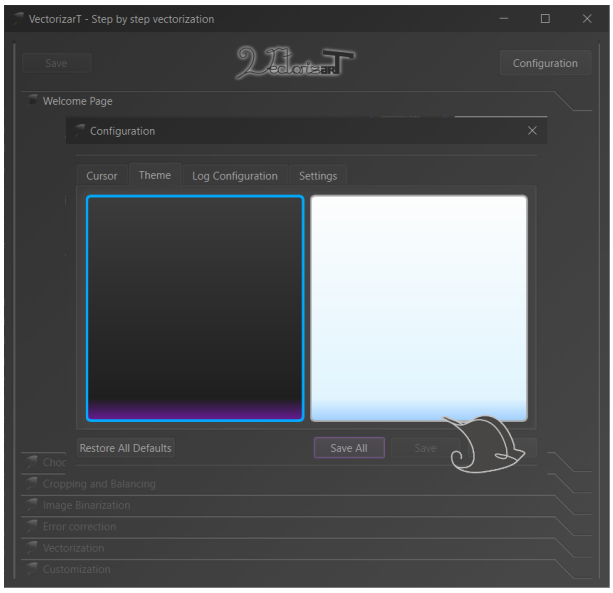

Figura A.61: Application configuration. Colour theme.

3. In the third tab, you can find the log configuration settings.

   The first option refers to whether the use of the log is enabled or not.

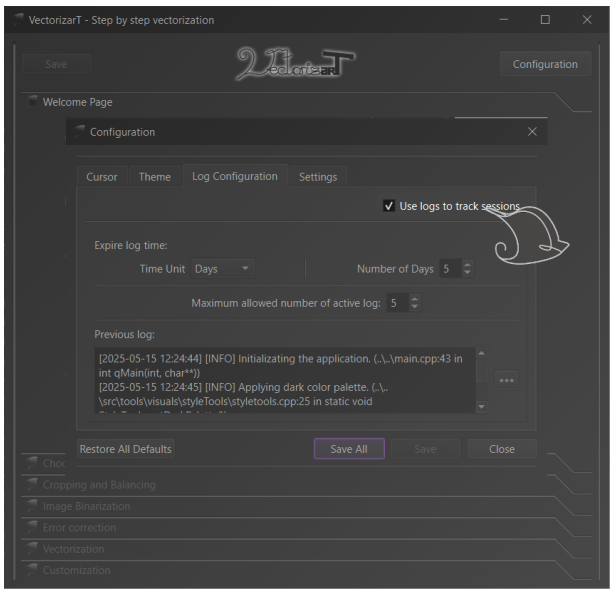

Figura A.62: Application configuration. Log use.

The next configuration refer to the expiration time for log entries and the maximum number of logs allowed.

The expiration time can be configured using different time units.

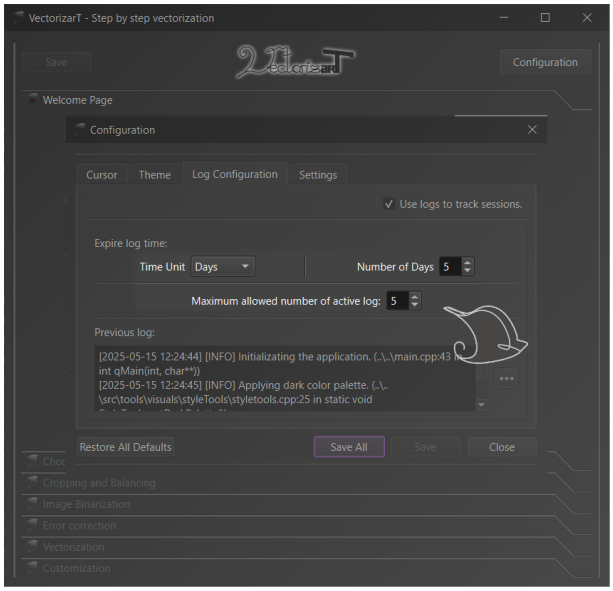

Figura A.63: Application configuration. Log expiration.

Additionally, the previous generated log is shown, along with a button that allows the user to change the folder where the logs will be stored.

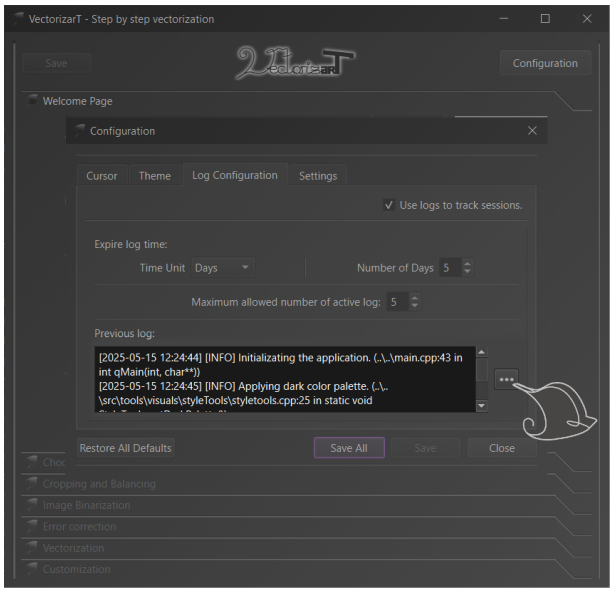

Figura A.64: Application configuration. Log path.

4. Finally, the settings tab. In this tab, users can export and import settings from other users (or from another device).

In both cases, an auxiliary window will open allowing users to choose where to save the file or which file to import, respectively.

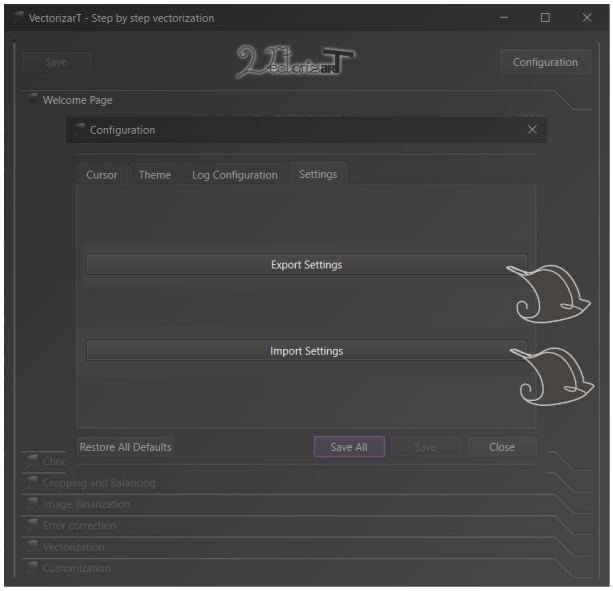

Figura A.65: Application configuration. Settings.

To make changes, use the buttons located at the bottom of the window.

The “Restore all defaults” button will close the window and reset all data to their default values.

The “Save all” button will sequentially save all configuration tabs and close the window.

The “Save” button will save the current tab. It will only be active when there is a change to save.

The “Cancel” button will close the window, cancelling all unsaved changes.

Figura A.66: Application configuration. Window buttons.

Índice de términos

Capas en dibujo digital, 6

Diagrama de Gantt, 29

Estimación por Puntos de Caso de Uso, 25

Gráficos/imágenes rasterizados, 18

Metodología Kanban, 13
Metodología Modelo en V, 13

Open Source, 4

Patrón de diseño MVC, 20
Programa/Código embebido, 58
Programación orientada a objetos, 21
Pruebas de Caja Blanca, 76
Pruebas de caja negra, 76
Pruebas de integración, 75
Pruebas de sistema, 76
Pruebas unitarias, 73

QDebug, 75
QGraphicsView, 71
QHash, 48
QLabel, 71
QLineEdit, 71
QProcess, 65
QSpinBox, 71
QString, 47
QToolBox, 72
QWidget, 72

Tablero y Tarjetas Kanban, 14
Tableta digitalizadora, 6
Tableta gráfica, 6

VTracer, 58

Índice de abreviaciones

API(s)	Application Programming Interface o Interfaz de Programación de Aplicaciones
CBD	Component-Based Development (Siglas inglesas del Desarrollo Basado en Componentes)
EF	Environmental Factors - Factores Ambientales
IAs	Inteligencias Artificiales
MVC	Modelo-Vista-Controlador
PF	Factor de Productividad, Productivity Factor
POO	Programación Orientada a Objetos
RF	Requisito Funcional
RnF	Requisito no Funcional
SO	Sistema Operativo
TCF	Factores de Complejidad Técnica (en inglés; Technical Complexity Factors)
TFG	Trabajo de Fin de Grado
UCP	Use Case Point (Puntos de Caso de Uso)
UI	User Interface, Interfaz de Usuario en español
UUCP	Puntos de Caso de Uso sin ajustar o Unadjusted Use Case Points

Bibliografía

[1] Equipo Art Rocket. *Las mejores tabletas gráficas de 2024 | Art Rocket*. Visitada el: 07-09-2024. 2024. URL: https://www.clipstudio.net/aprende-a-dibujar/archives/160576.

[2] Lenovo Website Developers. *Portátiles Lenovo Yoga 2-en-1 | Portátiles ultrafinos elegantes y de primera calidad | Lenovo*. Visitada el: 08-09-2024. 2024-08-16 22:55:34. URL: https://www.lenovo.com/es/es/c/laptops/yoga/yoga-2-in-1-series/.

[3] Apple Website Developers. *iPad - Apple (ES)*. Visitada el: 08-09-2024. 2024. URL: https://www.apple.com/es/ipad/.

[4] Adobe Team. *Adobe: Creative, marketing and document management solutions*. Visitada el: 08-09-2024; Price URL: https://www.adobe.com/creativecloud/plans.html. 2024. URL: https://www.adobe.com/.

[5] Inkscape Website Developers. *Inkscape - Draw Freely. | Inkscape*. Visitada el: 08-09-2024. 2024-10-14 15:09:06. URL: https://inkscape.org/.

[6] Servicio de Informática Profesional S.A. (SIPSA). *¿Qué es el V-Model?* Visitada el: 19-09-2024. 29 Noviembre 2023. URL: https://es.linkedin.com/pulse/quÃ¯-es-v-model-servicios-de-informatica-profesion-qotgf.

[7] Usuarios StackOverflow. *Qt Programming: More productive in Python or C++*. Visitada el: 26-09-2024. 29/06/2010. URL: https://stackoverflow.com/questions/3139414/qt-programming-more-productive-in-python-or-c.

[8] Usuarios Qt Forum. *C++ vs python in using Qt*. Visitada el: 26-09-2024. 2-11-2020. URL: https://forum.qt.io/topic/120487/c-vs-python-in-using-qt.

[9] MDN contributors. *MVC*. Visitada el: 10-10-2024. Dec 20, 2023. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC.

[10] Desconocido. *Puntos de Casos de Uso | Metodologías para el desarrollo de software*. Visitada el: 27-10-2024. 7 de Noviembre 2020. URL: https://fundamentosdetiutsjr.blogspot.com/2020/11/puntos-de-casos-de-uso.html.

[11] Carla Giani. *Diagrama de Gantt | Enciclopedia Concepto*. Visitada el: 01-11-2024. 24 de Octubre de 2024. URL: https://fundamentosdetiutsjr.blogspot.com/2020/11/puntos-de-casos-de-uso.html.

[12] . *Sueldos | Indeed.com*. Visitada el: 15-11-2024. Actualizada cada día. URL: https://es.indeed.com/career/salaries.

[13] Sanford Pun y Chris Tsang. *VTracer | Vision Cortex*. Visitada el: 10-12-2024. 1-11-2020. URL: https://www.visioncortex.org/vtracer-docs.

[14] Python Software Foundation. *Initialization, Finalization, and Threads*. Visitada el: 10-12-2024. Actualizada cada nueva versión. URL: https://docs.python.org/3/c-api/init.html.

[15] Qt Team. *How to setup Qt and openCV on Windows - Qt Wiki*. Visitada el: 20-05-2023. 24 January 2020, at 20:20. URL: https://wiki.qt.io/How_to_setup_Qt_and_openCV_on_Windows.

[16] Qt Group. *Qt Documentation*. Visitada durante todo el desarrollo, ultima vez: 01-05-2025. 24 January 2020, at 20:20. URL: https://doc.qt.io.