UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

"Programa basado en Python para integrar la gestión de documentos y procesos de trabajo en una empresa"

TRABAJO FINAL DE GRADO

Autor/a:
Antonio Miguel Mejías Velló

Tutor/a:
Dr. Alfonso Martínez García

GANDIA, 2019

Debo agradecer especialmente a Henar Langa Chordà

por su sacrificio y solidaridad incondicionales

para permitir que siga cumpliendo mis retos.

Y además por su dedicación y el compromiso demostrado

día a día a mi tutor Dr. Alfonso Martínez García y a mi

compañero Manuel Jorge Onievas que

han hecho de este programa una realidad.

RESUMEN

Este Trabajo de Fin de Grado (TFG) propone el desarrollo de un programa para cubrir la necesidad de digitalización y automatización de la gestión de los procedimientos y el trabajo que se llevan a cabo en una empresa concreta.

Para ello, se han recabado las necesidades y requisitos para diseñar dicho programa a propuesta de los trabajadores de la oficina. Con ello se ha determinado la estructura que seguirá el programa y el aspecto escogido.

Posteriormente, haciendo uso de la distribución Anaconda, se han implementado los diferentes módulos que componen el *software* utilizando lenguaje Python 3. Ya que en un futuro se pretende seguir con su desarrollo para permitir la integración de otros módulos, la escalabilidad será uno de los puntos para tener en cuenta.

El programa está compuesto por un entorno gráfico agradable al usuario basado en la biblioteca PyQt5, utilizando para su desarrollo la herramienta QtDesigner. El almacenamiento y la gestión de la aplicación se realiza mediante bases de datos relacionales integradas SQLite3, mientras un entorno gráfico claro y homogéneo permite la interacción con los usuarios.

Palabras clave: Python, PyQt5, sistema de gestión documental, programación, desarrollo de software.

RESUM

Aquest Treball de Fi de Grau (TFG) proposa el desenvolupament d'un programa per a cobrir les necessitats de digitalització y automatització de la gestió dels procediments i el treball que es duen a terme en una empresa concreta.

Per açò, s'han sol·licitat les necessitats i requisits per a dissenyar aquest programa a proposta dels treballadors de l'oficina. Amb aquest procés s'ha determinat l'estructura que seguirà el programa y l'aspecte escollit.

Posteriorment fent us de la distribució Anaconda s'han implementat els diferents mòduls que composen el software utilitza llenguatge Python 3. Ja que en un futur es pretén seguir amb el seu desenvolupament per a permetre la integració d'altres mòduls, la escalabilitat serà un del punts clau a tenir en compte.

El programa està composat per un entorn gràfic agradable a l'usuari basat en la biblioteca PyQt5 utilitzant per al seu desenvolupament la ferramenta QtDesigner. L'emmagatzemament y la gestió de la aplicació es realitza mitjançant bases de dades relacionals integrades SQLite3, mentre un entorn gràfic clar i homogeni permet la interacció amb el usuaris.

Paraules clau: Python, PyQt5, sistema de gestió documental, programació, desenvolupament de software.

ABSTRACT

This Final Degree Project (TFG) proposes the development of software to cover the need for digitalization and automation of the procedures and work management carried out in a specific company.

To do this, the needs and requirements to design this program at the proposal of the office workers have been met. This has determined the structure that will follow the program and the aspect chosen.

Subsequently, using the Anaconda distribution, the different modules that make up the software have been implemented using Python 3 language. Since in the future it is intended to continue with its development to allow the integration of other modules, scalability will be one of the points to have in bill.

The software consists of a user-friendly graphic environment based on the PyQt5 library using the QtDesigner tool for its development. Software storage and management is done through SQLite3 integrated relational databases, while a clear and homogeneous graphical environment allows interaction with users.

Keywords: Python, PyQt5, Document Management System, programming, software development.

ÍNDICE

ÍNDICE DE FIGURAS

Capítulo 1.
INTRODUCCIÓN

"La historia se repite. Es uno de los errores de la historia".

– Charles Darwin

En este Capítulo se presenta la motivación inicial de este trabajo, así como los objetivos que bajo la premisa de mejorar la productividad se tratarán de alcanzar. Asimismo, se presenta la estructura que ha seguido este trabajo con un breve resumen de lo tratado en cada capítulo.

1.1 MOTIVACIÓN

Los departamentos de gestión de las empresas tienen entre sus objetivos la mejora de los procesos que se desarrollan en éstas ya sean procesos de producción como de gestión. Como uno de los objetivos de cualquier empresa suele encontrarse el aumento de la productividad y por tanto es una de las bases sobre las que deben trabajar los departamentos de gestión. Cualquier mejora en el control y gestión de sus procesos de trabajo, de gestión o productivos redundan pues en un aumento de la productividad.

Muchos departamentos de gestión se encargan además de mantener certificaciones oficiales, como por ejemplo medioambientales o de calidad, que requieren asegurar la correcta gestión de la documentación asociada ya que se otorgan después de exhaustivos procesos de auditoría. Para superar dichas auditorías se requiere de personal con una experiencia dilatada en el puesto y en la empresa porque deben conocer en profundidad todos los departamentos de la empresa, los procedimientos y sus múltiples interacciones. No obstante, hoy en día la permanencia en un mismo puesto no suele permitir al personal responsable de un departamento concreto adquirir todos estos conocimientos de la organización para asegurar el cumplimiento de todos los requisitos exigidos por estas auditorías.

Una opción para facilitar la gestión tanto de la documentación, autorizaciones oficiales, requerimientos normativos, así como de los procedimientos internos de una empresa es integrar las múltiples funciones y tareas mediante la digitalización y manejo de éstos a través de un programa informático.

Por todo lo anterior se realiza el presente Trabajo Final de Grado (TFG) con objeto de diseñar, desarrollar e implementar un programa con los requisitos que se expondrán.

1.2 OBJETIVOS

El objetivo general de este TFG consiste en el diseño, desarrollo e implementación de un programa basado en Python para la gestión de documentos y procedimientos de una empresa.

Este objetivo se puede descomponer en los siguientes objetivos específicos:

- Analizar las necesidades y definir los requisitos que deberá cumplir el *software*.
- Desarrollar las bases de datos necesarias para su integración en el programa.

- Desarrollar una interfaz gráfica de usuario intuitiva y funcional que permita su uso con conocimientos básicos
- Implementar e integrar las funciones necesarias para el programa cumpliendo con los requisitos, tareas y funciones exigidas.

1.3 ESTRUCTURA

La memoria de este trabajo sigue la estructura siguiente:

- **Capítulo 1.**

  Introducción a la estructura, motivación y objetivos que persigue el TFG.

- **Capítulo 2.**

  En este Capítulo se detallan los conceptos relacionados con los entornos a digitalizar de la empresa. Se expondrán además características del entorno empresarial y normativa que afectará a la hora de determinar algunos requisitos. Junto a lo demandado por los usuarios, se utilizarán para diseñar e implementar el sistema.

- **Capítulo 3.**

  En él se presentan los diseños que se siguen para la implementación tanto de la parte gráfica como de algunos módulos y partes importantes como por ejemplo como serán las interacciones entre la parte gráfica y las bases de datos utilizadas.

- **Capítulo 4.**

  En este Capítulo se indican de manera breve las herramientas utilizadas para la implementación del *software*. Además, se exponen los desarrollos y objetos más importantes utilizados en el *software*.

- **Capítulo 5.**

  Se presentan las conclusiones sobre los objetivos iniciales conseguidos y las mejoras que se proponen para crear nuevas funcionalidades.

Capítulo 2.
MARCO DE TRABAJO

Cuando todo parezca ir en tu contra, recuerda que el avión despega contra el viento.

– Henry Ford

Para comprender el punto de partida y la evolución del desarrollo de este TFG se expondrán primeramente conceptos relacionados con el punto de partida para comprender la necesidad de la digitalización, comprender la empresa y otros factores que determinarán las características del desarrollo de la herramienta.

En cada apartado se aporta una breve explicación de los conceptos que se han considerado más importantes para la contextualización del entorno, aplicado a continuación al cliente/usuario de este Trabajo.

Una de las partes más importante para entender el objeto de este *software* en concreto es la de Sistema de Gestión de los que se detallarán los tipos Ambiental y de Calidad.

2.1 INTRODUCCIÓN

El entorno de trabajo es una parte fundamental para determinar los requisitos del usuario, en este caso una empresa. Uno de los departamentos está dedicado a la gestión de un Sistema de Gestión de Calidad (en adelante SGC) y otro de Gestión Ambiental (en adelante SGA) además de gestionar el mantenimiento de las infraestructuras del Centro. En el apartado 2.3 de este trabajo se define y se explica lo que implica un sistema de gestión, así como la normativa por la que se rigen.

Entre otros problemas acuciantes del departamento, destaca la falta del personal para realizar todas estas funciones de manera tradicional, se hace presente la necesidad de digitalizar todas las funciones posibles facilitando y permitiendo optimizar el tiempo dedicado a estas tareas. Además, con la digitalización de los diferentes procesos se permite asegurar y mejor el control que se tiene sobre estos conformando una mejora a la hora de pasar las diferentes auditorías de los sistemas de gestión.

Por otro lado, los continuos cambios en el personal encargado de la dirección y gestión del departamento representan un riesgo continuo y considerable en la pérdida del conocimiento, facilitando la aparición de errores y deficiencias sobre todo en los trámites o actividades con periodos de ejecución superiores al año. Esta herramienta por tanto supondrá también una manera sólida de establecer y conservar el conocimiento en el departamento constituyéndose en un pilar fundamental para la empresa.

La función principal de la empresa es la compra, almacenamiento, envío y reparación, tanto de materiales como maquinaria, es decir gestión logística. Está compuesta físicamente por un recinto de 80000 m^2 y cuenta con unos 37 edificios. Las dos unidades principales de la empresa se encargan del Mantenimiento y del Abastecimiento (compras de material, entrada, almacenamiento y envío).

Por otro lado, existe otro departamento más pequeño. Este se encarga de las funciones de apoyo como es el mantenimiento de la infraestructura de todo el recinto, mantenimiento de la maquinaria propia de la empresa y todo lo relacionado con instalaciones.

2.2 DIGITALIZACIÓN EN LA EMPRESA

La aplicación de la digitalización empresarial no es un concepto para nada novedoso, pero aun así a día de hoy podemos observar que incluso grandes empresas del país siguen tratando de alcanzar el nivel de digitalización y versatilidad que le exigen los usuarios [1].

Aparte de las ventajas mencionadas en la introducción, cuando una empresa se acerca a la utilización plena de sus recursos, los incrementos en la productividad [2] tienen que venir desde otras áreas como puede ser lo cambios en los procesos o en el caso que atañe al TFG, la digitalización.

En la empresa ya existe un *software*, SAGL (Sistema de Apoyo de Gestión Logística) para la gestión de las funciones logísticas que funciona junto con otros centros de trabajo. Dicho programa es una gran base de datos que recoge desde los contratos de compra de materiales, cantidad, tipo de material, descripciones, hasta todas las tareas de reparación, mantenimiento preventivo e historial de éstas sobre cierto material. Sin querer entrar más al detalle es una base de datos en continua mejora e infinitud de funciones. Pero como se ha explicado anteriormente, las funciones de mantenimiento de la infraestructura y el resto de las funciones de apoyo, funciones administrativas diversas, no cuentan con ningún *software* que permita integrar y compartir la información entre los diferentes trabajadores y agilizar las tareas.

2.2.1 Preservación de la información en la empresa

La digitalización de la empresa tiene un componente importante como es la conservación de documentación y registros tanto por su valor como por la función de certificar sistemas de gestión.

Para conseguir un *software* fiable, el desarrollo se basará en algunas recomendaciones basadas en la norma como UNE-ISO/TR 18492 IN, Conservación a largo plazo de información electrónica basada en documentos [3]. Además, se seguirán recomendaciones recogidas en la guía [4] creada por el Grupo de Trabajo del Subcomité de Gestión de documentos y archivos de ISO TC 46/SC 11/WG 7, sobre conservación de documentos electrónicos en el ámbito de la gestión documental.

Para el almacenamiento de la documentación e información y siguiendo las recomendaciones de la normativa anterior, se hará uso del estándar SQL definido por la norma ISO/IEC 19075:2016 Información tecnología - Lenguajes de base de datos-SQL [5].

2.3 SISTEMAS DE GESTIÓN

Se aprecia ya en los puntos anteriores que existe una intención más allá de la simple digitalización, de conservar la información, otorgarle cierta trazabilidad y fiabilidad al sistema.

La empresa cuenta con dos sistemas de gestión certificados, basados en las normas UNE-EN ISO 14001 y 9001 [6] [7]. Las empresas actualmente deben contar con una imagen institucional que transmita confianza y fiabilidad. Una manera interesante con la que “demostrar” fehacientemente que se cumplen determinados requisitos es certificarse mediante auditorías por organismos autónomos en una norma ISO. Se puede decir que estos sistemas son un conjunto de herramientas para integrar una determinada función en la infraestructura de gestión de la empresa.

Físicamente esto se traduce en una serie de documentos que primero establecen las bases del sistema de gestión definiendo su propia estructura, establece responsables y autoridades en la gestión, integra la normativa aplicable en la estructura de la empresa y determina procedimientos y procesos (ver Figura 2.1).

Las normas sobre sistemas de gestión establecen, por lo general, unos principios de funcionamiento que deben cumplir todos aquellos que se quieran certificar en ellas. Inicialmente se debe establecer un «alcance» que delimite la aplicación en las funciones y la estructura, ya que se corre el riesgo de no cubrir algunos aspectos importantes de la empresa o de incurrir en un costo excesivo en su aplicación en departamentos o aspectos que no sea necesario controlar. Por otro lado, se deben determinar las «partes interesadas» (afectadas o que puedan ser afectadas por el sistema) así como el «contexto» en el que opera la empresa para poder comenzar a detectar diferentes «riesgos» que puedan venir por factores internos o externos. Todos estos apartados, tienen como objetivo recoger los «requisitos del sistema y los legales» e integrarlos en la infraestructura misma de la empresa.

Como se puede observar, existen diferentes tareas derivadas de la gestión de un sistema de este tipo que posteriormente se deberán contemplar para establecer requisitos y funciones para el *software*.

Continuando con los conceptos, a la definición lógica de tareas, de responsables, de normativa, así como otros requisitos, representa una agrupación de funciones que se conoce como «proceso». Estos procesos son absolutamente necesarios para controlar el desempeño del sistema de gestión. Sobre ellos se definen «indicadores» medibles sobre las salidas o una parte del proceso para que el responsable del Sistema pueda comprobar que se desarrolla con normalidad o de manera conforme.

La definición de «conforme» significa que el proceso se está desarrollando dentro de los parámetros marcados por el propio gestor. Relacionado con el anterior, existen las «No Conformidades» para las que se deberá hallar evidencias de por qué falla un proceso, el origen de un error o desviación, responsables o departamentos afectados. El objetivo de éstas, no es tratar de buscar culpables, sino corregir el proceso y evitar la repetición del mismo error.

Prácticamente en esto se basan todos los sistemas de gestión, crear la parte documental recogiendo toda la información dispersa, compilarla y ponerla a disposición de todos los usuarios. El siguiente paso es asegurar que toda la información es transmitida, recibida y comprendida por las partes interesadas. Poner en marcha los procesos definidos y controlarlos periódicamente. Por último, se analizará con la Alta Dirección de la empresa los resultados obtenidos, mejoras posibles, acciones derivadas de las auditorías y establecer los objetivos para el año siguiente.

El objetivo de cualquier sistema de gestión, en una determinada materia, es asegurar la aplicación de toda la normativa y objetivos mediante la definición y control de los procesos, reduciendo así los errores y evitando su repetición, así como riesgos para la empresa. Todo esto redunda en la reducción de costes a medio y largo plazo además de limitar sorpresas desagradables en forma de sanciones administrativas o costosos errores.

Figura 2.1 Contenido documental típico de un Sistema de Gestión

2.3.1 Sistema de Gestión Ambiental

Un Sistema de Gestión Ambiental (en adelante SGA), que siga la norma UNE-EN ISO 14001 aplica las conceptos y pasos mencionados en el punto anterior con el objetivo final de mejorar y reducir la afectación de la empresa, tanto sus actividades como productos consumidos y producidos, al medioambiente. En la sociedad actual supone un valor añadido ya que refuerza la imagen de responsabilidad ambiental de manera positiva frente a otras empresas y es sinónimo de confianza. Todas las grandes empresas, como por ejemplo Volkswagen "el compromiso con la sostenibilidad, el medio ambiente y la sociedad constituyen los valores principales de la responsabilidad social corporativa de Volkswagen-Audi España" refuerzan y focalizan el medioambiente como pilar fundamental de su imagen corporativa.

Tras la determinación del alcance y las partes interesadas que se debe realizar en un sistema de gestión, en un SGA deberán identificarse además los «aspectos ambientales». Este peculiar concepto es «todo elemento de las actividades, productos o servicios de una organización que puede interactuar con el medio ambiente» (según UNE-EN ISO 14001). Es necesario recordar que cada concepto se relaciona con una o varias tareas que se deberán realizar en el sistema de gestión y que por tanto en apartados posteriores se integrará como un requisito del *software*.

Una vez determinados cuales son los aspectos ambientales de interés, se evalúan cuantitativa o cualitativamente los efectos beneficiosos o negativos que tienen para establecer cuáles y cómo tratarlos. Esta tarea, la «evaluación de aspectos», será la que oriente los «objetivos ambientales» en la organización.

Debe estar previsto también un proceso para actuar sobre efectos no controlados de estos aspectos ambientales como «funcionamientos anormales o situaciones de emergencia».

Otra de las características de un SGA, es la presencia de procesos para el «control operacional» de los principales aspectos ambientales como por ejemplo control operacional de: emisiones atmosféricas, aguas residuales, consumos de materias primas, consumos energéticos, emisiones de Componentes Orgánicos Volátiles (disolventes), trabajos con gases fluorados, tratamientos silvícolas…

Como se puede apreciar en el ejemplo de la Figura 2.2 los procesos generan documentos que deben guardarse y registrarse correctamente para demostrar su realización y resultados. Existen, además, muchos otros documentos que se denominan «Registros» derivados de los procedimientos de control operacional y de la verificación del seguimiento de los diferentes procesos. Éstos son un punto fundamental para sacar las conclusiones y verificar cómo progresa el sistema de gestión.

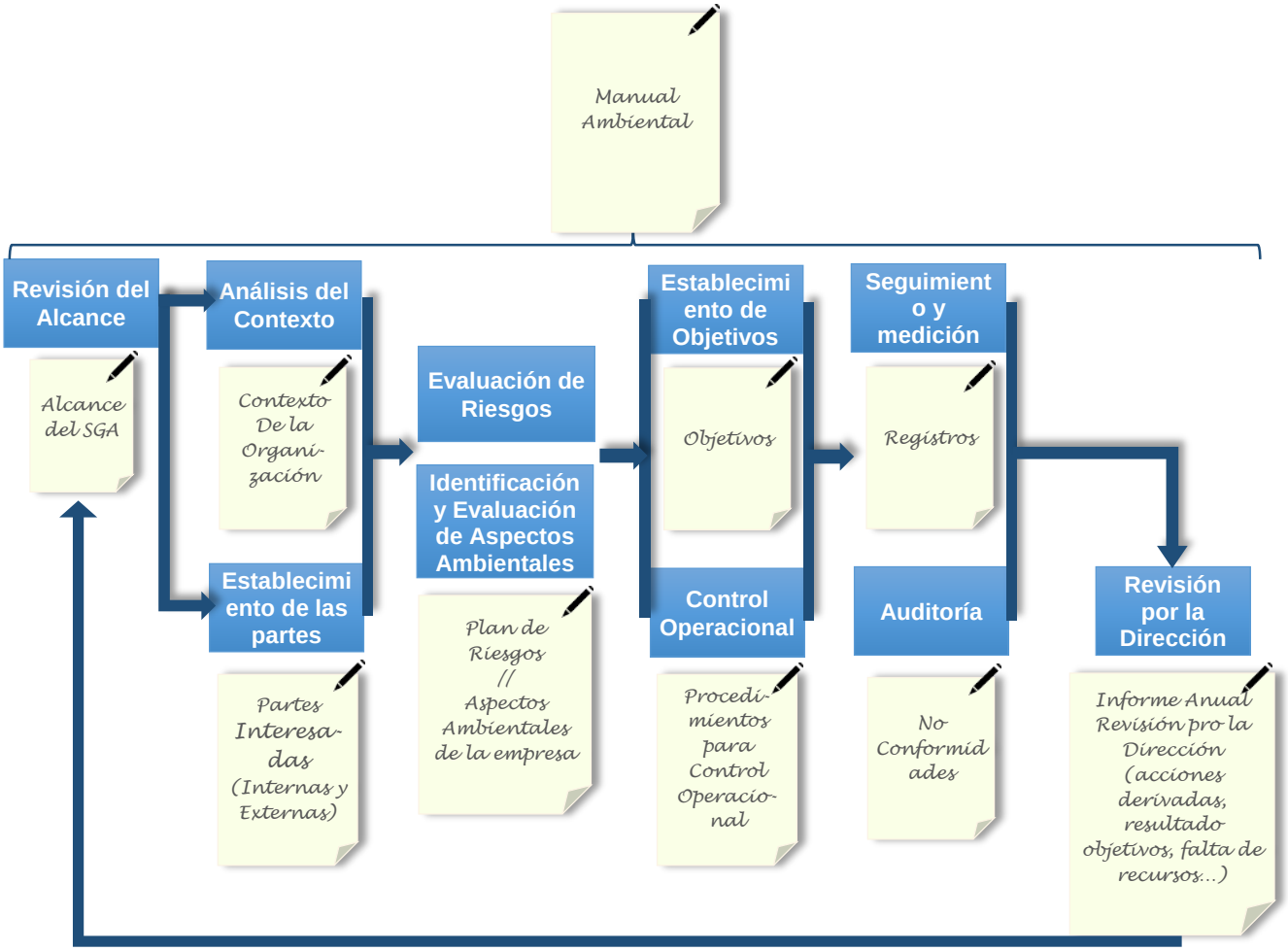


Figura 2.2 Ejemplo de un ciclo de un SGA anual.

2.3.2 Sistema de Gestión de la Calidad

Siguiendo con lo indicado para un sistema de gestión y un SGA, un SGC tiene muchas de las partes en común con los SGA como el alcance, el contexto, los riesgos, el control operacional, las auditorías, los procesos, los objetivos, la medición y seguimiento. La principal diferencia, claro está, es el objetivo que persigue este sistema. Un SGC persigue la mejora de la Calidad percibida por los clientes, mejorando procesos, evitando errores y cumpliendo con los requisitos que solicita el cliente de un producto o servicio prestado.

Uno de los esquemas más utilizados para resumir el funcionamiento de un SGC es el ciclo PHVA (Planear, Hacer, Verificar y Actuar) como el del ejemplo de la Figura 2.3.

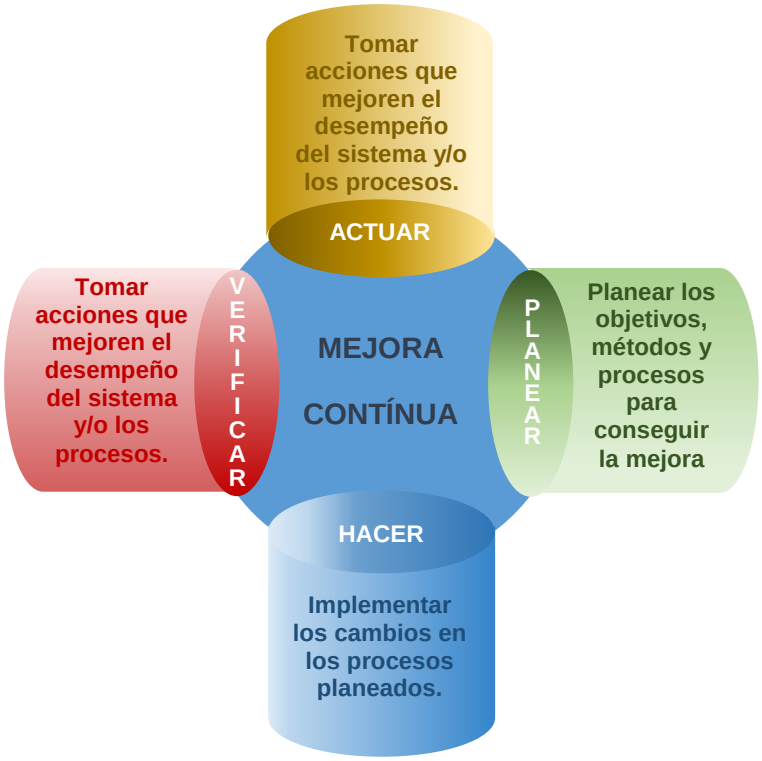


Figura 2.3 Ciclo de mejora continua de los sistemas de gestión.

Como se observa, se trata de implementar una mejora continua planificando objetivos graduales. Se controlarán los cambios que se realizan y comprobarán en cada iteración, ya sean aplicados a un proceso o al sistema. Por último se analizarán qué efectos han tenido volviendo de nuevo comenzar el ciclo.

Dentro de lo que sería la planificación se necesita desarrollar un entorno enfocado a procesos y describir todos y cada uno de los ítems que determina la UNE-EN ISO 9001. Se recogerá en una «ficha de procesos» que incluya: responsables, fuentes de entrada, salidas, procedimientos que describan las actividades, indicadores de eficiencia y eficacia, entre otros. La representación gráfica de esto sería, el «mapa de procesos» (Figura 2.4) que engloba las actividades de una empresa como la indicada. Se puede ver el mapa de procesos de la empresa en la figura siguiente:

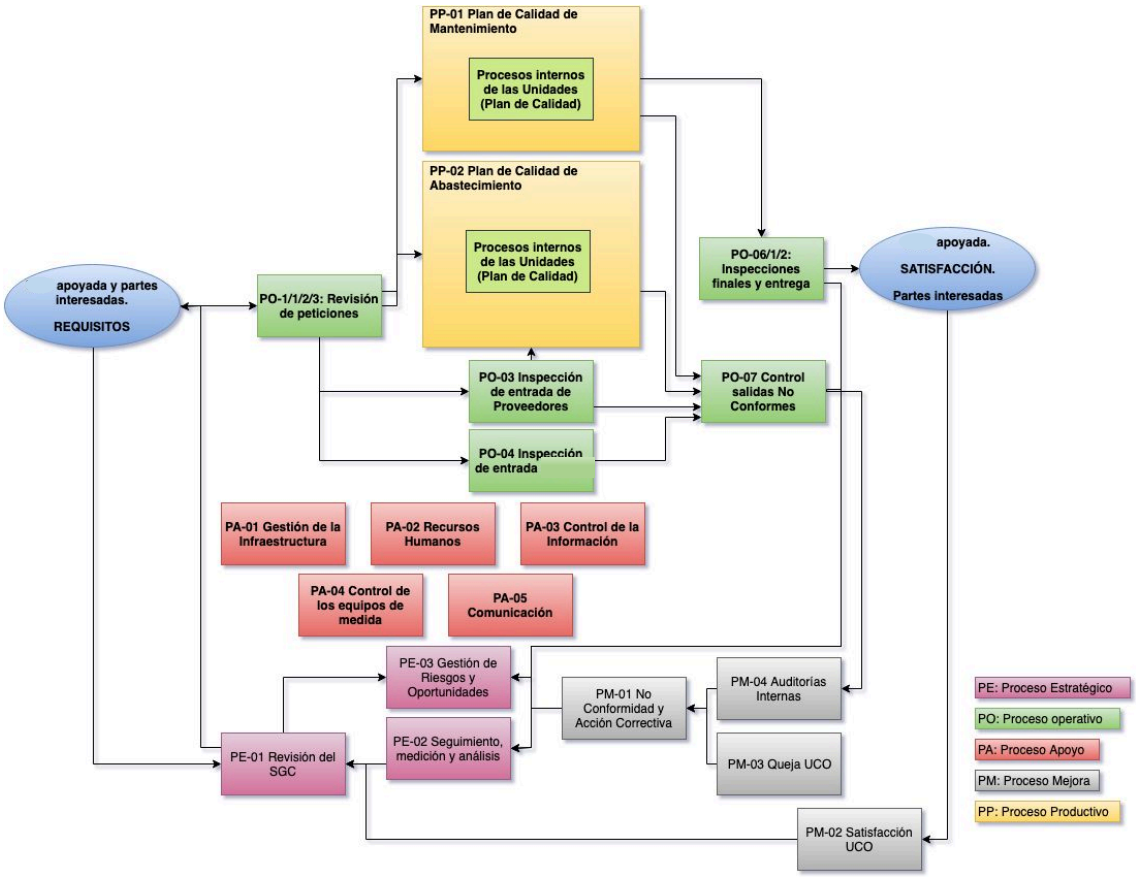


Figura 2.4 Mapa de procesos de la empresa

Con lo anterior quedaría contenido la parte de Planificación del ciclo PHVA. Por otro lado, en cada uno de los procesos hay que dejar constancia de los cambios realizados o controlarlos. Por lo que en cada registro que se genere o en los planes de cambios que se realicen es importante establecer plazos para comprobar la realización de la actividad; y además verificar la eficacia de las acciones. Se observa que la ejecución de las acciones (Hacer) debe ir acompañada de la Verificación necesaria para cumplir con los establecido para un SGC.

Por último, se analizará si se necesitan planear nuevas acciones para mejorar el sistema o subsanar algunas de las anteriores (Actuar).

En este punto, se debe indicar que puede que para algunos de los documentos mencionados se optará por almacenarlos tal cual, o guardar los datos que contenga en una base de datos.

2.3.3 Sistema Integrado de Gestión

La normativa sobre sistemas de gestión se ha ido desarrollando y actualizando a lo largo de los años hasta crear una estructura y funcionamientos que permiten trabajar en conjunto a los diferentes Sistemas. Esto permite aunar esfuerzos y procedimientos comunes que reducen el trabajo a realizar (ver Figura 2.5). Aunque no se haya incluido como objetivo en este TFG debido a la estructura de la empresa se tratará en el punto 5.2 MEJORAS.

Figura 2.5 Diagrama de un Sistema Integrado de Gestión.

2.4 PROCESO DE DESARROLLO

Con el objeto de desarrollar un *software* de la mayor calidad posible con los recursos disponibles se ha seguido un proceso de desarrollo iterativo y creciente [8]. Es un modelo basado en etapas repetitivas que en este caso permite un desarrollo más ágil y adaptado a las necesidades del usuario [9]. Bajo la premisa de que en este enfoque el usuario en concreto no conoce, en un primer momento, los requisitos finales de lo que buscan para satisfacer sus necesidades. Dicho de otra forma en el enfoque incremental, no existe una especificación completa del sistema hasta que el incremento final se especifica [10].

Por una parte, el desarrollo iterativo permite consultar al usuario en cada iteración, si los requisitos de la parte del módulo en desarrollo se adaptan a lo deseado. Con cada iteración se considera que lo desarrollado hasta el momento se encuentra en “borrador” y se permiten ciertas adaptaciones antes de la finalización del módulo. Esto permite además tener en cuenta los cambios en los procesos que se producen durante el desarrollo.

Por otro lado, el desarrollo incremental permite al programador aprovechar lo que se va aprendiendo a lo largo del proyecto. Lo aprendido de las versiones tempranas, incrementales y entregables del sistema [11] permite desarrollar módulos de manera casi independiente. En cada entrega de un módulo, se plantean modificaciones del siguiente módulo, lo que da versatilidad para adaptarse mejor a los cambios que puedan surgir. Facilitando la corrección de errores o introducción de cambios en las diferentes fases tanto por parte del cliente como del propio desarrollador.

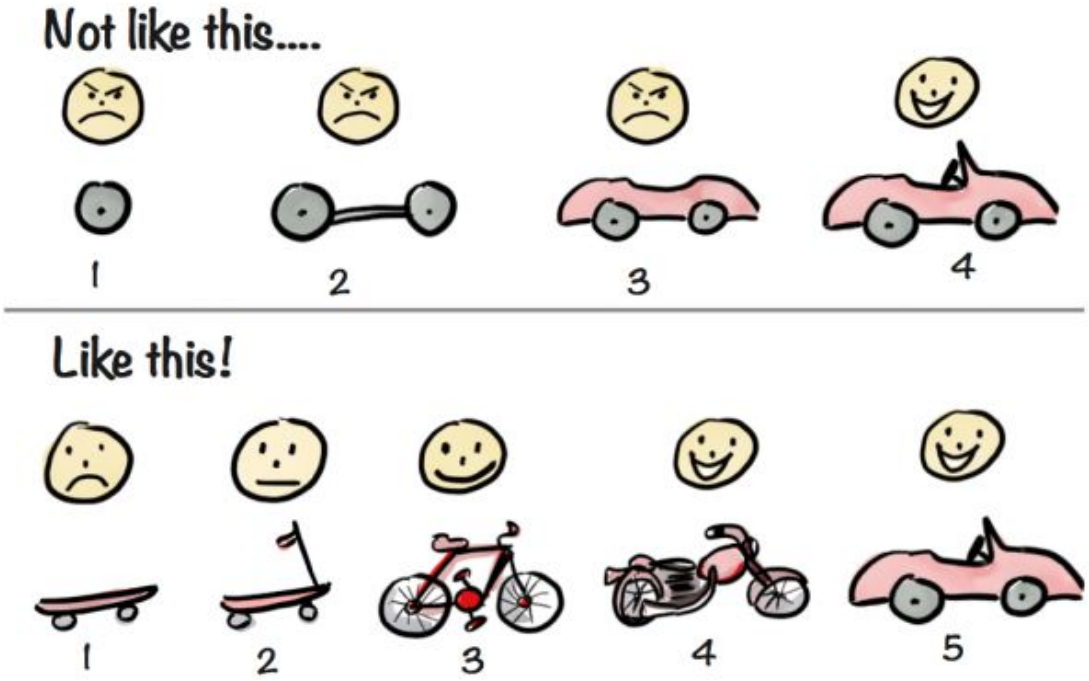


Figura 2.6 «Iterative & incremental development, horizontal vs vertical» by Henrik Kniberg [12]

El proceso iterativo e incremental (ver Figura 2.6) se creó en respuesta a las debilidades del modelo en cascada tradicional. Se puede entender por lo tanto como un desarrollo parcial dentro cada módulo entregable (iterativo) y que se adapta por iteraciones a lo requerido añadiendo funciones a través de diferentes módulos entregables (incremental) [13].

El proceso de desarrollo consiste de:

1. Etapa de inicialización.
2. Etapa de iteración.
3. Lista de control de proyecto.

Se han seguido en su mayor parte, características del proceso de desarrollo de este *software* (ver Figura 2.7) definidas para procedimientos iterativos e incrementales, definidos en el PMBOOK [14]. Las directrices seguidas han sido las siguientes:

- Los requisitos se han establecido por el cliente tras reiteradas entregas. Se realizó una reunión con el personal afectado definiendo los módulos, apariencia y funcionalidades.
- Las entregas se han llevado a cabo periódicamente, según la incorporación de nuevas funcionalidades.
- Los cambios implementados se han realizado tras la entrega periódica según los nuevos requisitos indicados o *debugging*. Normalmente los tiempos de verificación con los usuarios de dichos cambios han sido de dos semanas.
- La interacción con el cliente ha sido periódica, al tiempo de la finalización de nuevas funcionalidades.
- No se ha incorporado un control de riesgos ni costes asociados.

Predictivos	Iterativos	Incrementales	Ágiles
Los requisitos son definidos por adelantado antes de que comience el desarrollo	Los requisitos pueden ser elaborados a intervalos periódicos durante la entrega		Los requisitos se elaboran con frecuencia durante la entrega
Entregar planes para el eventual entregable. Posteriormente, entregar solo un único producto final al final de la línea de tiempo del proyecto	La entrega puede ser dividida en subconjuntos del producto global		La entrega ocurre frecuentemente con subconjuntos del producto global valorados por el cliente
El cambio es restringido tanto como sea posible	El cambio es incorporado a intervalos periódicos		El cambio es incorporado en tiempo real durante la entrega
Los interesados clave son involucrados en hitos específicos	Los interesados clave son involucrados periódicamente		Los interesados clave son involucrados continuamente
El riesgo y los costos son controlados mediante una planificación detallada de las consideraciones que mayormente se conocen	El riesgo y los costos son controlados mediante la elaboración progresiva de los planes con nueva información		El riesgo y los costos son controlados a medida que surgen los requisitos y limitaciones

Figura 2.7 Características de los procesos Iterativos e incrementales. (PMBOOK 6 Ed.) [14]

2.5 REQUISITOS

Aunque no se exista una especificación completa de los requisitos de *software* o establecimiento de requisitos del *software* (ERS) como tal, complicado para el tipo de desarrollo (unipersonal), se han seguido algunas pautas de las recomendaciones definidas por el estándar IEEE 830 [15]. Los requisitos cumplen las siguientes características:

- *Consistente*: coherencia con los requerimientos y con los de los documentos oficiales establecidos.
- *Inequívoca*: se establecen con el mayor detalle posible para evitar implementaciones erróneas.
- *Correcta*: el *software* cumple con los requisitos establecidos.
- *Verificable:* se pueden inspeccionar los diferentes requisitos con el propio código y versiones del *software* desarrolladas.
- *Completa:* se definen por completo los requisitos incluyendo las respuestas a las diferentes entradas de datos.
- *Trazable:* es difícil cumplir la trazabilidad porque no se define la línea entre usuario y desarrollador además de no considerarse necesario porque se considera excesivo para el alcance del proyecto.
- *Priorizable:* no se ha establecido por el alcance del proyecto.
- *Modificable:* al tener un alcance muy limitado son fácilmente modificables.

Los tipos de requisitos que se definirán en este documento serán los siguientes:

- Requisitos de usuarios: necesidades que se expresarán por escrito o verbalmente por los usuarios.
- Requisitos funcionales: las funciones que deberá tener el *software* al acabar.

Aunque no formen parte propiamente de la ERS también se definirán algunos formatos de entrada de datos y listados almacenados.

2.5.1 Acceso con credenciales

Se requiere por los responsables de la empresa que cada usuario debe tener un acceso propio y diferenciado del resto, contando con usuario y contraseña que será requerido por el *software* en el momento de su inicio. Además, como función adicional

procederá a desconectarse, o al menos solicitar las credenciales nuevamente tras diez minutos de inactividad.

2.5.2 Partes

Se solicita tener tres módulos diferenciados, uno para el Sistema de Gestión Ambiental, otro para el Sistema de Gestión de Calidad y otro para el Sistema de Protección Contra incendios. Las funciones requeridas serán determinadas para cada una en siguientes apartados.

2.5.3 Funciones generales

Gestión y almacenamiento de datos

La aplicación deberá almacenar datos en texto plano, documentos PDF e imágenes en BBDD.

Gestión y almacenamiento de documentación

Se deberá permitir la modificación, actualización, eliminación de la información almacenada en las BBDD.

Extracción de datos

El programa contará con la posibilidad de extraer los datos o documentos para disponer de ellos de una manera directa (exportación de archivos) o indirecta (acceso a las BBDD).

2.5.4 Módulo SGA

Apartados mínimos que deberá tener: alcance, procesos, partes interesadas, análisis del contexto, definición de indicadores, control de indicadores, no conformidades, análisis de riesgos, requisitos legales, control de documentos, tareas, objetivos, auditoría, formación, competencia, residuos, consumos, consumo de COV, emisiones atmosféricas, aguas residuales, situaciones de emergencia, carteles de información y revisión por la dirección.

2.5.5 Módulo SGC

Apartados mínimos que deberá tener: alcance, procesos, partes interesadas, análisis del contexto, definición de indicadores, control de indicadores, no conformidades, análisis de riesgos, requisitos legales, control de documentos, tareas, objetivos, inspecciones de entrada, auditoría, formación, competencia, riesgos y revisión por la dirección.

2.5.6 Módulo del sistema de protección contra incendio

Este módulo servirá como prototipo para comprobar si se pueden integrar las tareas de control y gestión del mantenimiento del resto de infraestructuras. Pretende recoger la información y documentación más importante e imprescindible que acompaña al mantenimiento del sistema contraincendios del recinto.

Los apartados mínimos que incluirá serán instalaciones y sistemas, documentación y manuales, mantenimiento preventivo, mantenimiento correctivo, control de incidencias, personal y formación, normativa y registro de comunicaciones.

Capítulo 3.
DISEÑO DEL PROGRAMA Y BASES DE DATOS

Preguntarse cuándo los ordenadores podrán pensar es como preguntarse cuándo los submarinos podrán nadar

– Edsger W. Dijkstra

En el siguiente capítulo se presentan los diseños que se han establecido para las diferentes partes del programa. De manera lógica se presentan las interacciones y funcionamiento de diferentes partes del *software*. Se muestra a grandes rasgos cómo se gestionará la interacción entre BBDD e interfaces gráficas. Se presenta además cómo se trabajará sobre las BBDD y algunos aspectos sobre seguridad y acceso al programa.

3.1 INTRODUCCIÓN

La elección del diseño se realizará con el objetivo de mantener la sencillez en el código a desarrollar, pero otorgando la funcionalidad mínima requerida. Las funciones necesarias para el cumplimiento de los requisitos se agruparán para simplificar de la mayor manera tanto el diseño gráfico como del código [16].

3.2 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO (GUI)

Se desarrollarán tres módulos, uno por cada sistema de gestión y otro para el sistema de protección contra incendios. La decisión de agrupar por módulos, SGC, SGA y PCI, permitirá en un futuro que cada usuario acceda a uno de los sistemas a los que se le conceda acceso, mejorando la integridad y la seguridad de los sistemas.

Asimismo, las diferentes funciones se agruparán por su similitud utilizando pestañas. En cada una de las pestañas se situará un pequeño entramado de menús y submenús relacionados con la función en concreto y si es adecuado a la función, presentará ciertos datos, preferentemente en forma de tablas ya sean documentos, planos, o datos en texto. Este diseño (ver Figura 3.1) permite homogeneizar el funcionamiento de los diferentes módulos a la vez que mantiene el sistema lo más sencillo posible [17].

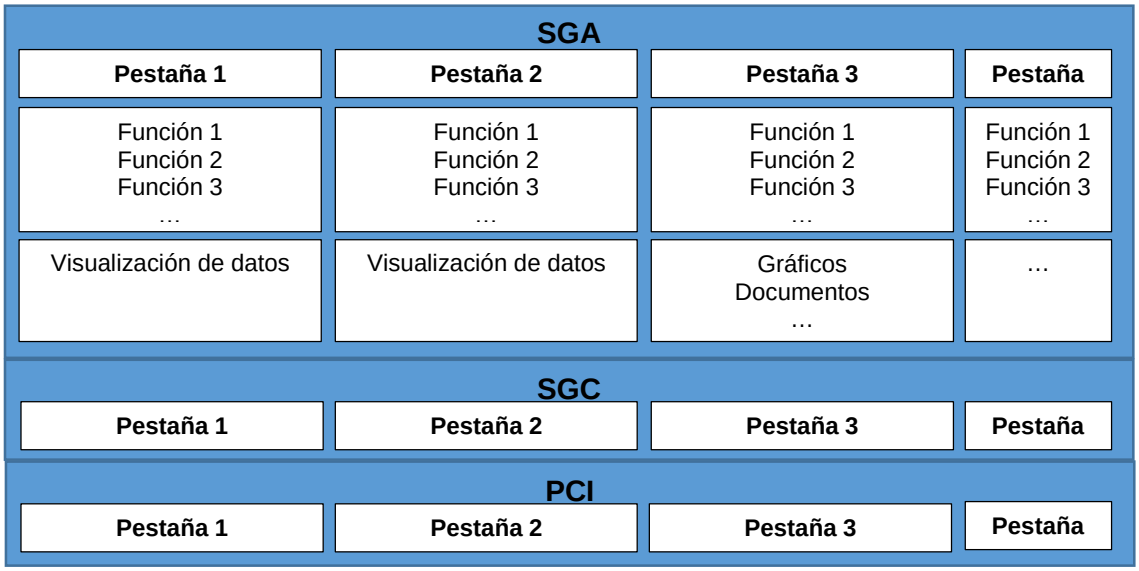


Figura 3.1 Ejemplo diseño de la estructura gráfica.

Otra ventaja de este diseño es que facilita escalar la funcionalidad del *software*, mediante la agregación de nuevas pestañas con nuevas opciones.

3.3 FUNCIONES CON ALMACENAMIENTO DE DATOS

Algunas de las funciones requieren de almacenamiento y consulta de datos para lo que se utilizarán una o varias BBDD (ver Figura 3.2). Las BBDD deberán utilizar el lenguaje SQL (ver punto 2.2.1). Las consultas y modificaciones estarán programadas dentro de cada función en concreto pudiendo actuar el usuario solamente sobre algunos de los datos a visualizar desde los controles que se le ofrezcan en cada pestaña.

Las BBDD pueden ser utilizadas por una o más pestañas según las necesidades. Desde un menú de opciones, se podrá escoger la ruta para indicar donde se encuentra cada base de datos para dar la posibilidad de modificar las ubicaciones de éstas.

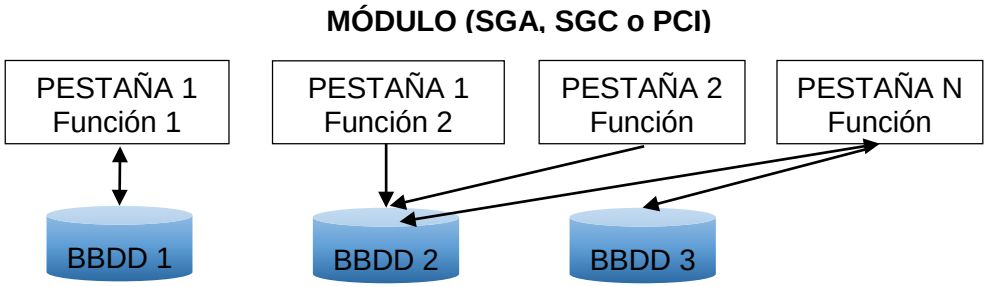


Figura 3.2 Ejemplo de estructura de trabajo para consulta/presentación de datos por pestañas

3.3.1 Bases de datos relacionales

El lenguaje SQL (*Structured Query Language*) se utiliza para acceder y operar sobre BBDD relacionales. Este tipo de BBDD deben cumplir el modelo relacional debiendo en definitiva planificar las diferentes tablas que se quieren definir, los tipos de datos que almacenar y las relaciones entre tablas que permitirán el correcto funcionamiento de éstas.

En este TFG, no se expondrá el trabajo de diseño estructural de las BBDD utilizadas por el programa, pero sí cómo se trabaja con ellas. Las bases con las que se trabaja son “Gest_residuos_v3.db”, “base_datos_pci.sqlite” y “bdauxiliar.sqlite” con SQLite.

Para ello se ha hecho uso de SQLite3 para Python, la interfaz (API) para BBDD SQLite [18] que permiten utilizar esta librería para implementar BBDD sobre disco duro. Algunas características que podrían definir SQLite es que utiliza una variante del lenguaje SQL no estandarizada y no necesita de conexión a servidor, a diferencia de MySQL, y trata la base de datos como un documento estructurado. Las diferencias con MySQL no son especialmente relevante para este programa, se puede decir incluso que, debido a su versatilidad a la hora de poder utilizar herramientas de terceros para acceder a los datos, por su posibilidad de mover las BBDD de manera sencilla y su ligereza, lo convierten en una opción mejor para esta aplicación.

Los tipos de datos que maneja esta API son los siguientes

- NULL: sin tipo de valor.
- INTEGER: valor entero con signo guardado en 8, 16, 24, 32, 48, o 64 *bits*
- REAL: valor real con coma flotante de 64 *bit*.
- TEXT: *array* de caracteres en formato UTF-8, UTF-16BE o UTF-16-LE.
- BLOB: Binary Large Objects, guarda bytes de datos en un *array*.

3.4 USUARIO Y CONTRASEÑA

La contraseña y el usuario se crearán manualmente por el administrador de la aplicación. Se realizará un cifrado irreversible con un hash cuyo resultado será almacenado en un archivo. El archivo donde se almacene la información a su vez será cifrado, esta vez reversible mediante una contraseña, de modo que la información no sea accesible de un modo sencillo (Figura 3.3). Para el cifrado se utilizará una función hash, que es una operación criptográfica que genera un código de un tamaño fijo independientemente del tamaño de la entrada. Además, el algoritmo debe cumplir unos requisitos tan exigentes que debe ser computacionalmente inviable que dos entradas diferentes generen el mismo código de salida [19].

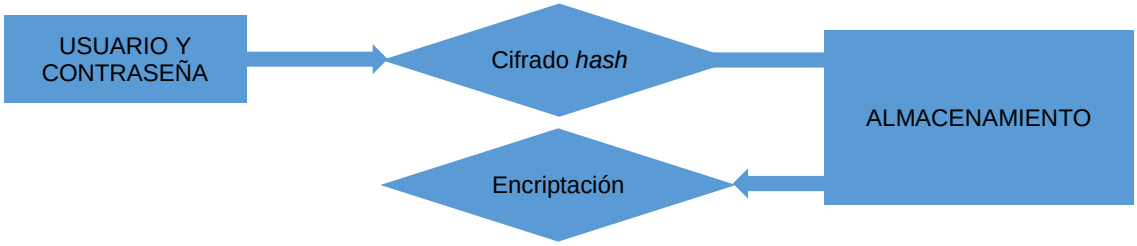


Figura 3.3 Proceso de almacenamiento de usuario y contraseña.

La comprobación para permitir el acceso al programa será, por tanto, cifrar la contraseña y usuario introducidos, descifrar el archivo y comprobar si coinciden los resultados (Figura 3.4).

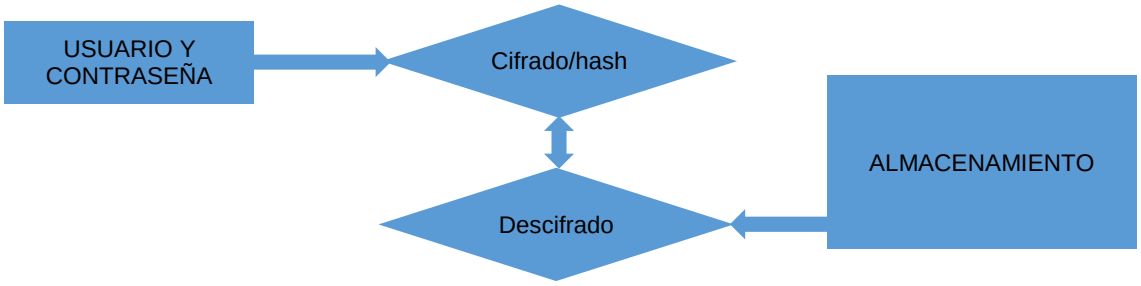


Figura 3.4 Comprobación de usuario y contraseña.

3.5 CONTROL DE VERSIONES

El Control de Versiones ha sido realizado localmente. El control de funciones y cambios han sido registrados sobre la cabecera del documento de programación. A lo largo del desarrollo de esta aplicación se tienen más de 20 versiones diferentes que han permitido recuperar en muchos casos funciones abandonadas o acometer cambios mayores. La versión actual del *software* es la 3.0.5.

Capítulo 4.
IMPLEMENTACIÓN Y VERIFICACIÓN

La presión del anti-intelectualismo ha ido abriéndose paso constantemente a través de nuestra vida política y cultural, alimentada por la falsa noción de que la democracia significa que mi ignorancia es tan válida como tu conocimiento.

–Isaac Asimov

En este capítulo se describe cómo se han llevado a cabo los diseños y con qué herramientas se ha implementado este *software*. Desde la implementación de las BBDD, pasando por la presentación de los datos y el almacenamiento de éstos, se explican las partes más significativas de la solución escogida.

4.1 INTRODUCCIÓN

Para la implementación de este programa se ha optado por la programación en Python del código y el uso de la biblioteca PyQt5 para desarrollar la interfaz gráfica de usuario. Utilizando la estructura modular descrita en el punto 3.2 se ha desarrollado un programa que permite la gestión de diferentes BBDD relacionales, la gestión usuarios, almacenamiento de documentos y las demás funciones requeridas.

4.2 LENGUAJE DE PROGRAMACIÓN

El lenguaje utilizado ha sido Python, en concreto la versión 3.7.3. Es la tercera versión de mantenimiento de Python 3.7, lanzada el 25 de marzo de 2019 que mejora algunas características y optimizaciones.

Es un lenguaje orientado a objetos e interpretado, no necesita compilar ya que se va interpretando a medida que sea necesario. Es un lenguaje muy versátil, sencillo y dinámico que permite que sea reconocido como uno de los lenguajes de iniciación más utilizados. Su veteranía (30 años) y su constante evolución hacen que cuente con una gran comunidad de usuarios [20] y por tanto mucha diversidad de API y librerías, que facilitan enormemente el desarrollo, como las que se mencionan a lo largo de este TFG.

El tipo de archivos utilizado para todos los archivos Python de este TFG ha sido PYTHON que corresponde con la extensión *.py*.

4.3 ENTORNO DE PROGRAMACIÓN Y HERRAMIENTAS

4.3.1 Manejo de entornos y gestor de paquetes Python

Para este trabajo no se ha utilizado una versión “limpia” de Python, sino una distribución de Conda que permite crear entornos virtuales personalizando las diferentes versiones de las herramientas utilizadas y proveyendo de herramientas de programación como Spyder, que se utilizará para la programación en sí.

Conda es un sistema multiplataforma para la gestión de paquetes y entornos. Tiene licencia BSD, *Open Source*. En un principio fue creada para el manejo de programas en Python, pero actualmente permite compilar y gestionar diferentes tipos de *software*. La distribución utilizada es Anaconda©.

Para la programación se instaló la versión 4.7.12 de Conda que viene junto la distribución Anaconda3 con Python 3.7 64-Bit.

4.3.2 Spyder

Es el entorno físico de desarrollo para la programación del código de este programa. Se ha utilizado la versión Spyder 3.3.4, de este IDE (*Integrated Development Environment*) que permite la programación en Python integrando edición avanzada, *debugging*, ejecución interactiva del código, corrector y asistente, así como otras funcionalidades [22].

Como se verá en el punto siguiente, para el entorno gráfico se ha utilizado la librería PyQt5, para la que Spyder (ver Figura 4.1) cuenta con librerías integradas que facilitan su funcionamiento.

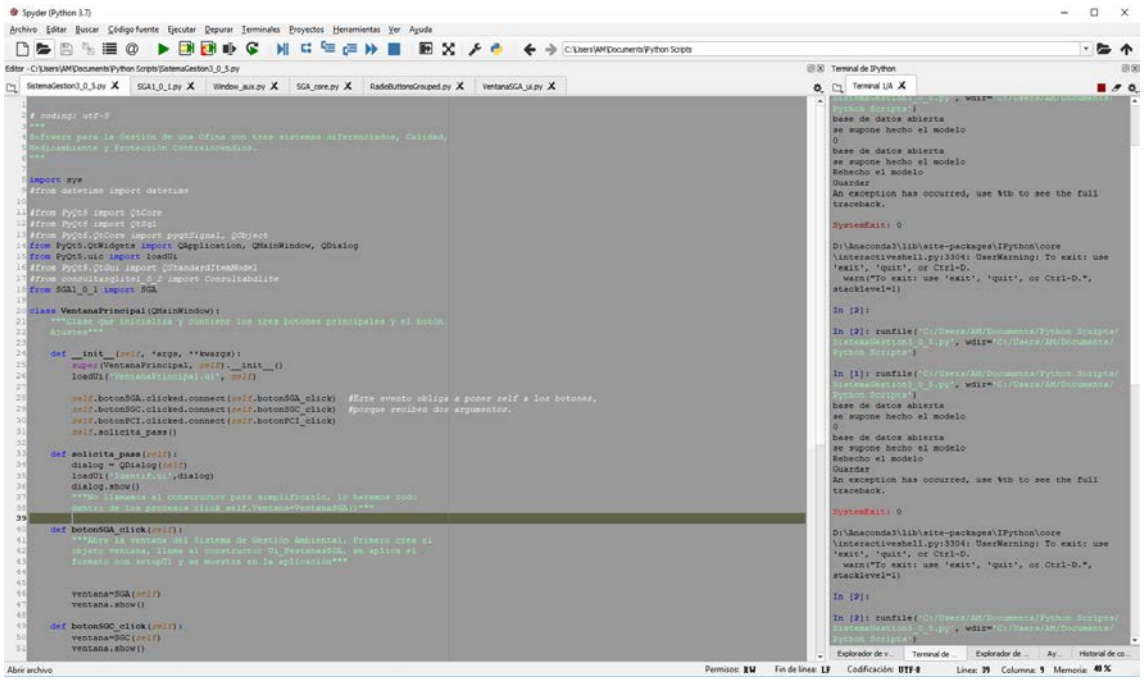


Figura 4.1 Entorno IDE Spyder utilizado para el desarrollo del código.

4.3.3 PyQt5

Es un módulo para crear aplicaciones con entorno gráfico en Python. En conjunto con su herramienta Designer que se expondrá a continuación, permite el diseño y

desarrollo de entornos complejos de una manera relativamente sencilla a otras librerías gráficas [23].

La librería Qt provee de funciones para crear entornos gráficos de usuario y funciones de apoyo. No obstante, se desarrolló en C++ y por tanto tuvo que ser adaptado para su uso en entornos como, por ejemplo, Python, a través de lo que se llama *bindings*. El *binding* para Python, desarrollado por Riverbank Computing es PyQt.

PyQt en su versión 5 provee funciones gráficas con los módulos como QtGUI, de la cual se hace uso en este trabajo y se trata de la clase base para los componentes de la interfaz gráfica de usuario. También se hace uso de QtSQL un módulo que añade funcionalidades para trabajar con BBDD SQL. Por último, QtWidgets extiende las funcionalidades de QtGUI, permitiendo utilizar una gran variedad de botones, pestañas, visores en los desarrollos.

4.3.4 QtDesigner

Es la herramienta para diseñar las interfaces gráficas de usuario. Con ella se diseña sobre un entorno gráfico las diferentes ventanas, menús y otros elementos de interfaz *GUI*(ver Figura 4.2). Arrastrando y soltando en el lugar deseado se pueden añadir los diferentes controles para posteriormente modificar o establecer el formato deseado.

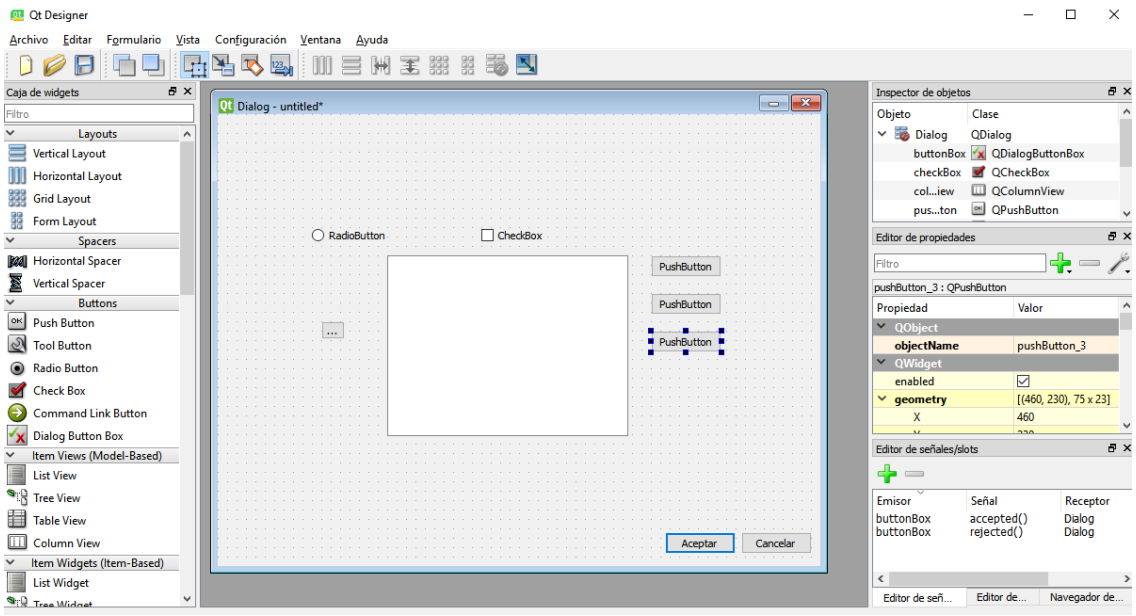


Figura 4.2 Ejemplo de ventana diseñada con la herramienta Qt Designer.

La salida finalizada del diseño se almacena como un archivo *.ui* con formato XML que contiene el diseño de los *widgets*, posiciones y formatos escogidos [23]. En el punto 4.5 IMPLEMENTACIÓN DE LOS INTERFACES GRÁFICOS, se verán los modos de utilización sobre el código bien cargando el formato con la función *loadUI()* o convirtiendo el archivo *.ui* a *.py*.

4.4 IMPLEMENTACIÓN DE FUNCIONES DE TRABAJO CON BASES DE DATOS

Para el almacenamiento de la información requerida se utilizan BBDD en SQLite. Al almacenar documentos, estas BBDD integradas, pueden llegar a tener un tamaño considerable, por lo que se crean varias BBDD diferentes.

- GestResiduosv3.sqlite (ver Figura 4.3)
- bdauxiliar.sqlite
- base_datos_pci.sqlite

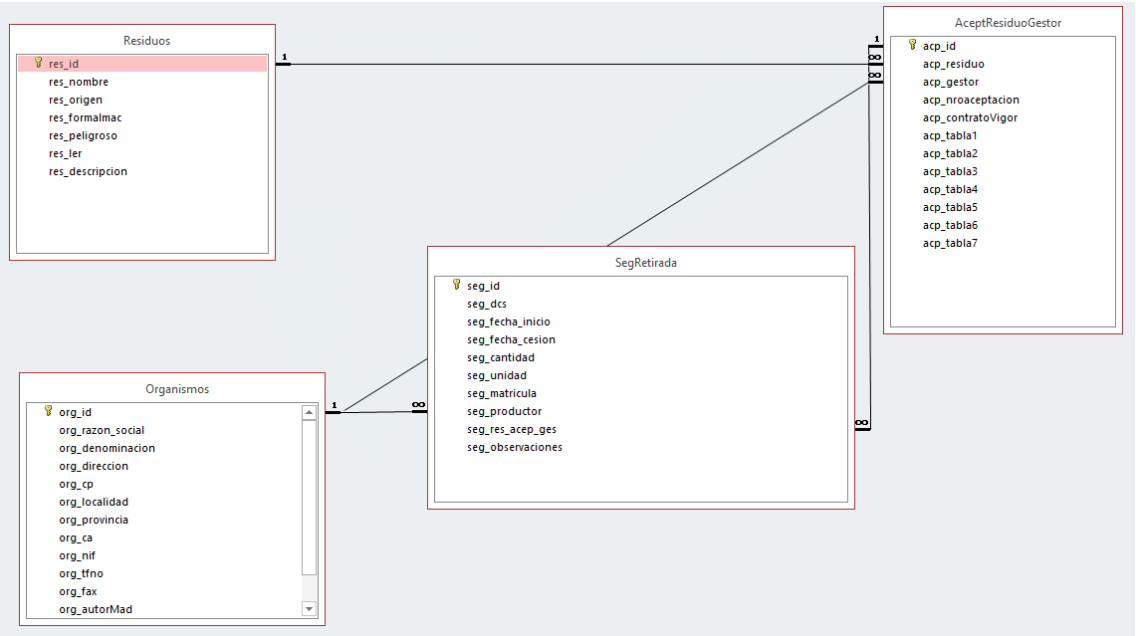


Figura 4.3 Esquema de la base de datos de residuos, generado con MS Access.

La base de datos para la gestión de residuos, maneja una cantidad de datos importante y se implementó por motivos ajenos al TFG primero en Microsoft Access. Para trabajar con el mismo tipo de BBDD se duplicó su estructura en SQL y por último se copiaron los datos que contenían las tablas de la base de Access. Para la creación y

carga de los datos iniciales de las BBDD se ha utilizado la herramienta «DB Browser for SQLite» con licencia GNU, *General Public License* (ver Figura 4.4).

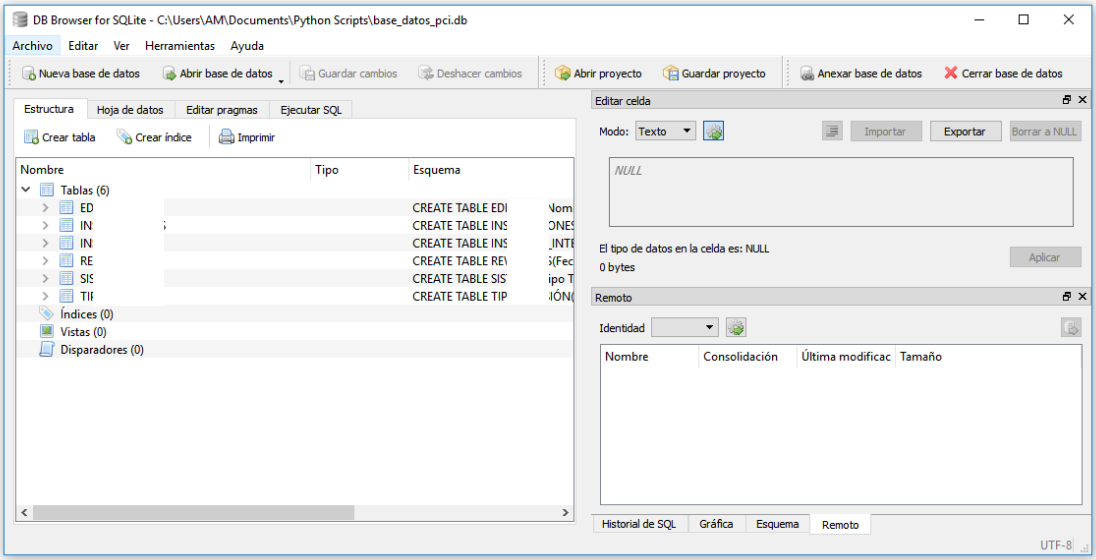


Figura 4.4 Vista de DB Browser for SQLite.

4.4.1 Uso del módulo QtSql

Como ya se ha mencionado, PyQt5 dispone de módulos que no son dedicadas exclusivamente al trabajo con la interfaz gráfica. En el caso de PyQt5.QtSQL [24], provee de un conjunto de objetos y funciones que permiten trabajar con BBDD permitiendo simplificar la interacción entre los objetos gráficos y éstas. Dentro de QtSQL se pueden identificar dos partes, la primera de para manejar *drivers* de BBDD y conexiones y la segunda para módulos de control y manejo de las BBDD.

Drivers y conexión

Por un lado, se identifica la parte de *driver* y conexión. Provee de una serie de objetos para la conexión y el control de los diferentes tipos de BBDD, en este caso del tipo SQLite.

```
db=QtSql.QSqlDatabase.addDatabase('QSQLITE') #driver de SQLite3
db.setDatabaseName(self.bdauxiliar)
db.open()
```

Figura 4.5 Ejemplo utilizado que carga el driver, indica nombre de la BBDD y realiza la conexión.

Módulos de control y uso de QtSQL

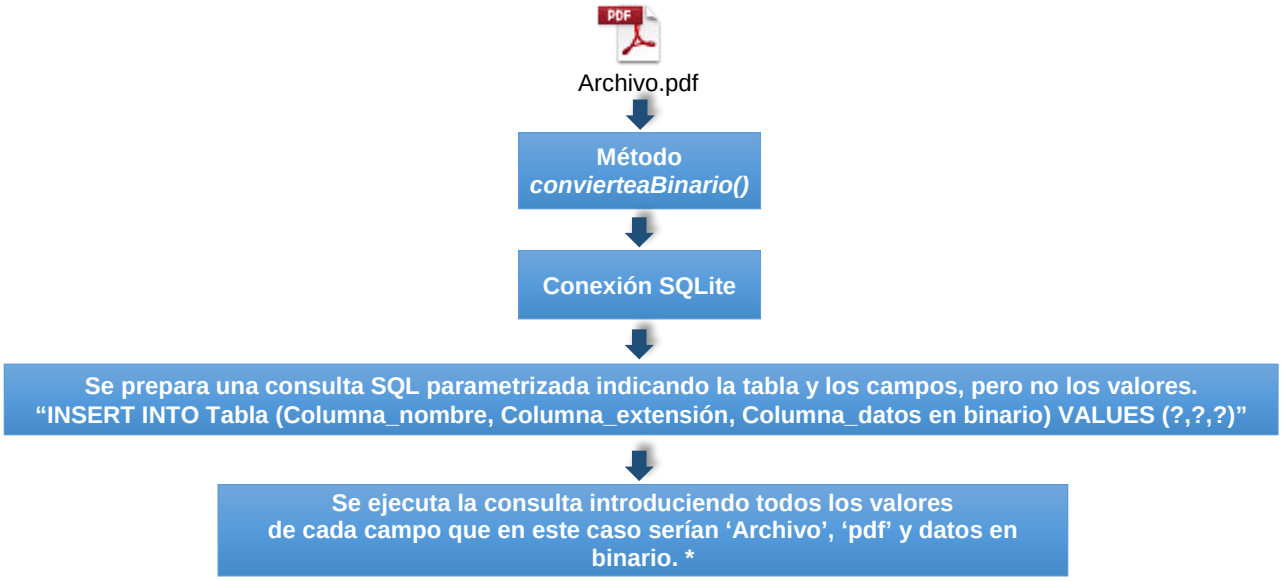
Para el manejo de la base de datos se utiliza lo que se denomina como la capa de SQL API. Para realizar consultas SQL se utiliza el objeto QSqlQuery o para el control de errores, QSqlError. El sentido de utilizar estos objetos es que ya realizan diferentes conjuntos de pasos que se deben dar si se desea programar desde cero las funciones de consulta o control de errores SQL.

Por último, para la interacción con los diferentes w*idgets* gráficos utilizados, PyQt utiliza los llamados modelos. Un modelo es un contenedor intermedio de las relaciones y datos de una determinada consulta. Se utiliza para realizar las modificaciones sobre los datos cargados en la memoria, permitir su manejo desde la parte gráfica y determinar cómo y cuándo almacenarlo en la base de datos permanente. El modelo se encarga de determinar cómo cargar las cabeceras, como representar la información sobre las tablas facilitando la programación SQL con el entorno gráfico.

El modelo más utilizado ha sido una extensión de *QSqlRelationalTableModel()*. Se ha tenido que extender la funcionalidad porque en muchos casos este objeto no permitía definir relaciones más complejas entre diferentes tablas.

4.4.2 Almacenamiento y recuperación de documentos en BBDD

Como se ha mencionado en el punto 3.3.1 sobre BBDD relacionales, las BBDD SQLite permiten almacenar cualquier tipo de archivo como datos binarios utilizando el tipo de datos BLOB. En este trabajo se necesitan almacenar archivos PDF, DOC y en algunos casos imágenes. Para ello se deben convertir estos archivos a datos binarios, en el caso de Python a *array* de *bytes* [25]. Posteriormente se insertan los datos referentes al nombre, la extensión del archivo y los datos en binario en la base datos SQLite.



*Se analizan los parámetros limitando su extensión y características para evitar un ataque por Inyección SQL.

Figura 4.6 Proceso de carga de archivo en la base de datos

Para recuperar un archivo desde la BBDD bastará con realizar los pasos inversos. Primero deberemos realizar la conexión a la BBDD y solicitar los datos almacenados, como el nombre, la extensión y el *array* de bytes que le corresponde. El nombre y la extensión se pasan como parámetros al método *QFileDialog.getSaveFileName()* así como el campo BLOB de datos que compone el documento. La ventana tipo "diálogo" *getSaveFileName()*, permite escoger el lugar donde guardar el archivo.

El control de si corresponde a un campo tipo BLOB e iniciar la conversión corresponde exclusivamente al programador, no se realizan comprobaciones.

Se requieren por tanto mínimo dos campos de datos para almacenar un archivo. Para este TFG se han establecido 4 campos que siguen el formato campo1_nombre, campo1_tipo, campo1_datos y campo1_detalle, siendo este último campo opcional para añadir observaciones o descripciones sobre el archivo.

El campo1_tipo almacena la extensión del archivo, en caso de que la pestaña muestre un icono de archivo, consulta este campo e inserta la imagen correspondiente al icono asociado por los usuarios a esta extensión. Solo se realiza con documentos tipo PDF, XLS, DOC e imágenes.

4.5 IMPLEMENTACIÓN INTERFACES GRÁFICOS

La implementación gráfica se desarrolla separadamente del código. Como se puede observar en el punto 4.3.4 QtDesigner, se ha utilizado la herramienta Designer de PyQt5 con la que se obtiene el resultado final del entorno gráfico diseñado con objetos QtWidget (tipo de ventana, pestañas, botones, visor de tablas…). Se guarda en un archivo *.ui* que contiene en formato XML los estilos, tamaños, posiciones y demás información del formato [23].

Se aplica al programa definiendo primero en el código una Clase ventana o *layout* (en este TFG se han utilizado QtDialog, QtMainWindow y Qt), que heredará de la clase a la que pertenezca la ventana del diseño *.ui* (ver diagrama de la Figura 4.7).

La herencia, en los lenguajes orientados a objeto como es Python permite heredar a una clase, los atributos y comportamientos definidos en otra que hará de clase base.

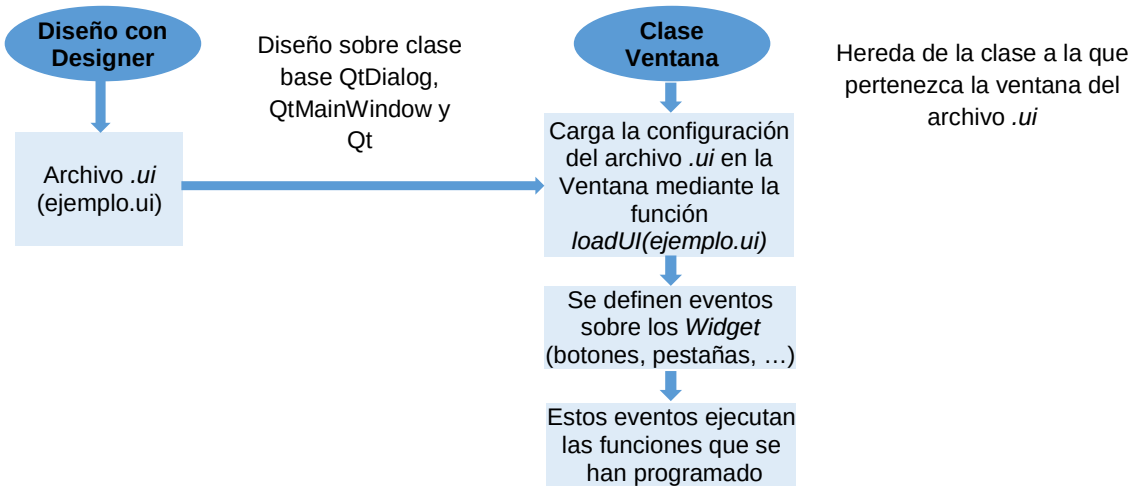


Figura 4.7 Proceso de carga del diseño gráfico con el código.

Una vez definida la clase sobre la que se aplicará el entorno diseñado, se utilizará la función *loadUI (nombre del archivo ui)* del conjunto PyQt5.uic para cargar el formato que se ha diseñado previamente con la herramienta Designer sobre la clase creada.

4.6 ESTRUCTURA INTERNA DEL PROGRAMA

La estructura implementada es muy estática, no crea objetos más allá de los programado inicialmente [25]. El patrón utilizado es *Singleton*, es decir de cada una de las clases, en su mayoría ventanas con las funciones asignadas, sólo se permite

instanciar una. Implementando el método especial en Python "*def __new__(cls):"* se permite solamente la ejecución de una sola clase que sean del mismo tipo, así solo habrá un solo módulo de SGC, SGA y PCI para evitar ejecuciones o funcionamientos indeterminados.

La estructura es la siguiente:

1. Clase *VentanaPrincipal()*, que se incluye en el *script* VentanaPrincipal.py
   1.1. Clase *SGA()*, incluido en el archivo SGA1_0_1.py
   1.2. Clase *SGC()*, incluido en el archivo SGC3_1_3.py
   1.3. Clase *PCI()*, incluido en el archivo PCI2_1_1.py
   1.4. Clase ventana auxiliar, implementada como *Window_aux()* en el archivo ventana_aux.py

Existen otros módulos auxiliares genéricos para realizar ciertas funciones comunes a los tres como:

- consultasqlite1_0_5.py con la clase *Consultabdlite()*. Esta tiene como atributos un modelo (de SQL, como los indicados en el punto 4.4.1) y el atributo base de datos donde le indicamos la base de datos de trabajo. Realiza las funciones necesarias para interactuar entre el modelo y la base de datos. Para ello se le ha programado las funciones *conectar(), modelo SQL(), update(), modo()* y *tablas()*.
- Exportadatos0_4_3.py con la clase *ExportarDatos()*. Es utilizada para agrupar las órdenes necesarias para exportar cada documento o tabla al tipo de archivo deseado. Así pues, para su uso, primero consulta qué elemento gráfico está activo y así determina qué debe exportar. Si es una tabla, seguirá lo indicado en el punto 4.12 para exportar a *.xls*. Si en cambio es un documento, realizará todos los pasos necesarios para reconstruir el archivo y servirlo para poder guardarlo donde se requiera.

4.7 VENTANA PRINCIPAL

La ventana principal es un objeto que hereda del objeto QMainWindow, se inicializa y se carga el formato desde el archivo *.ui* diseñado con anterioridad (ver punto 4.5).

```
class VentanaPrincipal(QMainWindow):
    def __init__(self, *args, **kwargs):
        super(VentanaPrincipal, self).__init__()
        #Carga el formato a la ventana desde el archivo VentanaPrincipal.ui
        loadUi('VentanaPrincipal.ui', self)
```

Figura 4.8 Ejemplo de inicialización y carga de ventana prediseñada.

A continuación, se enlazan los botones a las funciones que ejecutan el módulo asignado e importado previamente. Por ejemplo, el módulo de protección contra incendio se ejecutará cuando se haga clic al botón PCI (que se observa en la figura Figura 4.10) según:

```
def botonPCI_click(self):
    ventana=PCI(self)
    #muestra la ventana cargando el módulo PCI
    ventana.show()
```

Figura 4.9 Ejemplo de funcionamiento de los eventos (en este caso clic).

El resultado final queda de la siguiente manera, permitiendo acceder a cada uno de los módulos y a la ventana auxiliar:

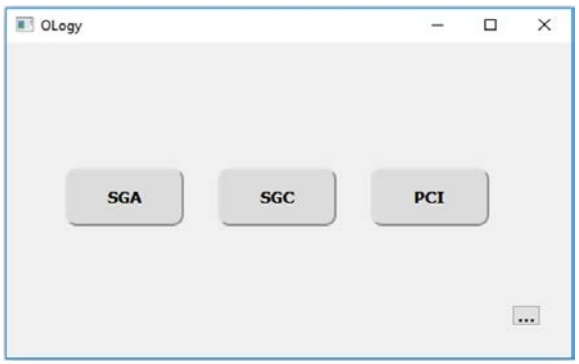


Figura 4.10 Vista de la pantalla principal con los tres módulos y el menú auxiliar.

4.8 VENTANA AUXILIAR DE OPCIONES

Esta ventana se utiliza para indicar las ubicaciones de las bases de datos. Con el objetivo de permitir que la aplicación y las bases de datos estén en lugares diferentes se ha dotado de la opción de poder seleccionar la ruta donde está cada una de las bases de datos. Con esto se puede permitir, por ejemplo, que un usuario pueda ejecutar el *software*, pero no tenga acceso directo a los archivos de las bases de datos por encontrarse en otra carpeta con unos derechos de acceso diferentes. Para ello se crea la clase *Window_aux()* contenida en el módulo Window_aux.py. Esta clase hereda del objeto PyQt5.QWidget y no de QMainWindow como en el caso anterior.

Posteriormente en la clase *Window_aux()*, se procede a cargar la vista de la ventana, ejecutarla y preparar las funciones asignadas a cada botón según el ejemplo siguiente:

```
class Window_aux(QWidget):
        def __init__(self, parent=None):
            super().__init__()
            dialog= QDialog(self)
            loadUi('ventana_auxiliar.ui', dialog) #carga formato
              #conecta el evento clic con la función pushButton_click
              dialog.pushButton.clicked.connect(self.pushButton_click)
              …
              '''Ejecuta la ventana del tipo QDialog y la mantiene abierta sobre la ventana principal QMainWindow'''
              dialog.exec()
```

Figura 4.11 Muestra de la clase Window_aux.

El resultado es el mostrado en la Figura 4.12 quedando por último la selección y carga de la ruta de las bases de datos.

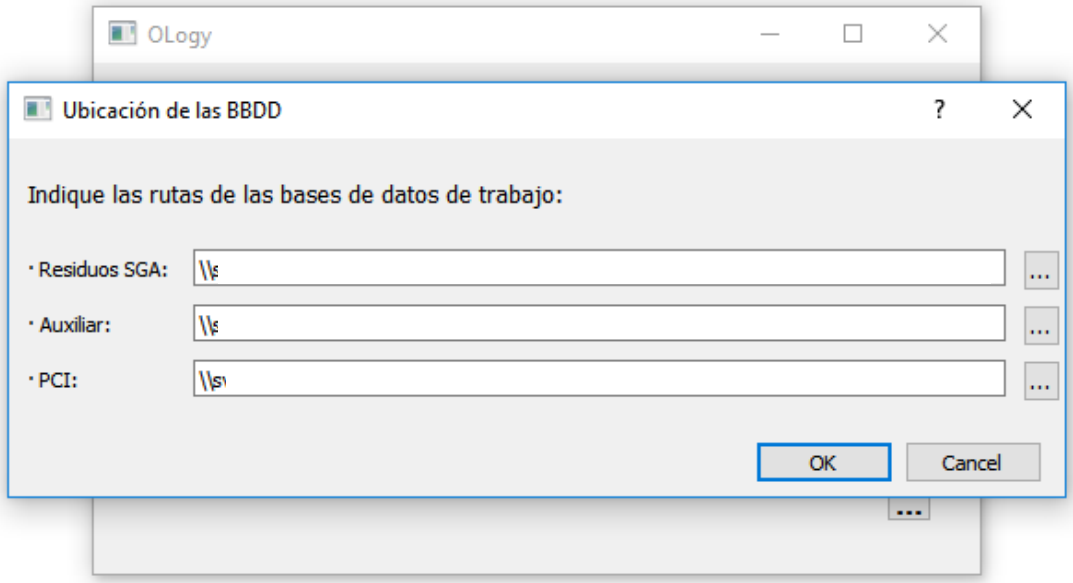


Figura 4.12 Vista de ventana auxiliar para selección de las rutas de las Bases de Datos.

Para la carga y selección de rutas se dispone del objeto QFileDialog que permite la creación de un menú tradicional de selección de archivos de manera rápida mostrando el resultado de la Figura 4.13, permitiendo modificar la mayor parte de los campos y comportamientos del submenú.

Lo anterior cobra sentido si se otorga acceso a usuarios solamente con nivel de administrador a esta ventana auxiliar de opciones. Por el momento no se ha limitado el acceso de los usuarios a esta ventana. De este modo, se evitaría que se pueda cambiar la ubicación de la base de datos y que se desvíe la información a una ubicación no deseada voluntaria o involuntariamente.

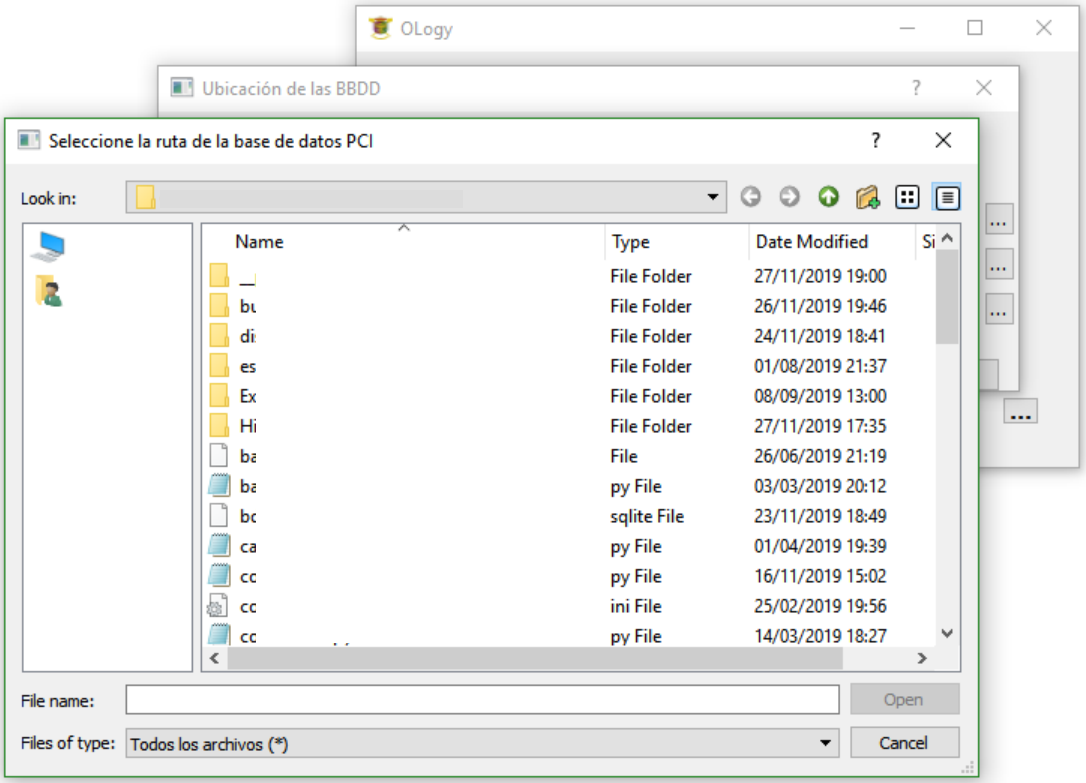


Figura 4.13 Ejemplo creado con el módulo QFileDialog de PyQt5.Widgets

4.9 CLASES PRINCIPALES Y PESTAÑAS DE FUNCIONES

Las clases principales SGA, SGC y PCI son un objeto de la clase QMainWindow como la *VentanaPrincipal()* que contendrán básicamente objetos pestaña. Para representar las distintas funciones dentro de estas clases principales será con las pestañas de *tabWidget* pertenecientes a la librería PyQt5.QtGui. Dentro de cada uno de los módulos se mantiene una organización del código similar. Después de cargar el formato a la clase principal, se establecen los eventos relacionados con cambios de pestaña, clics en los botones y funciones a realizar por defecto.

Para definir las funciones de las pestañas se presenta un problema. Al encontrarse todos los 'objetos pestaña' dentro de la misma clase principal/ventana (SGA, SGC o PCI) se definirán todos los métodos y funciones dentro de esta clase principal. No obstante, con objeto de mantener el código limpio y organizado, se han separado a la hora de programar las funciones por cada pestaña. Por tanto, en el código quedan claramente separados los botones, visualizadores gráficos, tablas y funciones auxiliares que requiera cada pestaña (ver Figura 4.14).

```
class PCI(QMainWindow):
    def __init__(self, *args, **kwargs):
        #Carga el formato a la ventana desde el archivo Ventana_pci.ui
        #Inicializa valores y asigna eventos
    #PESTAÑA1##########################################################
    def función_evento1_pestaña1:
    def función_evento2_pestaña1:
    def función_evento3_pestaña1:
    #PESTAÑA2##########################################################
    def función_evento1_pestaña2
    def función_evento2_pestaña2
```

Figura 4.14 Agrupación de las funciones por pestañas.

En el ejemplo que se muestra a continuación, todas las funciones de los botones, trabajo con las tablas y carga de documentos están programadas dentro de la clase SGA. No obstante, las funciones pertenecientes al resto de pestañas también están programados en la misma clase, haciéndose necesaria su organización para tener un código limpio y comprensible.

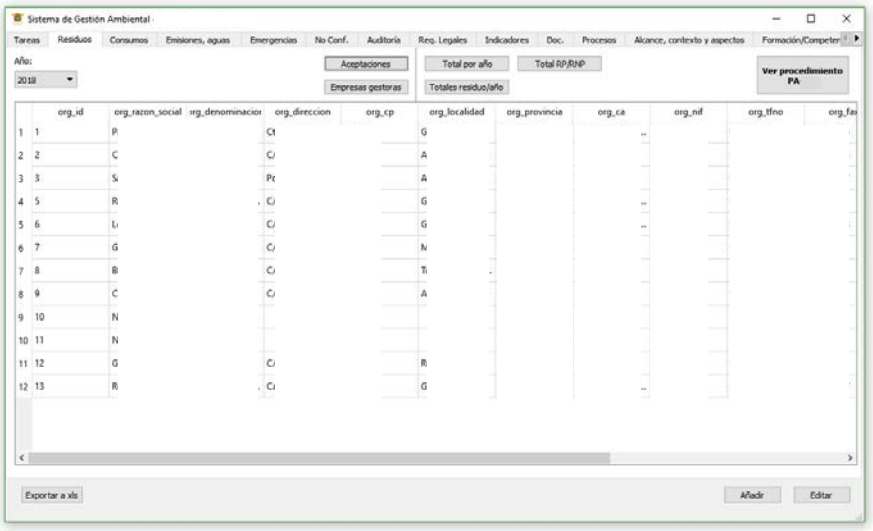


Figura 4.15 Pestaña Residuos y sus diferentes funcionalidades.

4.9.1 Modificando datos de la aplicación

En las pestañas donde se permite editar datos de una tabla, se ha implementado mediante un submenú similar al de la Figura 4.17. Aunque la base de datos se carga en un 'objeto modelo' QSQLRelationalModel (ver punto 4.4.1) presenta un problema considerable a la hora de volver a cargar los datos nuevos o actualizados del modelo a la base de datos. En una base de datos sencilla con relaciones entre tablas sencillas, a este objeto 'modelo' se le pueden indicar cuáles son las relaciones. De este modo, permite modificar la base de datos inmediatamente al actualizar la tabla física (mostrada en pantalla) sin ningún tipo de complicación.Los modelos de PyQt5.QtSQL ofrecen facilitar el manejo de la representación, modificación y actualización de las base de datos.

El problema es que sólo funciona correctamente con relaciones sencillas entre tablas. Para las bases de datos más complicadas, la modificación e inserción de datos no permite simplemente aplicar el método 'actualizar' al objeto 'modelo' o que lo haga automáticamente como se ha mencionado. Es el caso de GestResiduos, que tiene un número considerable de interrelaciones (Figura 4.3).

Al sobrepasar en demasía el alcance de este TFG, el implementar un método que respetase el funcionamiento de modelos relacionales complejos, se optó por establecer los menús Modificar y Añadir. Que varia según los casos, pero es similar al mostrado en la Figura 4.17. Lo que muestra este menú son las tablas que forman la base de datos. Para modificar un registro se deberá acceder mediante filtrado de los diferentes registros

y cambiar los datos seleccionados. Al ser modificada una tabla a la vez, se evita el problema descrito con los modelos relacionales complejos.

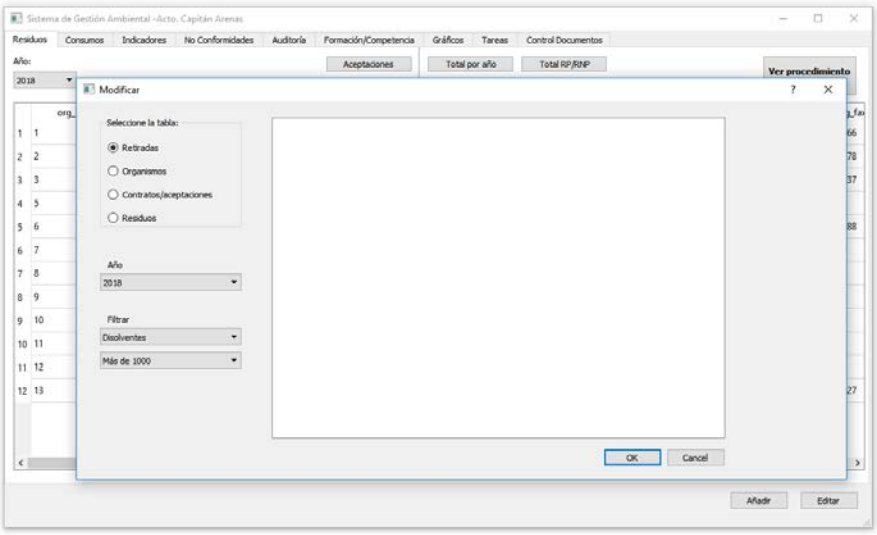


Figura 4.16 Ejemplo de menú "modificar" de las tablas de datos.

4.10 IDENTIFICACIÓN DE USUARIOS

La ventana principal queda bloqueada al inicio en la parte posterior para que no se pueda utilizar la aplicación hasta que se validen las credenciales. En caso de fallo en la autentificación se informa mediante mensaje de que el usuario o la contraseña no existen. Por el momento no hay ni un número máximo de intentos ni limitación por intentos/tiempo que son buenas prácticas para controlar el acceso de usuarios.

Siguiendo el diseño recogido en el punto 3.4 USUARIO Y CONTRASEÑA, en un documento se almacenará el hash resultante del usuario y de la contraseña. Un *hash* no es más que el resultado de operaciones matemáticas que transforman datos en una cadena encriptada irrecuperable.

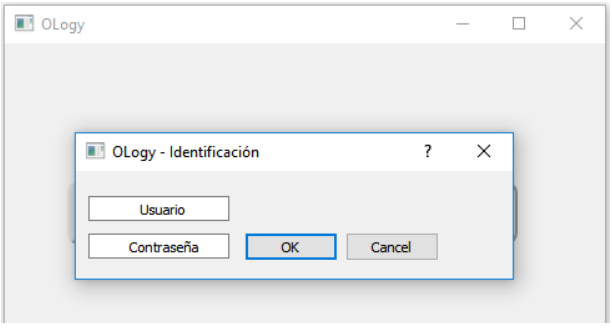


Figura 4.17 Aspecto de la ventana de autenticación de usuarios.

Para la encriptación del usuario y la contraseña se ha utilizado la biblioteca haslib cuenta entre otros con los algoritmos md5, sha1, sha224, sha256, sha384, y sha512. El resultado de introducir el usuario y la contraseña, separadamente por los algoritmos es guardado en un archivo. A continuación, este documento se cifra mediante una clave. Para ello se ha empleado la herramienta pyAesCrypt, cifrando el documento que contiene los *hash* de los usuarios y las contraseñas.

Para acceder a la aplicación, primero se realiza el *hash* del usuario y la contraseña con uno de los algoritmos indicados. El siguiente paso es descifrar el documento que contiene los usuarios y las contraseñas. Por último, se comprueba si el *hash* del usuario y de la contraseña aparecen en la lista. En caso de que sea negativo, se rechaza el acceso al usuario, indicando que el "Usuario y/o contraseña incorrectos o no existen".

4.11 CARGA DE PLANOS E IMÁGENES

En las partes donde se cargan imágenes se ha permitido su visualización mediante la carga de la imagen a un objeto QLabel. Posteriormente se le añaden objetos QScroll para navegar por ella.

```
def cargaplano(objetoimagen):
    planoimagen = QtGui.QImage(objetoimagen)
    leepixel=QtGui.QPixmap.fromImage(planoimagen).scaledtoWidth(self.planos_pci.width,
Qt.KeepAspectRatio, )
        self.planos_pci.setPixmap(leepixel)
```

Figura 4.18 Muestra del código que carga la imagen al visor.

Se observa en la Figura 4.19 que la función carga un objeto tipo imagen, ya que existe una función intermedia encargada de convertir los archivos que se hayan almacenado en la base de datos como tipo PDF y JPG, al tipo utilizado por QPixmap, que es PNG.

El ejemplo más claro para la sección «Planos del Sistema» en el módulo PCI. En la siguiente Figura 4.20 se visualizan las imágenes de los planos que aparecen seleccionables en la lista. Se permite "Exportar" el archivo original y añadir o eliminar nuevos planos.

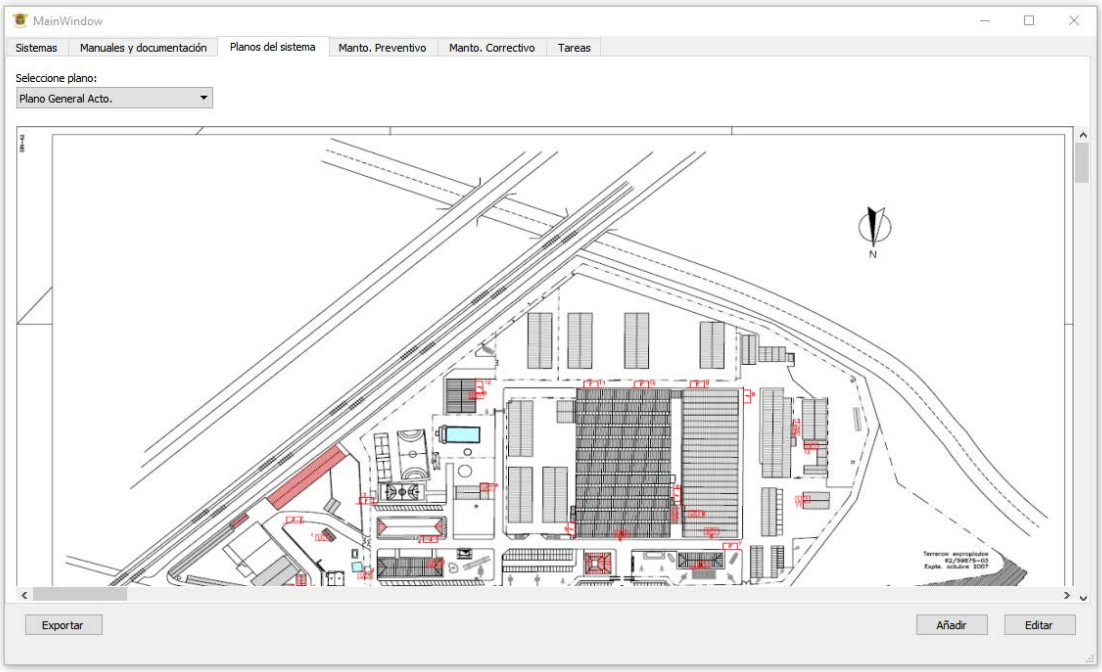


Figura 4.19 Vista de la carga de planos del sistema de protección contra incendio.

4.12 EXPORTACIÓN DE DATOS

La exportación de los datos de las tablas se realiza por el momento solamente a formato Excel. Algunas de las pestañas, disponen de la opción de exportar a *.xls* los datos mostrados en las tablas como la mostrada en la Figura 4.16.

```
def exportarxls(self):
    nombrearchivo,_ = QFileDialog.getSaveFileName(self, 'Exportar a archivo xls', '', ".xls(*.xls)")
    libro = xlwt.Workbook()
    hoja = libro.add_sheet("sheet", cell_overwrite_ok=True)
    """..."""
    for f in range(modelo.rowCount()):
        texto = modelo.headerData(f, QtCore.Qt.Vertical)
        hoja.write(f+1, 0, texto, style=estilo)
    for col in range(modelo.columnCount()):
        for f in range(modelo.rowCount()):
            texto = modelo.data(model.index(f, col))
            hoja.write(f+1, col+1, texto)
    libro.save(nombrearchivo)
```

Figura 4.20 Extracto del código para crear un archivo .xls

Para ello se hace uso del módulo nativo xlwt (Figura 4.20) que ofrece funciones para crear libros de Excel, añadir páginas, insertar o editar datos, etc.

No en todas las pestañas, el botón exportar se comporta del mismo modo. En el caso de los planos o los gráficos permite guardar el documento originalmente almacenado.

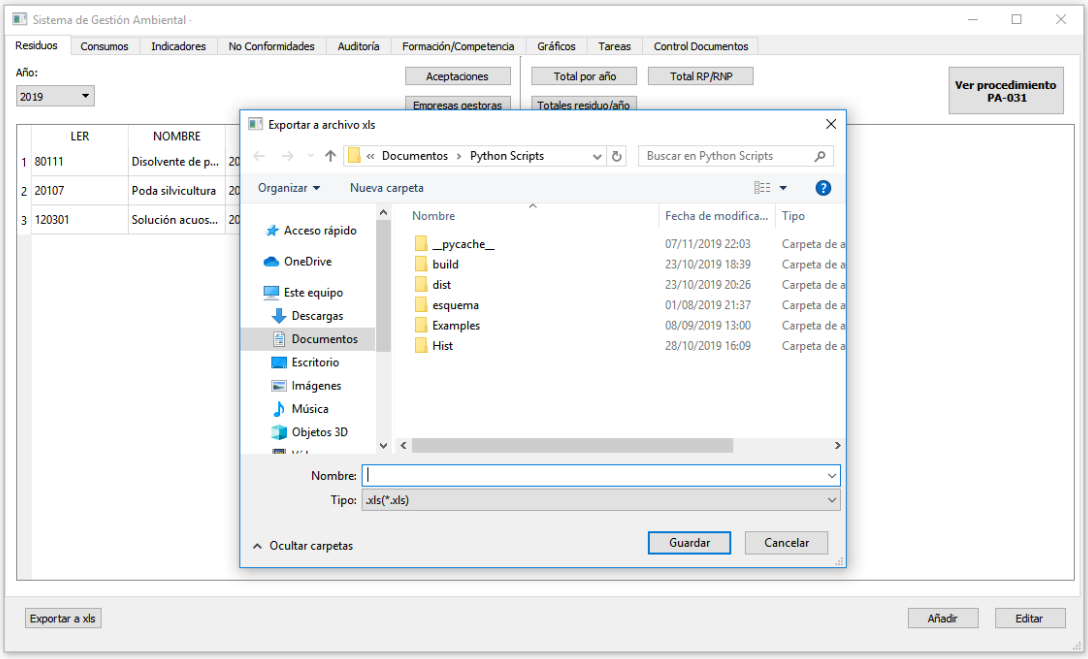


Figura 4.21 Ventana de exportación de datos a .xls.

4.13 EJECUCIÓN FINAL Y PRUEBAS

Mediante es uso de la herramienta cx_freeze [27] se ha creado un archivo *.exe* autoejecutable. Para su correcta ejecución se necesita crear un *script* de Python "setup.py" indicando manualmente las características del archivo, las bases de datos que deberán ser incluida, así como las librerías principales y en el caso de este TFG los archivos *.ui* creados.

No obstante, al utilizar PyQt5, se necesitan indicar sus componentes importados en el código (ver puntos 4.4, 4.7, 4.8, 4.9 y 4.12) individualmente, por ejemplo, PyQt5.QtSQL, PyQt5.Gui, PyQt5.Widgets, PyQt.Core, PyQt5.uic… A continuación, se debe ejecutar el comando *«Python setup.py build»* situados en ventana de comandos en la carpeta que contiene todos los *scripts* programados. Esto crea el archivo *.exe*, la carpeta *lib* con todos los *scripts* de Python necesarios y algunos archivos *.dll*, pero se

obtiene un error. Para subsanarlo se deberán incluir manualmente dos *scripts* faltantes de la librería xlwt (ver punto 4.12) en la carpeta de las librerías que genera el comando *build* junto al EXE. Por otro lado, se deberá crear también una carpeta con el nombre *Platforms* con el archivo *qwindows.dll* (que es el archivo del *framework* PyQt5).

La carpeta con todos los archivos necesarios, se copia en los ordenadores de los usuarios. A continuación, se crea una carpeta en la red conteniendo las bases de datos. Diariamente se hace una copia de seguridad de las bases de datos en un servidor.

A lo largo del desarrollo se han ido ejecutando las diferentes versiones con nuevas funciones implementadas incrementalmente. Los cambios y/o mejoras por tanto se han añadido siguiendo el principio de eficiencia, programando primero los cambios menos exigentes para conseguir un *software* con la mayor funcionalidad posible en el tiempo disponible. Los usuarios, por tanto, han podido verificar el funcionamiento o solicitar de manera periódica sirviendo además estas pruebas sobre el usuario final para detectar errores o funcionamientos erróneos.

Capítulo 5.
CONCLUSIONES Y PROPUESTAS PARA LA MEJORA

“Science advances one funeral at a time.”

— Max Planck

Se determinan las conclusiones del desarrollo llevado a cabo y se indican mejoras relacionadas tanto como con nuevas funciones como complementar otras ya existentes. Éstas mejoras son fruto en su mayoría de la demanda de los usuarios.

5.1 CONCLUSIONES

Tanto los módulos del Sistema de Gestión Ambiental como de Calidad, recogen todos los requisitos documentados de los sistemas permitiendo almacenar y controlar los documentos, así como los datos de todos los apartados necesarios. Esto facilita que con los cambios de personal se mantenga la trazabilidad y cumplimiento exigidos por las normas UNE-EN ISO 14001 y 9001.

Los usuarios finales determinan que, aunque el programa cumple con sus especificaciones, para cumplir por completo los requisitos de los Sistemas de Gestión se requiere incorporar funcionalidades de análisis de datos y creación de informes. Asimismo, también requieren que para nuevas versiones se incorporen listas de chequeo y control periódico de tareas.

En cuanto al módulo de gestión de los sistemas contraincendios se ha logrado crear un sistema sencillo para controlar y mantener eficazmente las instalaciones y su documentación. Aunque en este módulo, por parte de los usuarios se considera que se mejora mucho toda la gestión de la documentación y por tanto que cumple con las expectativas, se solicita que cuente con un gestor de tareas que indique qué se debe realizar y chequear su cumplimiento. Para ello se tendrá en cuenta como propuesta de mejora para los módulos relacionados con mantenimiento de instalaciones.

Se consigue crear un *software* capaz de cumplir con las exigencias recopiladas de los usuarios. Proporciona las funcionalidades exigidas y con unas prestaciones suficientes para permitir la mejora de la productividad en la empresa. En líneas generales se puede asumir que se ha cumplido el objetivo de este TFG.

5.2 MEJORAS

Algunas de las mejoras que se pueden realizar a esta aplicación, a parte de las mencionadas en el apartado anterior pueden ser:

- Desarrollo de la documentación.

Aunque en el código se han especificado bien las funciones y subrutinas de las diferentes partes, es necesario documentar todo el programa. Dado al riesgo que supone que el desarrollo haya sido realizado por una sola persona, se hace absolutamente imprescindible que antes de proceder con ninguna mejora se deben

crear manuales de usuario, se defina claramente las especificaciones del *software* y se especifique punto a punto las estructuras y variables utilizadas. Para ello se propone el uso de DOCSIE, que ofrece un entorno sencillo pero potente de generar documentación *web* y estructuras documentales muy completas y fácilmente exportables.

- Extender la aplicación al mantenimiento de otras infraestructuras del centro.

Como se ha probado útil y funcional para la gestión de las instalaciones contra incendios, se procederá a desarrollar e integrar otros módulos principales como son instalaciones térmicas, instalaciones eléctricas e instalaciones de abastecimiento de agua.

- Establecer un sistema integrado de gestión

Debido a que la gestión de la prevención de riesgos laborales sea ajena a las dependencias para las que se ha diseñado ese programa, no se ha diseñado el programa orientado a crear un Sistema de Gestión Integrado. No obstante, sí que permite mediante unos pocos cambios en las bases de datos, y añadiendo un módulo, si en un futuro fuese necesario, implementar esta funcionalidad.

- Personalizar la aplicación para cada tipo de usuario.

Cada usuario no utiliza todos los módulos de la aplicación por lo que, para mejorar su experiencia, así como la seguridad de acceso a los datos, se puede personalizar el acceso a los diferentes módulos y subapartados. Mediante la creación de un perfil de acceso en el momento del alta del usuario, se puede restringir el acceso de una manera sencilla a ciertas pestañas.

- Implementación sobre entorno *web*.

Uno de los más populares *frameworks* para desarrollo *web* con Python es Django. Una de las opciones para valorar es exportar el diseño al entorno *web*. Mejoraría la compatibilidad al trabajar desde navegador y facilitaría el acceso desde cualquier punto siguiendo la filosofía de los entornos *cloud*.

- Mejoras en la seguridad de acceso, identificación y de la seguridad de conexiones.

La seguridad es un punto clave para la empresa. Como ya se ha mencionado, se deberá limitar aún más el acceso de los diferentes usuarios a determinadas partes del

software. Además, si se utilizan bases de datos con conexiones se deberán seguir las pautas de seguridad marcadas por la empresa.

Anexo I

BIBLIOGRAFÍA

[1] Europa Press, *La digitalización empresarial en España dispararía un 5% el PIB y crearía un millón de empleos al año*, 2018. [En línea]. Disponible en: https://bit.ly/37HFI6i. [Accedido: 02-sep-19]

[2] "Digitalización y productividad", Observatorio ADEI. Google, Madrid,España, 2015. [En línea]. Disponible en: https://bit.ly/33mM2gn. [Accedido: 01-sep-19]

[3] *UNE-ISO Conservación a largo plazo de información electrónica basada en documentos.* Norma UNE-ISO/TR 18492:2008. [Accedido: 24-sep-19]

[4] *ISO Preservación de los documentos digitales: Guía para comenzar. Version española*, ISO/TC 46/SC 11, [En línea]. Disponible en: https://www.iso.org/committee/48856.html. [Accedido: 23-sep-19]

[5] *ISO Información tecnología - Lenguajes de base de datos- SQL*, Norma ISO/IEC 19075:2016.

[6] *UNE-EN ISO Sistemas de gestión ambiental. Requisitos con orientación para su uso.* Norma UNE-EN ISO 14001:2015.

[7] *UNE-EN ISO Sistemas de gestión de la calidad. Requisitos.* Norma UNE-EN ISO 9001:2015.

[8] A. Cockburn, *Agile Software Development The Cooperative Game: An Overview*, 2 ed., Ed. Addison Wesley, 2007.

[9] C. Larman y V. Basili, "Iterative and Incremental Development: A Brief History", *Computer,* pp. 47-56 , jul. 2003.

[10] I. Sommerville,"Iteración de Procesos" , *Ingenieria de Software*, 7 Ed., Ed.M. Martín-Romo, Madrid: PEARSON EDUCACIÓN, pp. 65-69, 2005.

[11] V. B. &. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development", *IEEE Trans. Software Eng*., pp. 390-396, dic. 1975.

[12] "Desarrollo iterativo e incremental ", *Se hace camino al andar…*, 22-sep-2019 [En línea]. https://bit.ly/2XNtuoc. [Accedido: 10-oct-19]

[13] S. Thomas, "Revisiting the Iterative Incremental Mona Lisa", *It's a Delivery Thing*, 3-dic-2012. [En línea]. Disponible en: https://bit.ly/2OlmYlG. [Accedido: 10-oct-19]

[14] "La guía de los fundamentos para la dirección de proyectos", *Guía del PMBOOK*, Ed. 6, Ed. Newtown Square: Project Management Institute, Inc., 2017.

[15] *IEEE Especificación de Requisitos Software*, IEEE Std. 830-1998. [En línea]. Disponible en: https://bit.ly/33nhRWv. [Accedido: 20-sep-19]

[16] I. Gordon, "Software Quality Attributes", *Essential Software Architecture*, 2 Ed., Springer, 1998.

[17] T. Mandel, "The Golden Rules of User Interface Design", *The elements of User Interface Design*, Ed. Jhon Wiley & Sons, pp. 5-1 a 5-28, 1997.

[18] Python Software Foundation, "sqlite3 — DB-API 2.0 interface for SQLite databases", 2019. [En línea]. Disponible en: https://docs.python.org/3.7/library/sqlite3.html. [Accedido: 25-nov-19]

[19] M. Stamp, "A Taxonomy of Cryptography", en *Information Security: Principles and Practice*, vol. 2, Ed. San Jose, California: WILEY, pp.40-41, 2011.

[20] P. Krill, "Python popularity reaches an all-time high", InfoWorld, 10-jun-2019. [En línea]. Disponible en: https://bit.ly/2QRgbld. [Accedido: 13-nov-19]

[21] "Anaconda Documentation", 2019. [En línea]. Disponible en: https://docs.anaconda.com/anaconda/. [Accedido: 05-sep-19]

[22] “Spyder: The Scientific Python Development Environment — Documentation”, 2019. [En línea.]. Disponible en: https://docs.spyder-ide.org/. [Accedido 05-sep-19]

[23] B. Harwani, *Qt5 Python GUI Programming Cookbook*, Ed. Birmingham: Packt, 2018.

[24] Alan D. Moore, "Exploring SQL with QtSQL", en *Mastering GUI Programming with Python*, Ed.Packt, pp. 472-474 2019.

[25] Pynative, Files and images as a Blob in MySQL. [En línea]. Disponible en: https://bit.ly/2QQ4Lyh. [Accedido: 14-oct-19]

[26] I. Sommerville, "Iteración de Procesos" , en *Ingenieria de Software*, 7 Ed., M. Martín-Romo, Ed. Madrid: PEARSON EDUCACIÓN, pp. 65-69, 2005.

[27] Anthony Tuininga, “cx_Freeze Documentation”, Release 6.0. 30-ago-2019. [En línea]. Disponible en: https://bit.ly/37GCssb. [Accedido: 1-oct-19]

[28] Biblioteca de la Universidad Pública de Navarra. Oficina de Referencia. “Guía para citar y referenciar. IEEE Style”, 2016. [En línea]. Disponible en: http://www2.unavarra.es/gesadj/servicioBiblioteca/tutoriales/Citar_referenciar_(IEEE).pdf. [Accedido: 20-nov-19]

[29] Biblioteca de la Universidad Pública Carlos III de Madrid. "Guía temática sobre citas bibliográficas UC3M: IEEE v11.12.2018". [En línea]. Disponible en: https://bit.ly/2XM3V7a [Accedido: 24-nov-19]

Anexo II.

ABREVIATURAS

ADEI	Análisis y el Desarrollo Económico de Internet
AENOR	Asociación Española de Normalización y Certificación
API	*Application Programming Interface*
BBDD	Bases de datos
EN	*European Norm*
ERS	Establecimiento de Requisitos del *Software*
GUI	*Graphic User Interface*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
ISO	*International Organization for Standardization*
PCI	Protección contra incendio
PDF	*Portable Document File*
SAGL	Sistema de Apoyo de Gestión Logística
SGC	Sistema de Gestión Ambiental
SGC	Sistema de Gestión de la Calidad
SQL	*Structured Query Language*
TFG	Trabajo Final de Grado
UNE	Una Norma Española