Universitat de les Illes Balears

Escola Politècnica Superior

**Memoria del Trabajo de Fin de Grado**

# TmpMaker, aplicación para la elaboración de documentos notariales

José Santiago Ibarbia

Grado de Ingeniería Informática

Año académico 2016-17

DNI/NIE del alumno: X7540175-Q

Trabajo tutelado por Dr Antoni Lluís Mesquida Calafat
Departamento de Ciencias Matemáticas e Informática

| Se autoriza a la Universidad a incluir este Trabajo en el Repositorio Institucional para su consulta en acceso abierto y difución en línia, con finalidades exclusivamente académicas y de investigación. | Autor | | Tutor | |
|---|---|---|---|---|
| | Sí | No | Sí | No |
| | X | | X | |

Palabras clave del Trabajo:
TmpMaker, Gestión de documentos, Actas notariales, Plantillas Word

# Índice

# Lista de ilustraciones

# Lista de tablas

# Acrónimos

**ACID:** Atomicity, Consistency,Isolation, and Durability

**BBDD:** Base de datos

**BCL:** Base Class Library

**CLR:** Common Language Runtime

**CPU:** Central Processing Unit

**DNI:** Documento Nacional de Identidad

**LINQ:** Language Integrated Query

**NIE:** Número de Identidad de Extranjero

**SGBD:** Sistema Gestor de base de datos

**SQL:** Structured Query Language

**UML:** Unified Modeling Language

**VBA:** Visual Basic for Applications

# Glosario

**Addon:**

Extenciones o plugins instalables en los proyectos de la fundación Mozilla.

**BBDD:**

Abreviatura utilizada para referirse a una base de datos. También se suele utilizar DB, aunque estas se refieren al término en inglés.

**BCL:**

Biblioteca de clases base incluida en .NET Framework formada por más de cientos de tipos de datos y que permiten al programador crear sus propios tipos haciendo uso de la herencia, entre otras muchas cosas.

**CLR:**

Es un entorno para los programas que se ejecutan en .NET. Éste se encarga de compilar un código intermedio en el lenguaje máquina.

**Cookie:**

Información básica almacenada, relacionada con la experiencia de usuario, a través de la cual se pretende ofrecer ciertas acciones sobre la actividad previa del usuario en un sitio web.

**Diagrama de Gantt:**

Es una herramienta que sirve para representar gráficamente los tiempos relacionados con una serie de tareas o actividades a lo largo de un periodo determinado.

**Diseño Low-Fidelity:**

Prototipos que implementan aspectos generales del sistema sin entrar en detalles. Permiten abarcar un abanico de posibilidades bastante amplio.

**Framework:**

Es un conjunto estandarizado de conceptos, prácticas y criterios, que en el caso particular del software, suele conllevar a unos artefactos o módulos software asociados, con el fin de estandarizar y facilitar la tarea de la programación.

**Layout:**

Esquema que será utilizado para distribuir los elementos y formas dentro de un diseño.

**Librería:**

Es un archivo o conjunto de archivos que se utilizan para facilitar la programación. Concepto muy similar al de Framework.

**LINQ (Language Integrated Query):**

Conjunto herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: objetos, xmls, bases de datos, etc...

**Macros (Excel o Word):**

Conjunto de comandos escritos en VBA que se pueden integrar en dichas aplicaciones con el fin de automatizar ciertas tareas.

**Path:**

Es la forma de referenciar un archivo informático o directorio en un sistema de archivos de un sistema operativo.

**Plugin:**

Es una aplicación (o programa informático) que se relaciona con otra para agregarle funcionalidades o diseños nuevos.

**SGBD:**

Siglas de Sistema gestor de bases de datos, que son una serie de programas que administran y gestionan la información que contiene una base de datos**.**

**SQL:**

Siglas de Structured Query Language o lenguajes de consulta estructurada.

Este lenguaje se caracteriza por ser utilizado en bases de datos relacionales.

**Stores/ Tiendas de aplicaciones:**

Servicio a través del cual los desarrolladores y empresas ofrecen sus softwares al mundo entero, permitiendo a los usuarios la descarga de los mismos, ya sea de manera gratuita o de pago.

**Tests Unitarios /UnitTests:**

Son pruebas que se ejecutan para comprobar el correcto funcionamiento de un módulo software.

**Tooltip:**

Es una herramienta de ayuda visual, que funciona al situar el cursor sobre algún elemento gráfico, mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.

**UML:**

Siglas de Unified Modeling Language, lenguaje de modelado unificado en castellano.

Se trata de un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

**VBA:**

Es el lenguaje de **Macros** de Microsoft Visual Basic que se utiliza para programar aplicaciones Windows y que se incluye en varias aplicaciones Microsoft. **VBA** permite a usuarios y programadores ampliar la funcionalidad de programas de la suite Microsoft Office.

**WebService:**

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

# 1 Introducción

Este es un proyecto en el que se ha diseñado e implementado un sistema para simplificar de manera sustanciosa el trabajo de los/as notarios/as, facilitando la elaboración de documentos notariales y la clasificación de los mismos.

En el primer capítulo de la memoria se expone la base del proyecto, contando con una descripción del mismo y una pequeña contextualización del panorama actual y de las tecnologías que disponen los/as notarios/as hasta hoy (año 2017).

Además se añaden las motivaciones que llevaron a emprender el proyecto y los objetivos que como estudiante y becario en una notaría me he llegado a plantear.

## 1.1 Descripción del proyecto

La realización del trabajo de final de grado se llevó a cabo en la notaría Armando Mazaira, cumpliendo una serie de objetivos mínimos de funcionalidad antes de una fecha establecida.

Al acabar el periodo de prácticas, se aumentó la funcionalidad del software con el fin de poderlo presentar como proyecto de final de grado.

El proyecto consiste en una aplicación, llamada TmpMaker, que facilita ciertos aspectos de las tareas realizadas por notarios y administrativos, que suelen necesitar un sistema para crear documentos, donde necesitan reutilizar cierta información: párrafos, referencias, citas, etc.

En el caso particular de los notarios, ellos crean documentos con párrafos exactamente iguales y utilizan para ello plantillas u otros documentos similares para hacer un copiar-pegar. Este proyecto consiste en diseñar un sistema para facilitar la creación de estos documentos, almacenando fragmentos de texto a los que mediante un click se puedan consultar, modificar o adjuntar a un documento de texto. De esta manera, un notario podría crear un documento mediante una secuencia de clicks y optimizar el tiempo y esfuerzo requerido para sus tareas habituales de edición.

TmpMaker es un software que permite crear documentos mediante una serie de clicks, partiendo de la definición de un tipo de documento.

El usuario puede crear distintos tipos de documentos en función de sus necesidades y editarlos a su medida. Cada documento puede dividirse en secciones, donde cada sección puede tener múltiples plantillas y/o documentos.

La aplicación está dividida funcionalmente y gráficamente en dos partes. La primera consta del panel de selección de tipos de documento y configuración del sistema, donde se pueden definir distintos tipos de documentos, tales como emails, actas notariales, cartas formales, artículos, entre otros. De esta manera se pueden clasificar documentos de la manera que deseé el usuario.

Al seleccionar uno de los tipos de documentos creados, se abre automáticamente un documento en un editor de texto y una pantalla nueva de TmpMaker, los cuales interactuarán entre sí, mediante una serie de acciones disponibles en la nueva pantalla de gestión de documentos de la aplicación.

Se disponen de diversas acciones, tales como definir secciones o zonas del documento y crear plantillas o subdocumentos que se pueden relacionar con cada una de las secciones asociadas al documento.

De esta manera, podremos crear un documento nuevo asociado al tipo de documentos ‘emails’, cuyas secciones sean ‘Saludo’, ‘Cuerpo’, ‘Despedida’, ‘Firma’ y asociar diversos textos a cada una de las secciones, como puede ser ‘Saludo1’, ’Saludo formal 1’, ‘Saludo estándar’, etc.

Podríamos reflejar las posibilidades de un documento tipo ‘emails’ como la siguiente lista:

- Emails
  - Saludo
    - Saludo 1
    - Saludo formal 1
    - Saludo estándar
  - Cuerpo
    - Información contacto
    - Aviso aceptación crédito
    - Notificación pagos y deudas
  - Despedida
    - Despedida formal 1
    - Despedida estándar
  - Firma
    - Firma oficial

TmpMaker está pensado como una extensión del editor de textos utilizado por la Notaría Armando Mazaira, que es Microsoft Word 2010 [1], aunque también da soporte a otras versiones de Word.

Se trata de un software que no se comercializará en las primeras versiones, y que puede ser abierto, ya que puede ser útil para cualquier persona que trabaje con este editor de textos y necesite de un sistema con dichas características.

## 1.2 Contextualización y antecedentes

Los editores de texto actuales permiten gran variedad de acciones, muchas de las cuales los notarios utilizan a diario. En la notaría en donde se me encomendó el proyecto, decidieron utilizar Word 2010, ya que la interfaz que proporciona es más amigable que en la versión anterior y las licencias del nuevo Microsoft Office no entraban dentro del presupuesto. A pesar de ello, Word no aportaba todo lo que necesitaban para optimizar su trabajo y no encontraron una aplicación en donde se pueda automatizar ciertas tareas. En primera instancia, la notaría comenzó a utilizar unas macros creadas por terceros para disponer de una serie de acciones simples, basadas en el formato del texto, pero con el tiempo, quedaron desactualizadas y los trabajadores no sabían instalarlas o cargarlas en nuevos equipos configurados; asunto que empeoró con la migración de todos sus equipos de Windows 7 al 8 [2].

Por otra parte, una de las aplicaciones que se utilizaba para cubrir algunas necesidades era Clipboard Master, que no es más que un clipboard algo más avanzado y que almacena más datos. El problema es que no se notaba una mejora en la productividad, ya que la funcionalidad que ofrecía era muy limitada.

Además de estas soluciones para sus problemas cotidianos, se sumaba la lentitud de sus equipos, ya que con tantos programas abiertos y ventanas de Word y Excel, su consumo de CPU y memoria se veían afectados. La solución a todos estos problemas era sencilla: centralizar todo en un único software que actúe junto a un editor de texto.

Los editores de texto actuales [3] son muy potentes; permiten un elevado número de acciones y muchos son personalizables (vía *macros* o extensiones). Muchos son avanzados e incluso de código abierto. Uno de ellos es LibreOfficeWriter, el cual permite un abanico de acciones mucho más amplio que Word y Microsoft Office en general. Debido a que Microsoft Word tiene un buen soporte técnico y que ya se disponía de las licencias, se optó por trabajar con Microsoft Word.

## 1.3 Motivación

Durante el periodo que estuve de becario en la notaría, se me plantearon diversos proyectos, teniendo yo el control sobre los mismos, y plena libertad a la hora de tomar decisiones. Para algunos de ellos no disponía ni de recursos, tiempo ni conocimientos, como por ejemplo un escáner de DNI/NIE que rellene un formulario. Otros eran muy simples o no muy útiles para la propia empresa.

Una de las principales causas por las que me he decantado por este proyecto es esta misma libertad, ya que considerando el abanico de posibilidades que me dieron, consideré que este proyecto era el que más podía aportar a la empresa, facilitando muchas de las tareas que tenían los trabajadores en ese momento.

Mi tutor de prácticas Armando Mazaira, me dio simplemente una serie de requisitos de usuario y un ordenador personal para poder comenzar a desarrollarlo, aunque por otra parte, disponía de un límite temporal para acabar el mismo, que era de sólo un mes.

La motivación principal fue poder cumplir grandes objetivos personales y aprender lo máximo posible por mi cuenta, haciendo una aplicación útil, con un fácil mantenimiento, personalizable y con una tecnología que no había utilizado hasta el día de hoy, que puedo decir que trabajo en una empresa puntera en el desarrollo en la misma tecnología.

## 1.4 Objetivos

Los objetivos marcados estaban muy marcados por la necesidad de presentar un software que cumpliese los requisitos mínimos antes de octubre del 2015. Sin embargo, mientras se iba desarrollando el software, nuevas ideas fueron surgiendo y por lo tanto, la opción de ampliar la funcionalidad en una segunda versión comenzó a tomar fuerza.

Como bien he dicho, la primera restricción era temporal:

- Disponer de un producto entregable antes de una fecha límite que era el 3/10/2015.

Luego se divide los objetivos en dos tipos; los del proyecto y los personales.

Los del proyecto son:

- Diseñar una interfaz con Windows Forms que asegure un alto grado de usabilidad.

- Disponer de un software testeado y que no necesite un mantenimiento software en sus primeras versiones.
- Cubrir todos los requisitos especificados en el documento de requisitos.
- Permitir que el software sea escalable.
- Disponer de un modelo de base de datos relacional consistente.
- Utilizar estándares de diseño de aplicaciones Desktop.

Luego tenemos los personales:

- Aplicar lo aprendido a lo largo de la carrera, focalizándome en mi ámbito que es la gestión.
- Aprender a desarrollar en un nuevo lenguaje de programación y a utilizar nuevas herramientas software.
- Coger experiencia en la toma de requisitos con usuarios potencialmente no técnicos
- Realizar tests que abarquen todas las casuísticas y aprender a realizar tests de estrés.
- Aprender a monitorizar los tiempos de una aplicación y analizar los mismos.

## 1.5 Términos usados en esta memoria

A lo largo de la memoria utilizaremos 4 conceptos fundamentales a través de los cuales se especifica la definición y el comportamiento del programa.

**Tipo de documento**

Este concepto se utiliza para asociar un grupo de documentos (archivos .docx), en un grupo común. Un ejemplo podría ser ‘Emails’

En este caso, el usuario crearía un tipo de documento ‘Emails’ y se le asociaría un documento vacío que se puede utilizar como plantilla para la elaboración de N documentos de tipo ‘Emails’, que comparten secciones, plantillas y características.

Es responsabilidad del usuario almacenar cada documento creado con esta plantilla con el nombre y ruta que desee.

**Documento**

Corresponde únicamente al fichero .docx. Éste puede ser modificado y es el que está visible en el editor de texto mientras tenemos el software en ejecución.

**Sección**

Segmentos del documento creados por el usuario. No hay límites y sus nombres tienen que ser autodescriptivos. De esta manera podemos clasificar textos/plantillas y facilitar la tarea de creación y vinculación de las mismas al documento.

**Plantilla**

Es la entidad en la que se basa el proyecto, la cual se puede añadir mediante un click al documento final o incluso editarse en un panel de texto de la propia aplicación. Estas plantillas pueden ser fragmentos de texto, tablas o hasta documentos enteros.

## 1.6 Estructura de la memoria

La memoria está dividida en nueve capítulos, cada uno con un objetivo distinto de cara al lector.

En el segundo capítulo (*2. Marco tecnológico*), se presentan las circunstancias y decisiones que se tuvieron que escoger para el desarrollo del proyecto, en cuanto a tecnicidades.

En tercer capítulo (*3. Planificación*), se presentan las tareas junto a la planificación temporal de todos los aspectos del proyecto y los riesgos implicados.

En el cuarto capítulo (*4.Análisis*), se presentan aspectos como la identificación de usuarios y el análisis de requisitos funcionales y de usuario.

El quinto capítulo (*5. Diseño*), se presentan los aspectos relacionados con el diseño, tanto modelos como las maquetas de la interfaz gráfica de usuario y sus componentes.

El sexto capítulo (6. *Implementación*), se presentan las características en cuanto a implementación de las distintas pantallas existentes en la aplicación, explicando los estándares utilizados y aspectos técnicos relacionados con el desarrollo.

El séptimo capítulo (*7. Validación*) se presentan los resultados y temas tratados durante las reuniones de validación del software junto al tutor de prácticas en la notaría.

El séptimo capítulo (8. *Testing*) se presentan los t test aplicados y sus resultados, además de un error especial que no pudo ser solventado.

El noveno capítulo (9. *Conclusiones y cierre*) el último con contenido implícito, presentamos las conclusiones, ideas futuras para la ampliación del software (futura versión 2) y por último cerramos el proyecto.

# 2 Marco tecnológico

En este capítulo se procede a explicar los detalles y conceptos técnicos que han influido en el devenir del proyecto, principalmente explicando el por qué se decidió elegir un dispositivo Desktop y no otros, qué herramientas software y hardware fueron necesarias para el desarrollo, diseño y la planificación del mismo y por último, el impacto que tuvieron estas decisiones.

## 2.1 Elección de plataforma

Partiendo de la idea propuesta por mi tutor de prácticas, disponía de dos opciones a la hora de implementar la aplicación.

La primera era hacer una aplicación para Android/Ios, la cual podría ser comercializada en las **Stores** [4] y producir cierto beneficio a la empresa en caso de que la aplicación fuera exitosa.

La segunda, podríamos hacer una aplicación Desktop con Windows Forms para poder simplificar la integración con un editor de textos y facilitar las tareas de mantenimiento, impresión y almacenaje de archivos.

Ambas opciones tenían sus pros y sus contras. A continuación las enumeran y explican en detalle:

**Desarrollo ios/android mobile:**

Listado de pros:

- Posibilidad de venta del producto en una AppStore.
- Facilidad de distribución del código mediante Stores.
- Posibilidad de desarrollar en lenguajes y entornos nuevos.
- Adquisición de nuevos conocimientos de interfaces mobile y patrones de diseño [5].

Listado de contras:

- Infraestructura muy limitada para poder desarrollar tanto en Ios como en Android.
- Posibles usuarios finales de la aplicación muy limitados en número, por lo que no habría beneficios.

- Se necesita una sincronización con datos de otro equipo, ya sea para imprimir, cargar documentos y/o almacenar los mismos.
- Excesivo tiempo de aprendizaje y de desarrollo.

**Desarrollo Desktop con Windows Forms:**

Listado de pros:

- Integración con un editor de textos.
- Infraestructura tecnológica suficiente para el desarrollo.
- Mayor flexibilidad a la hora del diseño de la interfaz de usuario.
- Ofrece la posibilidad de adaptar el código a posteriori y lanzar una versión mobile con otras características.
- Mejor documentación sobre los lenguajes de programación que se desean utilizar
- Posibilidad de desarrollar en lenguajes y utilizar nuevas tecnologías.
- Adquisición de nuevos conocimientos de interfaces desktop y patrones de diseño [6] [7].

Listado de contras:

- Necesidad de instalación.
- Necesidad de licencia de Microsoft Office
- Dificultad en la gestión de procesos mediante software.
- Pérdida de eventos con pantallas táctiles.

Una vez analizados los pros y las contras, nos decantamos por hacer un desarrollo para plataformas Desktop, debido a que el número de pros es superior al de la otra opción. Además hay que tener en cuenta que la multipantalla también es un valor añadido, ya que los trabajadores disponen de dos monitores, por lo que podrían personalizar a su gusto la disposición del programa en sus pantallas.

## 2.2 Tecnologías adoptadas

Una vez seleccionado y especificado el tipo de software y de plataforma que se quería obtener, especificamos las tecnologías adoptadas para el desarrollo.

Por motivos de infraestructura, se disponía de un equipo con Windows 8 Professional, al igual que todos los futuros usuarios del sistema, por lo que por compatibilidad y por facilidades de integración con Microsoft Word, el desarrollo se decidió hacer en Microsoft .NET **Framework** [8].

Debido a que son muchos los lenguajes de programación que se podían utilizar, se decidió implementarlo mediante vb.net, ya que la documentación sobre el lenguaje es muy extensa y fácil de seguir. Además, era un lenguaje muy potente que no había desarrollado con anterioridad.

Por último, se utilizó SQLite como sistema gestor de base de datos, ya que es portable, simple, de código libre y de características muy adecuadas para nuestra base de datos relacional de baja complejidad.

## *.NET Framework*

La plataforma .NET de Microsoft es un paquete software disponible para ser añadido al SO Windows.

Éste dispone de un gran número de soluciones predefinidas para las necesidades de programación y de integración de aplicaciones software.

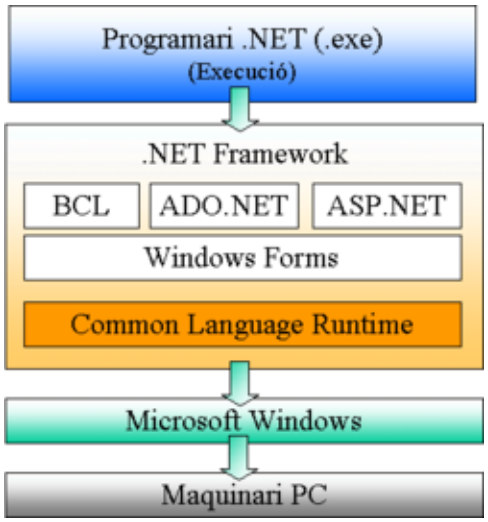


**Ilustración 1 - Arquitectura .NET Framework**

Los principales componentes del marco de trabajo son:

- El conjunto de lenguajes de programación.
- La biblioteca de clases base o **BCL**.
- El entorno común de ejecución para lenguajes, o **CLR** (Common Language Runtime) por sus siglas en inglés.

El marco de trabajo .NET soporta más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Este aspecto interesaba mucho, ya que permite desarrollar en distintos lenguajes de programación y crear librerías con una versatilidad que no proporcionan otros frameworks.

Por otra parte, al estar Microsoft office desarrollado con esta tecnología, la integración, basándonos en el **BCL**, es algo primordial para facilitar el desarrollo y asegurar un software de calidad, evitándonos utilizar librerías de código libre que son susceptibles a fallos.

Ya por último y no menos importante, disponemos de **LINQ**[9] para realizar las consultas de una manera más clara y de manera óptima. LINQ es uno de los muchos puntos fuertes de .NET, a través del cual se tiene la posibilidad de realizar consultas a nivel de código, ordenar listas u objetos en función de algún parámetro o incluso filtrar y controlar posibles errores.

## *VB.NET*

A la hora de elegir el lenguaje de programación que quería utilizar para el desarrollo del software, surgieron bastantes posibilidades, gracias a las características ya explicadas del Framework de .NET.

La elección fue desarrollar en VB.Net por distintos motivos:

- Es un lenguaje que desconocía y tenía la oportunidad de aprender uno nuevo y muy utilizado, aunque ahora en desuso.
- El lenguaje tiene muchas características ideales para desarrollos de **WebService**, que podrían ser utilizados en siguientes versiones del software cuando quiera hacerse multiplataforma.
- Existe una versión, *VBA*, integrada en las aplicaciones de Microsoft Office.
- La documentación disponible es extensa y fácil de encontrar.
- Es fácilmente extensible mediante *librerías DLL*.

## *SQLite*

Como **SGBD**, utilizamos SQLite, que es un sistema de gestión de base de datos relacional, compatible con *ACID* [10], y que tiene ciertas características que son por las cuales decidimos utilizarlo.

Sus principales ventajas son las siguientes:

- Es una *biblioteca* escrita en C de un tamaño de aproximadamente 300 Kb.
- El motor de SQLite no es un proceso independiente, sino que está integrado en el propio programa.
- Desde .NET se puede acceder usando el proyecto de código abierto System.Data.SQLite.
- Soporta hasta 2 TB de información.
- Varios procesos o hilos pueden acceder a la misma base de datos sin problemas.
- Coste 0

## 2.3 Herramientas utilizadas

Para el desarrollo del software y la planificación del mismo, se utilizaron diversas herramientas software para llevarlo a cabo.

### *Equipo*

PC sobremesa con Windows 8 Professional.

4 GigaBytes de Memoria.

Procesador Intel Core i5-4460S a 2.9 GHz.

Doble pantalla de 21 “

### *Software*

**Visual Studio 2015 Community**

Entorno de desarrollo utilizado para implementar el software. Dispone de muchas características, de las cuales destacamos el analizador de código y sus opciones y sistema de depuración de código.

**Microsoft Office 2010 Pro con licencia anual**

Paquete de software oficial de Windows que contiene Microsoft Word. El editor de texto por defecto con el que interactúa la aplicación.

### SQLite Manager

Es un **Addon**de Firefox para trabajar de manera sencilla con las bases de datos SQLite. Al ser una base de datos muy simple, sólo se utiliza para creación de consultas y para creación/eliminación de tablas y bases de datos.

### Adobe Photoshop CS5

Herramienta de diseño utilizada para crear el logo, botones, editar imágenes entre otros diseños.

### Diagramador Gliffy

Diagramador online, muy útil a la hora de hacer diagramas de flujo, **UML**, Entidad-Relación, etc.

## *Sistema de control de versiones*

### GIT

Es el sistema de control de versiones distribuido más utilizado del mundo. Se calcula que más del 30% del código del mundo está en un repositorio GIT. GIT [11] no modela ni almacena sus datos de este modo. En cambio, GIT modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en GIT, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, GIT no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.

Para el desarrollo de TmpMaker, sólo se necesitaban las funcionalidades básicas que nos proporciona, como son:

- Facilidad de trabajo mediante ramas
- Utilización de un repositorio local
- Merge automático de los cambios
- Simplicidad de control de versiones

# 3 Planificación

A la hora de comenzar con la planificación, ya se contaba con el factor temporal en contra, ya que se disponía de menos de dos meses para cubrir un amplio abanico de funcionalidades. Por esto, la idea de dividir el proyecto en dos fases y realizar dos versiones cobró fuerza.

La idea principal fue crear una versión inicial con la funcionalidad básica (fase 1) y luego ampliar la funcionalidad en la fase 2, fuera de las limitaciones temporales establecidas por el periodo de prácticas. Una vez comenzada la fase 2, nuevas ideas aparecieron y se llegó a plantear una futura fase 3. Por eso mismo, se tomó la decisión de seguir un modelo de ciclo de vida que nos permita ir lanzando versiones del software con el trascurrir del tiempo.

## 3.1 Ciclo de vida

Un ejemplo de ciclo de vida que nos sirve para el modelo de desarrollo que se pretendía seguir es el ciclo de vida incremental [12], que consiste en desarrollar por partes el producto software, y después integrarlas a medida que se completan.

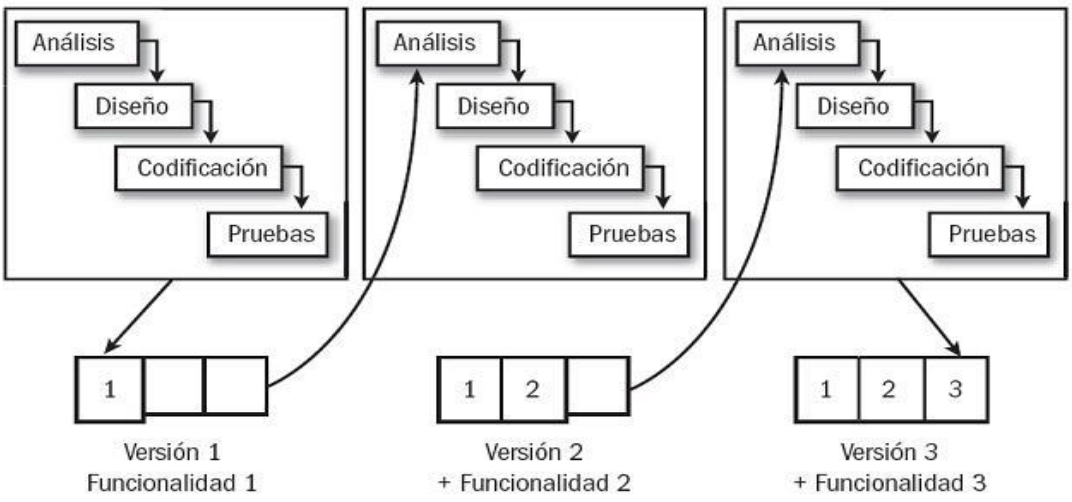


**Ilustración 2 - Modelo de ciclo de vida incremental**

El agregar cada vez más funcionalidad al sistema es exactamente lo que buscábamos, ya que de esta manera podíamos tener un software testeado y cubriendo unas funcionalidades mínimas en poco menos de un mes y medio; mientras que todas las ideas y funciones que no se puedan llevar a cabo, se pueden realizar modularmente en siguientes iteraciones, volviendo a realizar una planificación, análisis, diseño, implementación, pruebas, etc.

En primera instancia, la fase 1 está relacionada con todas las funcionalidades que permiten crear un documento a partir de fragmentos de texto y subdocumentos. La fase 2, añade funcionalidades más diversas:

- Pantalla inicial y gestión de tipos de documento
- Tratamiento de imágenes
- Cambio de idioma
- Cambio de ruta de almacenamiento
- Numeración de páginas
- Optimizaciones y mejoras en cuanto a usabilidad

Por último, la fase 3, implica la adaptación de la aplicación para las plataformas IOS y Android.

En esta memoria, se presentan los resultados de la fase 1 y 2, dejando la fase 3 como un trabajo futuro.

A continuación podemos ver el **Diagrama de Gantt** mostrando la planificación temporal grosso modo del proyecto, entrando luego en detalle en cada fase y especificando las tareas más relevantes de cada elemento.

## 3.2 Tareas

A continuación presentamos las tareas llevadas a cabo en cada fase, identificada cada una de las mismas con un nombre único que corresponde a la siguiente especificación:

- T → Tarea
- Numeración: “Número de fase o ciclo”. ”Número de subfase”. ”Número de tarea (letra en orden A-Z)”.
- Si el número es un carácter ‘X’, significa que se aplica en todas las fases y/o subfases.

Ejemplos: TF.1.2.A o  TF.X.1.B (TF.1.1.B y TF.2.1.B)

| Identificador | Tarea | Relacionada con |
|---|---|---|
| T.X.1.1.A | Identificación de usuarios y roles | |
| T.X.1.1.B | Captura y análisis de requisitos de usuario | TF.X.1.1.A |
| T.X.1.1.C | Captura y análisis de requisitos de sistema | TF.X.1.1.B |
| T.X.1.1.D | Gestión de riesgos | |
| T.X.1.2.A | Descarga del software necesario | |
| T.X.1.1.B | Instalación del software y activación de licencias | TF.X.1.2.A |
| T.X.1.1.C | Configuración del software | TF.X.1.1.B |
| T.1.2.1.A | Diseño logo y selección de colores base | |
| T.X.2.1.B | Diseño de las maquetas de pantallas | |
| T.1.2.1.C | Diseño de botones, listados y fondos | TF.X.2.1.A, |

| | | |
|---|---|---|
| | | TF.X.2.1.B |
| T.X.2.2.A | Identificación de clases necesarias | |
| T.X.2.2.B | Diseño del diagrama Entidad/Relación | TF.X.2.2.A |
| T.X.3.1.A | Crear las bases de datos | TF.X.2.2.B |
| T.X.3.1.B | Implementar controlador con las consultas **SQL** necesarias | TF.X.3.1.A |
| T.X.3.1.C | Diseñar las pantallas sin funcionalidad | TF.X.2.1.B |
| T.X.3.1.D | Cargar las **BBDD** con datos para poder desarrollar | TF.X.3.1.A |
| T.1.3.1.E | Cargar las pestañas con las secciones | TF.X.3.1.D,<br>TF.X.3.1.C |
| T.1.3.1.F | Cargar las secciones con las plantillas | TF.X.3.1.D,<br>TF.X.3.1.C |
| T.1.3.1.G | Sincronizar el software con Microsoft Word | |
| T.1.3.1.H | Permitir mover texto entre las aplicaciones | TF.1.3.1.G |
| T.1.3.1.I | Permitir el almacenamiento de los datos (**BBDD** y disco) | |
| T.1.3.1.J | Carga de imágenes redimencionables y documentos enteros | |
| T.1.3.1.K | Tratamiento de fechas y números según formato notarial | TF.1.3.1.H |
| T.1.3.1.L | Encabezados y pies de páginas como secciones aparte | TF.1.3.1.E,<br>TF.1.3.1.F |
| T.1.3.1.M | Ordenación y filtrado de elementos | TF.1.3.1.F |

| | | |
|---|---|---|
| T.1.3.1.N | Edición del formato de texto dentro de la aplicación. | TF.1.3.1.H |
| T.1.3.2.A | Validar diseño y funcionalidad con Armando Mazaira | |
| T.1.3.2.B | Reajustar cambios que puedan surgir | TF.1.3.2.A |
| T.X.3.3.A | Realizar tests que cubran todas las funcionalidades | |
| T.2.3.1.E | Almacenamiento de tipos de documentos | TF.X.3.1.B |
| T.2.3.1.F | Control del paso de pantalla de inicio a sistema de gestión de plantillas | TF.X.3.1.B, TF.X.3.1.C |
| T.2.3.1.G | Configuración por idioma y ruta de almacenamiento | |
| T.1.4.1.A | Presentar el resultado final al equipo | TF.X.3.3.A |
| T.1.4.1.B | Realizar una formación a los trabajadores | TF.1.4.1.A |
| T.1.4.1.C | Instalar el software en las máquinas necesarias | |

**Tabla 1 - Listado de tareas y sus relaciones**

## 3.3 Planificación temporal

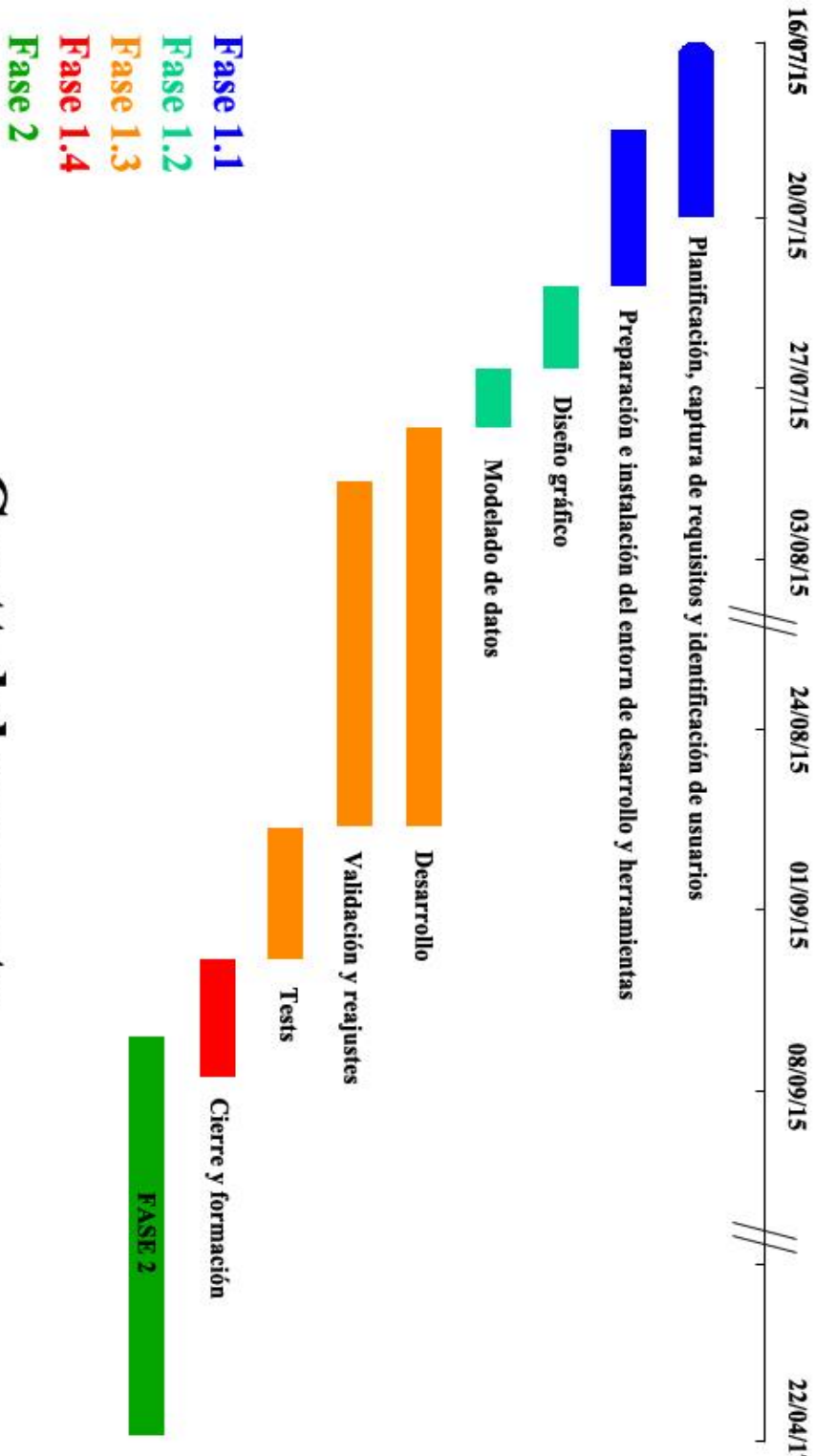


**Ilustración 3 - Diagrama de gantt del proyecto**

# 3.4 Riesgos

Una vez presentadas las tareas y su disposición temporal, se procese a analizar los riesgos que existen en nuestro proyecto, sólo focalizándose en los más posibles e importantes.

## *3.4.1 Identificación y análisis de riesgos*

A continuación podemos ver los riesgos en una tabla, en la cual se distinguen cuatro campos.

- Id: Identificador único del riesgo
- Riesgo: Nombre descriptivo del riesgo en cuestión.
- Probabilidad: Valor decimal comprendido entre los valores 0 y 1 (donde 0 es que no existe probabilidad y 1 significa que es un hecho seguro) indica una estimación de la probabilidad de que ocurra el acontecimiento descrito en el campo anterior.
- Impacto: Decimal entre los valores 0 y 1 (donde 0 es que el impacto es mínimo y 1 sería el máximo) indica una estimación del impacto que tendría sobre el proyecto la materialización de un riesgo.

Por otra parte, los riesgos están priorizados por el rango probabilidad/impacto, resultante de multiplicar ambos valores entre sí.

| Id | Riesgo | Probabilidad | Impacto | P-I |
|---|---|---|---|---|
| R1 | Imposibilidad de cumplir con los plazos establecidos de entrega. | 0,2 | 0,9 | 0.18 |
| R2 | Caducidad de licencias para el desarrollo en futuras fases | 0,3 | 0,3 | 0.09 |
| R3 | Cambios de funcionalidades en la fase de validación de la implementación. | 0,5 | 0,15 | 0.075 |

| | | | | |
|---|---|---|---|---|
| R4 | Cancelación del proyecto por parte del notario. | 0,08 | 0,9 | 0.072 |
| R5 | Cambios en el Framework de .NET y de las librerías de Microsoft.Office que puedan influir en el código. | 0,1 | 0,3 | 0.03 |
| R6 | Cambio de editor de textos por parte de la notaría. | 0,02 | 0,8 | 0.016 |

**Tabla 2 – Tabla de riesgos priorizados**

Los riesgos que más tendremos en cuenta son R1, R2, R3, R4 y R6, ya que el impacto del R5, es demasiado bajo como para priorizarlo por encima de los otros. Sin embargo, elaboramos un plan de acción para cada uno de los riesgos, con el fin de saber cómo actuar en el caso de que cualquiera de los cinco se materialice.

### *3.4.2 Plan de respuesta ante riesgos*

A continuación se presenta, para cada uno de los riesgos, una serie de acciones a llevar a cabo. Ante la imposibilidad de aplicar la primer acción, se presenta la siguiente como una alternativa válida.

**R1 - Imposibilidad de cumplir con los plazos establecidos de entrega.**

Plan de acción:

Acordar de palabra con Armando Mazaira, tutor de prácticas en la empresa, que se seguiría con el desarrollo y se presentaría el software con todas las funcionalidades al ser acabado, aunque se haya superado el plazo de entrega.

Alternativa:

Dejar una versión lo suficientemente estable, con las funcionalidades prioritarias y realizar la formación online a los trabajadores.

**R2 - Caducidad de licencias para el desarrollo en futuras fases**

Plan de acción:

Cambiar de software a uno de código libre y/o utilizar una licencia de prueba básica en el caso de que la misma nos proporcione las funcionalidades que necesitamos para continuar con el desarrollo.

Alternativa:

Adquirir la licencia personal del software, aunque implique un coste extra no planificado.

**R3 – Cambios de funcionalidades en la fase de validación de la implementación.**

Plan de acción:

Rechazar los cambios que impliquen una reestructuración elevada, implicando estas más de 5 horas de desarrollo. Éstas, se realizarían en una fase posterior a la actual.

Por otra parte, los cambios que sean relativamente simples o pequeños, se pueden realizar siempre y cuando haya habido un análisis y un diseño (si aplica) previamente.

Alternativa:

Volver a comenzar el ciclo, descartando todo lo que tenga que ser cambiado y volviendo a la fase de planificación y análisis con el fin de abarcar la funcionalidad desde la otra perspectiva.

Cabe destacar que hay que a hacer un análisis y planificación temporal, ya que se puede ver gravemente afectado por estos cambios.

**R4 - Cancelación del proyecto por parte del notario.**

Plan de acción:

Continuar el proyecto con las limitaciones que implica la pérdida de licencias y del equipo, pero fuera de la notaría.

Esto implica que no haya limitaciones en el rango temporal, al pasar a ser un proyecto propio y la posibilidad de amplificar las funcionalidades que se realizan en la fase 1.

Alternativa:

Cancelar el proyecto y comenzar uno nuevo, marcado o no, por la empresa en dónde esté en ese momento.

**R5 - Cambios en el Framework de .NET y de las librerías de Microsoft.Office que puedan influir en el código**

Plan de acción:

Actualizar el código y aplicar los cambios pertinentes con el fin de seguir asegurando que todas las funcionalidades descritas en el documento de requisitos sigan cumpliéndose.

Para esto puede surgir la necesidad de gestionar los paquetes y librerías y definir un sistema de control de versiones de las mismas, además de documentar los cambios que han surgido y afectado a nuestro proyecto.

Alternativa:

Utilizar otras librerías que nos proporcione .NET y aplicar los cambios pertinentes en el código.

**R6 - Cambio de editor de textos por parte de la notaría.**

Plan de acción:

Debido a que implicaría una reestructuración de casi el 80% del programa, continuar con el editor definido al principio, con la posibilidad de añadir nuevas características en una fase o versión posterior.

Esto implicaría que el software dejaría de ser útil para la notaría, salvo que mantuvieran el editor de textos anterior y en nuevo simultáneamente.

Alternativa:

Volver a comenzar el proyecto reutilizando todo lo posible como puede ser la arquitectura y el diseño de las pantallas.

Esto implica volver a iniciar la Fase 1 y analizar las funcionalidades que podríamos cumplir y las que no.

# 4 Análisis

Después de haber estudiado los objetivos, los detalles técnicos y la planificación del proyecto se procede a la identificación de usuarios y el análisis de requisitos, tanto de usuario como de sistema.

## 4.1 Identificación de usuarios

Partiendo de ciertas condiciones e ideas a la hora de definir el alcance del proyecto surge una cuestión muy importante. ¿Qué usuarios se pueden beneficiar de este software?

Está claro que hay un usuario principal, que son los propios trabajadores de la notaría que encomendó el proyecto, pero ¿podemos ampliar el abanico de usuarios y expandir la aplicación a otros entornos de trabajo o hasta el ámbito personal?

La respuesta es sí. Obviamente, las funcionalidades desarrolladas en la Fase 1 se pensaron sólo para usuarios de notarías. Pero a partir de la Fase 2, se añadieron las nuevas funcionalidades, que pueden ser interesantes para otro tipo de usuarios, con otro tipo de objetivos o propósitos.

Por lo tanto, sacando el proyecto del ámbito notarial, podemos encontrar los siguientes usuarios. Eso sí, sin necesidad de cambio de permisos y/o identificación de roles.

**Listado de usuarios potencialmente atractivos:**

| Usuario | Ejemplo de utilidad |
|---|---|
| Administrativos/as | Elaboración de albaranes y registros de compra-venta. |
| Estudiantes | Apuntes, trabajos, planes de estudio. |
| Empresarios/as | Elaboración de presupuestos o Faxes. |

| | |
|---|---|
| Escritores/as | Crear un libro basado en una estructura. |
| Secretarios/as | Envío de correos electrónicos. |
| Profesores/as | Elaboración de exámenes y actas docentes. |

**Tabla 3 - Listado de usuarios y posibles usos de la aplicación**

## 4.2 Requisitos de usuario

A continuación se dispone de una tabla con todos los requisitos de usuario obtenidos tras una serie de reuniones con el principal interesado y sus empleados.

Debido a que el proyecto se divide en distintas fases, se añade una columna que indica la los requisitos según la iteración/fase en la que fueron analizados.

| Id | Requisito | Fase |
|---|---|---|
| RQ1 | El usuario he de poder crear y documentos. | 1 |
| RQ2 | El usuario he de poder crear zonas o secciones dentro de un documento. | 1 |
| RQ3 | El sistema ha de permitir crear, modificar y borrar plantillas de secciones de un documento. | 1 |
| RQ4 | Un usuario ha de poder controlar donde guardar los documentos y plantillas. | 2 |
| RQ5 | El usuario he de poder personalizar encabezados y pies de página con un mínimo de estilo. | 1 |
| RQ6 | El usuario debe poder encontrar de manera fácil las plantillas y documentos creados. | 1-2 |
| RQ7 | El usuario he de poder añadir imágenes y controlar su tamaño y proporción. | 2 |

| | | |
|---|---|---|
| RQ8 | El usuario ha de poder añadir a un documento el contenido de otro documento ya existente, sin necesidad de almacenarlo como plantilla. | 1 |
| RQ9 | El usuario ha de poder introducir una fecha numérica y que sea convertida en formato notarial | 1 |
| RQ10 | El usuario ha de poder introducir un número y convertirlo en formato notarial | 1 |
| RQ11 | El usuario ha de poder elegir idioma | 2 |
| RQ12 | El usuario he de poder crear tipos de documentos. | 2 |

**Tabla 4 - Listado de requisitos de usuario**

## 4.3 Requisitos de sistema

Una vez detallados todos los requisitos de usuario obtenidos en varias reuniones con los usuarios de la notaría, los cuales son genéricos y detallados en lenguaje natural, se procede a especificar un listado de requisitos de sistema, ya con una estructura fija y un lenguaje más técnico, aunque comprensible por las personas técnicas que formen parte del proyecto.

Los requisitos los presentamos divididos en funcionales / no funcionales y a su vez entre fase 1 y fase 2.

Además, disponemos de una tabla en donde priorizamos los requisitos en cuanto a relevancia, con un número del 0 al 2, en donde 0 implica que es un requisito obligatorio/crítico, mientras que el 2 es un requisito de baja relevancia sobre el resultado final.

### *4.3.1 Funcionales*

| Id | Requisitos funcionales | Fase | Prioridad |
|---|---|---|---|
| RS1 | El usuario ha de poder diferenciar tres tipos elementos: documentos, secciones y plantillas | 1 | 0 |
| RS2 | El usuario ha de poder asociar n secciones a un documento, sin | 1 | 0 |

| | | | |
|---|---|---|---|
| | duplicidad de nombres. | | |
| RS3 | El usuario ha de poder asociar n plantillas a una sección, sin duplicidad de nombres. | 1 | 0 |
| RS4 | El usuario ha de poder borrar y actualizar documentos, secciones y plantillas. | 1 | 0 |
| RS5 | El usuario ha de poder sobreescribir el contenido de una plantilla. | 1 | 0 |
| RS6 | El usuario ha de poder convertir una fecha seleccionada en el editor, al formato notarial (texto en mayúsculas) | 1 | 0 |
| RS7 | El usuario ha de poder convertir una número seleccionado en el editor, al formato notarial (texto en mayúsculas) | 1 | 0 |
| RS8 | El usuario ha de poder asociar un tipo 'documento' a un documento .docx de microsoft word. | 1 | 1 |
| RS9 | El usuario ha de disponer de la base de datos creada y cargada en la primera ejecución de TmpMaker. | 1 | 1 |
| RS10 | El usuario ha de poder mover el contenido de una plantilla al editor de texto con un solo click | 1 | 1 |
| RS11 | El usuario ha de poder mover el contenido de una plantilla a una caja de texto donde se pueda formatear y modificar. | 1 | 1 |
| RS12 | El usuario ha de poder copiar el contenido de otro documento al documento abierto. | 1 | 1 |
| RS13 | El sistema debe tratar los encabezados y pies de página como unas secciones prefijadas y siempre disponibles. | 1 | 1 |
| RS14 | El usuario ha de poder copiar el contenido de una plantilla en el punto donde se sitúa el cursor en el editor de texto. | 1 | 1 |
| RS15 | El usuario ha de poder ordenar las secciones mediante una modal y ver los cambios dinámicamente. | 1 | 2 |

| | | | |
|---|---|---|---|
| RS16 | El usuario ha de poder almacenar las fechas de creación, y última modificación de documentos, secciones y plantillas. | 1 | 2 |
| RS17 | El usuario ha de poder cargar documentos mediante el explorador de windows. | 1 | 2 |
| RS18 | El usuario ha de poder ver una previsualización del contenido de una plantilla ya sea en un **Tooltip** o en un panel. | 2 | 1 |
| RS19 | El usuario ha de poder gestionar tipos de documentos en la pantalla inicial: creación, modificación y edición. | 2 | 2 |
| RS20 | El usuario ha de poder abrir un documento en el editor de texto | 2 | 2 |
| RS21 | El usuario ha de poder cambiar el idioma de la aplicación y la ruta de almacenamiento de los ficheros. | 2 | 2 |
| RS22 | El usuario ha de poder añadir numeración de página tanto en el encabezado como en el pie de página. | 2 | 2 |
| RS23 | El usuario ha de poder mover un texto seleccionado en el editor, a la caja de texto de edición del propio programa. | 2 | 2 |
| RS24 | El usuario ha de poder cargar imágenes y redimencionarlas mediante una modal y disponer de un panel de previsualización. | 2 | 2 |
| RS25 | El usuario ha de poder utilizar un área de acciones rápidas que se aplican sobre el propio editor. Este área ha de incluir las conversiones en formato notarial. | 2 | 2 |
| RS26 | El usuario ha de poder filtrar las plantillas por nombre, para facilitar la localización de las mismas. | 2 | 2 |
| RS27 | El usuario ha de poder añadir una descripción a los documentos para facilitar la distinción de los mismos. | 2 | 2 |

**Tabla 5 - Listado de requisitos funcionales priorizados**

## 4.3.2 No funcionales

| Id | Requisitos de sistema no funcionales | Fase | Prioridad |
|---|---|---|---|
| RS28 | El sistema debe asegurar una consistencia de datos el 100% del tiempo, utilizando una base de datos portable y local. | 1 | 2 |
| RS29 | El sistema debe poder instalarse en equipos Windows y utilizar los procesos del sistema de manera óptima, cerrando todos los procesos al finalizar una ejecución. | 1 | 0 |
| RS30 | El sistema debe proporcionar visores adecuados para que el usuario seleccione plantillas, secciones y documentos. | 1 | 0 |
| RS31 | El sistema debe notificar un error de sistema mediante un diálogo. | 1 | 1 |
| RS32 | El sistema no puede tardar más de 5 segundos en la carga inicial (en conjunto con el editor de texto). | 2 | 2 |

**Tabla 6 - Listado de requisitos de sistema no funcionales priorizados**

# 5 Diseño

A continuación, en el siguiente capítulo, se dispone a presentar los modelos diseñados, tanto los modelos de base de datos relacionales y las estimaciones de uso de la misma, como la interfaz gráfica de usuario.

Se puede ver un modelo entidad-relación junto a una especificación de las tablas creadas y el diseño pantalla a pantalla.

## 5.1 Modelo de datos

A continuación vemos el diseño y modelado de los datos en la base de datos relacional que hemos utilizado para almacenar la información necesaria para el correcto funcionamiento del software. En la misma, almacenamos tanto la configuración del sistema como si se tratase de una **Cookie** o la información de los documentos y plantillas y sus asociaciones, para formar dinámicamente las pestañas y botones en las pantallas diseñadas.

### *5.1.1 Tablas y relaciones*

A continuación presentamos las entidades/clases necesarias para implementar toda la funcionalidad asociada con nuestra base de datos relacional.

| | |
|---|---|
| ***Configuration*** | Contiene el lenguaje y el **Path** final donde almacenar los datos. También contendrá los futuros campos de configuración, como tamaño de pantallas, posición, etc. |
| ***Document*** | Datos de los tipos de documentos. Clase de mayor nivel, como pueden ser email, albarán, factura, entre otros. No hay que confundirlo con el |

| | |
|---|---|
| | documento Word que se editará. |
| ***Section*** | Segmentos en los que dividimos el documento. Completamente abstractos. Pueden ser desde párrafos a Títulos, tablas, índices, capítulos o subdocumentos. Asociados a un único documento |
| ***Template*** | Cada plantilla o subdocumento asociado a una section. Cada una tiene un nombre y una descripción para diferenciarlas. No se puede repetir el nombre. |
| ***HeaderFooter*** | Son templates asociados a dos tipos de sections predefinidos: encabezados y pies de página.<br><br>Son compartidos por todos los documentos y tienen las otras características de los templates. |

**Tabla 7 - Listado de entidades de la BBDD**

### 5.1.2 *Modelo Entidad/Relación*

A continuación se presenta el modelo entidad/relación, del cual se pueden ver cada uno de los atributos de las clases mencionadas anteriormente.

Del modelo se puede destacar la similitud entre una sección y un encabezado o pie de página, ya que sustancialmente son iguales, aunque los segundos son se pueden compartir por todos los documentos y no necesitan una relación directa.

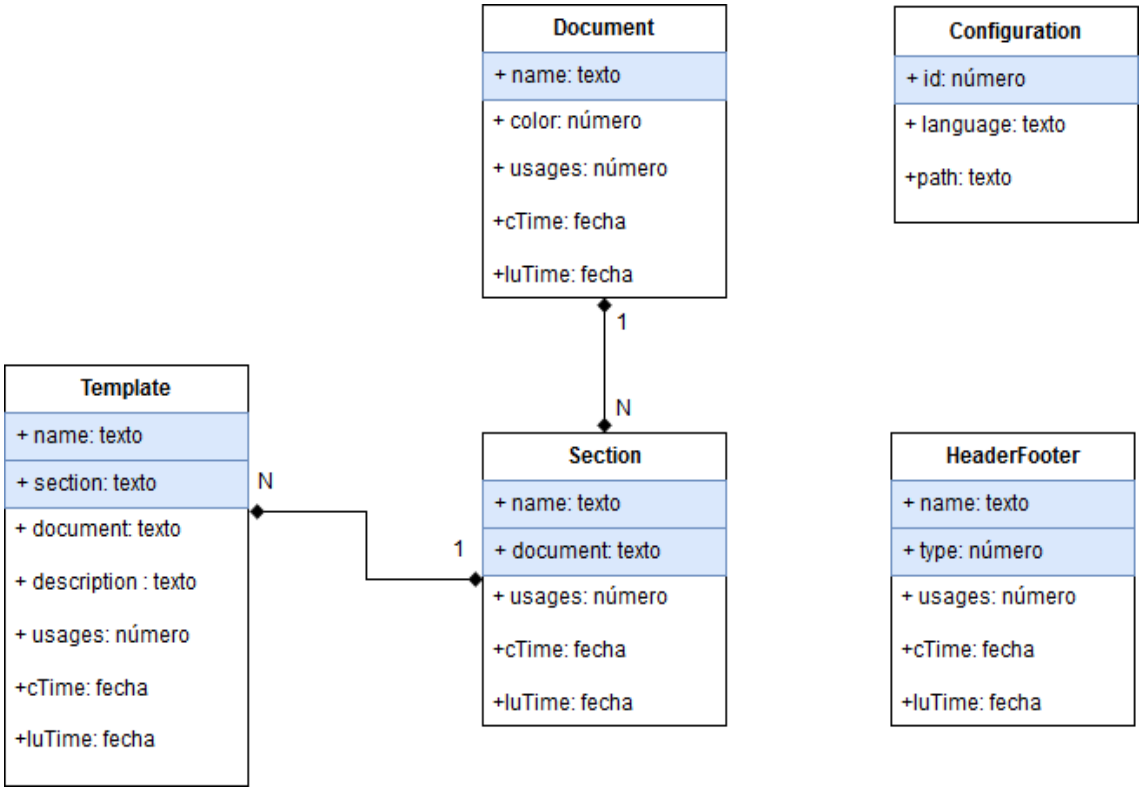


**Ilustración 4 - Modelo entidad/relación de al BBDD**

## 5.1.3 *Volúmenes de datos*

Debido a que la base de datos es un fichero en la propia máquina cliente, los volúmenes de datos no deberían ser superiores a los 10 Gb de media. De todas maneras, las tablas almacenan muy poca información y no incluyen los textos almacenados ni los formatos de los documentos y/o plantillas. Se estima que la base de datos no supere los 3 MB, por lo que no presenta ningún problema en cuanto a volumen.

Por otra parte, al estar en entorno local, no habrá accesos concurrentes, por lo que las conexiones no deberían ser ningún problema mientras se vayan cerrando a medida que se utilizan. En cambio, hay una problemática. ¿Dónde almacenamos los datos, textos, imágenes, tablas y sus formatos si no es en la base de datos?

La solución adoptada es guardarlo en archivos .docx independientes, respetando los nombres de las plantillas e impidiendo duplicados.

De esta manera, un usuario puede utilizar una plantilla como documento y así aseguramos una alta escalabilidad.

## 5.2 Interfaz gráfica de usuario

Para el diseño de la interfaz gráfica de usuario se decidió seguir un modelo cíclico propio, basado en buenas prácticas de proyectos anteriores.

Por limitaciones y características del proyecto, se tuvo que adaptar el modelo a seguir, el cual se caracteriza por seguir las siguientes cuatro etapas:

- Diseño: Se definen estándares, formatos, colores, márgenes y todos los elementos de diseño de una vista.
- Validación: Otra persona con un perfil de diseñador dentro del proyecto debe validar el diseño para proceder con el prototipado. En nuestro caso particular, sólo había un perfil, por lo que esa fase no la teníamos en cuenta y se realizó una autovalidación al hacer el diseño.
- Prototipado: **Low-Fidelity** de las páginas utilizando los elementos y formatos diseñados en la etapa 1 (Diseño).
- Evaluación: Validación y evaluación del prototipo, en este proyecto en particular, por Armando Mazaira durante la Fase 1.
  En caso de no aprobarse, se vuelve a la etapa pertinente.

### *5.2.1 Elementos comunes de la GUI*

Con la intención de presentar una aplicación con *Windows forms*, que asegure un cierto grado de usabilidad, seguimos unos estándares de diseño basados en las aplicaciones de Windows.

Diferenciamos nuestras pantallas en dos tipos visualmente distintos:

**Pantalla principal**: Gestión de tipos de documento

- ✓ Diseño colorido y atractivo en cuanto a visibilidad.
- ✓ Funcionalidad sencilla y limitada.
- ✓ Pantalla única. No necesita compartir espacio.

**Pantalla de edición:** Gestión de plantillas y documentos

- ✓ Diseño con estilo Windows Forms.
- ✓ Funcionalidad compleja.

- ✓ Espacio limitado por ser compartido con la ventana de Word.

Por otra parte, los diálogos siguen el mismo formato del tipo de pantalla que los levanta.

Debido a que todas las pantallas son una herramienta que usa el usuario para utilizar y experimentar todas las funcionalidades que ofrece el sistema, cada una de ellas debe resolver al menos una de todas las funcionalidades que realiza en su conjunto el sistema.

Los elementos comunes entre pantallas la pantalla principal y la de edición son los siguientes:

1. Menú
2. Logo
3. Zona de contenido
4. Zona de elementos de acción

## Colores base

Los colores seleccionados que utiliza nuestra aplicación en sus elementos no dinámicos o personalizables por el usuario son los siguientes:

**Ilustración 5 - Colores primarios de TmpMaker**

Por otra parte, en los elementos dinámicos se utilizan otros colores y pueden ser personalizados por el usuario.

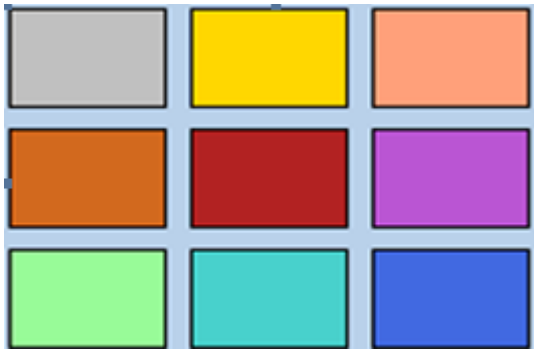

**Ilustración 6 - Colores secundarios de TmpMaker**

**Controles y elementos utilizados**

En las distintas pantallas y escenarios posibles en TmpMaker, se utilizan diversos tipos de elementos gráficos para representar la información.

Los elementos están incluidos en la **Librería** `System.Windows.Forms` [13] de .NET y todos utilizan un *FlowLayout* como contenedor base:

- ✓ Panel
- ✓ TabControl
- ✓ FontDialog
- ✓ ColorDialog
- ✓ OpenFileDialog
- ✓ MenuStrip
- ✓ ToolStripMenuItem
- ✓ TextBox
- ✓ RichTextBox
- ✓ Buttons
- ✓ Labels
- ✓ ComboBox
- ✓ RadioButton
- ✓ PictureBox
- ✓ ListBox

## *5.2.2 Prototipado Low-Fidelity*

A continuación, se presenta el diseño realizado de todas las pantallas, en las cuales se numeran y describen los elementos que incluye.

El formato en que presentamos la información es el siguiente:

- ➢ Título
- ➢ Descripción de la pantalla, paneles y elementos.
- ➢ Elementos utilizados
- ➢ Requisitos relacionados
- ➢ Imagen

## *Gestión de tipos de documentos*

Es la pantalla principal, donde mostramos el logo y el título del software en la cabecera.

El contenido lo dividimos en dos paneles, siendo el izquierdo un panel donde se añaden los tipos de documentos en formato botón ordenados de izquierda a derecha y de arriba a abajo.

Por último, el panel de la derecha muestra las distintas acciones posibles que tiene la pantalla principal.

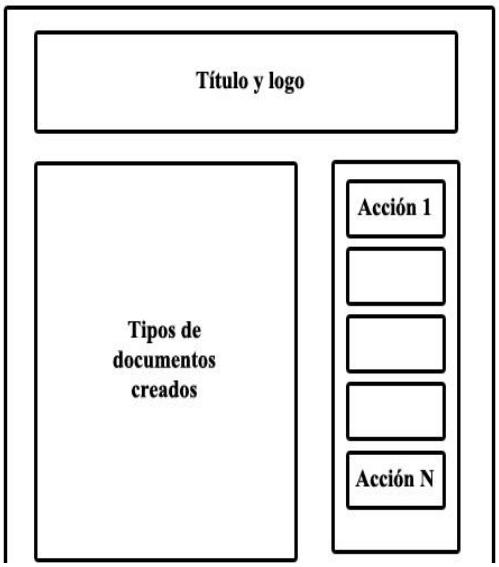


**Ilustración 7 - Diseño de la Pantalla de inicio**

*Elementos*

- ✓ FlowLayout
- ✓ Panel
- ✓ Button
- ✓ Label
- ✓ PictureBox

## *Creación de un tipo de documento*

En la siguiente ventana donde podemos crear un tipo de documento la dividimos en 4 contenedores.

En el superior (“Input nombre”), se incluye un label y un textbox.

En el siguiente, dispondremos de nueve botones con colores de fondo distintos, que representan el color del “Tipo de documento”, para poder diferenciarlos a simple vista en los paneles.

En cuanto al panel “Descripción”, contiene un textbox para diferenciar los tipos y añadir una descripción.

Por último tenemos los botones de acción guardar y cancelar.

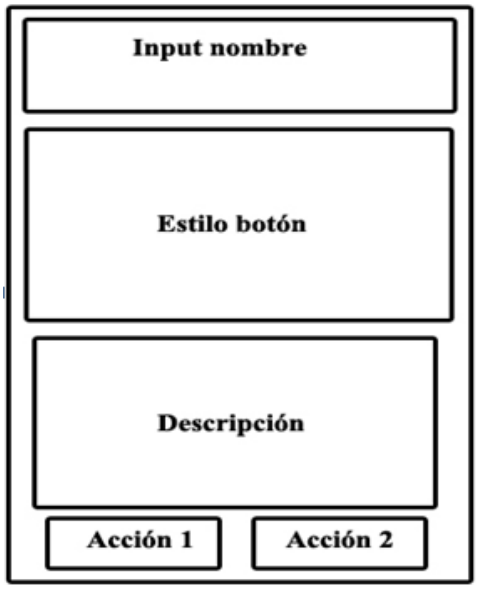


**Ilustración 8 - Diseño de pantalla de creación de tipo**

*Elementos*

- ✓ Panels
- ✓ Textbox
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout

## *Edición de tipos de documento*

En esta pantalla, se puede editar todos los tipos de documentos existentes. Básicamente, se divide la ventana en dos zonas. La izquierda es igual que el panel de botones de la ventana principal (botones ordenados de izquierda a derecha y de arriba a abajo) aunque con una pequeña diferencia. Al clickar se quedan seleccionados, es decir, funcionan como los radiobuttons.

En la parte de la derecha, tenemos todas las acciones previsualizables (1, 2 y 3) y finales (4 y 5).

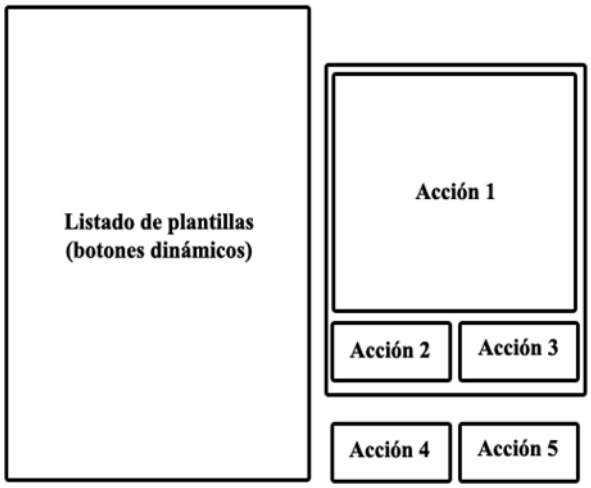


**Ilustración 9 -Diseño de pantalla de edición de tipos**

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout

## *Configuración aplicación*

La configuración en las primeras dos versiones del software es muy básica, por lo que la pantalla no es muy compleja. Sólo presenta un listado de botones en un panel y un textbox y un dialog explorador de ficheros y directorios, además de los botones de acción final (Guardar y Cancelar)

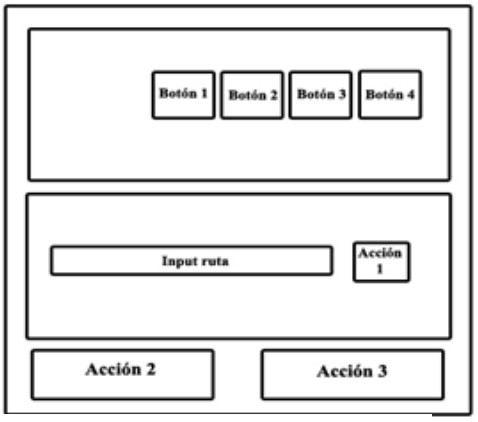


**Ilustración 10 - Diseño de pantalla de configuración**

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout
- ✓ FolderBrowserDialog
- ✓ TextBox

## *Pantalla completa de edición de documentos*

A continuación se presenta cómo se dispondría la pantalla del usuario en el caso de que utilizase un único monitor.

Por simplicidad, el usuario debe poder manejar el editor de texto y TmpMaker simultáneamente, por lo que se divide la pantalla en dos zonas. La zona izquierda es dedicada al editor de texto y ocupa un 65% de la pantalla.

Por otra parte, el 35 % restante es dedicado a la aplicación en sí, que se decidió tenerlo a la derecha, por motivos relacionados con la lectura de izquierda a derecha en el mundo occidental.

La ventana no es redimencionable pero si movible. Mientras que Windows se encarga de almacenar las preferencias de posicionamiento de ventanas.

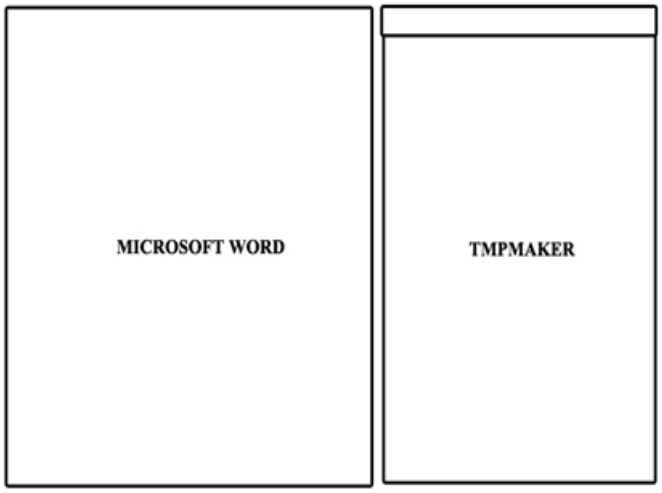


**Ilustración 11 - Diseño a pantalla completa**

*Elementos*

- ✓ Windows forms
- ✓ Editor de texto

## *Pantalla de edición de documentos (I)*

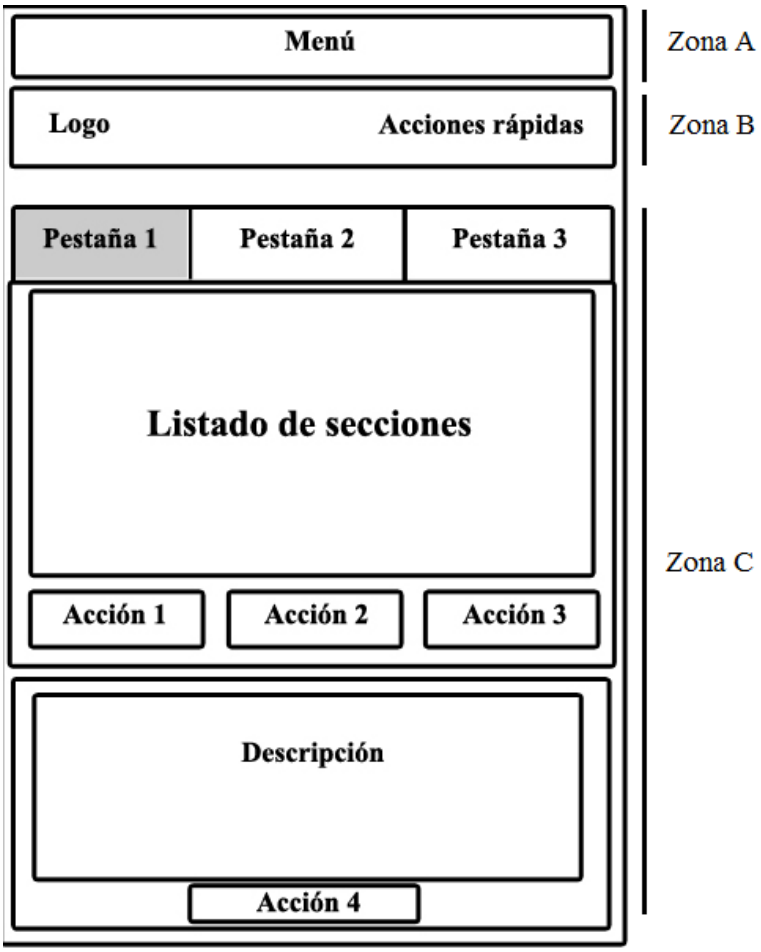


**Ilustración 12 - Diseño de pantalla de edición de documentos- Pestaña 1**

La siguiente pantalla se abre simultáneamente con Word y corresponde al panel derecho de la pantalla anterior (Pantalla completa de edición de documentos).

Aquí disponemos de más elementos y zonas que las pantallas anteriores, ya que abarca mucha más funcionalidad.

Comenzando por el menú (Zona A), es un elemento estático y contiene acciones que abren otras ventanas, ya sea para la inserción de documentos, imágenes, sección de ayuda, etc. Cada componente del menú puede tener submenús de manera recursiva.

Por otra parte, más abajo tenemos el panel donde se pinta el logo y un listado de botones (Zona B), alineados a la derecha, en la zona de acciones rápidas, que interactúan directamente con el editor de texto.

Después, en la Zona C separamos el contenido en pestañas o Tabs. La pestaña 1 es estática (todos los documentos la tendrán) e incluye el contenido del documento, separado en dos paneles:

- El primer panel muestra las secciones definidas por el usuario/usuarios ordenadas de arriba abajo y de izquierda a derecha, junto a sus botones de acción para gestionar las mismas.
- El segundo panel nos sirve para modificar y ver la descripción de los documentos (por defecto deshabilitado). Para modificarlo hay que utilizar los botones de acción.

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout
- ✓ FolderBrowserDialog
- ✓ PictureBox
- ✓ GroupBox
- ✓ TabControl
- ✓ TabPage

## *Pantalla de edición de documentos (II)*

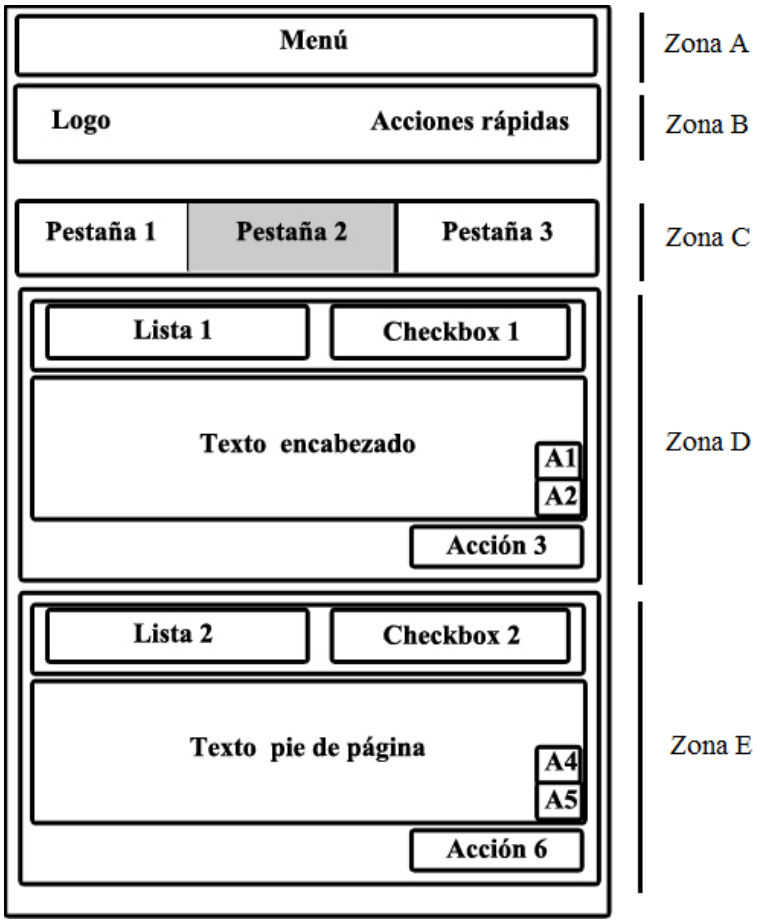


**Ilustración 13 - Diseño de pantalla de edición de documentos- Pestaña 2**

En esta pantalla, las zonas A y B no varían respecto a la anterior. Lo mismo pasa con las pestañas de la zona C, en la que sólo varía la pestaña seleccionada, que pasa a ser la "Pestaña 2"

La pestaña 2 siempre será la segunda y corresponde con un tipo especial de secciones del documento. Aquí se puede editar y gestionar en su totalidad los distintos encabezados y pies de página.

Al crear un tipo de documento nuevo, el usuario verá sólo dos pestañas. La del documento (Pestaña 1) y la de los encabezados y pies de página (Pestaña 2).

Básicamente, la gestión de encabezados y pies de página es igual, aunque se duplican los componentes para no llevar al usuario a confusión y facilitar los cambios simultáneos.

La zona D, contiene el panel de gestión de los encabezados, donde se incluye de una lista en donde encontramos todos los encabezados cargados en nuestra app (Lista 2) y un checkbox

para añadir el número de página (Checkbox 1). Por otra parte, se incluye un TextBox (Texto encabezado) en donde se puede crear el encabezado y/o modificar existentes.

Ya por último, se pueden ver distintos botones de acción alineados a la derecha (A1, A2, Acción 3) para cumplir con ciertas funcionalidades como almacenar los datos, sincronizarlos con el editor, entre otras.

La zona E es exactamente igual, con la diferencia de que tratamos pies de páginas en lugar de encabezados. La funcionalidad es la misma.

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout
- ✓ FolderBrowserDialog
- ✓ PictureBox
- ✓ GroupBox
- ✓ TabControl
- ✓ TabPage
- ✓ Checkbox
- ✓ ComboBox

## *Pantalla de edición de documentos (III)*

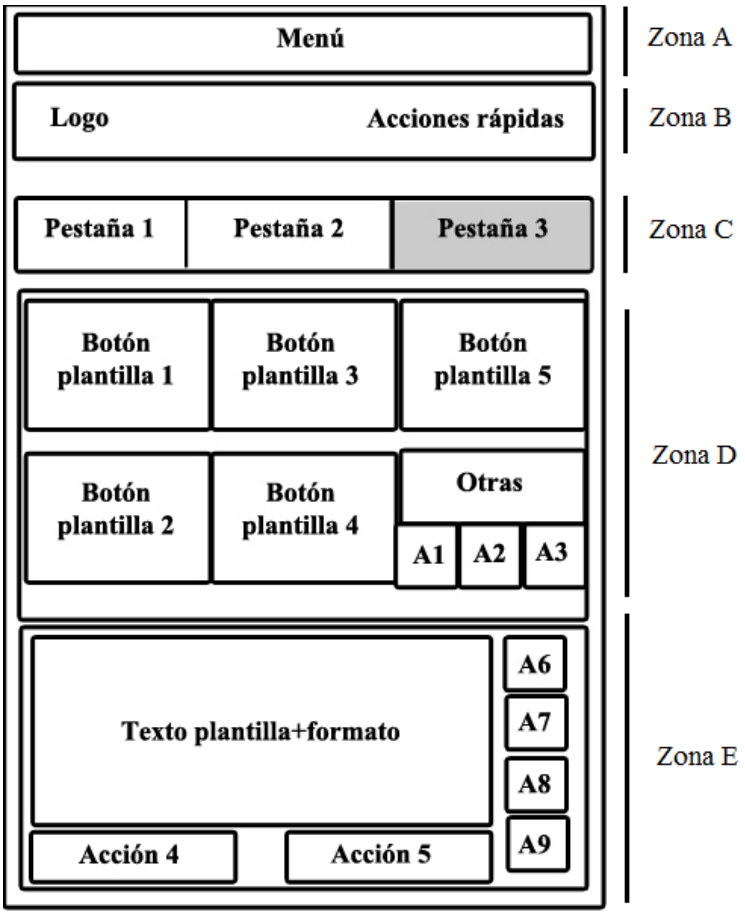


**Ilustración 14 - Diseño de pantalla de edición de documentos- Otras pestañas**

En esta pantalla, las Zonas A y B no varían respecto a la anterior. Lo mismo pasa con la Zona C, en la que sólo varía la pestaña seleccionada, que pasa a ser la "Pestaña N", ya que a partir de la pestaña 3, habrá tantas pestañas como secciones tenga asignadas el documento. Se presentan ordenadas de izquierda a derecha y de arriba abajo, como un flowlayout normal y corriente.

Las pestañas están enlazadas con el listado de secciones visto en la Pantalla de edición de documentos (I), por lo que hay dos maneras distintas de navegar entre ellos.

La parte de contenido de la pantalla se divide en dos paneles distintos, donde cada uno incluye elementos que cubren funcionalidades muy dispares. Estas funcionalidades son: gestión de plantillas y edición de contenido/comunicación con el editor.

El primer panel (Zona D) se encarga de la gestión de plantillas, donde se añaden 5 botones (habilitados o no en función del número de plantillas existentes) que corresponden a las

plantillas creadas para dicha sección. También están los distintos botones de acción que permiten crear una plantilla nueva, cargar alguna existente, ordenar, borrar y listarlas todas.

Por otra parte, el siguiente panel (Zona E) sirve para ver el contenido de las plantillas existentes, con su formato incluido (color, estilo, fuente…) y modificarlo en caso de que sea necesario. También hay diversos botones de acción para poder realizar tareas de movimiento de texto entre el editor y la aplicación.

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout
- ✓ FolderBrowserDialog
- ✓ PictureBox
- ✓ GroupBox
- ✓ TabControl
- ✓ TabPage
- ✓ ComboBox
- ✓ RichTextBox

## *Ordenación de plantillas*

El siguiente es un diálogo donde se nos permite ordenar las plantillas de manera dinámica.

Basado en un **Layout** en el que los elementos están ordenados de arriba abajo y de izquierda a derecha. Incluye una serie de botones y una caja de texto para poder filtrar por nombre (Acción 1). Los botones se agrupan por tipo de filtrado y los que comparten características, comparten color de fondo.

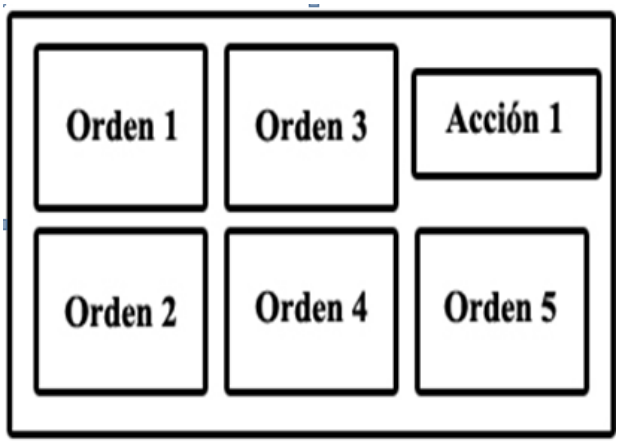


**Ilustración 15 - Diseño de filtrado de plantillas**

*Elementos*

- ✓ Panels
- ✓ Labels
- ✓ Buttons
- ✓ FlowLayout
- ✓ TextBox

# 6 Implementación

En el siguiente capítulo, se entra en detalle en la implementación de todas las funcionalidades del sistema, presentándose pantalla a pantalla, siguiendo la misma estructura que el Capítulo 5.2.1 . Por último, se detallan los aspectos técnicos más relevantes a la hora del desarrollo de TmpMaker.

## 6.1 Pantalla inicial


**Ilustración 16 - Pantalla inicial**

La pantalla de inicio dispone de las funcionalidades básicas para abrir documentos y configurar el sistema. Los botones dentro del panel de tipos disponibles, se generan de forma dinámica con el contenido de la tabla 'documents' de la base de datos. El usuario puede categorizarlas por color o un sistema de nomenclatura propio.

Por otra parte, el panel dispone de un scroll infinito para ver todos los tipos, que no suelen superar los 10 en número, aunque se permita la posibilidad.

Al darle click a uno de los tipos, pasaremos a la fase de edición del documento y se abrirá la pantalla principal de edición.

Por último, tenemos los botones de acción, con sus propios eventos que se encargan de abrir distintas modales o diálogos.

**Requisitos**

- RS8: El usuario ha de poder asociar un tipo 'documento' a un documento .docx de Microsoft Word.
- RS17: El usuario ha de poder gestionar tipos de documentos en la pantalla inicial: creación, modificación y edición.
- RS19: El usuario ha de poder gestionar tipos de documentos en la pantalla inicial: creación, modificación y edición.
- RS20: El usuario ha de poder abrir un documento en el editor de texto

## 6.2 Nuevo tipo de documento

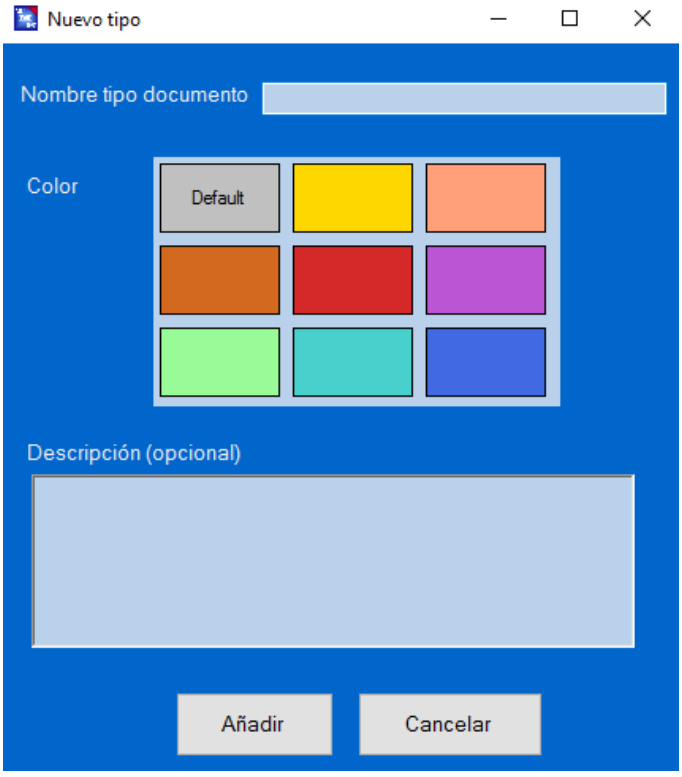


**Ilustración 17 - Pantalla de nuevo tipo de documento**

Se puede crear un nuevo tipo de documento y asignarle un color. El único valor obligatorio es el nombre, ya que el color tiene un valor por defecto y la descripción es opcional.

Al seleccionar añadir, se registra el tipo de documento, se carga en el panel de tipos de documentos disponibles de la pantalla inicial y se cierra la pantalla actual.

**Requisitos**

- RS19: El usuario ha de poder gestionar tipos de documentos en la pantalla inicial: creación, modificación y edición.

## 6.3 Abrir documento existente

Este botón se encarga de abrir la modal de selección de ficheros de Windows. Al seleccionar un fichero .docx y seleccionar abrir, pasaremos a la fase de edición del documento y se abrirá la pantalla principal de edición (sin secciones ni plantillas cargadas).

**Requisitos**

- RS20: El usuario ha de poder abrir un documento en el editor de texto

## 6.4 Edición de tipos de documento

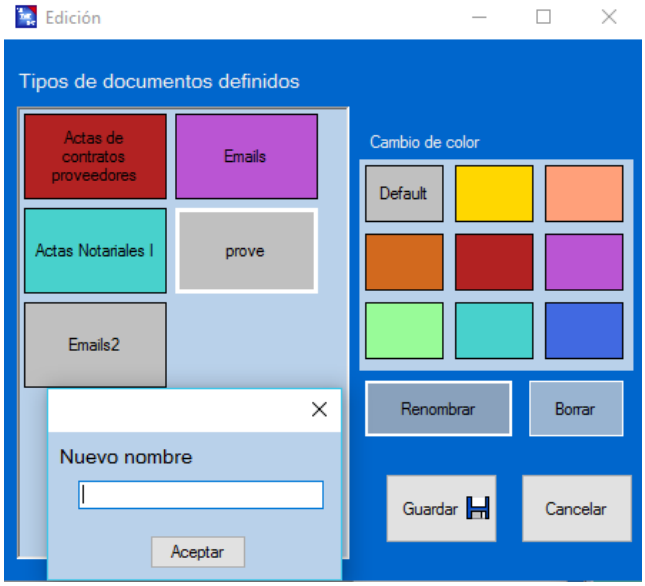


**Ilustración 18 - Pantalla de edición de tipos**

En esta pantalla, se pueden editar los tipos de documento y previsualizar los cambios antes de guardarlos, para una gestión más eficiente del tiempo. Antes de guardar los cambios siempre se pide confirmación en un diálogo.

Los botones del panel disponen de un atributo indicando si están seleccionados o no, simulando el comportamiento de un RadioButton, ya que se pueden editar de uno en uno.

El botón de "Renombrar" abre el diálogo que se puede ver en la imagen, en donde se puede añadir el nuevo nombre.

**Requisitos**

- RS19: El usuario ha de poder gestionar tipos de documentos en la pantalla inicial: creación, modificación y edición.

## 6.5 Configuración

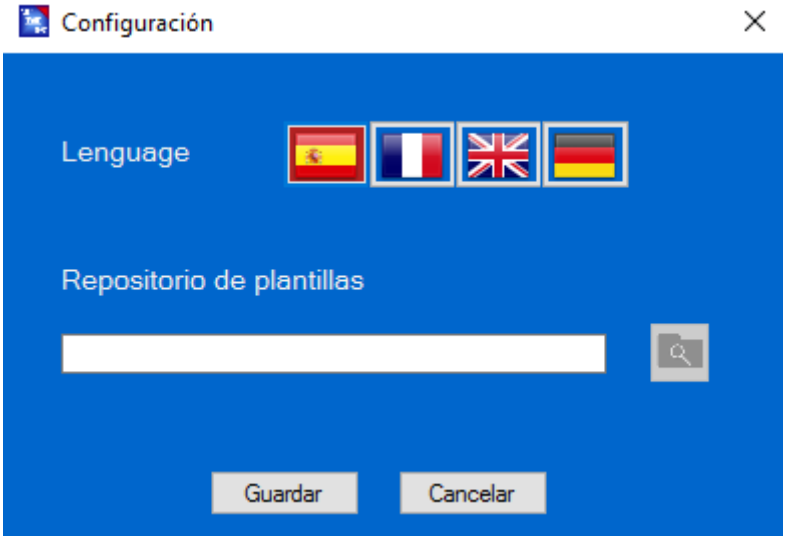


**Ilustración 19 - Pantalla de configuración**

La pantalla de configuración permite dos acciones: "Cambiar el idioma" y "Cambiar el directorio" en donde se almacenarán las plantillas.

La manera en la que la aplicación cambia de idioma se basa en dos aspectos: distintos archivo de recursos según el lenguaje y la cultura (`CurrentCulture` [14]) de la aplicación. De esta manera, se definen distintos ficheros de recursos con distintas traducciones y/o elementos, que mediante un cambio en la cultura de la aplicación, al refrescar las ventanas, se puede ver cómo cambian todos los elementos a los definidos por estos mismos ficheros.

De manera similar, pero más simple, se implementa la funcionalidad del cambio de directorio. Los formularios principales, cargan una variable global con el valor del directorio almacenado en la base de datos, el cual es utilizado a la hora de guardar un elemento y seleccionar un directorio por defecto en el diálogo de exploración de directorios de Windows.

Cabe destacar que los elementos ya almacenados con anterioridad, se copian a este nuevo directorio definido por el usuario, para evitar perder los datos utilizados con anterioridad.

**Requisitos**

- RS21: El usuario ha de poder cambiar el idioma de la aplicación y la ruta de almacenamiento de los ficheros.

## 6.6 Sincronización y pantalla principal

Una vez seleccionado un tipo de documento, se procede a inicializar el formulario principal, junto a la inicialización de un nuevo proceso de Microsoft Word, en paralelo.

La disposición de la pantalla depende del último uso del editor de texto, pero como disposición inicial, éste quedará desplazado a la izquierda ocupando más de media pantalla. Mientras que TmpMaker permanecerá abierto a un tamaño fijo a la derecha de la misma.

Cualquier diálogo de error o de aviso que sea lanzado por la aplicación Word, pausa el inicio del software hasta que se retome la inicialización de Microsoft Word.

La pantalla que podemos ver al finalizar la carga es la siguiente:

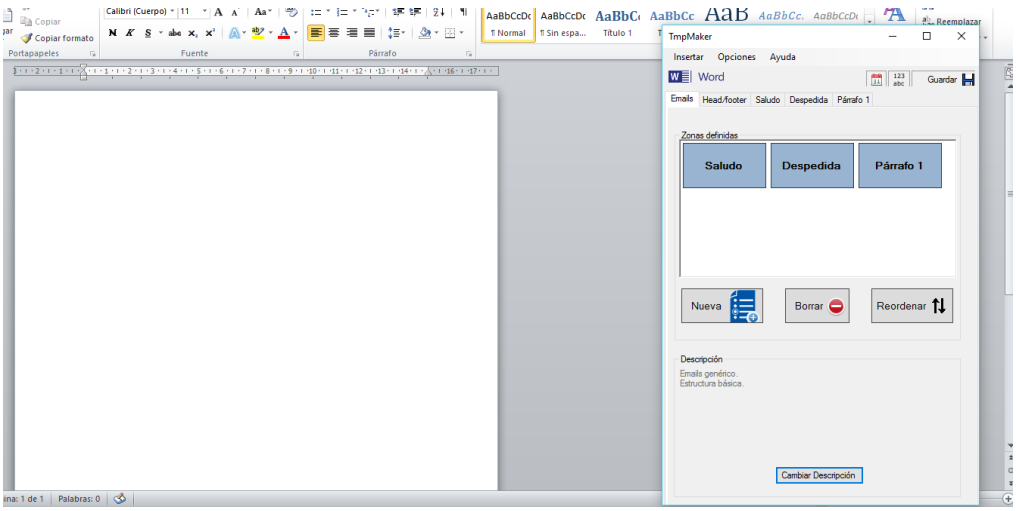


**Ilustración 20 - Vista del software a pantalla completa**

## 6.7 Pantalla principal

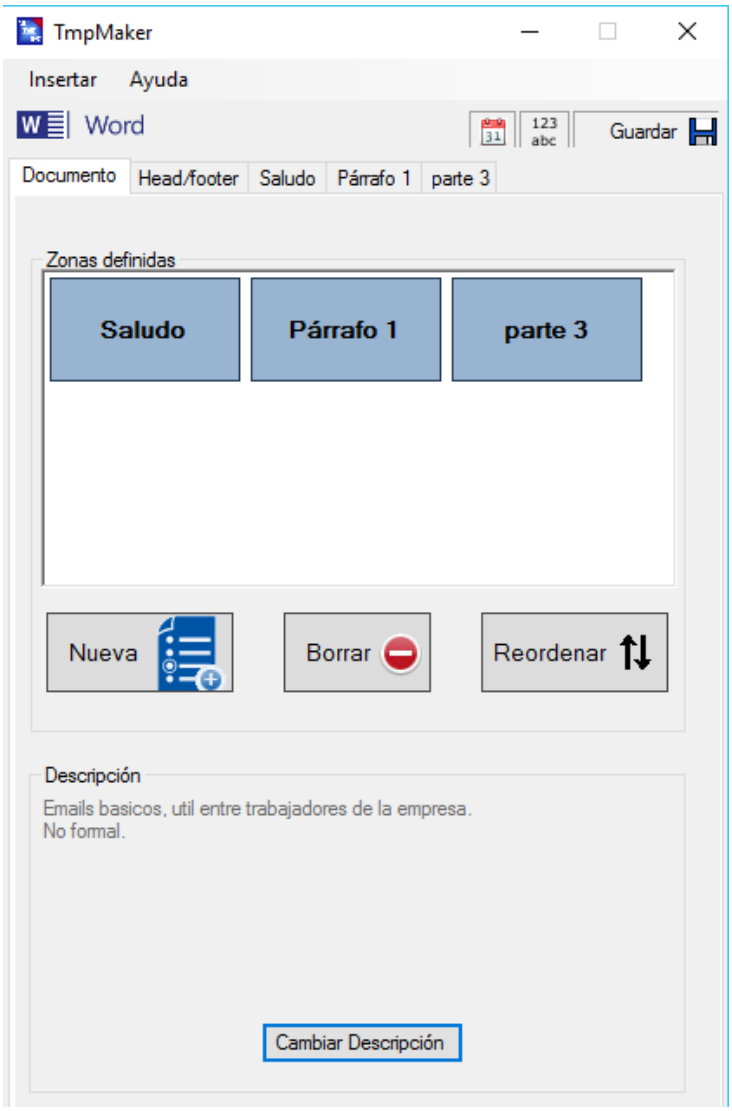


**Ilustración 21 - Pantalla del programa principal**

En esta pantalla se añade toda la información asociada a un tipo de documento, que corresponde con las secciones y la descripción del mismo. Por eso mismo, dividimos la pantalla en dos zonas:

### 6.7.1 Secciones del documento

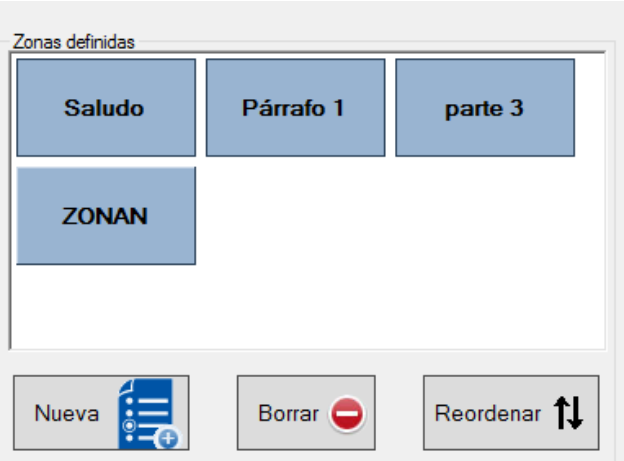


**Ilustración 22 - Panel de secciones del documento**

En esta parte de la pantalla principal, se muestran todas las secciones o zonas definidas para ese tipo de documento, las cuales corresponden con las pestañas del componente tabs. Éstas, se ordenan de izquierda a derecha y de arriba abajo. Al hacer click sobre uno de los botones, se refresca la pantalla y nos movemos al la pantalla de Gestión de plantillas.

En esa misma pestaña podemos añadir secciones al documento, clickando sobre el botón 'Nueva', el cual se encarga de abrir un diálogo en el que sólo hay que añadir un nombre a la nueva sección. Si el número de secciones es elevado o sus nombres demasiado largos, las pestañas se disponen de la siguiente manera:

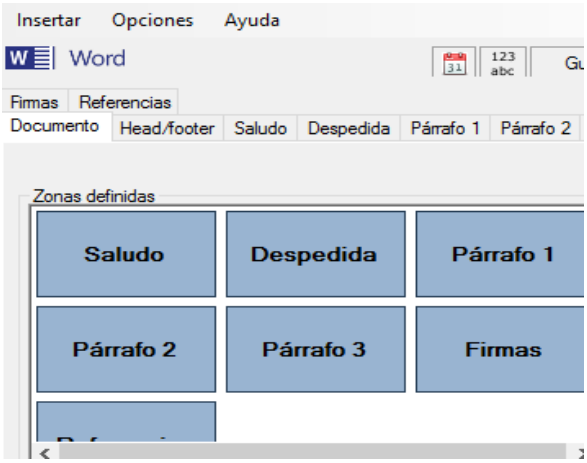


**Ilustración 23 - Disposición de pestañas y botones de sección**

Cuando se cambia de pestaña o se clicka sobre un botón del panel de "Zonas definidas" y se refresca la pantalla, toda la información recogida para el pintado de la misma, se almacena como variables estáticas, para impedir la recarga en caso de que el usuario quiera desplazarse entre zonas múltiples veces.

**Requisitos**

- RS1: El usuario ha de poder diferenciar tres tipos elementos: documentos, secciones y plantillas.
- RS2: El usuario ha de poder asociar n secciones a un documento, sin duplicidad de nombres.
- RS4: El usuario ha de poder borrar y actualizar documentos, secciones y plantillas.
- RS15: El usuario ha de poder ordenar las secciones mediante una modal y ver los cambios dinámicamente.

### *6.7.2 Descripción del documento*

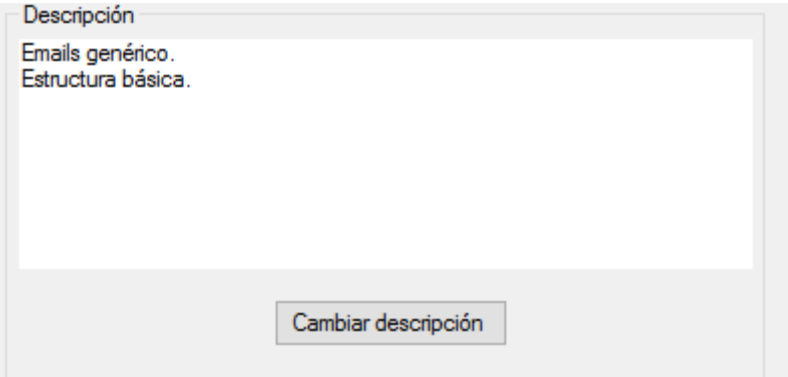


**Ilustración 24 - Panel de cambio de descripción del documento**

En esta otra de la pantalla principal, se puede consultar y modificar la descripción del tipo de página.

En primera instancia, el campo de texto está deshabilitado, aunque se puede habilitar presionando el botón de “Cambiar descripción”.

**Requisitos**

- RS27: El usuario ha de poder añadir una descripción a los documentos para facilitar la distinción de los mismos.

## 6.8 Gestión de encabezados y pies de página

La pestaña de “Head/Footer” nos sirve para gestionar los encabezados y pies de página. Ésta carga todos los encabezados por una parte y luego todos los footers, para que el usuario pueda seleccionarlos desde el componente de listado.

La lista es desplegable, y se puede modificar el contenido, añadiendo elementos o borrándolos. A su derecha, se añade un checkbox para añadir el número de página.

Para añadir la plantilla deseada al documento .docx, se precisa clickar sobre el botón de 'Enviar a Word', que se encarga de realizar un Paste sobre una sección del documento, cuyos tipos en la librer como nombre "Footer" o "Header". En el caso del número de página, se añade a estas mismas secciones un item `Word.PageNumbers`. Para almacenar los cambios, se puede hacer una sobreescritura  o guardarlo como una nueva plantilla 


**Ilustración 25 - Pantalla de gestion de encabezados y pies de página**

## Requisitos

- RS3: El usuario ha de poder asociar n plantillas a una sección, sin duplicidad de nombres.
- RS4: El usuario ha de poder borrar y actualizar documentos, secciones y plantillas.
- RS5: El usuario ha de poder sobreescribir el contenido de una plantilla.
- RS10: El usuario ha de poder mover el contenido de una plantilla al editor de texto con un solo click.
- RS11: El usuario ha de poder mover el contenido de una plantilla a una caja de texto donde se pueda formatear y modificar.

- RS13: El sistema debe tratar los encabezados y pies de página como unas secciones prefijadas y siempre disponibles.
- RS22: El usuario ha de poder añadir numeración de página tanto en el encabezado como en el pie de página.

## 6.9 Gestión de plantillas

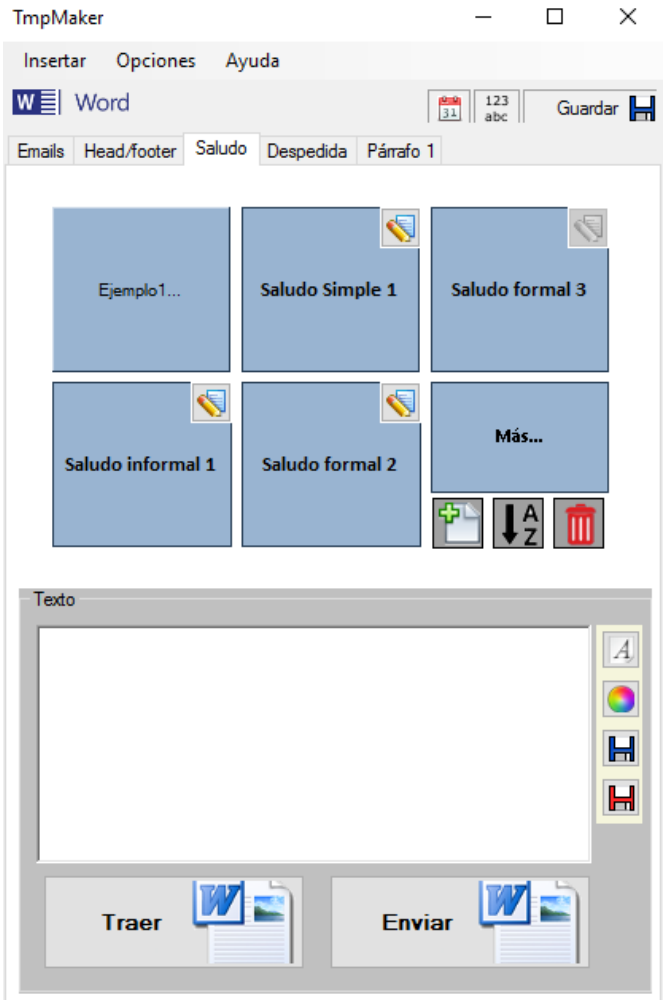


**Ilustración 26 - Pantalla de gestión de plantillas**

En el panel A se tienen las primeras cinco plantillas que el usuario añadió a la sección, aunque se pueden ver en su totalidad en un listado, clickando el botón 'Más…' en el caso de que esté activado.

Cada uno de los botones dispone de tres lanzadores de eventos:

1. OnClick sobre el botón azul: es el más relevante, ya que si clickamos sobre él (zona azul), copiaremos el contenido de la plantilla al editor de texto, en la posición que marca el cursor dentro del documento.
2. MouseOver : evento se lanza al pasar el mouse por encima, y realiza la acción de ir a buscar el contenido de la plantilla a un diccionario y formar un resumen de unas cuantas letras para poder distinguir el contenido. Este texto se añade al centro del botón y vuelve al estado inicial al sacar el cursor del propio botón.
3. OnClick sobre el botón de edición  : El último evento se lanza al clickar sobre el icono situado en la esquina superior derecha; aquí copiamos el contenido de la plantilla al panel inferior, para poder editarlo y guardar los cambios posteriormente.

Por otra parte se puede añadir una plantilla, reordenarlas y eliminar alguna de ellas, utilizando los siguientes botones: 

Pasando al panel inferior, se dispone de varios botones de acción. Los dos superiores:  y  permiten modificar el formato de texto de la caja de texto, mientras que los últimos:  y  nos permiten guardarlo como una plantilla nueva y sobreescribir la seleccionada con anterioridad, respectivamente.

Por último, disponemos de dos botones para mover texto entre la aplicación y Microsoft Word

- Traer: Incluye el texto seleccionado (incluido formato) en la caja de texto, para poder modificarlo y/o almacenarlo como plantilla.


- Enviar: Copia el contenido de la caja de texto al editor, justo donde esté situado el cursor o acaba un texto que seleccione el usuario.


**Requisitos**

- RS1: El usuario ha de poder diferenciar tres tipos elementos: documentos, secciones y plantillas
- RS3: El usuario ha de poder asociar n plantillas a una sección, sin duplicidad de nombres.
- RS4: El usuario ha de poder borrar y actualizar documentos, secciones y plantillas.

- RS5: El usuario ha de poder sobreescribir el contenido de una plantilla.
- RS10: El usuario ha de poder mover el contenido de una plantilla al editor de texto con un solo click.
- RS11: El usuario ha de poder mover el contenido de una plantilla a una caja de texto donde se pueda formatear y modificar.
- RS14: El usuario ha de poder copiar el contenido de una plantilla en el punto donde se sitúa el cursor en el editor de texto
- RS16: El usuario ha de poder almacenar las fechas de creación, y última modificación de documentos, secciones y plantillas.
- RS17: El usuario ha de poder cargar documentos mediante el explorador de windows.
- RS18: El usuario ha de poder ver una previsualización del contenido de una plantilla ya sea en un Tooltip o en un panel.
- RS23: El usuario ha de poder mover un texto seleccionado en el editor, a la caja de texto de edición del propio programa.

## 6.10 Ordenación de plantillas

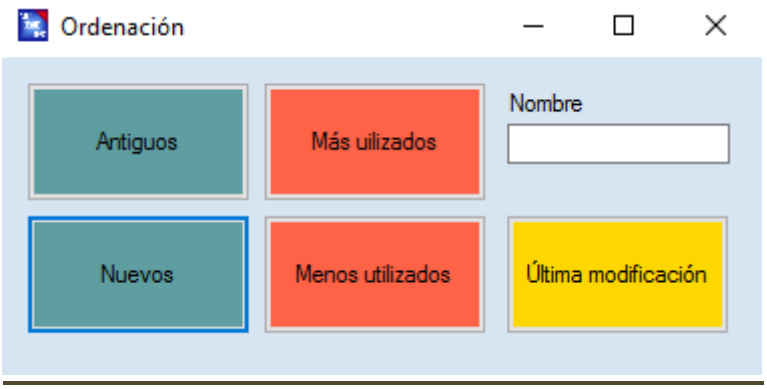


**Ilustración 27 - Pantalla de ordenación y filtrado de plantillas**

En esta pantalla, se pueden ordenar las plantillas por antigüedad, usos, última modificación y/o filtrarlas por nombre. Todo de manera dinámica.

Para hacerlo, se va a buscar la información a base de datos de manera ordenada y luego se modifica la lista de plantillas en función de los resultados de la consulta. Por último, se refresca el panel de plantillas con el fin de ver aplicados los nuevos cambios "on the fly".

### Requisitos

- RS26: El usuario ha de poder filtrar las plantillas por nombre, para facilitar la localización de las mismas.

# 6.11 Otras acciones

A continuación se presentan las acciones que se pueden realizar mediante el menú principal y el de acciones rápidas.

## *6.11.1 Añadir imagen*

En el menú principal, se puede encontrar junto a la acción de añadir documento (mediante explorador de directorios de Windows) la acción de añadir una imagen.

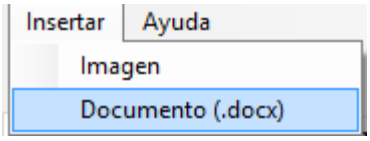


**Ilustración 28 - Menú principal de TmpMaker**


**Ilustración 29 - Pantalla de inserción y edición de imágenes**

Aquí se puede escalar la imagen y añadirla al documento seleccionando aceptar, previsualizando el resultado en el panel superior.

### Requisitos

- RS24: El usuario ha de poder cargar imágenes y redimencionarlas mediante una modal y disponer de un panel de previsualización.

### 6.11.2 Descripción del documento

En la barra de acciones rápidas, además de poder guardar el documento .docx, disponemos de dos botones que realizan dos funciones similares:

- Formatear número 

  Si el usuario selecciona un número en el documento Word y clicka en el botón, lo convierte en texto y en mayúsculas.

- Formatear fecha 

  Si el usuario selecciona una fecha en el documento Word y clicka sobre este botón, siempre que esta esté en formato 'dd/MM/yyyy', se sustituirá la selección por el texto en mayúsculas.

El formato se realiza analizando cada carácter de la selección del usuario y montando el string final mediante el uso de StringBuilder.

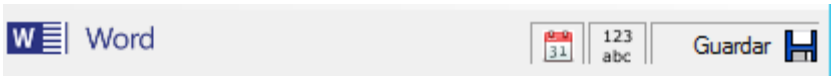


**Ilustración 30 - Sección de acciones rápidas**

**Requisitos**

- RS6: El usuario ha de poder convertir una fecha seleccionada en el editor, al formato notarial (texto en mayúsculas)
- RS7: El usuario ha de poder convertir una número seleccionado en el editor, al formato notarial (texto en mayúsculas)
- RS25: El usuario ha de poder utilizar un área de acciones rápidas que se aplican sobre el propio editor. Éste área ha de incluir las conversiones en formato notarial.

## 6.12 Aspectos técnicos

A continuación exponemos un listado de aspectos destacables en cuanto a especificaciones técnicas del desarrollo:

1. VisualStudio nos proporciona una serie de funcionalidades de creación de tipos de proyecto, que genera una estructura de clases que sigue marca la arquitectura y la estructura que se debe seguir para trabajar con aplicaciones del tipo Windows Forms, así que seguimos esta arquitectura.
2. Partiendo de la arquitectura base proporcionada por el entorno de desarrollo, se decide seguir el patrón DAO [15]para las comunicaciones con las bases de datos que existen en el proyecto.
3. Con el fin de optimizar tiempos y utilización de recursos, cada componente como pueden ser tabs, botones (de los paneles de sección y plantillas), listados y datos obtenidos de base de datos y/o documentos .docx almacenados en disco, se implementan como clases estáticas.
4. Las clases utilizadas de la librería `Microsoft.Office.Interop` [16] para poder trabajar con el editor Microsoft Word son `Word.Application`, `Word.Document`, `Word.Paragraph` y `Word.Selection`.
5. Se utiliza un enumerado para diferenciar plantillas, enumerados y pies de página, pero comparten la misma clase, ya que son elementos muy similares y comparten muchas funciones.
6. A la hora de sobreescribir un fichero, siempre realizamos dos acciones: Comprobamos que el fichero exista(`FileSystem.FileExists`) y luego copiamos el contenido(`FileSystem.CopyFile`).
7. El cambio de idioma se realiza mediante un cambio de la cultura de la instancia del hilo principal de la aplicación, para después hacer un refresh del formulario y pintar el fichero de recursos asociado a dicha cultura.
8. La carga de las descripciones de las plantillas en la pestaña de gestión de plantillas, se realiza con la carga del tab, lanzando múltiples hilos concurrentes para rellenar una estructura en donde almacenamos toda la información relacionada con cada plantilla.
9. Al iniciar el programa, el controlador de consultas SQL, se encarga de contemplar el caso de que la base de datos no esté creada.

   En caso de no estarlo, la crea y añade cada una de las tablas necesarias.

10. En cuanto a nomenclatura de tipos y variables, seguimos el estándar utilizado en el grupo Logitravel, basado completamente en las recomendaciones de Microsoft [17] para el desarrollo en VB.Net.
11. El **Plugin** Resharper [18], nos ofrece la posibilidad de formatear las variables con el objetivo de facilitar la lectura y evitar los errores más típicos de la programación relacionados con la nomenclatura de variables y constantes.

# 7 Validación

Si nos detenemos a mirar la planificación temporal y las tareas previstas, podemos ver cómo había planificada una tarea que se ejecutaba en paralelo a la fase de desarrollo, llamada "Validación y reajustes". Esta tarea fue muy relevante y cobró mucha fuerza en nuestro proyecto, ya que surgieron muchos reajustes durante las semanas que duró el desarrollo.

La validación se realizaba dos veces por semana y consistía en una demostración de la funcionalidad al tutor de prácticas y presentación de la interfaz gráfica final, ventana a ventana y diálogo a diálogo.

A continuación se presenta los cambios surgidos por semana, presentando las problemáticas, los cambios aplicados y los rechazados.

## Semana 1

**Problemas**

1. Disposición de la ventana mejorable. Siempre ha de prevalecer el documento Word sobre la aplicación.
2. Las ventanas y modales han de ser fijas y no poder agrandarse.

**Cambios aplicados**

1. Documento de Word desplazado, centrado a la izquierda y ocupando un 65% de la pantalla.
2. Se deshabilitó la opción de hacer redimencionables las ventanas.

**Rechazados**

Ninguno

## Semana 2

**Problemas**

1. Poder maximizar TmpMaker cuando hay dos pantallas disponibles.
2. Ordenar por fecha de creación siempre por defecto.

**Cambios aplicados**

1. Todas las consultas a base de datos ordenan por fecha de creación descendente.

**Rechazados**

1. No podemos maximizar  la ventana en según qué casos. Siempre ha de tener el mismo comportamiento.

# Semana 3

**Problemas**

Ninguno

# Semana 4

**Problemas**

1. No se puede enviar un segmento de texto del editor de texto a la aplicación con un solo click.
2. Hay excepciones no controladas al clickar sobre un botón de plantilla antes de que éste cargue el resumen.

**Cambios aplicados**

1. Se añade un botón para poder traer texto del documento al programa principal. Éste lo copia en el panel de edición.
2. Se controla la excepción reestructurando la carga de resúmenes a la vez que se cargan las plantillas.

**Rechazados**

Ninguno

Una vez realizadas todas las validaciones, por parte del Notario, concluimos en que el software ya era entregable y que toda la funcionalidad básica, en concreto todos los requisitos incluidos de la Fase 1, especificados en el documento de requisitos estaban implementados y aceptados.

# 8 Testing

Para verificar el correcto funcionamiento de la aplicación, se realizaron múltiples pruebas de código. En algunos de los test se vieron errores que se solucionaron y algunos tiempos elevados pudieron ser reducidos.

A continuación presentamos la serie de tests, sus resultados y los errores que no pudieron ser corregidos.

### Tests de cobertura

Se probaron todas las funcionalidades especificadas en el Documento de Requisitos de Usuario y cubre la totalidad de las mismas. Algunas de las funcionalidades se poder realizar desde distintos lugares y de diferente manera, como puede ser el adjuntar un nuevo documento como plantilla.

### Tests unitarios

Se probaron las llamadas a todas las funciones de la aplicación una a una, con distintos parámetros.

Se hicieron las llamadas con parámetros específicos que conllevan a error, para saber si el sistema puede recuperarse de los mismos.

El caso de prueba por excelencia es la creación de plantillas y documentos con el mismo nombre y situados en el mismo archivo.

Los resultados mostraron múltiples errores, como la pérdida del lenguaje definido en la configuración, errores de sincronización con el editor o errores de almacenamiento al tratar elementos con el mismo nombre, que se han solucionado a posteriori.

### Tests de estrés

Sólo se pudieron hacer tests de estrés a la base de datos, ya que no podríamos generar plantillas, documentos y ficheros .docx de manera automática.

Se hicieron una serie de inserciones, modificaciones y borrados masivos, con el fin de comprobar que la base de datos y las transacciones SQL estaban bien programadas; haciendo uso de las funciones de la clase SQLiteController.

El sistema aguantó el test de estrés, en los cuales se lanzaron 10000 consultas, modificaciones, inserciones y borrados de todas las tablas exceptuando la de configuración, a causa de que no espera una utilización masiva.

### Tests de eficiencia

Mediante el sistema de monitorización del Visual Studio y la utilización de clocks o timers en el código medimos temporalmente la eficiencia de las funciones críticas.

La monitorización de los tiempos muestra los siguientes resultados, basándonos en una media de 10 ejecuciones, sin tener en cuenta las tareas gestionadas por Word o el sistema operativo como pueden ser el guardado de documentos.

| **Tarea propia de TmpMaker** | **Tiempo (ms)** |
|---|---|
| Carga de la pantalla inicial | 1800 |
| Creación de tipo de documento | 160 |
| Modificación/borrado de tipo de documento | 150 |
| Cambio de configuración | 170 |
| Arranque Word + gestión plantillas | 3140 |
| Mover texto entre aplicaciones | 40 |
| Carga de plantillas por sección + preview | 1300 |
| Ordenación de plantillas | 70 |
| Creación de plantillas | 210 |

| | |
|---|---|
| Conversiones en formato notarial | 20 |

**Tabla 8 - Tiempos medios de funciones críticas**

## 8.1 Errores de sistema no solucionados

Los errores surgidos se pudieron resolver casi en su totalidad, exceptuando un error en particular, el cual me fue imposible de controlar debido a la complejidad del mismo y de la poca información encontrada sobre el tema.

El problema se localizó en la comunicación entre TmpMaker y Microsoft Word, a la hora de abrir un documento y consultar su contenido, tanto a la hora de editarlo, obtener el resumen del contenido o mover el contenido al documento Word. Algunas veces falla la comunicación entre los mismos, lanzándose un mensaje de error del sistema a la hora de mover texto del input de la aplicación al documento .docx.

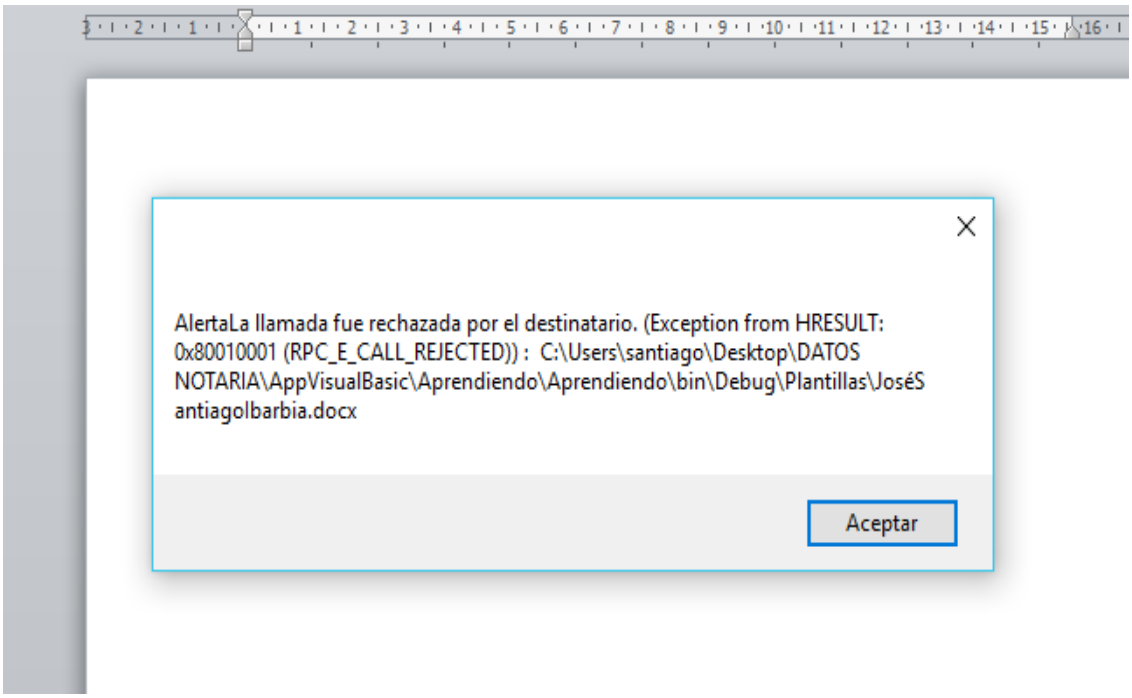


**Ilustración 31 - Mensaje de error de Microsoft Word**

El error es propio de Microsoft Word, al interactuar con el propio sistema Operativo, a la hora de intentar Abrir un fichero el cual otro proceso intenta modificar y/o consultar.

Se ha tratado de gestionar el error en la propia aplicación, pero éste es transparente incluso para la librería, por lo que no se puede tratar en un try catch y se hace imposible su interpretación. De todas maneras, este error no es persistente y sólo ocurre en determinados momentos en el que el sistema se encuentra con muchas peticiones y bloqueos internos. En casi la totalidad de los casos, al lanzar la acción por segunda vez, el comportamiento ha sido el adecuado.

# 9 Cierre y Conclusiones

Por último, se presenta el cierre y las conclusiones del proyecto, que se dividen en posibles ampliaciones futuras y opinión personal.

## 9.1 Cierre del proyecto

Podemos preguntarnos ahora: ¿Podemos dar por cerrado el proyecto? La respuesta es no. Excepto si no tenemos pensado iterar otra vez más y lanzar otra versión del software.

Como se ha comentado anteriormente, hay muchas ampliaciones posibles y nuevas funcionalidades, por lo que podríamos reabrir el proyecto y continuarlo siguiendo el modelo de ciclo de vida iterativo que se ha comentado en el capítulo 3 (Planificación).

Sin embargo, sí que realizamos unas tareas propias de un proyecto finalizado, aunque sea parcialmente.

**Fase 1**

El día 10/09/2015 se presentó el software resultante ya en funcionamiento, cumpliendo los términos y límites temporales que habíamos impuesto. Se llegó a cubrir toda la funcionalidad de la Fase 1 y se demostró el funcionamiento del programa a los trabajadores de la empresa, que eran un total de 8 personas.

Una vez presentado el proyecto, se realizó una formación de 25 minutos a todos los trabajadores, para luego proceder con la instalación del software en todas las máquinas y la distribución de una copia del mismo.

Después de 4 días de pruebas por parte de los trabajadores y algunas dudas que salieron, pudimos dar por cerrada la Fase 1 y continuar con otras tareas.

**Fase 2**

Una vez desarrolladas las funcionalidades extras de la Fase 2 y acabados los **Tests** unitarios, se instaló el software en distintos ordenadores personales para validar el funcionamiento.

Por otra parte, cabe comentar que ninguno de los riesgos previstos (Ver subcapítulo 3.4 Riesgos) fueron materializados, por lo que concluimos que la planificación de Riesgos fue acertada y la estimación de probabilidad fue un tanto sobrevalorada.

De esta manera, cerramos el proyecto.

## 9.2 Posibles ampliaciones

Como se ha comentado a lo largo del documento, el proyecto está pensado para continuar añadiendo funcionalidades y nuevas fases o iteraciones en el ciclo de vida del proyecto.

Hasta ahora se han realizado dos iteraciones, pero ya he comenzado con la siguiente fase.

### *9.2.1 Fase 3*

En esta iteración, se introducen unos cambios que bien podrían considerarse un nuevo proyecto, ya que implica un análisis y diseño de todas las nuevas pantallas.

Principalmente se quiere implementar las mismas funcionalidades para una plataforma Mobile, sea en Android o IOS. Para esto se debe prescindir del editor de textos y ajustar los componentes a las nuevas pantallas y nuevos eventos que nos proporciona un dispositivo móvil.

Esto nos abriría nuevas puertas y nuevos usuarios, ya que podemos llegar a una plataforma en donde la distribución del software es muy sencilla, pero conlleva a una pérdida de funcionalidades importantes.

### *9.2.2 Fase 4*

La siguiente ampliación es crear una web que amplifique la funcionalidad de la aplicación mobile, pudiendo dar formato al texto y siendo accesible desde todos los dispositivos.

Se requiere de una pequeña inversión y de mucho desarrollo y diseño, ya que comienzan a tener importancia los aspectos de seguridad [19] y comunicación de servicios.

El desarrollo de la web implica la utilización de webservices para la comunicación de la parte cliente con el servidor y la elaboración de ciertos scripts, haciendo uso de bootstrap o Jquery.

## 9.3 Opinión personal

Finalmente, me gustaría exponer qué ha significado para mí, la realización de este proyecto de final de grado.

Lo que había pensado como un proyecto sencillo, en donde muchas pautas estaban marcadas pero otras las podía definir yo mismo por mi propia cuenta, acabó siendo más complicado de lo esperado, ya que fue dificultoso ampliar el abanico de funcionalidades que habíamos acordado en la notaría. Puedo decir que no fue sencillo planificar la duración ni el alcance del proyecto a partir de la segunda fase, ya que entrar en el mundo laboral y continuar con los estudios me impedía avanzar sustancialmente en el proyecto.

Por otra parte, ha sido una experiencia gratificante, ya que he trabajado en una empresa y he podido analizar sus necesidades y convertirlas en un software sencillo, en un tiempo relativamente corto. Todo con el apoyo de mi tutor de prácticas y del tutor el proyecto de final de grado.

Un punto que me gustaría destacar es que el haber elegido un lenguaje nuevo como vb.net, me ha abierto las puertas al mundo laboral, ya que ahora mismo me encuentro en un momento en el que sigo utilizando ese lenguaje de programación, algunas herramientas y sobre todo, muchos de los conocimientos aprendidos a lo largo de este tiempo, focalizándome en los estándares y guías de programación, conceptos teóricos, técnicas de modelado y diseño, entre otros.

Sin embargo, hay ciertos aspectos que me hubiese gustado que hubiesen sido diferentes; a mi pesar, sólo estuve desarrollando la Fase 1 durante un mes, dedicándole bastantes horas pero no las suficientes como para poder considerar el resultado un software entregable. Por ello, me hubiese gustado haber comenzado el proyecto como una web y no como una aplicación de escritorio, que tengo pensada realizar en la Fase 4 del proyecto.

Otro aspecto que me hubiese gustado cambiar es haber hecho la Fase 2 implementando distintas librerías y experimentar con otros lenguajes de programación y herramientas software, para amplificar mi aprendizaje en ciertos aspectos más técnicos.

De todas maneras, puedo decir que estoy muy satisfecho por el esfuerzo realizado y por las horas de trabajo, incluyendo la documentación y horas de aprendizaje, que he realizado. Me he mostrado a mí mismo que estoy preparado para afrontar cualquier proyecto que pueda surgir a lo largo de mi carrera profesional.

# 10 Bibliografía


[1] "*Manual de Office 2010*" MEDIAactive

MARCOMBO .SA Junio 2010

ISBN: 978-267-1680-4

[2] "*Windows 8, ventajas y limitaciones a la hora de instalarlo*" [Internet]

https://norfipc.com/articulos/windows8-conocer-ventajas-limitaciones-antes-instalarlo.html

[3] "*Los 6 editores de texto más eficaces*" [Internet]

Universia Argentina. Septiembre 2013.

http://noticias.universia.com.ar/en-portada/noticia/2013/09/02/1046312/6-editores-texto-mas-eficaces.html

[4] "Tiendas alternativas a Google Play" [Internet]

Junio 2015

https://www.xatakandroid.com/aplicaciones-android/tiendas-alternativas-a-google-play-estas-son-las-mejores-a-dia-de-hoy

[5] "Patrones para el diseño móvil" [Documento]

Pau Giner. 2013

http://mastermoviles.usj.es/material/usabilidad-USJ.pdf

[6] "Metodología para el diseño y desarrollo de interfaces de usuario" [Documento]

Alejandro Báez - Cristian Castañeda - Diego Castañeda. Junio 2005.

http://pegasus.javeriana.edu.co/~fwj2ee/descargas/metodologia(v0.1).pdf

[7] "Diseño Web Centrado en el Usuario". [Internet]

Yusef Hassan - Francisco J. Martín Fernández - Ghzala Iazza. 2004

https://www.upf.edu/hipertextnet/numero-2/diseno_web.html

[8] ”Microsoft .NET” [Internet]

https://es.wikipedia.org/wiki/Microsoft_.NET

[9] “Introducción a LINQ en c#” [Internet]

https://docs.microsoft.com/es-es/dotnet/articles/csharp/programming-guide/concepts/linq/getting-started-with-linq

[10] “ACID (atomicity, consistency, isolation, and durability)” [Internet]

Margaret Rouse. Julio 2016

http://searchsqlserver.techtarget.com/definition/ACID

[11] GIT Official Site

https://git-scm.com/

[12] “El ciclo de vida iterativo e incremental” [Internet]

Javier Garzás. Octubre 2012

http://www.javiergarzas.com/2012/10/iterativo-e-incremental.html

[13] “System.Windows.Forms Namespaces” [Internet]

https://msdn.microsoft.com/es-es/library/system.windows.forms(v=vs.85).aspx

[14] “How to: Set the Culture and UI Culture for Windows Forms Globalization” [Internet]

https://msdn.microsoft.com/en-us/library/b28bx3bh(v=vs.90).aspx

[15] “Patrones de Diseño (Active Record vs DAO)” [Internet]

Cecilio Álvarez . Julio 2014

https://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao

[16] “Cómo: Tener acceso a objetos de interoperabilidad de Office mediante las características de Visual C#” [Internet]

https://docs.microsoft.com/es-es/dotnet/articles/csharp/programming-guide/interop/how-to-access-office-onterop-objects

[17] Documentación de desarrollo en .NET [Internet]

https://msdn.microsoft.com/es-es/library/aa139615.aspx

[18] Resharper Official Site [Internet]

https://www.jetbrains.com/resharper/

[19] "Fundamentos de seguridad en redes: aplicaciones y estándares"

William Stallings. 2003

ISBN: 84-205-4002-1