Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

XGen: Generador de Formularios Dinámicos en HTML5

Alumno: Enrique Tamames Sobrino

Tutor: Valentín Cardeñoso Payo
Tutor HP: Julio Mateos Tello

A todos aquellos que han
hecho posible este proyecto

Tabla de Contenidos:

1. INTRODUCCIÓN

1.1. MOTIVACIÓN DEL TRABAJO DE FIN DE GRADO

La oferta open source de productos para la gestión de formularios no es muy abundante, encontrándose con frecuencia circunscrita al ámbito de la construcción de formularios. Recientemente han aparecido con mucha fuerza algunas iniciativas en el campo de las fuentes abiertas para la construcción de formularios. Y, como suele suceder en la tecnología de fuentes abiertas, vienen de la mano de un estándar que está ganando gran aceptación en el mercado: *XForms***.**
Análogamente a lo que sucede en el caso de otros elementos funcionales, existen en el mercado soluciones paquetizadas que integran entre sus funcionalidades servicios de gestión de formularios.

1.1.1. Breve Investigación

Se trata de realizar una búsqueda de información referente al estándar *XForms* de forma que podamos acotar el análisis de nuestro sistema y lo podamos enfocar correctamente. Esta investigación consiste en conocer el funcionamiento del estándar *XForms* de forma que seamos capaces de entender su modelo propuesto. Actualmente, no está demasiado adaptado y el mercado no ofrece una amplia gama de posibilidades respecto a aplicaciones que lo integren. Por esto, resulta más difícil enfocar el sistema por un camino fácil.

Los principales objetivos que definiremos serán:

- Familiarizarnos con el software existente que implementa XForms
- Conocer más a fondo el estándar XForms
- Conocer diferentes métodos a nivel de implementación (especialmente si existe alguno para la plataforma .NET)
- Realizar una pequeña búsqueda de la tecnología empleada para cada uno de las soluciones existentes.

1.1.2. Aplicaciones Open Source Disponibles

Actualmente sólo Opera soporta XForms nativamente. Sin embargo, existen varios plugins y extensiones que dan soporte a otros navegadores. Firefox soporta XForms a través de una extensión. Para IE6 está formsPlayer, un plugin que extiende al navegador haciendo que soporte XForms. XForms también puede ser usado a través de varias tecnologías de servidor que convierten el código de XForms a formularios de HTML en tiempo de ejecución y de manera transparente. Estas implementaciones incluyen a los proyectos de código abierto Chiba (ahora BetterForm) y Orbeon. Recientemente ha sido dada a conocer una nueva herramienta, llamada AJAXForms, que transforma, en tiempo de compilación, documentos XHTML/XForms en páginas HTML con Javascript, que sí entienden los navegadores actuales. Estas páginas gestionan, sin interactuar con el servidor, tanto la presentación como la lógica de la interfaz de usuario y su comunicación con el servidor se restringe al intercambio de datos utilizando técnicas AJAX. Otra aplicación es XSLTForms que transforma documentos XHTML/XForms en páginas XHTML con Javascript en los navegadores con XSLT.

1.1.2.1. FormFaces

Es un procesador puro de XForms implementado en JavaScript. Entre las principales características están:

- Permite el diseño/construcción de formularios
- Basado en el estándar XForms
- Basado en la tecnología AJAX
- Compatible con la mayoría de navegadores web: Internet Explorer, Mozilla, Firefox, Opera, Safari, etc.
- Independencia del lado del servidor: PHP, Java, .NET, etc.
- Funcionalidad autocompletar (validación as-you-type)
- Posibilidad de emular la apariencia del formato en papel (ej: PDF)
- Validación de datos en servidor

1.1.2.2. Orbeon

Es una solución que implementa el estándar XForms para el desarrollo de formularios Web. Probablemente se trate de la solución más completa y de la cuál sacaremos las ideas más interesantes. Orbeon maneja formularios complejos (validación de formularios, colecciones de datos). Numerosos organismos como

gobiernos, bancos o administraciones públicas usan este sistema para la gestión de sus formularios. Las características principales y de mayor importancia son:

- Permite el diseño/construcción de formularios
- Basado en el estándar XForms
- Basado en tecnología AJAX
- Compatible con la mayoría de navegadores web: Internet Explorer, Mozilla, Firefox, Opera, Safari, etc.
- Funcionalidad autocompletar (validación as-you-type)
- Posibilidad de emular la apariencia del formato en papel (ej: PDF)
- Validación de datos en servidor

1.1.2.3. Formul@

- Permite el diseño/construcción de formularios
- Lenguaje Java
- Utiliza Google Web Toolkit

1.1.2.4. Present@

- Permite el diseño/construcción de formularios
- Incorpora la funcionalidad de administración y gestión de la asociación formulario-procedimiento
- Posibilidad de emular la apariencia del formato en papel (ej: PDF)

1.1.2.5. Solicit@

- Permite el diseño/construcción de formularios
- Incorpora la funcionalidad de administración y gestión de la asociación formulario-procedimiento
- Validación de datos en servidor

A continuación damos una serie de valoraciones atendiendo a varios criterios (estratégicos, funcionales y técnicos) refiriéndonos a las aplicaciones software anteriormente comentadas.

	Tipo de Licencia	Coste versión Community	Coste versión Enterprise	Basado en fuentes abiertas	Ámbitos de la Adminis-tración a los que está dirigido	Comunidad Activa	Posibilidad de contratar soporte profesional	Coste soporte profesional	Documentación abierta al público	Cursos de formación abiertos al público	Numero de instalaciones del producto en la Administración española
FormFaces	BSD	0	No existe	Cumple / Soportado	Todos	Cumple / Soportado	No Disponible	No Disponible	Cumple / Soportado	Cumple / Soportado	No Disponible
Formul@	EUPL	0	No aplica	Cumple / Soportado	Todos	No Cumple /No Soportado	No aplica	No aplica	Limitada	No Cumple / No Soportado	No Disponible
Orbeom	LGPL	0	No existe	Cumple / Soportado	Todos	Cumple / Soportado	Desde 1995$/año	Desde 1995$/año<br	Cumple / Soportado	Cumple / Soportado	No disponible
Present@	EUPL	0	No aplica	No disponible	Todos	No Cumple / No Soportado	No aplica	No aplica	Limitada	No Cumple / No Soportado	No disponible
Solicit@	EUPL	0	No aplica	No Cumple / No Soportado	EELL	No Cumple / No Soportado	No aplica	No aplica	Limitadar>	No Cumple / No Soportado	>50

Figura 1.1. Criterios Estratégicos (*fuente: CENATIC*)

	Permite el diseño/construcción de formularios	Incorpora la funcionalidad de administración y gestión de la asociación formulario-procedimiento	Realiza validación de datos en el sevidor	Permite el relleno del formulario FromFaces y su envío posterior
FormFacesl	Cumple / Soportado	No Cumple / No Soportado	Cumple / Soportado	Cumple / Soportado
Formul@	Cumple / Soportado	No Cumple / No Soportado	No Cumple / NSoportado	No Disponible
Orbeom Forms	Cumple / Soportado	No Cumple / No Soportado	Cumple / Soportado	Cumple / Soportado
Present@	Cumple / Soportado	Cumple / Soportado	No Cumple / No Soportado	No Disponible
Solicit@	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	No Disponible

Figura 1.2. Criterios Funcionales (*fuente: CENATIC*)

	Cumple el estándar XForms del W3C	Posibilidad de emular la apariencia del formato en papel	Autocompletar (validación as-you-type)	Editor WYSIWYG	Admite instalación de alta disponibilidad (cluster)	Nombre de algunas herramientas/productos/componentes/con las que esta preparado de manera especial para integrarse
FormFaces	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	JBOSS JBPM
Formul@	No cumple / No soportado	No Cumple / No Soportado	No disponible	No disponible	Cumple / Soportado	Otros modulos Wand@(@aries,Trew@,..)@firma
Orbeom Forums	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	Cumple / Soportado	Italio BPMS
Present@	No cumple / No soportado	Cumple / Soportado	No disponible	Cumple / Soportado	No Cumple / No Soportado	Otros modulos Wand@(@aries,Trew@,..)@firma
Solicit@	No Cumple / No Soportado	No Cumple / No Soportado	No disponible	Cumple / Soportado	No Cumple / No Soportado	Otros modulos Wand@(@aries,Trew@,..)@firm

Figura 1.3. Criterios Técnicos (*fuente: CENATIC*)

1.1.3. Aplicaciones de ámbito comercial

1.1.3.1. Microsoft Infopath

Es una aplicación usada para desarrollar formularios de entrada de datos basados en XML. Su primera publicación, InfoPath 2003, fue como parte de Microsoft Office 2003. La principal característica de Infopath es la habilidad de poder crear y ver documentos XML con soporte para XML Schema. Infopath puede conectarse a sistemas externos usando servicios web XML.

1.1.3.2. Adobe Livecycle Forms

Se trata de un software desarrollado por Adobe para el desarrollo de formularios dinámicos en HTML5. Tiene características muy interesantes como el relleno de formularios offline o la mejora de tiempos de respuesta. También ofrece la posibilidad de exportar los formularios HTML5 a formato PDF.

Como podemos observar, existe una deficiencia de soluciones simples tanto de gestión como de construcción de formularios web para la tecnología .NET, donde la única alternativa viable podría ser Infopath. Este Trabajo Fin de Grado tiene la finalidad de cubrir esta carencia mediante el desarrollo de una aplicación web basada en .NET.

1.2. OBJETIVOS DEL TRABAJO DE FIN DE GRADO

1.2.1. Objetivos Finales

Tras estudiar con detenimiento el proyecto de investigación que se necesitaba y evaluar las aplicaciones que existen hoy en día en el mercado, se ha llegado a la conclusión de que se desarrollaría una aplicación web tanto para la gestión como para el desarrollo de formularios web HTML5 cuya funcionalidad sería mayor que en la

mayoría de las aplicaciones del mercado excepto ORBEON, que implementa demasiada funcionalidad y por lo tanto requeriría un tiempo excesivo.

Nuestra aplicación no serviría únicamente para el desarrollo e interpretación de formularios, sino también se gestionan una serie de usuarios que pueden registrarse en el sistema, una serie de portfolios pertenecientes a los usuarios y los propios formularios, pertenecientes a los portfolios que representan una plantilla de formulario.

El objetivo principal de este proyecto es por tanto crear una aplicación web que posibilite a los usuarios la creación de formularios dinámicamente. En los siguientes apartados se explicará más detenidamente y se aportará más información.

1.2.2. Consideraciones Iniciales

Durante las primeras fases del proyecto se estudió la posibilidad de implementar toda la funcionalidad que el cliente propuso. Tenemos que tener en cuenta que se necesitó realizar un estudio de forma que pudiéramos tener una referencia no sólo del tiempo aproximado del desarrollo de dicha funcionalidad, sino también del aprendizaje de implementación.
Una de la funcionalidad que no se ha considerado ha sido el relleno de formularios offline. Se llegó a un acuerdo con el cliente de que es una funcionalidad extra que podría suponer demasiado tiempo el integrarla en nuestra aplicación y que por lo tanto, se podría considerar como ampliación de la aplicación en un futuro.

Por otro lado, se propuso al cliente cierta funcionalidad que podría ser interesante para una aplicación web real, y es la gestión de usuarios y de portfolios que contengan formularios, de forma que pudiéramos realizar la gestión de formularios. Como es obvio, se trata de una funcionalidad más importante y por lo tanto, tras hablarlo con el cliente, aceptó.

1.3. ESTRUCTURA DE LA DOCUMENTACIÓN

1.3.1. Metodología UPEDU

UPEDU (*Unified Process for EDUcation*) es una metodología de ingeniería del software especializada para la educación, desarrollada por Pierre-N. Robillard y Patrick d'Astous de la Escuela Politécnica de Montreal y Philippe Kruchten de *Rational Software*. Es una adaptación del proceso unificado de Rational (RUP, *Rational Unified Process*) y por tanto mantiene sus propiedades más destacadas, como son:

- Es iterativa y una implementación del modelo en espiral.
- El proceso se divide en el tiempo en cuatro fases: inicio, elaboración, construcción y transición, cada una consistente en una o más iteraciones. Para cada fase el proceso define el conjunto de artefactos (o entregables) y el flujo de trabajo.
- Las actividades están organizadas en disciplinas. En UPEDU éstas son: requisitos, análisis y diseño, implementación, pruebas, gestión de configuraciones y cambios, gestión del proyecto.

Su meta es asegurar la producción de software de calidad que cubra las necesidades de sus usuarios finales con una planificación y presupuesto predeterminados. Es esencial guiar el desarrollo de un proyecto software con una metodología adecuada, y hemos decidido utilizar UPEDU para el desarrollo de XGen porque es la que mejor se ajusta a nuestras necesidades. En la sección de *Gestión del Proyecto* se puede encontrar más información al respecto.

2. GESTIÓN DEL PROYECTO

2.1. PLAN DE DESARROLLO SOFTWARE

2.1.1. Introducción

2.1.1.1. Propósito

El propósito del *Plan de Desarrollo Software* es recoger la información necesaria para gestionar el proyecto. Describe a grandes rasgos lo que va a ser el desarrollo del software y es el plan de más alto nivel, que será utilizado para dirigir el esfuerzo de desarrollo.

Los usuarios del *Plan de Desarrollo Software* son normalmente los **jefes de proyecto**, quienes lo utilizan para planificar el proyecto y los recursos necesarios, además de realizar un seguimiento para comprobar que se cumplen los tiempos previstos, y los **miembros del equipo de desarrollo, quienes lo usan para entender lo que deben hacer, cuándo deben hacerlo y de qué otras actividades dependen.**

En este caso en concreto, diferenciamos dos usuarios:

- El **alumno** encargado de realizar el proyecto, quien elaborará el plan y realizará un seguimiento a lo largo de todo el proceso (reúne los dos roles explicados anteriormente).
- El **tutor** del proyecto, quien tendrá que dar su visto bueno para poder comenzar la realización de sucesivas fases.

2.1.1.2. Alcance

El presente apartado describe el plan general usado para el desarrollo del proyecto "Generador de Formularios HTML5", un generador de formularios dinámicos HTML5. Los detalles individuales de cada iteración se describirán en los documentos *Planes de Iteración* (CD-ROM). El plan descrito en este apartado se basa en los requisitos del producto contemplados en el documento *Visión* (CD-ROM).

2.1.1.3. Estructura del apartado

Después de esta introducción, el resto del apartado está organizado en las siguientes secciones:

Vista General del Proyecto	Proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los entregables que serán producidos y utilizados durante el mismo.
Organización	Describe la estructura organizativa del equipo de desarrollo.
Proceso de Gestión	Explica los costes y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.

Tabla 2.1 - Estructura del apartado

2.1.2. Vista General del Proyecto

2.1.2.1. Propósito, Alcance y Objetivos

El propósito principal de este proyecto es el de desarrollar un gestor de formularios dinámicos basado en HTML5.

Según la información disponible en la wiki de *cenatic* y como hemos comentado anteriormente:

- *La oferta open source de productos para la gestión de formularios no es muy abundante, encontrándose con frecuencia circunscrita al ámbito de la construcción de formularios.*
- *Recientemente han aparecido con mucha fuerza algunas iniciativas en el campo de las fuentes abiertas para la construcción de formularios. Y, como suele suceder con la tecnología de fuentes abiertas, vienen de la mano de un estándar que está ganando gran aceptación en el mercado:* ***XForms.***
- *Análogamente a lo que sucede en el caso de otros elementos funcionales, existen en el mercado soluciones paquetizadas que integran entre sus funcionalidades servicios de gestión de formularios.*

Los generadores de formularios son herramientas útiles para agilizar el proceso de captura/presentación de la información del modelo de datos enfatizándose en la velocidad y aumento de la productividad.

2.1.2.2. Suposiciones y Restricciones

Como en cualquier proyecto informático, habrá que seguir una metodología que marque las distintas fases y proporcione disciplina a la realización. La más extendida es RUP, pero por su complejidad no conviene emplearla en proyectos relativamente sencillos, ya que puede resultar contraproducente (aumenta la duración del proyecto más de lo necesario). Por ello, se ha decidido utilizar UPEDU, la cual es notablemente más sencilla sin dejar de ser eficiente, y es bastante apropiado utilizarla ya que se ideó precisamente con motivos educativos. Además, el equipo de desarrollo ya está familiarizado con ella, lo que puede ahorrar tiempo.

El equipo de desarrollo está compuesto por una única persona, el alumno encargado del proyecto, el cual asumirá los roles que sean necesarios en las distintas etapas del proceso.

Este proyecto ha surgido debido a la deficiencia de soluciones simples de gestión de formularios en la tecnología .NET donde la única alternativa viable no es suficientemente potente. HP será una parte fundamental del proyecto, pues deberán validar las diferentes fases desde el punto de vista puramente funcional. Es requisito imprescindible que la aplicación dé soporte a los intereses de dicha organización.

Se supone que la aplicación debe ser desarrollada siguiendo la propuesta .NET que es ampliamente conocido. Como es evidente, la aplicación debería funcionar sin ninguna dificultad sobre la plataforma Windows.

Las herramientas que se empleen en la realización del proyecto deberán ser de dominio público (Freeware, OpenSource, etc.) o se poseerán las licencias pertinentes.

2.1.2.3. Entregables

A continuación se indican y describen cada uno de los entregables generados y utilizados en el proyecto. Esta lista se basa principalmente en la proposición de UPEDU de artefactos a crear.

Como dichos artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada fase y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

- **Gestión del Proyecto**
  - Plan de Desarrollo Software
    - Planes de Iteración
      - Diagramas de Gantt
  - Plan de Medidas
  - Medidas del Proyecto
  - Lista de Riesgos

- **Requisitos**
  - Visión
  - Glosario
  - Especificación de Requisitos Software
  - Especificación de Casos de Uso
  - Prototipo de Interfaz de Usuario

- **Análisis y Diseño**
  - Modelo de Análisis
    - Diagrama de Clases de Análisis
    - Diagramas de Secuencia
  - Documento de Arquitectura Software
    - Diagrama de Clases de Diseño
  - Realización de Casos de Uso
    - Diagramas de Secuencia

- **Implementación**
  - Modelo de Implementación

- **Pruebas**

- Plan de Pruebas
- Casos y Resultados de las Pruebas

2.1.2.4. Evolución del Plan de Desarrollo Software

El *Plan de Desarrollo Software* se revisará y actualizará siempre que sea necesario. Esto suele hacerse una vez por semana, pero para equipos de desarrollo grandes. Puesto que no es el caso, seguramente se haga cada dos semanas, coincidiendo en algún caso con la finalización de las iteraciones en que se divide el proyecto.

2.1.3. Organización

2.1.3.1. Estructura Organizativa

Al tratarse de un *Trabajo de Fin de Grado* realizado por un alumno individualmente, la estructura es de tutor-alumno. Tal y como se explicó brevemente, el tutor supervisará el trabajo del alumno y será necesaria la conformidad de éste al finalizar una fase o iteración para poder continuar con la siguiente.

El trabajo lo desarrollará el alumno, ejerciendo en cada momento el rol que sea necesario. La naturaleza de estos roles se explica a continuación.

2.1.3.2. Roles y Responsabilidades

A continuación asociaremos los diferentes roles que existen en la metodología UPEDU con sus responsabilidades:

Analistas	Coordinan la especificación de requisitos y el modelo de casos de uso organizando metódicamente la funcionalidad del sistema y delimitándolo. Por ejemplo, establece qué actores y casos de uso existen y cómo interaccionan entre sí. Los roles pueden ser el Analista de procesos de negocio, diseñador de negocio, analista de sistema y especificador de requisitos.
Desarrolladores	Definen responsabilidades, operaciones, atributos y relaciones para uno o más componentes, y determina cómo serán adaptados al entorno de desarrollo. Los roles pueden ser el arquitecto de software, el diseñador de interfaz de usuario, el diseñador de bases de datos, el implementador y el integrador.
Organizador	Supervisan el proceso de desarrollo. Son las personas que tiene la responsabilidad total o parcial del planeamiento y ejecución del proyecto. Los roles pueden ser el jefe de proyecto, jefe de control de cambios, jefe de configuración, jefe de pruebas, jefe de despliegue, revisor de gestión de proyectos y el gestor de pruebas.
Probadores	Son los responsables de las principales actividades de prueba, que comprende realizar las pruebas necesarias y guardar constancia de los resultados. Incluye: o Identificar la mejor forma de implementar una prueba. o Implementar pruebas individuales. o Preparar y ejecutar las pruebas. o Anotar los resultados y verificar el funcionamiento. o Analizar y reparar errores de ejecución.
Documentadores	Son los responsables de todo el proceso documental a lo largo del ciclo de vida del proyecto. Se ocupa de divulgar el proceso de análisis, desarrollo, construcción e implementación de un sistema software.

Jefe de Proyecto	Asigna recursos, establece prioridades, coordina la interacción entre clientes y usuarios y, en general, mantiene al equipo concentrado en el objetivo final. También establece un conjunto de pautas o 'buenas prácticas' que garanticen la integridad y calidad de los entregables del proyecto.

Tabla 2.2 - Roles y Responsabilidades

2.1.4. Proceso de Gestión

2.1.4.1. Estimaciones del Proyecto

El coste estimado para el proyecto no se puede medir como se procedería en un entorno real, ya que se trata de un Proyecto con un fin académico. Sin embargo, podríamos hacer una aproximación en lo que respecta al equipo de desarrollo teniendo en cuenta que a un Ingeniero Técnico Informático se le contrata por un salario base de 22.777,04€ brutos por el ejercicio de una jornada de trabajo ininterrumpida de 7 horas diarias. Además se establecen 23 días de vacaciones anuales.

No se incurrirá en el coste que supondría adquirir alguna licencia informática para el desarrollo del proyecto, siempre se buscarán herramientas gratuitas y/o de código libre. Sobre el resto de costes, en principio, todos indirectos (luz, material de oficina, etc.), no lo consideraremos a priori.

Puesto que se pretende presentar este proyecto para su evaluación en el mes de Junio, se ha establecido como fecha límite la tercera semana de dicho mes (exactamente el 21 de Junio). Eso nos deja con 129 días, incluyendo laborales y no laborales, durante los cuales se pretende trabajar 20 horas semanales (4 horas/día).

Teniendo en cuenta el sueldo bruto calculado anteriormente, para la realización del proyecto habría que destinar 62,40 € diarios. En nuestro caso, debido a la reducción de la jornada laboral, serían 53,48 € diarios (8,91€/h). Según lo planificado el proyecto durará 89 días (laborables), por lo tanto el presupuesto resultaría un total de 4759,72 €.

Las horas extraordinarias realizadas se compensarán, con carácter general, por tiempo de descanso retribuido, a razón de 1,68 por hora trabajada (15 €/h). No se tendrá en cuenta, a efectos de la duración máxima de la jornada ordinaria laboral, el exceso de horas trabajadas para prevenir, reparar siniestros u otros daños extraordinarios y urgentes.

2.1.4.2. Plan de Proyecto Inicial

2.1.4.2.1. Plan de Fases

En la página siguiente se muestra un diagrama de Gantt con la disposición y duración de las cuatro fases en que se divide el proyecto, tal y como dicta el estándar de UPEDU. Se va a seguir un esquema iterativo-incremental, realizando una sola iteración para la fase de inicio, dos iteraciones en la fase de elaboración, otras dos iteraciones en la fase de construcción y una última iteración en la fase de transición. En el siguiente apartado se describen los objetivos principales que deben haberse conseguido al finalizar cada una de ellas.

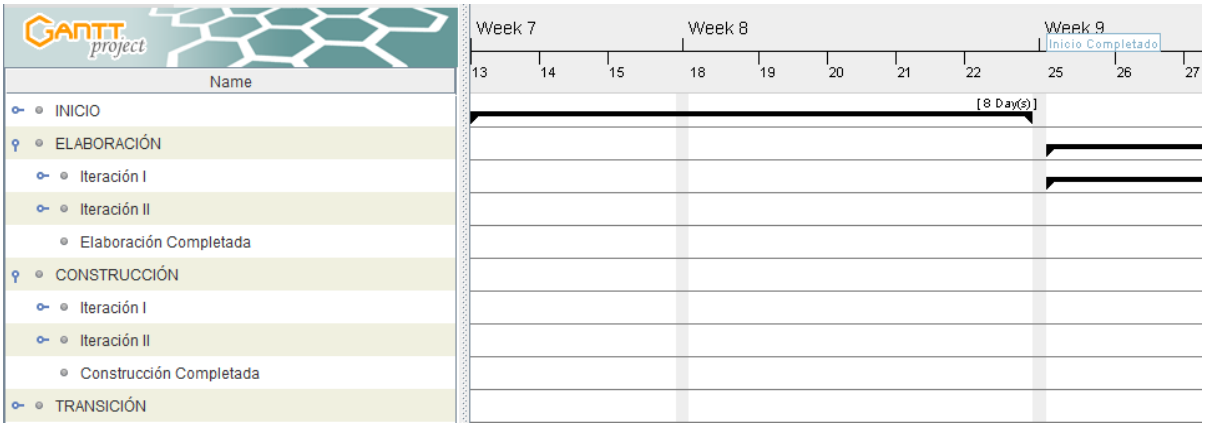

Ilustración 2.1 - Primera Fase, del 13 de Febrero al 27 de Febrero

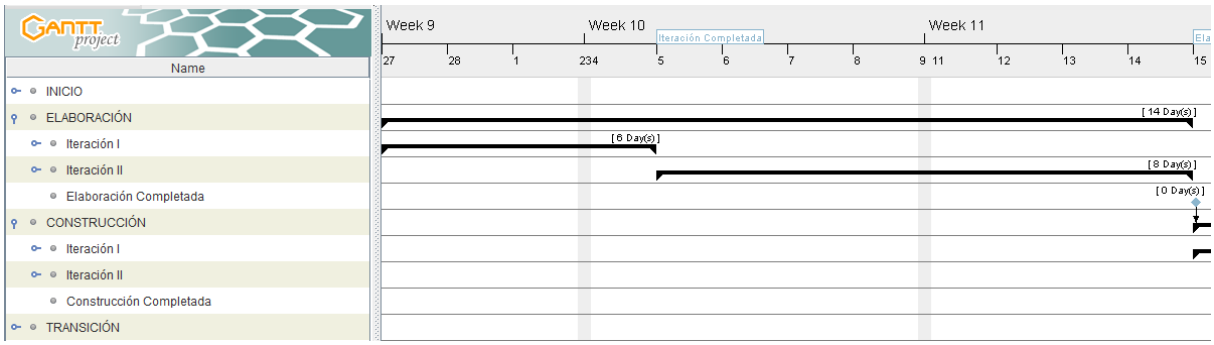

Ilustración 1.2 - Segunda Fase, del 27 de Febrero al 15 de Marzo

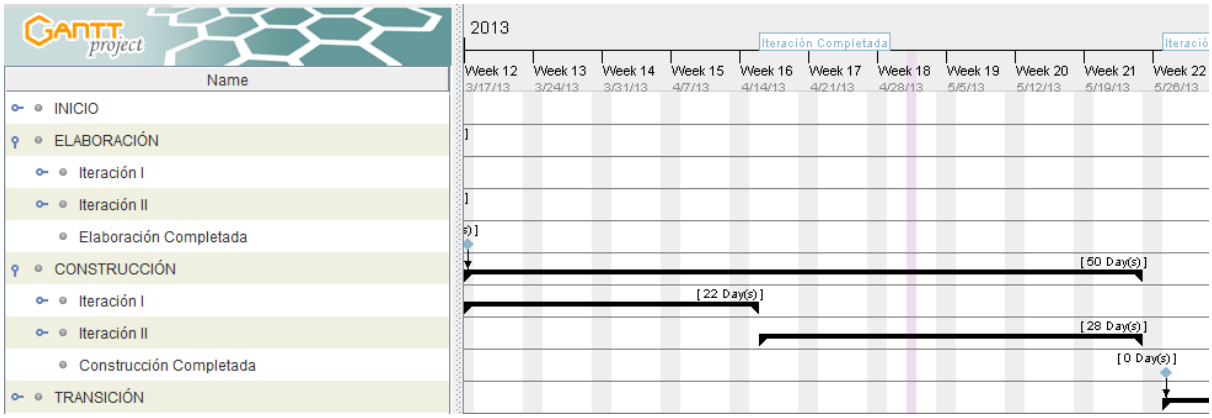

Ilustración 2.2 - Tercera Fase, del 15 de Marzo al 27 de Mayo

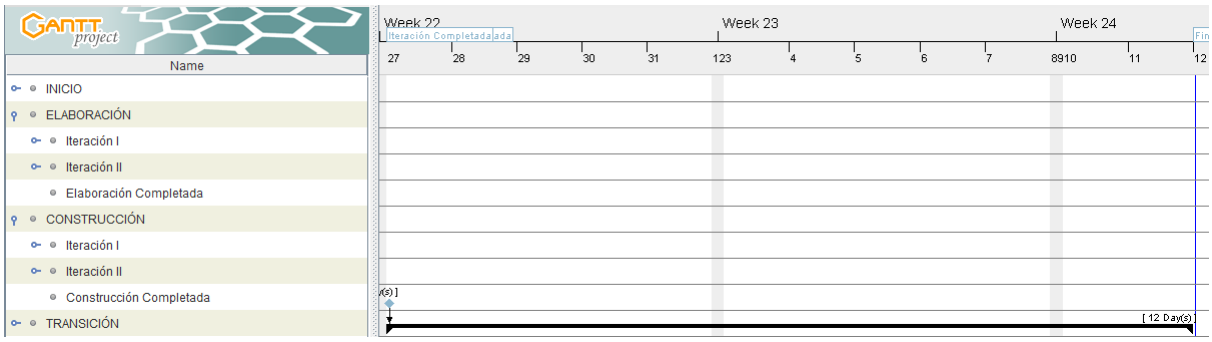

Ilustración 2.3 - Cuarta Fase, del 27 de Mayo al 12 de Junio

2.1.4.2.2. Objetivos de cada Iteración

Inicio	Versión inicial de las actividades de gestión del proyecto (plan de fases, planes de iteración, lista de riesgos, plan de medidas), documentos de definición del proyecto (glosario, visión), requisitos del sistema y perfil inicial de los casos de uso.
Elaboración (Iteración I)	Documento actualizado de requisitos y especificación de casos de uso para poder comenzar a modelar desde la perspectiva del analista. Diagrama de Clases de Análisis, Arquitectura del Sistema (Inicial) y Prototipo de interfaz de usuario.
Elaboración (Iteración II)	Ampliar el documento de casos de uso creado en la iteración anterior para obtener la realización de casos de uso. Definir totalmente la arquitectura del sistema. Diseñar la aplicación en base a las

	posibilidades que ofrezca el lenguaje de programación. Completar el documento de interfaz de usuario y crear un plan de pruebas inicial.
Construcción (Iteración I)	Desarrollar el modelo de implementación. Construir una primera versión de la aplicación que implemente una parte de la funcionalidad total (casos de uso principales).
Construcción (Iteración II)	Implementar el resto de funcionalidad del sistema y probarlo. Integrar ambas construcciones en una versión final. En este momento será necesario bastante esfuerzo para la recogida de medidas.
Transición	Recopilar los resultados de las pruebas, contrastarlos y corregir los defectos restantes. Iterar hasta no detectar errores. Lanzar la versión definitiva de la aplicación (Demo), crear los manuales y preparar toda la documentación para ser encuadernada en formato B5.

Tabla 2.3 - Objetivos de cada Iteración

2.1.4.2.3. Planificación

Ver el apartado *Planes de Iteración*.

2.1.4.2.4. Recursos

Se dispone de una sola persona para realizar el proyecto, la cual desempeñará todos los roles citados anteriormente. El rol que corresponde a cada actividad se puede ver en el documento *Plan de Iteración* correspondiente (CD-ROM), junto con la descomposición de cada iteración.

Se pretende desarrollar el proyecto utilizando el lenguaje de programación C#, de los cuales se tiene un conocimiento medio, pero posiblemente no suficiente. Por lo tanto habrá que dedicar tiempo a perfeccionar su entendimiento antes de comenzar con la parte de implementación, posiblemente al mismo tiempo que se desarrolle el diseño.

2.1.4.3. Monitorización y Control

2.1.4.3.1. Gestión de Requisitos

Los requisitos del sistema se recogen en el documento de *Visión* (CD-ROM). Los cambios que han sufrido los requisitos se han reflejado en los documentos de *Visión* (CD-ROM) y de *Especificación de Requisitos Software* (CD-ROM).

2.1.4.3.2. Control de Calidad

A medida que se descubran defectos en las especificaciones, se crearán nuevas versiones de los documentos corrigiéndolos y se subirán a un repositorio común.

Todos los entregables serán revisados a su finalización por el tutor del proyecto, para asegurar que la metodología se ha aplicado correctamente y, por tanto, son de una calidad suficiente.

2.1.4.3.3. Informes y Medidas

Para realizar el seguimiento del proyecto se usarán medidas del esfuerzo y del tiempo. Se contrastará lo planificado junto con el tiempo real empleado para medir el progreso.

Para más información, ver el apartado *Plan de Medidas*.

2.1.4.3.4. Gestión de Riesgos

Los riesgos se identificarán durante la fase de Inicio y se realizará un control de ellos durante todo el ciclo de vida del proyecto. Los riesgos se revisarán y documentarán al menos una vez por iteración.

Para más información, ver el apartado *Lista de Riesgos*.

2.1.4.3.5. Gestión de Configuraciones

Puesto que el equipo de desarrollo lo compone una única persona no se considera necesario utilizar una herramienta de gestión de configuraciones.

De todas formas se utilizará una herramienta de control de versiones (*Assembla*) para informar a las diferentes

partes interesadas y de esta forma hacer más visible si cabe el desarrollo del proyecto.

El alumno seguirá una nomenclatura a la hora de guardar versiones de los diferentes documentos y del código fuente:

Documentos:

Se guardará cada documento con el nombre especificado en la lista de entregables del apartado 2.3 seguido de su versión, para así conservar todas las versiones anteriores. Ejemplo:

Plan de Desarrollo Software 1.03.docx

Código fuente:

Se guardará una copia estable (*commit*) de la aplicación al finalizar cada semana durante la fase de construcción. Se hará una breve descripción de las diferencias con respecto a la anterior versión.

El propio alumno acepta, ejecuta y valida los cambios que surjan (incluso podría llegar a sugerir alguno de ellos), luego no es necesario un gestor de cambios.

2.1.5. Plan de Proyecto Final

2.1.5.1. Tiempo

El primer aspecto que llama la atención es la diferencia que existe entre el tiempo que se estimó que iba a ser necesario dedicarle al proyecto y el tiempo en realidad ha sido necesario dedicarle. El periodo total de desarrollo del proyecto ha sido desde el 13 de Febrero del 2013 hasta el 5 de septiembre del mismo año, es decir, 129 días + 76 días = 205 días. Ha sido necesario un 58,9% más de tiempo del previsto. Los motivos que han desencadenado esto se encuentran detallados en la *Lista de Riesgos*.

2.1.5.2. Coste

Para este apartado hay que tener en cuenta dos aspectos: el tiempo añadido que ha llevado el proyecto y las horas extra que han sido necesarias para evitar mayores retrasos.

Si multiplicamos el valor del proyecto estimado (4759,72 €) por 1,589 (que es el valor adicional), resulta 7563,19 € (tenemos que añadir también las horas extras que las contabilizamos a 15 €/h).

2.2. PLAN DE MEDIDAS

2.2.1. Introducción

2.2.1.1. Propósito

El propósito del *Plan de Medidas Software* es especificar las métricas básicas que se van a recoger y las medidas que se van a calcular a lo largo del proyecto. Se debe especificar un conjunto de objetivos de las medidas para controlar el progreso del proyecto.

2.2.1.2. Alcance

Este apartado define un programa de medidas muy simple, que incluye los objetivos que se pretenden conseguir, las medidas que se van a obtener para ello, y las métricas básicas a partir de las cuales se van a calcular para hacer un seguimiento del proyecto. Los detalles de cada actividad en particular se describen en los *Planes de Iteración*. El plan descrito aquí se basa en los requisitos recogidos en el apartado *Especificación de Requisitos Software*.

2.2.1.3. Estructura del apartado

Objetivos de las Medidas	Los objetivos que se quieren alcanzar con la toma de medidas, respecto a completitud, mejora y calidad del proyecto.
Medidas	Las medidas que habrá que calcular en ciertos puntos del proyecto para conseguir los objetivos.
Métricas Básicas	Las métricas que habrá que recoger manual o automáticamente para poder calcular las medidas.

Tabla 2.4 - Estructura del apartado

2.2.2. Objetivos de las Medidas

Para asegurar la calidad del producto, una buena práctica es definir una serie de medidas que deberán tomarse a lo largo del ciclo de vida del proyecto de modo que se pueda evaluar en cada fase los resultados obtenidos en relación con los esperados y tomar las decisiones necesarias en caso de inconsistencias.

Los objetivos del Plan de Medidas para este sistema son los siguientes:

- **EVALUAR**:
  - La calidad del producto.
  - La calidad del código.
  - La calidad de los documentos y de los modelos.
- **ASEGURAR:**
  - Un seguimiento documentado del proyecto.
  - El progreso de las tareas en comparación con lo estimado.
- **FACILITAR**:
  - La planificación y estimación de nuevos proyectos.

2.2.3. Medidas

2.2.3.1. Del Producto

El producto está compuesto por documentos, modelos y código fuente. A continuación se enumeran algunas de sus características con sus medidas:

	Característica	Medida
Código	Tamaño	SLOC (Source Lines Of Code)
Documentos	Tamaño	Total de páginas
Modelo	Tamaño	Número de Casos de Uso Número de Actores Número de Clases Métodos por Clase Atributos por Clase

Tabla 2.5 - Medidas de Producto

2.2.3.2. Del Proceso

Para un seguimiento exhaustivo del proceso hay que tomar medidas de las actividades al nivel más bajo que se haya planificado. Se guardará un registro con el tiempo dedicado a cada una de las actualizaciones del plan que se hayan realizado.

Medida	Comentarios
Esfuerzo	Unidades de tiempo por personal (horas/hombre).
Duración	Tiempo dedicado a la actividad.
Artefactos	Número de artefactos que se han creado durante el proceso de desarrollo del producto (nótese que no todos formarán parte de él).

Tabla 2.6 - Medidas de Proceso

2.2.3.3. Medidas

Nombre	Duración
Definición	La duración se calculará en base a la hora de comienzo y finalización de cada actividad. Se calculará en horas.
Objetivo	Midiendo la duración de las actividades podremos calcular el esfuerzo, tanto para lo concerniente al producto como para el proceso.
Procedimiento de Recogida	Se guardará registro del tiempo dedicado a cada actividad.
Responsabilidad	Cada miembro (en este nuestro caso será una persona únicamente) será responsable de guardar constancia de ello.

Tabla 2.7 - Medida: Duración

Nombre	Esfuerzo
Definición	Total de horas dedicadas al proyecto por el total del personal.
Objetivo	Calcular el esfuerzo empleado en comparación con el estimado. Servirá para futuras estimaciones.
Procedimiento de Recogida	Se calcula a partir de las medidas de la duración de las tareas del proyecto.
Responsabilidad	Jefe de proyecto.

Tabla 2.8 - Medida: Esfuerzo

Nombre	SLOC
Definición	Source Line Of Code
Objetivo	Calcular el tamaño del programa en base a la cantidad de líneas de código
Procedimiento de Recogida	Con ayuda del IDE (Entorno de Desarrollo), contar las líneas totales de cada fichero
Responsabilidad	Desarrollo software

Tabla 2.9 - Medida: SLOC

2.2.4. Anexos

Para guardar constancia del tiempo dedicado a cada actividad por cada miembro del equipo se seguirá el siguiente formato:

Responsable	*Actividad*	*Artefacto*	*Fecha*	*Hora Inicio*	*Hora Fin*	*Duración*

Para ello se utilizará una plantilla creada con el programa Microsoft® Office Excel 2010 con el nombre *Seguimiento de las Actividades.xls* (Incluido en el CD-ROM)

2.3. MEDIDAS DE PROYECTO

2.3.1. Medidas de Proceso

Para un seguimiento exhaustivo del proceso hay que tomar medidas de las actividades al nivel más bajo que se haya planificado. Se dispone de un registro con el tiempo dedicado a cada una de las actualizaciones del plan que se hayan realizado, tal y como se indica en el apartado *Anexos* del apartado *Plan de Medidas*.

Este fichero se ha actualizado cada vez que se realizaba una actividad relativa al desarrollo del proyecto de forma que se poseen datos reales respecto a la duración del esfuerzo realizado:

Medida	Resultado
Esfuerzo: Fase Inicio	43h (8%)
Esfuerzo: Fase Elaboración I	33h (6%)
Esfuerzo: Fase Elaboración II	44h (8%)
Esfuerzo: Fase Construcción I	124h (22%)
Esfuerzo: Fase Construcción II	245h (44%)
Esfuerzo: Fase Transición	68h (12%)
Total de Esfuerzo:	557h
Total de Artefactos:	**25**

Tabla 2.10 - Duración/Esfuerzo

2.3.2. Medidas de Producto

El producto está compuesto por código fuente, documentos, modelos y la aplicación en sí misma. A continuación se enumeran las medidas globales realizadas al finalizar el proyecto.

Medida	Resultado
Código: SLOC	7275
Documentos: Páginas	259
Modelo: Casos de uso	10
Modelo: Actores	1
Modelo: Clases	26
Modelo: Métodos por clase (media)	2,71
Modelo: Atributos por clase (media)	1,69

Tabla 9.11 - Medidas de Producto

NOTA: La medida de aplicación (en concreto el *Tiempo de Respuesta*) toma más importancia a la hora de la realización de los test de la aplicación, de modo que lo reflejaremos en los apartados *Casos y Resultados de las Pruebas*.

2.3.3. Distribución del Tiempo

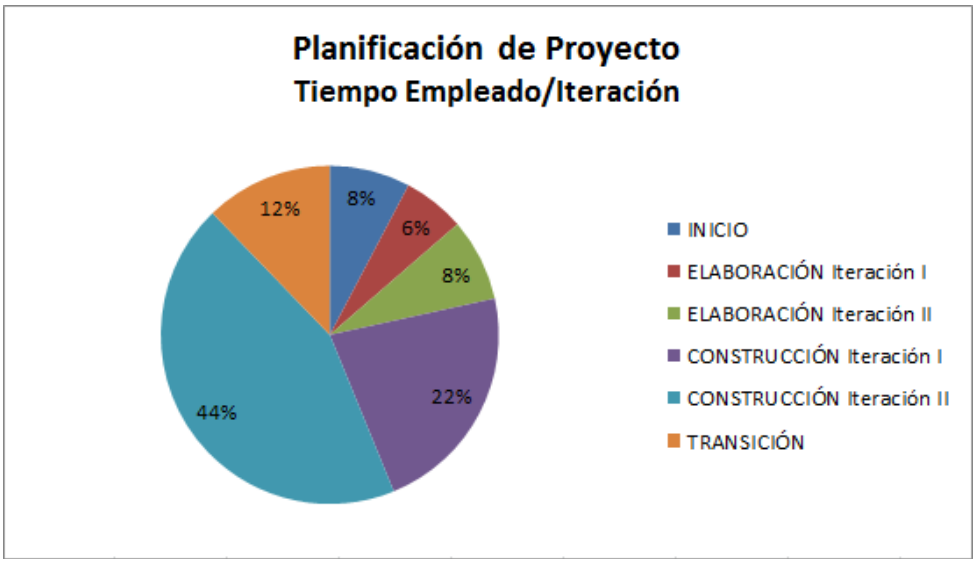

Ilustración 2.4 - Distribución de Tiempo/Iteración

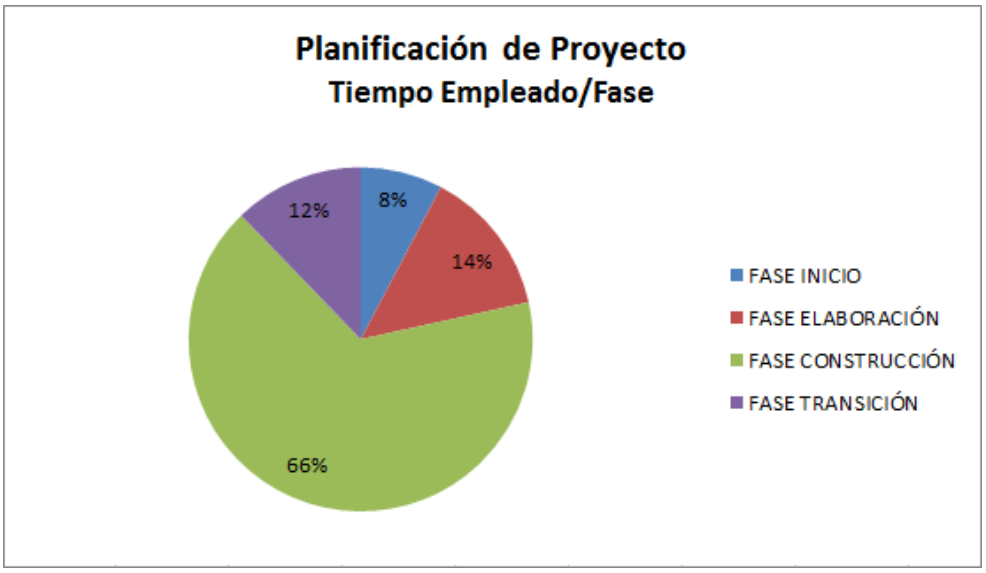

Ilustración 2.5 - Distribución de Tiempo/Fase

2.4. LISTA DE RIESGOS

2.4.1. Introducción

2.4.1.1. Propósito

Este apartado está pensado para recoger los riesgos percibidos que puedan poner en peligro el éxito u objetivos del proyecto. Un riesgo informático se podría definir como la ausencia de seguridad en el procesamiento automático de datos. A continuación identificaremos los eventos que puedan llevar a un resultado significativamente negativo. Sirve de guía para ciertas actividades del proyecto y es la base alrededor de la cual se organizan las iteraciones.

2.4.1.2. Alcance

La *Lista de Riesgos* se actualiza constantemente a lo largo del ciclo de vida del proyecto. Se crea al principio de la fase de inicio y se mantiene según van surgiendo nuevos riesgos o eliminándose los viejos.

Se revisará, por lo menos, al término de cada iteración para verificar la misma, y se añadirá un apartado nuevo describiendo los riesgos que afectaron al proyecto, en qué medida afectaron y la acción realizada para disminuir sus efectos. Se pretende que esta información sirva de apoyo para futuros proyectos.

2.4.1.3. Estructura del apartado

Primero se enumerarán los riesgos percibidos con una breve descripción de cada uno. A continuación se organizará una tabla para cada uno, la cual contendrá su identificador, categoría, nivel de impacto, descripción del impacto, probabilidad, estrategia a aplicar y el plan de contingencia ideado.

2.4.2. Riesgos

Los riesgos software los podemos clasificar en tres categorías:

- *De proyecto*: Restricciones de recursos, interfaces externas, relaciones con los proveedores, políticas internas, problemas de coordinación interna del equipo o del grupo, financiación no adecuada...
- *De proceso*: Proceso software no documentado, falta de revisiones efectivas, no prevención de defectos, proceso de diseño pobre, gestión pobre de requisitos, planificación ineficaz...
- *De producto*: Falta de experiencia en el dominio, diseño complejo, interfaces definidas deficientemente, sistemas de legado poco comprendidos, requisitos vagos o incompletos...

2.4.2.1. <RSK-01> Escaso Tiempo Disponible

<RSK-01> Escaso tiempo disponible					
Categoría	**Impacto**	**Descripción del Impacto**	**Probabilidad**	**Estrategia**	**Plan de Contingencia**
Proyecto	Crítico	O bien se crea un producto de baja calidad, o bien sobrepasamos la restricción temporal.	Alta	Reducción	Dedicar horas extra para cumplir con la fecha límite.

2.4.2.2. <RSK-02> Pérdida de Artefactos ya completados

<RSK-02> Pérdida de artefactos ya completados					
Categoría	**Impacto**	**Descripción del Impacto**	**Probabilidad**	**Estrategia**	**Plan de Contingencia**
Proceso	Catastrófico	Retraso en la planificación por tener que rehacer actividades.	Muy Baja	Evitación	Creación de copias de seguridad periódicamente con redundancia en dispositivos externos.

2.4.2.3. <RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar

<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar					
Categoría	**Impacto**	**Descripción del Impacto**	**Probabilidad**	**Estrategia**	**Plan de Contingencia**
Producto	Crítico	Retraso en el desarrollo del producto.	Muy Alta	Reducción	Formarse en dichas tecnologías en horario no laboral.

2.4.2.4. <RSK-04> Desarrollo de artefacto incorrecto

<RSK-04> Desarrollo de artefacto incorrecto					
Categoría	**Impacto**	**Descripción del Impacto**	**Probabilidad**	**Estrategia**	**Plan de Contingencia**

Producto	Crítico	Retraso en Desarrollo	Media	Reducción	Análisis del artefacto para encontrar la inconsistencia y corrección.

2.4.2.5. <RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema

<RSK-05>	**Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema**				
Categoría	**Impacto**	**Descripción del Impacto**	**Probabilidad**	**Estrategia**	**Plan de Contingencia**
Proyecto	Crítico	Podría suponer no avanzar en el proyecto.	Baja	Reducción	Mantenerlo informado en todo momento y consultar dudas con antelación.

2.4.3. Clasificación

2.4.3.1. Inicio

Prioridad	**Código del Riesgo**
1	**<RSK-01> Escaso tiempo disponible**
2	**<RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema**
3	<RSK-02> Pérdida de artefactos ya completados
4	<RSK-04> Desarrollo de artefacto incorrecto
5	<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar

2.4.3.1.1. Informe de Riesgos Producidos

Durante esta iteración se ha producido el siguiente riesgo:

- <RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema: Se trata de la no disponibilidad del cliente para resolver cualquier tipo de duda respecto a las especificaciones del sistema. Es el único riesgo producido al final de esta etapa del proyecto.

2.4.3.2. Elaboración, Iteración I

Prioridad	**Código del Riesgo**
1	**<RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema**
2	**<RSK-01> Escaso tiempo disponible**
3	<RSK-02> Pérdida de artefactos ya completados

4	<RSK-04> Desarrollo de artefacto incorrecto
5	<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar

2.4.3.2.1. Informe de Riesgos Producidos

Durante esta iteración se ha producido el siguiente riesgo:

- <RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema: Se trata del mismo riesgo de forma que se ha producido un pequeño retraso en el avance del proyecto debido a que algunas cuestiones de las cuales se pretendía hablar eran críticas para el desarrollo del sistema.

2.4.3.3. Elaboración, Iteración II

Prioridad	Código del Riesgo
1	**<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar**
2	**<RSK-01> Escaso tiempo disponible**
3	<RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema
4	<RSK-02> Pérdida de artefactos ya completados
5	<RSK-04> Desarrollo de artefacto incorrecto

2.4.3.3.1. Informe de Riesgos Producidos

En esta fase del proyecto, se han producido los siguientes riesgos:

- <RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar: Se ha dedicado un tiempo excesivo a la búsqueda de información de un método de implementación del sistema. Era de esperar debido a que el conocimiento respecto la tecnología usada para la implementación no era muy avanzado.

- <RSK-01> Escaso tiempo disponible: ya que efectivamente, tal y como se había previsto, la planificación era muy ajustada debido al poco tiempo del que se disponía, y algunos artefactos han requerido más tiempo para poder ser terminados.

2.4.3.4. Construcción, Iteración I

Prioridad	Código del Riesgo
1	**<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar**
2	**<RSK-01> Escaso tiempo disponible**
3	<RSK-04> Desarrollo de artefacto incorrecto
4	<RSK-02> Pérdida de artefactos ya completados

5	<RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema

2.4.3.4.1. Informe de Riesgos Producidos

En esta fase del proyecto, se han producido los siguientes riesgos:

- <RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar: De nuevo se ha producido este riesgo debido a que son éstas etapas las iniciales respecto a la implementación y por lo tanto de búsqueda de información.

2.4.3.5. Construcción, Iteración II

Prioridad	Código del Riesgo
1	**<RSK-01> Escaso tiempo disponible**
2	**<RSK-02> Pérdida de artefactos ya completados**
3	**<RSK-04> Desarrollo de artefacto incorrecto**
4	<RSK-03> Dedicar demasiado tiempo en aprender las tecnologías que se pretenden usar
5	<RSK-05> Indisponibilidad del cliente para resolver dudas respecto a las especificaciones o requisitos del sistema

2.4.3.5.1. Informe de Riesgos Producidos

En esta fase del proyecto, se han producido los siguientes riesgos:

- <RSK-01> Escaso tiempo disponible. De nuevo en esta etapa se ha producido este riesgo debido a que la planificación ha sido muy ajustada y se ha necesitado más tiempo del estipulado.

2.4.3.6. Transición

En esta etapa no se ha producido ningún tipo de riesgo. Hemos finalizado los entregables de forma satisfactoria.

3. REQUISITOS

3.1. ESPECIFICACIÓN DE REQUISITOS SOFTWARE (SRS)

3.1.1. Introducción

3.1.1.1. Propósito

El propósito de este apartado es proveer de una especificación de requisitos de software que estudia y desarrolla un generador de formularios dinámicos HTML5. El objetivo de este apartado es describir el comportamiento externo de la aplicación. Se describirán los requisitos funcionales y no funcionales, restricciones de diseño, y otros factores necesarios para proporcionar una descripción completa y comprensible de los requisitos del software.

3.1.1.2. Alcance

Toma las especificaciones generales del documento *Visión* (CR-ROM) y las traduce en especificaciones de requisitos formales, identificando individualmente cada uno de ellos y clasificándolo según los estándares de la metodología a seguir.

A partir de este apartado se especificarán tanto el Modelo de Casos de Uso como el diagrama de clases y los modelos dinámicos de la parte de análisis, lo que sentará las bases para establecer el Modelo de Arquitectura Software y el Modelo de Diseño a seguir.

3.1.1.3. Estructura del apartado

Tras esta primera introducción, el resto del apartado se organizará de la forma siguiente:

Descripción General del Sistema	Se describe las características generales del producto, especificando las interfaces que tendrá.
Requisitos	Se describe de manera formal los requisitos software para ayudar a comprender el sistema.
Clasificación de Requisitos Funcionales	Se trata de un listado de todos los requisitos funcionales ordenados por prioridad.
Anexos	Referencia a elementos utilizados en el proyecto.

Tabla 3.1 - Estructura del apartado

3.1.2. Descripción General

3.1.2.1. Perspectiva del Producto

Una de los elementos en términos de informática con el que el usuario trabaja día a día son los formularios. Una forma de agilizar este proceso es automatizarlo, dando al usuario capacidad de crear sus propios formularios, editarlos o borrarlos. Esta herramienta ofrecerá una forma rápida y sencilla de gestionar los formularios de forma que los clientes sean capaces de desempeñar mejor su trabajo y por lo tanto de aumentar su productividad.

3.1.2.1.1. Interfaces del Sistema

La persistencia de los datos se implementará en archivos con formato .cs (C# Files) además de otros propios de CSS, JavaScript o XML.

3.1.2.1.2. Interfaces de Usuario

En el sistema existe una única interfaz web de usuario desarrollada con el GUI de .NET, la cual permite ejecutar todas las funcionalidades. Dicha interfaz estará basada en componentes propios de un interfaz web (ventanas, botones, enlaces, etc.).

3.1.2.1.3. Interfaces Hardware

Se deberá poder ejecutar en un ordenador personal.

3.1.2.1.4. Interfaces Software

El *CLR* es el encargado de compilar una forma de código intermedio llamado Common Intermediate Language (CIL, anteriormente conocido como MSIL, por Microsoft Intermediate Language), al código de maquina nativo, mediante un compilador en tiempo de ejecución. No debe confundirse el CLR con una máquina virtual, ya que una vez que el código esta compilado, corre nativamente sin intervención de una capa de abstracción sobre el hardware subyacente.

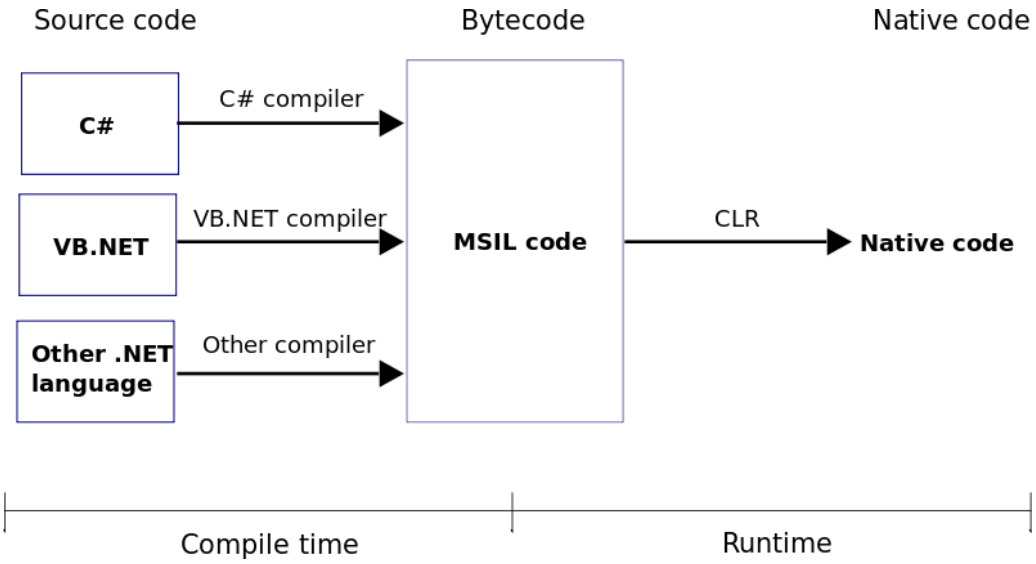

Ilustración 3.1 - Componente CLR

3.1.2.1.5. Interfaces de Comunicación

Nuestro framework de trabajo propone *COM* (Component Object Model) que es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología, en nuestro caso C#.

3.1.2.1.6. Restricciones de Memoria

Los equipos donde se ejecutará la aplicación tienen una antigüedad considerable (5 años), por lo que se tendrá en cuenta respecto a los recursos. En cuanto al procesamiento supondremos un Intel Centrino a 1.8GHz o equivalente y memoria RAM de 2GB DDR2.

3.1.2.1.7. Operaciones

La funcionalidad de la herramienta debe ser fácilmente comprensible. En caso contrario, se habilitarán sistemas de ayuda o un manual de usuario que permitan al usuario comprender las operaciones que se ofrecen, de forma que obtenga todos los conocimientos necesarios para poder usarla sin dificultades y mediante un aprendizaje más rápido.

3.1.2.2. Funciones del Producto

La principal función de este sistema es la de crear formularios que contengan las características o componentes propios del mismo y permita ejecutarlos para poder visualizar su comportamiento.

El sistema mostrará una ventana al usuario para que el usuario personalice sus propios formularios y sus componentes de manera visual.

Aparte de lo dicho anteriormente, el sistema, siguiendo el patrón CRUD (Create, Read, Update, Delete) también permitirá leerlos, actualizarlos y borrarlos.

3.1.2.3. Características del Usuario

Los usuarios pueden ser cualquier tipo de empresa o persona interesada en este sistema. Hoy en día los formularios es un instrumento principal de trabajo por lo tanto cualquier persona podría utilizar este sistema. Como es evidente, los conocimientos en informática deberían ser mínimos.

3.1.2.4. Restricciones

El sistema deberá adaptarse a los cambios que surjan en las especificaciones por parte de las partes interesadas.

3.1.3. Requisitos Específicos

En este apartado se describen los requisitos del sistema formalmente. Los requisitos lo dividimos en funcionales y no funcionales. Los primeros definen el comportamiento del sistema mientras que los segundos imponen restricciones sobre el diseño o la implementación.

3.1.3.1. Requisitos Funcionales

3.1.3.1.1. <FRQ-0001> Construir Formulario

FRQ-0001	**Construir Formulario**
Descripción	El sistema deberá permitir al usuario crear formularios de manera dinámica de forma que el usuario vea en cada momento lo que está editando. El usuario será capaz de determinar la estructura, así como la adición y posicionamiento de los elementos.
Importancia	Vital.

3.1.3.1.2. <FRQ-0002> Guardar Formulario

FRQ-0002	**Guardar Formulario**
Descripción	El sistema deberá permitir al usuario guardar los formularios construidos de forma que asignará un identificador único a cada formulario.
Importancia	Vital.

3.1.3.1.3. <FRQ-0003> Editar Formulario

FRQ-0003	**Editar Formulario**
Descripción	El sistema deberá permitir al usuario editar los formularios construidos
Importancia	Vital.

3.1.3.1.4. <FRQ-0004> Borrar Formulario

FRQ-0004	**Borrar Formulario**
Descripción	El sistema deberá permitir al usuario borrar los formularios construidos.
Importancia	Importante.

3.1.3.1.5. <FRQ-0005> Guardar Información Formulario

FRQ-0005	**Guardar Información Formulario**
Descripción	El sistema deberá permitir al usuario guardar la información insertada en el formulario por el usuario.
Importancia	Vital.

3.1.3.1.6. <FRQ-0006> Editar Información Formulario

FRQ-0006	**Editar Información Formulario**
Descripción	El sistema deberá permitir al usuario editar la información relacionada con el formulario.
Importancia	Vital.

3.1.3.1.7. <FRQ-0007> Borrar Información Formulario

FRQ-0007	**Borrar Información Formulario**
Descripción	El sistema deberá permitir al usuario borrar la información de todos los campos relacionados con el formulario.
Importancia	Importante.

3.1.3.1.8. <FRQ-0008> Crear Portfolio

FRQ-0008	**Crear Portfolio**
Descripción	El sistema deberá permitir crear al usuario un Portfolio o carpeta que contenga formularios. Dentro de este Portfolio, los formularios tendrán la misma estructura y los mismos campos.
Importancia	Importante.

3.1.3.1.9. <FRQ-0009> Mostrar Formularios

FRQ-0009	**Mostrar Formularios**
Descripción	El sistema mostrará una relación de formularios pertenecientes al usuario seleccionables de forma independiente.
Importancia	Vital.

3.1.3.1.10. <FRQ-0010> Ejecutar Formulario

FRQ-0010	**Ejecutar Formulario**
Descripción	El sistema deberá permitir ejecutar los formularios ya construidos de forma que se visualicen de forma gráfica.
Importancia	Vital.

3.1.3.1.11. <FRQ-0011> Exportar Formulario

FRQ-0011	**Exportar Formulario**
Descripción	El sistema deberá permitir la posibilidad de exportar el formulario a un formato de documento portátil o *PDF*.
Importancia	Deseable.

3.1.3.1.12. <FRQ-0012> Publicar Formulario

FRQ-0012	**Publicar Formulario**
Descripción	El sistema deberá permitir la posibilidad publicar el formulario en la web.
Importancia	Vital.

3.1.3.1.13. <FRQ-0013> Validación de Datos del Formulario

FRQ-0013	**Validación de Datos del Formulario**
Descripción	El sistema deberá validar los datos introducidos en los campos del formulario por parte del usuario de manera que se indique de forma adecuada qué campos son incorrectos.
Importancia	Importante.

3.1.3.1.14. <FRQ-0014> Registro de Usuario

FRQ-0014	**Registro de Usuario**
Descripción	El sistema deberá permitir al usuario realizar un registro de forma que pueda acceder al sistema en posteriores ocasiones
Importancia	Vital.

3.1.3.1.15. <FRQ-0015> Login

FRQ-0015	**Login**
Descripción	El sistema deberá permitir al usuario loguearse de forma que acceda al sistema
Importancia	Vital.

3.1.3.2. Requisitos no Funcionales

3.1.3.2.1. <NFR-0001> Facilidad de Uso

NRF-0001	**Facilidad de Uso**
Descripción	El sistema deberá ser relativamente fácil de usar. Debe proveer al usuario una interfaz intuitiva en base a ventanas y formularios. El usuario deberá tener conocimientos básicos de informática.
Importancia	Vital.

3.1.3.2.2. <NFR-0002> Facilidad de Mantenimiento

NRF-0002	**Facilidad de Mantenimiento**
Descripción	El sistema deberá adaptarse a nuevos cambios como pueden ser de hardware o de software (navegador).
Importancia	Importante.

3.1.3.2.3. <NFR-0003> Fiabilidad

NRF-0003	**Fiabilidad**
Descripción	El sistema deberá tener los menos fallos posibles en un entorno determinado y durante un tiempo específico y si ocurrieran, se deberá comportar de forma adecuada.
Importancia	Deseable.

3.1.3.2.4. <NFR-0004> Prestaciones

NRF-0004	**Prestaciones**
Descripción	El sistema deberá estar disponible las 24 horas del día salvo causas externas.
Importancia	Deseable.

3.1.3.2.5. Restricciones de Diseño

3.1.3.2.6. <NFR-0005> Servidor Web

NRF-0005	**Servidor Web**
Descripción	El sistema contará con un servidor web IIS (Internet Information Services) es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.
Importancia	Vital.

3.1.3.2.7. <NFR-0006> Sistema Gestor de Bases de Datos

NRF-0006	**Sistema Gestor de Bases de Datos**
Descripción	El sistema contará con un SGBD que será Microsoft SLQ Server
Importancia	Vital.

3.1.3.2.8. <NFR-0007> Framework .NET

NRF-0007	**Framework .NET**
Descripción	El sistema se apoyará en un framework .NET de Microsoft.
Importancia	Vital.

3.1.3.2.9. <NFR-0008> Navegador Web

NRF-0008	**Navegador Web**
Descripción	El sistema se desarrollará para ser ejecutado en un navegador web.
Importancia	Vital.

3.1.3.2.10. Manual de Usuario

3.1.3.2.11. <NFR-0009> Manual de Usuario

NRF-0009	**Manual de Usuario**
Descripción	El sistema proporcionará una ayuda al usuario de forma que pueda entender mejor la funcionalidad del sistema, sus diferentes componentes y sus correspondientes características.
Importancia	Importante.

3.1.4. Clasificación de Requisitos Funcionales

Lista de los requisitos antes mencionados por orden de preferencia: Vital, Importante, Deseable.

FUNCIONALIDAD	TIPO
Construir Formulario	**Vital**
Guardar Formulario	**Vital**
Editar Formulario	**Vital**
Guardar Información Formulario	**Vital**
Editar Información Formulario	**Vital**
Mostrar Formularios	**Vital**
Ejecutar Formulario	**Vital**
Publicar Formulario	**Vital**
Registro de Usuario	**Vital**
Login	**Vital**
Borrar Formulario	**Importante**

Borrar Información Formulario	**Importante**
Crear Portfolio	**Importante**
Validación de Datos del Formulario	**Importante**
Exportar Formulario	**Deseable**

Tabla 3.10 - Clasificación de los Requisitos Funcionales

3.2. ESPECIFICACIÓN DE CASOS DE USO

3.2.1. Modelo de casos de Uso

3.2.1.1. Introducción

El sistema se basa en un generador de formularios dinámicos que permitirá tanto la creación, edición, borrado de los formularios como de la información que contienen. Solo existirá un tipo de usuario y será el responsable de manejar toda la funcionalidad y capacidad del sistema.

3.2.1.2. Descripción General de los Actores

3.2.1.2.1. Usuario General

En nuestro sistema solo tenemos un usuario que será el encargado de utilizar nuestro sistema. Este sistema está orientado a satisfacer las necesidades de este usuario y por lo tanto la consecución de sus objetivos.

3.2.1.3. Jerarquía de Casos de Uso

Se mostrará un diagrama de casos de uso en el cual se representará todos y cada uno de ellos junto con sus relaciones en caso de que las hubiera.

3.2.1.4. Diagramas del Modelo de Casos de Uso

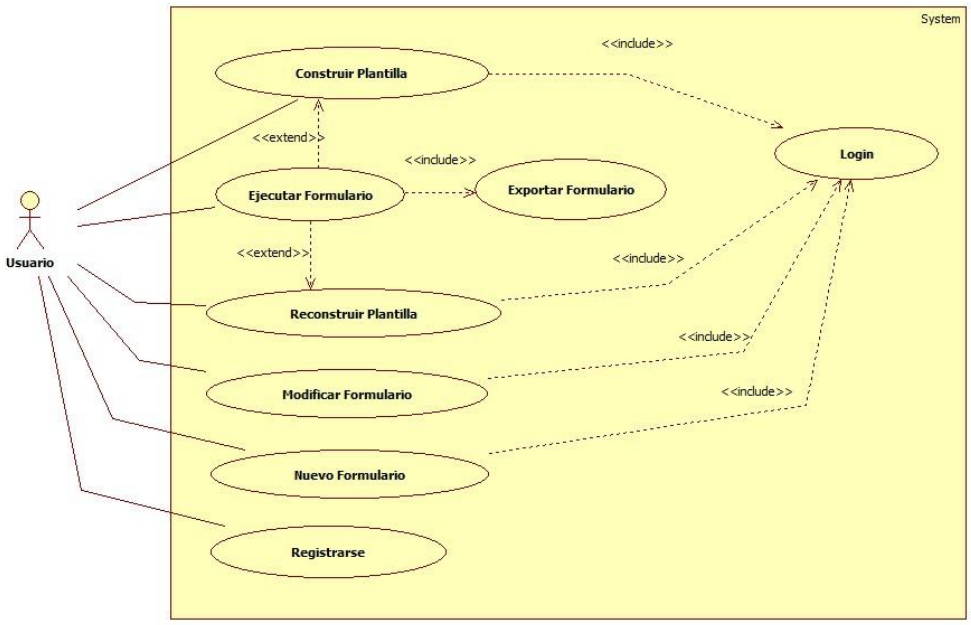

Ilustración 6 - Diagrama de Casos de Uso

3.2.2. Descripción de los Casos de Uso

A continuación explicamos los distintos apartados en que se divide la especificación de cada caso de uso:

- **Identificación del caso de uso:** Cada caso de uso del sistema está identificado con un número. *Ejemplo: UC-001*
- **Dependencias:** Se trata de usuario principal del caso de uso en cuestión.
- **Descripción:** Descripción textual del caso de uso.
- **Precondición:** Condiciones que se deben dar para la previa realización del caso de uso.
- **Secuencia normal:** Secuencia de pasos o acciones que se deben dar para la realización del caso de uso en una ejecución normal (la que cabe esperar con un uso adecuado del sistema).
- **Postcondición:** Condiciones que se cumplen y son siempre ciertas una vez finalizado la secuencia de acciones del caso de uso.
- **Excepciones:** Conjunto de pasos y posibles situaciones anómalas o imprevistas que pueden desembocar en la finalización prematura del caso de uso y cómo afectan a su secuencia normal de acciones.
- **Comentarios:** Comentarios aclaratorios a la descripción o funcionalidad del objetivo en texto libre.

3.2.2.1. <UC-001> Construir Plantilla

UC-001	**Construir Plantilla**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea construir una plantilla de formulario.	
Precondición	El usuario está identificado y registrado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de construir una nueva plantilla de formulario.
	2	El sistema solicita al usuario que introduzca los datos principales de la nueva plantilla de formulario.
	3	El usuario introduce los datos principales de la nueva plantilla y confirma.
	4	El sistema muestra una serie de opciones de edición.
	5	El usuario selecciona las opciones disponibles de forma que determina la estructura de la plantilla y confirma.
	6	El sistema crea una nueva plantilla y el caso de uso finaliza.
Postcondición	La nueva plantilla de formulario construido ha sido registrada en el sistema.	
Excepciones	Paso	Acción
	5	Si el usuario decide ejecutar el formulario se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso continúa.
Comentarios		

3.2.2.2. <UC-002> Reconstruir Plantilla

UC-002	**Reconstruir Plantilla**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea reconstruir/modificar una plantilla formulario.	
Precondición	El usuario está registrado e identificado en el sistema y la plantilla del formulario existe en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de reconstruir/modificar la plantilla del formulario.
	2	El sistema muestra la plantilla con la estructura que el usuario ha creado y una serie de opciones de edición.
	3	El usuario selecciona una serie de opciones de edición de forma que modifique la estructura de la misma y confirma.
	4	El sistema modifica la plantilla y el caso de uso finaliza.
Postcondición	La estructura de la plantilla seleccionada ha sido modificada.	
Excepciones	Paso	Acción
	3	Si el usuario decide ejecutar el formulario se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso continúa.
Comentarios		

3.2.2.3. <UC-003> Ejecutar Formulario

UC-003	**Ejecutar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea ejecutar un formulario.	
Precondición	El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de ejecutar el formulario.
	2	El sistema muestra el formulario con la información que el usuario ha creado.
	3	El usuario visualiza la información referente al formulario y confirma.
	4	El sistema cierra la zona de visualización del formulario y el caso de uso finaliza.
Postcondición	La información del formulario ha sido mostrada al usuario.	
Excepciones	Paso	Acción
	3	Si el usuario decide exportar el formulario se inicia el caso de uso *Exportar Formulario UC-006*, a continuación, este caso de uso continúa.
Comentarios		

3.2.2.4. <UC-004> Nuevo Formulario

UC-004	**Nuevo Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea crear un formulario rellenando la información de los campos.	
Precondición	El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción de creación de un nuevo formulario.
	2	El sistema muestra el formulario y los campos correspondientes.
	3	El usuario rellena la información de los campos del formulario.
	4	El sistema valida los datos introducidos, indicando al usuario si cada campo ha sido rellenado correctamente.
	5	El usuario visualiza la información y confirma.
Postcondición	El nuevo formulario junto con su información ha sido registrado en el sistema.	
Excepciones	Paso	Acción
	5	Si el usuario decide ejecutar el formulario se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso continúa.
Comentarios		

3.2.2.5. <UC-005> Modificar Formulario

UC-005	**Modificar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea modificar la información de los campos del formulario.	
Precondición	El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona el formulario correspondiente.
	2	El sistema muestra el formulario junto con la información disponible de los campos.
	3	El usuario modifica la información de los campos del formulario.
	4	El sistema valida los datos introducidos, indicando al usuario si cada campo ha sido rellenado correctamente.
	5	El usuario visualiza la información y confirma.
Postcondición	El nuevo formulario junto con su información modificada ha sido registrado en el sistema.	
Excepciones	Paso	Acción
	5	Si el usuario decide ejecutar el formulario se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso continúa.
Comentarios		

3.2.2.6. <UC-006> Exportar Formulario

UC-006	**Exportar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea exportar el formulario.	
Precondición	El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción exportar formulario.
	2	El sistema muestra el formulario con el formato final de exportación.
	3	El usuario visualiza el formulario y selecciona la opción correspondiente para guardar el formulario con el nuevo formato.
	4	El sistema guarda el formulario, muestra un mensaje de éxito y el caso de uso finaliza.
Postcondición	El formulario ha sido exportado como un nuevo fichero con formato específico.	
Comentarios		

3.2.2.7. <UC-007> Login

UC-007	**Login**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando cualquier usuario desee identificarse en el sistema.	
Precondición	El usuario está registrado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario introduce sus datos personales y selecciona la opción para identificarse.
	2	El sistema verifica que el usuario está registrado.
	3	El sistema muestra las funciones y datos disponibles para el usuario y el caso de uso finaliza.
Postcondición	El usuario ha quedado identificado en el sistema.	
Excepciones	Paso	Acción
	3	Si el usuario no está registrado, el sistema le notifica la situación de error y el caso de uso queda sin efecto.
Comentarios		

3.2.2.8. <UC-008> Registrarse

UC-008	**Registrarse**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando cualquier usuario desee registrarse en el sistema.	
Precondición	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción adecuada para realizar el registro en el sistema.
	2	El sistema muestra una serie de campos que son necesarios para el registro.
	3	El usuario rellena todos los campos disponibles con su información.
	4	El sistema valida los datos introducidos por el usuario e indica en caso de error los campos que el usuario debe rellenar correctamente. El sistema registra al usuario y el caso de uso finaliza
Postcondición	El usuario ha quedado registrado en el sistema.	
Comentarios		

3.3. GLOSARIO DEL PROYECTO

3.2.3. Introducción

3.2.3.1. Propósito

Este texto contiene la definición de los términos (abreviaturas, acrónimos, etc.) utilizados en el proyecto para poder entender ciertos conceptos que, de otra forma, pudieran ser ambiguos o simplemente desconocidos. Irá siendo ampliado conforme se avance en el proyecto y aparezcan nuevos términos.

3.2.3.2. Alcance

Se describirán todos los términos que abarque el proyecto, tanto aquellos que hagan referencia a la funcionalidad del sistema como los que definen el lenguaje de modelado y la metodología seguida. Los actores del sistema serán descritos con detalle en el apartado de *Especificación de Requisitos Software*, por lo que aquí no aparecerán.

3.2.3.3. Estructura del apartado

El apartado siguiente se divide en varios grupos de términos:

- Referidos a la metodología utilizada (UPEDU).
- Referidos al entorno de la aplicación.
- Referidos al entorno de desarrollo.

3.2.4. Definiciones

3.2.4.1. Sobre la Metodología

3.2.4.1.1. <Actividad>

Unidad de trabajo que puede ejecutar un individuo (o varios) en un rol específico. Tiene un propósito claro y se expresa en términos de actualizar artefactos.

3.2.4.1.2. <Artefacto>

Pieza de información producida, modificada y utilizada en un proceso. Producto tangible del proyecto.

3.2.4.1.3. <Componente>

Parte modular de un sistema que encapsula su contenido y que es reemplazable en su entorno. Un componente define su comportamiento en términos de sus interfaces proporcionadas y requeridas.

3.2.4.1.4. <CRQ>

Requisito de Restricción (***C****onstraint* ***Req****uirement*), son restricciones de diseño dadas por el cliente o identificadas en las fases de planificación o análisis del proyecto.

3.2.4.1.5. <Diagrama de Gantt>

Es una popular herramienta gráfica cuyo objetivo es el de mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

3.2.4.1.6. <Entregable>

Artefacto terminado. A entregar en la fecha especificada en la planificación.

3.2.4.1.7. <Fase>

Desde la perspectiva de desarrollo de software, cada fase es el intervalo de tiempo entre dos hitos importantes. Al final de cada una se hacen varias comprobaciones para determinar si se han alcanzado los objetivos planteados. De ser así, se puede pasar a la siguiente fase. Un proceso de desarrollo de software se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición.

3.2.4.1.8. <Fiabilidad>

Referido al comportamiento de un sistema o dispositivo, se define como la "probabilidad de que el dispositivo desarrolle una determinada función, bajo ciertas condiciones y durante un período de tiempo determinado".

3.2.4.1.9. <FRQ>

Requisito Funcional (***F**unctional **Req**uirement*), son requisitos especificados por el cliente que identifican opciones y funcionalidades que debe soportar el sistema.

3.2.4.1.10. <Funcionalidad>

Conjunto de características que hacen que algo sea práctico y utilitario.

3.2.4.1.11. <Gestión de proyectos>

Es la aplicación de conocimiento, habilidades, herramientas y técnicas a las actividades de los proyectos, con el fin de conseguir o superar las necesidades y expectativas de este.

3.2.4.1.12. <Gestión de riesgos>

Práctica de valorar y controlar los riesgos que afectan a un producto, proceso o proyecto software. En general, la idea es describir inicialmente los objetivos y después se describen los riesgos en términos de incertidumbre, pérdidas y tiempo.

3.2.4.1.13. <GQM>

El paradigma GQM, Objetivo-Pregunta-Métrica (***G**oal-**Q**uestion-**M**easure*), de Basili y Rombach, es un marco de trabajo para un amplio espectro de actividades de medidas de software. La idea es que cualquier actividad de medida de software debe ir precedida por la identificación de un objetivo de ingeniería, lo que lleva a preguntas, que conducen a una o varias métricas específicas.

3.2.4.1.14. <Hito>

Corresponde con un punto de control de objetivos intermedios, que se establece en un punto temporal anterior a la finalización del proyecto.

3.2.4.1.15. <Interfaz>

Representa la declaración de un conjunto de funciones y obligaciones que alguien más lleva a cabo. En el diagrama de componentes, representa aquello que un componente es capaz de hacer por alguien más (interfaz proporcionada) o aquellos servicios que posiblemente requiera utilizar un componente y que son proporcionados por otro (interfaz requerida).

3.2.4.1.16. <IRQ>

Requisito de Datos (***I**nformation **Req**uirement*), da información sobre qué datos de la aplicación hay que hacer persistentes y en qué momento.

3.2.4.1.17. <Iteración>

Es cada una de las partes en que se divide una fase. Descomponer una fase en varias iteraciones e ir desarrollando incrementalmente cada una de las disciplinas del sofware (requisitos, análisis, diseño, implementación y pruebas) nos ayuda a comprender mejor el producto a desarrollar y, por tanto, entraña menos riesgos que intentar abarcarlo todo al mismo tiempo.

3.2.4.1.18. <Medida>

Es el proceso de asignación efectiva y empírica de números o símbolos a atributos de entidades del mundo real para caracterizarlas de acuerdo con reglas definidas claramente.

3.2.4.1.19. <Métricas>

Caracterizan numéricamente atributos sencillos, como longitud, número de decisiones, número de operadores (en el caso de programas), o número de errores encontrados, coste y tiempo (para procesos).

3.2.4.1.20. <NFR>

Requisito No Funcional (***N**on **F**unctional **R**equirement*), son requisitos que la aplicación debe cumplir pero no forman parte de su funcionalidad, sino de su rendimiento. Generalmente están asociados a medidas de calidad, robustez y fiabilidad del sistema.

3.2.4.1.21. <Patrón>

Un patrón es una colaboración parametrizada, junto con las pautas sobre cuándo utilizarlo. Un parámetro se puede sustituir por diversos valores, para producir distintas colaboraciones.

3.2.4.1.22. *<Portabilidad>*

Capacidad de un sistema software de funcionar sobre diferentes entornos hardware y software.

3.2.4.1.23. *<Producto o servicio>*

Objetivo final de todo proyecto, será único y deberá satisfacer a un mercado.

3.2.4.1.24. *<Proyecto>*

Esfuerzo temporal emprendido con el fin de obtener un producto, que por su complejidad, duración y desarrollo en grupo necesita de una planificación.

3.2.4.1.25. *<Recurso>*

Elemento necesario para llevar a cabo una tarea.

3.2.4.1.26. *<Riesgo>*

Es una medida de la probabilidad y pérdida de que se produzca un resultado inadecuado que afecte al producto, proceso o proyecto software.

3.2.4.1.27. *<RUP>*

Proceso Unificado de Rational (***R**ational **U**nified **P**rocess*). Es la metodología de IBM Rational para el desarrollo y construcción de software, está dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental, hace uso de UML como soporte a la metodología.

3.2.4.1.28. *<SRS>*

Especificación de Requisitos Software, (***S**oftware **R**equirements **S**pecification*), apartado que especifica los requisitos que debe cumplir el sistema a desarrollar.

3.2.4.1.29. *<UML>*

Lenguaje Unificado de Modelado (***U**nified **M**odeling **L**anguage*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

3.2.4.1.30. *<UPEDU>*

Acrónimo de ***U**nified **P**rocess for **Edu**cation* (Proceso Unificado en Educación) es un proceso de desarrollo de software enfocado a la educación, desarrollado por Pierre-N. Robillard (Escuela Politécnica de Montréal), Philippe Kruchten (Rational Software) y Patrick d'Astous (Escuela Politécnica de Montréal). Es una variante de RUP.

3.2.4.1.31. *<Usabilidad>*

Proviene del término inglés *usability*. En castellano significa capacidad de uso, es decir, la característica que distingue a los objetos diseñados para su utilización de los que no. Sin embargo la acepción inglesa es más amplia y se refiere a la facilidad o nivel de uso, es decir, al grado en el que el diseño de un objeto facilita o dificulta su manejo. Nosotros nos basamos en la segunda acepción.

3.2.4.2. Sobre la Aplicación

3.2.4.2.1. *<Formulario>*

Un formulario es un documento con espacios (campos) en donde se pueden escribir o seleccionar opciones. Cada campo tiene un objetivo, por ejemplo, el campo "Nombre" se espera que sea llenado con un nombre, el campo "año de nacimiento", se espera que sea llenado con un número válido para un año, etc.

Presentan una visión ordenada de múltiple información sobre algo, y son útiles para llenar bases de datos. Pueden ser llenados de forma online con validación de datos, y son muy útiles para las encuestas, registración de usuarios, ingreso a sistemas, suscripciones, etc.

Los formularios por internet son llamados formularios web, y generalmente son hechos a través de etiquetas *HTML*, aunque también existen otros medios como Flash, Java, etc.

3.2.4.2.2. *<Submit>*

Significa enviar los datos rellenados del formulario.

3.2.4.2.3. <Reset>

Borrar los datos actualmente introducidos en el formulario

3.2.4.2.4. <Validar>

Establecer un formato correcto para cada uno de los campos, impidiendo al usuario realizar un submit del formulario actual.
Por ejemplo: Un DNI tiene un formato específico y debería de ser validado antes de enviar la información.

3.2.4.2.5. <Archivo Adjunto>

Es un archivo (document word, document de texto, fotografía, etc.) que se carga normalmente en un campo del formulario y que se envía junto con el formulario.

3.2.4.2.6. <Exportar Documento>

Significa "forzar" a una aplicación a crear un documento que la misma aplicación no podrá editar luego. A través de la exportación se crea un documento externo cuya extensión podría ser por ejemplo *PDF*.

3.2.4.3. Sobre el entorno de Desarrollo

3.2.4.3.1. <.NET Framework>

Es un framework de Microsoft cuya funcionalidad y acometido principal es el rápido y económico desarrollo de aplicaciones sin descuidar la robustez y seguridad de dichas aplicaciones.

3.2.4.3.2. <ASP.NET>

Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web *XML*.

3.2.4.3.3. <MVC>

Es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno *MVC* se ve frecuentemente en aplicaciones web, donde la vista es la página *HTML* y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

3.2.4.3.4. <Lenguaje de Programación C#>

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma *.NET*.

3.2.4.3.5. <HTML5>

Es la quinta revisión importante del lenguaje básico de la *World Wide Web*, *HTML*. Podemos definir la especificación *HTML5* como nuevos elementos de markup o sintaxis, utilizados por los diseñadores para crear páginas web junto con las etiquetas utilizadas a día de hoy. Estas nuevas etiquetas suponen para desarrolladores y diseñadores, unas herramientas más avanzadas y se traducen en mejores experiencias para el usuario final. La familia *HTML5* incluye las nuevas etiquetas y tecnologías como CSS3, Geolocalización, Almacenamiento Web (Web Storage),etc.

3.2.4.3.6. <XForms>

Es un formato *XML* diseñado por el *W3C* para poder definir interfaces de usuario, principalmente formularios web. XForms ha sido diseñado para ser la nueva generación de formularios *HTML/XHTML*, pero es lo suficientemente genérico como para que pueda ser usado, de una manera independiente, para describir cualquier interfaz de usuario e incluso para realizar tareas simples y comunes de manipulación de datos.

3.2.4.3.7. <W3C>

El World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce recomendaciones para la World Wide Web.

3.2.4.3.8. <XML>

XML son las siglas de *Extensible Markup Language*, una especificación/lenguaje de programación desarrollada por el W3C. XML es una versión de SGML, diseñado especialmente para los documentos de la

web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones.

3.2.4.3.9. <SGML>

Son las siglas de Standard *Generalized Markup Language*. Consiste en un sistema para la organización y etiquetado de documentos. El lenguaje *SGML* sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

3.2.4.3.10. <CSS>

CSS son las siglas de Cascading Style Sheets - Hojas de Estilo en Cascada - que es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe cómo se va a mostrar un documento en pantalla, por impresora, por voz (cuando la información es pronunciada a través de un dispositivo de lectura) o en dispositivos táctiles basados en Braille.

3.2.4.3.11. <PDF>

Portable Document Format (formato de documento portable) es el formato de archivos desarrollado por Adobe Systems y creado con los programas Adobe Acrobat Reader, Acrobat Capture, Adobe Distiller, Adobe Exchange, y el plugin Amber de Adobe Acrobat. Esta tecnología ha tenido éxito estandardizando el formato de los documentos que se utilizan y transfieren en Internet. El PDF es como un formato de archivos universal.

3.2.4.3.12. <jQuery>

Es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

3.2.4.3.13. <AJAX>

Acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y *XML*), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

3.2.4.3.14. <URL>

Un localizador de recursos uniforme, más comúnmente denominado *URL* (*Uniform Resource Locator*), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc. Los localizadores uniformes de recursos fueron una innovación en la historia de la Internet.

3.2.4.3.15. <CIL>

Common Intermediate Language. Es el lenguaje de programación legible por humanos de más bajo nivel en el Common Language Infrastructure y en el .NET Framework. Los lenguajes del .NET Framework compilan a CIL. CIL es un lenguaje ensamblador orientado a objetos, y está basado en pilas. Es ejecutado por una máquina virtual.

3.2.4.3.16. <CLR>

Common Language Runtime. Es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. El CLR es el encargado de compilar una forma de código intermedio llamada Common Intermediate Language (CIL, anteriormente conocido como MSIL, por Microsoft Intermediate Language), al código de maquina nativo, mediante un compilador en tiempo de ejecución.

3.2.4.3.17. <MSIL>

Microsoft Inmediate Language. Este MSIL es un lenguage intermedio común a todos los sistemas operativos que soporten el framework .NET.

3.2.4.3.18. <WYSIYG>

What You See Is What You Get. Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final,

frecuentemente el resultado impreso. Se utiliza en contraposición a otros procesadores de texto, hoy en día poco frecuentes, en los que se escribía sobre una vista que no mostraba el formato del texto, hasta la impresión del documento. En el caso de editores de HTML este concepto se aplica a los que permiten escribir la página sobre una vista preliminar similar a la de un procesador de textos, ocupándose en este caso el programa de generar el código fuente en HTML.

3.2.4.3.19. <Servidor Web>

Es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se utiliza el protocolo HTTP para estas comunicaciones. El término también se emplea para referirse al ordenador que ejecuta el programa.

3.2.4.3.20. <SGBD>

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto. Los SGBD también proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y recuperar la información si el sistema se corrompe. Permite presentar la información de la base de datos en variados formatos. La mayoría de los SGBD incluyen un generador de informes. También puede incluir un módulo gráfico que permita presentar la información con gráficos y tartas.

3.2.4.3.21. <IIS>

Internet Information Services es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.

3.2.4.3.22. <HTTP>

Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

4. ANÁLISIS Y DISEÑO

4.1. MODELO DE ANÁLISIS

4.1.1. Introducción

4.1.1.1. Propósito

El propósito de este apartado es interpretar los requisitos software del cliente en un conjunto de especificación de modelado UML.

4.1.1.2. Alcance

Se trata de explicar el funcionamiento tanto de forma estática (diagramas de clases), como dinámica (diagramas de secuencia). Este desarrollo servirá como base para construir el modelo de diseño.

4.1.1.3. Estructura del apartado

Modelo de Análisis (Estático)	Describe la parte estática de la aplicación a través de un diagrama de clases. Más adelante se detalla cada clase y sus características.
Diagramas de Secuencia (Dinámico)	Describe la parte dinámica de la aplicación, organizada por casos de uso, a través de sus diagramas de secuencia correspondientes.

Tabla 11 - Estructura del apartado

4.1.2. Estructura del Modelo de Análisis

En este apartado podemos encontrar una descripción estática del modelo de análisis de la aplicación.

4.1.2.1. Diagrama de Clases de Análisis

Se expone las principales clases de análisis y las relaciones que existen entre ellas.

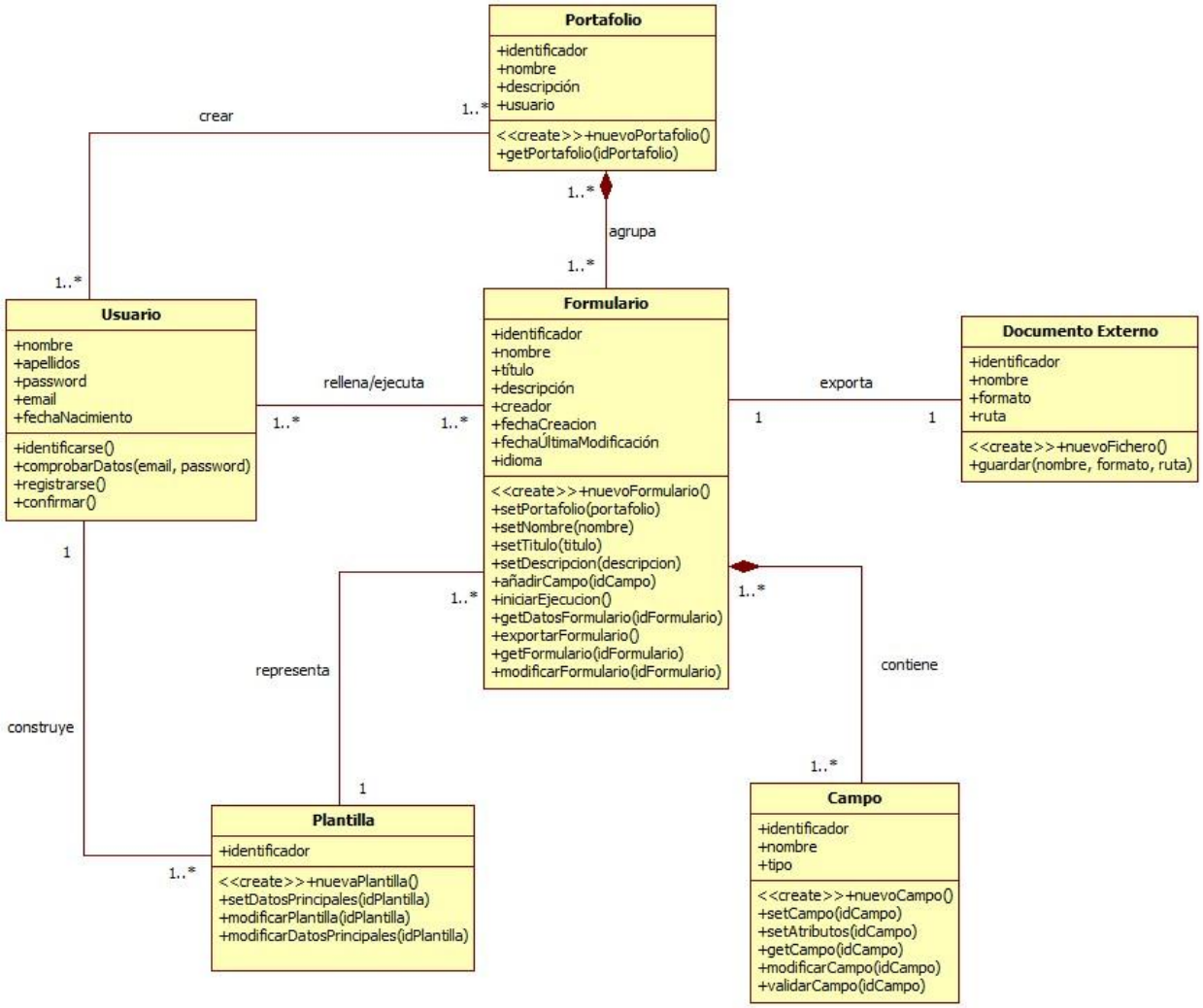

Ilustración 4.1 - Diagrama de Clases de Análisis

4.1.2.2. Descripción de las Clases

A continuación describiremos brevemente cada una de las clases anteriores, incluyendo qué atributos y operaciones poseen y para qué sirven.

4.1.2.2.1. Clase Formulario

Descripción

Es la clase principal del modelo de análisis. Esta clase representa el objeto formulario de nuestro sistema.

Relaciones

Tiene relación con las clases *Usuario*, *Portfolio*, *Documento Externo* y *Campos.* El *Formulario* que contiene una serie de *Campos* y está contenido en un *Portfolio*, lo utiliza un *Usuario*. A su vez este *Formulario* puede ser exportado como *Documento Externo*.

Atributos

- **identificador**: Integer

  Identifica de manera univoca al formulario

- **nombre**: String

  Nombre del formulario

- **título**: String

  Título del formulario

- **descripción**: String

  Descripción breve del formulario. Indica para que propósito ha sido generado y su funcionalidad.

- **creador**: String

  Usuario creador del formulario.

- **fechaCreación**: Date

  Fecha en la que se creó el formulario

- **fechaÚltimaModificación**: Date

  Fecha de última modificación del formulario

- **idioma**: String

  Idioma en el que está escrito el formulario

Operaciones

+ nuevoFormulario()

Nos permite crear un nuevo formulario

+ setPortfolio(**Portfolio:integer**)

Asigna un Portfolio al formulario

+ setNombre(**nombre:string**)

Asigna un nombre al formulario

+ setTitulo(**titulo:string**)

Asigna un título al formulario

+ setDescripción(**descripción:string**)

Asigna una descripción al formulario

+ añadirCampo(**idCampo:integer**)

Añadimos un campo al formulario

+ iniciarEjecución()

Ejecutamos el formulario de forma que lo podamos previsualizar

+ getDatosFormulario(**idFormulario:integer**)

Obtenemos los datos principales del formulario correspondiente al parámetro especificado

+ exportarFormulario()

Exportamos el formulario a un documento externo

+ getFormulario(**idFormulario:integer**)

Obtenemos el formulario correspondiente al parámetro especificado

+ modificarFormulario(**idFormulario:integer**)

Modificamos el formulario correspondiente al parámetro especificado

4.1.2.2.2. Clase Portfolio

Descripción

Es la clase contenedora de formularios. Esta clase agrupa una serie de formularios de igual funcionalidad.

Relaciones

Tiene relación con las clases *Usuario* y *Formulario.* El *Usuario* es el encargado de crear el *Portfolio* y como hemos expuesto anteriormente contiene una serie de *Formularios*.

Atributos

- **identificador**: Integer

  Identifica de manera univoca al Portfolio

- **nombre**: String

  Nombre del Portfolio

- **descripción**: String

  Descripción breve del Portfolio sobre la funcionalidad que comparten los formularios que contiene.

- **usuario**: String

  Usuario creador del Portfolio.

Operaciones

+ nuevoPortfolio()

Nos permite crear un nuevo Portfolio

+ getPortfolio(**idPortfolio:integer**)

Obtenemos el Portfolio actual del usuario correspondiente al parámetro especificado

4.1.2.2.3. Clase Usuario

Descripción

Es la clase que representa al objeto usuario.

Relaciones

Tiene relación con las clases *Portfolio* y *Formulario*. El *Usuario* puede crear tanto el *Portfolio* como el *Formulario.*

Atributos

- **nombre**: String

  Nombre del usuario

- **apellidos**: String

  Apellidos del usuario

- **password**: String

  Contraseña del usuario que utiliza para entrar en el sistema

- **email**: String

  Correo electrónico del usuario

- **fechaNacimiento**: Date

  Correo electrónico del usuario

Operaciones

+ identificarse()

Nos identificamos en el sistema.

+ comprobarDatos(email:String,password;String)

Buscamos y comprobamos si los datos introducidos son correctos

+ registrarse()

Nos permite registrarnos en el sistema

+ confirmar()

Confirmamos para registrarnos

4.1.2.2.4. Clase Campo

Descripción

Es la clase que representa el objeto campo de un formulario. El formulario está compuesto básicamente por campos correctamente posicionados.

Relaciones

Tiene relación con la clase *Formulario.* Como hemos comentado anteriormente, el *Formulario* está compuesto por varios *Campos*.

Atributos

- **identificador**: Integer

  Identifica de manera univoca al campo

- **nombre**: String

  Nombre del campo

- **tipo**: String

  Indica el tipo de campo que es. Existen multitud de campos disponibles que podemos insertar en un formulario como pueden ser por ejemplo campos de tipo texto, de tipo fecha o campos de tipo numérico.

Operaciones

+ nuevoCampo()

Creamos un nuevo campo

+ setCampo(**:string**)

Asignamos el campo correspondiente al parámetro especificado

+ setAtributos(**idCampo:integer**)

Establecemos los atributos del campo correspondiente al parámetro especificado

+ getCampo(**idCampo:integer**)

Obtenemos el campo correspondiente al parámetro especificado

+ modificarCampo(**idCampo:integer**)

Modificamos el campo correspondiente al parámetro especificado

+ validarCampo(**idCampo:integer**)

Validamos el formato del campo correspondiente al parámetro especificado

4.1.2.2.5. Clase Documento Externo

Descripción

Es la clase que representa el formulario exportado como *Documento Externo*, generalmente con otro formato.

Relaciones

Tiene relación con la clase *Formulario.* El *Formulario* es exportado como documento externo y la información está contenida en dicho documento.

Atributos

- **identificador**: Integer

  Identifica de manera univoca al documento externo

- **nombre**: String

Nombre del documento exportado

- **formato**: String

  Indica el formato con el que será exportado. Generalmente solo existirá un formato de exportación y será PDF.

- **ruta**: String

  Representa la ruta dónde este documento se encuentra almacenado.

Operaciones

+ nuevoFichero()

Creamos un nuevo fichero/documento externo

+ guardar(**nombre:string,formato:string,ruta:string**)

Guardamos el fichero/documento externo con nombre, formato y ruta indicados.

4.1.2.2.6. Clase Plantilla

Descripción

Es la clase que representa la plantilla del formulario. Cada formulario tiene una plantilla que representa la estructura y contiene campos.

Relaciones

Tiene relación con la clase *Formulario.* El *Formulario* representa una *Plantilla*, la cual contiene campos que se podrán rellenar.

Atributos

- **identificador**: Integer

  Identifica de manera univoca a la plantilla

Operaciones

+ nuevaPlantilla()

Creamos una nueva plantilla

+ setDatosPrincipales(**idPlantilla:integer**)

Definimos los datos que son el título, nombre y descripción de la plantilla pasado como parámetro.

+ modificarPlantilla(**idPlantilla:integer**)

Modificamos la plantilla correspondiente al parámetro especificado

+ modificarDatosPrincipales(**idPlantilla:integer**)

Modificamos los datos principales de la plantilla correspondiente al parámetro especificado.

4.1.3. Diagramas de Secuencia

4.1.3.1. <UC-001> Construir Plantilla

4.1.3.1.1. Diagrama de Secuencia

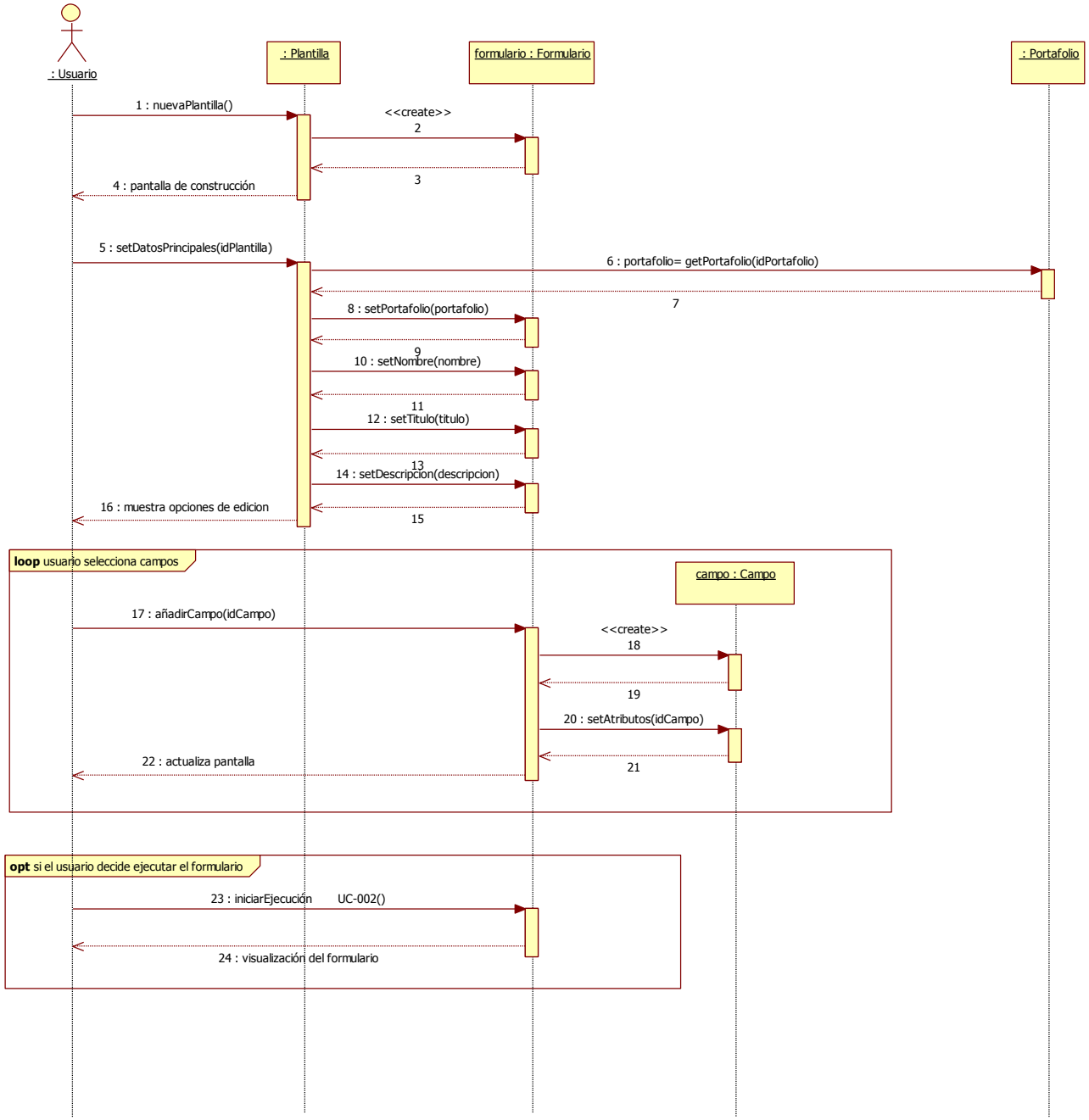

Ilustración 4.2 - <UC-001> Construir Plantilla

4.1.3.2. <UC-002> Reconstruir Plantilla

4.1.3.2.1. Diagrama de Secuencia

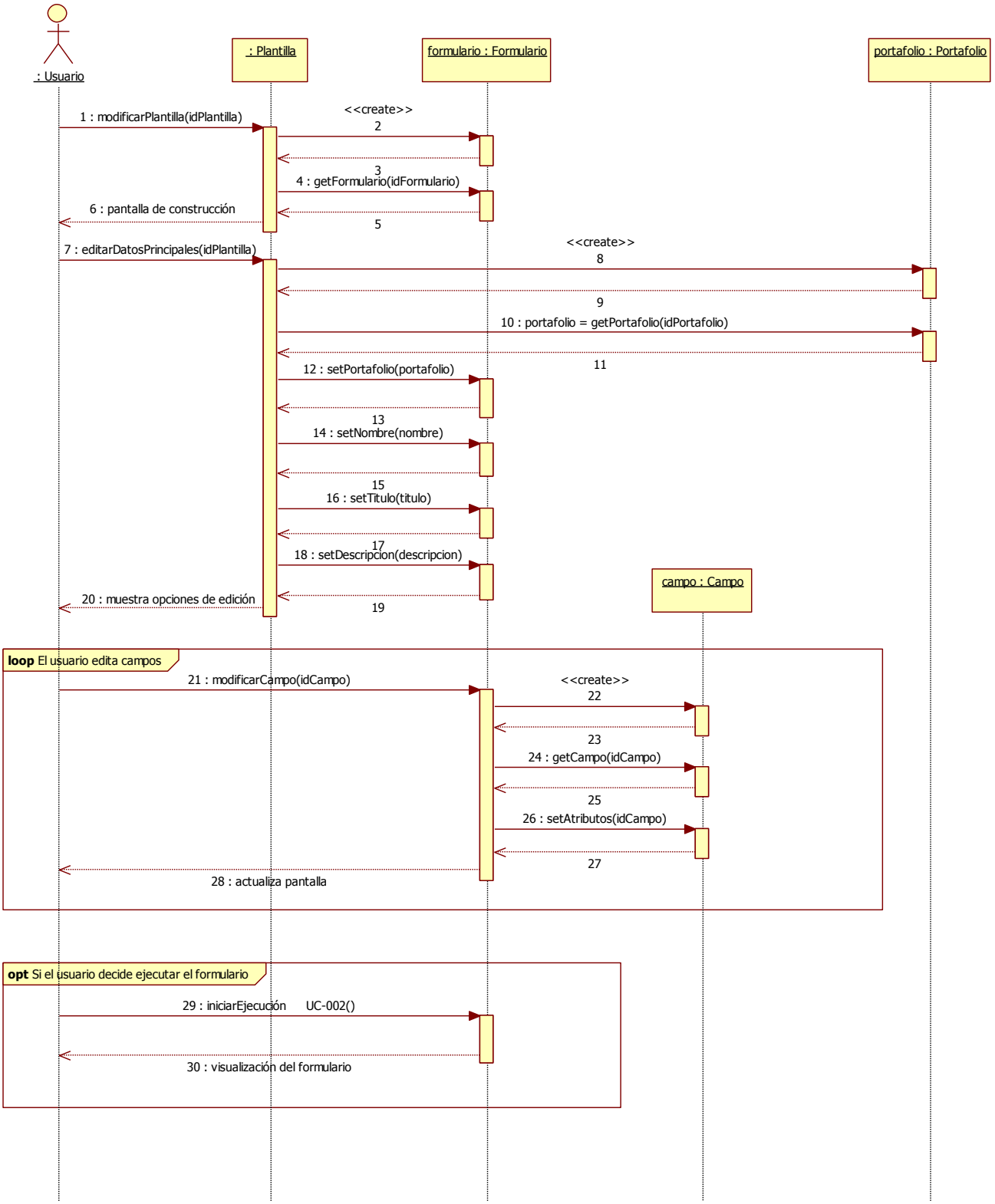

Ilustración 4.3 - <UC-002> Recontruir Plantilla

4.1.3.3. <UC-003> Ejecutar Formulario

4.1.3.3.1. Diagrama de Secuencia

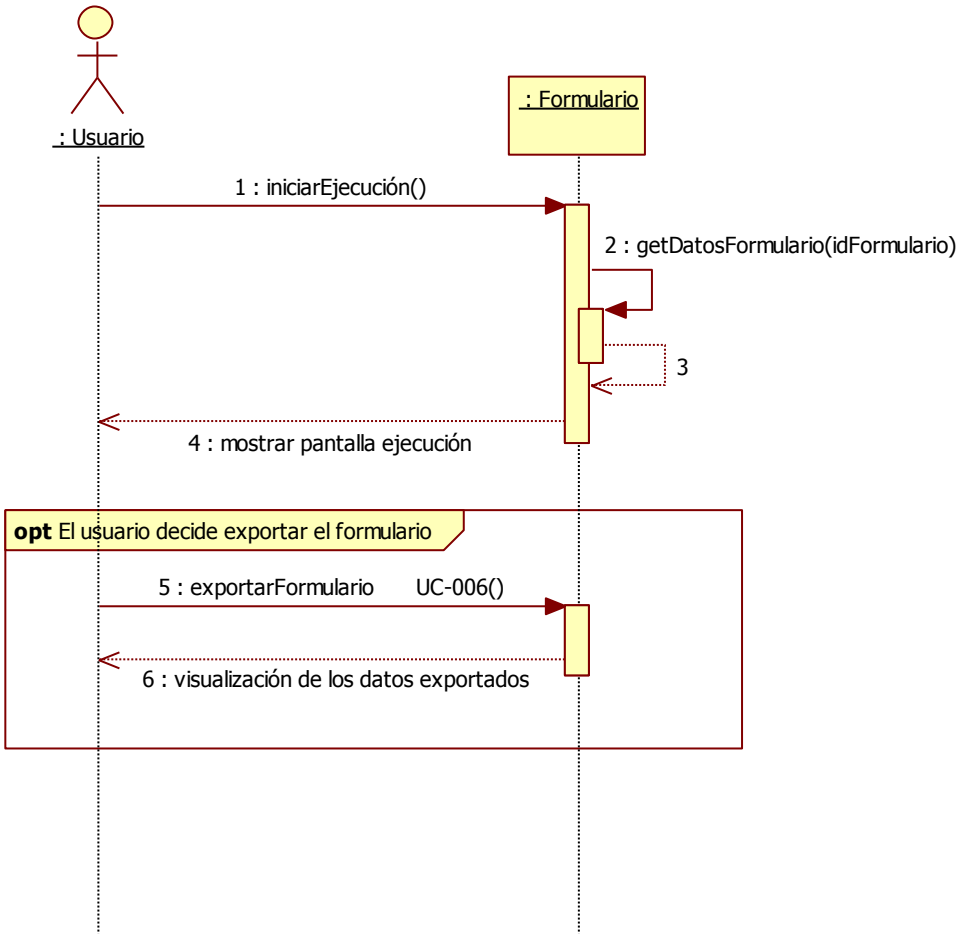

Ilustración 4.4 - <UC-003> Ejecutar Formulario

4.1.3.4. <UC-004> Nuevo Formulario

4.1.3.4.1. Diagrama de Secuencia

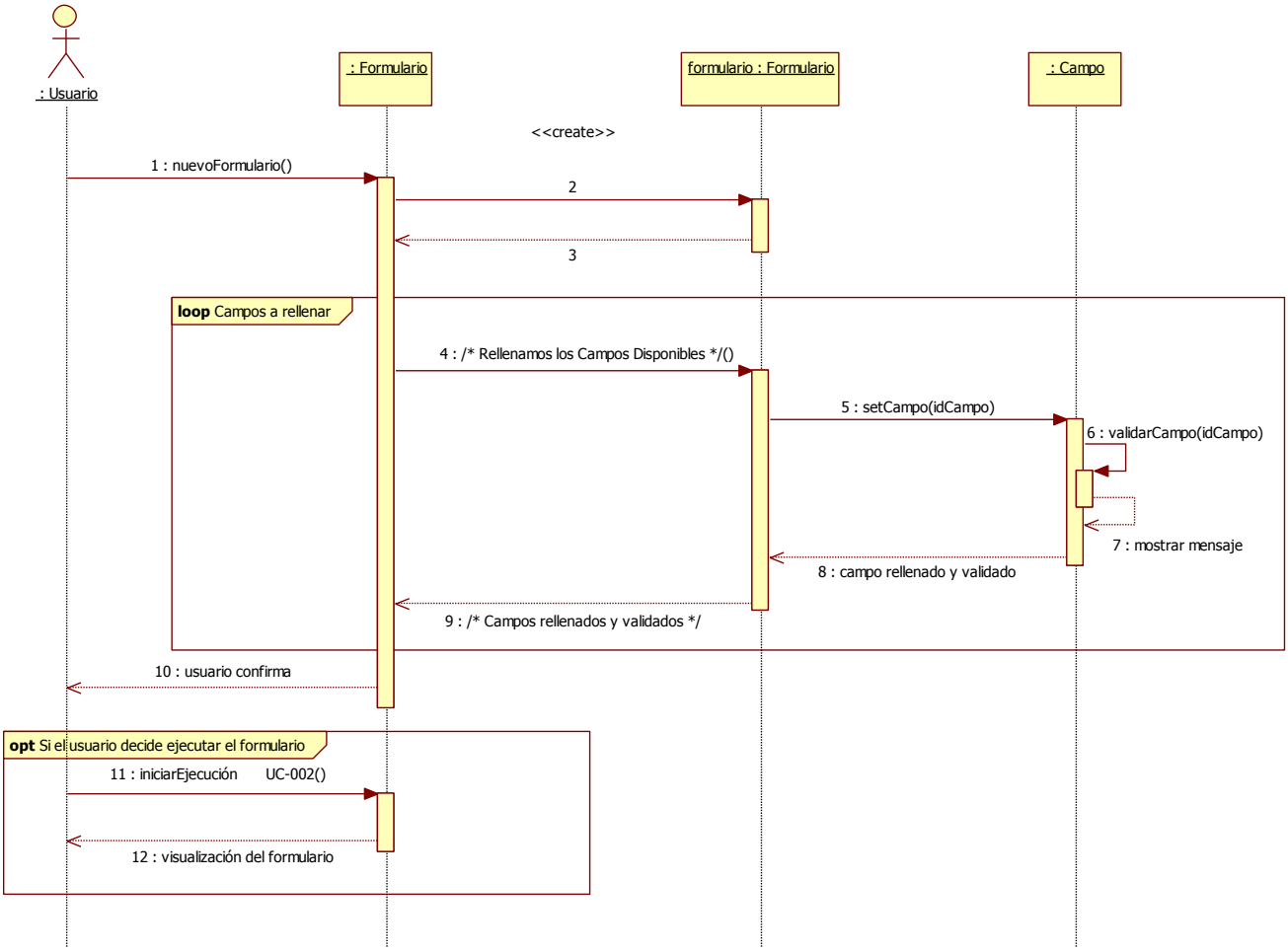

Ilustración 4.5 - <UC-004> Nuevo Formulario

4.1.3.5. <UC-005> Modificar Formulario

4.1.3.5.1. Diagrama de Secuencia

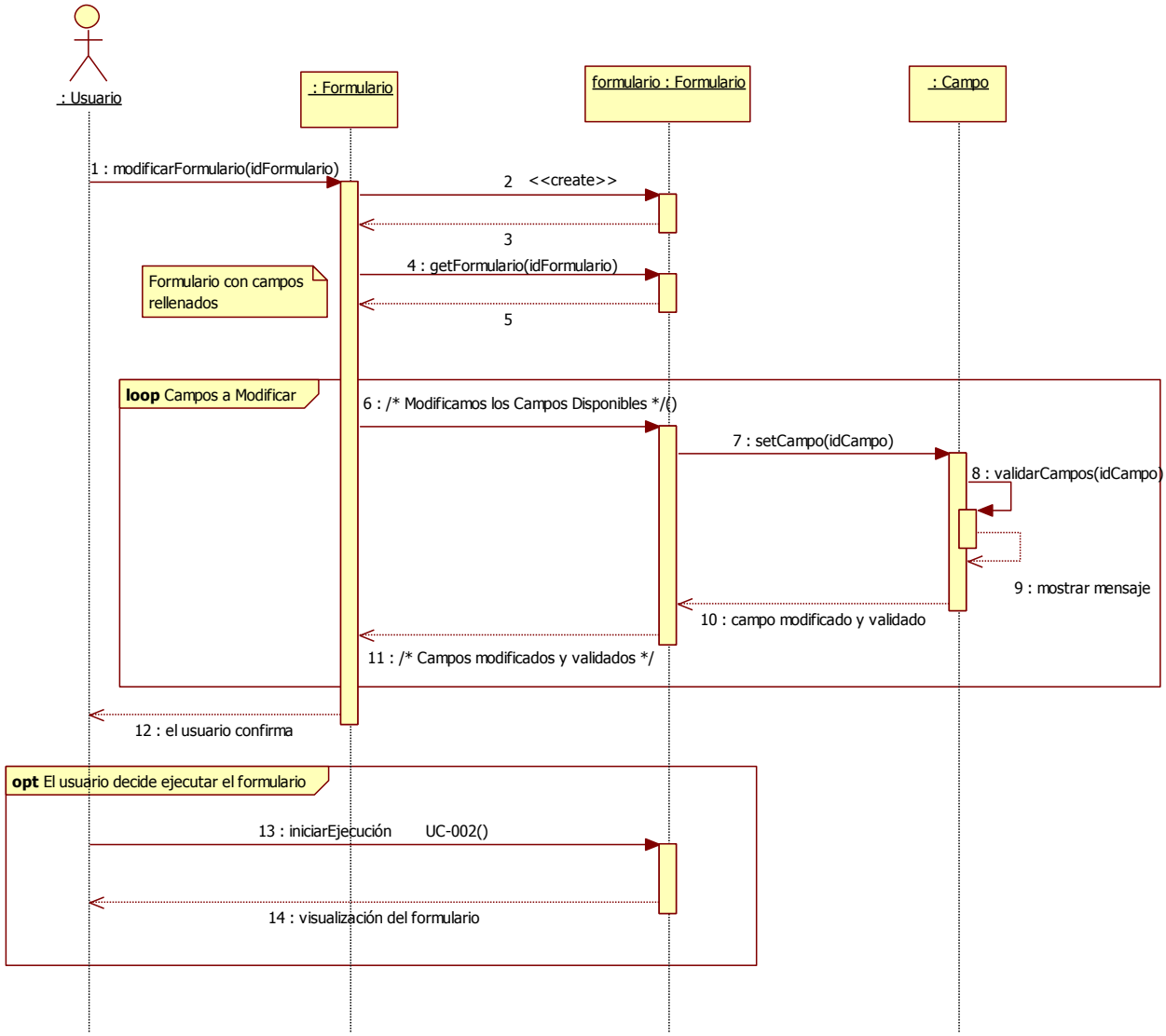

Ilustración 4.6 - <UC-005> Modificar Formulario

4.1.3.6. <UC-006> Exportar Formulario

4.1.3.6.1. Diagrama de Secuencia

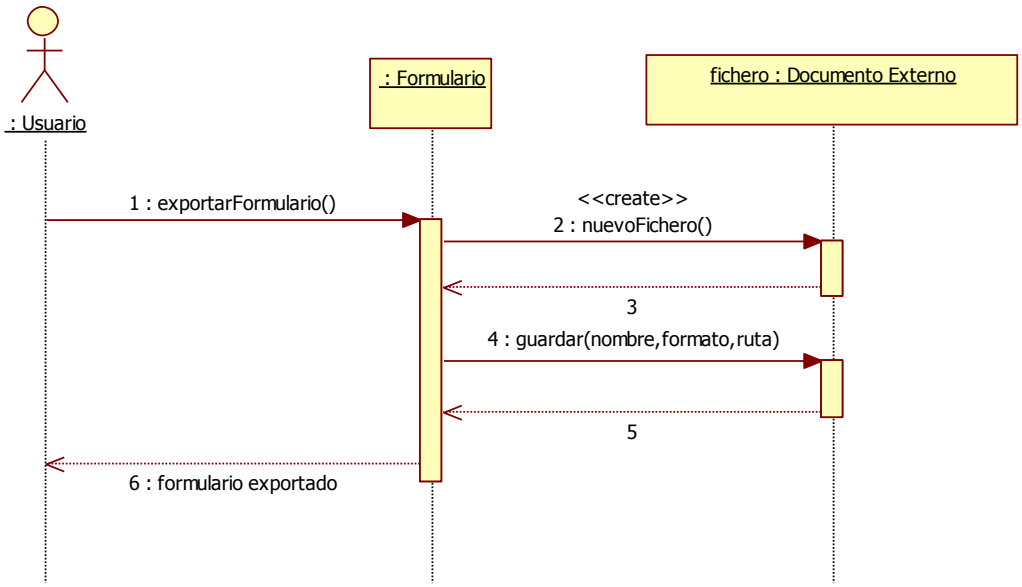

Ilustración 4.7 - <UC-006> Exportar Formulario

4.1.3.7. <UC-007> Login

4.1.3.7.1. Diagrama de Secuencia

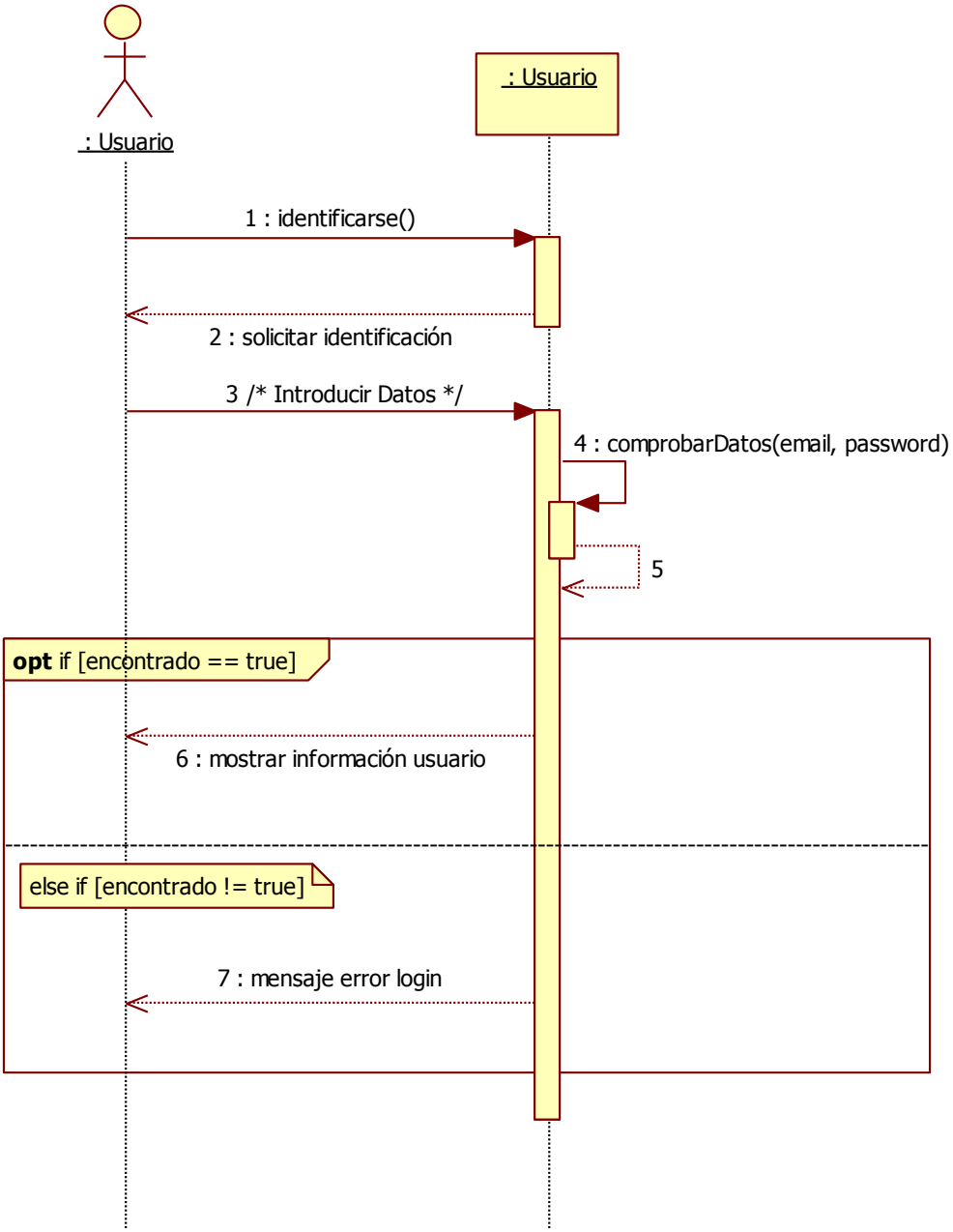

Ilustración 4.8 - <UC-007> Login

4.1.3.8. <UC-008> Registrarse

4.1.3.8.1. Diagrama de Secuencia

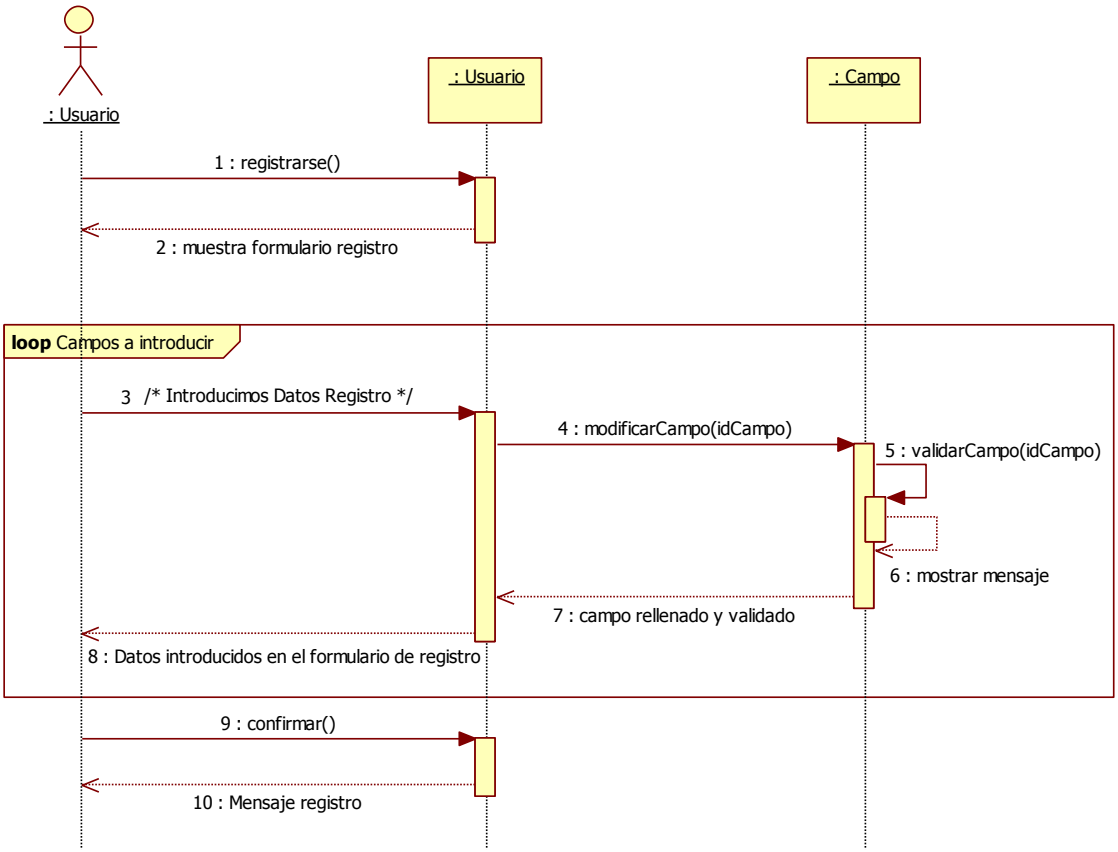

Ilustración 4.9 - <UC-008> Registrarse

4.2. ARQUITECTURA SOFTWARE

4.2.1. Introducción

4.2.1.1. Propósito

Este documento muestra una descomposición arquitectónica del sistema indicando las partes bien diferenciadas y más importantes.

4.2.1.2. Alcance

Se trata de exponer las principales características a nivel de arquitectura en cuando a la herramienta "XGen". La arquitectura del sistema se dividirá en subsistemas más pequeños y manejables. Estos subsistemas se descomponen en paquetes que contienen clases y cuyas clases interaccionan entre sí y así lograr la funcionalidad del sistema.

4.2.1.3. Estructura del apartado

Organización del Sistema	Se expone la forma de organizar o estructurar nuestro sistema.
Representación Arquitectónica	Se expone la arquitectura que se adapta mejor a nuestro sistema.
Objetivos y Restricciones Arquitectónicas	Abarca el conjunto de requisitos y objetivos que tienen algún tipo de impacto sobre la arquitectura del sistema, así como las restricciones que se aplicarán al diseño y a la implementación.
Vista de Casos de Uso	Lista los casos de uso o escenarios del modelo de casos de uso que representan la funcionalidad principal del sistema.
Vista de Despliegue	Describe la configuración de la red física sobre la que se desplegará y funcionará el sistema.

Tabla 12 - Estructura del apartado

4.2.2. Organización del Sistema

La organización del sistema refleja la estrategia básica usada para estructurar dicho sistema. Deben tomarse decisiones sobre la totalidad del modelo organizacional de un sistema al principio del proceso de diseño arquitectónico.

Tres de los estilos existentes más importantes que conforman nuestro espacio de diseño son: Repositorio, Cliente-Servidor y Capas. Estos estilos se pueden combinar según el sistema que estemos modelando. Recopilando información de diversas fuentes bibliográficas nos decantamos por un modelo de sistema Cliente-Servidor, este modelo se organiza como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios.

Los principales componentes de este modelo son:

- Un conjunto de servidores que ofrecen servicios a otros subsistemas.
- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.
- Una red de comunicaciones que permiten acceder a los clientes a estos servicios.

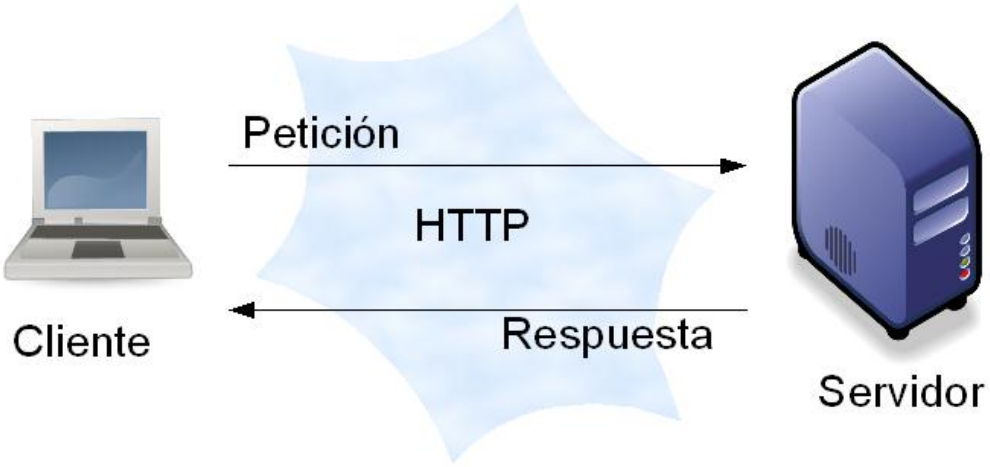

Ilustración 4.10 - Cliente/Servidor

Los clientes acceden a los servicios proporcionados a través de llamadas a procedimientos remotos usando el protocolo de petición-respuesta tal como marca el protocolo HTTP usado en WWW. Básicamente, un cliente realiza una petición a un servidor y espera hasta que recibe una respuesta.

Por ello, como nos interesa que el cliente pueda acceder al sistema desde el mayor número posible de medios, la distribución es una característica obligada y además sabemos que la arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico. Como la ventaja más importante del modelo Cliente-Servidor es que es una arquitectura distribuida, se puede hacer uso efectivo de los sistemas en red con muchos procesadores distribuidos. Es fácil añadir un nuevo servidor e integrarlo con el resto del sistema o actualizar los servidores sin afectar al resto del sistema.

Con toda esta información encontramos como principales ventajas la centralización del control de la aplicación, ya que todo el control estará alojado en la parte de la lógica de aplicación del servidor. Además al estar altamente organizado tenemos una facilidad tanto para la escalabilidad del sistema como para el mantenimiento del mismo.

Aunque sin embargo, puede ser necesario realizar cambios a los clientes y servidores existentes para obtener los mayores beneficios debido a la integración de un nuevo servidor. Por ello una desventaja que podemos encontrar es una alta congestión del tráfico pero que al requerir pequeñas cantidades de información no es un requisito crítico.

4.2.3. Representación Arquitectónica

La arquitectura software de un sistema es el diseño de más alto nivel de la estructura de un sistema. Define de manera abstracta las tareas que van a llevar a cabo los componentes, sus interfaces y la comunicación entre ellos. Suelen existir una serie de patrones arquitectónicos que definen y proporcionan un marco general.

Nuestro caso se basa en una arquitectura propuesta por Microsoft. Procederemos a continuación a exponer brevemente las características principales de este *Framework .NET*:

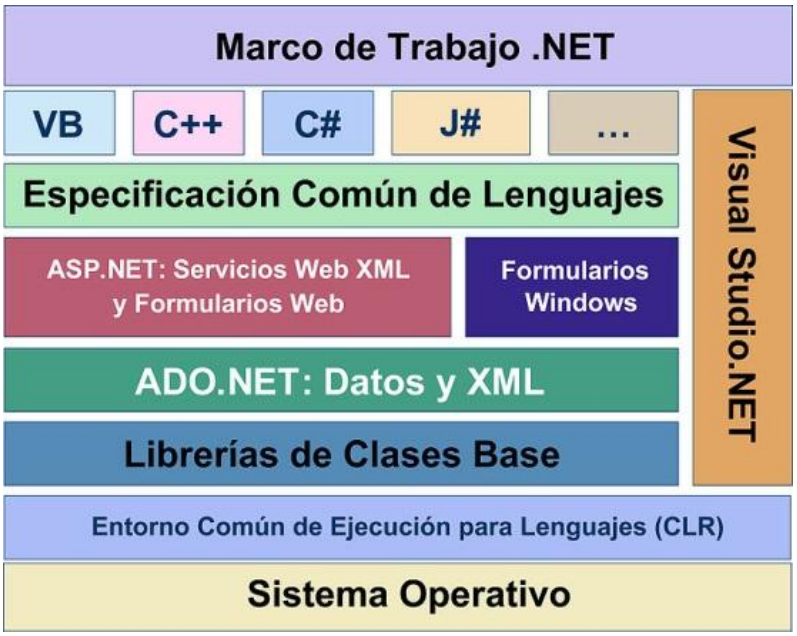

Ilustración 4.11 - Framework .NET de Microsoft

El entorno común de ejecución (Common Language Runtime, CLR) administra el código en tiempo de ejecución y proporciona los servicios básicos (administración de memoria, control de excepciones, control de hilos de ejecución).

Biblioteca de clases base (Base Class Library, BCL) es una colección de código OO que puede ser empleado desde cualquier lenguaje .NET. Contiene los tipos básico, clases para entrada/salida, seguridad, etc. Al tener definidos los tipos de datos para todos los lenguajes, facilita el intercambio de datos entre aplicaciones desarrolladas en distintos lenguajes.

La capa de datos y XML gestiona el acceso a los datos y el tratamiento de datos XML. Los datos los gestiona mediante ADO.NET y gran parte de la información de .NET (configuración, estructura de archivos y de aplicaciones) se gestiona mediante XML. También aporta facilidad para importar, exportar y tratar datos de/hacia XML.

ASP.NET utiliza Web Forms (para aplicaciones Web basadas en ASP) y los servicios Web. Windows Forms proporciona un conjunto de componentes de interfaz para desarrollar aplicaciones cliente basadas en Windows.

Nuestra aplicación se basa en una aplicación interactiva en tiempo real con el usuario, lo cual ha sido determinante para escoger un patrón arquitectónico Modelo-Vista-Controlador (MVC).

Existen otras alternativas para sistemas interactivos, pero el MVC seguramente sea el más difundido y el mejor estudiado. Además, el Framework desde el cual vamos a trabajar está especialmente adaptado para implantar un patrón arquitectónico MVC (http://www.asp.net/mvc).

El MVC surge de la necesidad de separar la lógica de presentación (interfaz) de la lógica de negocio (procesamiento), ya que la interfaz tiende a cambiar con más frecuencia que el modelo (diferentes formas de mostrar los datos, por ejemplo). Además la interfaz de usuario normalmente se descompone en dos partes: presentación y actualización.

Dado esto, la aplicación interactiva se divide en tres áreas: procesamiento, entradas y salidas. El procesamiento está representado por el modelo, que contiene la funcionalidad principal y los datos. La vista muestra la información al usuario, es decir, la salida. Como mecanismo de recogida de eventos, modificación del modelo y actualización del modelo, entradas, se idea el controlador.

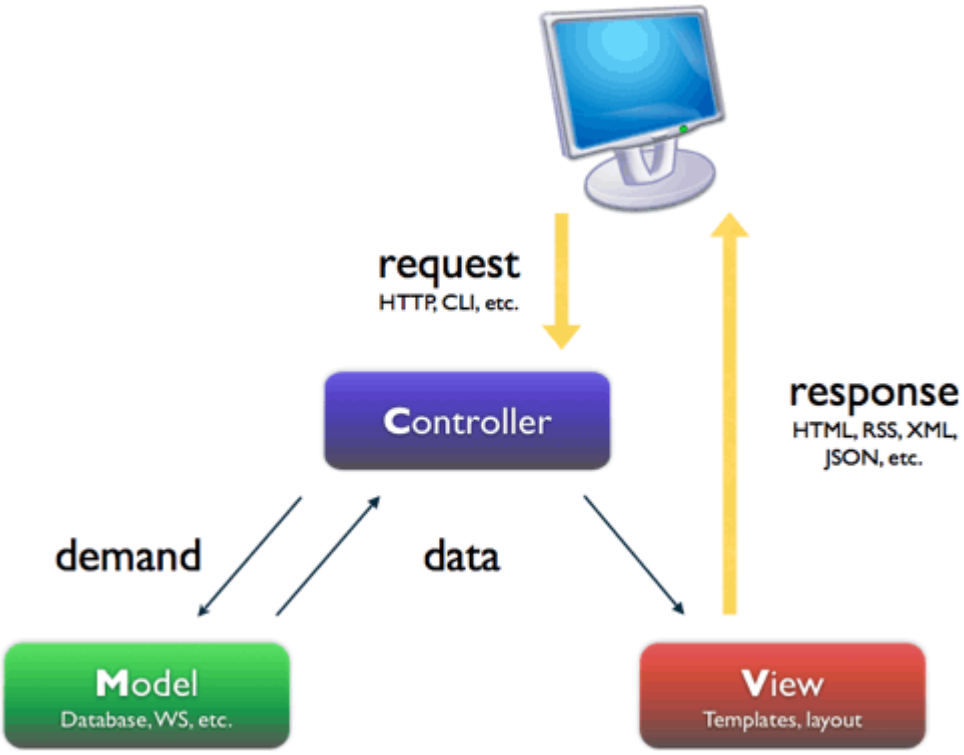

Ilustración 4.12 - Patrón Arquitectónico MVC

Este patrón (MVC Pasivo) se usa cuando un controlador manipula el modelo de forma exclusiva. El controlador modifica el modelo e informa a la vista de que el modelo ha cambiado y debe ser actualizado. El modelo en este escenario es completamente independiente de la vista y del controlador, no existe forma de que notifique los cambios en su estado, por eso el MVC aplicado como se ha descrito se denomina pasivo.

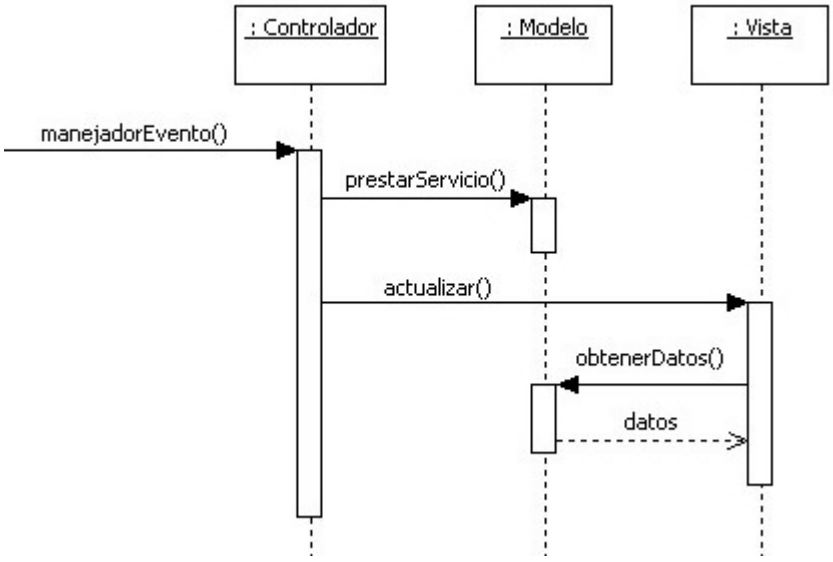

Ilustración 4.13 - MVC Pasivo

Como framework disponemos de ASP.NET MVC Framework, basado en ASP.NET, permite construir aplicaciones web teniendo como referencia el patrón MVC. ASP.NET MVC es, básicamente, una implementación del patrón Modelo - Vista - Controlador para tecnología ASP.NET. El patrón MVC no es ni nuevo (data de finales de los años 70) ni está pensado para aplicaciones web, pero en realidad en aplicaciones web encaja perfectamente.

Todos los aspectos transversales de ASP.NET (autenticación, cache, sesión, roles, etc.) siguen siendo los mismos en ASP.NET MVC.

Si a algo puede sustituir ASP.NET MVC es a Webforms, es decir a las páginas .aspx. Pero Microsoft ya ha anunciado que esto no sucederá y ambos frameworks (ASP.NET MVC y Webforms) se seguirán evolucionando.

4.2.4. Objetivos y Restricciones Arquitectónicas

A continuación expondremos algunos de los objetivos principales que se reflejaron en los requisitos software y que están relacionados con la arquitectura software.

4.2.4.1. Facilidad de Uso

El sistema debe ser intuitivo y fácil de usar, accesible a usuarios con conocimientos básicos del manejo de ordenadores. Esto también conlleva una interfaz agradable a la vista.

En la web principal (http://www.asp.net/mvc) existen numerosos tutoriales de cómo estructurar la interfaz y los documentos.

Además se aplicarán algunas técnicas de HCI (Human-Computer Interaction) de forma que saquemos partido a través de un desarrollo web. Algunas técnicas que se aplicarán serán por ejemplo:

- Correcta estructuración de menús
- Combinación de colores correcta de forma que sea llevadero para el usuario.
- Proporcionar feedback sobre la navegabilidad del sistema

Para este desarrollo disponemos de la herramienta IDE *Visual Studio Express 2012 for Web*, de forma que nos facilite en mayor medida el desarrollo de la aplicación.

4.2.4.2. Facilidad de Mantenimiento

La decisión de utilizar el patrón arquitectónico Modelo-Vista-Controlador nos permite modificar el modelo teniendo que cambiar únicamente zonas localizadas del controlador y, tal vez, de la vista.

Además deja abierta la posibilidad de ampliar el sistema con otras vistas diferentes sin que el modelo tenga que modificarse para nada.

4.2.4.3. Sistemas de Ayuda

El sistema contará con un sistema de ayuda, que contendrá tanto explicaciones textuales como gráficas, de forma que el usuario pueda resolver todas las incógnitas que se le presenten durante la utilización del mismo. Crearemos un sistema de ayuda web de forma que el usuario pueda acceder desde su navegador.

4.2.5. Vista de Casos de Uso

Ver el siguiente apartado *Realización de Casos de Uso*.

4.2.6. Vista de Despliegue

La arquitectura de una aplicación Web es similar a la de un sitio Web, se basa en el modelo Cliente/Servidor. Como en el caso del sitio Web, tenemos el navegador en la parte del cliente, el servidor Web en la parte del servidor y una conexión de red. Pero en las aplicaciones Web hay que considerar que existe una lógica de negocio sensible a las interacciones del usuario.

Originariamente la Web únicamente contaba con contenidos estáticos. La necesidad de ofrecer servicios más sofisticados a través de este medio llevó a soluciones tecnológicas que permitiesen cierto grado de interacción con el servidor más allá de solicitar una página en concreto.

A medida que la sofisticación de los servicios ofrecidos vía Web ha ido aumentando, se ha ido añadiendo, al menos a nivel lógico y no necesariamente a nivel físico, nuevos elementos en la arquitectura de las aplicaciones Web (por ejemplo el servidor de aplicaciones, bases de datos, etc…). Las nuevas disposiciones de elementos se han basado en patrones arquitectónicos del contexto de las aplicaciones distribuidas.

Hemos optado por una arquitectura en tres niveles de forma que podamos descargar cierta carga de trabajo en la capa del cliente. De esta forma dividimos la funcionalidad para optimizar el uso de recursos. Se consiguen soluciones mucho más flexibles y escalables. Los tres niveles son:

- **Cliente**: Contiene los componentes de usuario que son únicos para cada uno de ellos. Esto es la lógica de aplicación específica del usuario y la interfaz.
- **Aplicación**: Constituye un entorno multiusuario y mantiene las partes compartidas de la aplicación. Las operaciones con un uso intensivo de datos deben ejecutarse en este nivel. También es el punto donde se puede llevar a cabo la coordinación de transacciones (operaciones de múltiples usuarios)
- **Almacenamiento**: Es el nivel de la base de datos. Se especializa en dar un servicio de persistencia a los datos de la aplicación y permite manejar grandes volúmenes de ellos.

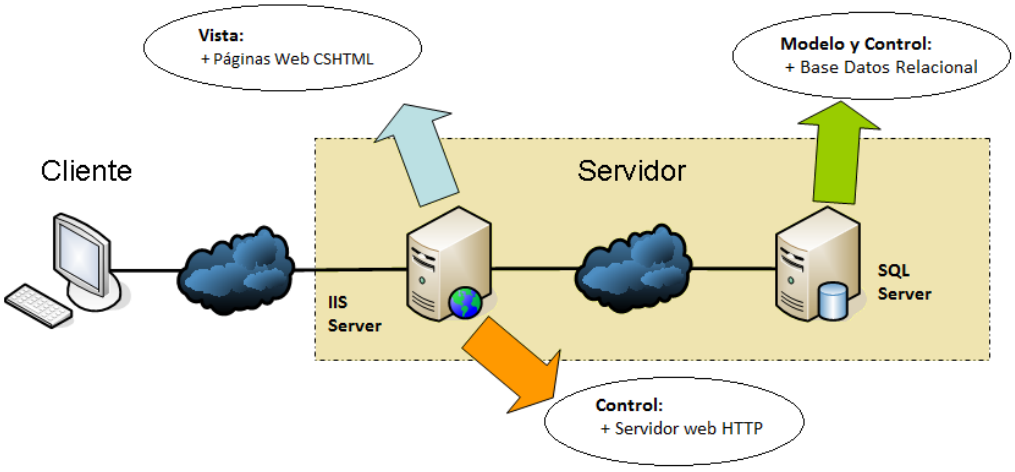

Ilustración 4.14 - Arquitectura Física

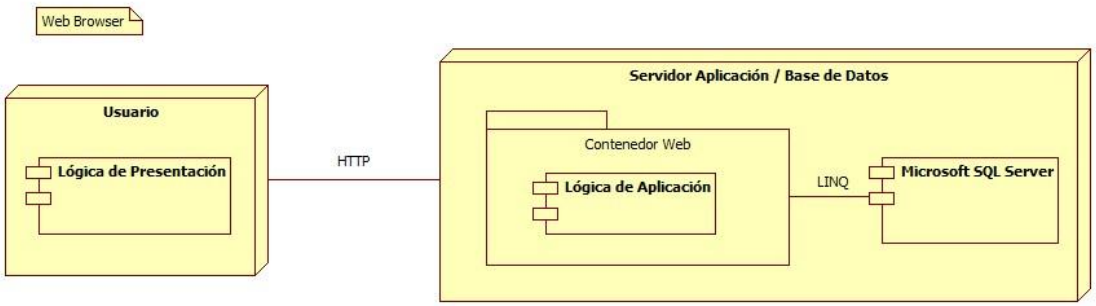

Ilustración 4.15 - Diagrama de Despliegue

Al mantener buena parte del código en el nivel intermedio, la aplicación se aísla de la interfaz de usuario y de la base de datos. Por lo tanto, se pueden hacer cambios en el código en el nivel de aplicación sin que esto afecte al resto. Además, los datos están centralizados y fácilmente accesibles sin necesidad de moverlos completamente hasta el cliente.

Este modelo se utiliza ampliamente en el entorno Web. Se incorpora la figura del servidor de aplicaciones que permite que el sistema gestione la lógica de negocio y el estado. Al ser una aplicación Web, la interfaz con el cliente sigue siendo básicamente HTTP y por lo tanto el servidor de aplicaciones incorpora un servidor Web.

La división de la aplicación en niveles supone la necesidad de establecer interfaces entre ellas. Si además se quiere construir una arquitectura independiente de los fabricantes la mejor opción es utilizar elementos que implementen interfaces estándar.

Para la interfaz entre el servidor de aplicaciones y la base de datos las opciones son múltiples, pero nosotros vamos a utilizar el lenguaje de programación C#, el tipo de base de datos es relacional y la base de datos concreta es Microsoft SQL Server. Por lo que respecta a la interfaz entre el cliente y el servidor de aplicaciones, la opción básica sigue siendo HTTP. Existen otras opciones que permiten un mayor grado de interactividad entre el código en el cliente y en el servidor.

Toda la lógica de negocio se mantiene en el servidor. Esto permite tener clientes que sólo requieren el uso de un navegador Web sencillo. Básicamente HTML, formularios para la entrada de datos y lenguajes de script (VBScript, JavaScript,...) para el control previo de los datos introducidos, p.e. validar un NIF. Ésta es la solución que supone una mayor carga de proceso y comunicaciones, la interacción entre el cliente y el servidor Web se basa en una sucesión de pregunta/respuesta síncronas/asíncronas.

A continuación explicamos la estructura de nuestra aplicación XGen:

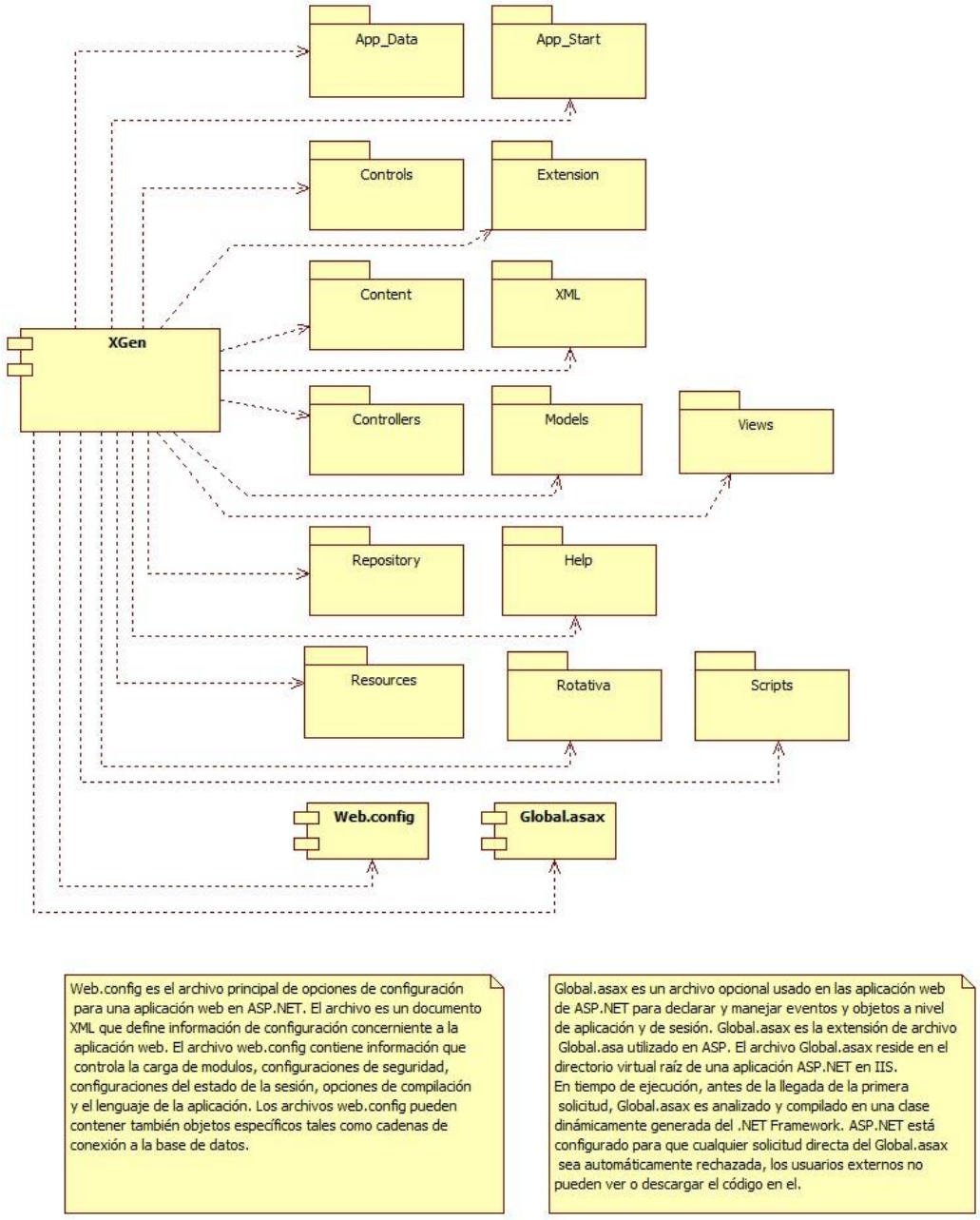

Ilustración 4.16 - Estructura de la Aplicación

La aplicación XGen accede y se sirve de diferentes elementos para su funcionamiento como son *App_Data* y *App_Start* (predefinidos por .NET), *Controls* (opciones de validación del formulario), *Extension* (extensión del plugin flexigrid), *Content* (Imágenes y ficheros CSS), *XML* (ficheros XML cuyo contenido son los formularios), *Repository* (contiene la clase que gestiona la base de datos), *Help* (ayuda del sistema), *Resources* (aplicación en varios idiomas), *Rotativa* (librería de exportación PDF) y *Scripts* (ficheros javascript de distintas librerías).

Además de los elementos anteriormente comentados, .NET nos aporta dos tipos de ficheros predefinidos (*Web.config* y *Global.asax*), que explicamos detalladamente en la ilustración 4.16.

4.2.7. Vista Lógica

4.2.7.1. Estructura General

En este apartado se describirán las partes fundamentales del modelo de diseño en cuanto a descomposición en paquetes y/o subsistemas. Dentro de dichos paquetes y/o subsistemas existirán clases que se describirán de la misma forma.

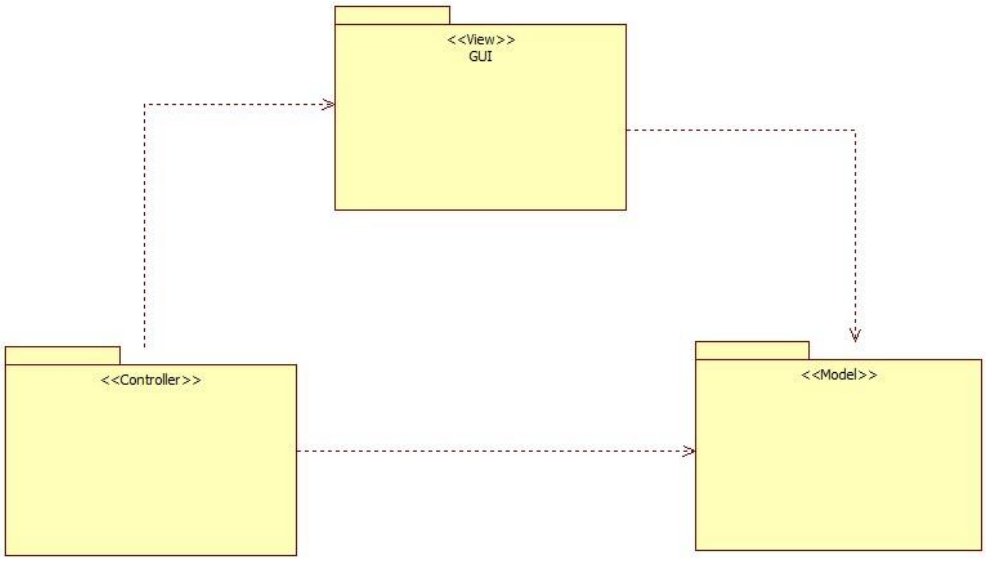

Ilustración 4.17 - Descomposición en Paquetes (1)

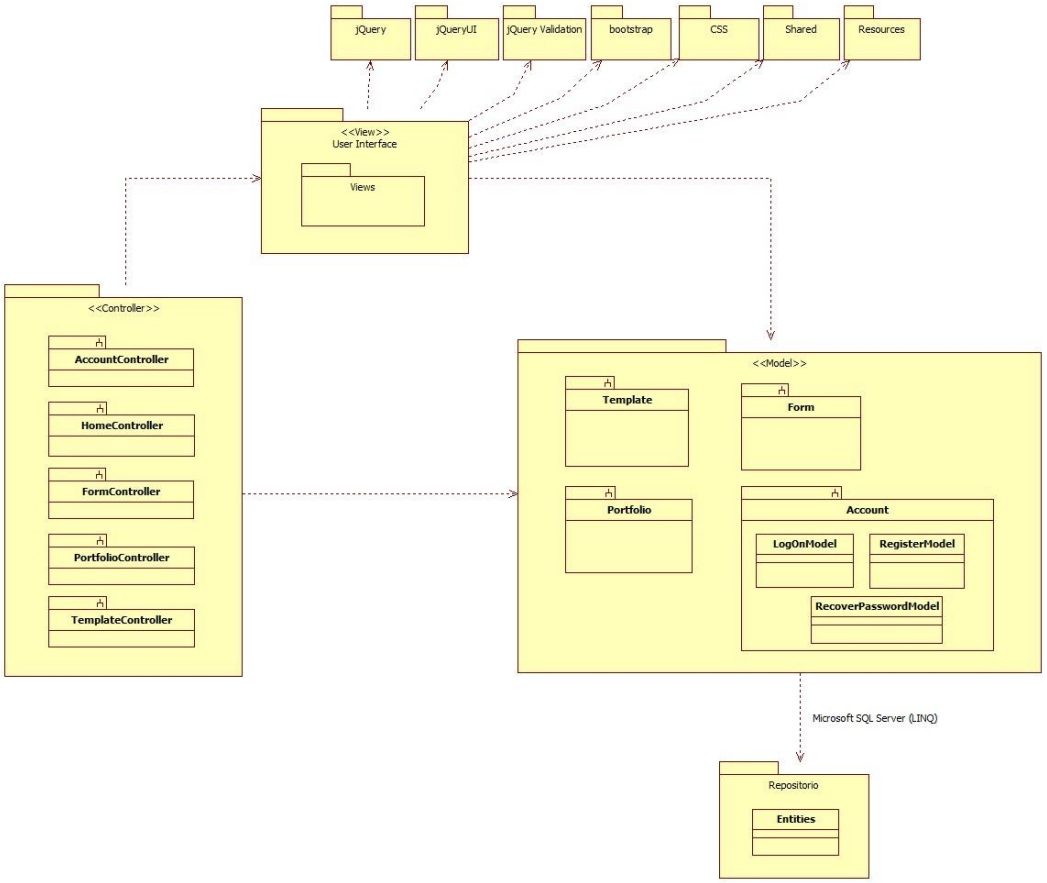

Ilustración 4.18 – Descomposición en Paquetes (2)

4.2.7.2. Descomposición del paquete View

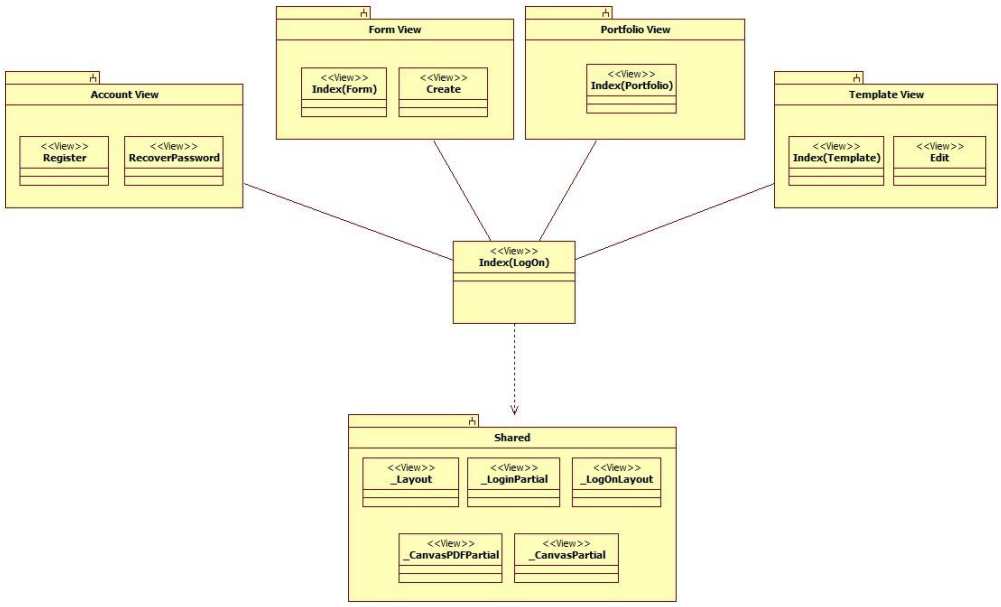

Ilustración 4.19 - Paquete View

4.2.7.3. Descomposición del paquete Controller

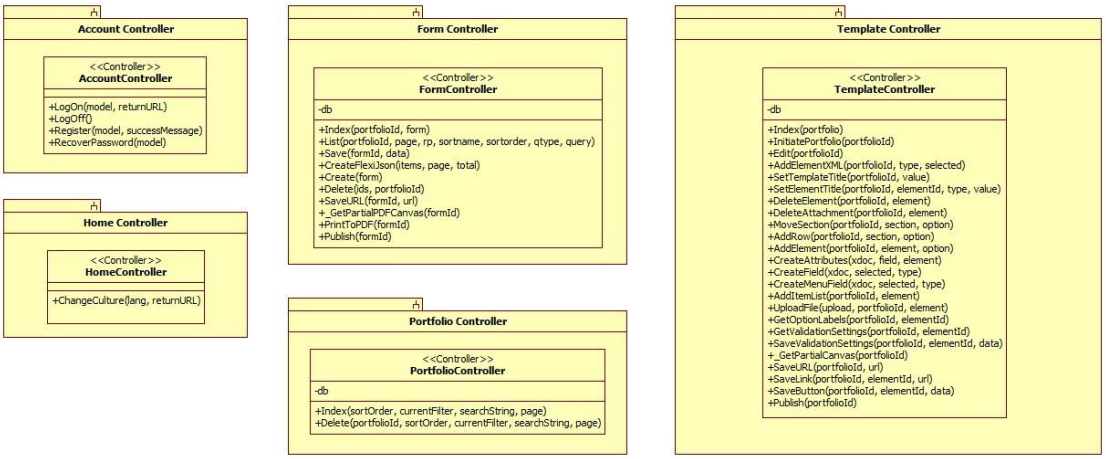

Ilustración 4.20 - Paquete Controller

4.2.7.4. Descomposición del paquete Model

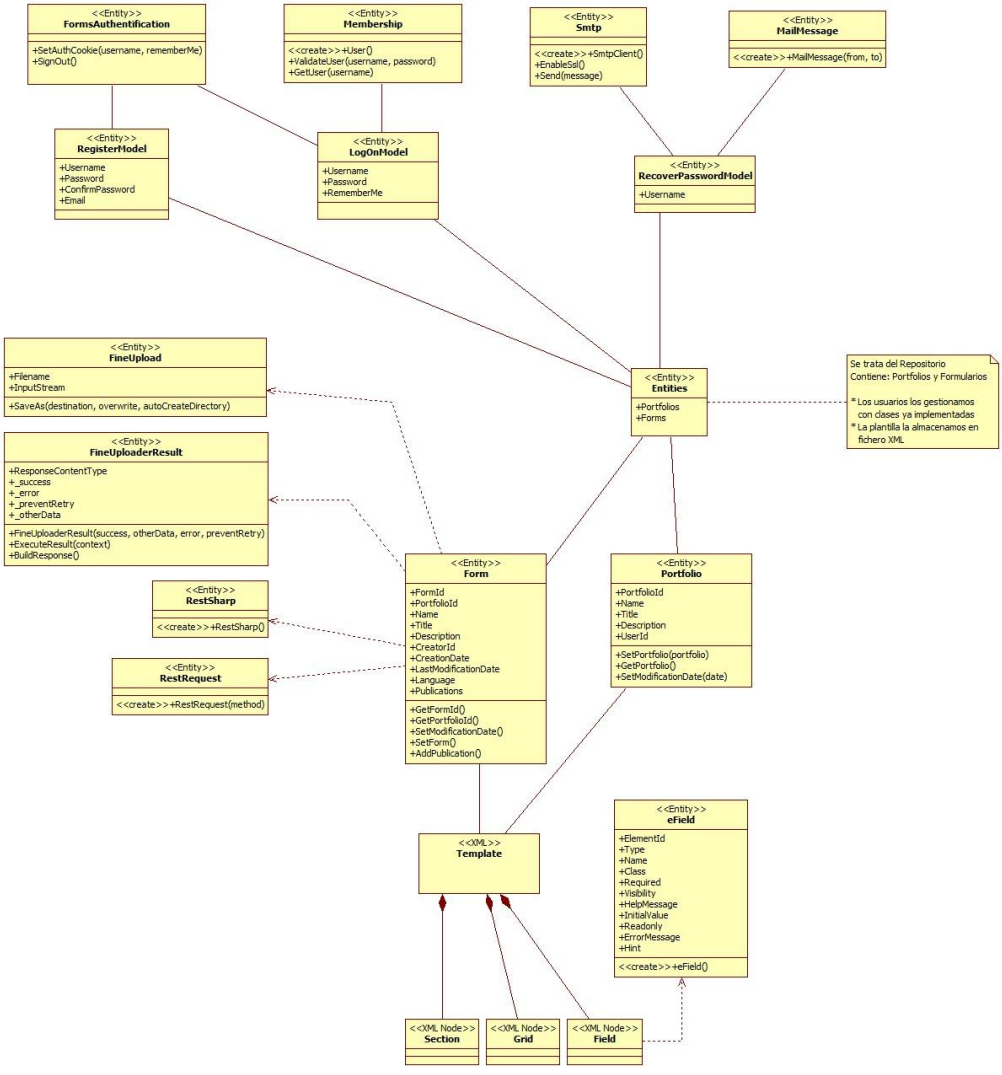

Ilustración 4.21 - Paquete Model

4.2.8. Vista de Datos

En nuestro sistema existen una serie de elementos que se almacenarán en la base de datos del sistema. A continuación describiremos la estructura de estos elementos de forma que podamos exponer la forma en la que se generan.

4.2.8.1. Especificación XForms

Explicaremos a continuación la estructura de la especificación **XForms**. El modelo XForms tiene la capacidad de trabajar con una gran cantidad de estándares o interfaces de usuario como mostramos a continuación:

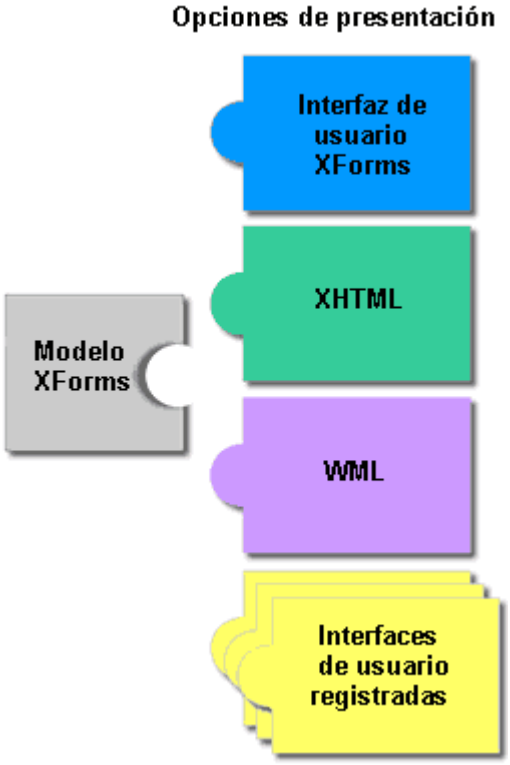

Ilustración 4.22 - Modelo XForms

La **Interfaz de usuario de XForms** proporciona un conjunto estándar de controles visuales con el objetivo de reemplazar los controles del actual XHTML.

Los formularios de XForms recogen datos, lo que se expresa como **instance data**, datos de instancia XML. **XForms Model** se encarga de describir la estructura de esos datos de instancia, lo que muestra un intercambio estructurado de datos.

Finalmente, tiene que existir un canal por el que los datos de instancias se muevan hacia y desde un procesador XForms. Para ello, el **Protocolo de envío de XForms** define la forma en la que XForms envía y recibe datos, incluyendo la capacidad de cancelar y reanudar la finalización de un formulario:

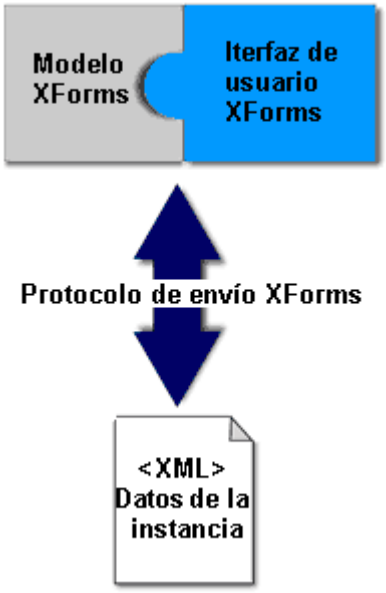

Ilustración 4.23 - Visión Global XForms

4.2.8.2. Propuesta XForms

A continuación exponemos la manera de gestionar los formularios en nuestro sistema, teniendo en cuenta que se tiene que ajustar a la especificación XForms. Como bien sabemos, lo que a la especificación XForms propone es básicamente una división Modelo, Vista y Controlador:

- Un **Modelo** que contenga la definición de los datos así como elementos extras, en nuestro caso atributos que sirven como validación de los campos pertenecientes al formulario. Como es obvio, toda la definición de los datos del formulario se encontrarán en un documento **XML**.

- Una **Vista** que sirve como presentación de los datos y de los controles necesarios en la interfaz de usuario de forma que se pueda gestionar el formulario correctamente y de forma eficiente.

- Un **Controlador** de forma que gestione la información contenido en el formulario. Sirve como intermediador entre el modelo y la vista, atendiendo a los eventos producidos por parte del usuario.

Lo que básicamente tenemos que implementar es un sistema que genere un fichero XML que contenga la información necesaria como por ejemplo campos de formularios junto con atributos. Los atributos pueden ser de diferentes tipos, existen atributos de definición (identificador, nombre y tipo) y atributos de validación (campo de sólo lectura, campo requerido, mensaje de error en caso de que el usuario no rellene correctamente el campo, etc.)

4.2.8.3. Creación y Edición de la Plantilla de Formulario

En este apartado se expone la forma en la que realizamos tanto la creación de una plantilla de formulario como la edición del mismo. Esquemáticamente lo podemos representar de la siguiente manera:

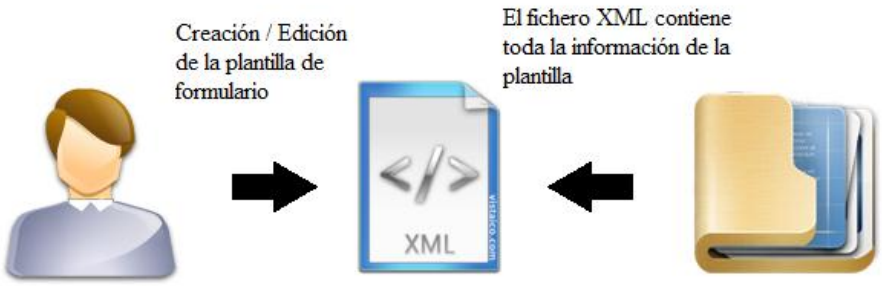

Ilustración 4.24 - Vista de Datos - Creación/Edición de la Plantilla de Formulario

El usuario crea una plantilla. Esta plantilla obtiene toda su información de un fichero XML (recién creado en caso de que el usuario esté creando un nuevo portfolio) y dicha información se interpreta de forma visible para el usuario. El usuario en cualquier momento es capaz de modificar dicha estructura, modificando al mismo tiempo la información contenido en el fichero XML.

4.2.8.4. Creación y Edición del Formulario

En este apartado se expone la forma en la que realizamos tanto la creación de un formulario como la edición del mismo. Esquemáticamente lo podemos representar de la siguiente manera:

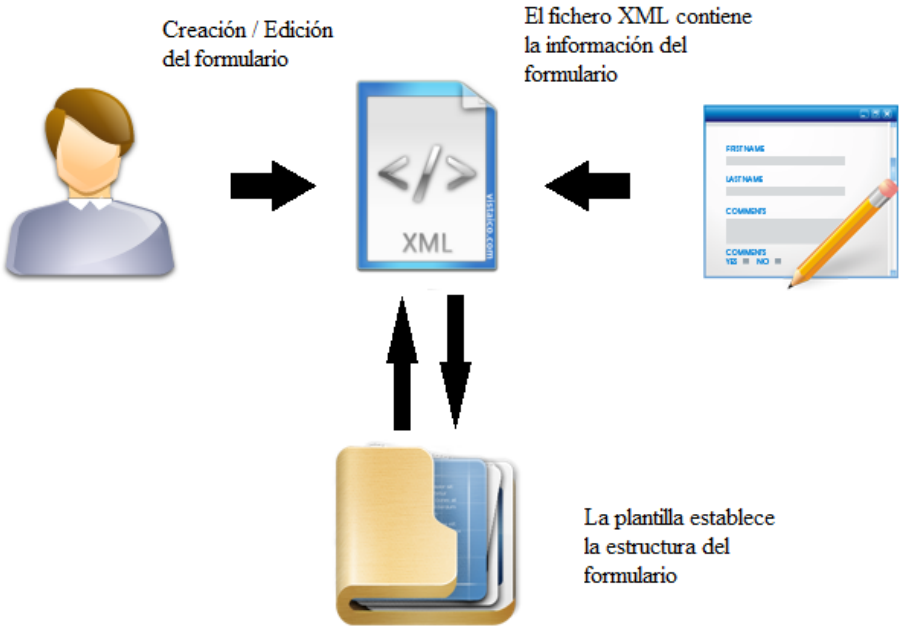

Ilustración 4.25 - Vista de Datos - Creación/Edición de Formulario

El usuario crea un formulario. Este formulario obtiene toda su información del fichero XML (que contiene la estructura de la plantilla previamente establecida) y es interpretada de forma visible, permitiendo al usuario modificar la información del formulario.

4.2.8.5. Exportación como documento externo (PDF)

Se trata del formulario exportado como documento en formato PDF. Este formulario basado en **XForms** se exportará utilizando una librería adecuada.

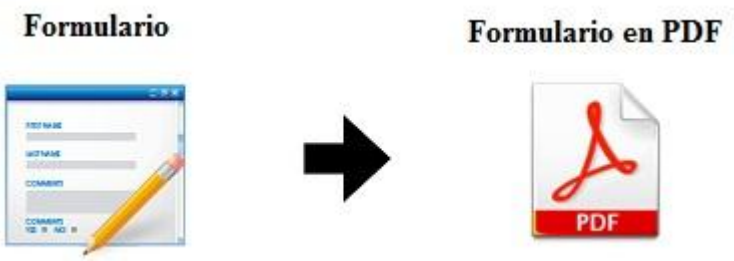

Ilustración 4.26 – Formulario en formato PDF

4.2.8.6. Publicación de formulario

Para la publicación del formulario básicamente el proceso consiste en enviar el fichero XML correspondiente al formulario a través de una petición HTTP al servidor en cuestión. El usuario (mediante una ventana emergente) puede cambiar esta dirección del servidor, pudiendo enviar el mismo formulario a varios servidores distintos. Lo representamos gráficamente de la siguiente manera:

Ilustración 4.27 - Publicar Formulario en Internet

A la hora de la publicación de nuestro formulario (fichero XML), se pueden dar muchos casos por el cual el envío sea defectuoso (no disponibilidad del servicio, mala recepción de los datos, fallos en la conexión, etc.), que en cualquier caso se le avisará al usuario. Esto es debido a que no depende sólo de nuestro servidor sino también de servicios externos que no podemos controlar. Una vez enviado el formulario al servidor externo, la recepción también es algo que no depende de nuestro sistema y que por lo tanto tampoco podemos controlar. Como consecuencia, el usuario receptor del formulario tendrá que implementar su propio sistema de recepción para obtener el fichero XML.

La decisión de usar POST en vez de GET es que el método POST es más versátil. Por ejemplo, puedes enviar un archivo completo usando POST. También, el tamaño de los datos enviados no está limitado como en el caso de GET.

4.3. Descripción de la Interfaz

4.3.1. Introducción

4.3.1.1. Propósito

Este apartado refleja la organización y estructura de la interfaz de usuario. La interfaz de usuario es con lo que el usuario interaccionará para poder usar el sistema. La interfaz consta de una serie de elementos como pueden ser menús o botones que serán útiles al usuario.

4.3.1.2. Alcance

Se establecerá un prototipo de interfaz de usuario de forma que se tendrá en cuenta todos los roles que utilizarán dicho interfaz. Por ejemplo, el probador utilizará este prototipo para planificar sus actividades de prueba. El diseñador tendrá que evaluar y comprender la interfaz de forma que posteriormente pueda ser implementada. Por otro lado, el usuario podrá percibir dicha interfaz como una pequeña parte de lo que será la interfaz final.

4.3.1.3. Estructura del apartado

El apartado se organizará como mostramos a continuación:

Elementos visibles	Son características que siempre son mostradas al usuario
Elementos de interacción	Se tratan de los elementos de interfaz que permiten seleccionar algunos otros elementos o que contengan y muestren información.
Elementos de Ayuda	Pantalla del sistema de ayuda

Tabla 4.3 - Estructura del apartado

Podemos fijarnos en que incorporado al navegador y por defecto, se mostrarán una serie de componentes como son los siguientes:

- **Barra de Título**: Es la primera barra de la ventana, donde aparece el título de la página Web.
- **Barra de Menú**: Desde los menús de esta barra se puede trabajar con todos los comandos del navegador. Al hacer clic sobre un menú se despliegan las opciones correspondientes para que elijamos una de ellas.
- **Barra de Herramientas**: Contiene iconos para ejecutar de forma inmediata algunos de los comandos más habituales que nos facilitan la navegación, como Atrás, Adelante, Actualizar, etc.
- **Barra de Direcciones**: Muestra la dirección de navegación que se actualiza cada vez que cambiamos de pantalla.
- **Barra de Estado**: Contiene información sobre el estado de la página Web. Indica si la página se está cargando, o si faltan algunas imágenes por cargarse.

4.3.2. Elementos Visibles

4.3.2.1. Barra de Opciones

4.3.2.1.1. Propósito

Esta barra permanece activa y visible durante toda la duración de la sesión de usuario. Se ofrece funcionalidad básica del sistema como puede ser login (entrar en el sistema) o logout (salir del sistema), registrarse, cambio de idioma o consulta de la ayuda del sistema.

4.3.2.1.2. Objetos y Acciones

La barra de opciones se descompone de una serie de opciones:

- **Inicio**: Esta opción aporta un enlace (haciendo click en el logo principal de la aplicación) a la página principal desde cualquier punto de navegabilidad del sistema.

- **Logout:** Esta opción aparece en lugar de login cuando el usuario está dentro del sistema. Haciendo click en este enlace el usuario cerraría sesión.
- **Idioma**: Debido a la creciente proliferación de aplicaciones web y el acceso alrededor del mundo, se da la posibilidad de cambiar de idioma.
- **Ayuda**, al pulsar sobre ella se muestra:
  - *Contenidos de la Ayuda*: abre en una pestaña nueva, independiente de la la aplicación, un sistema de ayuda con tabla de contenidos, índice y buscador propios.

4.3.2.1.3. Objetos Compuestos

La mayoría de los objetos enumerados en el apartado anterior se componen a su vez de cuatro elementos:

- **Icono**: una imagen que representa la acción a realizar. Cada imagen está asociada a una sola acción, y cada acción a una sola imagen (tal vez en diferentes tamaños). Este icono indicará por si solo la acción a realizar.
- **Nombre de la acción**: lo más breve posible, para economizar espacio.
- **Descripción de la acción**: Esta descripción aparece cuando se deja el cursor sobre la acción. Se trata de una descripción un poco más extendida pero que no ocupará más de dos líneas.

4.3.2.1.4. Apariencia

Este elemento se posicionará en la parte superior de la ventana del navegador. Siempre permanecerá visible y no se podrá redimensionar. Los elementos emergentes se desplegarán hacia abajo (en caso por ejemplo del idioma).

Estos elementos desplegables tendrán la anchura igual que el elemento *idioma*.

4.3.2.2. Barra de Menú

4.3.2.2.1. Propósito

Ofrece al usuario una serie de opciones mediante las cuales pueden trabajar con elementos como portfolios o plantillas. Ejemplo: *Creación de un Portfolio* o *Modificación de una plantilla de formulario*.

4.3.2.2.2. Objetos y Acciones

Varía de forma dinámica los elementos que puede contener y sirven para realizar acciones sobre los objetos del sistema.

- **Atrás**: Haciendo click en este enlace, nos llevaría a la pantalla anteriormente visitada.
- **Nuevo Portfolio**: Haciendo click en este enlace creamos un nuevo portfolio.
- **Editar Plantilla**: Haciendo click nos transportaría a la pantalla de construcción de la estructura del formulario.
- **Editar Configuración de la Plantilla**: Se abre una ventana emergente de forma que aparezcan las opciones de configuración de la plantilla.
- **Editar Configuración del Formulario**: Se abre una ventana emergente de forma que aparezcan las opciones de configuración del formulario.
- **Editar URL Action**: Se abre una ventana emergente de forma que permita modificar la URL action de la plantilla.

4.3.2.2.3. Objetos Compuestos

En esta ocasión cada objeto se compondrá de dos elementos:

- **Icono**: imagen asociada al elemento (portfolio o plantilla) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre para realizar acciones sobre los portfolios o sobre las plantillas. Se mostrará cuando el ratón esté sobre el icono.

4.3.2.2.4. Apariencia

Se trata de una barra o porción de la pantalla que se sitúa en la parte inferior a la barra de opciones de la aplicación. Será de un tamaño adecuado de forma que los iconos que contiene se puedan visualizar correctamente.

4.3.2.3. Barra de Elementos (Campos de Formulario)

4.3.2.3.1. Propósito

Ofrece al usuario una serie de elementos que puede insertar a la hora de la construcción de un formulario.

4.3.2.3.2. Objetos y Acciones

Los objetos representados en esta barra son todos aquellos campos que pueden ser insertados en un formulario. Si el número de ellos es demasiado grande, se tratará de agrupar los elementos por funcionalidad, de forma que no se ocupe gran parte de la pantalla.

4.3.2.3.3. Objetos Compuestos

En esta ocasión cada objeto se compondrá de dos elementos:

- **Icono**: imagen asociada al elemento (campo del formulario) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre del campo de formulario. En este caso, no hay dos elementos que se nombren de la misma forma.

4.3.2.3.4. Apariencia

Se trata de una barra o porción de la pantalla que ocupa la mayor parte del lateral izquierdo de la aplicación. Será suficientemente ancho como para que quepan los nombres de los elementos junto con el icono representativo.

4.3.2.4. Zona de Trabajo

4.3.2.4.1. Propósito

Ofrece al usuario los datos almacenados por él mismo (portfolio, formularios, etc.) y es la pantalla principal de nuestra aplicación. A la hora de construir una plantilla de formulario, podríamos decir que este área serviría como *canvas*.

4.3.2.4.2. Objetos y Acciones

Los objetos representados en esta barra son varios: lista de portfolios, lista de formularios o plantillas. Con los diferentes objetos comentados anteriormente estarán disponibles una serie de opciones como creación, lectura, modificación o borrado (CRUD).

4.3.2.4.3. Objetos Compuestos

Dependiendo del objeto con el que trabajemos, será distinto pero básicamente como se representa es de la siguiente forma:

- **Icono**: imagen asociada al elemento. (Ejemplo: Icono carpeta para el portfolio).
- **Nombre**: Se trata del nombre del objeto al que representa.

4.3.2.4.4. Apariencia

Se trata de un área de trabajo principal para nuestra aplicación. Varía de forma dinámica a medida que el usuario ejecuta diversas acciones.

4.3.3. Ventanas de Usuarios

4.3.3.1. Login

4.3.3.1.1. Propósito

Se trata de la ventana principal inicial de forma que permita al usuario loguearse en nuestra aplicación.

4.3.3.1.2. Objetos y Acciones

Los objetos representados en esta ventana son básicamente los campos del formulario login: *Usuario* y *Password*, junto con un botón para loguearse.

4.3.3.1.3. Objetos Compuestos

En esta ocasión cada acción que se puede ejecutar sobre los ítems se compondrá de dos elementos.

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata.
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.3.1.4. Apariencia

Esta pantalla ocupa la mayor parte y se trata de la parte inicial de nuestra aplicación. En ella se muestran los principales elementos del formulario login del sistema y es el principal elemento de interacción del usuario.

4.3.3.2. Registrarse

4.3.3.2.1. Propósito

Se trata de la ventana principal inicial de forma que permita al usuario registrarse en nuestra aplicación.

4.3.3.2.2. Objetos y Acciones

Los objetos representados en esta ventana son básicamente los campos del formulario de registro en el sistema: *Usuario, Password, Repetición del Password* e *Email*, junto con un botón para aceptar el registro.

4.3.3.2.3. Objetos Compuestos

En esta ocasión cada acción que se puede ejecutar sobre los ítems se compondrá de dos elementos.

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata.
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.3.2.4. Apariencia

Esta pantalla ocupa parte de la pantalla y en ella se muestran los principales elementos del formulario registro del sistema y es el principal elemento de interacción del usuario.

4.3.3.3. Recuperar Password

4.3.3.3.1. Propósito

Se trata de la ventana principal inicial de forma que permita al usuario recuperar el password.

4.3.3.3.2. Objetos y Acciones

El objeto representado en esta ventana es básicamente *Usuario*, junto con un botón para aceptar el envio del nuevo password.

4.3.3.3.3. Objetos Compuestos

En esta ocasión cada acción que se puede ejecutar sobre los ítems se compondrá de dos elementos.

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata.
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.3.3.4. Apariencia

Esta pantalla ocupa la mayor parte y en ella se muestran los principales elementos del formulario para la recuperación del password del sistema y es el principal elemento de interacción del usuario.

4.3.4. Ventanas de Portfolios

4.3.4.1. Mostrar Portfolios

4.3.4.1.1. Propósito

Se trata de la ventana principal en la que son mostrados los portfolios en forma de lista que conforman una tabla, junto con algunas características propias de cada portfolio como puede ser el nombre, el título y la fecha de última modificación.

4.3.4.1.2. Objetos y Acciones

Los objetos representados en esta ventana son ítems. Cada ítem corresponde a un elemento de una lista y pueden ser seleccionados independientemente. Las acciones a realizar podrán ser, modificar el portfolio o borrarlo. Existe también la posibilidad de ordenar la lista de portfolios así como realización de una búsqueda.

4.3.4.1.3. Objetos Compuestos

En esta ocasión cada acción que se puede ejecutar sobre los ítems se compondrá de dos elementos.

- **Icono**: imagen asociada al elemento (acción del formulario) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre de la acción a realizar.

4.3.4.1.4. Apariencia

Esta pantalla ocupa la mayor parte y se trata de la parte principal de la aplicación. En ella se muestran los principales elementos (portfolios) de esta pantalla.

4.3.4.2. Creación del Portfolio

4.3.4.2.1. Propósito

Definir las propiedades básicas de un portfolio como puede ser el nombre, el título y la descripción.

4.3.4.2.2. Objetos y Acciones

Como hemos comentado anteriormente, los objetos principales que aparecerán serán el nombre, título y descripción del formulario. Estos objetos serán 3 campos que el usuario tendrá que rellenar antes de continuar.

4.3.4.2.3. Objetos Compuestos

En esta ocasión cada objeto se compondrá de dos elementos:

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.4.2.4. Apariencia

Esta ventana de creación del portfolio será una ventana emergente de forma que se mostrará de forma principal frente a los demás componentes de la aplicación. En la parte inferior de la ventana emergente se mostrarán dos opciones de forma que el usuario pueda cancelar o aceptar la acción.

4.3.5. Ventanas de Formularios

4.3.5.1. Mostrar Formularios

4.3.5.1.1. Propósito

Se trata de la ventana principal en la que son mostrados los formularios en forma de lista que conforman una tabla, junto con algunas características propias de cada formulario como puede ser el nombre, el título, una breve descripción, la fecha de creación y de última modificación, idioma y número de publicaciones.

4.3.5.1.2. Objetos y Acciones

Los objetos representados en esta ventana son ítems. Cada ítem corresponde a un elemento de una lista y pueden ser seleccionados independientemente mediante un click, sombreándose el elemento seleccionado. Las acciones a realizar podrán ser, exportar a PDF, modificar, borrar o crear un nuevo formulario.

4.3.5.1.3. Objetos Compuestos

En esta ocasión cada acción que se puede ejecutar sobre los ítems se compondrá de dos elementos.

- **Icono**: imagen asociada al elemento (acción del formulario) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre de la acción a realizar.

4.3.5.1.4. Apariencia

Esta pantalla ocupa la mayor parte y se trata de la parte principal de la aplicación. En ella se muestran los principales elementos (formularios) de la pantalla y es el principal elemento de interacción del usuario.

4.3.5.2. Creación del Formulario

4.3.5.2.1. Propósito

Definir las propiedades básicas de un formulario como puede ser el nombre, el título, la descripción e idioma. Se puede indicar la aplicación que va a tener dicho formulario en la descripción.

4.3.5.2.2. Objetos y Acciones

Como hemos comentado anteriormente, los objetos principales que aparecerán serán el nombre, título, descripción e idioma del formulario. Estos objetos serán 4 campos que el usuario tendrá que rellenar antes de continuar.

4.3.5.2.3. Objetos Compuestos

En esta ocasión cada objeto se compondrá de dos elementos:

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.5.2.4. Apariencia

Esta ventana de creación del formulario será una ventana emergente de forma que se mostrará de forma principal frente a los demás componentes de la aplicación. En la parte inferior de la ventana emergente se mostrarán dos opciones de forma que el usuario pueda cancelar o aceptar la acción.

4.3.5.3. Creación/Edición de Formulario

4.3.5.3.1. Propósito

El propósito principal es la creación o edición de la información contenida en los campos de un formulario.

4.3.5.3.2. Objetos y Acciones

En la parte inferior del formulario existirá una serie de acciones que son las de cerrar, limpiar, exportar a PDF, guardar y ejecutar. Cada una de estas acciones se aplicarán al actual formulario que se está creando o actualizando.

Cada campo del formulario será identificado mediante su nombre y su valor si estuviera rellenado, además de una corta descripción.

4.3.5.3.3. Objetos Compuestos

En cuanto a los campos del formulario serán identificados con la información introducida en la pantalla de construcción por el usuario:

- **Nombre**: Se trata del nombre del campo del formulario
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.5.3.4. Apariencia

Se trata de la parte principal de la pantalla dónde el usuario puede crear/editar el formulario aparte de otra serie de opciones comentadas anteriormente en el apartado *Objetos y Acciones*. El usuario rellena los campos del formulario con el fin de guardarlo posteriormente.

4.3.5.4. Ejecución de Formulario

4.3.5.4.1. Propósito

El propósito principal es mostrar el formulario como producto final con los campos correctamente posicionados.

4.3.5.4.2. Objetos y Acciones

En la parte inferior del formulario existirá un botón para cerrar la ejecución del formulario. Será la única acción disponible y necesaria para esta ventana.

Cada campo del formulario será identificado mediante su nombre y su valor si estuviera rellenado, aparte de una pequeña descripción.

4.3.5.4.3. Objetos Compuestos

En esta ocasión cada objeto acción se compondrá de dos elementos:

- **Icono**: imagen asociada al elemento (acción del formulario) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre de la acción a realizar.

En cuanto a los campos del formulario:

- **Nombre**: Se trata del nombre del campo del formulario
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.5.4.4. Apariencia

Esta ventana de definición del formulario será una ventana emergente de forma que se pueda visualizar de una forma más destacada el formulario junto con sus campos.

4.3.6. Ventanas de Plantilla de Formularios

4.3.6.1. Creación de la Plantilla de Formulario

4.3.6.1.1. Propósito

Definir las propiedades básicas de una plantilla de formulario como puede ser el nombre, el título y la descripción. Se puede indicar la función del formulario mediante la descripción.

4.3.6.1.2. Objetos y Acciones

Como hemos comentado anteriormente, los objetos principales que aparecerán serán el nombre, título y descripción de la plantilla. Estos objetos serán 3 campos que el usuario tendrá que rellenar antes de continuar.

4.3.6.1.3. Objetos Compuestos

En esta ocasión cada objeto se compondrá de dos elementos:

- **Nombre**: Nombre o identificar del campo. Sirve para informar al usuario de qué tipo de campo se trata.
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.6.1.4. Apariencia

Esta ventana de definición de la plantilla del formulario será una ventana emergente de forma que se mostrará de forma principal frente a los demás componentes de la aplicación. En la parte inferior de la ventana emergente se mostrarán dos opciones de forma que el usuario pueda cancelar o aceptar la acción.

4.3.6.2. Construcción de la Plantilla de Formulario

4.3.6.2.1. Propósito

Definir y construir la estructura del formulario de forma que establezcamos una plantilla de formulario.

4.3.6.2.2. Objetos y Acciones

En la barra lateral izquierda existirá una *barra de elementos* (campos de formularios) que podemos insertar. Cada elemento será un campo de formulario correctamente especificado agrupado por funcionalidad. En la parte inferior existirán una serie de acciones como pueden ser cerrar, testar y publicar.

4.3.6.2.3. Objetos Compuestos

En esta ocasión cada objeto acción se compondrá de dos elementos:

- **Icono**: imagen asociada al elemento (acción del formulario) de forma que no dé lugar a la confusión.
- **Nombre**: Se trata del nombre de la acción a realizar.

En cuanto a los campos del formulario:

- **Nombre**: Se trata del nombre del campo del formulario
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción del campo en cuestión

4.3.6.2.4. Apariencia

La parte principal y más importante de esta ventana es la zona de construcción y por lo tanto de edición de la estructura del formulario y de sus campos. Este tipo de aplicación WYSIWYG hace que el usuario reciba un feedback instantáneo de forma que el usuario pueda visualizar el formulario con su formato final.

4.3.6.3. Ventana de Validación de Formulario

4.3.6.3.1. Propósito

El propósito principal es mostrar las opciones de validación del formulario.

4.3.6.3.2. Objetos y Acciones

En la parte inferior del formulario existirán dos botones mediante los cuales el usuario podrá aceptar o cancelar la acción.

Cada opción de validación será identificada mediante su nombre y su valor si estuviera rellenado, aparte de una pequeña descripción.

4.3.6.3.3. Objetos Compuestos

En esta ocasión cada objeto acción se compondrá de dos elementos:

- **Nombre**: Se trata del nombre del campo del formulario
- **Campo**: Se trata de un campo vacío de forma que el usuario pueda escribir sobre él.
- **Descripción del campo**: Se trata de una corta descripción de la opción de validación

4.3.6.3.4. Apariencia

Esta ventana será una ventana emergente de forma que se pueda visualizar de una forma más destacada y ordenada las opciones de validación.

4.3.7. Ventanas de Exploración

4.3.7.1. Exportar Formulario, Archivo Adjunto

4.3.7.1.1. Propósito

Permitir al usuario adjuntar un archivo de cualquier tipo aunque los más comunes sean imágenes, archivos de texto o PDF.

4.3.7.1.2. Objetos y Acciones

La ventana de exploración contiene una serie de elementos como pueden ser los siguientes:

- Un árbol de directorios para escoger la ruta donde se encuentra el modelo.
- Un cuadro de texto para escribir el nombre del archivo a cargar.
- Un botón Aceptar para adjuntar el documento
- Un botón Cancelar para detener la operación y cerrar la ventana, no cargándose el archivo.

4.3.7.1.3. Apariencia

Las opciones se ofrecerán en una ventana emergente estándar de Windows. Estas ventanas emergentes suelen poder moverse, aunque no ocultarse ni redimensionarse. El aspecto viene determinado por la interfaz gráfica del sistema operativo donde se esté ejecutando la aplicación.

4.3.8. Estándares Gráficos

4.3.8.1. Apariencia de las Ventanas

La apariencia de las ventanas vendrá determinada por las hojas de estilo o CSS, **(Cascading Style Sheets**). CSS es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas.

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que le aplicaremos a:

- Una web entera, de modo que se puede definir la forma de toda la web de una sola vez.
- Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, aplicando estilos visibles en un trozo de la página.
- Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en nuestra programación. Podemos definir, por ejemplo, varios tipos de párrafos: en rojo, en azul, con márgenes, sin ellos...

4.3.9. Jerarquías de Navegación

A continuación se representan las ventanas existentes en el sistema, de forma que algunas serán predeterminadas al navegador:

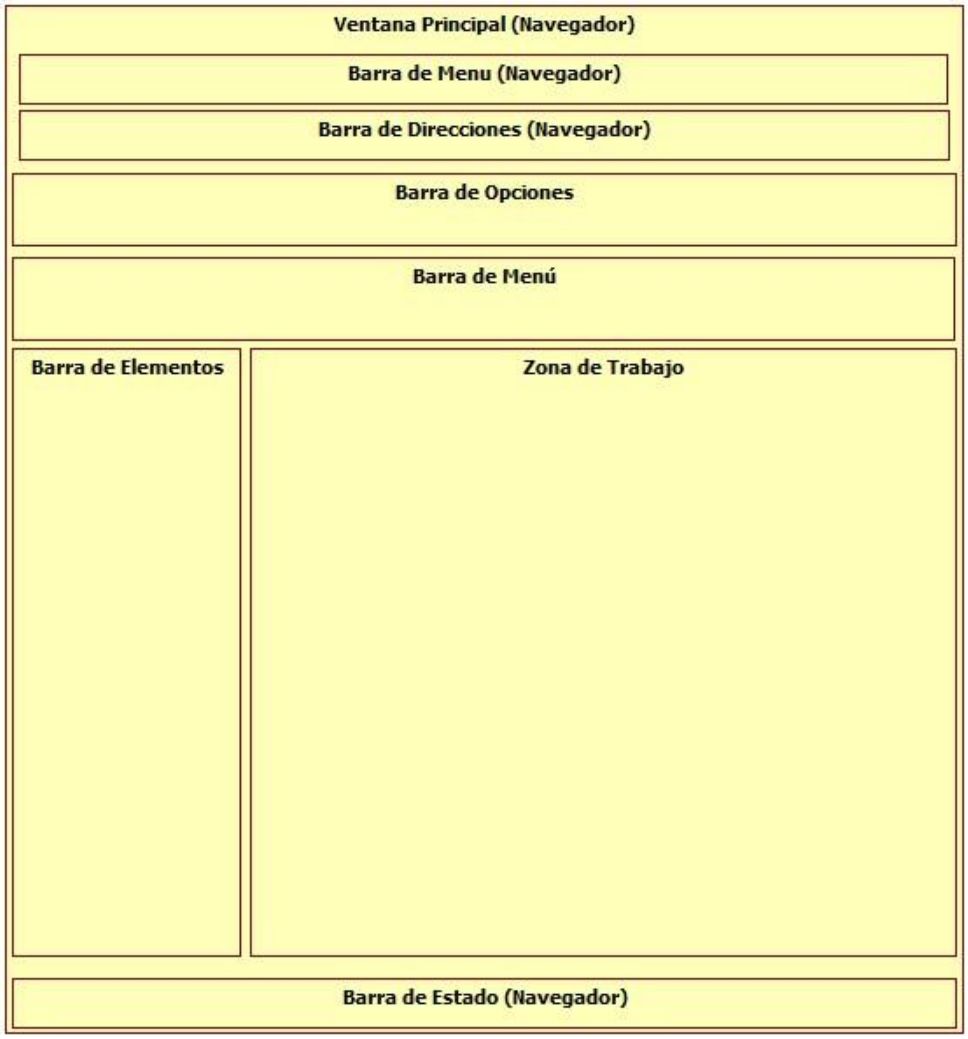

Ilustración 4.28 - Jerarquía de Navegación

Como es evidente, no todas las pantallas de nuestra aplicación seguirán esta estructura. Hay algunas en las que la *Barra de Elementos* no existe, lo que significa que la *Zona de Trabajo* abarcará toda la zona principal de nuestra aplicación.

4.4. REALIZACIÓN DE CASOS DE USO

4.4.1. Introducción

4.4.1.1. Propósito

En este apartado describimos el comportamiento entre el usuario y el sistema a través de diagramas de secuencia. Para ello primeramente realizaremos una especificación formal de los casos de uso principales de diseño.

4.4.1.2. Alcance

Representamos cómo el usuario interactúa con el sistema y así poder definir completamente los elementos que componen los paquetes y los subsistemas correspondientes. Esta interacción se representará mediante unos casos de uso que tendrán una descripción tanto textual como gráfica.

4.4.1.3. Estructura del apartado

Diagrama de Casos de Uso	Se trata de una descripción de los casos de uso principales de diseño.
Especificación de Casos de Uso	Contendrá una descripción textual (especificación formal de diseño) y una descripción gráfica (diagrama de secuencia de diseño)

Tabla 4.4 - Estructura del apartado

4.4.2. Estructura General de los Casos de Uso en Diseño

A continuación se muestra un diagrama de los casos de uso principales de nuestro sistema:

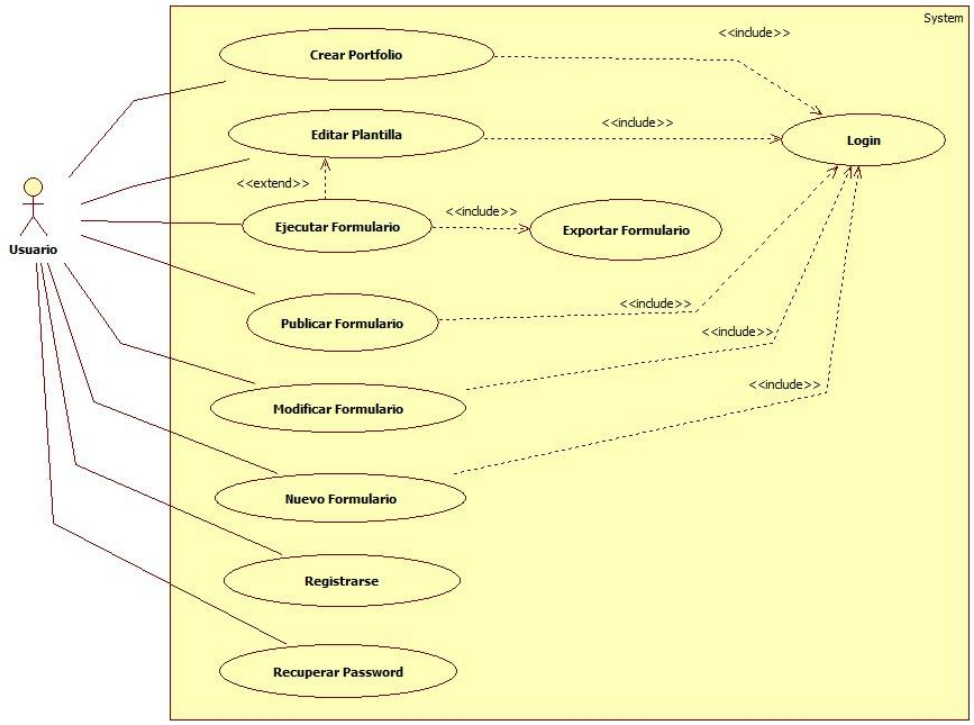

Ilustración 4.29 - Diagrama de Casos de Uso (Diseño)

4.4.3. <UC-001> Crear Portfolio

4.4.3.1. Flujo de Eventos – Diseño

UC-001	**Crear Portfolio**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea crear un nuevo portfolio.	
Precondición	El sistema se ha iniciado correctamente El usuario está identificado y registrado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario hace click en el hipervínculo "*Nuevo Portfolio*" en la barra de menú de la aplicación.
	2	El sistema muestra una ventana emergente mediante el cual solicita al usuario que introduzca los datos principales de la nueva plantilla de formulario. El sistema muestra los siguientes campos: "*Nombre*", "*Título*" y "*Descripción*".
	3	El usuario introduce los datos principales del nuevo portfolio y hace click en el hipervínculo "ok".
	4	El sistema valida los datos introducidos, indicando al usuario si cada campo ha sido rellenado correctamente. En caso de que la información no sea correcta, se mostrará un mensaje de error que el usuario podrá visualizar en la parte inferior del campo rellenado.
	5	El sistema muestra la plantilla en un canvas con la estructura inicial de la plantilla y una serie de opciones de edición en la barra de elementos. El sistema crea el fichero XML que relaciona con la plantilla.
	6	El usuario selecciona una serie de opciones disponibles en la barra de herramientas (campos de formulario) de forma que haciendo click en uno de los campos, se añadirá a la plantilla mostrada en el canvas.
	7	El sistema cierra el fichero XML correspondiente a la plantilla y el caso de uso finaliza.
Postcondición	La estructura de la plantilla seleccionada ha sido creada y guardada en el documento XML correspondiente. Ha sido creado un nuevo portfolio y almacenado en la base de datos.	
Excepciones	Paso	Acción
	3	Si no se ha seleccionado ningún elemento, se le informa al usuario mediante una ventana emergente.
	6	Si el usuario decide ejecutar el formulario se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso continúa.
Comentarios		

4.4.3.2. Diagramas de Interacción

4.4.3.2.1. Diagramas de Secuencia

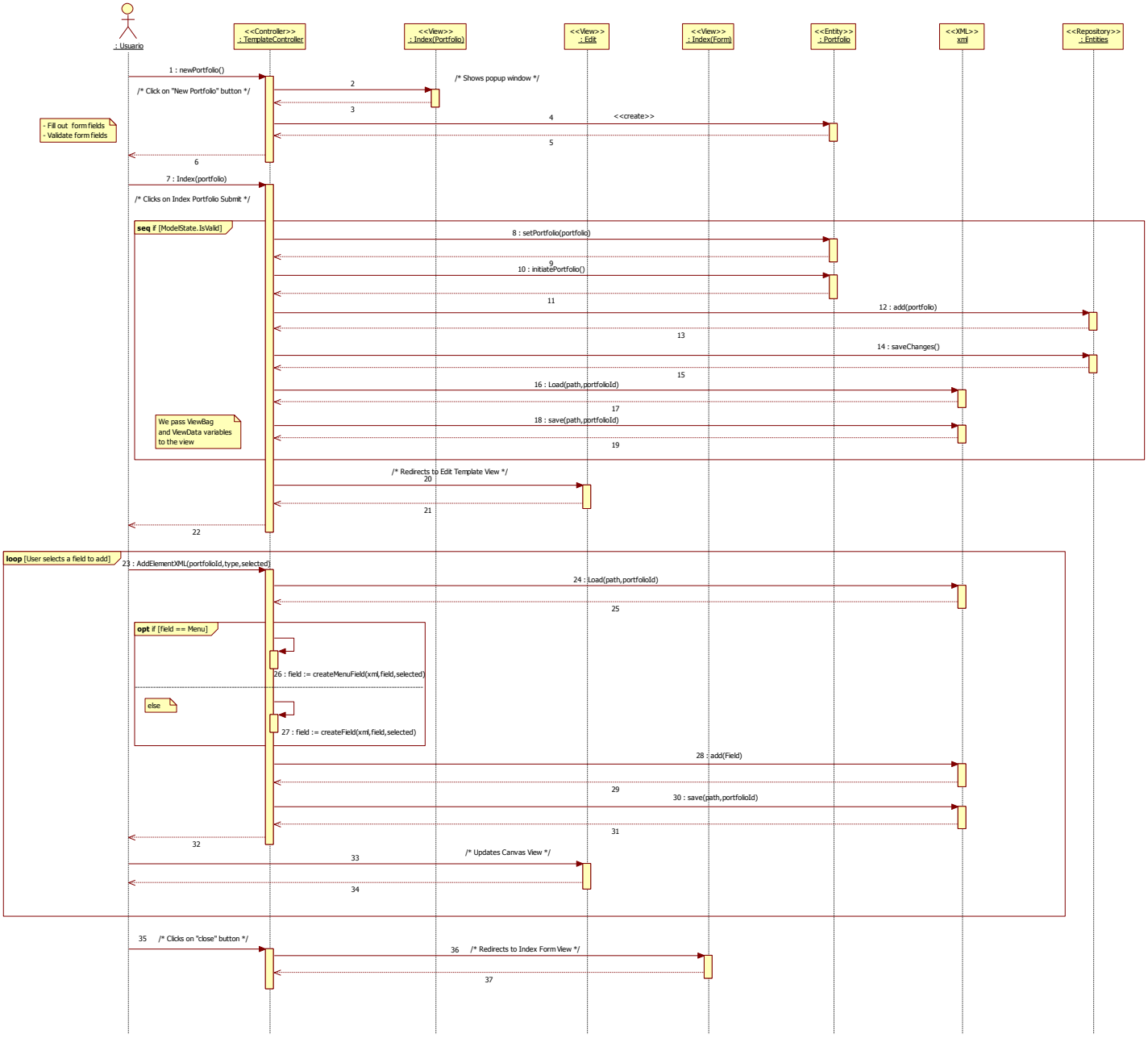

Ilustración 4.30 - <UC-001> Crear Portfolio

4.4.4. <UC-002> Editar Plantilla

4.4.4.1. Flujo de Eventos – Diseño

UC-002	**Editar Plantilla**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea editar una plantilla formulario.	
Precondición	El sistema se ha iniciado correctamente El usuario está registrado e identificado en el sistema La plantilla del formulario existe en el sistema como documento XML	
Secuencia normal	Paso	Acción
	1	El usuario hace click en el hipervínculo "*Editar Plantilla*" en la barra de menú de la aplicación.
	2	El sistema muestra la plantilla en un canvas con la estructura que el usuario ha creado y una serie de opciones de edición en la barra de elementos. El sistema abre el fichero XML que contiene la información sobre la plantilla del formulario.
	3	El usuario selecciona una serie de opciones disponibles en la barra de herramientas (campos de formulario) de forma que haciendo click en uno de los campos, se añadirá a la plantilla mostrada en el canvas.
	4	El sistema cierra el fichero XML correspondiente a la plantilla y el caso de uso finaliza.
Postcondición	La estructura de la plantilla seleccionada ha sido modificada y guardada en el documento XML correspondiente.	
Excepciones	Paso	Acción
	3	Si no se ha seleccionado ningún elemento, se le informa al usuario mediante una ventana emergente.
	4	Si el usuario decide hacer click en el hipervínculo "Testar", se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso finaliza.
Comentarios	Los elementos (campos de formulario) del canvas de la plantilla de formulario son seleccionables de forma que se añadirán los campos del formulario teniendo como referencia el elemento seleccionado.	

4.4.4.2. Diagramas de Interacción

4.4.4.2.1. Diagramas de Secuencia

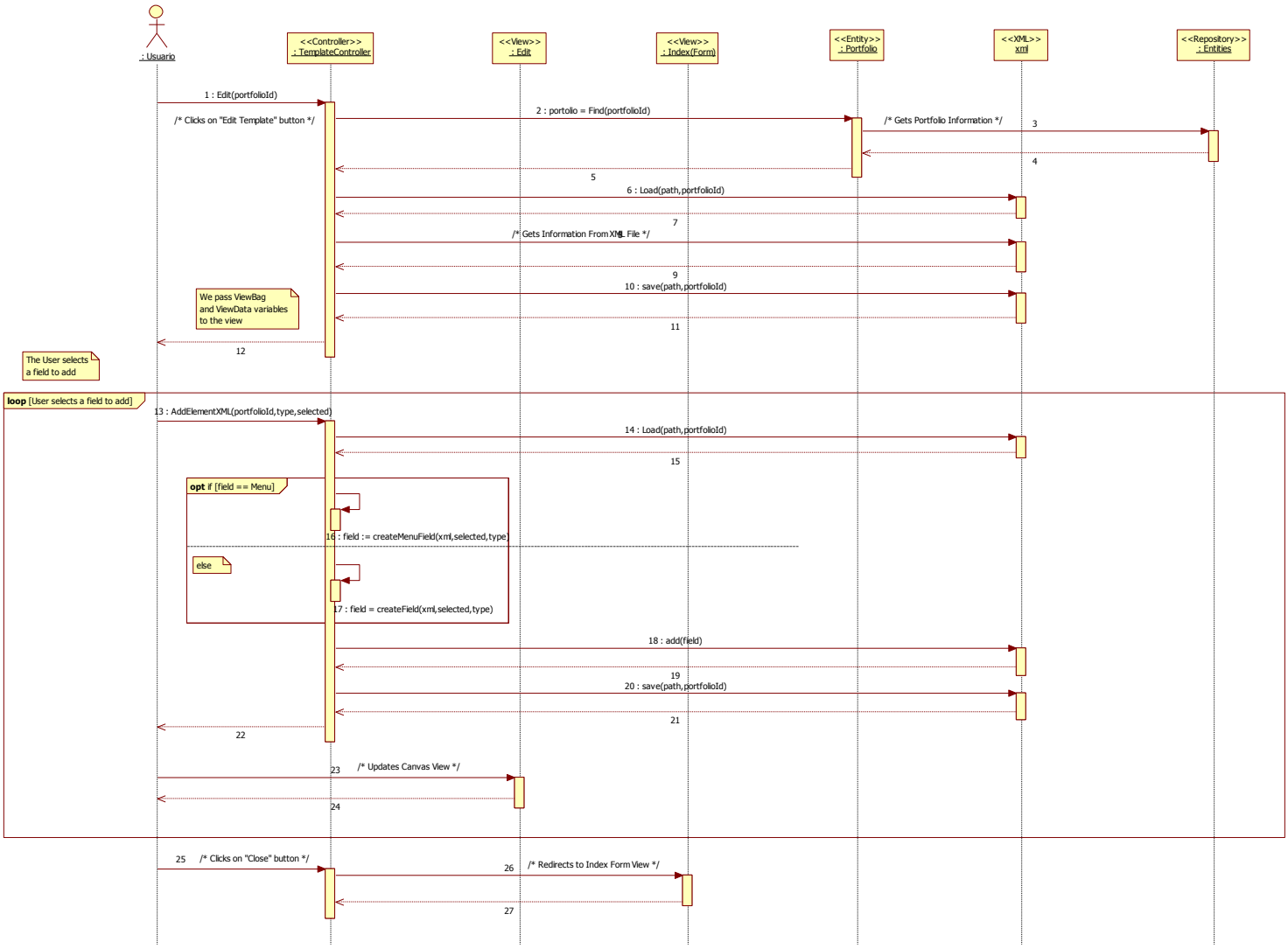

Ilustración 4.31 - <UC-002> Editar Plantilla

4.4.5. <UC-003> Ejecutar Formulario

4.4.5.1. Flujo de Eventos – Diseño

UC-003	**Ejecutar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea ejecutar un formulario.	
Precondición	El sistema se ha iniciado correctamente El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario hace click en el hipervínculo "*Testar*" en la barra de opciones disponible.
	2	El sistema abre el fichero XML correspondiente al formulario y muestra la información contenida creada por el usuario, mediante una ventana emergente.
	3	El usuario visualiza la información referente al formulario obtenida del fichero XML correspondiente y hace click en el hipervínculo "ok".
	4	El sistema cierra el diálogo donde se muestra el formulario y el caso de uso finaliza.
Postcondición	La información del formulario ha sido mostrada al usuario a través de una ventana emergente.	
Excepciones	Paso	Acción
	3	Si el usuario decide hacer click en el hipervínculo "PDF" se inicia el caso de uso *Exportar Formulario UC-007*, a continuación, este caso de uso continúa.
Comentarios	Los elementos (campos de formulario) del canvas de la plantilla de formulario son seleccionables de forma que se añadirán los campos del formulario teniendo como referencia el elemento seleccionado.	

4.4.5.2. Diagramas de Interacción

4.4.5.2.1. Diagramas de Secuencia

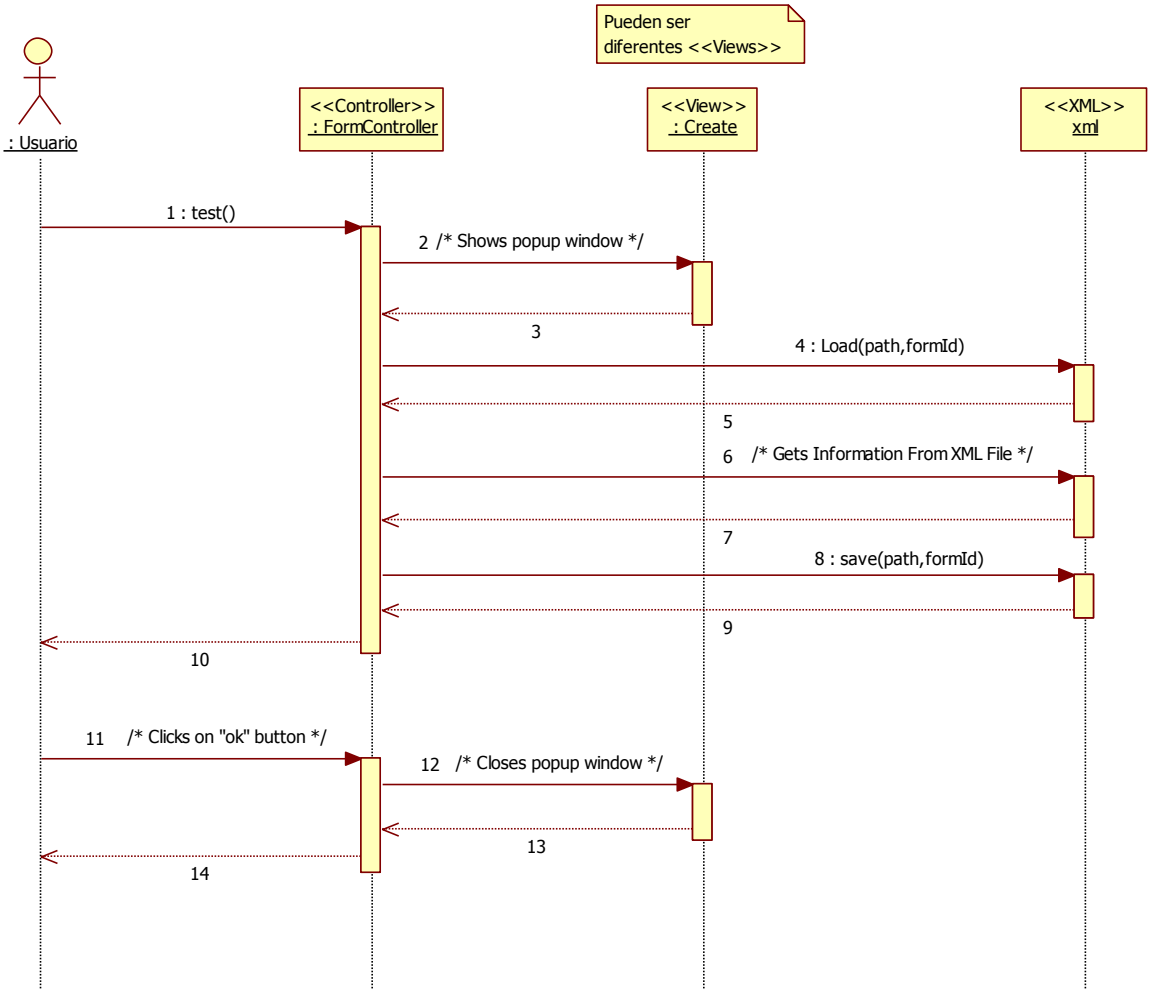

Ilustración 4.32 - <UC-003> Ejecutar Formulario

4.4.6. <UC-004> Publicar Formulario

4.4.6.1. Flujo de Eventos – Diseño

UC-004	**Publicar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea publicar un formulario.	
Precondición	El sistema se ha iniciado correctamente El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario hace click en el hipervínculo "*Publicar*" en la barra de opciones disponible.
	2	El sistema crea una petición HTTP, asigna el método a utilizar (POST), añade el fichero a enviar (formulario XML) y ejecuta la acción.
	3	El usuario visualiza la información que el sistema le muestra respecto a la publicación del formulario mediante una ventana emergente y hace click en el hipervínculo "*ok*".
	4	El sistema cierra el diálogo donde se muestra la información sobre la publicación y el caso de uso finaliza.
Postcondición	Un mensaje ha sido mostrado al usuario a través de una ventana emergente.	
Comentarios	La dirección del servidor al que se envía el fichero XML es modificable por el usuario. La respuesta que se recibe del servidor puede ser de diferente naturaleza, interpretando que si recibimos *StatusDescription* = "OK", entonces la petición se ha realizado correctamente.	

4.4.6.2. Diagramas de Interacción

4.4.6.2.1. Diagramas de Secuencia

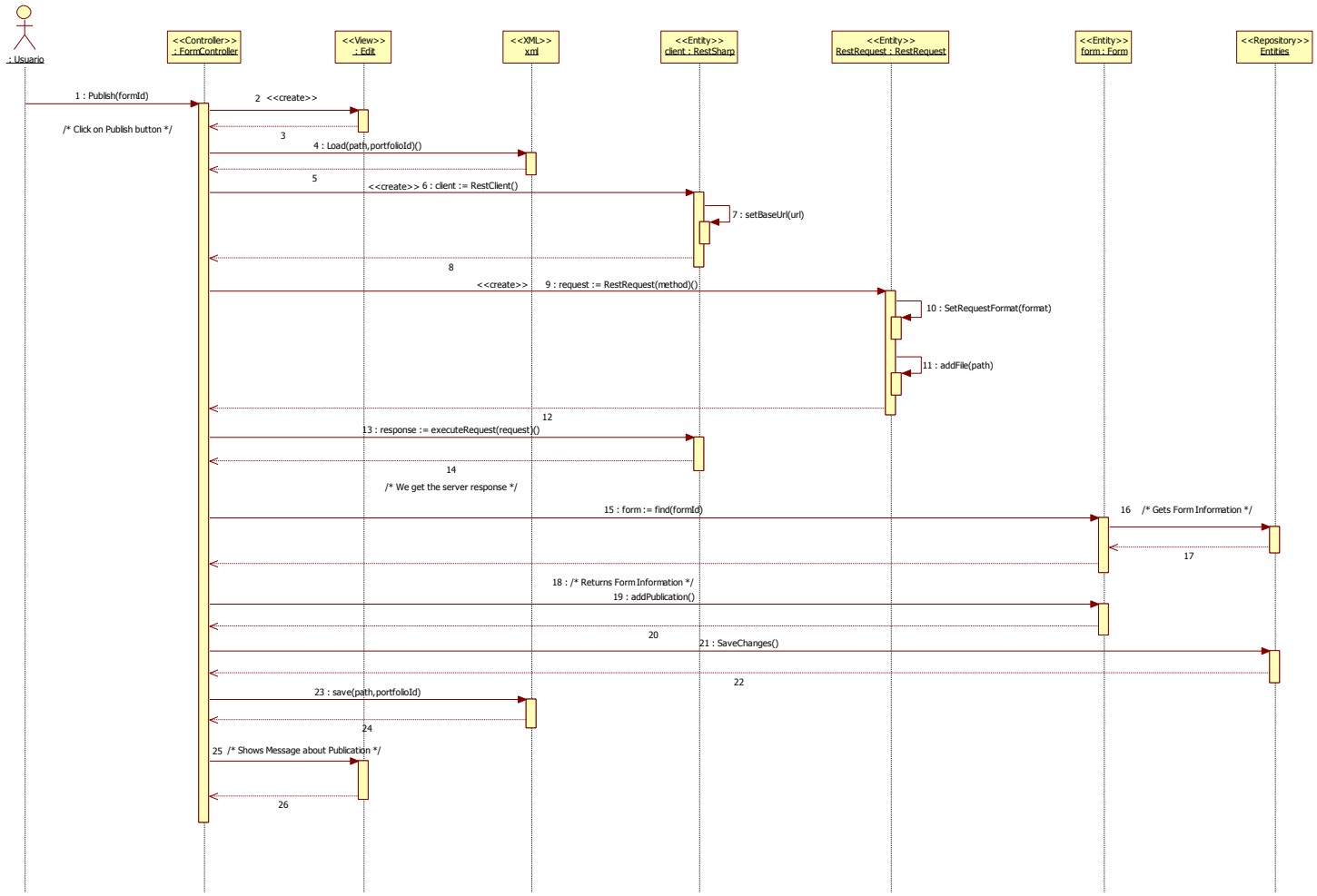

Ilustración 4.33 - <UC-004> Publicar Formulario

4.4.7. <UC-005> Nuevo Formulario

4.4.7.1. Flujo de Eventos – Diseño

UC-005	**Nuevo Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea crear un formulario rellenando la información de los campos.	
Precondición	El sistema se ha iniciado correctamente. El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario hace click en el hipervínculo "*Nuevo*" para la creación de un nuevo formulario.
	2	El sistema muestra los campos correspondientes para la creación de un nuevo formulario.
	3	El usuario rellena la información de los campos del formulario para el nuevo formulario.
	4	El sistema valida los datos introducidos, indicando al usuario si cada campo ha sido rellenado correctamente. En caso de que la información no sea correcta, se mostrará un mensaje de error que el usuario podrá visualizar en la parte inferior del campo rellenado.
	5	El usuario visualiza la información relativa al formulario y hace click en el hipervínculo "*Guardar*".
	6	El sistema guarda la información del formulario introducida por el usuario en un fichero XML y el caso de uso finaliza.
Postcondición	El formulario junto con su información ha sido modificado y guardado en el sistema en el fichero XML.	
Excepciones	Paso	Acción
	6	Si el usuario decide hacer click en el hipervínculo testar el formulario, se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso finaliza.
Comentarios		

4.4.7.2. Diagramas de Interacción

4.4.7.2.1. Diagramas de Secuencia

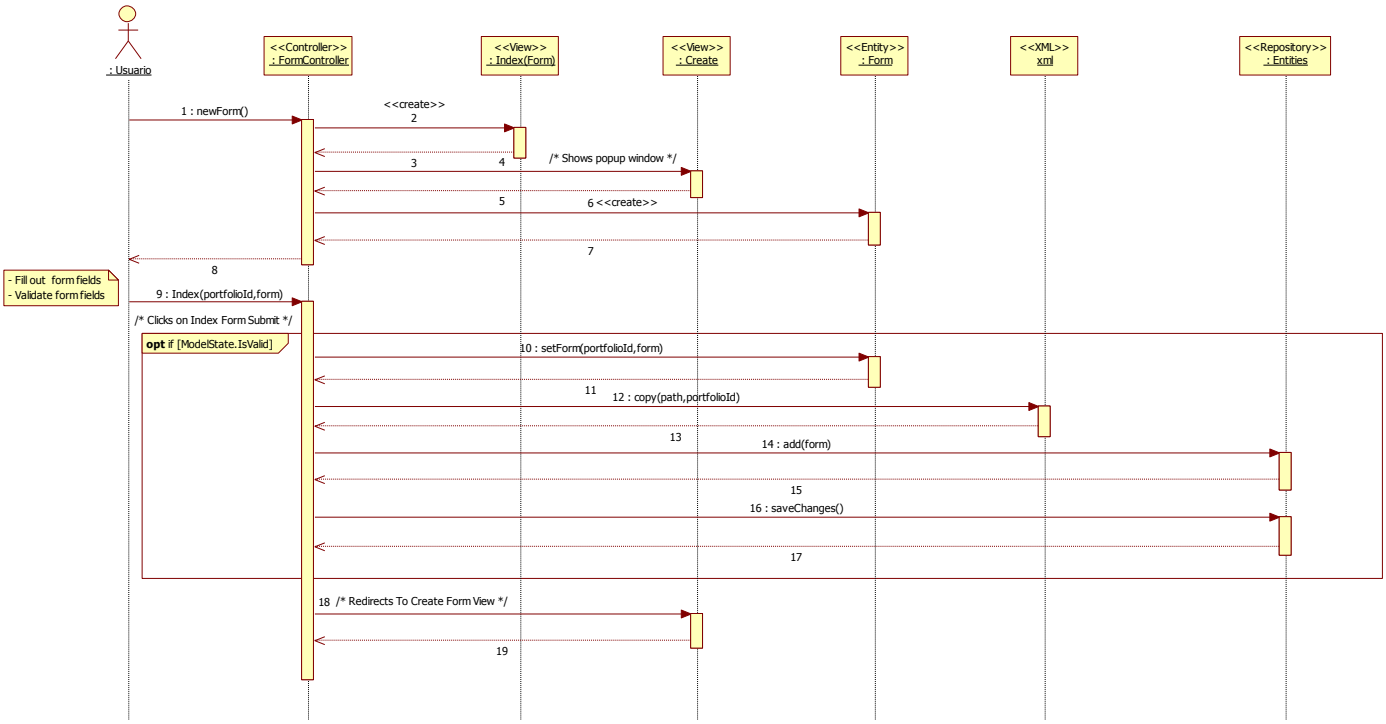

Ilustración 4.34 - <UC-005> Nuevo Formulario

4.4.8. <UC-006> Modificar Formulario

4.4.8.1. Flujo de Eventos – Diseño

UC-006	**Modificar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea modificar la información de los campos del formulario.	
Precondición	El sistema se ha iniciado correctamente. El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El sistema muestra una lista de los formularios pertenecientes al usuario en la pantalla "*Index*" correspondiente a los formularios.
	1	El usuario selecciona el formulario correspondiente, haciendo click sobre uno de ellos.
	2	El sistema muestra el formulario seleccionado por el usuario junto con los campos rellenados con la información correspondiente. El sistema obtiene la información necesaria la obtiene de fichero XML correspondiente al formulario.
	3	El usuario modifica la información de los campos del formulario.
	4	El sistema valida los datos introducidos, indicando al usuario si cada campo ha sido rellenado correctamente. En caso de que la información no sea correcta, se mostrará un mensaje de error que el usuario podrá visualizar en la parte inferior del campo rellenado.
	5	El usuario visualiza la información relativa al formulario y hace click en el hipervínculo "*Guardar*".
	6	El sistema guarda la información del formulario introducida por el usuario en el fichero XML y el caso de uso finaliza.
Postcondición	El formulario junto con su información ha sido modificado y guardado en el sistema en el fichero XML.	
Excepciones	Paso	Acción
	6	Si el usuario decide hacer click en el hipervínculo testar el formulario, se inicia el caso de uso *Ejecutar Formulario UC-003*, a continuación, este caso de uso finaliza.
Comentarios		

4.4.8.2. Diagramas de Interacción

4.4.8.2.1. Diagramas de Secuencia

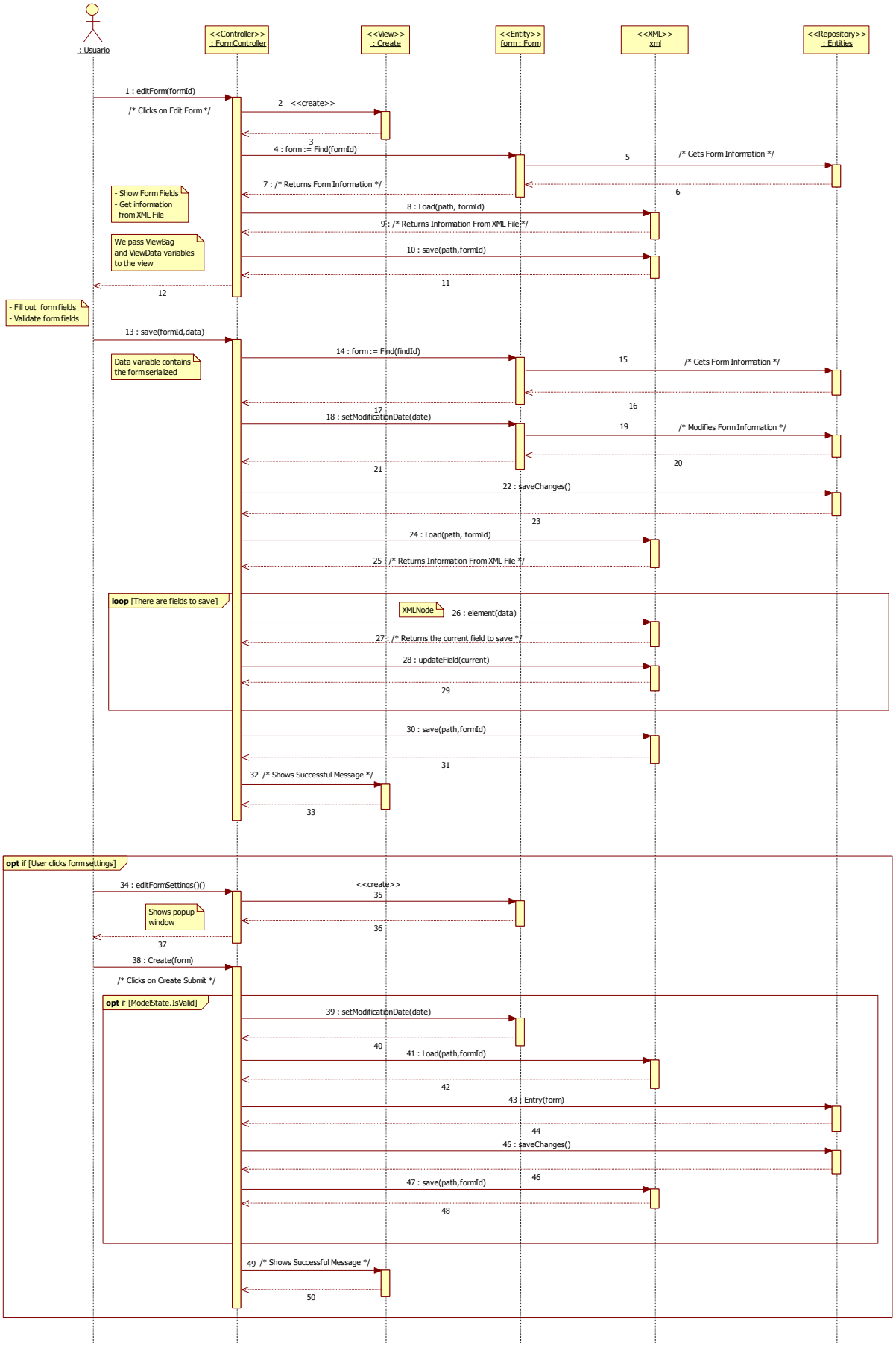

Ilustración 4.35 - <UC-006> Modificar Formulario

4.4.9. <UC-007> Exportar Formulario

4.4.9.1. Flujo de Eventos – Diseño

UC-007	**Exportar Formulario**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario desea exportar el formulario.	
Precondición	El sistema se ha iniciado correctamente. El usuario está registrado e identificado en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la opción exportar formulario haciendo click en el hipervínculo "*PDF*".
	2	El sistema muestra una ventana de exploración de forma que usuario pueda seleccionar tanto el nombre del fichero a guardar como la ruta dónde desea guardarlo.
	3	El usuario hace click en el hipervínculo guardar.
	4	El sistema guarda un documento PDF que contiene los datos del formulario y el caso de uso finaliza.
Postcondición	El formulario ha sido exportado como un nuevo fichero con formato específico.	
Excepciones	Paso	Acción
	1	Si el usuario hace click en cancelar la ventana emergente se cierra y por lo tanto el caso de uso queda sin efecto.
Comentarios		

4.4.9.2. Diagramas de Interacción

4.4.9.2.1. Diagramas de Secuencia

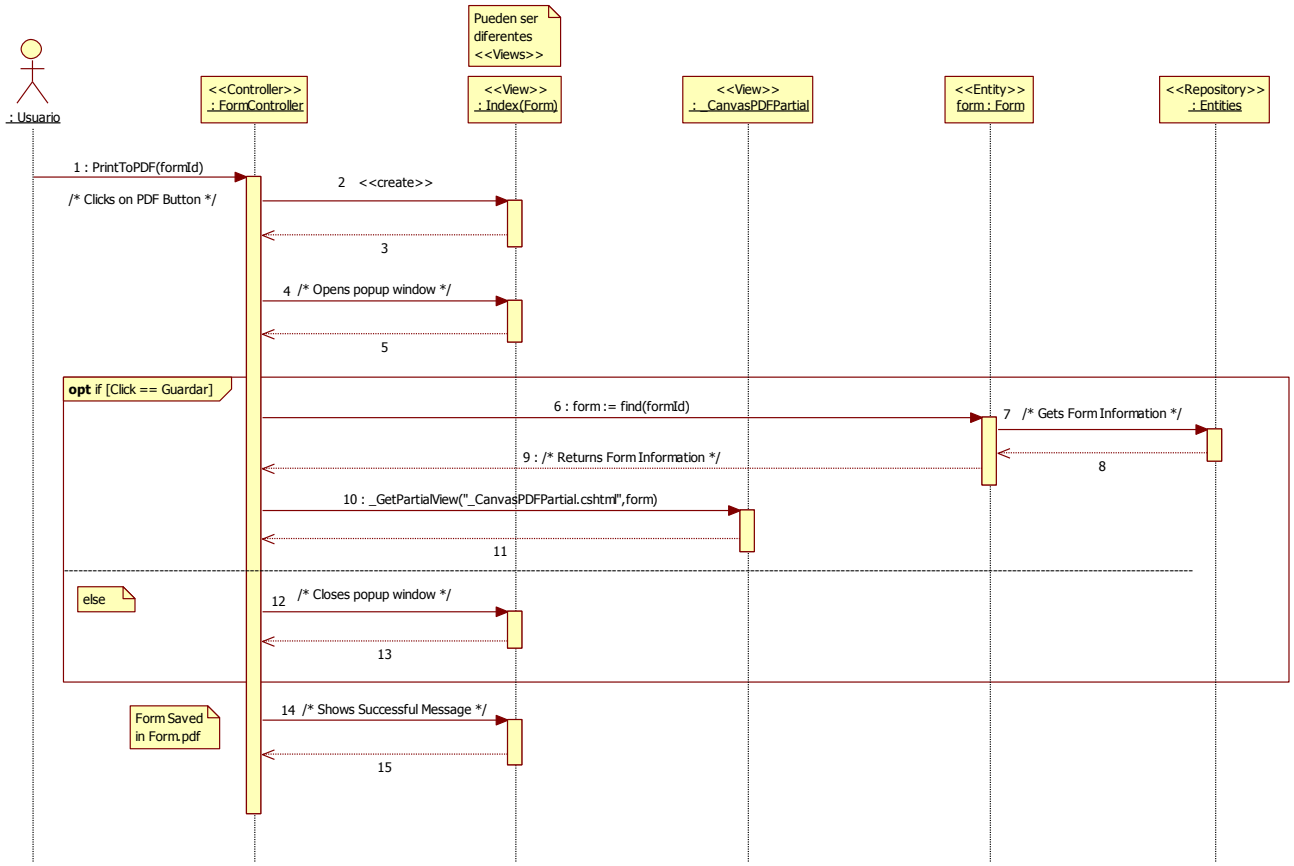

Ilustración 4.36 - <UC-007> Exportar Formulario

4.4.10. <UC-008> Login

4.4.10.1.Flujo de Eventos – Diseño

UC-008	**Login**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando cualquier usuario desee identificarse en el sistema.	
Precondición	El sistema se ha iniciado correctamente. El usuario está registrado en el sistema. Existe una tabla "*Users*" en la base de datos del sistema de forma que contenga el usuario que desea loguearse.	
Secuencia normal	Paso	Acción
	1	El sistema muestra un formulario para loguearse en la pantalla inicial de la aplicación. Dicho formulario contiene dos campos: Nombre de usuario y Password
	2	El usuario introduce sus datos personales y hace click en el hipervínculo "*Login*" de la pantalla inicial de la aplicación "*Index*".
	3	El sistema verifica los datos personales que el usuario ha introducido indicando en su caso en cada campo si el usuario ha introducido la información correctamente.
	4	El sistema muestra las funciones y datos disponibles relativos a los portfolios del usuario y el caso de uso finaliza.
Postcondición	El usuario ha quedado identificado en el sistema.	
Excepciones	Paso	Acción
	1	Si el usuario hace click en el hipervínculo "Registrarse", el sistema redirecciona a la pantalla "Register" de nuestra aplicación de forma que el usuario pueda introducir los datos para completar el registro. Se inicia el caso de uso *Registrarse UC-009*
	1	Si el usuario hace click en el hipervínculo "¿Ha olvidado la contraseña?", el sistema redirecciona a la pantalla "Recuperar Password" de nuestra aplicación de forma que el usuario pueda introducir los datos necesarios para recuperar la contraseña. Se iniciar el caso de uso *Recuperar Password UC-010*
	4	Si el usuario no está registrado, el sistema le notifica la situación de error y el caso de uso queda sin efecto.
Comentarios		

4.4.10.2.Diagramas de Interacción

4.4.10.2.1. Diagramas de Secuencia

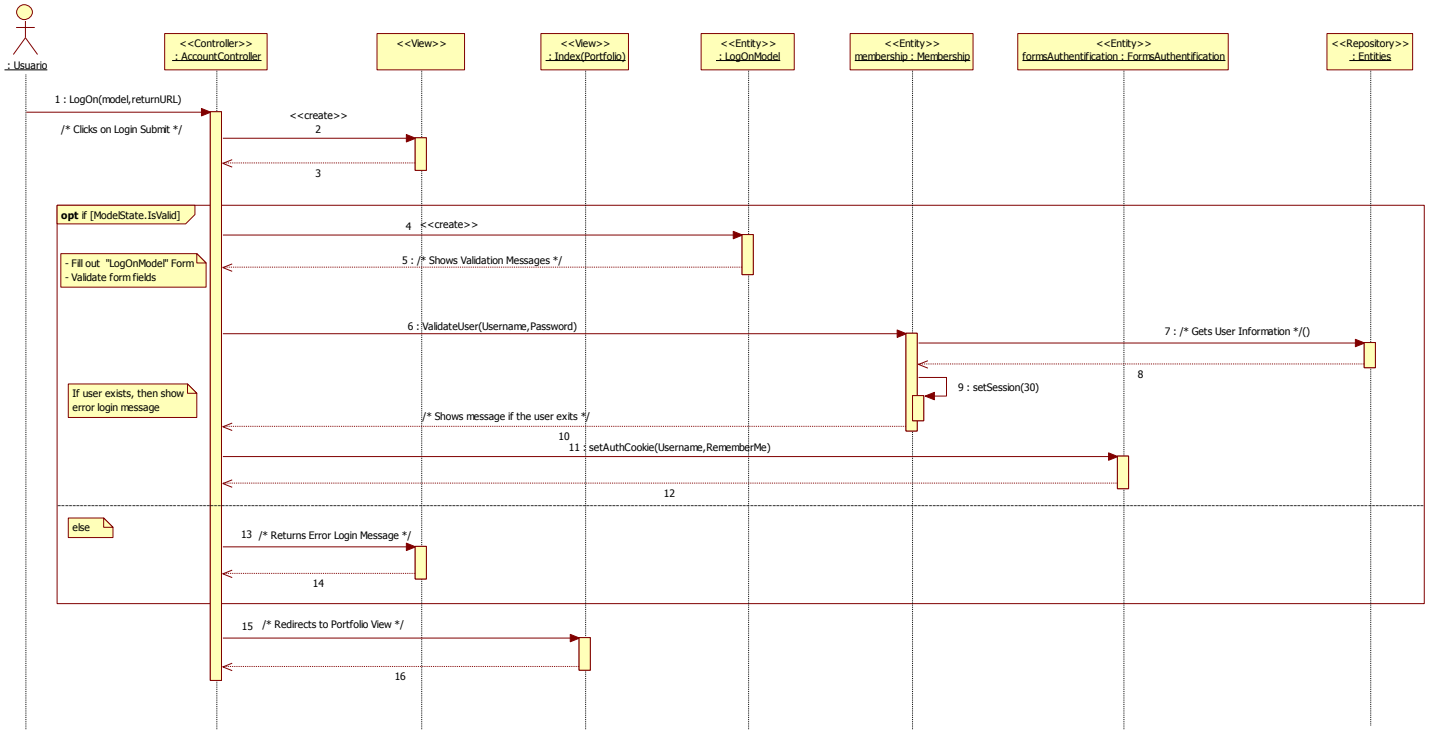

Ilustración 4.37 - <UC-008> Login

4.4.11. <UC-009> Registrarse

4.4.11.1.Flujo de Eventos – Diseño

UC-009	**Registrarse**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando cualquier usuario desee registrarse en el sistema.	
Precondición	El sistema se ha iniciado correctamente.	
Secuencia normal	Paso	Secuencia normal
	1	El usuario hace click sobre el hipervínculo “Registrarse” de la pantalla inicial de la aplicación “*Index*”.
	2	El sistema muestra una serie de campos que son necesarios para el registro.
	3	El usuario rellena todos los campos disponibles con su información personal.
	4	El sistema valida los datos introducidos por el usuario e indica en caso de error los campos que el usuario debe rellenar correctamente.
	5	El sistema registra al usuario mostrando el correspondiente mensaje de información y el caso de uso finaliza.
Postcondición	El usuario ha quedado registrado en el sistema. Se ha insertado una nueva fila con datos sobre el usuario tanto en la tabla “*Users*” como en la tabla “*Memberships*” de la base de datos del sistema.	
Excepciones	Paso	Acción
	4	Si los datos introducidos no son válidos, se informa al usuario de tal circunstancia a través de un mensaje de error y el caso de uso queda sin efecto.
	5	Si el usuario ya existe en el sistema, se informa al usuario de tal circunstancia a través de un mensaje de error y el caso de uso queda sin efecto.
Comentarios		

4.4.11.2.Diagramas de Interacción

4.4.11.2.1. Diagramas de Secuencia

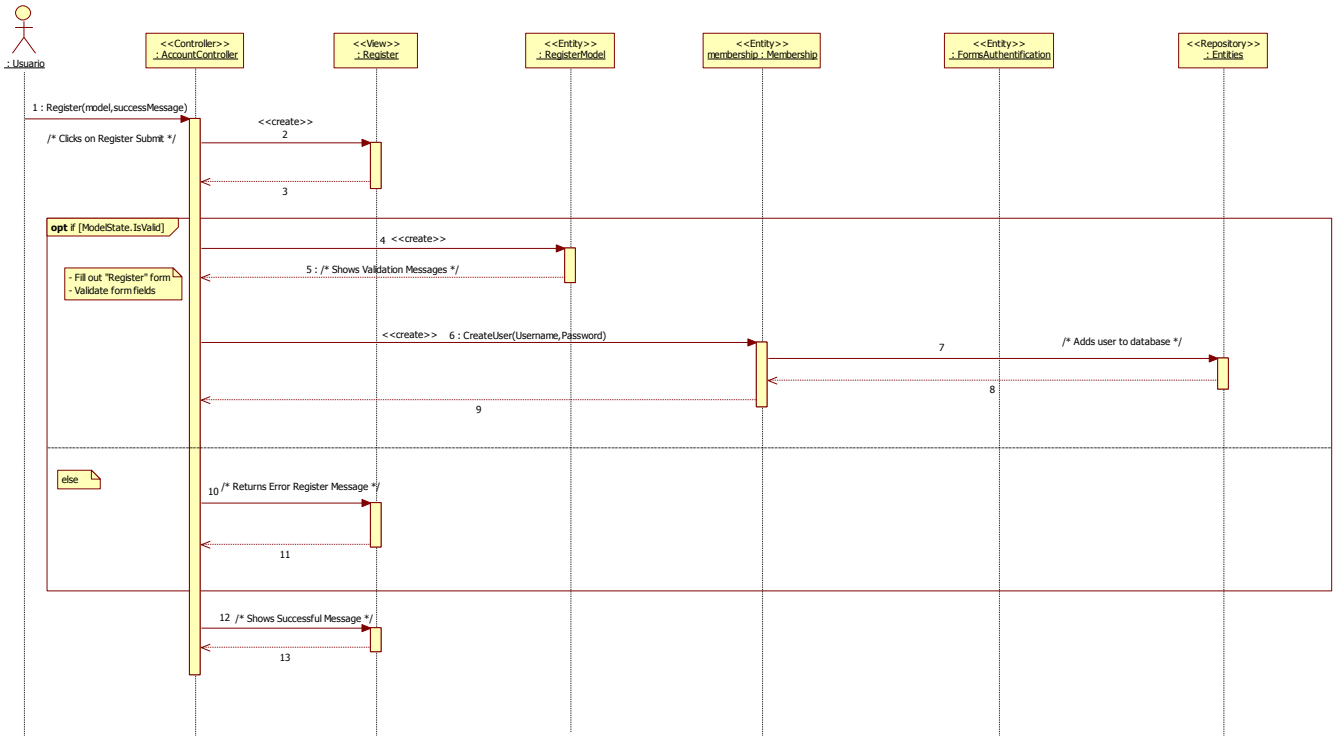

Ilustración 4.38 - <UC-009> Registrarse

4.4.12. <UC-010> Recuperar Password

4.4.12.1.Flujo de Eventos – Diseño

UC-010	**Recuperar Password**	
Dependencias	Usuario	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando cualquier usuario desee registrarse en el sistema.	
Precondición	El sistema se ha iniciado correctamente.	
Secuencia normal	Paso	Acción
	1	El usuario hace click sobre el hipervínculo *"¿Se te ha olvidado la contraseña?*" de la pantalla inicial de la aplicación "*Index*".
	2	El sistema muestra un campo de nombre de usuario.
	3	El usuario introduce su nombre de usuario en el campo correspondiente.
	4	El sistema valida el nombre de usuario e indica en caso de error el campo que debe de rellenar correctamente.
	5	El sistema envía un correo al usuario con la nueva contraseña modificada, muestra un mensaje de información y el caso de uso finaliza.
Postcondición	La contraseña correspondiente al usuario ha sido modificada. Se ha modificado la fila relativa al usuario con la nueva contraseña introducida en la tabla "*Memberships*" de la base de datos del sistema.	
Excepciones	Paso	Acción
	4	Si los datos introducidos no son válidos, se informa al usuario de tal circunstancia a través de un mensaje de error y el caso de uso queda sin efecto.
Comentarios		

4.4.12.2.Diagramas de Interacción

4.4.12.2.1. Diagramas de Secuencia

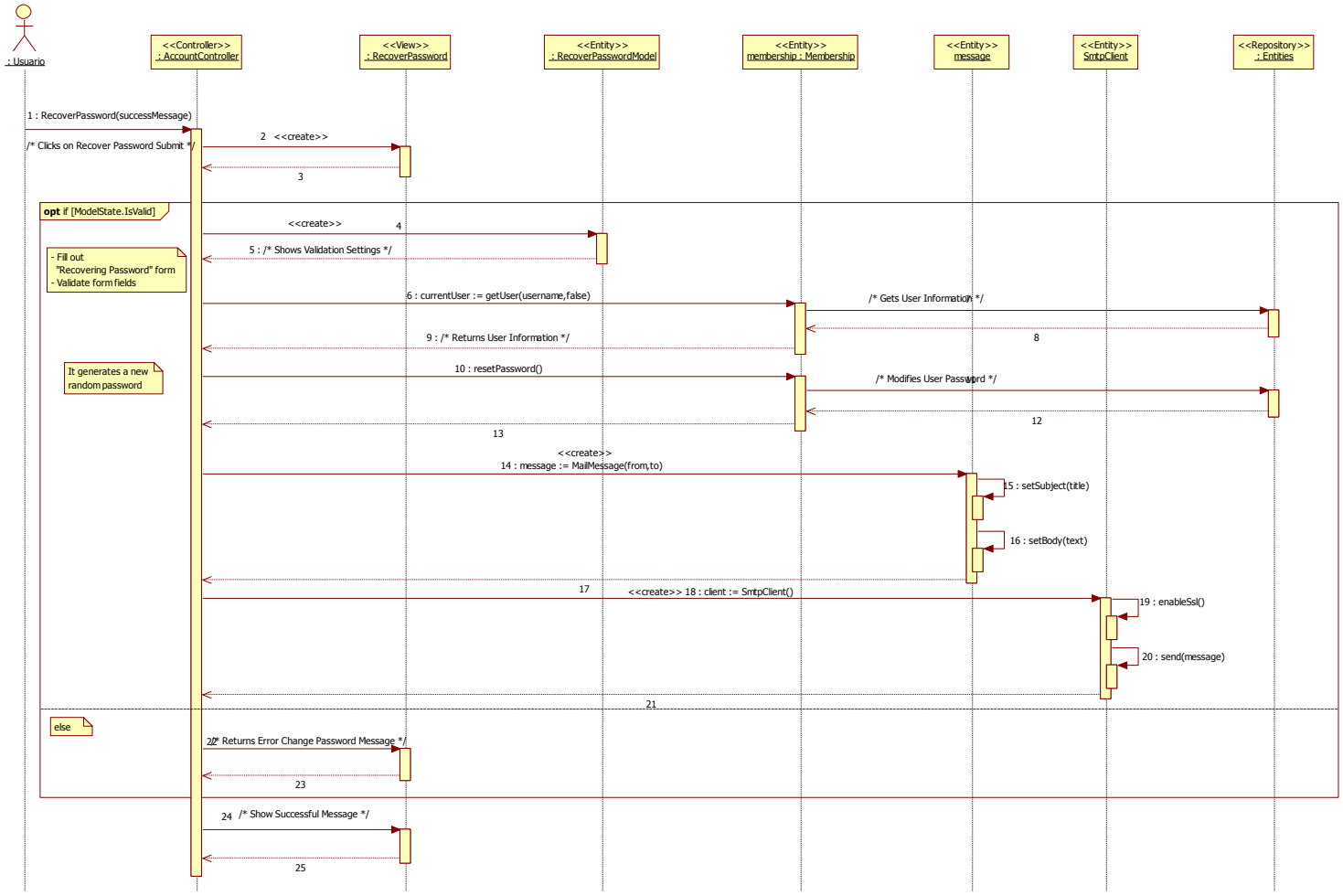

Ilustración 4.39 - <UC-010> Recuperar Password

5. IMPLEMENTACIÓN

5.1. MODELO DE IMPLEMENTACIÓN

5.1.2. Introducción

5.1.2.1. Propósito

El apartado contiene las principales partes del sistema en las cuales se compone. Se describirán detalladamente cada parte funcional del sistema a nivel de construcción.

5.1.2.2. Alcance

Este apartado se ha actualizado a lo largo de las dos iteraciones en la fase de construcción de modo que ha sufrido cambios. La fase de pruebas se apoyará en este modelo una vez terminada dicha fase.

5.1.2.3. Estructura del apartado

Estructura General	Vista general del modelo de implementación junto con los componentes de la aplicación.
Iteración I	Se trata de un modelo que se implementará durante la primera iteración en la fase de construcción. Se incluyen todos los componentes implementados.
Iteración II	Se trata de un modelo que se implementará durante la segunda iteración en la fase de construcción. Se incluyen todos los componentes implementados.

Tabla 5.13 - Estructura del apartado

5.1.3. Vista General del Modelo de Implementación

En este apartado se describen los componentes y cómo el conjunto de ellos forman nuestro sistema. Se iniciará con un modelo de implementación global y poco a poco se irá refinando.

5.1.3.1. Estructura del Modelo de Implementación

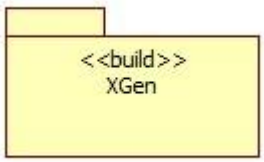

Ilustración 5.1 - Modelo de Implementación Inicial (Nivel 0)

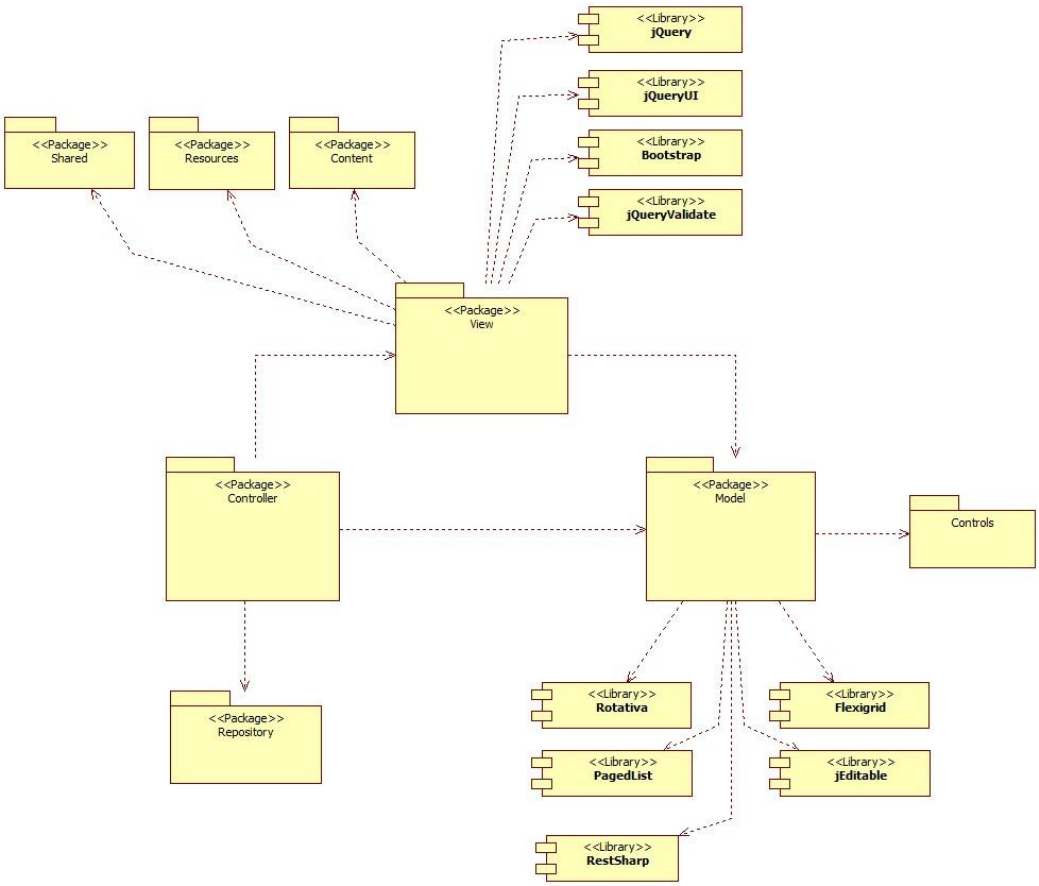

Ilustración 5.2 - Modelo de Implementación (Nivel 1)

5.1.3.2. Paquete Model

El paquete model de implementación representa las clases del sistema, junto con sus atributos y relaciones. Este componente es mostrado en el documento *Arquitectura software* en el apartado *Descomposición del Paquete Model*. Como hemos dicho anteriormente, este elemento muestra las clases principales, sus atributos y las relaciones con otras clases del sistema.

5.1.3.3. Paquete View

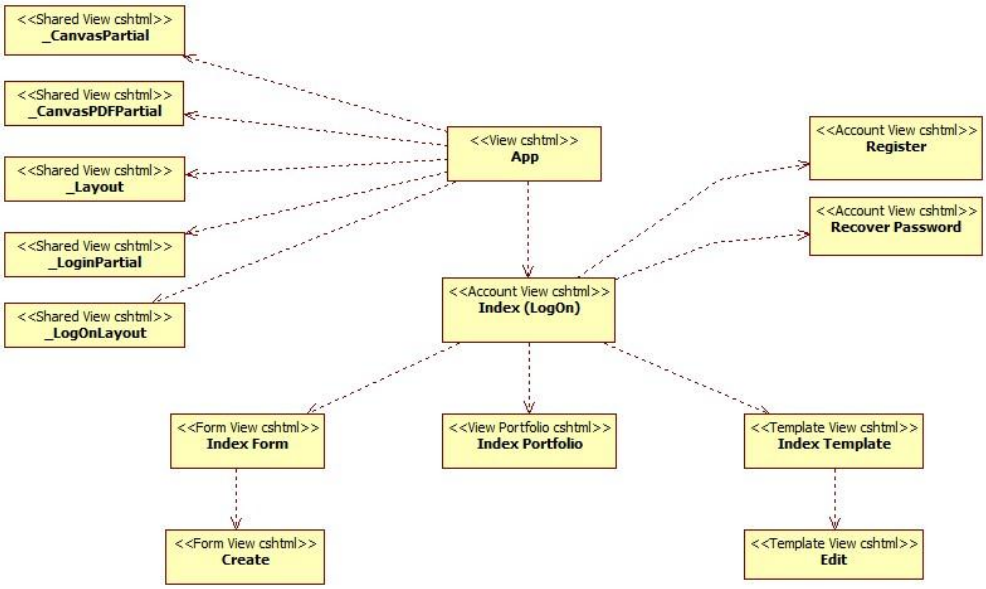

Ilustración 5.3 - Paquete View

5.1.3.4. Paquete Controller

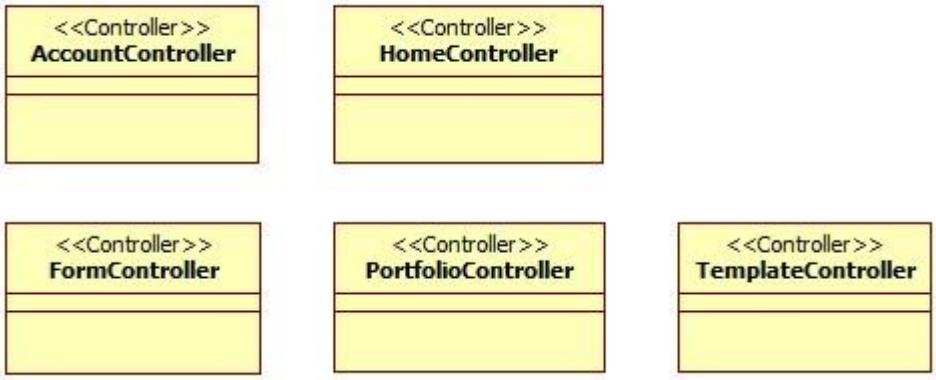

Ilustración 5.4 - Paquete Controller

5.1.3.5. Paquete Resources

Los distintos paquetes representados contiene las traducciones en los idiomas que hemos creído más convenientes (Inglés, Español y Francés) aunque el sistema podría haber sido traducido en todos los idiomas existentes. Quizás en un futuro se pudiera traducir en algún otro idioma, por lo tanto lo dejaremos como una pequeña extensión de XGen.

Hemos decidido agrupar las traducciones del sistema global mediante los siguientes elementos, de forma que se pueda referenciar dicha variable de una forma más específica:

- *Account*: Traducción de las variables relativas a las vistas de creación de la cuenta de usuario (Registrarse), recuperación de password y la pantalla inicial (LogOn).
- *Form*: Traducción de las variables relativas a las vistas de formularios, tanto la vista de formularios de usuario, como a la creación y edición de los mismos.
- *Portfolio*: Traducción de las variables relativas a las vistas de lo portfolios que incluye la vista general de los portfolios una vez que el usuario inicia sesión.
- *Shared*: Traducción de las variables relativas a las vistas de lo portfolios que incluye las vistas de layout, la cabecera de la aplicación y las vistas parciales de la edición de la plantilla de formulario.

- *Template*: Traducción de las variables relativas a las vistas de lo portfolios que incluye la vista de creación y edición de plantilla de formulario.

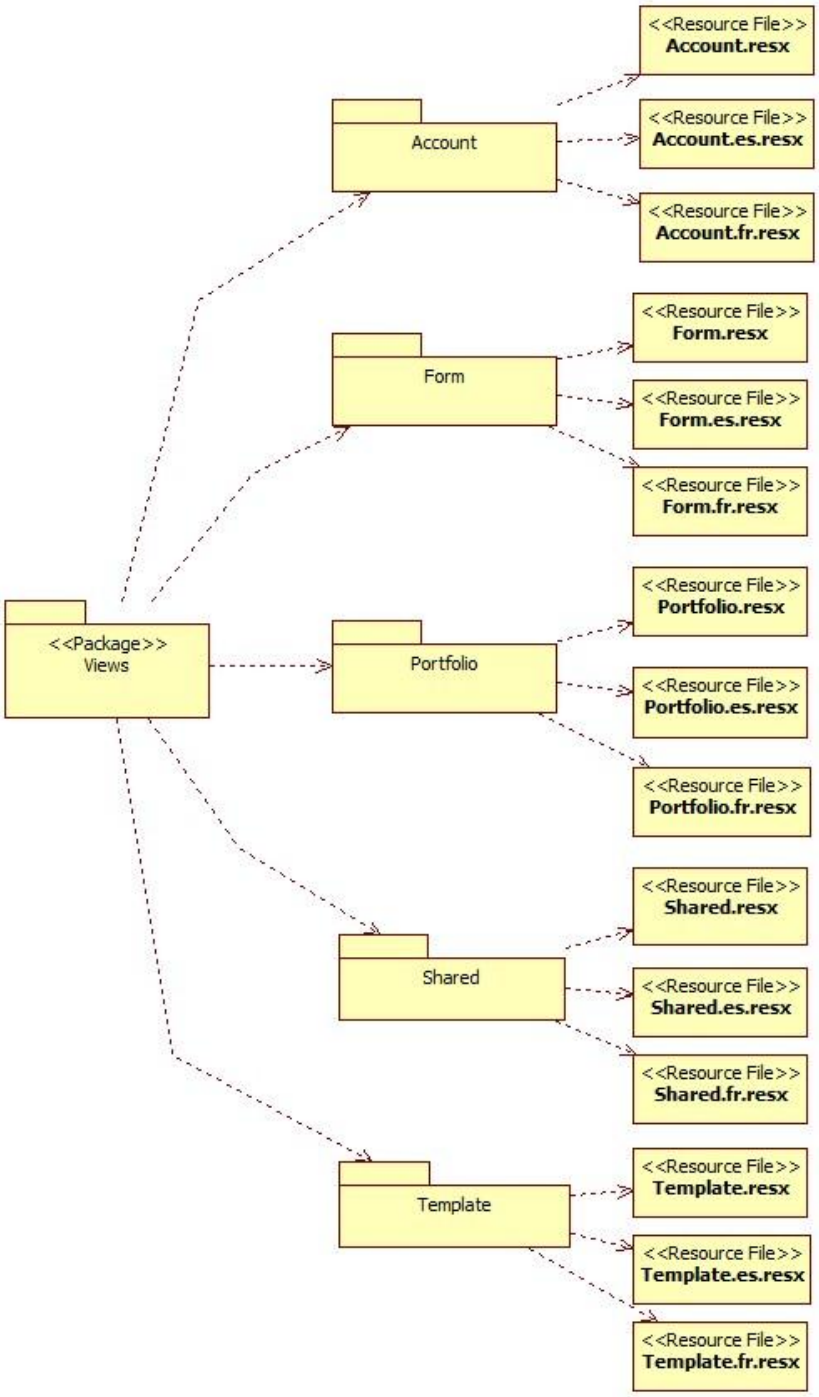

Ilustración 5.5 - Paquete Resources

5.1.3.6. Paquete Content

Los paquetes que componen este elemento son:

- *Flexigrid*: contiene los elementos de interfaz del componente flexigrid utilizado para la gestión de los formularios.
- images: imágenes usadas en la interfaz general del sistema.
- *menuImages*: imágenes usadas como iconos de los elementos del menú de la plantilla de formulario.

- *theme*: contiene elementos del componente jQuery UI
- *uploader*: contiene tanto elementos de interfaz del componente uploader (utilizado para subir imágenes y archivos adjuntos) como elementos almacenados por los usuarios en los formularios.
- *validationImages*: imágenes utilizadas en la interfaz de las opciones de validación.

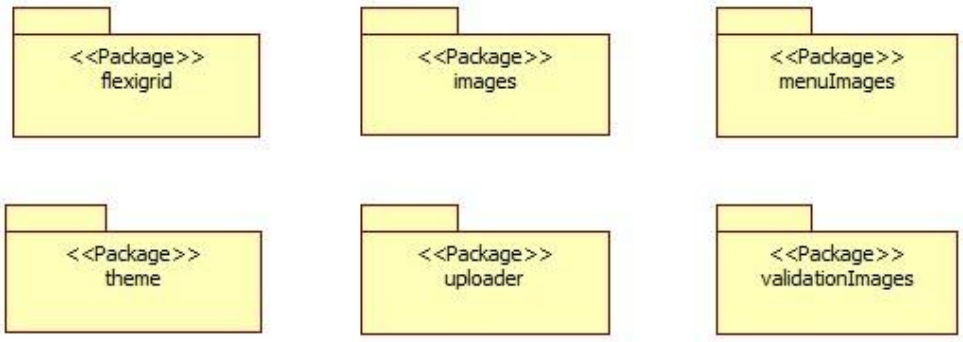

Ilustración 5.6 - Paquete Content

5.1.3.7. Librerías

La aplicación hace uso de varias librerías externas. A continuación describimos brevemente cada una:

- **jQuery:** es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones (FLV) y agregar interacción con la técnica AJAX a páginas web.
- **jQueryValidate:** es un plugin que sirve para la validación de formularios, de forma que nos aporta bastante funcionalidad que podemos cambiar de manera muy sencilla.
- **Bootstrap:** es el framework de Twitter que permite crear interfaces web con CSS y Javascript que adaptan la interfaz dependiendo del tamaño del dispositivo. Los diseños creados con Bootstrap son simples, limpios e intuitivos, esto les da agilidad a la hora de cargar y al adaptarse a otros dispositivos.
- **PagedList:** es un plugin utilizando en nuestro sistema para la paginación de nuestros *portfolios* de forma que el usuario pueda gestionarlos más cómodamente y de una forma más intuitiva y sencilla.
- **Flexigrid:** Se trata de un plugin para jQuery en forma de data-grid para la presentación de datos (en nuestro caso *formularios*). Entre las características más importantes está la de adaptase con datos de tipo XML o JSON usando AJAX para cargar el contenido.
- **jEditable:** Se trata de un plugin para jQuery que habilita al usuario a editar en línea, es decir, editar texto de forma que se actualice instantáneamente. Esto supone una mejor aceptación del usuario frente al sistema.
- **Rotativa:** Esta librería sirve para exportar los formularios a formato PDF. Estos ficheros PDF tendrán una estructura determinada dependiendo del formularios que se desee exportar.
- **RESTSharp:** Es una librería (REST) disponible para plataforma .NET que sirve como cliente HTTP de forma que podemos hacer peticiones a servicios web. Esta librería nos permite modificar información como por ejemplo el tipo de método usado (GET, POST, PUT), añadir cabeceras (header), de forma que podemos indicar el tipo de información enviada, enviar ficheros adjuntos, etc. En nuestro caso en particular enviaremos un fichero XML.

5.1.4. Iteración I

5.1.4.1. Diseño de Base de Datos

Nuestro sistema se apoya sobre una base de datos relacional Microsoft SQL Server y que almacena toda la información relativa a los usuarios, los portfolios y sus formularios. La información relativa a las plantillas de formularios las gestionaremos mediante ficheros XML.

Como es evidente, la base de datos sufrirá variaciones a lo largo de todo la fase de construcción, definiendo nuevos campos o incluso eliminando algunos otros.

Las tablas resultantes del diseño de la base de datos son: *Applications*, *Forms*, *Memberships*, *Portfolios* y *Users*. Existen algunas tablas que nos las aporta el propio framework .NET, como son las tablas *Applications*, *Memberships* y *Users*, es decir, la gestión de usuarios. *Visual Studio* trae consigo ciertas clases por defecto de tal forma que puede realizar la gestión de usuarios por sí mismo.

Entre tablas de nuestro propio sistema se encuentran *Forms* y *Portfolios*, que como su propio nombre indica, almacenan información de formularios y portfolios respectivamente. Nuestra base de datos es fácilmente interpretable, básicamente, los formularios están contenidos en un portfolio (FOREIGN KEY PortfolioId) y han sido creados por un usuario (FOREIGN KEY UserId).

Cada portfolio es creado por un usuario (FOREIGN KEY UserId) y por lo tanto solo puede ser modificado por este.

A continuación comentaremos las tablas de nuestra base de datos junto con los campos de cada una de ellas:

- **ApplicationName:** Nombre de la aplicación
- **ApplicationId:** Identificador de la aplicación (Clave Primaria)
- **Description:** Breve descripción de la aplicación

	Name	Data Type	Allow Nulls	Default
	ApplicationName	nvarchar(235)	☐	
🔑	ApplicationId	uniqueidentifier	☐	
	Description	nvarchar(256)	☑	

Ilustración 5.7 - Applications Table

- **FormId:** Identificador del formulario
- **PortfolioId:** Identificador del portfolio (Clave Foránea) al cual pertenece
- **Name:** Nombre del formulario
- **Title:** Título del formulario
- **Description:** Breve descripción del formulario
- **UserId:** Usuario creador del formulario
- **CreationDate:** Fecha de creación del formulario
- **LastModificationDate:** Fecha de última modificación del formulario
- **Language:** Idioma del formulario (el usuario introduce el idioma del formulario a la hora de la creación)
- **Publications:** Número de publicaciones del formulario.

	Name	Data Type	Allow Nulls	Default
🔑	FormId	uniqueidentifier	☐	
	PortfolioId	uniqueidentifier	☐	
	Name	nvarchar(MAX)	☐	
	Title	nvarchar(MAX)	☐	
	Description	text	☐	
	UserId	uniqueidentifier	☐	
	CreationDate	date	☐	
	LastModificationDate	date	☐	
	Language	nvarchar(MAX)	☐	
	Publications	int	☐	

Ilustración 5.8 - Forms Table

- **ApplicationId:** Identificador de la aplicación
- **UserId:** Identificador de usuario
- **Password:** Password de usuario
- **PasswordFormat:** Tipo de formato de password
- **PasswordSalt:** Bits aleatorios para generar nuevo password
- **Email:** Email de usuario
- **PasswordQuestion:** Pregunta del password
- **PasswordAnswer:** Respuesta del password
- **IsApproved:** Usuario reconocido
- **IsLockedOut:** Usuario bloqueado
- **CreationDate:** Fecha de creación de usuario
- **LastLoginDate:** Fecha de último acceso al sistema
- **LastPasswordChangedDate:** Fecha de última modificación del password
- **LastLockoutDate:** Fecha de último bloqueo
- **FailedPasswordAttemptCount:** Número de intentos fallidos
- **FailedPasswordAttemptWindowStart:** Fecha de password fallido
- **FailedPasswordAnswerAttemptCount:** Número de fallos de respuesta
- **FailedPasswordAnswerAttemptWindowsStart:** Fecha de respuesta fallida
- **Comment:** Breve comentario

	Name	Data Type	Allow Nulls	Default
	ApplicationId	uniqueidentifier	☐	
🔑	UserId	uniqueidentifier	☐	
	Password	nvarchar(128)	☐	
	PasswordFormat	int	☐	
	PasswordSalt	nvarchar(128)	☐	
	Email	nvarchar(256)	☑	
	PasswordQuestion	nvarchar(256)	☑	
	PasswordAnswer	nvarchar(128)	☑	
	IsApproved	bit	☐	
	IsLockedOut	bit	☐	
	CreateDate	datetime	☐	
	LastLoginDate	datetime	☐	
	LastPasswordChangedDate	datetime	☐	
	LastLockoutDate	datetime	☐	
	FailedPasswordAttemptCou	int	☐	
	FailedPasswordAttemptWir	datetime	☐	
	FailedPasswordAnswerAtte	int	☐	
	FailedPasswordAnswerAtte	datetime	☐	
	Comment	nvarchar(256)	☑	

Ilustración 5.9 - Memberships Table

- **PortfolioId:** Identificador del portfolio
- **Name:** Nombre del portfolio
- **Description:** Breve descripción del portfolio
- **Title:** Título
- **UserId:** Usuario creador del portfolio
- **CreationDate**: Fecha de creación del portfolio
- **LastModificationDate**: Fecha de última modificación del portfolio

	Name	Data Type	Allow Nulls	Default
🔑	PortfolioId	uniqueidentifier	☐	
	Name	nvarchar(MAX)	☐	
	Title	nvarchar(MAX)	☐	
	Description	text	☐	
	UserId	uniqueidentifier	☐	
	CreationDate	date	☐	
	LastModificationDate	date	☐	

Ilustración 5.10 - Portfolios Table

- **ApplicationId:** Identificador de la aplicación
- **UserId:** Identificador de usuario

- **UserName:** Nombre de usuario
- **IsAnonymous:** Indica si es anónimo o no
- **LastActivityDate:** Fecha de última actividad del usuario en el sistema

	Name	Data Type	Allow Nulls	Default
	ApplicationId	uniqueidentifier	☐	
🔑	UserId	uniqueidentifier	☐	
	UserName	nvarchar(50)	☐	
	IsAnonymous	bit	☐	
	LastActivityDate	datetime	☐	

Ilustración 5.11 - Users Table

A continuación mostramos el diagrama de bases de datos donde se puede apreciar las tablas anteriormente comentadas junto con sus relaciones (Claves primarias, claves foráneas, etc.):

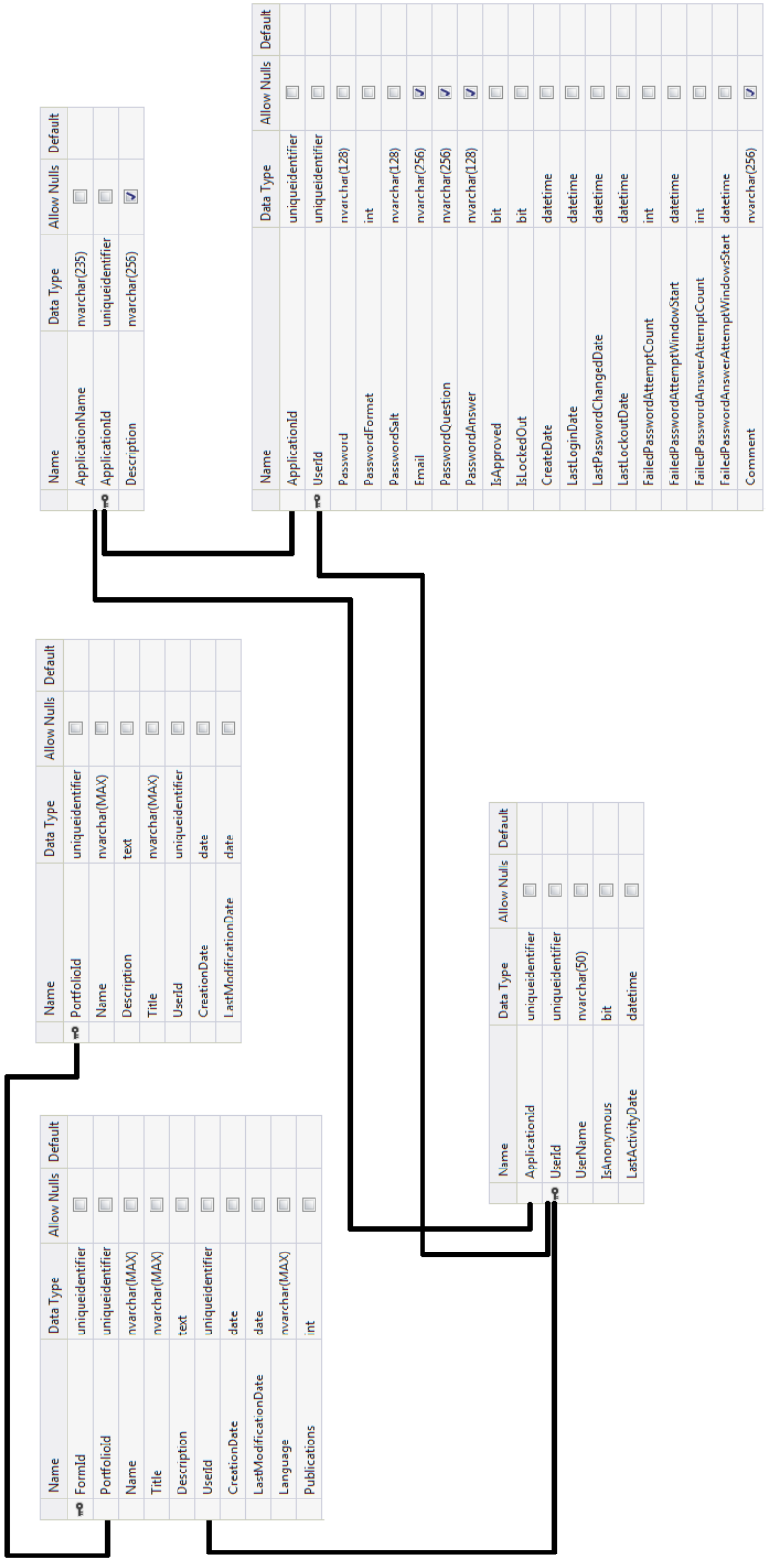

Ilustración 5.12 - Diagrama de Bases de Datos

5.1.4.2. Componentes y Subsistemas

Durante esta iteración debe implementarse una versión que permita al usuario realizar las funciones básicas relativas a la gestión de usuarios (iniciar sesión, registrarse y recuperar password). En esta iteración también se gestionan tanto los portfolios (creación, edición y eliminación) como la iniciación de la gestión de los formularios, en este caso únicamente se realiza la lectura de ellos, pero no la creación, edición o eliminación. Teniendo en cuenta todos estos aspectos en cuenta lo que tenemos que implementar es lo siguiente:

- Paquete *Model*: En este caso tenemos que implementar las clases necesarias para nuestro sistema, que son la clase *AccountModel* que incluye los modelos del inicio sesión, registro en el sistema y recuperación de password. También tenemos que implementar los modelos del formulario y el portfolio.
- Paquete *View*: Básicamente lo tenemos que implementar completamente, aunque la información que contienen dichas vistas será completada en la segunda iteración de implementación.
- Paquete *Controller:* Al igual que el paquete vistas, tenemos que implementar todas las clases controladoras, pero es evidente que la funcionalidad se completará en la siguiente iteración.
- Paquete *Content:* Este paquete se irá completando a medida que el sistema se implementa debido a que está formado por ficheros de interfaz (CSS) y las propias imágenes de la interfaz.

NOTA: El propio framework ASP.NET MVC integra por defecto la gestión de usuarios a través de las clases *Membership*, por lo tanto sería ilógico implementa nuestra propia gestión de usuarios.

5.1.4.3. Integraciones

Nos centraremos en definir las integraciones a través de los diferentes componentes del sistema especificando como se construyen y a qué paquete corresponden.

En cada integración se implementarán uno o varios casos de uso que anteriormente están recogidos en la fase de elaboración del sistema.

5.1.4.3.1. Integración 1: Paquete Model

El primer paso para el desarrollo del sistema es definir las clases de este paquete. Las únicas clases que no se definen en este caso son las clases *FineUpload* y *FineUploaderResult* que son las clases necesarias para subir imágenes y ficheros al servidor a través de los formularios.

5.1.4.3.2. Integración 2: Inicio sesión, Registro en el Sistema, Recuperación de Password

Primeramente se creará la interfaz de usuario siguiendo los pasos definidos anteriormente en el documento *Prototipo de la interfaz*. En la pantalla inicial se permitirá al usuario iniciar sesión, enlazar para registrarse en el sistema o para recuperar el password a través del correo electrónico.

5.1.4.3.3. Integración 3: Gestión de los Portfolios (Creación, edición y eliminación) y gestión de los formularios (únicamente lectura)

En este caso se realiza la gestión de los portfolios, que contiene a su vez formularios. Como bien sabemos, a la hora de crear un portfolio, tenemos que crear una plantilla que se asociará al portfolio, pues bien, en esta iteración únicamente crearemos el portfolio, lo podremos editar y eliminar.

En cuanto a la gestión de los formularios, el componente flexigrid nos agiliza el proceso de forma que nos aporta una interfaz sencilla y mucha funcionalidad. Realizaremos la lectura de los formularios y su visualización.

5.1.5. Iteración II

5.1.5.1. Componentes y Subsistemas

Durante esta iteración debe implementarse la versión completa de la aplicación XGen. Durante esta iteración se aborda la componente más complicada del proyecto, que es la construcción de las plantillas de formularios. Se termina la gestión de los formularios (edición y eliminación). Se implementa el resto de funcionalidad del sistema como es la ejecución y exportación de formularios.

Por último se finaliza la interfaz de usuario con imágenes y mensajes después de realizar un breve estudio con usuarios reales. De esta forma se pudo mejorar dicha interfaz y por lo tanto también la experiencia final de usuario.

Teniendo en cuenta todos estos aspectos en cuenta lo que tenemos que implementar es lo siguiente:

- Paquete *Model*: Se termina de implementar el resto de clases
- Paquete *View*: Vistas con más funcionalidad y con distintas componentes de interfaz (alertas, modal, tablas, etc.)
- Paquete *Controller:* Se implementa el resto de la funcionalidad en todas las clases controladoras
- Paquete *Content:* Se completa con imágenes de diferente naturaleza como pueden ser imágenes de menú, de opciones de validación o referentes a las creadas para la propia aplicación.
- Paquete *Resources*: Se inicia y termina el proceso de traducción del sistema a diferentes idiomas.

5.1.5.2. Integraciones

5.1.5.2.1. Integración 1: Paquete Model

Se termina la implementación de las clases del sistema con las clases *FineUpload* y *FineUploaderResult*. Estas clases son para subir imágenes al servidor a través de los formularios de los usuarios.

Se termina la implementación de las clases con los mensajes de error asociado a cada campo y el tipo de campo del que se trata.

5.1.5.2.2. Integración 2: Crear Portfolio, Editar Plantilla

En este caso de implementa posiblemente el que sea el caso de uso más complicado o en el cual intervienen más componentes. Al mismo tiempo se establecen los campos de los formularios que el sistema acepta o interpreta.

5.1.5.2.3. Integración 3: Edición y eliminación de formularios

Se termina la gestión de los formularios por parte del componente flexigrid. Terminamos de adaptar este componente a nuestro propio sistema de forma que podamos tanto editar como eliminar formularios.

5.1.5.2.4. Integración 4: Ejecución, exportación y publicación de formularios

Se implementa tanto la ejecución de formularios, que se muestra a través de una ventana emergente (interfaz bootstrap), la exportación de formularios a formato PDF utilizando la librería Rotativa, implementada para la plataforma .NET, como la publicación de formularios, utilizando la librería RESTSharp.

5.1.5.2.5. Integración 5: Traducción de la aplicación

Se implementa el paquete Resources que contiene la traducción del sistema en varios idiomas que son el inglés, español y francés. Con estos idiomas cubrimos una parte importante de los potenciales usuarios teniendo en cuenta que tanto el inglés como el español son el segundo y tercer idioma más hablados en el mundo respectivamente a nivel nativo.

6. PRUEBAS

6.1. PLAN DE PRUEBAS

6.1.1. Introducción

6.1.1.1. Propósito

Este apartado contiene el *Plan de Pruebas* de la aplicación *XGen*. Los objetivos de este plan son los siguientes:

- Identificar la información existente del proyecto y los artefactos a ser probados.
- Enumerar los requisitos de alto nivel necesarios para realizar las pruebas.
- Recomendar y describir las estrategias de prueba que se deben emplear.
- Identificar los recursos necesarios y proporcionar una estimación del esfuerzo para las pruebas.
- Enumerar los documentos entregables fruto del plan de pruebas.

6.1.1.2. Alcance

El *Plan de Pruebas* se aplicará a las pruebas que se llevarán a cabo en la aplicación programada. Se tratará de probar la calidad del sistema en cuanto al cumplimiento de las especificaciones definidas en el documento de análisis y verificar el correcto funcionamiento de los aspectos funcionales.

El plan de pruebas favorecerá a depurar aspectos como pueden ser la interfaz de usuario o tratamiento en cuanto al volumen de datos.

6.1.2. Requisitos de las Pruebas

Serán objetivos de las pruebas todos los casos de uso, requisitos funcionales, y requisitos no funcionales. Los casos de uso se encuentran recogidos en el apartado *Especificación de Casos de Uso*, y en el apartado *Especificación de Requisitos Software* se pueden encontrar todos los requisitos.

6.1.3. Estrategias de Prueba

La estrategia de pruebas define la forma de abordar la actividad de realización de pruebas y la definición de los objetivos de las pruebas. Las principales consideraciones que se realizan en esta sección son las técnicas a seguir y el criterio para determinar cuándo una prueba ha sido completada.

6.1.3.1. Pruebas de Funcionalidad

Estas pruebas de funcionalidad se basan principalmente en los casos de uso. El o los objetivos principales son la verificación, procesamiento y devolución de datos. La técnica utilizada se basa en una caja negra, es decir, la aplicación recibe unos datos de entrada y como resultado se producen unos datos de salida, comparando con los datos esperados.

Objetivo de la Prueba:	Garantizar el correcto funcionamiento de la funcionalidad del sistema, incluyendo navegación, datos de entrada, proceso y recuperación.
Técnica:	Ejecutar cada caso de uso, cada flujo posible, utilizando datos válidos y erróneos, para verificar: ❖ Los resultados esperados ocurren al introducir datos válidos ❖ Se muestran los mensajes de error apropiados al introducir datos incorrectos ❖ Las reglas del negocio se aplican de forma correcta
Criterio de completitud:	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han abordado.
Consideraciones especiales:	

6.1.3.2. Pruebas de Interfaz de Usuario

Este tipo de pruebas analizan la interacción de nuestro sistema con el usuario. El objetivo principal es asegurar la correcta navegabilidad de forma que el usuario pueda acceder a las todas las funciones del sistema. Además aseguran que los objetos de la interfaz funcionan correctamente.

Objetivo de la Prueba:	Verificar: ❖ La correcta navegación a través de las vistas de la aplicación ❖ Se muestran los elementos y opciones correctamente ❖ Aparecen los datos correspondientes en el lugar adecuado
Técnica:	Crear/Modificar pruebas para verificar la adecuada navegación y el estado de los objetos para cada vista de la aplicación.
Criterio de completitud:	Cada vista verificada con éxito será consistente con otras pruebas similares.
Consideraciones especiales:	

6.1.3.3. Pruebas de Integridad de Datos

Nuestro sistema obtiene datos de nuestra base de datos y por lo tanto es un elemento crucial. El objetivo principal es comprobar la integridad y consistencia de los datos almacenados.

6.1.3.4. Pruebas de Rendimiento

En este tipo de pruebas lo que se pretende es medir son tiempos de respuesta. Debido a que es una aplicación web, existe una serie de limitaciones impuestas en cuanto a los tiempos de respuesta, aunque no seremos especialmente rigurosos en este tipo de medidas.

6.1.3.5. Pruebas de Estrés

Estas pruebas sirven para encontrar errores a nivel de recursos, es decir, la poca asignación y la falta de ellos provocarían ciertos errores. Este tipo de pruebas sirven también para comprobar la carga de trabajo que puede soportar nuestro sistema. En nuestro caso, al tratarse de una aplicación web, la carga de trabajo disminuye, aparte de ser gestionada por el navegador en el cual se ejecute.

Objetivo:	Comprobar que el sistema funciona bien bajo condiciones de estrés debidas a poca memoria RAM o disco duro disponible.
Modo de Proceder:	Limitar la cantidad de memoria RAM y/o disco duro disponible en el equipo y ejecutar las pruebas de funcionalidad.
Criterios de Completitud:	Las pruebas de funcionalidad son satisfactorias bajo las condiciones especificadas.
Consideraciones Especiales:	¿Cómo limitar la memoria RAM?

6.1.3.6. Pruebas de Volumen de Datos

La idea es hacer trabajar al sistema bajo unas condiciones de gran cantidad de datos de forma que se pueda determinar si existe cualquier tipo de error. La idea para nuestro sistema sería añadir una cantidad grande de campos de diferente naturaleza (fecha, imagen, checkbox, lista dropdown, etc…). Después se compara si el sistema genera correctamente el formulario, durante cuánto tiempo lo hace (tiempo de respuesta) y si lo hace de forma correcta.

Objetivo:	Comprobar que el sistema funciona bien aunque se le someta a una simulación prolongada de un modelo suficientemente complejo.
Modo de Proceder:	Crear un modelo complejo (un formulario con un número de campos elevado y con muchas opciones para los campos correspondientes).
Criterios de	Se han ejecutado todos las pruebas planificadas y se han

Completitud:	alcanzado o excedido los límites del sistema sin que fallara.
Consideraciones Especiales:	¿Cómo de "complejo" es un formulario?

6.1.3.7. Pruebas de Seguridad y Control de Acceso

Estas pruebas son necesarias para que el usuario no acceda a opciones ni datos de otros usuarios de forma que pueda realizar modificaciones. También se realiza un control a nivel del sistema, por lo que únicamente los usuarios registrados pueden acceder al sistema.

6.1.3.8. Pruebas de Fallo / Recuperación

El objetivo de estas pruebas es que dado un fallo del sistema en un instante determinado de distinta naturaleza como puede ser hardware, software o de red, sea capaz de terminar anómalamente y de recuperarse a un estado anterior.

6.1.3.9. Pruebas de Configuración

Este tipo de pruebas sirven para comprobar que el sistema funciona en distintas configuraciones de hardware o software. En nuestro caso significaría la utilización de distintos navegadores web para la ejecución de nuestra aplicación. Deberemos asegurar la ejecución correcta independientemente de la configuración tanto software como hardware.

Objetivo:	Verificar que el objeto de prueba funciona en las configuraciones hardware / software necesarias.
Modo de Proceder:	Ejecutar la aplicación en distintos navegadores correspondientes a distintas plataformas como pueden ser Microsoft Explorer, Safari o Chrome. Existen muchos otros como Opera o Mozilla Firefox en los cuales también será ejecutada nuestra aplicación web.
Criterios de Completitud:	Para cada combinación entre el objeto de prueba y otros programas, la aplicación se ha comportado correctamente en su ejecución.
Consideraciones Especiales:	¿Qué otros programas necesitamos para realizar las pruebas? ¿Qué navegadores utilizan los usuarios?

6.1.4. Recursos

A continuación comentaremos las necesidades en cuanto a recursos de diferente naturaleza:

6.1.4.1. Trabajadores

Necesidades de personal en la fase de pruebas del sistema. Como en fases anteriores, será el alumno encargado de realizar la mayoría de las etapas en esta fase:

Trabajador	Recursos recomendados	Responsabilidades/Comentarios
Jefe de Pruebas/Diseñador de Pruebas	1	Supervisa las pruebas. ❖ Dirección Técnica ❖ Informes de gestión ❖ Identifica, prioriza e implementa casos de prueba. ❖ Genera Plan de Pruebas ❖ Genera Casos de Prueba
Probador	1	Realiza las pruebas. ❖ Registro de resultados ❖ Recuperar de los errores ❖ Documentar solicitudes de cambio

6.1.4.2. Sistema

Nuestro equipo de prueba tiene la siguiente configuración:

Equipo de Desarrollo y Prueba:

Procesador: Intel(R) Core(TM) 2 Duo Processor T5670 (1.8GHz)

RAM: 2GB DDR2

Disco Duro: 320 GB HDD

S.O: Windows 7 Professional

6.2. CASOS Y RESULTADOS DE LAS PRUEBAS

6.2.1. Introducción

6.2.1.1. Propósito

Este apartado reúne las especificaciones de dos artefactos de la metodología UPEDU: el apartado *Casos de Pruebas* y el apartado *Resultados de las Pruebas*. Se pretende con esto que quede más comprensible.

Este apartado identifica e informa de todos las condiciones que deben tenerse en cuenta en las pruebas. Dichas condiciones son obligatorias para una implementación aceptable del producto final. Están diseñadas siguiendo los casos de uso y restricciones identificadas en la etapa de *Análisis y Diseño*.

Además, para cada caso de prueba identificado se resumirán los resultados obtenidos al aplicarlo.

6.2.1.2. Estructura del apartado

Cada uno de los apartados en los que se divide este documento corresponde a un tipo de pruebas recogidas en el documento *Plan de Pruebas.* Cada apartado tiene recoge información diferente debido a que para cada tipo de pruebas los criterios son distintos.

6.2.2. Pruebas de Funcionalidad

Este apartado se divide en varias secciones, las cuales representan los casos de uso identificados en diseño. Cada caso de uso contendrá varios subapartados que seguirán la siguiente estructura:

1. *Elemento a probar*
2. *Clases de equivalencia para ese elemento:*
    a. Requisitos
    b. Clases válidas
    c. Clases no válidas
3. *Clases de prueba para ese elemento:*
    a. Entrada
    b. Resultados
    c. Clases de equivalencia cubiertas

6.2.2.1. Crear Portfolio

6.2.2.1.1. Crear Portfolio

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
Nombre de portfolio asociado a la plantilla no nulo	Campo no vacío. (1)	Campo vacío. (2)
Título de portfolio asociado a la plantilla no nulo	Campo no vacío. (3)	Campo vacío (4)
Descripción del portfolio asociado a la plantilla	Campo no vacío. (5)	Campo vacío. (6)
Lista de opciones (campos de formularios)	El sistema reconoce el campo seleccionado. (7.1) Añadir un elemento de formulario. (7.2) Se elimina el elemento del formulario. (7.3) Mover un elemento del formulario. (7.4)	El sistema no reconoce el campo seleccionado. (8.1) Añadir un campo de formulario cuando en la fila tiene ya 3 elementos. (8.2) Mover una sección arriba cuando no hay otra sección encima.

		(8.3) Mover una sección abajo cuando no hay otra sección abajo. (8.4)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre del portfolio no nulo Título del portfolio no nulo Descripción del portfolio no nulo. Pulsamos en "*OK*"	El sistema crea la plantilla del formulario, mostrando en la pantalla dicha plantilla.	1,3,5
Nombre de portfolio nulo Título del portfolio nulo Descripción del portfolio nulo. Pulsamos en "OK"	Se muestra un mensaje de error indicando que los campos son requeridos.	2,4,6
Pulsamos en "*Cancel*"	La ventana emergente se cierra y volvemos a la pantalla anterior	
Pulsamos en un campo de formulario de forma que de esta forma añadimos un elemento al formulario. (Sección, fila o campo)	El elemento se añade a la plantilla y el sistema lo añade correctamente	7.1, 7.2
Añadimos a la fila un elemento cuando hay 3 elementos en dicha fila	El formulario no se modifica y se informa al usuario que la fila está completa.	8.2
Pulsamos en "Borrar" en el correspondiente elemento del formulario (sección, fila o campo)	Se borra el elemento y el sistema nos indica que lo ha borrado correctamente	7.1, 7.3
Pulsamos en "Mover" un elemento del formulario. (sección únicamente)	La sección se mueve correctamente.	7.4
Movemos una sección hacia arriba que es la primera del formulario	El formulario no se modifica	8.3
Movemos una sección hacia abajo que es la última del formulario	El formulario no se modifica	8.4

6.2.2.2. Editar Plantilla

6.2.2.2.1. Editar Plantilla

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
El fichero XML que contiene información referente a la plantilla existe	El fichero XML con la información de la plantilla exista. (1)	El fichero XML con la información de la plantilla no exista. (2)
Nombre de portfolio asociado a la plantilla no nulo	Campo no vacío. (3)	Campo vacío. (4)
Título de portfolio asociado a	Campo no vacío. (5)	Campo vacío (6)

la plantilla no nulo		
Descripción del portfolio asociado a la plantilla	Campo no vacío. (7)	Campo vacío. (8)
Lista de opciones (campos de formularios)	El sistema reconoce el campo seleccionado. (9.1) Añadir un elemento de formulario. (9.2) Se elimina el elemento del formulario. (9.3) Mover un elemento del formulario. (9.4)	El sistema no reconoce el campo seleccionado. (10.1) Añadir un campo de formulario cuando en la fila tiene ya 3 elementos. (10.2) Mover una sección arriba cuando no hay otra sección encima. (10.3) Mover una sección abajo cuando no hay otra sección abajo. (10.4)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre del portfolio no nulo Título del portfolio no nulo Descripción del portfolio no nulo. Pulsamos en "*OK*"	El sistema crea la plantilla del formulario, mostrando en la pantalla dicha plantilla.	1,3,5,7
Nombre de portfolio nulo Título del portfolio nulo Descripción del portfolio nulo. Pulsamos en "*OK*"	Se muestra un mensaje de error indicando que los campos son requeridos.	1,4,6,8
Pulsamos en "*Cancel*"	La ventana emergente se cierra y volvemos a la pantalla anterior	
Pulsamos en un campo de formulario de forma que de esta forma añadimos un elemento al formulario. (Sección, fila o campo)	El elemento se añade a la plantilla y el sistema nos indica que lo ha insertado correctamente	1,9.1, 9.2
Añadimos a la fila un elemento cuando hay 3 elementos en dicha fila	El formulario no se modifica y se informa al usuario que la fila está completa.	1,10.2
Pulsamos en "Borrar" en el correspondiente elemento del formulario (sección, fila o campo)	Se borra el elemento y el sistema nos indica que lo ha borrado correctamente	1,9.1, 9.3
Pulsamos en "Mover" un elemento del formulario. (sección únicamente)	La sección se mueve correctamente.	1,9.4
Movemos una sección hacia arriba que es la primera del formulario	El formulario no se modifica	1,10.3
Movemos una sección hacia abajo que es la última del formulario	El formulario no se modifica	1,10.4

6.2.2.3. Ejecutar Formulario

6.2.2.3.1. Ejecutar Formulario

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
Fichero XML con la especificación del formulario	El fichero XML existe. (1)	El fichero XML no existe. (2)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Pulsamos en "*Test*"	El sistema abre una ventana emergente mostrando el formulario y todos sus componentes	1
Pulsamos en "*OK*"	El sistema cierra la ventana emergente que contiene al formulario	

6.2.2.4. Nuevo Formulario

6.2.2.4.1. Nuevo Formulario

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
Nombre del formulario no nulo	Campo no vacío. (1)	Campo vacío. (2)
Título del formulario no nulo	Campo no vacío. (3)	Campo vacío. (4)
Descripción del formulario no nulo	Campo no vacío. (5)	Campo vacío. (6)
Existen idiomas para seleccionar	Campo no vacío. (7)	Campo vacío. (8)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre no nulo Título no nulo Descripción no nulo Idioma no nulo	El sistema crea un formulario con la plantilla seleccionada anteriormente	1,3,5,7
Nombre nulo Título nulo Descripción nula Idioma no nulo (el idioma nunca puede ser nulo en menú desplegable, inglés por defecto)	Se muestra un mensaje en cada campo indicando el error correspondiente.	2,4,6,8

6.2.2.5. Modificar Formulario

6.2.2.5.1. Modificar Formulario

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
El fichero XML que contiene información referente al formulario existe	Fichero existente. (1)	El fichero no existe. (2)
Nombre del formulario no nulo	Campo no vacío. (3)	Campo vacío. (4)
Título del formulario no nulo	Campo no vacío. (5)	Campo vacío. (6)
Descripción del formulario no nulo	Campo no vacío. (7)	Campo vacío. (8)
Existen idiomas para seleccionar	Campo no vacío. (9)	Campo vacío. (10)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre no nulo Título no nulo Descripción no nulo Idioma no nulo	El sistema crea un formulario con la plantilla seleccionada anteriormente	1,3,5,7,9
Nombre nulo Título nulo Descripción nula Idioma nulo	Se muestra un mensaje en cada campo indicando el error correspondiente.	4,6,8,10

6.2.2.6. Exportar Formulario

6.2.2.6.1. Exportar Formulario

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
El fichero XML que contiene información referente al formulario existe	Fichero existente. (1)	El fichero no existe. (2)
Un nombre de archivo.	El archivo no existe todavía. (1.1) El archivo ya existe pero se puede sobreescribir. (1.2)	Nombre vacío. (2.1) No se puede escribir en la ruta especificada. (2.2) El archivo ya existe y no se puede sobreescribir. (2.3)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Un nombre de archivo que no existe.	El sistema guarda el formulario en formato PDF.	1.1
Un nombre de archivo existente pero sobrescribible.	El sistema guarda el formulario en formato PDF tras pedir confirmación al usuario.	1.2
Campo vacío.	No se hace nada.	2.1
Una ruta en la que no se tienen permisos de escritura.	Se muestra un error indicándolo.	2.2
Un archivo que está siendo usado o está protegido contra escritura.	Se muestra un error indicándolo tras intentar sobrescribirlo.	2.3

6.2.2.7. Login

6.2.2.7.1. Login

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
Nombre de usuario no nulo	Campo no vacío. (1)	Campo vacío. (2)
Contraseña no nula	Campo no vacío. (3)	Campo vacío (4)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre de usuario no nulo Contraseña no nula. Pulsamos en "*Login*"	El sistema valida el usuario, iniciando el sistema en función de los datos introducidos. Se cargan los datos del usuario.	1,3
Nombre de usuario nulo Contraseña nula Pulsamos en "*Login*"	Se muestra un mensaje de error indicando que los campos son requeridos	2,4

6.2.2.8. Registrarse

6.2.2.8.1. Registrarse

Clases de equivalencia

Requisitos	Clases válidas	Clases no válidas
Nombre de usuario no nulo	Campo no vacío. (1)	Campo vacío. (2)
Contraseña no nula	Campo no vacío. (3)	Campo vacío (4)
Confirmación de contraseña no nula e idéntica al campo de contraseña anterior	Campo no vacío. (5.1) Campo idéntico al campo de contraseña. (5.2)	Campo vacío. (6.1) Valor distinto al campo de contraseña. (6.2)
Correo Electrónico	Campo no vacío. (7.1) Formato correcto. (7.2)	Campo vacío. (8.1) Formato incorrecto. (8.2)

Clases de prueba

Entrada	Resultado	Clases cubiertas
Nombre de usuario no nulo Contraseña no nula Confirmación de contraseña no nula e igual al campo de contraseña Correo no nulo y con formato correcto Pulsamos en "*Registrarse*"	Se muestra un mensaje de confirmación de que el usuario ha sido registrado en el sistema.	1,3,5.1,5.2,7.1,7.2
Nombre de usuario nulo Contraseña nula Confirmación de contraseña nula Correo nulo Pulsamos en "*Registrarse*"	Se muestra un mensaje en cada campo indicando el error correspondiente.	2,4,6.1,8.1
Campo de confirmación de contraseña no igual al campo de contraseña Pulsamos en "*Registrarse*"	Se muestra un mensaje indicando que el usuario deber insertar la misma contraseña	6.2

Formato del campo de correo incorrecto Pulsamos en "*Registrarse*"	Se muestra un mensaje indicando que el campo de correo debe ser un formato correcto	8.2
Pulsamos en "*Borrar*"	Se borra el formulario actual de registro	

6.2.3. Pruebas de Interfaz de Usuario

La finalidad de este apartado es asegurar dos aspectos de la interfaz: la navegabilidad entre elementos y el aspecto de cada uno. Iremos recorriendo todas las ventanas organizándolas una vez más por casos de uso. Para cada una de las ventanas se listarán los elementos navegables y cómo deben funcionar. Mediante este tipo de pruebas mejoramos la experiencia de usuario.

6.2.3.1. Crear Portfolio

6.2.3.1.1. Crear Portfolio (pantalla principal)

Elemento	Funcionamiento
Botón "*Close*".	Retrocede a la vista anterior
Botón "*Test*".	Se abre una ventana con la plantilla en cuestión
Botón "*Publish*".	Se publica la plantilla de formulario en internet
Botón "*Back*".	Retrocede a la vista anterior
Botón "*URL*".	Se abre una ventana emergente donde puedes modificar la URL de publicación de la plantilla
Botón "*Settings*".	Se abre una ventana emergente donde el usuario puede modificar las opciones del portfolio.

6.2.3.1.2. Ventana emergente de cambio de URL de publicación

Elemento	Funcionamiento
Botón "*OK*".	Se guarda la URL del formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.1.3. Ventana emergente de configuración de la plantilla del formulario

Elemento	Funcionamiento
Botón "*OK*".	Se guarda la plantilla del formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.2. Editar Plantilla

6.2.3.2.1. Editar Plantilla (pantalla principal)

Elemento	Funcionamiento
Botón "*Close*".	Retrocede a la vista anterior
Botón "*Test*".	Se abre una ventana con la plantilla en cuestión
Botón "*Publish*".	Se publica la plantilla de formulario en internet
Botón "*Back*".	Retrocede a la vista anterior
Botón "*URL*".	Se abre una ventana emergente donde puedes modificar la URL de publicación de la plantilla
Botón "*Settings*".	Se abre una ventana emergente donde el usuario puede modificar la configuración del portfolio.

6.2.3.2.2. Ventana emergente de cambio de URL de publicación

Elemento	Funcionamiento
Botón "*OK*".	Se guarda la URL del formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.2.3. Ventana emergente de opciones de plantilla

Elemento	Funcionamiento
Botón "*OK*".	Se guarda el formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.3. Ejecutar Formulario

6.2.3.3.1. Ventana emergente con los campos del formulario

Elemento	Funcionamiento
Botón "*PDF*".	Exporta el formulario
Botón "*OK*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.4. Nuevo Formulario

6.2.3.4.1. Ventana emergente de configuración del formulario

Elemento	Funcionamiento
Botón "*OK*".	Se guarda el formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.4.2. Nuevo Formulario (pantalla principal)

Elemento	Funcionamiento
Botón "*Close*".	Retrocede a la vista de formularios
Botón "*Test*".	Se abre una ventana emergente con los campos del formulario (ejecutar formulario)
Botón "*Publish*".	Se publica el formulario en internet
Botón "*Clear*".	Limpia el formulario
Botón "*Save*".	Guarda el formulario
Botón "*PDF*".	Exporta el formulario
Botón "*Atrás*".	Retrocede a la vista anterior
Botón "*URL*".	Se abre una ventana emergente donde puedes modificar la URL de publicación del formulario
Botón "*Settings*".	Se abre una ventana emergente donde el usuario puede modificar la configuración del formulario

6.2.3.5. Modificar Formulario

6.2.3.5.1. Ventana emergente con campos de configuración del formulario

Elemento	Funcionamiento
Botón "*OK*".	Se guarda el formulario y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.5.2. Modificar Formulario (pantalla principal)

Elemento	Funcionamiento
Botón "*Close*".	Retrocede a la vista de formularios
Botón "*Test*".	Se abre una ventana emergente con los campos del

	formulario (ejecutar formulario)
Botón "*Publish*".	Se publica el formulario en internet
Botón "*Clear*".	Limpia el formulario
Botón "*Save*".	Guarda el formulario
Botón "*PDF*".	Exporta el formulario
Botón "*Atrás*".	Retrocede a la vista anterior
Botón "*URL*".	Se abre una ventana emergente donde puedes modificar la URL de publicación del formulario
Botón "*Settings*".	Se abre una ventana emergente donde el usuario puede modificar la configuración del formulario

6.2.3.6. Exportar Formulario

6.2.3.6.1. Ventana emergente para guardar el formulario en formato PDF

Elemento	Funcionamiento
Botón "*OK*".	Se guarda el formulario en formato PDF y se cierra la ventana
Botón "*Cancelar*".	Se cierra la ventana emergente
Botón "*x*".	Se cierra la ventana emergente

6.2.3.7. Login

6.2.3.7.1. Login (pantalla principal)

Elemento	Funcionamiento
Botón "*Login*".	Valida el usuario y entra en el sistema
Botón "*Crear Cuenta*".	Redirecciona para crear una cuenta de usuario
Botón "*Recuperar Password*".	Redirecciona para recuperar el password del usuario

6.2.3.8. Registrarse

6.2.3.8.1. Registrarse (pantalla principal)

Elemento	Funcionamiento
Botón "*Registrarse*".	Guarda el usuario en el sistema
Botón "*Iniciar Sesión*".	Redirecciona para iniciar sesión
Botón "*Limpiar*".	Borra los datos del formulario de registro de usuario

6.2.3.9. Recuperar Password

6.2.3.9.1. Recuperar Password (pantalla principal)

Elemento	Funcionamiento
Botón "*Enviar Correo*".	Envia un correo con la nueva contraseña al usuario cuyo nombre de usuario es el introducido
Botón "*Login*".	Redirecciona para iniciar sesión

6.2.4. Pruebas de Rendimiento

Para realizar un pequeño estudio sobre el rendimiento de la aplicación, hemos comparado dos modelos, uno creado con XGen y otro creado con el sistema ORBEON. Los resultados obtenidos son prometedores, el tiempo de respuesta en el caso de XGen son menores, quizás porque la funcionalidad es menor y la cantidad de información que tenemos que cargar por lo tanto también es menor.

Ambas aplicaciones se tratan de aplicaciones web, por lo tanto la forma de almacenamiento es muy similar. En caso de que ambas aplicaciones tuvieran una naturaleza diferente, es decir, una que fuera aplicación web y otra que fuera una aplicación de escritorio, tendríamos que indagar en la forma de almacenamiento y cómo eso podría afectar a los tiempos de respuesta.

6.2.5. Pruebas de Estrés

6.2.5.1. Limitación de Recursos

Hemos dedicado un equipo con ciertas limitaciones de hardware para la realización de estas pruebas. Se trata de un equipo portátil con las siguientes características:

Procesador: Intel(R) Core(TM) 2 Duo Processor T5670 (1.8GHz)

RAM: 2GB DDR2

Disco Duro: 320 GB HDD

S.O: Windows 7 Professional

Marcaremos como modelo de ordenador estándar el siguiente:

Procesador: 2.9GHz Dual-Core Intel Core i7

RAM: 8GB 1600MHz DDR3

Disco Duro: 750GB 5400-rpm HDD

S.O: Mac OS X (Mountain Lion)

Podemos observar que el procesamiento está bastante limitado, pasando de un procesador *i7 de última generación* a un procesador con la mitad de núcleos.

Por otra parte la memoria RAM está también muy limitada, la memoria RAM es exactamente una cuarta parte (2GB). Además la versión de la memoria RAM es DDR3, lo que significa que estos módulos pueden transferir datos a una tasa de reloj efectiva de 800-1600 MHz, comparado con el rango actual del DDR2 de 533-800 MHz.

Como es obvio en las aplicaciones web, es necesaria una conexión de internet. Esto es importante debido a que si la conexión es lenta, podríamos pensar que el tiempo de procesamiento es lento, lo cual no tiene nada que ver. Hemos calculado nuestro ancho de banda en alguno de los servicios ofrecidos en la red y podemos concluir que como media estamos utilizando una conexión de 10Mbps. Hemos dicho de forma media debido a que las pruebas no se han realizado en el mismo lugar y la conexión de red (WI-FI) como consecuencia, no ha sido la misma.

Como es normal, 10Mbps es una conexión suficiente para que XGen se pueda ejecutar y utilizar sin ningún tipo de problema.

6.2.5.2. Realización de las Pruebas

Se ha aplicado la batería de pruebas referidas a la funcionalidad al equipo con especificaciones menores con resultados satisfactorios.

Adicionalmente, hemos creado y trabajado con modelos de distinta naturaleza de forma que podamos abarcar un mayor abanico de posibilidades:

- ✓ Modelo con pocos campos y sencillos (time, output text, number, etc.)
- ✓ Modelo con muchos campos y sencillos (input text, text, date, etc.)
- ✓ Modelo con pocos campos pero complejos (varios tipos de menú con muchas opciones, etc.)
- ✓ Modelo con muchos campos y complejos. Además de rellenar la información de todos los campos (título, mensaje de error, ayuda, descripción del campo, etc.) de forma que la información procesada sea mayor.

6.2.6. Pruebas de Volumen de Datos

Nuestros datos (información sobre plantillas y formularios) se almacenan en ficheros XML, que habitualmente no ocupan una gran cantidad de espacio en disco (~1-50KB). En escasas ocasiones, por no decir ninguna, estos ficheros XML van a llegar a ocupar más de lo indicado ya que esto significaría que el usuario desea construir un formulario que tuviera una cantidad de campos considerable (más de 50 campos), pues bien, esto no ocurre, ya que los formularios sirven como una forma de entrada para recopilar información y por lo tanto, tienen que ser lo más sencillos y entendibles posibles por parte de los usuarios.

Tengamos en cuenta que cuanto mayor sea la información que debemos procesar, el tiempo de respuesta será mayor. Como consecuencia de esto, la experiencia de usuario se verá también deteriorada.

Aun así nosotros realizaremos las pruebas con una gran cantidad de información referente a los campos de formularios de forma que podamos verificar que a pesar de la gran cantidad de información procesada, el sistema responderá en un tiempo aceptable.

6.2.6.1. Modelos Complejos

Se ha estudiado el comportamiento de la aplicación al someterla a un modelo de formulario complejo, es decir, con muchos campos y del tipo complejo (menús de diferente tipo con muchas opciones, imágenes, etc.). Concretamente se ha estudiado el siguiente modelo.

- **Número de campos: 50**
- **Número de subcampos: 138**
- **Número de campos distintos: 6 (todos los tipos de campos complejos)**

Al cargar la plantilla con toda esta información, XGen ha sido capaz de funcionar y ejecutar la plantilla sin complicaciones. El tiempo de respuesta ha sido el adecuado y por lo tanto la experiencia de usuario es satisfactoria.

Además hemos creado un formulario con esta plantilla de forma que a la hora de la creación tuviera que cargar toda la información contenida en el fichero XML. El sistema no solo ha respondido satisfactoriamente sino que lo ha hecho en un tiempo de respuesta mínimo.

Podemos decir que la diferencia existente entre los tiempos de respuesta de un modelo sencillo (sin campos complejos y pocos campos) y un modelo complejo (con campos complejos y un número mayor de ellos) son prácticamente idénticos.

6.2.6.2. Archivos Grandes

Debemos decir que XGen es capaz de trabajar con archivos de gran tamaño (aunque en algunas situaciones no se dé el caso). Normalmente estos ficheros corresponden a modelos de plantilla de formularios, aunque existen otros tipos de ficheros como pueden ser ficheros adjuntos al formulario (imágenes o simplemente ficheros) o el propio formulario exportado como archivo PDF.

En cuanto a los ficheros XML que contienen tanto modelos de plantilla de formulario como los propios formularios, pueden tener un tamaño variable de 1-50KB y en escasas ocasiones (por no decir nunca) van a superar este valor. Las imágenes y los ficheros que se pueden adjuntar a los formularios pueden tener un tamaño no superior a 3MB. Esta decisión es variable de cada servicio, normalmente suele ser de 1-3MB, pero por ejemplo el servicio GMAIL deja adjuntar ficheros de hasta 25MB. En cuanto al fichero exportado en formato PDF depende de la cantidad de información que contenga el formulario, aunque habitualmente no ocupa una cantidad muy grande (incluso no llega a MB).

Con esto comentado y las pruebas realizadas, la aplicación web no ha presentado ningún problema en cuanto a la gestión de los distintos tipos de ficheros.

En nuestro caso particular de análisis del modelo complejo, el fichero XML de la plantilla de formulario llego a ocupar 27KB (recordemos que el número de campos era de 50, con un número de subcampos de 138). Como podemos comprobar este tamaño (27 KB) es bastante pequeño lo que significa que los tiempos de respuesta serán menores.

A la hora de exportación de un formulario con esta plantilla, el fichero (en formato PDF) aumentó de tamaño en 232KB, que sigue siendo un tamaño de almacenamiento pequeño.

6.2.7. Pruebas de Configuración

Para las pruebas de configuración se ha repetido la batería de *pruebas de funcionalidad* en distintos navegadores. Concretamente se ha ejecutado en los siguientes navegadores:

	Mozilla Firefox	Google Chrome	Safari	Internet Explorer
Versión	19.0.2	28.0.1500.95	5.1.7	9.0

Tabla 6.14 - Versiones de los Navegadores

La elección de los navegadores para realizar la las pruebas de configuración no ha sido un tema aleatorio sino que se han intentado cubrir un gran parte del mercado actual. Estos navegadores son los más utilizados por los usuarios para navegar o ejecutar aplicaciones web:

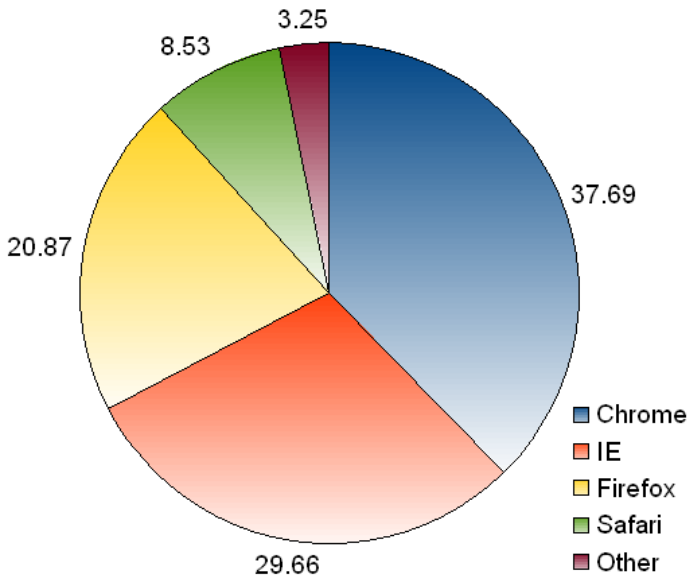

Ilustración 6.1 - Uso de navegadores 2013 (Fuente: http://stats.areppim.com)

7. CONCLUSIONES

7.1. OBJETIVOS ALCANZADOS

Tras el desarrollo del trabajo de fin de grado, es necesario casar conclusiones de forma que nos sirvan como referencias para futuras ampliaciones o proyectos. La primera pregunta que nos podemos hacer es si realmente hemos cumplido con los objetivos establecidos al principio del proyecto.

Para analizar los objetivos cumplidos, podemos comparar lo desarrollado con la *Especificación de Requisitos Software:*

- Nuestro sistema es capaz de gestionar portfolios (crear, leer, actualizar y eliminar) de forma eficiente y sencilla. Además de estas acciones, hemos ampliado la funcionalidad de forma que también podemos ordenar los portfolios por nombre, título o por fecha de última modificación y también el usuario puede realizar una búsqueda de los portfolios que se encuentran a su disponibilidad.
- En cuanto a las plantillas, el usuario puede crear sus plantillas dinámicamente al estilo WYSIWYG, que se trataba de unos de los requisitos importantes.
- Los formularios son gestionados mediante *flexigrid* que se trata de un plugin adaptado a nuestro sistema. Se pedía que los formularios fueran seleccionables de forma independiente y no solo se ha conseguido esto, sino que hemos añadido funcionalidad extra mediante este plugin, como puede ser la búsqueda, ordenación o paginación de los formularios.

  La ejecución de formularios también era un requisito vital de forma que el usuario pudiera ejecutar los formularios en una ventana emergente independiente.

  En cuanto a la publicación de los formularios, se ha habilitado una ventana donde el usuario puede cambiar la URL de publicación de formulario. El usuario recibirá un feedback adecuado a la hora de la publicación del formulario.
- El sistema también valida los campos de los formularios, dando la posibilidad al usuario de establecer un texto de mensaje de error que se mostrará en caso de que el usuario introduzca un valor que no es correcto (Ejemplo: Texto en un campo numérico).
- La gestión de usuario también la realiza el sistema XGen. Además de posibilitar al usuario el registrarse y loguearse en el sistema, el usuario puede también recuperar el password en caso de que no lo recuerde. Para esto último, nos hemos servido de librerías ofrecidas por la plataforma .NET.

Ahora comentaremos las conclusiones sacadas del proyecto respecto de la planificación y el tiempo empleado por el alumno:

- Como comentario general, cabe destacar que la poca experiencia en la planificación de proyectos ha dificultado el inicio del proyecto y el alumno tuvo que documentarse durante esta fase. El proyecto, como podemos observar, se ha retrasado respecto a la planificación inicial y el motivo puede ser que la planificación fuera demasiado optimista.
- En cuanto al tiempo diario establecido, tuvimos que aumentarlo mediante horas extras debido al retraso del proyecto. Podríamos decir que ha sido necesario aumentar el tiempo diario a 3 horas más a partir de la segundo iteración en la fase de elaboración.
- Basándonos en los resultados obtenidos en los tiempos de cada fase de la planificación del proyecto podemos decir que se adapta bastante bien a lo propuesto por UPEDU:

  UPEDU: 5%, 20%, 65% y 10% para inicio, elaboración, construcción y transición respectivamente.

  XGen: 8%, 14%, 66% y 12% para inicio, elaboración, construcción y transición respectivamente

  Como podemos ver, deberíamos haber dedicado más tiempo a la elaboración, mientras que la fase de construcción debería haber sido menor.

7.2. CONOCIMIENTOS APLICADOS Y ADQUIRIDOS

Existen aspectos importantes a la hora de la realización de un trabajo de fin de grado y son por ejemplo la aplicación de los conocimientos adquiridos durante toda la carrera o la capacidad de adquirir nuevos conocimientos y perfeccionarlos. Debemos de decir los conocimientos respecto al desarrollo web impartida en las asignaturas de toda la carrera eran nulos. El alumno ha adquirido los conocimientos con trabajos extra y desarrollando por su propia cuenta.

- Se ha adquirido conocimientos de nuevas tecnologías como desarrollo de aplicaciones web para plataforma .NET. Estos conocimientos incluyen conocimientos específicos de lenguajes como C#, HTML5 o CSS3.
- XGen tiene una fuerte componente de cliente y por lo tanto hemos tenido que perfeccionar nuestros conocimientos en JavaScript y en concreto en jQuery.
- La creación de plantillas se realiza de forma asíncrona mediante jQuery, lo que dificulta el seguimiento y la traza de los programas. Para ello se ha utilizado el debugger de Visual Studio 2012 que ha resultado una herramienta imprescindible para el desarrollo de esta aplicación.

7.3. LÍNEAS FUTURAS DE TRABAJO

XGen es un sistema que se puede cambiar a medida que pasa el tiempo y se puede adaptar a nuevas tecnologías que vayan surgiendo o incluso que hayan surgido ya.
A continuación, explicaremos posibles ampliaciones o cambios que se pueden realizar en la aplicación web:

7.3.1. Usuarios

XGen actualmente no da la posibilidad a los usuarios de modificar su información, sino que el usuario se registra y entra en el sistema. Pues bien, sería interesante crear una nueva funcionalidad de forma que posibilite al usuario modificar su propia información o rellenar nueva información como por ejemplo dirección, país o fecha de nacimiento. Para ello, tendríamos que modificar ligeramente la base de datos con nuevos campos, algo muy sencillo lógicamente.

7.3.2. Plantillas

La creación de plantillas tiene un número ajustado de campos y se puede ampliar fácilmente. El único problema respecto a este tema es que los campos disponibles para los formularios están ampliamente difundidos y reconocidos por la comunidad de usuarios. A pesar de esto, repetimos que en el caso hipotético que se creara un nuevo tipo de campo, éste se podría añadir a XGen sin mayores problemas.

7.3.3. Formularios

Una idea que podría ser interesante e iría relacionada con la gestión de usuarios, sería el tema de la compartición de formularios de forma que existan tipos de roles (Propietario, espectador o editor) y cada usuario podría crear sus propios formularios, pero también gestionar formularios pertenecientes a otros usuarios, según el rol que le ha sido asignado.

Por último podemos pensar en adaptar XGen a una de las tecnologías de cliente que recientemente han aparecido. Estos frameworks JavaScript basados en MVC ofrecen mayor productividad y mantenimiento de código. Una opción podría ser AngularJS.
AngularJS es un framework javascript *opensource* desarrollado por Google. Es un framework para crear Webapps en lenguaje cliente con Javascript ejecutándose con el conocido single-page applications (aplicación de una única página) que extiende el tradicional HTML con etiquetas propias (directivas) como pueden ser *ng-app, ng-controller, ng-model, ng-view…*
Un framework basado en MVC (Modelo-Vista-Controlador) increíblemente flexible, de muy fácil lectura y desarrollo rápido.
Permite extender HTML con tags personalizados, definir y vincular (data-binding) variables vista/controllador, consultas AJAX con peticiones HTTP, sistema óptimo de templating, manipulación de datos en JSON, inyección de dependencias, formularios de validación, desacoplamiento del DOM de Javascript, internacionalización i18n y l10n, etc.
AngularJS es compatible con los navegadores de última generación (Chrome, Firefox, Safari, Opera, Webkits, IE9+) y se puede hacer compatible para Internet Explorer 8 o anterior.

8. BIBLIOGRAFÍA

Entorno de Desarrollo

[1] http://tfs.visualstudio.com/ Fecha último acceso 26 de Marzo de 2013
[2] http://msdn.microsoft.com/es-es/library/vstudio/dd831853.aspx Fecha último acceso 26 de Marzo de 2013
[3] http://channel9.msdn.com/tags/VSTS/ Fecha último acceso 27 de Marzo de 2013
[4] http://en.wikipedia.org/wiki/.NET_Framework Fecha último acceso 27 de Marzo de 2013

XForms

[5] http://www.w3c.es/Divulgacion/GuiasBreves/XForms Fecha último acceso 3 de Abril de 2013
[6] http://www.xml.com/pub/a/2003/12/30/xforms.html Fecha último acceso 3 de Abril de 2013
[7] http://php.net/manual/en/features.xforms.php Fecha último acceso 5 de Abril de 2013
[8] http://www.ibm.com/developerworks/xml/library/x-xformsintro1/ Fecha último acceso 3 de Abril de 2013
[9] http://www.w3.org/TR/xforms/ Fecha último acceso 5 de Abril de 2013
[10] http://wiki.orbeon.com/forms/about-xforms Fecha último acceso 5 de Abril de 2013
[11] http://en.wikibooks.org/wiki/XQuery/Simple_XForms_Examples Fecha último acceso 5 de Abril de 2013
[12] http://homepages.cwi.nl/~steven/Talks/2012/08-07-steven-serialisation/examples.html Fecha último acceso 6 de Abril de 2013
[13] http://www.w3.org/MarkUp/Forms/2010/xforms11-for-html-authors/ Fecha último acceso 7 de Abril de 2013
[14] http://www.eyecatch.no/blog/2013/01/using-xforms-and-mvc-in-an-episerver-7-block/ Fecha último acceso 7 de Abril de 2013
[15] http://eric.van-der-vlist.com/blog/2013/03/06/when-mvc-becomes-a-burden-for-xforms/ Fecha último acceso 10 de Abril de 2013
[16] http://mirrors.fe.up.pt/pub/nongnu//xforms/xforms.pdf Fecha último acceso 11 de Abril de 2013
[17] http://xforms-module.script.soft32download.com/ Fecha último acceso 11 de Abril de 2013
[18] http://extxsltforms.sourceforge.net/sitKubera/index/index.xml Fecha último acceso 13 de Abril de 2013
[19] http://www.joncurtis.co.uk/post/ASPNet-MVC-XForms.aspx Fecha último acceso 13 de Abril de 2013
[20] http://137.69.120.115:8080/xformsdemo/ Fecha último acceso 13 de Abril de 2013

ASP.NET MVC

[21] http://www.codeplex.com/ Fecha último acceso 14 de Abril de 2013
[22] http://www.slideshare.net/rmaclean/aspnet-mvc-2630894 Fecha último acceso 14 de Abril de 2013
[23] http://www.w3schools.com/aspnet/mvc_intro.asp Fecha último acceso 15 de Junio de 2013
[24] http://www.asp.net/web-pages/tutorials/security/16-adding-security-and-membership Fecha último acceso 15 de Abril de 2013
[25] http://demos.telerik.com/aspnet-ajax/ Fecha último acceso 15 de Abril de 2013
[26] http://stackoverflow.com/questions/13418488/checkboxlist-in-asp-net-mvc4 Fecha último acceso 16 de Abril de 2013
[27] http://blog.osbornm.com/2010/07/21/using-simplemembership-with-asp.net-webpages/ Fecha último acceso 16 de Abril de 2013
[28] http://www.asp.net/single-page-application/overview/templates/durandaljs-template Fecha último acceso 16 de Abril de 2013
[29] http://www.dotnetcurry.com/ShowArticle.aspx?ID=847 Fecha último acceso 16 de Abril de 2013
[30] http://www.asp.net/mvc/tutorials/hands-on-labs/aspnet-mvc-4-models-and-data-access Fecha último acceso 16 de Abril de 2013

Autenticación

[30] http://mvc4beginner.com/Sample-Code/Authentication-Authorization/ASP-.NET-MVC-SignOn-Register-In-The-Same-Form.html Fecha último acceso 17 de Abril de 2013
[30] http://www.1stwebdesigner.com/css/35-excellent-jquery-and-css3-navigation-menus/ Fecha último acceso 17 de Abril de 2013

DataGrid
[31] http://mvc4beginner.com/Sample-Code/Insert-Update-Delete/Asp-.Net-MVC-Ajax-Insert-Update-Delete-Using-Flexigrid.html Fecha último acceso 18 de Abril de 2013

XML
[32] http://support.microsoft.com/kb/307379/es Fecha último acceso 18 de Abril de 2013

Bootstrap
[33] http://jacobrask.github.com/styledocco/styledocco/examples/bootstrap/docs/buttons.html Fecha último acceso 19 de Abril de 2013
[34] http://twitter.github.com/bootstrap/components.html Fecha último acceso 19 de Abril de 2013

Edit in Place
[35] http://www.appelsiini.net/projects/jeditable Fecha último acceso 22 de Junio de 2013
[36] http://www.codeproject.com/Articles/265211/Using-JEditable-plugin-as-ASP-NET-MVC3-jQuery-inline Fecha último acceso 22 de Abril de 2013

Internacionalización
[37] http://www.asp.net/mvc/overview/internationalization Fecha último acceso 23 de Abril de 2013
[38] http://afana.me/post/aspnet-mvc-internationalization.aspx Fecha último acceso 23 de Abril de 2013
[39] http://adamyan.blogspot.com.es/2010/02/aspnet-mvc-2-localization-complete.html Fecha último acceso 23 de Abril de 2013

Bases de Datos
[40] http://demo.easyquerybuilder.com/asp-net-mvc/ Fecha último acceso 24 de Abril de 2013
[41] http://msdn.microsoft.com/en-us/library/hh510202.aspx Fecha último acceso 24 de Abril de 2013
[42] http://stackoverflow.com/questions/11572953/querying-database-mvc-4 Fecha último acceso 26 de Abril de 2013

Procesamiento XForms
[43] https://community.emc.com/docs/DOC-4345 Fecha último acceso 26 de Abril de 2013
[44] https://code.google.com/p/ubiquity-xforms/wiki/UsingTheLibrary Fecha último acceso 26 de Abril de 2013

[45] http://solidforms.sourceforge.net/xforms.html Fecha último acceso 27 de Abril de 2013
[46] http://solidforms.sourceforge.net/domintegration.html Fecha último acceso 27 de Abril de 2013
[47] http://lists.w3.org/Archives/Public/www-forms/2011May/0009.html Fecha último acceso 28 de Abril de 2013
[48] http://chiba.sourceforge.net/ChibaUserGuide.pdf Fecha último acceso 28 de Abril de 2013
[49] http://en.wikibooks.org/wiki/XSLTForms Fecha último acceso 29 de Abril de 2013
[50] http://extxsltforms.sourceforge.net/sitKubera/index/index.xml Fecha último acceso 29 de Abril de 2013
[51] http://docs.1060.org/docs/3.0.0/book/developerreference/doc_ura_XForm.html Fecha último acceso 30 de Abril de 2013
[52] http://docs.1060.org/docs/3.0.0/book/solutiondeveloperguide/doc_guide_xforms.html Fecha último acceso 1 de Mayo de 2013
[53] http://tutorialgenius.blogspot.com.es/2011/06/c-mvc-3-render-xslt-htmlhelper.html Fecha último acceso 1 de Mayo de 2013

XML Dinámico
[54] http://forums.asp.net/t/1471269.aspx/1 Fecha último acceso 25 de Abril de 2013
[55] http://stackoverflow.com/questions/13137800/import-a-remote-xml-file-into-mvc-4-webapp Fecha último acceso 3 de Mayo de 2013
[56] http://www.yulebiao.com/questions/15385204/how-to-read-and-process-xml-file-in-mvc Fecha último acceso 3 de Mayo de 2013
[57] http://support.microsoft.com/kb/307548 Fecha último acceso 3 de Mayo de 2013
[58] http://www.google.ge/search?q=custom%20view%20engine Fecha último acceso 4 de Mayo de 2013
[59] http://www.yulebiao.com/questions/9254653/how-to-read-schema-based-xml-files-with-nested-elements-into-table-type-objects Fecha último acceso 4 de Mayo de 2013

[60] http://forums.asp.net/t/1134452.aspx/1 Fecha último acceso 4 de Mayo de 2013
[61] http://www.drdobbs.com/windows/parsing-xml-files-in-net-using-c/184416669 Fecha último acceso 4 de Mayo de 2013
[62] http://papermashup.com/parse-xml-with-jquery/ Fecha último acceso 5 de Mayo de 2013
[63] http://packages.nuget.org/packages/NXKit.Web.UI/ Fecha último acceso 5 de Mayo de 2013
[64] https://github.com/wasabii/nxkit Fecha último acceso 5 de Mayo de 2013
[65] http://stackoverflow.com/questions/2491853/server-side-xforms-form-validation-and-integration-into-asp-net Fecha último acceso 8 de Mayo de 2013
[66] http://www.tutarz.com/xforms/ Fecha último acceso 8 de Mayo de 2013
[67] http://forums.asp.net/t/1894278.aspx/1?How+to+Convert+Xml+to+Xslt+in+mvc3 Fecha último acceso 9 de Mayo de 2013
[68] http://tpeczek.blogspot.com.es/2009/12/xml-and-xslt-transformation-in-aspnet_21.html Fecha último acceso 9 de Mayo de 2013
[69] http://dhtmlx.com/docs/products/dhtmlxAjax/index.shtml Fecha último acceso 10 de Mayo de 2013
[70] http://www.sitepoint.com/display-data-asp-xmlxsl/ Fecha último acceso 12 de Mayo de 2013

PDF

[71] http://www.codeproject.com/Articles/335595/Rotativa-how-to-print-PDF-in-Asp-Net-MVC Fecha último acceso 12 de Mayo de 2013
[72] http://stackoverflow.com/questions/14197284/rotativa-download-with-saveas-dialog Fecha último acceso 12 de Mayo de 2013

Paginación y Ordenación

[73] http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/sorting-filtering-and-paging-with-the-entity-framework-in-an-asp-net-mvc-application Fecha último acceso 13 de Mayo de 2013
[74] http://www.c-sharpcorner.com/UploadFile/raj1979/paging-sorting-in-mvc-4/ Fecha último acceso 13 de Mayo de 2013

Acciones de Formulario

[75] http://www.cjorellana.net/2012/11/datepicker-con-mvc-4-razor.html Fecha último acceso 14 de Mayo de 2013
[76] http://forums.asp.net/t/1879457.aspx/1 Fecha último acceso 14 de Mayo de 2013
[77] http://www.eyecon.ro/bootstrap-datepicker/ Fecha último acceso 15 de Mayo de 2013
[78] http://stackoverflow.com/questions/253689/switching-a-div-background-image-with-jquery Fecha último acceso 15 de Mayo de 2013
[79] http://stackoverflow.com/questions/3151082/problem-with-asp-net-mvc-selectlist-and-listselectlistitems Fecha último acceso 15 de Mayo de 2013
[80] http://weblogs.asp.net/jeffwids/archive/2011/01/31/jquery-textbox-in-sortable-is-not-clickable-selectable.aspx Fecha último acceso 15 de Mayo de 2013
[81] http://docs.jquery.com/UI/API/1.8/Selectable Fecha último acceso 15 de Mayo de 2013
[82] http://www.codeproject.com/Articles/379980/Fancy-ASP-NET-MVC-Image-Uploader Fecha último acceso 16 de Mayo de 2013
[83] http://stackoverflow.com/questions/4444955/how-to-use-the-plupload-package-with-asp-net-mvc Fecha último acceso 17 de Mayo de 2013
[84] http://stackoverflow.com/questions/9301409/valums-fileuploader-with-multiple-instances Fecha último acceso 17 de Mayo de 2013
[85] http://stackoverflow.com/questions/13776851/how-to-use-multiple-select-in-mvc-4 Fecha último acceso 17 de Mayo de 2013
[86] https://github.com/eternicode/bootstrap-datepicker/issues/312 Fecha último acceso 17 de Mayo de 2013

Ventanas Emergentes

[87] http://stackoverflow.com/questions/16016761/mvc-4-modal-window-partial-view-and-validation Fecha último acceso 17 de Mayo de 2013
[88] http://www.codeproject.com/Articles/146396/A-Comparison-of-Three-jQuery-Modal-Dialogs-for-ASP Fecha último acceso 18 de Mayo de 2013
[89] http://www.codeproject.com/Articles/315535/How-to-render-MVC-View-on-a-Modal-Popup-Window Fecha último acceso 18 de Mayo de 2013

[90] http://demos.kendoui.com/web/colorpicker/index.html Fecha último acceso 18 de Mayo de 2013
[91] http://forums.asp.net/t/1742525.aspx/1 Fecha último acceso 18 de Mayo de 2013
[92] http://ricardocovo.com/2010/09/02/asp-mvc-delete-confirmation-with-ajax-jquery-ui-dialog/ Fecha último acceso 18 de Mayo de 2013
[93] http://bootboxjs.com/ Fecha último acceso 18 de Mayo de 2013
[94] http://twitter.github.io/bootstrap/javascript.html#popovers Fecha último acceso 18 de Mayo de 2013

Editor WYSIWYG
[95] http://www.aloha-editor.org/demos.php Fecha último acceso 18 de Mayo de 2013

ViewBag y ViewData
[96] http://www.variablenotfound.com/2010/12/viewbag-en-aspnet-mvc-3.html Fecha último acceso 19 de Mayo de 2013

Email
[97] http://www.itorian.com/2013/03/PasswordResetting.html Fecha último acceso 18 de Mayo de 2013
[98] http://mvc4beginner.com/Sample-Code/MVC4-Sending-Email/ASP.NET-MVC4-Sending-Email.html Fecha último acceso 18 de Mayo de 2013
[99] http://stackoverflow.com/questions/13619417/sending-email-asp-net-mvc4
[100] http://www.intstrings.com/ramivemula/c/how-to-send-an-email-using-c-net-with-complete-features/ Fecha último acceso 19 de Mayo de 2013
[101] http://www.codeproject.com/KB/aspnet/SMTPGmail.aspx Fecha último acceso 19 de Mayo de 2013
[102] http://www.aspdotnetfaq.com/Faq/How-to-send-HTML-Email-from-ASP-NET-using-your-Gmail-account.aspx Fecha último acceso 19 de Mayo de 2013
[103] http://thekevincode.com/2010/09/adding-email-confirmation-to-asp-net-mvc/ Fecha último acceso 20 de Mayo de 2013
[104] http://confirm.apphb.com/Account/Register Fecha último acceso 21 de Mayo de 2013
[105] http://stackoverflow.com/questions/14686383/failure-sending-mail-in-asp-net-mvc4-smtp-sendmail Fecha último acceso 22 de Mayo de 2013

Cross-Domain HTTP
[106] http://www.htmlgoodies.com/html5/tutorials/learn-how-to-use-the-jsonp-data-format-with-jquery.html#fbid=VfihTk9_W7l Fecha último acceso 22 de Mayo de 2013
[107] http://www.integratedwebsystems.com/2010/07/cross-domain-jsonp-using-asp-net-mvc-and-jquery/ Fecha último acceso 23 de Mayo de 2013
[108] http://stackoverflow.com/questions/13137800/import-a-remote-xml-file-into-mvc-4-webapp Fecha último acceso 23 de Mayo de 2013
[109] http://stackoverflow.com/questions/1968180/how-to-send-a-xml-file-over-http-and-https-protocol-and-get-result-back Fecha último acceso 24 de Mayo de 2013
[110] http://www.codeproject.com/Articles/10430/Post-XML-Data-to-an-ASP-NET-Page-using-C Fecha último acceso 24 de Mayo de 2013
[111] http://go4answers.webhost4life.com/Example/post-xml-url-c-82758.aspx Fecha último acceso 24 de Mayo de 2013
[112] http://csharp-codesamples.com/2009/02/data-transfer-using-self-hosted-wcf-service/
[113] http://http://www.thereforesystems.com/capture-xml-in-wcf-service/ Fecha último acceso 24 de Mayo de 2013
[114] http://csharp.net-informations.com/xml/csharp-xmltutorial.htm Fecha último acceso 25 de Mayo de 2013
[115] http://stackoverflow.com/questions/5653743/c-sharp-httpwebrequest-with-xml-structured-data Fecha último acceso 25 de Mayo de 2013
[116] http://stackoverflow.com/questions/3981564/cannot-send-a-content-body-with-this-verb-type Fecha último acceso 26 de Mayo de 2013
[117] http://www.codeproject.com/Articles/426769/Creating-a-REST-service-using-ASP-NET-Web-API Fecha último acceso 26 de Mayo de 2013
[120] http://stackoverflow.com/questions/13385206/cannot-put-post-using-rest-sharp-using-xml Fecha último acceso 26 de Mayo de 2013

APÉNDICE A: SOFTWARE EMPLEADO

A.1. Programas

A.1.1. Visual Studio Express 2012 for Web

Es un programa de desarrollo en entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Es de carácter gratuito y es proporcionado por la compañía Microsoft Corporation orientándose a principiantes, estudiantes y aficionados de la programación web y de aplicaciones, ofreciéndose dicha aplicación a partir de la versión 2005 de Microsoft Visual Studio.

Visual Studio Express permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002, se incorpora la versión Framework 3.5, Framework 4.0 y Framework 4.5 para las ediciones 2005, 2008, 2010 y 2012). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Cabe destacar que estas ediciones son iguales al entorno de desarrollo comercial de Visual Studio Professional pero sin características avanzadas.

Para programación con lenguaje ASP.NET. Está orientado a la programación y diseño web, incluyendo un editor visual WYSIWYG y otro HTML con autocompletado de código (IntelliSense), coloración de sintaxis y validación. Aparte de ASP.NET, también soporta Visual Basic .NET y C Sharp (C#). También tiene un servidor web local para realizar pruebas en ASP.NET, un depurador para ubicar errores en el código fuente y una herramienta de publicación en línea de sitios creados.

- Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de Microsoft SQL Server llamada SQL Server Express Edition cuyas principales limitaciones son que no soporta bases de datos superiores a 10 GB de tamaño, únicamente utiliza un procesador y 1 Gb de RAM y no cuenta con el Agente de SQL Server.

A.1.2. Gantt Project

Se trata de una herramienta para la gestión de proyectos del estilo a Microsoft Project. Con ella se puede descomponer el proyecto en un árbol de tareas y asignar recursos humanos a cada una de ellas. También se pueden establecer dependencias entre tareas. GanttProject muestra dos perspectivas del proyecto: un diagrama de Gantt con las tareas y un diagrama de carga de recursos para éstos. Con esta aplicación se pueden imprimir los diagramas, generar informes en formato PDF o HTML e intercambiar datos con Microsoft Project y otras aplicaciones.

Al igual que MSProject, nos permitía gestionar todo lo que necesitábamos, a lo cual hay que añadirle su carácter gratuito, por lo que se consideró útil para desarrollar XGen. La versión utilizada fue la 2.6.

A.1.3. StartUML

Es un proyecto de libre distribución para desarrollar especificaciones UML/MDA rápidamente y de forma flexible, extensible y gratuita en una plataforma Win32. El objetivo del proyecto StarUML es construir una herramienta de modelado de software capaz de competir con herramientas comerciales como por ejemplo Rational Rose, Together, etc.

Entre sus características se encuentran las siguientes:

- Creación de diagramas UML 2.0 de casos de uso, clases, secuencia, colaboración, máquinas de estados, actividad, componentes y despliegue.
- Soporte para varios lenguajes de programación: Java, C++ y C#.
- Generación automática de documentación en formato de Microsoft Office.
- Soporta el uso de patrones.
- Verifica el modelo que se está construyendo.
- Generación personalizada de código.

- Soporte para tecnología MDA.
- Extensibilidad.

El equipo de desarrollo también estaba muy familiarizado con esta herramienta, lo cual fue un aspecto importante a la hora de escoger qué herramienta de modelado utilizar. El hecho de que fuera gratuita también facilitó la decisión. La versión empleada fue la última estable distribuida, versión 5.0 lanzada el 30 de diciembre del año 2005.

A.1.4. Fiddler

Fiddler es un programa de depuración del servidor proxy de aplicación escrito por Eric Lawrence de Microsoft. Se captura el tráfico HTTP y lo registra para que el usuario lo revise. También se puede utilizar para analizar el tráfico de HTTP a medida que se envía. De manera predeterminada, el tráfico de WinINET de Microsoft HTTP (S) de pila se dirige automáticamente a través de Fiddler en tiempo de ejecución, pero a través de Fiddler, se puede dirigir el tráfico de cualquier navegador o aplicación.

A.1.5. Help and Manual

Es una herramienta para documentación que permite definir un conjunto de páginas de la ayuda mediante una interfaz muy sencilla y después ofrece la posibilidad de exportarlo a un conjunto muy amplio de formatos: HTML Help, Webhelp, Adobe PDF, Visual Studio Help, MS Word Manual, Classic Winhelp o eBook. Además permite crear la estructura de la documentación en 106 idiomas diferentes. El formato PDF es ideal para crear manuales de usuario, y Help & Manual tiene un *add-on* que permite definir una plantilla para utilizar en la exportación a dicho formato.

Como extra hay que decir que también es capaz de importar sistemas de ayuda desarrollados previamente en HTML, Compiled HTML Help y otros formatos para poder seguir desarrollándolos con esta aplicación.

En nuestro caso necesitábamos crear tanto un manual de usuario como un sistema de ayuda que fueran independientes de la plataforma/navegador en la que se ejecutase XGen.

A.1.6. Adobe Reader

El software Adobe Reader permite a los usuarios leer, imprimir e interactuar con archivos en formato de documento portátil (PDF), que Adobe creó en 1993. Adobe diseñó el lector como parte del paquete de Acrobat, una suite de software que permite a los usuarios crear, editar y convertir archivos PDF. Actualmente en su undécima versión, Adobe Reader funciona en todas las plataformas, incluyendo Windows, Linux y Mac.

A.2. LIBRERÍAS

A.2.1. PagedList

Es un plugin utilizando en nuestro sistema para la paginación de nuestros *portfolios* de forma que el usuario pueda gestionarlos más cómodamente y de una forma más intuitiva y sencilla.

A.2.2. Flexigrid

Se trata de un plugin para jQuery en forma de data-grid para la presentación de datos (en nuestro caso *formularios*). Entre las características más importantes está la de adaptase con datos de tipo XML o JSON usando AJAX para cargar el contenido.

A.2.3. jEditable

Se trata de un plugin para jQuery que habilita al usuario a editar en línea, es decir, editar texto de forma que se actualice instantáneamente. Esto supone una mejor aceptación del usuario frente al sistema.

A.2.4. Rotativa

Esta librería sirve para exportar los formularios a formato PDF. Estos ficheros PDF tendrán una estructura determinada dependiendo del formulario que se desee exportar.

A.2.5. RESTSharp

Es una librería (REST) disponible para plataforma .NET que sirve como cliente HTTP de forma que podemos hacer peticiones a servicios web. Esta librería nos permite modificar información como por ejemplo el tipo de método usado (GET, POST, PUT), añadir cabeceras (header), de forma que por ejemplo podemos indicar el tipo de información enviada, o el método de envío. En nuestro caso en particular enviaremos un fichero adjunto XML.

APÉNDICE B: FORMATOS EXPORTACIÓN-PUBLICACIÓN

B.1. PDF

B.1.1. Definición

Portable Document Format (formato de documento portable) es el formato de archivos desarrollado por Adobe Systems y creado con los programas Adobe Acrobat Reader, Acrobat Capture, Adobe Distiller, Adobe Exchange, y el plugin Amber de Adobe Acrobat.

Esta tecnología ha tenido éxito estandarizando el formato de los documentos que se utilizan y transfieren en Internet. El PDF es como un formato de archivos universal.

B.1.2. Historia

Hace unos años, mucha de la documentación que se servía en soporte digital, generalmente almacenada en diskettes o CD-ROM's, tenía formato de texto plano y más adelante en archivos de Microsoft Word que, sin duda muchos usuarios han pasado por ello, no era siempre posible abrir si no se disponía de la versión más reciente de este popular procesador de textos. Luego estos formatos y algunos otros también bastante extendidos han ido cediendo terreno a dos tipos de documentos que poseen grandes ventajas:

HTML, lenguaje en el que se elaboran la mayoría de páginas Web accesibles desde Internet.

PDF, estándar para libros electrónicos con grandes capacidades de presentación.

Ambos formatos, especialmente el segundo, poseen la enorme ventaja frente a muchos otros que han de poder visualizarse con iguales o muy similares resultados independientemente del programa o versión del mismo que se utilice para interpretarlos, amén de permitir crear documentos muy atractivos a la vista y nada complejos de manipular o gestionar internamente por los equipos que los carguen.

B.1.3. Motivación para usarse en XGen

La estandarización de este tipo de formato y el tratamiento de archivos ONLINE ha hecho que sea el formato más común. En nuestro sistema es de especial interés el tratamiento de ficheros de formularios (exportar formulario), por lo tanto se adecúa perfectamente a la funcionalidad de nuestro sistema.

B.1.4. Especificación

Independientemente de cómo se hayan creado los ficheros PDF, todos ellos comparten la misma estructura interna compuesta de cuatro partes:

- *Cabecera*: Información sobre la especificación del estándar PDF que se ha seguido en donde se indica, por ejemplo, la versión.
- *Cuerpo*: Descripción de los elementos usados en las páginas del fichero.
- *Tabla de referencias cruzadas*: Información de los elementos usados en las páginas del fichero.
- *Coda*: Indica dónde encontrar la tabla de referencias cruzadas.

Hay que notar que cuando un fichero PDF es modificado y se añade nuevo contenido, éste tendrá nuevas secciones de cuerpo, tabla de referencias cruzadas y coda pero al guardar este documento podemos optimizarlo para que las secciones duplicadas se fusionen en sólo una y se reorganice el fichero.

B.2. XML

B.2.1. Definición

El formato XML (Lenguaje de Marcas Extensible) es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

B.2.2. Historia

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (*Generalized Markup Language*), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (*Standard Generalized Markup Language*), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información.

En el año 1989 Tim Berners Lee creó la Web, y junto con ella el lenguaje HTML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar. Los navegadores Web sin embargo siempre han puesto pocas exigencias al código HTML que interpretan y así las páginas Web son caóticas y no cumplen con la sintaxis. Estas páginas Web dependen fuertemente de una forma específica de lidiar con los errores y las ambigüedades, lo que hace a las páginas más frágiles y a los navegadores más complejos.

Otra limitación de SGML es que cada documento pertenece a un vocabulario fijo, establecido por el DTD. No se pueden combinar elementos de diferentes vocabularios. Asimismo es imposible para un intérprete (por ejemplo un navegador) analizar el documento sin tener conocimiento de su gramática (del DTD). Por ejemplo, el navegador sabe que antes de una etiqueta `<div>` debe haberse cerrado cualquier `<p>` previamente abierto. Los navegadores resolvieron esto incluyendo lógica *ad hoc* para el HTML, en vez de incluir un analizador genérico. Ambas opciones, de todos modos, son muy complejas para los navegadores. Se buscó entonces definir un subconjunto del SGML que permita:

- Mezclar elementos de diferentes lenguajes. Es decir que los lenguajes sean extensibles.
- La creación de analizadores simples, sin ninguna lógica especial para cada lenguaje.
- Empezar de cero y hacer hincapié en que no se acepte nunca un documento con errores de sintaxis.

Para hacer esto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que necesiten interpretar el documento.

B.2.3. Uso en XGen

Este formato es el formato principal de nuestra aplicación debido a que el estándar XForms utiliza el formato XML para su especificación.

En XGen, los ficheros XML son utilizados para establecer la estructura de los formularios que contienen a su vez toda la información (campos de formularios) de los propios formularios.

Por otro lado, a la hora de la publicación online de los formularios, son los propios ficheros XML los que se envían a través de una llamada HTTP en el cliente. El usuario receptor por lo tanto podrá recibir el fichero XML en el servidor.

B.2.4. Especificación

A continuación se listan las principales características de este formato. Para más información conviene visitar la especificación oficial del W3C.

B.2.4.1. Partes de un documento XML

Un documento XML se compone principalmente de dos partes:

- *Prólogo*: es opcional, declara un documento como XML, lo enlaza con su definición de tipo de documento (DTD) y puede contener una o varias instrucciones de procesamiento.
- *Cuerpo*: no es opcional, debe contener un y sólo un elemento raíz.

A su vez el cuerpo puede contener varios elementos de los que se describen a continuación:

- *Elementos*: pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.
- *Atributos*: los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.
- *Entidades predefinidas*: para representar caracteres especiales que, de esta forma, no sean interpretados como marcado en el procesador XML.
- *Secciones CDATA*: es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Sólo se utiliza en los atributos, no confundir con (`#PCDATA`) que es para los elementos. Permite que caracteres especiales no rompan la estructura.
- Comentarios: a modo informativo para el programador que han de ser ignorados por el procesador.

B.2.4.2. Documentos XML bien formados

Los documentos denominados como "bien formados" (del inglés *well formed*) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (*parser*) que cumpla con la norma.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.
- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc.
- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos "entendibles" por las personas.

APÉNDICE C: MANUAL DE USUARIO

C.1. Instalación en Localhost

XGen no necesita instalación ninguna, el usuario tiene únicamente que instalar el software *Visual Studio Express 2012 for Web* especificado en el documento *Software Empleado*. Para poder iniciar el sistema, el usuario tiene que abrir el fichero *demo.sln* (consultar apéndice *Contenido del CD-ROM*). Tras abrir el fichero se cargará el proyecto completo que contiene la solución XGen y que posteriormente se podrá ejecutar.

C.1.1. Instalación de la base de datos

Como es habitual en un sistema web, hace uso de una base de datos que tenemos que instalar en nuestro *localhost*. Para ello se adjuntan los scripts necesarios (consultar apéndice *Contenido del CD-ROM*)) para la generación de la base de datos que el usuario podrá ejecutar desde el propio IDE.

C.1.2. Fase de Producción

En el caso de que queramos que XGen pase a la fase de producción y por lo tanto a disposición de los clientes, deberemos contratar un servicio *hosting* adecuado (en nuestro caso un servidor IIS) y crear la base de datos en el servidor contratado.

Como resultado final, nuestro sistema se encontrará disponible en internet en la dirección contratada, por ejemplo:

<www.xgen.com>

Introduciendo esta dirección en el navegador, cualquier usuario podrá hacer uso de nuestro sistema.

C.2. Manual de Usuario y Sistema de Ayuda

Se ha incluido el manual de usuario de la aplicación como parte de la documentación del trabajo de fin de grado en el CD-ROM que se entrega junto con esta memoria. Se puede consultar en el directorio correspondiente (consultar apéndice *Contenido del CD-ROM*).

XGen incluye además un subsistema de ayuda en formato *Webhelp*, es decir, compuesto por un conjunto de páginas Web donde se almacena toda la información. Para consultarlo necesitará tener instalado un navegador Web (recomendamos Mozilla Firefox 19.0.2). En las primeras páginas de la ayuda encontrará información sobre cómo usarla.

Para acceder al sistema de ayuda desde la aplicación web, el usuario tiene que hacer click en el icono de ayuda, situado en la parte superior de la aplicación (que estará siempre visible). Si el usuario hace click, se abrirá una nueva pestaña donde aparecerá un conjunto de páginas con un índice de referencia.

Ilustración C.1 - Icono del Sistema de Ayuda

APÉNDICE D: CONTENIDO DEL CD-ROM

D.1. APLICACIÓN WEB (CÓDIGO FUENTE)

Directorio Principal

`/Código Fuente/`

Subdirectorios

`/Demo/`
Contiene el código fuente de la aplicación organizados en directorios
También contiene el fichero (*demo.sln*) para cargar nuestro sistema en Visual Studio (*Microsoft Visual Studio Solution*)

`/Packages/`
Contiene todos los paquetes que nuestra aplicación necesita para su ejecución

`/Scripts BBDD/`
Scripts de creación de las tablas de la base de datos

D.2. DOCUMENTACIÓN

Directorio Principal

`/Documentación/`

Subdirectorios

`/Documentos Metodología UPEDU/`
Contiene los documentos originales de la metodología UPEDU desarrollados durante todo el proyecto

`/Manual de Usuario/`
Contiene el manual de usuario en formato PDF

`/Memoria/`
Contiene la memoria del TGF en un único archivo

`/Sistema de Ayuda/`
Contiene el sistema de ayuda del sistema en formato HTML de forma que abriendo el archivo *index.html* el usuario puede visualizar esta ayuda en su navegador

`/Vídeo Demo/`
Breve vídeo de demostración de la aplicación

D.3. PROGRAMAS

Directorio Principal

`/Programas/`

Subdirectorios

`/ Adobe Acrobar Reader 10.1.4/`
Contiene el instalable del lector de archivos de PDF

`/Fiddler 4/`
Contiene el instalable del depurador HTTP