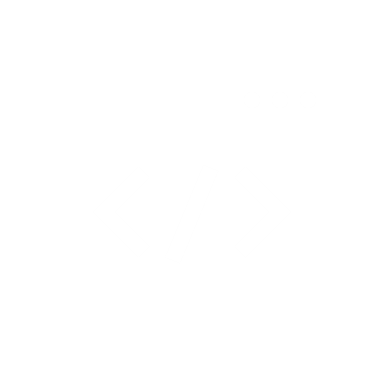
****

**Proyecto de Desarrollo**

**Trabajo de Fin de Grado**

**Julio 2019**

*Plantilla Para Trabajos De Fin De Grado De Desarrollo De La EII V1.77*

**Autor**

José Manuel Redondo López

**Autor**

José Manuel Redondo López

**Director**

José Manuel Redondo López

**Director**

José Manuel Redondo López

**Grado en Ingeniería Informática del Software**

Grado en Ingeniería Informática del Software

**Versión 1.76 (01/07/2019)**

**Modelo Métrica 3**

Copyright (C) 2019 **JOSÉ MANUEL REDONDO LÓPEZ**. [1]

*Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.*

*A copy of the license is included in the section entitled "GNU Free Documentation License".*

**NOTA MUY IMPORTANTE**: Esta plantilla es sólo una **guía orientativa**. Un TFG no tiene por qué tener ni todas sus secciones ni sólo las que aparecen aquí. **USAR ESTA PLANTILLA NO ES OBLIGATORIO PARA HACER UN TFG EN ABSOLUTO, SOLO UNA AYUDA OPCIONAL SI EL DIRECTOR/ALUMNO DESEA USARLA.** La plantilla debe ser adaptada a cada TFG (añadiendo secciones, quitando algunas o modificando los contenidos que aparezcan según cada caso particular), y el **alumno debe consultar siempre a su director, cuyas indicaciones SIEMPRE tienen prioridad** ante dudas acerca de cualquier tema.

Esta plantilla usa una **versión reducida de la metodología Métrica 3** (<https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.XRGxTY_taUk>), dado que aunque no es una metodología ágil, ayuda a documentar el proceso de creación del proyecto, el trabajo realizado y los problemas que se han tenido que resolver. El uso de esta metodología para hacer el TFG **no es en absoluto obligatorio**, y este documento puede adaptarse para usar metodologías ágiles tipo SCRUM o bien para documentar el proyecto de acuerdo con las plantillas ARC42 (<https://arc42.org/overview/>). En este último caso, el contenido de las plantillas ha sido mapeado sobre los distintos puntos de Métrica 3 para ayudar a explicar mejor lo que hay que poner en cada punto.

Finalmente, aunque este documento es un fichero único Word, nada impide usarlo solo como guía de puntos y generar la documentación automáticamente en varios formatos siguiendo la filosofía “**Documentation as Code**” que se pudo ver durante vuestros estudios.

Finalmente, **INSISTIR DE NUEVO EN QUE EL USO DE ESTE DOCUMENTO NO ES OBLIGATORIO** y que **SIEMPRE PREVALECERÁ EL CONTENIDO DE LAS ASIGNATURAS RELACIONADAS** sobre lo que ponga el mismo. La idea de este documento es servir de **AYUDA**, **NO** ser una **NORMA**.

Este es un documento de gran tamaño, por lo que no es raro que se puedan encontrar fallos. En ese caso, si quieres colaborar para mejorarla envía **un correo con tus incidencias al usuario redondojose del correo de Uniovi**. **GRACIAS**.

**Instrucciones de uso**

La plantilla está pensada para ser usada en Word 2016 o superior, y define una serie de estilos importantes:

- **Normal**: Texto normal del documento, a usar en todas las secciones para el texto

- **NormalTextoPlantilla**: Es equivalente al anterior, pero es el usado por las explicaciones, guías, sugerencias, etc. De la plantilla. Es decir, cosas que **no deberían ir** en el documento final. Se ha hecho con un estilo distinto para que sea más fácil localizar este tipo de contenido y eliminarlo en el documento de cara a su entrega. Tiene además otro color para hacer esto más sencillo visualmente. El documento final no debería tener ningún texto con este estilo ni con su variante para ejemplos (**NormalEjemploPlantilla**), que se usa para ejemplos concretos en cada apartado.

- **Ilustración**, **Pie de Tabla**: Estilos para pies de imagen y de tabla que luego se pueden usar para hacer automáticamente los índices correspondientes, ya que se han creado para usarlos. Es aconsejable que los pies de imágenes y tablas usen el estilo correspondiente para que los índices se generen bien.

- **Capiítulo1-5**: Estilos de capítulo para hacer el índice, que se ha creado para que usarlos a la hora de generarlos

- **Imagen**: Estilo para las imágenes en sí, les proporciona un marco y las centra.

Ninguno de estos estilos es obligatorio, pero se incluyen para facilitar el trabajo. **Tampoco lo son ninguna de las secciones, adáptalas, quítalas, añade lo que quieras en función de tu TFG.**

Índices, pies de página (título y autor) y bibliografía se pueden actualizar automáticamente con botón derecho encima – Actualizar todo. Es importante hacerlo en cuanto sea aconsejable para evitar dejar los elementos de la plantilla por defecto en esos apartados. En el caso de pies de página, hay que hacerlo **dos veces**: antes de la sección PSI y después de la sección PSI, ya que son dos secciones del documento independientes

**Historial de cambios**

**V1.77**: Revisión de la versión anterior con los siguientes añadidos:

* Mejora de la sección de referencias y más ejemplos

**V1.76**: Revisión de la versión anterior con los siguientes añadidos:

* Incluidas referencias a herramientas útiles para hacer diagramas, aportadas por Jordán Pascual Espada

**V1.75**: Revisión de la versión anterior con los siguientes añadidos:

* Incorporación de información sobre las plantillas de ARC42 usadas en la asignatura de Arquitectura del Software aportada por José Emilio Labra
* Sustitución de la información que mencionaba las WCAG 1.0 por las mucho más modernas WCAG 2.1
* Más información de referencia a la hora de hacer operaciones relacionadas con la seguridad y la administración de sistemas en los trabajos de fin de estudios, incluyendo checklist que ayuden a documentar medidas de seguridad tomadas a varios niveles.
* Mención expresa a los principios SOLID y a la asignatura “Software para Dispositivos Móviles”

**V1.72**: Revisión de la versión anterior con los siguientes añadidos:

* Instrucciones de uso extra relativas a actualización de elementos
* Cambio visual en la portada para indicar que es un proyecto de desarrollo a simple vista
* Mención a la asignatura de “Aspectos sociales, legales, éticos y profesionales de la Informática” en la sección CSI 1: Estándares y normas seguidos
* Mención a la asignatura “Administración de Sistemas y Redes” en las secciones DSI 5: Diagramas de despliegue y CSI 6: Elaboración de manuales de usuario.
* Mención a la asignatura “Seguridad de Sistemas Informáticos” en las secciones ASI 2: Establecimiento de Requisitos, CSI 1: Preparación del Entorno de Generación y Construcción e IAS 5: Pruebas de implantación del sistema. Con este añadido se han introducido nociones básicas para dotar al producto desarrollado de unos niveles de seguridad básicos.

**V1.7**: Esta versión es una revisión con el ánimo de hacer corrección de errores y actualización de elementos de la versión 1.4, en uso durante más de 14 años. Sigue siendo un documento- modelo **NO OBLIGATORIO** y puede no considerarse adecuado para algunas aplicaciones. Por ejemplo, la metodología seguida, Métrica 3 versión reducida y adaptada a un trabajo académico de carácter individual, **puede no adaptarse fácilmente al tipo de proyecto a realizar**. Además, está **pensada única y exclusivamente para proyectos de desarrollo**. Para proyectos de investigación **existe otra plantilla con estructura completamente distinta**. En algunas asignaturas se pueden suministrar además otros modelos de plantilla, como la de proyectos basados en la norma UNE, que **se deberán usar en lugar de esta** en caso de que este proyecto siga esa norma.

El autor entrega esta plantilla bajo la licencia **GNU Free Documentation License**. Esto significa que, si bien las secciones en color marrón deben borrarse, **ni el copyright notice ni la sección del apéndice con el contenido de la licencia deben eliminarse**. Esto no me otorga derecho alguno sobre tú trabajo ni su contenido, solamente debe hacerse para seguir las normas de la licencia usada y atribuir la plantilla usada como base a su autor original. La plantilla tiene **una única invariant section que tampoco puede eliminarse**, que es la referencia a donde está disponible públicamente este documento-modelo en la sección de referencias. La parte no modificable **solo es esa referencia**, no la sección en sí, a la que se le pueden añadir todas las que haga falta.

**En resumen:**

* **No borres el copyright notice de la página 2**
* **No borres la referencia a dónde está colgada la plantilla**
* **No borres el apéndice con el texto de la GNU Free Documentation License.**

Esas son las únicas condiciones de uso. Por favor, si no estás de acuerdo con estas condiciones de la GFDL, **te pido que no uses esta plantilla ni un trabajo derivado de ella**. No considero que se entrometan en el desarrollo de ningún TFG que las use y es un mínimo para preservar la autoría de este documento.

**Los cambios principales realizados son:**

* La plantilla se encuentra ahora protegida bajo la licencia GFDL, que garantiza su uso y modificación libre siempre que se sigan las normas de dicha licencia, que implica que no se elimine la atribución a su autor original.
* Cambio completo de formato adaptado a los nuevos logos e imagen corporativa de la Universidad, eliminando problemas conocidos con el Word y adaptándose a que ya no va a ser impresa (aunque el formato es compatible con su impresión).
* El texto de ejemplo se presenta ahora en un estilo y color diferente para facilitar su localización y eliminación de cara al documento final.
* Se han hecho adaptaciones para adherirse mejor a la estructura de la metodología Métrica 3, aunque este documento sigue siendo una versión recortada de la metodología, adaptada al contexto de proyectos académicos y al tiempo disponible para un proyecto de este tipo.
* Las antiguas secciones de memoria e introducción obligaban a repetir texto y generaban habitualmente problemas. Estas secciones se han eliminado y sustituido por una versión reducida de la sección PSI (Planificación del Sistema de Información) de Métrica 3, que permite definir los elementos iniciales de forma mucho más clara.
* El estudio de alternativas pasa ahora a formar parte de la sección de métrica 3 EVS (Estudio de Viabilidad del Sistema)
* La definición de interfaces de usuario se ha concentrado en una sola sección en lugar de dividirse en análisis y diseño, de manera que ahora se documenta y realiza mucho más al principio del desarrollo de la aplicación.
* La sección de implementación, desarrollo de las pruebas y manuales se han integrado en la de Construcción del sistema de información
* Se ha añadido el apéndice con la licencia (no eliminable)
* Se han hecho referencias explícitas a varias asignaturas cuyos contenidos impartidos **SIEMPRE** deben ser los usados en las secciones correspondientes, antes de cualquier cosa que pueda decir la plantilla al respecto. Los nombres de las asignaturas implicadas siempre aparecen entrecomillados y en **color rojo, negrita**.
* La sección de entregables se ha adaptado al nuevo sistema de entrega de contenidos online
* Se han eliminado resumen y palabras clave por que la sección de planificación de Métrica 3 ya cubre ese apartado.

Agradecimientos

Esta sección no es en absoluto obligatoria, pero es el lugar correcto para dedicar el proyecto a las personas/instituciones/empresas/… que se desee.

Índice de contenido

[Capítulo 1 Planificación del Sistema de Información 15](#_Toc23438269)

[PSI 1: Inicio del Plan de Sistemas de Información 16](#_Toc23438270)

[PSI 1.1: Análisis de la Necesidad del PSI 16](#_Toc23438271)

[PSI 1.2: Identificación del Alcance del PSI 17](#_Toc23438272)

[PSI 1.3: Determinación de Responsables 19](#_Toc23438273)

[PSI 2: Definición y Organización del PSI 20](#_Toc23438274)

[PSI 2.1: Especificación del Ámbito y Alcance 20](#_Toc23438275)

[PSI 2.2: Organización del PSI 23](#_Toc23438276)

[PSI 2.3: Definición del Plan de Trabajo 25](#_Toc23438277)

[PSI 3: Estudio de la Información Relevante 26](#_Toc23438278)

[PSI 3.1: Selección y Análisis de Antecedentes 26](#_Toc23438279)

[PSI 7: Definición de la Arquitectura Tecnológica 27](#_Toc23438280)

[PSI 7.1: Identificación de las Necesidades de Infraestructura Tecnológica 27](#_Toc23438281)

[PSI 7.2: Selección de la Arquitectura Tecnológica 27](#_Toc23438282)

[Capítulo 2 Estudio de Viabilidad del Sistema 29](#_Toc23438283)

[EVS 4, 5 y 6: Estudio y Valoración de Alternativas de Solución y Selección de Alternativa Final 30](#_Toc23438284)

[Sistema 1 30](#_Toc23438285)

[Sistema 2 30](#_Toc23438286)

[Capítulo 3 Análisis del Sistema de Información 31](#_Toc23438287)

[ASI 1: Definición del Sistema 32](#_Toc23438288)

[Determinación del Alcance del Sistema 32](#_Toc23438289)

[ASI 2: Establecimiento de Requisitos 33](#_Toc23438290)

[Obtención de los Requisitos del Sistema 33](#_Toc23438291)

[Identificación de Actores del Sistema 34](#_Toc23438292)

[Especificación de Casos de Uso 35](#_Toc23438293)

[ASI 3: Identificación de Subsistemas de Análisis 38](#_Toc23438294)

[Descripción de los Subsistemas 38](#_Toc23438295)

[Descripción de los Interfaces entre Subsistemas 38](#_Toc23438296)

[ASI 4: Análisis de los Casos de Uso 39](#_Toc23438297)

[Caso de Uso 1 41](#_Toc23438298)

[Caso de Uso 2 43](#_Toc23438299)

[ASI 5: Análisis de Clases 47](#_Toc23438300)

[Diagrama de Clases 47](#_Toc23438301)

[Descripción de las Clases 47](#_Toc23438302)

[ASI 8: Definición de Interfaces de Usuario 49](#_Toc23438303)

[Descripción de la Interfaz 49](#_Toc23438304)

[Definición del aspecto de la interfaz 50](#_Toc23438305)

[Descripción del Comportamiento de la Interfaz 50](#_Toc23438306)

[Diagrama de Navegabilidad 51](#_Toc23438307)

[ASI 10: Especificación del Plan de Pruebas 52](#_Toc23438308)

[Capítulo 4 Diseño del Sistema de Información 55](#_Toc23438309)

[DSI 3: Diseño de Casos de Uso Reales 56](#_Toc23438310)

[Caso de Uso 1.1 57](#_Toc23438311)

[Caso de Uso 1.2 57](#_Toc23438312)

[DSI 4: Diseño de Clases 59](#_Toc23438313)

[Diagrama de Clases 59](#_Toc23438314)

[DSI 5: Diseño de la Arquitectura de Módulos del Sistema 60](#_Toc23438315)

[Diagramas de Paquetes 60](#_Toc23438316)

[Diagramas de Componentes 60](#_Toc23438317)

[Diagramas de Despliegue 61](#_Toc23438318)

[DSI 6: Diseño Físico de Datos 63](#_Toc23438319)

[Descripción del SGBD Usado 63](#_Toc23438320)

[Integración del SGBD en Nuestro Sistema 63](#_Toc23438321)

[Diagrama E-R 63](#_Toc23438322)

[DSI 9: Diseño de la Migración y Carga Inicial de Datos 64](#_Toc23438323)

[DSI 10: Especificación Técnica del Plan de Pruebas 65](#_Toc23438324)

[Pruebas Unitarias 65](#_Toc23438325)

[Pruebas de Integración y del Sistema 65](#_Toc23438326)

[Pruebas de Usabilidad y Accesibilidad 66](#_Toc23438327)

[Pruebas de Accesibilidad 70](#_Toc23438328)

[Pruebas de Rendimiento 70](#_Toc23438329)

[Capítulo 5 Construcción del Sistema de Información 73](#_Toc23438330)

[CSI 1: Preparación del Entorno de Generación y Construcción 74](#_Toc23438331)

[Estándares y normas seguidos 74](#_Toc23438332)

[Lenguajes de programación 74](#_Toc23438333)

[Herramientas y programas usados para el desarrollo 76](#_Toc23438334)

[CSI 2: Generación del Código de los Componentes y Procedimientos 77](#_Toc23438335)

[CSI 3: Ejecución de las Pruebas Unitarias 78](#_Toc23438336)

[CSI 4: Ejecución de las Pruebas de Integración 79](#_Toc23438337)

[CSI 5: Ejecución de las Pruebas del Sistema 80](#_Toc23438338)

[Prueba de Usabilidad 80](#_Toc23438339)

[Pruebas de Accesibilidad 85](#_Toc23438340)

[CSI 6: Elaboración de los Manuales de Usuario 98](#_Toc23438341)

[Manual de Instalación 98](#_Toc23438342)

[Manual de Ejecución 99](#_Toc23438343)

[Manual de Usuario 99](#_Toc23438344)

[Manual del Programador 100](#_Toc23438345)

[CSI 8: Construcción de los Componentes y Procedimientos de Migración y Carga Inicial de Datos 101](#_Toc23438346)

[Capítulo 6 Implantación y Aceptación del Sistema 103](#_Toc23438347)

[IAS 1: Establecimiento del Plan de Implantación 104](#_Toc23438348)

[IAS 4: Carga de Datos al Entorno de Operación 105](#_Toc23438349)

[IAS 5: Pruebas de Implantación del Sistema 106](#_Toc23438350)

[IAS 7: Preparación del Mantenimiento del Sistema 107](#_Toc23438351)

[IAS 8: Establecimiento del Acuerdo de Nivel de Servicio 108](#_Toc23438352)

[IAS 9-10: Presentación y Aprobación del Sistema y Paso a Producción 109](#_Toc23438353)

[Capítulo 7 Apéndices 111](#_Toc23438354)

[Presupuesto 112](#_Toc23438355)

[Problemas Encontrados Durante el Desarrollo 113](#_Toc23438356)

[Conclusiones 114](#_Toc23438357)

[Ampliaciones 115](#_Toc23438358)

[Referencias Bibliográficas 116](#_Toc23438359)

[Contenido entregado en los anexos 119](#_Toc23438360)

[Contenidos 119](#_Toc23438361)

[GNU Free Documentation License 122](#_Toc23438362)

[ADDENDUM: How to use this License for your documents 129](#_Toc23438363)

Índice de Figuras

[Localización de la futura sala de exposiciones de la escuela 24](#_Toc23438479)

[Ejemplo de caso de uso 1 35](#_Toc23438480)

[Ejemplo de caso de uso 2 36](#_Toc23438481)

[Descripción de las actividades de un escenario con un diagrama de robustez (I) 42](#_Toc23438482)

[Descripción de las actividades de un escenario con un diagrama de robustez (II) 43](#_Toc23438483)

[Diagrama de clases de ejemplo 47](#_Toc23438484)

[Boceto de una interfaz 50](#_Toc23438485)

[Ejemplo simple de arquitectura del sistema 61](#_Toc23438486)

Índice de Tablas

[Equipos de trabajo y usuarios 24](#_Toc23438487)

[Definición del plan de trabajo 25](#_Toc23438488)

[Catalogo definitivo de requisitos 33](#_Toc23438489)

[Medidas de seguridad tomadas en el código de la aplicación 76](#_Toc23438490)

[Checklist de pruebas de Usabilidad 85](#_Toc23438491)

[Checklist Nivel A 93](#_Toc23438492)

[Checklist Nivel AA 94](#_Toc23438493)

[Checklist Nivel AAA 96](#_Toc23438494)

[Medidas de seguridad del servidor instalado 99](#_Toc23438495)

[Estructura general del fichero anexo entregado 120](#_Toc23438496)

[Estructura de la carpeta “desarrollo” del fichero anexo entregado 121](#_Toc23438497)

# Planificación del Sistema de Información

Fase de Planificación

PSI

## PSI 1: Inicio del Plan de Sistemas de Información

Es importante destacar que esta plantilla está pensada para proyectos de desarrollo fundamentalmente, pero que en función del tipo de proyecto (para empresa real [2], para labores de aprendizaje [3], con tecnologías móviles / sensores IoT [4] o con fines educativos [5] [6] [7]) el plan de trabajo variará enormemente. No es que un proyecto sea mejor que otro, sino que son distintos. Es importante tener claro que cada proyecto es distinto y por tanto su definición inicial, objetivos, tipos de usuario, etc. Serán prácticamente únicos, con lo que debemos adaptar el trabajo a realizar a ello desde el principio. Esta plantilla no está pensada para proyectos de investigación [8], donde probar la viabilidad de una idea es mucho más prioritario que su implementación como un producto finalizado.

El objetivo de esta actividad es determinar la necesidad del trabajo de fin de grado a construir y llevar a cabo su arranque formal. Como resultado, se obtiene una descripción general **adecuada para todos los posibles perfiles de lectores** (es decir, no entrando en tecnicismos), contemplando puntos como los siguientes:

* Definición inicial del trabajo a realizar
* Objetivos generales
* A qué tipo de usuarios afecta o, si es para una empresa real, qué parte de esta se va a ver beneficiada por este trabajo de fin de grado
* Si el trabajo de fin de grado es para una empresa real, es importante mencionar los factores que harían que el proyecto se considerase un éxito y los participantes en el mismo dentro del organigrama de la empresa. Lo primero también es oportuno incluso aunque no sea para una empresa real.

Se pude dividir en los siguientes apartados

### PSI 1.1: Análisis de la Necesidad del PSI

Analiza las expectativas del cliente del trabajo de fin de grado (ya sea real, el propio director del trabajo o bien simulado), así como los productos finales esperados, dejando claro por qué es necesario hacer el trabajo de fin de grado.

**EJEMPLO**

El director de la Escuela de Ingeniería Informática nos ha pedido el desarrollo de un sistema informático que automatice y coordine diversas tareas de gestión de un museo de informática que quiere organizar en los espacios de la escuela. Este museo será gestionado en parte por el personal de administración y los becarios de la escuela y sus piezas en exposición serán adquiridas mediante donaciones y/o compras en mercados de segunda mano. Para ello, nos ha pedido el desarrollo de un software que gestione toda esta infraestructura. Este software debe servir tanto para gestionar las donaciones de material entrante para el museo, como para su catalogación, identificación y exposición en alguna de las vitrinas o expositores que la escuela tendrá. Además, el software permitirá introducir información acerca de los elementos y automatizará la impresión de cartelería sobre ellos mediante el uso de plantillas de documentos personalizadas para cada tipo de elemento.

### PSI 1.2: Identificación del Alcance del PSI

Se define el trabajo de fin de grado en términos de los procesos del cliente que se vayan a ver afectados (simplificación, automatización…). Se determinan los objetivos estratégicos que deben ser considerados para que este trabajo de fin de grado se considere un éxito. **Se trata de elaborar un texto que sirva para preestablecer el conjunto de requisitos a cumplir por el proyecto.**

**EJEMPLO**

Actualmente el sistema que gestiona al museo de informática de la escuela se basa en los siguientes **procesos** que necesitan mejoras:

* Un profesor recibe a título personal las donaciones de materiales poniéndose el donante en contacto directo con él. Este proceso es muy ineficiente y requiere un contacto directo con el profesor responsable vía redes sociales o email. El sistema a desarrollar debería poder recoger las peticiones de donación de los interesados, almacenarlas y enviar un aviso a los profesores responsables de que hay interesados en hacer una donación, para que estos puedan ponerse en contacto con los donantes y que esta tenga lugar. Con este esquema podría haber varios profesores / responsables a la espera de donaciones y gestionarlas de forma dinámica.
* Si ningún profesor responsable puede establecer una cita con el donante para examinar la donación, o si el donante decide hacerlo así directamente (una vez dado el visto bueno por parte de algún responsable si la pieza tiene interés), cualquier donante puede depositar en la secretaría de la escuela la pieza que quiere donar para que se avise al profesor responsable para su retirada. El software a desarrollar debería permitir automatizar esos avisos y dejar pendientes de recogida los elementos que fueran llegando, para que alguno de los profesores responsables pueda pasar a por ellos en cuanto pueda.
* Las donaciones suelen llevar el nombre del donante como forma de reconocer que ha hecho dicha donación, si el donante autoriza a ello. En la actualidad el profesor responsable adjunta a cada pieza recogida el nombre del donante manualmente. El software a desarrollar debería permitir la identificación (opcional) del donante, tanto si la donación se hace contactando con los profesores a través del software como llevando la pieza donada a la conserjería (en cuyo caso el personal de administración procederá a preguntar por el nombre del donante cuando introduzca la pieza recogida en el sistema).
* Una vez la pieza donada esté en manos de algún profesor responsable, este procederá a su identificación y almacenamiento. La escuela dispone de varios almacenes donde guardar las piezas donadas hasta que se proceda a su exposición, y actualmente no se tiene un control exacto de dónde está cada una de ellas. El software a desarrollar debería ser capaz de guardar dónde está cada una de las piezas donadas y quién y cuándo la ha almacenado en cada sitio, para poder hacer un control automatizado de inventario.
* Tras la recogida de piezas, actualmente el profesor responsable, de forma manual y en su tiempo libre, cataloga y hace cartelería acerca de las donaciones para exponer las piezas. Dado que esta labor es demasiado costosa para un solo profesor, el software a desarrollar debería ser capaz de asignar a un responsable para una pieza, que se encargaría de su catalogación y hacer todas las gestiones de cara a su exposición. Esto permitiría la colaboración de varios profesores para esta labor sin posibilidad de colisiones, pudiendo involucrar también a otros miembros de la comunicada educativa, como becarios, en estas tareas.
* Debido a la carencia de soporte informático para ello, la cartelería se hace de forma puramente manual a través de plantillas de Microsoft Publisher. El software a desarrollar debería poder automatizar la creación de carteles mediante el uso de plantillas-modelo predefinidas que se rellenarían con la información pertinente de cada modelo concreto. De esta forma, se diseñarían manualmente una serie de carteles-modelo para cada tipo de pieza (CPUs, placas, gráficas, etc.) y el software a desarrollar permitiría generar el cartel de cada pieza mediante la selección de un modelo de cartel base y la colocación de fotos y texto en los huecos de dicho modelo según lo indicado por el usuario. Los carteles, una vez terminados, deberían poder imprimirse en PDF para exponerlos.
* Finalmente, con la información recabada de cada elemento y su cartel asociado, el software debería ofrecer un catálogo virtual de piezas que permitiría su filtrado, búsqueda y vista en detalle vía web. Esta vista web debería estar adaptada para dispositivos móviles.

Por tanto, los **objetivos estratégicos** a lograr para que el proyecto sea un éxito son:

* Automatizar la gestión de la entrega de donaciones
* Automatizar la gestión de las piezas entregadas, identificando donde está almacenadas y permitiendo a varios tipos de usuario que hagan la labor de su catalogación.
* Gestión del fondo de catálogo de piezas entregadas (búsquedas, filtrados...)
* Automatización de la exposición física
* Automatización de la exposición virtual

### PSI 1.3: Determinación de Responsables

Este apartado tiene especial sentido si el trabajo es complementario al de otro proyecto o se hace en parejas o grupos. En ese caso, hay que delimitar qué proyectante es responsable de cada cosa y hacer una descripción de las labores de cada uno, para dejar claro dónde están los límites de cada posible proyecto involucrado.

Si el proyecto es para una empresa real, además conviene mencionar quienes son los responsables de la dirección y del seguimiento del proyecto dentro de la misma.

**EJEMPLO**

* **La dirección** de la escuela se encargará de la supervisión de todas las fases del proyecto
* **El proyectante** se encargará de la creación de los módulos software descritos, sin establecer colaboración con otros proyectantes.
* **El director del proyecto** será el responsable de validar la consecución de los objetivos intermedios a medida que el proyecto se vaya desarrollando.
* **Una serie de usuarios escogidos entre becarios, personal de administración o externos** se encargarán de labores de pruebas / carga de datos.

## PSI 2: Definición y Organización del PSI

En esta actividad se detalla el alcance del trabajo fin de grado, se organiza el equipo de personas que lo va a llevar a cabo (si lo hubiera) y se elabora un calendario de ejecución (una planificación **siguiendo lo indicado en la asignatura de “Dirección y Planificación de Proyectos Informáticos“** de cuarto curso). Todos lo elaborado en esta actividad debe tener mucho más detalle que la actividad anterior, especificando objetivos, procesos del cliente que afecta, participantes, resultados y fechas de entrega. La sección anterior tenía un carácter “para todos los públicos”, a diferencia de esta. La sección se compone de los siguientes apartados.

### PSI 2.1: Especificación del Ámbito y Alcance

En esta tarea se describe **en detalle** el ámbito de los procesos del cliente que se van a mejorar gracias a este trabajo de fin de grado. Igualmente, se definirá el alcance, es decir, los objetivos específicos del trabajo de fin de grado. Puede ser necesario determinar distintos objetivos para cada fase del trabajo, si este tuviera varias fases. También conviene especificar qué cosas no se contemplan hacer por motivos de complejidad/tiempo.

**EJEMPLO**

En función de los objetivos estratégicos vistos, el proyecto se divide en las siguientes fases/objetivos generales, con los siguientes objetivos por cada fase:

**Fase 1: Gestión de entrega de donaciones**

Se desarrollará una página web en la que se registrarán como receptores de donaciones los profesores que quieran ser responsables del museo. Esta página permitirá a cualquier persona introducir los datos de una solicitud de donación que quiera hacer, especificando un e-mail de contacto, la forma preferida de hacerla llegar (mediante conserjería o entrega directa a algún profesor responsable en un lugar que no tiene por qué ser la escuela, previo acuerdo con el profesor) y una descripción de la pieza a entregar. Los profesores responsables recibirán un aviso mediante correo electrónico de la solicitud realizada, y uno de ellos irá a la web y se asignará la solicitud para ser tratada. La solicitud puede ser rechazada (en caso de que ya se tuviera una cantidad suficiente del tipo de pieza donada) o aceptada, en cuyo caso se procederá a establecer contacto directo con el donante vía email o bien se avisará a conserjería de que va a llegar una pieza nueva, que a su vez usará la web para avisar de que la entrega pendiente se ha realizado. Si la solicitud fuera rechazada, el sistema debería encargarse de enviar un correo de agradecimiento especificando la razón por la cual no ha sido aceptada.

Objetivos de la fase:

* Que cualquiera pueda hacer una solicitud de donación al museo desde su casa.
* Que se puedan aceptar o rechazar donaciones para evitar tener un fondo de piezas idénticas
* Usar el correo electrónico como vía de comunicación con el donante
* Automatizar las entregas de piezas

**Fase 2: Gestión de las piezas entregadas**

El profesor que recoja la pieza entregada de manos del donante o de conserjería completará la solicitud de donación pasando a ser una “donación entregada”. Para ello, deberá escribir el nombre del donante, el tipo de pieza (CPU, placa, tarjeta…), foto y dónde va a almacenarse dentro de la escuela. De esta forma, se hace una ficha preliminar de la pieza a la espera de completarse con más información.

Este proceso pasará la pieza al grupo de “donaciones sin catalogar”, a la espera de que algún usuario autorizado (profesores o becarios) examine la pieza y aporte información sobre la misma: descripción, año, fabricante, curiosidades/importancia histórica…terminada la ficha detallada de cada pieza, esta pasará a ser una “donación exponible”.

Objetivos de la fase:

* Confirmar la entrega física de la pieza donada
* Hacer una ficha de la pieza donada para darla de alta en el sistema
* Permitir a los usuarios autorizados completar la información de la ficha

**Fase 3: Gestión del fondo de catálogo**

Todas las piezas ya entregadas, independientemente de su estado actual (catalogada o no), deben poder ser buscadas y filtradas mediante la aplicación web a desarrollar. Además, debe permitirse su agrupación en tipos de piezas para poder hacer esta tarea más eficiente. Esto conlleva la creación de un sistema de listado / filtrado / búsqueda para el fondo del museo que permita la consulta avanzada de todo el material por parte de los usuarios autorizados y que sirva como punto de entrada para editar su información.

Objetivos de la fase:

* Búsqueda de piezas por nombre, descripción...
* Filtrado de piezas por tipo o estado actual
* Listado de piezas con y sin filtros
* Permitir entrar al detalle de cada pieza del listado.

**Fase 4: Exposición física**

Un diseñador gráfico hará unos carteles-modelo para la exposición de cada tipo de pieza. Las piezas ya catalogadas con la información necesaria tendrán la opción de crear un cartel físico automáticamente, mediante el uso de una copia de la plantilla correspondiente que se rellenará con la información existente en la ficha de la pieza catalogada. Este cartel, como hemos dicho, debe quedar en formato editable para poder hacer retoques finales manualmente si fuera necesario. Se plantea además la necesidad de generar automáticamente cartelería para todas las piezas catalogadas (filtradas o no previamente). Esta necesidad se deriva de la posibilidad de hacer exposiciones móviles de ciertos tipos de piezas o bien porque se haya actualizado el modelo de cartel por algún motivo. Por motivos de seguridad, los carteles generados no deberían sobrescribir a los que pudieran haber sido generados anteriormente, salvo que el usuario desee expresamente hacerlo.

Objetivos de la fase:

* Establecer tipos y modelos de cartel, así como los “huecos” donde residirá la información a aportada mediante las fichas.
* Que desde la ficha de cada pieza ya catalogada se permita crear automáticamente su cartel asociado.
* Que el proceso de gestión de cartelería pueda automatizarse para generar los carteles de todas las piezas catalogadas (filtradas o no previamente por criterios del a fase anterior) en una sola operación.
* Que los carteles generados queden accesibles en alguna ruta predeterminada para proceder a su edición manual si fuera necesario.

**Fase 5: Exposición virtual**

La fase final de este trabajo de fin de grado plantea el desarrollo de una página web pública que haga las veces de exposición virtual / promoción el museo, usando para ello la información del fondo del catálogo existente, aunque solo para informar a los visitantes de cada una de las piezas donadas ya catalogadas. Se plantea igualmente la necesidad de filtrado / búsqueda / listado de piezas, aunque solo desde el punto de vista de un usuario visitante que no tiene capacidad para editar la información. Además, la página web a desarrollar debe seguir unos criterios de diseño gráfico, usabilidad y accesibilidad adecuados para cualquier tipo de visitante y adaptada a múltiples dispositivos. Finalmente, la generación de cartelería vista debería poder aprovecharse para esta fase.

Objetivos de la fase:

* Ofrecer a los visitantes un museo virtual de piezas catalogadas desarrollando una página web pública que use el fondo de catálogo como fuente de datos y adaptada a múltiples tipos de usuario y dispositivos.
* Visualizar y ofrecer a los visitantes la posibilidad de descargar el cartel de cada pieza.
* Aprovecharse de la información y procesos creados para fases anteriores (filtros, búsquedas, listados…) para realizar la exposición virtual.
* Promocionar el museo y la escuela.

### PSI 2.2: Organización del PSI

Esta tarea describe la organización del trabajo para llevar a cabo el trabajo de fin de grado, algo que tiene especial sentido si este trabajo se hace en grupo, colaboración o ayuda de otras personas. Se describen los participantes, su perfil profesional si fuese relevante y qué función cumple cada uno de ellos. La idea es crear equipos de trabajo y un catálogo de usuarios de la aplicación.

Además, es necesario concretar el material necesario e incluso si se va a hacer uso de algún espacio físico concreto (laboratorios, salas especiales…) para la realización del trabajo.

**EJEMPLO**

A la vista de los objetivos de cada fase del trabajo de fin de grado, se establecen los siguientes equipos de trabajo y usuarios dentro de los mismos

|  |  |  |
| --- | --- | --- |
| Usuario | Perfil | Función |
| Equipo de supervisión | | |
| Dirección de la escuela | Informático | Supervisión de todas las fases del proyecto y de la contratación del diseñador gráfico que hará los modelos de cartelería |
| Director del proyecto | Informático | Supervisión de que se han conseguido los objetivos intermedios de todas las fases |
| Equipo de desarrollo | | |
| Alumno | Informático | Creación y pruebas de los módulos software de las cinco fases descritas, salvo de la parte del diseño de carteles. En esta parte tendrá que especificar los bloques de información rellenables por programa que irán en los carteles (titulo, descripción…) |
| Equipo de pruebas y carga de datos | | |
| Personal de Administración y Servicios | No Informático | Pruebas de usabilidad y accesibilidad del sistema una vez haya prototipos funcionales de las diversas fases de este |
| Becarios | Informático | Pruebas de integración del software. Comprobación de que las funcionalidades previstas se están cumpliendo correctamente. Carga de datos / catalogación de piezas donadas. |
| Otro personal de la Universidad / usuarios seleccionados | Informático / No Informático | Pruebas de integración y usabilidad / accesibilidad del software. Comprobación de que las funcionalidades previstas se están cumpliendo correctamente. |
| Equipo de diseño | | |
| Diseñador Gráfico Contratado | No Informático | Diseño de cartelería-modelo para cada una de las posibles piezas donadas, atendiendo a criterios de accesibilidad de los carteles diseñados |

Equipos de trabajo y usuarios

El material para llevar a cabo el trabajo consistirá en una máquina de desarrollo y pruebas para el proyecto cuyas características se detallarán cuando se fije la tecnología a usar y un servidor para despliegue de la web que permita hacer pruebas a usuarios externos. Este segundo servidor debe ser aportado por la propia escuela, junto con una IP pública y nombre DNS que permita acceder a él. En cuanto a los espacios físicos necesarios, la escuela aportará una sala de exposiciones en la que hacer la exposición física de materiales. Esta sala es la antigua cocina de la guardería, que se señala en el siguiente plano obtenido del GIS de la universidad de Oviedo



Localización de la futura sala de exposiciones de la escuela

### PSI 2.3: Definición del Plan de Trabajo

En esta tarea se deben especificar todos los productos (intermedios y final) a obtener durante la realización del trabajo de fin de grado, así como su fecha prevista de obtención y entrega. Es necesario planificar las distintas actividades y estimar los tiempos requeridos para realizarlas por parte de cada usuario identificado. Por tanto, se deben **seguir las indicaciones dadas por la asignatura de “Dirección y Planificación de Proyectos Informáticos” de cuarto curso para elaborar una planificación** que detalle las actividades del trabajo, participantes, tiempos y responsables de cada una de ellas, los resultados esperados y el plan de trabajo a seguir.

Pueden usarse los siguientes enlaces para consultar las técnicas necesarias para realizar la planificación del plan de trabajo:

* Diagramas de PERT y Gannt: <https://manuel.cillero.es/doc/metrica-3/tecnicas/planificacion/>
* Estimación de tiempo de las tareas por el método MARK II: <https://manuel.cillero.es/doc/metrica-3/tecnicas/tecnicas-de-estimacion/metodo-mark-ii/>

**EJEMPLO**

Este plan de trabajo se hace en el contexto de la realización de un trabajo de fin de grado para la Escuela de Ingeniería Informática de la Universidad de Oviedo. En este contexto se debe tener en cuenta que los horarios de trabajo son flexibles por la condición de estudiante del principal autor del trabajo. Se ha planteado una planificación basada en 4 horas diarias de trabajo en el trabajo (compatible con otras actividades del autor) y posibles fechas discontinuas debido a la asistencia a clases, prácticas en empresa y evaluaciones, sobre todo al principio de la ejecución del trabajo.

Este plan de trabajo contempla las fases identificadas anteriormente y plantea un tiempo estimado para las mismas y cuáles pueden ser desarrolladas en paralelo, así como sus responsables.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Actividad | Participantes | Responsables | Resultados Esperados | Plan de trabajo |
|  |  |  |  |  |  |

Definición del plan de trabajo

## PSI 3: Estudio de la Información Relevante

El objetivo de esta actividad es recopilar y analizar todos los antecedentes generales que puedan afectar al trabajo de fin de grado. En este contexto académico nos centraremos en **conceptos teóricos** que afecten al desarrollo del proyecto o bien del **sistema informático actual del cliente** que este proyecto va a sustituir. La información obtenida en esta actividad se tendrá en cuenta en la elaboración de los requisitos. Los apartados son:

### PSI 3.1: Selección y Análisis de Antecedentes

Se selecciona y categoriza la documentación, productos, conceptos teóricos y similares que van a ser estudiadas, teniendo en cuenta el tipo y carácter del sistema a construir y la situación actual del cliente. Esta actividad también puede incluir entrevistas con usuarios o personas que puedan describir mejor la situación actual del cliente.

Una vez elegidos los sistemas a estudiar, se valoran y se extraen conclusiones que puedan servir para la elaboración del catálogo de requisitos. Esto incluye el uso de estándares, procedimientos, normativas, etc., si existen y ya están siendo usados por los productos estudiados que se consideran similares o por el software actual del cliente que se va a sustituir. Nótese que todo esto es situacional: el cliente (real o imaginario) podría no tener nada previo, o algo que sea completamente descartable. En el tema de conceptos teóricos, solo deben incluirse estos que sean muy específicos del dominio del problema y no de conocimiento general de la carrera. Por ejemplo, en un proyecto de reconocimiento de imágenes, tiene sentido explicar los algoritmos que van a usarse. Nunca explicar cosas como “lenguaje orientado a objetos”, “Java” o cosas similares.

## PSI 7: Definición de la Arquitectura Tecnológica

El objetivo de esta es definir la arquitectura tecnológica que dará soporte al software y a todos los sistemas de soporte que necesite, contemplando si es necesario varias opciones. Definir la base tecnológica (hardware, frameworks, librerías, utilidades, servicios web, servicios en nube…) etc. desde el principio es imprescindible para dar una respuesta adecuada a los requisitos y, además, la elección de alguno de estos sistemas dependerá fundamentalmente de ellos y de las necesidades del cliente.

### PSI 7.1: Identificación de las Necesidades de Infraestructura Tecnológica

**Si no existiesen alternativas tecnológicas que estudiar, esta tarea puede eliminarse**. El objetivo de esta tarea es analizar las necesidades de infraestructura tecnológica y proponer alternativas tecnológicas viables que respondan a ellas. Se deben identificar los entornos necesarios (SO, distribución, software auxiliar), el tipo de conexiones de red que habrá entre cada elemento, aspectos de disponibilidad y balanceo de carga para servicios críticos, etc.

En un escenario normal es posible que haya varias alternativas tecnológicas que permitan cumplir los requisitos. Para determinar esto deben considerarse las tecnologías que el cliente tiene en la actualidad y los estándares de diferente tipo que se deben cumplir. En caso de haber varias tecnologías, se deberían describir todas definiendo sus componentes a alto nivel y representando gráficamente un esquema de cómo se integraría el sistema a construir en ellas.

Por otro lado, se debe especificar también como gestionar la infraestructura tecnológica propuesta (qué máquinas o partes tienen privilegios de administración o modificación de la infraestructura).

### PSI 7.2: Selección de la Arquitectura Tecnológica

Una vez descritas las opciones, en esta tarea se seleccionará la alternativa de plataforma tecnológica más adecuada para cumplir con los requisitos de entre las anteriores. Esta debe ser una decisión adecuadamente motivada dada su enorme importancia, teniendo en cuenta el impacto en la empresa del cliente (y su integración con los medios existentes), los medios con los que cuenta, el coste económico y el tiempo necesario para su implementación. La decisión debe estar motivada en función de las ventajas e inconvenientes que plantea la alternativa seleccionada frente a las demás.

# Estudio de Viabilidad del Sistema

Fase de Desarrollo

EVS

## EVS 4, 5 y 6: Estudio y Valoración de Alternativas de Solución y Selección de Alternativa Final

En esta sección se describirán, una por una, todas las alternativas estudiadas. Conviene estudiar 3 o 4 alternativas importantes, salvo que por algún motivo justificado se deba incluir un número menor o mayor de las mismas. En todo caso, siempre es conveniente cuidar de que en esta sección haya un conjunto de sistemas significativo. En función de lo dicho anteriormente, cada sistema podrá dividirse en tres secciones: “Descripción”, “Ventajas” e “Inconvenientes”, aunque es posible cualquier otra división que contenga los aspectos descritos, dependiendo de qué tipo de sistemas se estudien. En un contexto académico, el estudio de estas alternativas no debe ser demasiado extenso, salvo que sea necesario. **Por ese motivo, se recomienda incluir una descripción breve (una página a lo sumo), una captura (o varias si el carácter del sistema lo aconsejase) que muestra su aspecto y un enlace a su web oficial.**

### Sistema 1

### Sistema 2

# Análisis del Sistema de Información

Fase de Desarrollo

ASI

Si se está usando como base/guía de contenidos las plantillas ARC42 de la asignatura “**Arquitectura del Software**”, la sección de análisis debería contemplar los siguientes elementos de estas plantillas por lo menos en alguna de sus subsecciones:

* Introduction and Goals
  + Requirements Overview
  + Quality Goals
  + Stakeholders
* Architecture Constraints
* System Scope and Context
  + Business Context
  + Technical Context

## ASI 1: Definición del Sistema

### Determinación del Alcance del Sistema

Se trata de describir de nuevo el sistema, pero en lugar de repetir lo que ya hemos dicho de él, tenemos que constatar en este apartado hasta donde vamos a llegar en su construcción, es decir, qué límites vamos a poner en el desarrollo estableciendo qué se va a hacer y qué se va a omitir (en general, hasta donde se va a llegar). Podemos por tanto usar todo lo que hemos dicho en descripciones anteriores para ayudar a describir el alcance del sistema. Conviene dejar claro este apartado para así delimitar la labor de análisis y diseño que vamos a hacer a continuación y evitar así no describir aspectos que se han construido o describir cosas que finalmente no van a construirse.

En el caso de que quede claro implícitamente qué se va a hacer en el sistema, esta sección se puede omitir

## ASI 2: Establecimiento de Requisitos

En adelante será necesario hacer determinados diagramas en varias secciones del documento. Además de Visio o la herramienta Linux Dia, los siguientes enlaces contienen herramientas que podrían ser útiles para hacer diagramas propios de dichas secciones:

* <https://www.planttext.com/>
* <https://yuml.me/diagram/nofunky/usecase/draw>
* <https://www.websequencediagrams.com/>
* <https://www.quickdatabasediagrams.com/>
* <https://www.lucidchart.com/pages/es>

### Obtención de los Requisitos del Sistema

El producto de esta sección se crea para su aprobación formal, es decir, que los potenciales clientes deben ver a partir de él las especificaciones completas del sistema. Además, esta sección construirá una base para solicitar cambios en los requisitos antes de avanzar más en la construcción del sistema. En esta sección tiene prioridad siempre todo lo aprendido en la asignatura “**Ingeniería de Requisitos**” de tercer curso, cuyos conocimientos deben aplicarse aquí.

Los requisitos del sistema se deben mostrar en una tabla como la que se presentará a continuación con ejemplos, ordenados por algún criterio lógico en función de a lo que se refieran. Por ejemplo, si tenemos usuarios tiene sentido agrupar todos los requisitos que tengan que ver con los usuarios.

Tampoco es necesario crear una única tabla para todos los requisitos, pueden crearse varias tablas que agrupen los requisitos que se refieran a una entidad. En general se recomienda hacerlo como se haya explicado en la asignatura mencionada.

Este apartado debe incluir también antes de la tabla de requisitos, si existe como tal, la **especificación textual** que el cliente nos proporcione sobre la aplicación, fruto de las reuniones que hayamos tenido con él o de las entrevistas que podamos haber llevado a cabo. Ha de tenerse en cuenta que esta información es la que usaremos para extraer los requisitos de la aplicación, por lo que no debe faltar sin contamos con ella.

|  |  |  |
| --- | --- | --- |
| Código | Nombre Requisito | Descripción del Requisito |
| R1.1 | Insertar Usuario | Se debe añadir un usuario al sistema una vez leídos y validados sus datos. |
| R1.2 | Leer Datos Usuario | Deben pedirse los datos completos de un usuario del sistema |
|  |  |  |

Catalogo definitivo de requisitos

Esta tabla puede usarse para todos los **requisitos funcionales** (lo que la aplicación debe hacer), pero debemos especificar también otro tipo de **requisitos no funcionales**, de los que a continuación se muestran ejemplos típicos:

* **Requisitos de Usuario**: Si exigimos al usuario algún tipo de conocimiento previo para manejar la aplicación o alguna de sus partes, debemos especificarlo aquí.
* **Requisitos Tecnológicos**: Si el programa establece que debe funcionar con una versión concreta de un determinado programa o sistema, o bien en un entorno o sistema operativo concreto debemos también hacerlo constar.
* **Requisitos de Usabilidad**: Normas de usabilidad que la aplicación debe cumplir obligatoriamente (nosotros podemos especificar requisitos adicionales a los que el cliente solicite y desarrollarlos en la sección correspondiente).
* **Requisitos de Seguridad**: Si debemos implementar algún tipo de medida de seguridad en el sistema (encriptación de datos, etc.). Conviene hacer especial énfasis a la hora de establecer estos requisitos en qué partes se van a **recoger datos de entrada** por parte de los usuarios / elementos externos fuera de nuestro control (ficheros…). Estos puntos son la principal entrada de potenciales vulnerabilidades en nuestro sistema (datos de entrada sin validación pertinente) y por tanto conviene tenerlos identificados desde el principio para luego aplicar las validaciones que sean pertinentes a la hora de construir las partes del software que los modelan, de acuerdo con lo visto en la asignatura “**Seguridad de Sistemas Informáticos**” de tercer curso.
* **Requisitos de Tiempo de Respuesta**: Si el sistema debe proporcionar una respuesta en un tiempo acotado.

### Identificación de Actores del Sistema

Un actor es algo o alguien que reside fuera del sistema y que interactúa con el mismo (actor primario) o bien es algo o alguien sobre el que el sistema actúa (actor secundario). Un actor puede ser una persona, un dispositivo, otro sistema o un subsistema.

Los actores representan los roles que tienen entidades ajenas al sistema pero que se relacionan con el mismo: Un mismo individuo físico puede estar representado por varios actores en el sistema si dicho individuo tiene varios roles en el mismo.  Los actores se usarán luego en la especificación de los casos de uso.

A la hora de identificar actores debe analizarse la aplicación y sus usos. Ejemplos de posibles actores son: Usuario anónimo, usuario registrado, administrador, ...

### Especificación de Casos de Uso

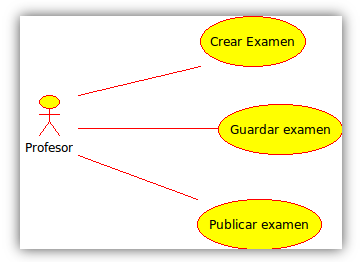
Un caso de uso es una descripción del comportamiento de un sistema cuando responde a una petición que se origina fuera del mismo (por parte de los actores del sistema). Un caso de uso describe "quién" puede hacer "qué" con el sistema. La creación de casos de uso se utiliza para capturar los requisitos funcionales del sistema y su comportamiento, haciéndose esto último a través de los escenarios que forman parte del mismo.

Un caso de uso se representa como una secuencia de pasos simples iniciadas por un actor de los identificados en la sección anterior, el cual interactúa con el sistema para llevar a cabo algún objetivo específico. A modo de ejemplo se pueden citar casos de uso como "Hacer una búsqueda", "Enviar datos de Compras", "Abandonar Operación en curso", etc. Podemos encontrar ejemplos de caso de uso adicionales y de los diagramas UML que representan gráficamente los casos de uso aquí:

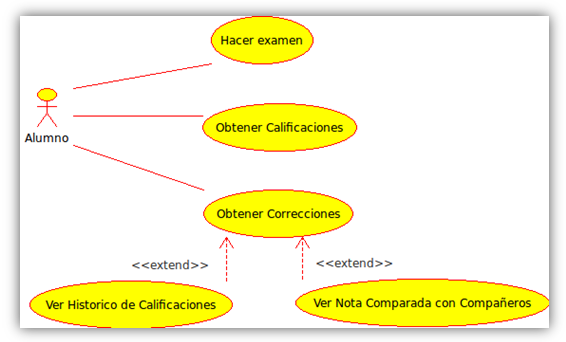
* <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>
* <http://www.visualcase.com/kbase/use_case_sample.htm>

Cada caso de uso puede generar uno o varios escenarios, que describen detalladamente cada posible vía para alcanzar el objetivo del caso de uso y que serán descritos en una sección posterior. A la hora de describir casos de uso no es necesario recurrir a terminología técnica, sino que es posible usar lenguaje cercano al usuario final. Además, un caso de uso puede derivar en más subcasos de uso si es necesario describir el comportamiento del sistema con un detalle mayor.

Esta sección debe incluir el clásico diagrama de casos de uso de la aplicación. Si el número de casos de uso fuese muy elevado, se pueden crear múltiples diagramas para que quede todo de forma más clara. A modo de ejemplo, se presentan los siguientes diagramas de un sistema para la creación y corrección de exámenes de tipo test:



Ejemplo de caso de uso 1



Ejemplo de caso de uso 2

Posteriormente, en estasección de especificación **se deberá describir también brevemente con texto el fin u objetivo de cada caso de uso** de los diagramas anteriores usando, por ejemplo, una tabla como la siguiente. En ella se especifica el nombre único del caso de uso (se suelen nombrar según el objetivo que describen) y una descripción breve de lo que intenta hacer el usuario con este caso de uso (aunque breve, no obstante, debemos asegurarnos de que es suficientemente completa y clara).

|  |
| --- |
| **Nombre del Caso de Uso** |
| Nombre único |
| **Descripción** | |
| Descripción breve de lo que intenta hacer el usuario con este caso de uso | |

A continuación, se incluyen un par de ejemplos correspondientes a los diagramas de casos de uso de ejemplo anteriores:

|  |
| --- |
| **Nombre del Caso de Uso** |
| Crear examen |
| **Descripción** | |
| El profesor creará un examen para que los alumnos lo hagan. El programa mostrará una pantalla que permitirá al profesor introducir las preguntas y ordenarlas, además de ver cómo va quedando el examen a medida que las introduce. Los exámenes que se pueden crear serán de uno de los tipos siguiente:   * Tipo test * Respuestas cortas * Emparejamiento | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Hacer examen |
| **Descripción** | |
| El programa cargará y mostrará el examen al alumno y dará las indicaciones oportunas para su realización. Anteriormente, el alumno podrá elegir uno de los exámenes disponibles que la aplicación le ofrecerá. | |

Podemos encontrar más información en:

* <http://en.wikipedia.org/wiki/Use_case>
* <http://www.gatherspace.com/static/use_case_example.html>
* <http://www.visualcase.com/tutorials/use-case-diagram.htm>
* <http://www.wilsonmar.com/1usecase.htm>

## ASI 3: Identificación de Subsistemas de Análisis

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

### Descripción de los Subsistemas

En esta sección debemos enumerar todos los subsistemas que identifiquemos inicialmente en la aplicación. Los subsistemas son agrupaciones de paquetes y clases que tienen un objetivo propósito común. Ejemplos de subsistemas pueden ser todas las clases que manejen la base de datos (subsistema “base de datos”), clases que agrupen un conjunto de servicios relacionados, clases del cliente de esos servicios, etc.

### Descripción de los Interfaces entre Subsistemas

Una vez identificados los subsistemas, debemos también describir cómo será la comunicación entre los mismos. Ejemplos de ello son por ejemplo el uso de un protocolo (IP, SOAP) o una API o Interfaz de operaciones. En general conviene destacar si estos subsistemas se comunicarán localmente (dentro de la propia máquina) o por red.

## ASI 4: Análisis de los Casos de Uso

En esta sección y las siguientes tiene prioridad los contenidos de las asignaturas **“Arquitectura del Software”** y **“Ingeniería del Proceso Software”** de tercer curso. Además, en caso de que sea un proyecto con una parte web, deben aplicarse también los contenidos que se estimen oportuno de las asignaturas “**Software y Estándares para la Web**” y “**Sistemas Distribuidos e Internet**” de tercer curso.

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios. Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo. Los escenarios deben reflejar la forma en la que un sistema se comporta y se suelen describir en lenguaje coloquial, intentando no recurrir excesivamente a tecnicismos para poder ser entendidos por el usuario final. La intención de los mismos es definir el comportamiento deseado del software de manera que complementen a los requisitos funcionales antes descritos. Si se está usando una metodología de desarrollo ágil, los escenarios se detallan como acciones breves del usuario.

Es también frecuente incluir escenarios que describan acciones erróneas o equivocadas que el programa debe tener en cuenta, para asegurar un comportamiento adecuado y seguro en todos los posibles casos. A la hora de incluir estos escenarios en la descripción de un sistema se deben tener en cuenta aquellos casos en los que el sistema pueda tener un posible problema de seguridad o de fiabilidad. Por ejemplo, casos en los que el usuario introduzca información errónea, se produzca algún error al hacer algún cálculo o bien temas relacionados con algunos requisitos no funcionales.

Como ya se ha dicho, los escenarios se generan a partir de los casos de uso identificados. Como mínimo debe existir un escenario primario o principal que describa el flujo normal de los eventos que transcurran en el caso de uso, es decir, lo que debe ocurrir normalmente cuando este se ejecuta. Por ejemplo, para un caso de uso “Registro en el sistema”, la secuencia de pasos asociada al escenario principal del mismo “El usuario se da de alta correctamente” sería:

* El sistema muestra la pantalla de logon.
* El usuario introduce su nombre y clave de usuario
* El sistema valida la información introducida
* El usuario entra correctamente en el sistema.

Además, también existirán escenarios que describan caminos secundarios o alternativos que son variantes del principal mostrado anteriormente. Por ejemplo, para el caso de uso anterior unos posibles escenarios secundarios serían:

* **Escenario Alternativo 1**: Alta errónea porque el usuario ya existe
* **Escenario Alternativo 2**: Alta errónea porque la contraseña no cumple las especificaciones requeridas.
* **Escenario Alternativo 3**: Alta errónea porque faltan campos obligatorios en el formulario.
* etc.

Los casos de uso es frecuente que se vuelvan complejos y generen un gran nº de escenarios secundarios (pueden tener un gran nº de pasos y cada uno de ellos genera varios escenarios secundarios). Por ello, se usan diagramas de actividad o robustez para poder representar mejor esa complejidad.

Por otro lado, los escenarios alternativos no deben contemplar los errores a nivel de sistema (fichero no encontrado, error de conexión), sino que suelen incluirse en su propia sección de "Excepciones" para tratar de no repetir la misma información de errores entre escenarios. Los escenarios alternativos son más secuencias de pasos alternativas a la "normal" o “principal”, pero que luego pueden estar relacionados con la misma. Por ejemplo, el escenario alternativo anterior “Alta errónea porque faltan campos obligatorios en el formulario" finalizará en "Ir al paso 1 del escenario principal" (volver a pedir los datos de login).

Bajo esta perspectiva, el número de escenarios que caben dentro de una aplicación medianamente compleja puede ser muy elevado. Entonces, ¿dónde poner el límite? ¿Cuáles deben ser representados? El analista debe ser capaz de seleccionar aquellos escenarios que aporten información útil al diseño. Por ejemplo, en toda aplicación web cabe considerar el escenario de "fallo de conexión". Sin embargo, si no se requiere un tratamiento específico para esta circunstancia, esa información es redundante dado que más que informar, despista al diseñador. Sin embargo, si queremos que, por ejemplo, para dar de alta un cargo en una actividad de la empresa ésta deba estar abierta, y en consecuencia que esto sea controlado por el sistema en desarrollo, sí deberemos considerar el escenario "Intento de asignación de cargo a actividad cerrada" dentro del caso de uso "Asignar cargo a actividad".

Una vez vista una pequeña descripción de los conceptos involucrados en esta sección, pasaremos a describir qué elementos deberían aparecer en la documentación final para contemplar todo lo dicho. Para ello se debe describir la secuencia de pasos de la que constan los escenarios y sus alternativas, clasificándolos según el caso de uso al que correspondan de una forma similar a la que se muestra en el ejemplo siguiente. Debe además tenerse en cuenta que no hay una plantilla estándar para describir los casos de uso y sus escenarios en una documentación, sino que es frecuente adaptar su descripción al proyecto que se está describiendo. A continuación, se da un ejemplo de tabla para la descripción de los mismos, que puede adaptarse reduciéndose o ampliándose en función de las necesidades:

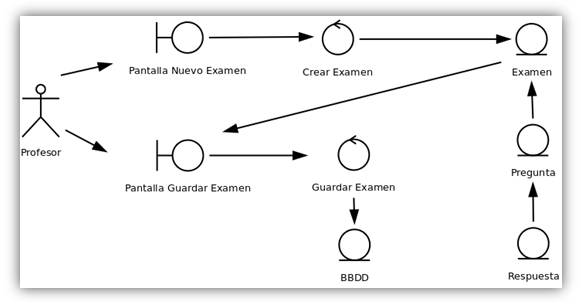
|  |  |
| --- | --- |
| Nombre del caso de uso | |
| Precondiciones | Descripción de todas las condiciones que deben cumplirse para iniciar el caso de uso. Esto quiere decir que, si el sistema no está en estado descrito por sus precondiciones, el comportamiento del caso de uso no está determinado. |
| Poscondiciones | Describe que cambios en el estado del software se producirán tras completar el caso de uso |
| Actores | Qué actores están involucrados en el caso de uso (quién lo inicia, quién lo termina) |
| Descripción | Se usará para capturar la esencia del caso de uso (su escenario principal), describiendo el contenido del mismo y sus operaciones |
| Variaciones (escenarios secundarios) | Aquí deben describirse todas las posibles variaciones contempladas sobre el escenario principal, es decir, la descripción de todos los escenarios secundarios identificados |
| Excepciones | Condiciones excepcionales o errores que puedan ocurrir en el escenario principal y/o los secundarios descritos antes |
| Notas | Cualquier aclaración necesaria que no se haya contemplado en los puntos anteriores |

Para la documentación, los casos de uso complejos o de importancia elevada es mejor documentarlos con un diagrama de actividad o robustez además del texto que hagamos siguiendo la tabla anterior. De esta forma, se debería dividir la sección por cada caso de uso, y dentro de cada una de **ellas incluir primero un diagrama con la secuencia de pasos que contempla el mismo seguido de tablas como la mostrada antes, una por cada caso de uso**. A continuación, se muestran unos ejemplos de esto para aclarar el contenido de esta sección.

### Caso de Uso 1

**NOTA**: No debemos olvidarnos de incluir una explicación del diagrama con aquello que pudiese no quedar del todo claro en el mismo.

(El siguiente ejemplo de diagrama de robustez corresponde al escenario principal del caso de uso “Crear examen” dado en un ejemplo anterior)

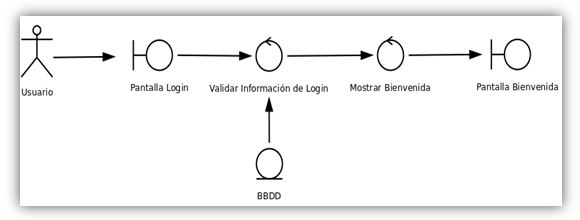


Descripción de las actividades de un escenario con un diagrama de robustez (I)

|  |  |
| --- | --- |
| Crear Examen | |
| Precondiciones | El usuario debe estar validado con rol de profesor |
| Poscondiciones | Debe existir un nuevo examen con identificador único en el sistema |
| Actores | Iniciado y terminado por el profesor |
| Descripción | El profesor:  Accederá a la pantalla de nuevo examen  Rellenará la información necesaria para confeccionar el examen  Asignará las preguntas al examen escogiendo entre las existentes  Guardará el examen  Recibirá una notificación del éxito en la operación |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El identificador del examen indicado ya existe en el sistema  Volver al paso 2 del escenario principal, manteniendo el resto de información en la pantalla  **Escenario Alternativo 2**: El usuario no encuentra la pregunta que busca entre las existentes  Dar la posibilidad al usuario de editar nuevas preguntas, conectando con el escenario principal del caso de uso “Editar Nuevas preguntas”  Actualizar la lista de preguntas disponible  Volver al paso 3 del escenario principal |
| Excepciones | **La base de datos no está disponible**: No se pueden obtener preguntas ni guardar exámenes  Notificar un error asociado al problema encontrado |
| Notas | - |

### Caso de Uso 2

(El siguiente ejemplo de diagrama de robustez corresponde al escenario principal del caso de uso “Registro en el sistema” mencionado en un ejemplo anterior)



Descripción de las actividades de un escenario con un diagrama de robustez (II)

|  |  |
| --- | --- |
| Registro en el Sistema | |
| Precondiciones | No |
| Poscondiciones | El usuario debe estar validado y con un rol asignado |
| Actores | Iniciado por un usuario de cualquier tipo de la aplicación finalizado por un usuario con el rol asociado a la información de logon introducida |
| Descripción | El sistema muestra la pantalla de logon.  El usuario introduce su nombre y clave de usuario  El sistema valida la información introducida  El usuario entra correctamente en el sistema |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: Alta errónea porque faltan campos obligatorios en el formulario  Notificar el hecho al usuario  Volver al paso 1 del escenario principal  **Escenario Alternativo 2**: Usuario y/o contraseña inválidos  Notificar el hecho al usuario, sin dar detalles de lo que falta por seguridad  Volver al paso 1 del escenario principal  **Escenario Alternativo 3**: Usuario y/o contraseña inválidos introducidos más de 5 veces seguidas  Notificar el hecho al usuario, sin dar detalles de lo que falta por seguridad  Volver al paso 1 del escenario principal, pero cambiando el contenido de la pantalla de login por un mensaje de forma que no se pueda volver a introducir información de login  Mantener esa invalidación para la IP del usuario que se conecta durante 30 minutos |
| Excepciones | **La base de datos no está disponible**: No se pueden obtener nombres ni contraseñas de usuario  Notificar un error asociado al problema encontrado |
| Notas | En caso de que la BBDD no esté disponible, se debe mostrar un error pidiéndolo al usuario que lo intente en unos minutos en la pantalla de login |

Podemos encontrar más información en: [http://en.wikipedia.org/wiki/Scenario \_(computing)](http://en.wikipedia.org/wiki/Scenario%20_(computing)).

Por último, dado que los diagramas de robustez que se pueden usar para la descripción de la secuencia de pasos a dar en un escenario son un concepto relativamente nuevo y poco conocido, se ha incluido el siguiente anexo para su mejor comprensión.

#### Anexo: Diagramas de Robustez

El propósito principal de estos diagramas es poder analizar los pasos de cada caso de uso para validar que la lógica de negocio que modela es correcta, y que la terminología usada es consistente con la de otros casos de uso que se han analizado previamente. Por tanto, pueden usarse para comprobar si nuestros casos de uso son lo suficientemente robustos para representar a los requisitos del sistema construido. Otra ventaja de hacerlos es que permite identificar objetos o responsabilidades de objetos que son necesarias para soportar la lógica modelada por cada caso de uso, pero que se llaman fuera del mismo, sirviendo como puente hacia otros diagramas como diagramas de secuencia o diagramas de clase.

En un diagrama de robustez aparecen los siguientes conceptos:

* **Actores**: Los identificados en la sesión correspondiente.
* **Elementos limítrofes o “Boundary Elements”:** Representan elementos software como pantallas, informes, páginas HTML o interfaces del sistema con los que cada actor interactúa. También se les denomina “Elementos de Interface”
* **Elementos de Control o “Control Elements”:** Sirven como unión entre elementos “Boundary” y las entidades que veremos a continuación, implementando la lógica necesaria para gestionar los elementos y sus interacciones. También se les suele denominar elementos de proceso o controladores. Es importante entender que es posible implementar controladores dentro del sistema con elementos que no sean objetos (si son muy simples, pueden representarse con métodos de una entidad o clase “Boundary” simplemente).
* **Entidades o “Entity Elements”:** Estos son los tipos de entidad que se encuentran en el modelo conceptual (“Estudiante”, “Aula”, etc.).
* **Otros casos de uso (opcional):** Dado que en muchas ocasiones unos casos de uso invocan a otros, puede ser necesario representar esto en los diagramas de robustez.
* El uso de estos diagramas será el correcto si vemos que nos reportan las siguientes ventajas:
* **Control de corrección:** Ayudan a estar seguro de que la descripción de cada caso de uso y sus escenarios es correcta y que no describe comportamientos no razonables o imposibles. En ocasiones puede ser necesario cambiar los nombres usados en la descripción del caso de uso con los nombres que aparecen en los diagramas de robustez.
* **Control de completitud:** Ayuda a asegurarnos que los casos de uso encajan con las operaciones que pretendemos hacer.
* **Identificación de objetos:** Es posible que nos hayamos olvidado de identificar algunos objetos en las partes del análisis hechas anteriormente y estos diagramas nos ayudarán a saberlo. También pueden ayudarnos a identificar discrepancias y conflictos entre nombres que hayamos asignado a las entidades anteriormente. En caso de encontrar estos fallos, debemos modificar los diagramas de manera acorde.
* **Ayuda a una fase preliminar del diseño:** Estos diagramas suelen ser más sencillos y fáciles de leer que los de secuencia.

Para una realización correcta de estos diagramas debemos tener en cuenta:

* Que las entidades que representemos en este diagrama DEBEN aparecer en el diagrama de clases de entidades del análisis hecho anteriormente.
* Que los diagramas tienen que describir los procesos de los casos de uso/escenarios que hayamos identificado. En caso de encontrar discrepancias, se debe identificar cual es el error y arreglarlo en aquel diagrama que corresponda.

Podemos encontrar más información en:

* <http://www.agilemodeling.com/artifacts/robustnessDiagram.htm>
* <http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_robustness.html> (Especialmente recomendado si este tipo de diagramas no se ha visto anteriormente, detallando además posibles errores a la hora de construir estos diagramas)

Por último, debemos recordar que si los escenarios representan operaciones muy sencillas o triviales no es necesario hacer un diagrama para los mismos. Tampoco tiene mucho sentido desarrollar muchos diagramas casi iguales; si varias operaciones funcionan prácticamente de la misma forma, bastaría con indicar que el diagrama correspondiente a la operación X es muy similar al mostrado para la operación Y, indicando en texto las diferencias. Es también importante tener herramientas que nos permitan realizar fácilmente estos diagramas. Actualmente herramientas como DIA (instalable mediante el “centro de software” de versiones recientes de Ubuntu) o Enterprise Architect son ejemplos de herramientas que lo permiten.

## ASI 5: Análisis de Clases

En la fase de análisis podemos identificar ya posibles clases del sistema, a partir de los casos de uso y subsistemas ya vistos. Estas clases no tienen por qué ser definitivas ni contener todas las operaciones y atributos que finalmente tendrán (sólo los que sean obvios según los requisitos y casos de uso), pero sirven como punto de partida para el esquema completo que se desarrollará en la fase de diseño. Por ello, estas clases no deben tener nombres de operaciones con sus parámetros exactos, sino más bien una indicación de lo que deben hacer esas operaciones. Por ejemplo, en lugar de “insertarUsuario(Usuario:u)” es mejor poner simplemente “insertar usuario”. La descripción de las clases se hará agrupándolas por el subsistema al que pertenecen.

### Diagrama de Clases

Previamente a describir las clases una por una, debemos incluir **un diagrama de clases global que muestre la relación entre todas ellas**. Se recuerda que esta es sólo la fase de análisis y no es necesario lograr un nivel de profundidad de detalle muy elevado, sino que lo que se busca es una idea aproximada (pero precisa) de cómo va a ser el sistema a construir. En este diagrama también pueden aparecer los subsistemas identificados anteriormente. A continuación, se muestra un ejemplo:

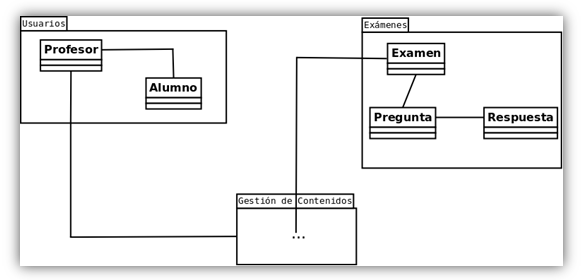


Diagrama de clases de ejemplo

### Descripción de las Clases

Las clases deberían estar organizadas por los subsistemas identificados en anteriormente, rellenando una tabla como la siguiente por cada clase del mismo:

#### Subsistema 1

|  |
| --- |
| **Nombre del Caso de Uso** |
| Hacer examen |
| **Descripción** | |
| El programa cargará y mostrará el examen al alumno y dará las indicaciones oportunas para su realización. Anteriormente, el alumno podrá elegir uno de los exámenes disponibles que la aplicación le ofrecerá. | |
| **Responsabilidades** | |
|  | |
| **Atributos Propuestos** | |
| **NombreAtributo**: Descripción de su propósito  **NombreAtributo2**: Descripción de su propósito | |
| **Métodos Propuestos** | |
| **NombreMétodo**: Descripción de lo que hace  **NombreMetodo2**: Descripción de lo que hace | |

***Subsistema 2***

|  |
| --- |
| **Nombre del Caso de Uso** |
|  |
| **Descripción** | |
|  | |
| **Responsabilidades** | |
|  | |
| **Atributos Propuestos** | |
| **NombreAtributo**: Descripción de su propósito  **NombreAtributo2**: Descripción de su propósito | |
| **Métodos Propuestos** | |
| **NombreMétodo**: Descripción de lo que hace  **NombreMetodo2**: Descripción de lo que hace | |

## ASI 8: Definición de Interfaces de Usuario

En esta sección y las siguientes tiene prioridad los contenidos de la asignatura **“Comunicación Persona Máquina”** de segundo curso. No obstante, en caso de que **se esté diseñando una aplicación móvil**, para realizar la misma de forma lo más correcta posible se deberán tener en cuenta **los principios SOLID** impartidos en la asignatura “**Software para Dispositivos Móviles**”: <https://github.com/solid/solid-spec>. Nótese que el uso de estos principios afecta a más secciones que solo la definición de interfaces de usuario.

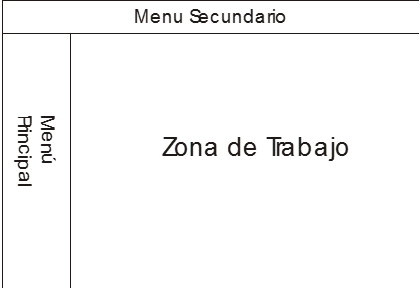
A la hora de diseñar un interfaz de usuario, debemos cumplir con las normas de comunicación persona-máquina existentes, procurando que el interfaz sea usable, permita manejar el programa de manera eficiente y que no sea propenso a provocar errores en los usuarios. Esto debe hacerse así porque los diseños obedecen al resultado de hacer un diseñado centrado en el usuario, que simplemente nos lleva a simular en la pantalla el trabajo que realiza sobre una mesa o, en general, su entorno de trabajo existente hasta el momento. Un enlace que puede ser de ayuda a la hora de tomar determinadas decisiones a la hora de construir el interface de la aplicación es el siguiente: <http://www.ambysoft.com/essays/userInterfaceDesign.html>.

### Descripción de la Interfaz

En esta sección debemos crear la especificación de las interfaces entre el usuario y el sistema a construir, incluyendo todos los diferentes tipos de pantallas que van a existir, los cuadros de diálogo o los informes que le proporcionarán al usuario.

En este apartado también es importante identificar posibles grupos de usuarios para así aplicar las pantallas a dichos grupos, así como detallar otros aspectos, como lo que vamos a incluir en las pantallas para cumplir con normas de accesibilidad y usabilidad.

Para los distintos tipos de pantallas, una opción es empezar por un esquema que muestre la disposición de estas, que permita identificar donde irá cada elemento y las diferentes zonas de trabajo. Se muestra un ejemplo con este dibujo, aunque lo más normal es usar un programa de wireframing como Balsamiq Mockups (<https://balsamiq.com/wireframes/>) o similar para hacer este tipo de esquemas de pantalla:



Boceto de una interfaz

### Definición del aspecto de la interfaz

**Otra posible opción es diseñar ya las pantallas definitivas sin funcionalidad**, solo para ver como quedarán en el producto final (es decir, crear un prototipo), lo que tiene la ventaja de poder enseñarle al cliente el aspecto de la aplicación desde un primer momento sin necesidad de mostrar los esquemas previos (esa sección podría omitirse en este caso).

Esta sección debe mostrar ya la interfaz definitiva de la aplicación (que evidentemente deberá ser una evolución del diseño mostrado en el apartado anterior) y las diferentes partes de las que consta, suponiendo que ya ha sido examinado, validado y aceptado por el cliente. Aunque luego durante el resto de la construcción del software se hagan retoques a la interfaz que se reflejen aquí si así fuese aconsejable, conviene tener claro el aspecto aproximadamente final del producto antes de construir su funcionalidad concreta, para lograr una mejor comprensión de las tareas a realizar y que el cliente sepa desde un primer momento lo que va a obtener, su aspecto y cómo se comporta (subsecciones siguientes).

Como es más que probable que todas las pantallas tengan elementos de interfaz comunes, esta es la sección donde se va a hablar de cada uno de esos elementos, su propósito y su función (barra de menús, barras de estado, etc.).

### Descripción del Comportamiento de la Interfaz

En este apartado debemos especificar cosas como los convenios que vamos a crear para validar la entrada de datos de la aplicación, los mensajes de error que mostraremos y el tipo de ayuda que vamos a proporcionar al usuario.

### Diagrama de Navegabilidad

En esta sección incluiremos un diagrama que muestre la navegación que habrá entre las pantallas del programa y su relación con las computaciones que tienen lugar en las mismas. Debemos mostrar solo las transiciones entre pantallas y no el contenido de cada pantalla en sí, ya que este diagrama tiene simbología especial para todos sus elementos. Podemos encontrar más información en:

* [www.agilemodeling.com/artifacts/uiFlowDiagram.htm](http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm).

## ASI 10: Especificación del Plan de Pruebas

En esta sección crearemos y diseñaremos el plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos ya en la fase de implementación. Siempre tendrá prioridad sobre sus contenidos lo impartido en la asignatura “**Calidad, Validación y Verificación del Software**” de cuarto curso.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Se contemplarán hasta cinco tipos de pruebas:

* **Pruebas Unitarias**: Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, o en este caso una clase individual que cumple con una función concreta. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. A partir de los casos de uso, los escenarios y clases vistos anteriormente, debemos desarrollar pruebas unitarias [Campo] que consideremos necesarias y especificar los resultados que se espera encontrar una vez ejecutada la operación sobre cada una de ellas. Es conveniente tabular estas pruebas para su aplicación posterior.
* **Pruebas de Integración**: Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, verificando que éstos funcionan correctamente cuando estos son ensamblados para cumplir con una función concreta. Para ello, cada escenario debe probarse con el mayor número de entradas posibles (y relevantes) que sea posible, incluyéndose entradas con datos correctos y con datos incorrectos para probar que el sistema reacciona correctamente ante errores de los usuarios. Para elaborar estas pruebas debemos tener en cuenta las características de la aplicación.
* **Pruebas del sistema**: Las pruebas del sistema son pruebas de integración del sistema construido completo, que permiten probar el conjunto de todo el sistema y que sus relaciones con otros sistemas que necesite son correctas, verificando así que todas sus especificaciones funcionales y técnicas se cumplen.
* **Pruebas de Usabilidad**: Este tipo de pruebas determinan la satisfacción del cliente con el producto final. Podemos especificar aquí cuales de estos aspectos son los más importantes en la aplicación a crear y establecer unas pautas generales por las cuales queremos medir en qué medida hemos conseguido estos aspectos. No se trata de diseñar ya los mecanismos de prueba de estos aspectos, ya que eso se hará posteriormente.
* **Pruebas de Código**: Para determinar la existencia de código muerto, cobertura de las pruebas y otros análisis automáticos del mismo. Se recomienda hablar si incluir este tipo de pruebas o no con el director del proyecto.

Para elaborar las pruebas debemos desarrollarlas a partir de los casos de uso y escenarios antes descritos, empleando tablas como las que se muestran a continuación. Para elaborarlas debemos tener en cuenta el escenario principal del caso de uso y sus posibles alternativas y excepciones. Estas tablas sirven como ejemplo para pruebas de integración o del sistema. Si resultase más sencillo, puede hacerse con pequeñas tablas independientes para cada caso. A continuación, se muestra un pequeño ejemplo para un caso de uso:

|  |  |
| --- | --- |
| Caso de Uso 1: Añadir Usuario | |
| Prueba | **Resultado Esperado** |
| Añadir un usuario no existente | El sistema posee un usuario más |
| Prueba | **Resultado Esperado** |
| Añadir un usuario que ya existe | El sistema no posee un usuario más y se muestra un dialogo notificándolo |
| Prueba | **Resultado Esperado** |
| Cancelar la Operación | El sistema permanece sin cambios. |

# Diseño del Sistema de Información

Fase de Desarrollo

DSI

Si se está usando como base/guía de contenidos las plantillas ARC42 de la asignatura “**Arquitectura del Software**”, la sección de diseño debería contemplar los siguientes elementos de estas plantillas por lo menos en alguna de sus subsecciones:

* Solution Strategy
  + Technological decisions and reaching the quality goals
  + Design Patterns
  + Organizational decisions
* Runtime View
* Deployment View
* Cross-cutting Concepts
* Design Decisions

## DSI 3: Diseño de Casos de Uso Reales

En esta sección y las siguientes tiene prioridad los contenidos de las asignaturas **“Arquitectura del Software”, “Ingeniería del Proceso Software”** y “**Diseño del Software**” de tercer curso. Además, en caso de que sea un proyecto con una parte web, deben aplicarse también los contenidos que se estimen oportuno de las asignaturas “**Software y Estándares para la Web**” y “**Sistemas Distribuidos e Internet**” de tercer curso.

Esta sección se usará, entre otras cosas, para evolucionar y detallar los diagramas de robustez que hemos desarrollado en el análisis usando diagramas de interacción y de estados. La estructura a seguir es la de incluir el diagrama en sí (dibujo) y luego hacer una lista explicando cada uno de los pasos existentes en dicho diagrama que lo requieran. Los diagramas deben incluir nombres de clases, métodos y parámetros “reales”, con la intención de que puedan trasladarse directamente a la implementación del sistema (el objetivo de un buen diseño). Por este motivo no debemos escatimar en detalles a la hora de desarrollarlos, ya que se supone que de estos diagramas podremos extraer directamente la implementación de la aplicación.

Cuando se menciona que los diagramas del diseño son una evolución de los del análisis significa que podemos aprovechar el trabajo ya realizado para su creación, adaptando la información en los diagramas correspondientes del análisis para desarrollar los nuevos diagramas (se debe lograr coherencia entre las entidades desarrolladas en ambas fases).

En cuanto al número de diagramas a contemplar, al igual que se dijo anteriormente no es necesario incluir en el diseño todos y cada uno de los posibles diagramas, sino que podremos ahorrar extender aquellos en los que:

* Las operaciones sean muy simples (en cuyo caso se puede optar por no decir nada al respecto o colocar simplemente un texto explicativo).
* La operación sea muy similar o idéntica a una que ya está desarrollada en esta sección. En este caso lo mejor es poner en el diagrama original a que otros casos se aplica o representa y no repetirlo.

### Caso de Uso 1.1

#### Diagramas de Interacción (Comunicación y Secuencia)

Diagramas de secuencia de diseño correspondiente al escenario principal del caso de uso y a aquellos alternativos que merezcan ser desarrollados (estableciendo un paralelismo con la nomenclatura usada en el análisis). Podemos encontrar más información en:  <http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_interaction.html>

En el caso de que creamos necesario mostrar el comportamiento de varios objetos que colaboran para la consecución de un determinado fin común, debemos crear un diagrama de comunicación de objetos que represente este comportamiento. Podemos encontrar más información acerca de los mismos en esta dirección: <http://www.agilemodeling.com/artifacts/communicationDiagram.htm>

#### Diagramas de Estados de las Clases

En caso necesario, si es conveniente dar más detalles de cómo se comporta un determinado escenario, podemos incluir diagramas de estados de las clases involucradas en el mismo para indicar las distintas fases por las que una clase pasará durante el mismo. Podemos encontrar más información sobre estos diagramas en:  <http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm>

#### Diagramas de Actividades

Si hay alguna operación o funcionalidad dentro del sistema que merezca la pena destacar se documentará mediante un diagrama de actividades. Puede haber tantos como consideréis necesarios. Más información acerca de estos diagramas se puede encontrar en estos enlaces:

* [http://dn.codegear.com/article/31863#activity-diagrams](http://dn.codegear.com/article/31863)
* <http://www.agilemodeling.com/artifacts/activityDiagram.htm>

### Caso de Uso 1.2

Misma estructura anterior, etc.

## DSI 4: Diseño de Clases

En esta sección representaremos diagramas que muestren los paquetes (y la relación existente entre ellos) y las clases que formarán parte de la implementación final del sistema (incluyendo también las relaciones existentes entre ellas), explicando todo aquel dato acerca de la utilidad de los mismos y justificando todo aquello que consideremos necesario.

* Debemos extraer la información de métodos y atributos a incluir en cada clase (así como las relaciones existentes entre ellas) de los diagramas que hemos desarrollado en el punto anterior.
* También debemos mostrar las clases adicionales que pueden ser necesarias en función de lo que hemos desarrollado anteriormente.
* Debemos mostrar todas las asociaciones y agregaciones (en general, relaciones) que necesitemos, en función de dichos diagramas.
* El conjunto de atributos de las clases debemos crearlo según los métodos, relaciones y en general de las necesidades de cada clase del sistema.
* La jerarquía de clases debemos pensarla de acuerdo a las necesidades de cada subsistema y aplicación.
* Debemos aclarar todos aquellos aspectos que no queden claros en el diagrama, usando notas.
* En general, debemos tener en cuenta que el diseño es una evolución del análisis, por lo que las clases que incluyamos en el diseño deben corresponderse razonablemente con todo el trabajo hecho en esa parte del documento.

### Diagrama de Clases

Es necesario mostrar por lo menos un diagrama global de clases. Si el diagrama de clases global fuese muy grande, podemos dividir el diagrama en varios diagramas más pequeños según la estructura de la aplicación, mostrando el principal abreviado tal y como se explica en los diagramas de paquetes. Otra opción es hacer que la hoja que tenga el diagrama se imprima en otro formato que nos proporcione más espacio (en horizontal, A3…).

## DSI 5: Diseño de la Arquitectura de Módulos del Sistema

### Diagramas de Paquetes

Aunque el concepto de paquete se puede aplicar a varios elementos del modelo, lo más típico en un TFG es agrupar clases con ellos. Los paquetes agruparán clases que estén relacionadas con una determinada funcionalidad y este diagrama debe mostrar también las relaciones existentes entre dichos paquetes (por ejemplo, dos paquetes estarán relacionados si algunas de sus clases se usan entre ellas o se envían mensajes). En esencia se trata de mostrar la organización lógica de la aplicación (en contraposición con la organización física, que aparecerá en los diagramas posteriores), presentando como se agrupan las clases de dicha aplicación en paquetes y la relación existente entre los mismos. Más información sobre estos diagramas se puede encontrar en:

* <http://www.agilemodeling.com/artifacts/packageDiagram.htm>
* <http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_packagediagram.html>

Los diagramas de paquetes son especialmente útiles cuando el diagrama de clases de todo el sistema es demasiado grande para visualizarse correctamente. Podemos entonces hacer un diagrama de paquetes (cada paquete contendrá N clases del sistema) y luego detallar individualmente cada uno de ellos donde corresponda.

#### Paquete 1

Descripción de los paquetes del diagrama (que tipo de elementos contiene, para qué sirven los mismos…).

#### Paquete 2

### Diagramas de Componentes

Los diagramas de componentes muestran los diferentes componentes de un sistema y sus dependencias. Un componente representa un módulo de código físico. Muchas veces se suele identificar un componente con un paquete, pero no siempre es así ya que los componentes representan el empaquetado físico del código. Por tanto, una misma clase puede estar presente en varios componentes, pero definida en un único paquete.

Es también una práctica común el incluir estos diagramas dentro de los de despliegue (que veremos a continuación) para que la distribución física de las distintas partes de la aplicación (y los componentes que las forman) en las distintas máquinas disponibles quede más clara.

Para tener más información sobre este tema, se pueden consultar los siguientes enlaces:

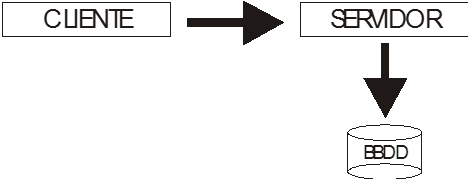
* <http://www.agilemodeling.com/artifacts/componentDiagram.htm>
* <http://www.visual-paradigm.com/VPGallery/diagrams/Component.html>
* <http://www.ibm.com/developerworks/rational/library/dec04/bell/>

 Si se integra con el diagrama siguiente (el de despliegue), en las descripciones que hagamos de los elementos debemos describir también los componentes. En caso contrario, describiremos los componentes en esta sección.

### Diagramas de Despliegue

El sistema creado puede estar compuesto por varios procesos software y máquinas que colaboran para llevar a cabo la tarea encomendada, y esta sección es la indicada para representar estos procesos y máquinas, así como la relación existente entre ellos. Debemos ofrecer un diagrama (un ejemplo extremadamente simple de cómo podemos hacerlo se muestra a continuación) y posteriormente una descripción de que es cada parte de la aplicación y su función (relacionándolo con las clases y paquetes vistos en el análisis).

En estos casos, si existen varias máquinas involucradas (físicas o virtuales) que ejecutan diversos procesos / software, se debe indicar que proceso(s) ejecuta cada una y que quede claro cuales están desplegados en la misma máquina y cuales en máquinas remotas, así como si la vinculación entre ellas se hace en red local o bien a través de internet / servicios de nube. En caso de que existan, deben indicarse la presencia de elementos como proxys, firewalls, VPNs, routers, servidores DNS, etc. En la arquitectura, de acuerdo con lo visto en la asignatura “**Administración de Sistemas y Redes**” de tercer curso.



Ejemplo simple de arquitectura del sistema

Aunque el nivel de detalle puede variar en función del sistema diseñado, los elementos mostrados deben dejar clara la distribución de la aplicación. Podemos encontrar más información al respecto en los siguientes enlaces:

* <http://www.agilemodeling.com/artifacts/deploymentDiagram.htm>
* <http://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html>

## DSI 6: Diseño Físico de Datos

Esta sección es la indicada para describir todo lo relativo al sistema de gestión de bases de datos que vamos a usar en nuestra aplicación (si usamos alguno). Esto incluye hablar del sistema en sí, la versión utilizada, las clases empleadas al usarlo, como se integra en el sistema y el diagrama E-R de las tablas y relaciones creadas, todo ello en los apartados que se muestran a continuación. No obstante, en esta sección deben aplicarse los contenidos relacionados que se han importado en la asignatura “**Bases de Datos**” de segundo curso y “**Repositorios de Información**” de tercer curso.

### Descripción del SGBD Usado

(texto)

### Integración del SGBD en Nuestro Sistema

 (texto)

### Diagrama E-R

 (texto)

## DSI 9: Diseño de la Migración y Carga Inicial de Datos

Esto solo es necesario si se parte de un sistema anterior y parte de los requisitos es migrar los datos del mismo al nuevo.

## DSI 10: Especificación Técnica del Plan de Pruebas

El proceso de pruebas se extiende durante todo el proceso de construcción del software y por tanto en esta sección debemos describir cómo vamos a aplicar las pruebas diseñadas y especificar más otro tipo de pruebas que vamos a realizar al software. Siempre tendrá prioridad sobre los contenidos de esta sección lo impartido en la asignatura “**Calidad, Validación y Verificación del Software**” de cuarto curso.

No debemos olvidarnos mencionar entonces bajo que máquina (incluyendo sistema operativo y entorno de desarrollo) se realizan las pruebas, y en qué orden se realizan las diferentes pruebas contempladas, entre otros aspectos.

**NOTA**: Es perfectamente posible que, aunque un sistema conste de múltiples procesos independientes que colaboran entre sí (por ejemplo, un cliente y un servidor), las pruebas se hagan en sólo una máquina (Ej.: cliente y servidor que se conectan a localhost).

### Pruebas Unitarias

Esta sección contendrá la descripción de la aplicación de las pruebas unitarias que hemos descrito anteriormente (qué datos se van a introducir finalmente sobre qué clases, ahora que conocemos finalmente todas las clases que se van a implementar gracias al diseño), indicando cuando las vamos a realizar y que medidas tomaremos en caso de encontrar fallos. También debemos hablar de si hemos dividido las pruebas entre los diferentes subsistemas de alguna forma.

En este apartado debemos también describir las posibles **pruebas de regresión** que queramos implementar. Estas pruebas de regresión se deben automatizar para poder pasarlas periódicamente cada vez que hagamos cambios importantes en el sistema.

**Se recomienda encarecidamente el empleo de una herramienta de automatización de estas pruebas** unitarias para facilitar mucho esta labor. Posibles herramientas son JUnit (Java) o NUnit. Realmente esta sería la forma más adecuada de afrontar esta tarea.

### Pruebas de Integración y del Sistema

Las pruebas funcionales y del sistema que hemos especificado en el análisis deben aplicarse para garantizar que el sistema funciona correctamente. Debemos describir así cómo y cuándo vamos a aplicar este tipo de casos de prueba dentro del sistema, así como las entradas y salidas de estas pruebas. No debemos olvidar que la aplicación de las pruebas y qué ocurre cuando se pasan irá en una sección posterior.

Debemos recordar pues que, tanto en esta sección como en la anterior, ya tenemos las pruebas diseñadas en la fase de análisis, y que ahora se trataría de describir cómo y cuándo las vamos a aplicar, qué datos vamos a introducir en cada caso y qué vamos a hacer en caso de acierto o fallo de las pruebas. No se trata de repetir lo mismo de nuevo.

A modo de guía, debemos pensar que lo que diseñemos aquí debe ser directamente aplicable sobre el código de la aplicación.

### Pruebas de Usabilidad y Accesibilidad

En esta sección deben aplicarse los contenidos relacionados que se han importado en la asignatura “**Comunicación Persona Máquina**” de segundo curso. Este tipo de pruebas determinan la satisfacción del cliente con el producto final y por tanto debemos darle mucha importancia a las mismas, no dejándolas de lado en ningún caso.

Las pruebas de usabilidad tratan de evaluar la aplicación en su funcionamiento real. Debemos realizar un conjunto de pruebas para verificar que las distintas partes del programa se pueden usar adecuadamente. Los objetivos son:

* Mejorar la calidad de la interacción de los usuarios con nuestra aplicación.
* Reducir el tiempo necesario para hacer las distintas tareas en la aplicación y de esta forma aumentar la productividad.
* Debemos tener los siguientes elementos en cuenta, para describirlos completamente en esta sección antes de hacer las pruebas en sí:
* **Usuarios**: Distintos grupos de usuarios de la aplicación sobre los que vamos a realizar las pruebas. Cada usuario tendrá un perfil de uso distinto y debemos tenerlo en cuenta a la hora de hacer las pruebas. Por lo general es suficiente con efectuar las pruebas con un número reducido de usuarios (de 3 a 5). También debemos dejar bien claro, en caso de que los usuarios lo deseen, la profesión y la edad de estos.
* **Lugar de realización**: En un laboratorio o bien en la propia casa u oficina del cliente.
* **Metodología**: Describir que vamos a hacer en estas pruebas (el conjunto de pasos a seguir en las mismas).

Aunque en la siguiente sección se dan detalles acerca de cómo hacer cuestionarios para efectuar estas pruebas, estos no son la única herramienta para ello. Existen otros tipos de herramientas como pruebas basadas en las opiniones dadas de forma oral por los usuarios, heurísticas, etc. Dependiendo fundamentalmente del producto y para quien este destinado. No obstante, los cuestionarios porque son una herramienta adecuada para gran parte de los casos y no exigen muchos recursos. Se recomienda consultar al director de proyecto acerca de la mejor forma de hacer este tipo de pruebas en el proyecto concreto a desarrollar.

#### Diseño de Cuestionarios

Detallamos a continuación algunas pautas a seguir en el diseño de los cuestionarios que podamos necesitar, si se estima oportuno el uso de esta herramienta. Una norma muy importante que cumplir es que él tiempo necesario para rellenar los cuestionarios no deben superar los 15 minutos.

**Cuestionario de Evaluación**

Debemos elaborar un cuestionario para que los usuarios lo hagan y determinar así distintos aspectos del mismo y de su interacción con la aplicación. Los puntos a tocar son (esto es un esquema que puede ampliarse si se desea para adaptarlo a la aplicación en sí):

* **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
* **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
* **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
* **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

**Cuestionario para el responsable de las Pruebas**

Además, debemos desarrollar un cuestionario para que el responsable de las pruebas pueda anotar distintos aspectos que observe durante la realización de las pruebas. Un posible desarrollo de todos estos aspectos se describe a continuación.

#### Actividades de las Pruebas de Usabilidad

**Preguntas de carácter general**

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

|  |
| --- |
| **¿Usa un ordenador frecuentemente?** |
| 1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca |
| **¿Qué tipo de actividades realiza con el ordenador?** |
| 1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente |
| **¿Ha usado alguna vez software como el de esta prueba?** |
| 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca |
| **¿Qué busca Vd. Principalmente en un programa?** |
| 1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias |

**Actividades guiadas**

Hacer un compendio de actividades que se puedan hacer con la aplicación para que nuestros usuarios las hagan y comenten los problemas y dificultades que según su opinión presenta la aplicación (si existiesen). Debemos recoger estas opiniones en el documento. Posibles actividades que probar son:

* Añadir un usuario a la aplicación
* Eliminar un artículo de la aplicación
* …

**Preguntas Cortas sobre la Aplicación y Observaciones**

Un posible cuestionario de preguntas cortas (a desarrollar más en cada proyecto) es el siguiente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Facilidad de Uso** | **Siempre** | **Frecuentemente** | | **Ocasionalmente** | | **Nunca** |
| *¿Sabe dónde está dentro de la aplicación?* |  |  | |  | |  |
| *¿Existe ayuda para las funciones en caso de que tenga dudas?* |  |  | |  | |  |
| *¿Le resulta sencillo el uso de la aplicación?* |  |  | |  | |  |
| **Funcionalidad** | **Siempre** | **Frecuentemente** | | **Ocasionalmente** | | **Nunca** |
| *¿Funciona cada tarea como Vd. Espera?* |  |  | |  | |  |
| *¿El tiempo de respuesta de la aplicación es muy grande?* |  |  | |  | |  |
| **Calidad del Interfaz** | | | | | | |
| **Aspectos gráficos** | **Muy Adecuado** | | **Adecuado** | **Poco Adecuado** | **Nada Adecuado** | |
| *El tipo y tamaño de letra es* |  | |  |  |  | |
| *Los iconos e imágenes usados son* |  | |  |  |  | |
| *Los colores empleados son* |  | |  |  |  | |
| **Diseño de la Interfaz** | | | **Si** | **No** | **A veces** | |
| *¿Le resulta fácil de usar?* | | |  |  |  | |
| *¿El diseño de las pantallas es claro y atractivo?* | | |  |  |  | |
| *¿Cree que el programa está bien estructurado?* | | |  |  |  | |
| **Observaciones** | | | | | | |
| Cualquier comentario del usuario | | | | | | |

Evidentemente debemos extender este cuestionario de ejemplo para adaptarlo a nuestras necesidades.

**Cuestionario para el responsable de las Pruebas**

|  |  |
| --- | --- |
| **Aspecto Observado** | **Notas** |
| *El usuario comienza a trabajar de forma rápida por las tareas* |  |
| *Tiempo en realizar cada tarea* |  |
| *Errores leves cometidos* |  |
| *Errores graves cometidos* |  |
| *<Incluir otras cosas>* |  |
| *…* |  |
| *…* |  |

Los resultados de las pruebas se deben analizar conjuntamente para así establecer una conclusión respecto a cuatro aspectos de usabilidad:

* **Facilidad de aprendizaje**: Capacidad para aprender la funcionalidad de la aplicación desarrollada y desarrollar las tareas de manera adecuada, midiendo cuanto se tarda en hacer las distintas tareas.
* **Eficiencia**: Cuanto mejora la labor de los usuarios por usar la aplicación respecto a lo que se hacía anteriormente.
* **Errores**: Cuantos errores cometen los usuarios en las distintas tareas, lo que decrementa la usabilidad del mismo.
* **Satisfacción del usuario**: Impresión general de los usuarios al usar la aplicación.

### Pruebas de Accesibilidad

El grueso de las **pruebas de accesibilidad** será descrito en la parte de desarrollo de las pruebas. A nivel de diseño simplemente puede indicarse qué tipo de pruebas van a realizarse para el programa concreto, que normas WCAG van a seguirse u otras consideraciones generales acerca de las mismas, dentro de las posibles vías de actuación. WAI propone básicamente tres tipos de revisiones:

* **Revisión preliminar**: Identifica rápidamente problemas de accesibilidad de un sitio Web. Todos los procesos de esta revisión también tienen lugar en la siguiente revisión.
* **Evaluación de conformidad**: Permite indicar si un sitio web cumple con los estándares de accesibilidad (Ej.: WCAG). Puede encontrarse más información en: [www.w3.org/WAI/eval](http://www.w3.org/WAI/eval), aunque en esta plantilla se intentarán dar pautas para hacer una evaluación adecuada.
* **Otras evaluaciones**: WAI también proporciona otra serie de documentos para evaluar aplicaciones en contextos específicos y para implicar a usuarios discapacitados en la evaluación. En el primer caso se tratan de documentos complementarios a los anteriores que describen consideraciones para la evaluación de sitios grandes y complejos durante el proceso de desarrollo y mantenimiento, o bien para sitios existentes cuyas páginas son generadas dinámicamente.

### Pruebas de Rendimiento

El sistema desarrollado consumirá una determinada cantidad de recursos (memoria y tiempo de proceso) que debemos procurar que sean los menores posibles. Por ello, la aplicación debe al menos medirse para ver cuántos recursos consume y se debe intentar eliminar posibles cuellos de botella en el rendimiento que se puedan presentar en uno o varios subsistemas de esta. Debemos diseñar medios para hacer estas mediciones en nuestra aplicación.

Para esta tarea debemos pues medir el tiempo que tardan las operaciones más importantes que haga el sistema y comprobar si es posible mejorarlas de algún modo una vez que el sistema esté en funcionamiento. En este apartado debemos diseñar que actuaciones vamos a llevar a cabo para hacer estas pruebas y las actuaciones que vamos a llevar a cabo para mejorar el rendimiento en aquellos puntos en los que encontremos problemas. Una herramienta que podemos aplicar para este fin es la opción “Tools – View Speed Report” de las “Web Developer Extensions” de Firefox.

Una consideración a tener en cuenta es determinar qué actividades serán las más frecuentes y si su rendimiento es adecuado o no, ya que por esta vía conseguiremos una mejor optimización de la aplicación.

En el caso de que el programa deba tener algún requisito respecto al tiempo que debe tardar o a la memoria ocupada, debe tenerse especial cuidado en este aspecto.

Por último, también debemos diseñar pruebas de carga para determinar cómo responde el sistema con un alto número de usuarios o un gran volumen de datos. Para esta tarea puede ayudar una aplicación como JMeter (<http://jakarta.apache.org/jmeter/>).

# Construcción del Sistema de Información

Fase de Desarrollo

CSI

Si se está usando como base/guía de contenidos las plantillas ARC42 de la asignatura “**Arquitectura del Software**”, la sección de construcción debería contemplar los siguientes elementos de estas plantillas por lo menos en alguna de sus subsecciones:

* Quality Requirements
  + Quality Scenarios
* Risks and Technical Debts

## CSI 1: Preparación del Entorno de Generación y Construcción

En esta sección se describen una serie de aspectos relativos a la implementación del software desarrollado.

### Estándares y normas seguidos

Si no hubiera ningún estándar concreto que no fuera de uso típico, puede omitirse. No obstante, en esta sección deben incluirse aquellas normas, estándares o leyes (RGPD, LSSI…) que sean de aplicación de acuerdo con lo estudiado en la asignatura “**Aspectos sociales, legales, éticos y profesionales de la Informática**” de cuarto curso.

En este apartado podría también mencionarse el uso de material de apoyo específico, tutoriales oficiales u otras fuentes de información que hayan sido necesarias para entender la tecnología con la que se ha hecho cada parte del proyecto [9] [10]. El uso de contenidos de asignaturas específicas también puede ser mencionado aquí [11] [12].

### Lenguajes de programación

Enumerar los lenguajes usados para cada parte del sistema, si hubiera más de uno. El criterio de selección puede ser variado, por idoneidad para la tarea, por velocidad de ejecución [13], etc. Si hemos usado alguna característica del lenguaje avanzada no vista en la carrera, también convendría mencionarla en este apartado [14] [15] [16] [17].

En este apartado conviene destacar también los **procedimientos de validación de los puntos de entrada de datos del usuario / elementos externos** identificados en la fase de requisitos implantados a través de diversas tecnologías de validación ofrecidas por los lenguajes seleccionados o bien por frameworks de validación externos que nos convenga usar (en este caso pueden mencionarse en el punto siguiente). Las estrategias de validación seguidas pueden variar en función de los datos que se lean y los lenguajes usados y, si son una parte central de la aplicación a desarrollar pueden mencionarse a lo largo del análisis y diseño de la aplicación también. Esto se hace aquí con el único propósito de cumplir unos mínimos de seguridad en el producto desarrollado, aunque el TFG no se centre en este aspecto. Se puede encontrar una guía completa de qué validar y cómo validar en este enlace:<https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md>. No obstante, se recomienda seguir siempre las indicaciones dadas en la asignatura “**Seguridad de Sistemas Informáticos**” de tercer curso. Como guía para documentar qué tipo de validaciones a las entradas de datos se han realizado se puede usar esta tabla como base:

|  |  |  |  |
| --- | --- | --- | --- |
|  | SI | NO | N/A |
| Se hace validación en el cliente y en el servidor |  |  |  |
| Se han usado validadores nativos de un framework |  |  |  |
| Se han validado las entradas JSON y XML con esquemas adecuados |  |  |  |
| Se hacen conversiones de tipo adecuadas con captura estricta de excepciones a las entradas leídas como string |  |  |  |
| Se comprueba el rango de los valores de entrada numéricos |  |  |  |
| Se comprueba la longitud mínima y máxima para strings |  |  |  |
| Se comprueba el conjunto de valores admitidos para entradas que solo puedan poseer un nº finito de valores de forma estricta |  |  |  |
| Se usan expresiones regulares para validar cualquier otro tipo de dato estructurado que no encaje en las categorías anteriores |  |  |  |
| Se usan listas blancas de entradas admitidas, salvo para bloqueo de clientes conocidos |  |  |  |
| Si hay entradas de texto libre Unicode, se normalizan a una codificación común |  |  |  |
| Se usan librerías de codificación de entradas anti XSS |  |  |  |
| Solo se permiten insertar datos en lugares considerados como seguros, usando codificación de datos adecuada a cada lugar |  |  |  |
| Si se admiten entradas en formato HTML, se usa una librería para “limpiar” su contenido |  |  |  |
| Se usa el flag HTTPOnly |  |  |  |
| Se implementa CSP |  |  |  |
| Se usa la cabecera X-XSS-Protection |  |  |  |
| Se usan frameworks de desarrollo modernos y actualizados |  |  |  |
| Se toman medidas para validar los archivos que el usuario pueda subir |  |  |  |
| Se toman medidas para validar las direcciones de e-mail que vayan a usarse |  |  |  |

Medidas de seguridad tomadas en el código de la aplicación

### Herramientas y programas usados para el desarrollo

Entornos de desarrollo, SGBD, frameworks, etc. Incluyendo versiones completas y enlaces a páginas oficiales de descarga o venta.

## CSI 2: Generación del Código de los Componentes y Procedimientos

En esta sección debemos describir **las clases más relevantes** de nuestra aplicación de manera que se detallen todos sus atributos y métodos, así como su descripción y función exacta, tal y como aparecen en el código de la aplicación. Para hacerlo lo mejor posible debemos elaborar esta descripción una vez cerrado el desarrollo de la aplicación en sí, para evitar problemas de versiones por revisiones al código que surjan durante el desarrollo. Para ello se recomienda emplear una herramienta que vuelque las clases y sus comentarios a un formato que podamos usar en este documento y que contemple todos los aspectos vistos antes (es decir, que el resultado sea similar a la tabla mostrada más abajo). Herramientas como Javadoc o similares son adecuadas para este fin.

Debe tenerse en cuenta que aquí solo se incluirán las clases más importantes, para el resto pueden referenciarse los anexos o repositorios tipo GitHub que se hayan usado para el desarrollo…

## CSI 3: Ejecución de las Pruebas Unitarias

Describir aquí el resultado de las pruebas unitarias que ya hemos diseñado y descrito anteriormente, así como los resultados obtenidos y las actuaciones que hemos llevado a cabo ante los errores y problemas encontrados.  Siempre tendrá prioridad sobre los contenidos de esta sección lo impartido en la asignatura “**Calidad, Validación y Verificación del Software**” de cuarto curso.

## CSI 4: Ejecución de las Pruebas de Integración

Ejecutamos las pruebas funcionales ya diseñadas anteriormente y anotamos el resultado que obtenemos, comparándolo con el que especificamos anteriormente. Podemos hacerlo a partir de una tabla modificada de la anterior como ésta (si es más sencillo, puede hacerse con pequeñas tablas independientes para cada caso). Siempre tendrá prioridad sobre los contenidos de esta sección lo impartido en la asignatura “**Calidad, Validación y Verificación del Software**” de cuarto curso:

|  |  |
| --- | --- |
| ***Caso de Uso 1.1: Añadir Usuario*** | |
| **Prueba** | **Resultado Esperado** |
| Añadir un usuario no existente | El sistema posee un usuario más |
|  | **Resultado Obtenido** |
|  | El sistema efectivamente posee un usuario más |
| **Prueba** | **Resultado Esperado** |
| Añadir un usuario que ya existe | El sistema no posee un usuario más y se muestra un dialogo notificándolo |
|  | **Resultado Obtenido** |
|  |  |
| **Prueba** | **Resultado Esperado** |
| Cancelar la Operación | El sistema permanece sin cambios. |
|  | **Resultado Obtenido** |
|  |  |

## CSI 5: Ejecución de las Pruebas del Sistema

### Prueba de Usabilidad

A partir de los cuestionarios que se diseñaron anteriormente y de los procedimientos explicados, mostramos aquí el resultado de aplicarlos sobre todos los usuarios que hayamos empleado para estas pruebas.

Es muy importante a su vez indicar los cambios producidos en la interfaz a partir de las sugerencias recogidas por los usuarios. Una forma adecuada de representar esto sería poner la pantalla inicial y poner la pantalla después de los cambios efectuados.

Además de los cuestionarios diseñados anteriormente, podemos pasar esta guía de usabilidad desarrollada por Yusef Hassan Montero [18], aunque siempre deben usarse los contenidos impartido en la asignatura “**Comunicación Persona Máquina**” de segundo curso:

|  |  |
| --- | --- |
| **Criterios** | **¿Cumplido?** |
| ***Generales*** | |
| **¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?** | **SI** |
| **¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?** |  |
| **¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web?** El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio. |  |
| **¿La estructura general del sitio web está orientada al usuario?** Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario. |  |
| **¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web?** Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales. |  |
| **¿Es coherente el diseño general del sitio web?** Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación. |  |
| **¿Es reconocible el diseño general del sitio web**? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar. |  |
| **¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza**? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario. |  |
| ***Identidad e Información*** | |
| **¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?** |  |
| El Logotipo, **¿es significativo, identificable y suficientemente visible?** |  |
| El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?** |  |
| **¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster’...?** |  |
| **¿Se proporciona mecanismos para ponerse en contacto con la empresa?** (email, teléfono, dirección postal, fax...) |  |
| **¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?** |  |
| En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?** |  |
| ***Lenguaje y Redacción*** | |
| **¿El sitio web habla el mismo lenguaje que sus usuarios?** Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario. |  |
| **¿Emplea un lenguaje claro y conciso?** |  |
| **¿Es amigable, familiar y cercano?** Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario. |  |
| **¿1 párrafo = 1 idea?** Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo. |  |
| ***Rotulado*** | |
| Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí". |  |
| **¿Usa rótulos estándar?** Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...". |  |
| **¿Usa un único sistema de organización, bien definido y claro?** No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas. |  |
| **¿Utiliza un sistema de rotulado controlado y preciso?** Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de" |  |
| El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*. |  |
| ***Estructura y Navegación*** | |
| La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada... |  |
| En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?** |  |
| En el caso de ser puramente hipertextual, **¿Están todos los clústeres de nodos comunicados?** Aquí se mide la distancia entre nodos. |  |
| **¿Los enlaces son fácilmente reconocibles como tales?** **¿Su caracterización indica su estado (visitados, activos...)?** Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad |  |
| En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento. |  |
| **¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace?** Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador... |  |
| **¿Se ha controlado que no haya enlaces que no llevan a ningún sitio?** Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo, enlaces a la "home" desde la misma página de inicio) |  |
| **¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación?** ...como *breadcrumbs*, enlaces a la página de inicio...recuerde que el logo también es recomendable que enlace con la página de inicio. |  |
| Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean. |  |
| **¿Se ha evitado la redundancia de enlaces?** |  |
| **¿Se ha controlado que no haya páginas "huérfanas"?** Páginas huérfanas: que, aun siendo enlazadas desde otras páginas, éstas no enlacen con ninguna. |  |
| ***Layout de la Página*** | |
| **¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia?** (como por ejemplo la zona central) |  |
| **¿Se ha evitado la sobrecarga informativa**? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2. |  |
| **¿Es una interfaz limpia, sin ruido visual?** |  |
| **¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?** |  |
| **¿Se hace un uso correcto del espacio visual de la página?** Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia. |  |
| **¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página?** (La jerarquía visual se utiliza para orientar al usuario) |  |
| **¿Se ha controlado la longitud de página?** Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar. |  |
| ***Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)*** | |
| **¿Se encuentra fácilmente accesible?** Es decir: directamente desde la página home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página. |  |
| **¿Es fácilmente reconocible como tal?** |  |
| **¿Permite la búsqueda avanzada?** (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera) |  |
| **¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?** |  |
| **¿La caja de texto es lo suficientemente ancha?** |  |
| **¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?** |  |
| ***Elementos Multimedia*** | |
| **¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?** |  |
| **¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario?** (prestar especial atención a usuarios de otros países y culturas) |  |
| **¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?** |  |
| **¿Se ha evitado el uso de animaciones cíclicas?** |  |
| ***Ayuda*** | |
| Si posee una sección de Ayuda, **¿Es verdaderamente necesaria?** Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección. |  |
| En enlace a la sección de Ayuda, **¿Está colocado en una zona visible y "estándar"?** La zona de la página más normal para incluir el enlace a la sección de Ayuda es la superior derecha. |  |
| **¿Se ofrece ayuda contextual en tareas complejas?** (transferencias bancarias, formularios de registro...) |  |
| Si posee *FAQs*, **¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?** |  |
| ***Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)*** | |
| **¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?** |  |
| **¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?** |  |
| **¿Existe un alto contraste entre el color de fuente y el fondo?** |  |
| **¿Incluyen las imágenes atributos 'alt' que describan su contenido?** |  |
| **¿Es compatible el sitio web con los diferentes navegadores?** **¿Se visualiza correctamente con diferentes resoluciones de pantalla?** Se debe prestar atención a: *JScript*, *CSS*, tablas, fuentes... |  |
| **¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?** |  |
| **¿Se ha controlado el peso de la página?** Se deben optimizar las imágenes, controlar el tamaño del código *JScript*... |  |
| **¿Se puede imprimir la página sin problemas?** Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible. |  |
| ***Control y Retroalimentación*** | |
| **¿Tiene el usuario todo el control sobre el interfaz?** Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos... |  |
| **¿Se informa constantemente al usuario acerca de lo que está pasando?** Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso. |  |
| **¿Se informa al usuario de lo que ha pasado?** Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente. |  |
| Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error. |  |
| **¿Posee el usuario libertad para actuar?** NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón... |  |
| **¿Se ha controlado el tiempo de respuesta?** Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos |  |
| ***Aclaraciones*** | |
| **¿Se ha evaluado adecuadamente la orientación del usuario?** (Donde estoy, como volver, que he visitado, que va a pasar)2 |  |
| **¿Se ha usado correctamente la publicidad?**3 |  |

Checklist de pruebas de Usabilidad

Tras rellenar esta tabla se deben poner a continuación una enumeración de todas aquellas aclaraciones y/o observaciones que queramos hacer acerca de la misma.

### Pruebas de Accesibilidad

A continuación, se detallan las tareas que se recomiendan hacer para asegurarnos de que el programa creado cumple con los estándares de accesibilidad. Esta sección está muy enfocada a proyectos web, pero algunas ideas pueden extrapolarse a programas de escritorio en caso necesario. Siempre deben usarse prioritariamente los contenidos impartido en la asignatura “**Comunicación Persona Máquina**” de segundo curso. Un proyecto debería hacer una evaluación de conformidad completa del mismo, pero se ha incluido un procedimiento más abreviado (revisión preliminar) para aquellos casos que por alguna razón se necesite pasar un procedimiento más rápido o menos exhaustivo. Una forma de proceder es hacer la revisión preliminar y, una vez superada arreglando todos los defectos encontrados, intentar ir a por la evaluación de conformidad completa, que permitirá reutilizar todo el estudio hecho en la revisión preliminar para hacerla. Los siguientes enlaces proporcionan información adicional sobre estos temas:

* <http://accesibilidadweb.dlsi.ua.es/?menu=principios-2.1>
* <http://accesibilidadweb.dlsi.ua.es/?menu=niveles-2.1>

#### Revisión Preliminar

Como paso previo a hacer los pasos descritos, es necesario enumerar el material utilizado para ello, como navegadores (Edge, Firefox…), lectores de pantalla o cualquier otra herramienta usada para hacer las pruebas.

**Paso 1. Selección de un grupo de páginas representativo de la aplicación**

Para pasar las pruebas de accesibilidad es necesario escoger una muestra de páginas de la aplicación representativa para así no tener que someter a toda la aplicación al proceso de revisión. Una muestra representativa está formada por:

* La página de entrada al sitio / principal / home
* Páginas con distinta organización y funcionalidad como, por ejemplo:
  + Páginas con tablas
  + Páginas con formularios
  + Páginas con diagramas o gráficos
  + Páginas con scripts o aplicaciones

Páginas con resultados generados dinámicamente (por ejemplo, con un gestor de contenidos)

**Paso 2. Examinar las páginas usando un navegador gráfico**

Esta actividad comprende las siguientes operaciones:

* Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú “Images” de la “Web Developer Extension” de Firefox de la que se habló anteriormente.
* Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
* Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con CTRL + Rueda del ratón).
* Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En Firefox podemos ir a “Herramientas – Opciones – Contenido”, aunque también podemos hacerlo cambiando la CSS de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32 y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.
* Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el scroll horizontal). Esto puede hacerse fácilmente con el plugin “Web Developer Extension” de Firefox. Este plugin, una vez instalado tiene una opción “Resize”. Mediante su sub-opción “Edit Resize Dimensions” podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas.  Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
* Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
* Probar a visualizar la página sin sus hojas de estilo CSS para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú “CSS” de la “Web Developer Extension” de Firefox de la que se habló anteriormente.
* Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web GrayBit  (<http://graybit.com/main.php>).

Usar el teclado para navegar a través de los enlaces y controles de formularios, usando Tab para desplazarse.

Herramientas útiles para comprobar estos aspectos pueden ser: Web Developer Extension para Firefox o similares para otros navegadores.

**Paso 3. Examinar las páginas usando uno o varios navegadores especializados**

Es muy necesario comprobar cómo nuestras páginas se comportan ante diferentes clases de navegadores como, por ejemplo:

* Navegadores por voz como Firevox (<http://firevox.clcworld.net/>) (gratuito) o JAWS (<http://www.freedomscientific.com/jaws-hq.asp>). También puede encontrarse una lista de herramientas similares aquí: <http://en.wikipedia.org/wiki/Comparison_of_screen_readers>
* Navegadores de texto como Lynx. Si se trabaja en Windows es posible ejecutarlo usando Cygwin o quizá el subsistema de Linux para Windows.

En estos navegadores es necesario verificar que la información transmitida por ambos tipos es similar a la mostrada en el navegador gráfico y asegurarse de que el orden en el que se transmite dicha información es coherente.

Por otro lado, también es conveniente probar la página con diferentes navegadores (IE, Firefox, Opera, Chrome…) y con distintas versiones de cada navegador. Aunque podría hacerse mediante VMware, instalar un gran nº de navegadores y versiones de los mismos es muy costoso. No obstante, una herramienta que nos proporcionará fácilmente acceso a un elevadísimo número de navegadores distintos, con múltiples versiones de cada uno y con distintos sistemas operativos, es BrowserShots (<http://browsershots.org/>). En esta página Web podremos introducir la URL de nuestro sitio (en caso de que no la tengamos disponible se puede probar con la IP de la máquina de desarrollo, asegurándose de que es accesible). Esto invocará a un servicio web que distribuirá la petición a múltiples máquinas con distintos navegadores instalados, devolviéndonos las capturas de pantalla en todos y cada uno de esos navegadores de nuestro sitio web, transcurrido un tiempo de proceso de la petición (en función de la hora del día y de la carga de trabajo del servicio puede tardar bastante). Si no cerramos la página o cancelamos la petición y tenemos paciencia, una vez terminado el proceso tendremos imágenes de nuestra web en multitud de navegadores y SO diferentes fácilmente, algo que de otra manera nos supondría un coste muy elevado. Debido al coste en tiempo de esta herramienta, se recomienda hacerlo con la versión final del interfaz de la aplicación.

**Paso 4. Utilizar herramientas automáticas de evaluación de accesibilidad**

Este último paso consiste en pasar a la muestra de páginas seleccionadas herramientas de evaluación automática de la accesibilidad como:

* TAW (<http://www.tawdis.net/taw3/cms/es>)
* HERA (<http://www.sidar.org/hera/index.php.en>)
* EvalAccess (<http://sipt07.si.ehu.es/evalaccess2/index.html>)
* WAVE (<http://wave.webaim.org/>)

Debemos tener en cuenta:

* Que hay que pasar un mínimo de dos herramientas de esta clase.
* Que sólo pueden probar algunos aspectos de la accesibilidad. Una vez eliminados todos los errores de comprobación automática de la página, debemos hacer todo lo posible por eliminar todos los de comprobación manual (de todas las herramientas pasadas).

**Paso 5. Resumen de resultados**

Finalmente, se incluirá un pequeño informe final con los siguientes aspectos:

* Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
* Indicar como se detectó cada problema.

#### Evaluación de Conformidad

La evaluación de conformidad combina la evaluación manual de la página con la evaluación semiautomática. Se suele emplear cuando se desarrolla un sitio nuevo o bien cuando se evalúa un sitio existente. En el caso de un TFG, debemos intentar pasar una evaluación de conformidad completa al sitio para considerar que realmente ha hecho un esfuerzo adecuado por lograr un nivel de accesibilidad óptimo.

**Paso 1. Determinar el alcance de la evaluación**

Este paso contempla:

* Definir y divulgar el nivel deseado de conformidad WCAG (A, AA, AAA). Se recuerda que las webs destinadas a la administración pública deben tener al menos un nivel AA, siendo este un nivel mínimo exigible en un TFG. La revisión 2.1 de las normas WCAG se puede encontrar aquí: <https://www.w3.org/TR/WCAG21/>
* Seleccionar un conjunto representativo de páginas para la evaluación manual, siguiendo el criterio expuesto en la revisión preliminar.

**Paso 2. Utilizar herramientas de evaluación automática de accesibilidad**

Este paso contempla:

* Validar los lenguajes utilizados (HTML, CSS,…) con las herramientas adecuadas que existan para ello. La “Web Developer Extension” de Firefox posee opciones para ello.
* Usar al menos dos herramientas de evaluación automática en la muestra de páginas seleccionada (ver la revisión preliminar). También se debe usar al menos 1 herramienta en la totalidad del sitio.
* Anotar y resolver los problemas encontrados.

**Paso 3. Evaluar manualmente la muestra de páginas**

Para ello debemos:

* Utilizar el checklist de puntos de control del WCAG que aparecerá tras esta explicación, según versión. Al final de esta sección se ha incluido este checklist adaptado a Word y listo para rellenarse de las WCAG 2.1. No obstante, debería usarse el de las últimas normas WCAG si se encuentra uno disponible.
* Examinar las páginas con al menos 3 navegadores gráficos en diferentes versiones y en diferentes plataformas (para lo cual puede usarse el servicio web Browsershots ya visto). Tener en cuenta especialmente estos puntos, que son una lista extendida de los que se enunciaron en la revisión preliminar. Para su comprobación podemos usar las mismas herramientas y procedimientos recomendados en dicha sección:
  + Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú “Images” de la “Web Developer Extension” de Firefox de la que se habló anteriormente.
  + Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
  + Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con CTRL + Rueda del ratón).
  + Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En Firefox podemos ir a “Herramientas – Opciones – Contenido”, aunque también podemos hacerlo cambiando la CSS de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32 y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.
  + Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el scroll horizontal). Esto puede hacerse fácilmente con el plugin “Web Developer Extension” de Firefox. Este plugin, una vez instalado tiene una opción “Resize”. Mediante su sub-opción “Edit Resize Dimensions” podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas.  Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
  + Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
  + Probar a visualizar la página sin sus hojas de estilo CSS para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú “CSS” de la “Web Developer Extension” de Firefox de la que se habló anteriormente.
* Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web GrayBit (<http://graybit.com/main.php>). En lo referente a colores, también debemos usar herramientas de verificación de color:
  + **Color Contrast Analyzer** ([http://www.visionaustralia.org.au/info.aspx ?page=628](http://www.visionaustralia.org.au/info.aspx%20?page=628)): permite verificar que los colores de fondo y texto de una imagen o página web contrastan lo suficiente para ser distinguibles por cualquier persona, según los algoritmos desarrollados por el W3C.
  + **Vischeck** (<http://www.vischeck.com/>): muestra cómo ven una página web personas con distintos tipos de daltonismo: deuteranopia, protanopia y tritanopia.
* Usar el teclado para navegar a través de los enlaces y controles de formularios, usando Tab para desplazarse.
* Desactivar scripts, applets, Flash, etc. y comprobar que la página sigue siendo navegable. Para esto es especialmente útil el plugin NoScript de Firefox
* Examinar las páginas usando uno o varios navegadores especializados: al menos uno de texto y uno de voz (ver revisión preliminar).
* Leer y evaluar el contenido de las páginas, para buscar texto no claro o incongruente. Sobre todo, debe primar que el texto de las páginas sea lo más claro posible.

**Paso 4. Elaborar el informe de conclusiones**

Finalmente se incluirá un informe de conclusiones que contemplará los siguientes aspectos:

* Resumen de los principales problemas encontrados y la solución dada a los mismos
* Resumen de los aspectos positivos que potencian la accesibilidad de la página
* Recomendación de fututas actividades para mejorar la accesibilidad de la página:
  + Reparación de barreras de accesibilidad encontradas y que no hayan podido ser solucionadas del todo o adecuadamente.
  + Ampliación de aspectos positivos de accesibilidad
  + Monitorización del sitio.

Por último, a modo de resumen se destacan aquí algunos aspectos muy importantes que no deben dejarse nunca de lado cuando se desarrolla una página, para minimizar los problemas cuando se haga una revisión de cualquier clase sobre la misma. No obstante, todo esto se contempla en el checklist del WCAG:

* Poner un texto alternativo para las imágenes
* Poner texto alternativo en los hipervínculos
* Revisar los formularios una vez creados (navegación, claridad)
* Marcos: títulos en los marcos y existencia de tags <noframes>
* Desactivar scripts y Flash y comprobar que la página no pierde funcionalidades
* Navegar sólo con el teclado y comprobar que se puede acceder a todo el sitio.
* Hacer especial hincapié en la revisión de la página de inicio (más importante).

#### Checklist del WCAG 2.1

La siguiente tabla es el checklist que el WCAG proporciona para verificar las pautas de accesibilidad de la aplicación web, pero adaptado a Word para poder ser editado fácilmente. Cada punto tiene antes un hiperenlace que va directamente a la web del WCAG para proporcionar explicaciones adicionales sobre el mismo (que pueden ir bien para arreglar los problemas detectados).

**Puntos de verificación Nivel A:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | SI | NO | N/A |
| 1.1.1 Contenido no de texto: Proporcionar alternativas de texto para contenido que no sea de texto |  |  |  |
| 1.2.1 Solo audio y solo vídeo (pregrabado): Proporcione una alternativa al contenido solo de vídeo y solo de audio |  |  |  |
| 1.2.2 Subtítulos (pregrabados): Proporcione subtítulos para vídeos con audio |  |  |  |
| 1.2.3 Descripción de audio o alternativa multimedia (pregrabado): El vídeo con audio tiene una segunda alternativa |  |  |  |
| 1.3.1 Información y relaciones: Estructura lógica |  |  |  |
| 1.3.2 Secuencia significativa: Presente el contenido en un orden significativo |  |  |  |
| 1.3.3 Características sensoriales: Utilice más de un sentido para las instrucciones |  |  |  |
| 1.4.1 Uso del color: No utilice una presentación que dependa únicamente del color |  |  |  |
| 1.4.2 Control de audio: No reproduzca audio automáticamente |  |  |  |
| 2.1.1 Teclado: Accesible solo con teclado |  |  |  |
| 2.1.2 Sin bloqueo de teclado: No impida el uso de teclado a los usuarios |  |  |  |
| 2.1.4 Accesos directos: Proporcionar un mecanismo para configurar los métodos abreviados de teclado |  |  |  |
| 2.2.1 Temporización ajustable: Los límites de tiempo tienen controles de usuario |  |  |  |
| 2.2.2 Pausa, Detener, Ocultar: Proporcionar controles de usuario para mover contenido |  |  |  |
| 2.3.1 Tres destellos o menos: Ningún contenido parpadea más de tres veces por segundo |  |  |  |
| 2.4.1 Bloques de derivación: Proporcione un enlace "Skip to Content" |  |  |  |
| 2.4.2 Página titulada: Utilice títulos de página útiles y claros |  |  |  |
| 2.4.3 Orden de enfoque: Orden lógico |  |  |  |
| 2.4.4 Propósito del enlace (en contexto): El propósito de cada enlace se desprende claramente de su contexto |  |  |  |
| 2.5.1 Gestos de puntero: todas las funciones que utilizan gestos multipunto o basados en rutas para hacer una operación se pueden hacer con un solo puntero sin gestos, a menos que sea esencial. |  |  |  |
| 2.5.2 Cancelación del puntero: Para ayudar a evitar la activación involuntaria de los controles, evite la activación no esencial del evento descendente (por ejemplo, onmousedown) al hacer clic, tocar o presionar la pantalla durante mucho tiempo. |  |  |  |
| 2.5.3 Etiqueta en Nombre: Para los componentes de la interfaz de usuario con etiquetas que incluyen texto o imágenes de texto, el nombre contiene el texto que se presenta visualmente. |  |  |  |
| 2.5.4 Motion Actuation: La funcionalidad que se activa moviendo el dispositivo (como agitar o desplazar un dispositivo móvil) o por el movimiento del usuario (como saludar a una cámara) se puede desactivar y la funcionalidad equivalente se proporciona a través de controles estándar como botones |  |  |  |
| 3.1.1 Idioma de la página: La página tiene un idioma asignado |  |  |  |
| 3.2.1 Enfoque: Los elementos no cambian cuando reciben enfoque |  |  |  |
| 3.2.2 En la entrada: Los elementos no cambian cuando reciben la entrada |  |  |  |
| 3.3.1 Identificación de errores: Identificar claramente los errores de entrada |  |  |  |
| 3.3.2 Etiquetas o Instrucciones: Etiquetar elementos y dar instrucciones |  |  |  |
| 4.1.1 Análisis: No hay errores de código importantes |  |  |  |
| 4.1.2 Nombre, Rol, Valor: Construir todos los elementos pensando en la accesibilidad |  |  |  |

Checklist Nivel A

**Puntos de verificación Nivel AA:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | SI | NO | N/A |
| 1.2.4 Subtítulos (en vivo): Los vídeos en directo tienen subtítulos |  |  |  |
| 1.2.5 Descripción de audio (pregrabada): los usuarios tienen acceso a la descripción de audio para el contenido de vídeo |  |  |  |
| 1.3.4 Orientación: El contenido no restringe su vista y operación a una sola orientación de visualización, como vertical u horizontal, a menos que una orientación de visualización específica sea esencial |  |  |  |
| 1.3.5 Identificar propósito de entrada: los campos de entrada que recopilan ciertos tipos de información de usuario tienen definido un atributo de autocompletar adecuado |  |  |  |
| 1.4.3 Contraste (mínimo): La relación de contraste entre texto y fondo es de al menos 4,5:1 |  |  |  |
| 1.4.4 Cambiar el tamaño del texto: El texto se puede redimensionar al 200% sin pérdida de contenido o función |  |  |  |
| 1.4.5 Imágenes de texto: No utilice imágenes de texto |  |  |  |
| 1.4.10 Reflujo: No se produce ninguna pérdida de contenido o funcionalidad y se evita el desplazamiento horizontal cuando el contenido se presenta con un ancho de 320 píxeles |  |  |  |
| 1.4.11 Contraste no de texto: Existe una relación de contraste de al menos 3:1 para diferenciar objetos gráficos (como iconos y componentes de gráficos o gráficos) y componentes de interfaz personalizados por el autor (como botones, controles de formulario y enfoque indicadores/esquemas) |  |  |  |
| 1.4.12 Espaciado de texto: No se produce pérdida de contenido o funcionalidad cuando el usuario adapta la altura de la línea de texto/espaciado a 1,5 veces el tamaño de la fuente, el espaciado de párrafo a 2 veces el tamaño de la fuente, el espaciado de palabras a .16 veces el tamaño de la fuente y el espaciado de letras a .12 veces el tamaño de la fuente |  |  |  |
| 1.4.13 Contenido en Hover o Focus: Cuando se presenta contenido adicional en el hover del cursor o en el focus del teclado: el contenido que aparece se puede descartar (generalmente a través de la tecla Esc) sin mover el puntero o el foco del teclado, a menos que el contenido presente un error de entrada o no oculte ni interfiera con el contenido de otra página; el puntero se puede mover al nuevo contenido sin que el contenido desaparezca; el nuevo contenido debe permanecer visible hasta que el puntero o el foco del teclado se aleje del control desencadenador, se descarte el nuevo contenido o el nuevo contenido ya no sea relevante |  |  |  |
| 2.4.5 Múltiples maneras: Ofrecer varias maneras de encontrar páginas |  |  |  |
| 2.4.6 Encabezados y etiquetas: Utilice encabezados y etiquetas transparentes |  |  |  |
| 2.4.7 Enfoque visible: Asegúrese de que el foco del teclado sea visible y claro |  |  |  |
| 3.1.2 Idioma de las piezas: Indique a los usuarios cuándo cambia el idioma de una página |  |  |  |
| 3.2.3 Navegación consistente: Utilice los menús de forma coherente |  |  |  |
| 3.2.4 Identificación consistente: Utilice iconos y botones de forma consistente |  |  |  |
| 3.3.3 Sugerencia de error: Sugerir correcciones cuando los usuarios cometen errores |  |  |  |
| 3.3.4 Prevención de errores (Legal, Financiero, Datos): Reducir el riesgo de errores de entrada de datos confidenciales |  |  |  |
| 4.1.3 Mensajes de estado: los mensajes de estado se programar considerando roles o propiedades del usuario y presentarse al mismo mediante tecnologías de asistencia sin recibir el foco. |  |  |  |

Checklist Nivel AA

**Puntos de verificación Nivel AAA:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | SI | NO | N/A |
| 1.2.6 Lenguaje de señas (pregrabado): Proporcione traducciones de lengua de signos para vídeos |  |  |  |
| 1.2.7 Descripción de audio ampliada (pregrabada): Proporcione una descripción de audio ampliada para los vídeos |  |  |  |
| 1.2.8 Alternativa de medios (pregrabado): Proporcione una alternativa de texto a los videos |  |  |  |
| 1.2.9 Solo audio (en vivo): Proporcione alternativas para el audio en directo |  |  |  |
| 1.3.6 Identificar propósito: el propósito de los componentes de interfaz de usuario, iconos y regiones se puede determinar programáticamente |  |  |  |
| 1.4.6 Contraste (mejorado): La relación de contraste entre texto y fondo es de al menos 7:1 |  |  |  |
| 1.4.7 Audio de fondo bajo o sin fondo: El audio es claro para que los oyentes escuchen |  |  |  |
| 1.4.8 Presentación visual: Ofrezca a los usuarios una gama de opciones de presentación |  |  |  |
| 1.4.9 Imágenes de texto (sin excepción): No utilice imágenes de texto |  |  |  |
| 2.1.3 Teclado (sin excepción): Solo accesible con teclado, sin excepción |  |  |  |
| 2.2.3 Sin tiempo: Sin límites de tiempo |  |  |  |
| 2.2.4 Interrupciones: No interrumpa a los usuarios |  |  |  |
| 2.2.5 Re-autenticación: Guarde los datos del usuario al volver a autenticar |  |  |  |
| 2.2.6 Tiempos de espera: Se advierte a los usuarios de la duración de cualquier inactividad del usuario que pueda causar la pérdida de datos, a menos que los datos se conserven durante más de 20 horas cuando el usuario no realice ninguna acción |  |  |  |
| 2.3.2 Tres destellos: Ningún contenido parpadea más de tres veces por segundo |  |  |  |
| 2.3.3 Animación de interacciones: La animación de movimiento activada por interacción se puede desactivar, a menos que la animación sea esencial para la funcionalidad o la información que se transmite |  |  |  |
| 2.4.8 Ubicación: Haga saber a los usuarios dónde se encuentran |  |  |  |
| 2.4.9 Propósito del enlace (solo enlace): El propósito de cada enlace se desprende claramente de su texto |  |  |  |
| 2.4.10 Encabezados de sección: Dividir el contenido con encabezados |  |  |  |
| 2.5.5 Tamaño de destino: Los objetivos en los que se puede hacer clic tienen al menos 44 por 44 píxeles de tamaño a menos que se proporcione un objetivo alternativo de ese tamaño, el objetivo esté en línea (por ejemplo, un vínculo dentro de una oración), el objetivo no se modifique por el autor (por ejemplo, una casilla de verificación predeterminada) o el tamaño del objetivo es esencial que sea pequeño para la funcionalidad |  |  |  |
| 2.5.6 Mecanismos de entrada simultánea: El contenido web no restringe el uso de las modalidades de entrada disponibles en una plataforma, excepto cuando la restricción sea esencial, necesaria para garantizar la seguridad del contenido o necesaria para respetar la configuración del usuario |  |  |  |
| 3.1.3 Palabras inusuales: Explicar cualquier palabra extraña |  |  |  |
| 3.1.4 Abreviaturas: Explicar las abreviaturas |  |  |  |
| 3.1.5 Nivel de lectura: Los usuarios con nueve años de escuela pueden leer su contenido |  |  |  |
| 3.1.6 Pronunciación: Explique las palabras que sean difíciles de pronunciar |  |  |  |
| 3.2.5 Cambio a petición: No cambie elementos en su sitio web hasta que los usuarios pregunten |  |  |  |
| 3.3.5 Ayuda: Proporcione ayuda e instrucciones detalladas |  |  |  |
| 3.3.6 Prevención de errores (Todos): Reducir el riesgo de todos los errores de entrada |  |  |  |

Checklist Nivel AAA

#### Accesibilidad con Dispositivos Móviles

En caso de que la página a analizar esté destinada a un dispositivo móvil (o tenga una parte o una versión destinada para los mismos), se proporcionan aquí enlaces a herramientas e información útil en estos casos:

* **W3C mobileOK Checker**: <http://validator.w3.org/mobile/>
* **TAW Ok Basic**: <http://validadores.tawdis.net/mobileok/es>
* **Ready.mobi**: <http://ready.mobi/launch.jsp?locale=en_EN>

Otro aspecto a tener en cuenta con estos dispositivos es que si tenemos que evaluar la página en varios de ellos puede ser muy difícil conseguir el hardware necesario.  No obstante, para esto nos pueden servir emuladores de estos, que permitan comprobar sobre un mismo PC y de forma rápida y sencilla como se ve nuestro sitio en distintos dispositivos móviles con gran fiabilidad. Podemos pues usarlos para enriquecer nuestras pruebas de usabilidad, mostrando cómo se ve nuestra aplicación en un dispositivo móvil. Ejemplos son:

* **.mobi**: <http://mtld.mobi/emulator.php>
* **The Openwave Phone Simulator**: [http://developer.openwave.com/dvl/tools\_and\_sdk/ phone\_simulator/](http://developer.openwave.com/dvl/tools_and_sdk/%20phone_simulator/)

Una vez pasadas las pruebas indicadas en estos enlaces (para las que también pueden tomarse ideas de las pruebas de accesibilidad anteriores), se debe hacer un informe similar a los de la sección anterior indicando:

* Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
* Indicar como se detectó cada problema.

## CSI 6: Elaboración de los Manuales de Usuario

En esta sección puede surgir la duda de si documentar también la instalación de productos complementarios necesarios para que el software funcione, como servidores web, frameworks, etc. Salvo que el proyecto trate de la modificación de uno de estos expresamente, no merece la pena hacerlo. Lo aconsejable es que en el manual pertinente se haga referencia a manuales de instalación de estos comprobados u oficiales, pudiéndose aprovechar los indicados en la asignatura “**Administración de Sistemas y Redes**” de tercer curso o bien en la asignatura “**Administración de Servidores Web**” [19] [20] del Máster en Ingeniería Web.

### Manual de Instalación

Elaborar un manual que contemple todos los pasos necesarios para instalar nuestro sistema, incluyendo la instalación de otras herramientas o software cualquiera (sea o no comercial) necesario para que funcione. Debemos explicarlo todo paso a paso de forma clara y acompañarlo por capturas de pantalla adecuadas, aunque teniendo en cuenta lo dicho anteriormente de referenciar manuales de instalación de productos de terceros para aquellos productos complementarios a nuestra propia aplicación. Lo que si merece la pena en este apartado es describir las medidas de seguridad tomadas sobre dichos productos complementarios una vez instalados, siguiendo las indicaciones de las asignaturas descritas. Como guía puede seguirse la siguiente tabla, extraída de un tutorial de mínimos de seguridad que os puede ser útil: <https://geekflare.com/apache-web-server-hardening-security/>. Este tutorial trata específicamente sobre el servidor Apache, pero es sencillo realizar los mismos pasos o equivalentes en otro tipo de servidores (Nginx, IIS, Tomcat…)

|  |  |  |  |
| --- | --- | --- | --- |
|  | SI | NO | N/A |
| El banner del servidor ya no informa sobre su versión o da información sobre el SO subyacente |  |  |  |
| Se ha deshabilitado el listado de directorios, salvo que sea una funcionalidad buscada (en cuyo caso solo está activa donde sea estrictamente necesario) |  |  |  |
| Se ha deshabilitado la cabecera ETag |  |  |  |
| El servidor NO se ejecuta como usuario root |  |  |  |
| El directorio de instalación del servidor tiene permisos adecuados |  |  |  |
| No se permite a los usuarios sobrescribir la configuración del servidor |  |  |  |
| Se limita al mínimo el nº de métodos HTTP permitidos |  |  |  |
| Se deshabilita el método HTTP TRACE |  |  |  |
| Se configura la cabecera X-Frame-Options para que solo se permitan IFrames desde el mismo dominio de la web |  |  |  |
| Se deshabilitan los Server-Side Includes, salvo que sean necesarios para la funcionalidad |  |  |  |
| Se habilita la cabecera X-XSS-Protection |  |  |  |
| Se deshabilita el soporte de HTTP 1.0 |  |  |  |
| El Timeout por defecto se configura a unos 60 segundos |  |  |  |
| Se usa SSL con una clave de longitud 2048 por lo menos |  |  |  |
| Se usa SSL con un algoritmo de cifrado seguro |  |  |  |
| Se deshabilita el soporte de SSLv2 y SSLv3, admitiéndose solo TLSv1.2 o cualquier versión posterior no considerada vulnerable |  |  |  |
| Se usa el WAF mod\_security y sus *core rules* |  |  |  |
| Se hace log de forma adecuada |  |  |  |
| Solo se usan los módulos estrictamente necesarios |  |  |  |

Medidas de seguridad del servidor instalado

### Manual de Ejecución

Este manual contemplará todos los pasos necesarios para el arranque de nuestro sistema, lo que es especialmente importante en caso de sistemas con clientes y servidores o distintos procesos que deban arrancarse independientemente.

Por otra parte, también debemos incluir procedimientos para parar adecuadamente la aplicación.

### Manual de Usuario

El manual de usuario es algo muy importante debido a que es el documento que servirá a los usuarios de nuestro sistema para saber cómo funciona cada una de las partes de nuestra aplicación. Debemos pues describir cómo funcionan todas las opciones de la misma, que parámetros tiene, que cosas debemos hacer para que todas las operaciones funcionen correctamente y cualquier otro aspecto que consideremos oportuno para explicar el funcionamiento del sistema.

No debemos escatimar detalles en este manual ya que es la herramienta para que los usuarios comprendan nuestro sistema. También debemos hacer el mayor uso posible de capturas de pantalla para mejorar nuestras explicaciones.

### Manual del Programador

En este manual debemos describir cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. Debemos por tanto hacer una descripción general de los distintos aspectos involucrados en la construcción del sistema que puedan ser más difíciles de entender y también describir los procedimientos necesarios para hacer ciertas ampliaciones que hayamos contemplado en el diseño del sistema (añadir nuevas entidades, nuevos atributos a entidades existentes, nuevos servicios que usen a los ya desarrollados, modificaciones en la interfaz, etc.).

## CSI 8: Construcción de los Componentes y Procedimientos de Migración y Carga Inicial de Datos

Esto solo tiene sentido si se debe trabajar en la carga de datos de un sistema anterior.

# Implantación y Aceptación del Sistema

Fase de Desarrollo

IAS

## IAS 1: Establecimiento del Plan de Implantación

Si el programa es para un cliente real, se describirá aquí el plan para que este programa se incorpore e integre con sus sistemas informáticos actuales.

## IAS 4: Carga de Datos al Entorno de Operación

En caso de que fuera necesario, se puede incluir aquí las operaciones necesarias para la carga de datos una vez el programa esté en producción.

## IAS 5: Pruebas de Implantación del Sistema

Si fuese oportuno, se pueden mencionar aquí pruebas específicas para asegurar que la implantación del sistema en el entorno de producción (uso por parte de los usuarios finales, interacción con sistemas existentes…). Este es también un buen apartado para hacer un estudio de la seguridad del sistema final implantado, aunque no sea un requisito central de la aplicación. La forma más sencilla de hacerlo es instalar un analizador automático gratuito de vulnerabilidades como OpenVAS siguiendo este procedimiento:

* <https://blog.ehcgroup.io/index.php/2018/06/21/escaneo-de-vulnerabilidades-con-openvas-9-parte-1-instalacion-y-configuracion/>
* <https://blog.ehcgroup.io/index.php/2018/06/21/escaneo-de-vulnerabilidades-con-openvas-9-parte-2-escaneo-de-vulnerabilidades/>

Y lanzarlo contra la máquina física/virtual **de nuestra propiedad o sobre la que nos hayan autorizado** para comprobar si presenta vulnerabilidades graves. En caso de que ese sea el caso, la propia descripción de la vulnerabilidad dice como solventarlas. Conviene arreglar aquellas que se presenten como críticas o muy graves para no presentar un producto inherentemente inseguro. Al final del proceso, se puede incluir en esta sección el informe generado por la herramienta directamente a modo de documentación. No obstante, se deben seguir las indicaciones de la asignatura “**Seguridad de Sistemas Informáticos**” de tercer curso para gestionar este aspecto.

También debe tenerse en cuenta si el proyecto debe cumplir con alguna garantía de cara a resistir posibles ataques de potenciales usuarios maliciosos, por lo que habría que hacerle algunas pruebas de pentesting al efecto [21], bien de forma manual, o con potenciales herramientas especializadas [22] [23].

## IAS 7: Preparación del Mantenimiento del Sistema

Si el sistema requiriese algún mantenimiento periódico de alguna clase, puede describirse aquí.

## IAS 8: Establecimiento del Acuerdo de Nivel de Servicio

En caso de que el sistema tuviera un SLA al estar destinado para un cliente real, conviene incluir aquí el texto de dicho acuerdo.

## IAS 9-10: Presentación y Aprobación del Sistema y Paso a Producción

Si el sistema es para un cliente real y está ahora mismo en producción, puede incluirse en esta sección detalles sobre como va su explotación o similares. Por ejemplo, si está en un market móvil, se puede mencionar cuantas descargas tiene, su valoración, etc.

# Apéndices

## Presupuesto

Aquí se especificará el presupuesto del desarrollo del software tal y como se ha enseñado a realizar en la asignatura “**Dirección y Planificación de Proyectos Informáticos**” de cuarto curso.

## Problemas Encontrados Durante el Desarrollo

Enumeramos los problemas encontrados en el desarrollo y la solución que le hemos dado, si hubo alguno que merezca la pena destacar y no lo hubiéramos hecho ya como parte de otras secciones.

## Conclusiones

Conclusiones del sistema: Qué hemos elaborado, si los resultados están dentro de lo esperado, si hemos cumplido las expectativas, justificación de haber escogido las mejores opciones para cada uno de los aspectos del sistema, etc.

## Ampliaciones

Cualquier labor de ampliación que tengamos contemplada en el sistema debe ser descrita aquí, mencionando en qué consiste, cómo ampliará el sistema, qué ventajas nos aporta y por qué no se ha incluido en el sistema diseñado, entre otros aspectos.

## Referencias Bibliográficas

La siguiente guía explica como incluir referencias bibliográficas correctamente en Word y es la que se recomienda usar (y se ha usado en este documento): <https://support.office.com/es-es/article/agregar-citas-en-un-documento-de-word-ab9322bb-a8d3-47f4-80c8-63c06779f127>

La referencia a esta plantilla en ResearchGate (la primera de todas) **debe dejarse en el documento tal cual aparece a continuación**:

|  |  |
| --- | --- |
| [1] | J. M. Redondo, «Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo,» 17 6 2019. [En línea]. Available: https://www.researchgate.net/publication/327882831\_Plantilla\_de\_Proyectos\_de\_Fin\_de\_Carrera\_de\_la\_Escuela\_de\_Informatica\_de\_Oviedo. |
| [2] | J. M. Redondo y F. Ortín, «A SaaS framework for credit risk analysis services,» *IEEE Latin America Transactions,* vol. 15, nº 3, pp. 474-481, 2017. |
| [3] | J. M. Redondo y L. Varela, «Filesync and Era Literaria. Realistic Open Source Webs To Develop Web Security Skills,» *Journal of Web Engineering,* vol. 17, nº 5, pp. 1-22, 2018. |
| [4] | O. Zaleski, M. Navarro, S. Díaz, J. M. Redondo y M. Labrador, «Clinical Gait Assessment Comparison: Smartphone-based versus Inertial Measurements Units,» de *IEEE Southeast Con*, 2018. |
| [5] | A. M. Fernández, B. M. González, J. M. Redondo y A. J. Sánchez, «Tirsus IV: Navegación Multidimensional en Aplicaciones Hipermedia sobre Acontecimientos Históricos,» de *Interacción 2000*, Granada, 2000. |
| [6] | A. J. Sánchez, A. M. Fernández, B. M. González y J. M. Redondo, «Tirsus II, Aplicación de Hipermedia para la Enseñanza de la Historia,» de *Interacción 2000*, Granada, 2000. |
| [7] | J. M. Redondo, A. J. Sánchez, A. M. Fernández y B. M. González, «Cinemedia Astur: Herramienta Generadora de Títulos Hipermedia de Interfaz Flexible,» de *Interacción 2000*, Granada, 2000. |
| [8] | I. Lagartos, J. M. Redondo y F. Ortín, «Efficient Runtime Metaprogramming Services for Java,» *Journal of Systems and Software,* vol. 153, pp. 220-237, 2019. |
| [9] | J. M. Redondo, «FanCines: Understand C Sharp concepts the easy way,» 1 5 2019. [En línea]. Available: https://www.researchgate.net/publication/333104441\_FanCines\_Understand\_C\_Sharp\_concepts\_the\_easy\_way. |
| [10] | J. M. Redondo, «Admin-zines: Understand Infrastructure Administration concepts the easy way,» 8 2019. [En línea]. Available: https://www.researchgate.net/publication/335023411\_Admin-zines\_Understand\_Infrastructure\_Administration\_concepts\_the\_easy\_way. |
| [11] | F. Ortin, J. M. Redondo y J. Quiroga, «Design and evaluation of an alternative programming paradigms course,» *Telematics and Informatics,* vol. 34, nº 6, pp. 813-823, 2017. |
| [12] | F. Ortin, D. Zapico y J. M. Redondo, «Utilización de patrones de diseño en la asignatura de Procesadores de Lenguaje,» de *XV Jornadas de Enseñanza Universitaria de la Informática*, Barecelona, 2009. |
| [13] | J. M. Redondo and F. Ortin, "A comprehensive evaluation of common python implementations," *IEEE Software,* vol. 32, no. 4, pp. 76-84, 2014. |
| [14] | F. Ortin, M. García and J. M. Q. J. Redondo, "Combining static and dynamic typing to achieve multiple dispatch," *Information–An International Interdisciplinary Journal,* vol. 16, no. 12, pp. 8731-8750, 2013. |
| [15] | J. M. Redondo, «New Features of C Sharp 8 and beyond,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330514620\_New\_Features\_of\_C\_Sharp\_8\_and\_beyond. |
| [16] | J. M. Redondo, «New Features of C Sharp 7,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330358763\_New\_Features\_of\_C\_Sharp\_7. |
| [17] | J. M. Redondo, «New Features of C Sharp 5 and 6,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330223681\_New\_Features\_of\_C\_Sharp\_5\_and\_6. |
| [18] | H. Montero, «No solo usabilidad: revista sobre personas, diseño y tecnología,» 30 3 2003. [En línea]. Available: http://www.nosolousabilidad.com/articulos/heuristica.htm. |
| [19] | J. M. Redondo, Introducción Práctica a la Administración Segura de Servidores Apache Bajo Linux, Oviedo, Asturias: Servicio de Publicaciones de la Universidad de Oviedo, 2019. |
| [20] | J. M. Redondo, «Improving Student Assessment of a Server Administration Course Promoting Flexibility and Competitiveness,» *IEEE Transactions on Education,* vol. 62, nº 1, pp. 1-8, 2019. |
| [21] | J. M. Redondo, "Are You Breathacking? Introduction to hacking with a practical example," 1 7 2019. [Online]. Available: https://www.researchgate.net/publication/334303691\_Are\_You\_Breath-hacking\_Introduccion\_en\_Espanol\_al\_mundo\_del\_hacking\_con\_un\_ejemplo\_practico. |
| [22] | J. M. Redondo and D. Cuesta, "Towards Improving Productivity in NMap Security Audits," *Journal of Web Engineering,* vol. 18, no. 7, pp. 1-38, 2019. |
| [23] | I. Llaneza, J. M. Redondo y L. Vinuesa, «Towards Lightweight Mobile Pentesting Tools to Quickly Assess Machine Security Levels,» *IEEE Latin American Transactions,* p. pp, 2019. |

## Contenido entregado en los anexos

### Contenidos

Aparte del documento principal del trabajo de fin de grado se sube al sistema un fichero comprimido adjunto con una serie de contenidos adicionales: diagramas adicionales, documentación tipo Javadoc, código fuente, ejecutable, …. Aunque lo subido depende del tipo de proyecto a desarrollar, en esta sección se dan unas guías generales de como subirlo correctamente.

La plataforma de la Universidad tiene un límite de tamaño del fichero subido que suele ser de unos 40Mb. Si el adjunto fuese más grande, se recomienda crear un fichero README.TXT en el fichero comprimido subido y usar la función de compartir un enlace a un fichero de Dropbox, el OneDrive de la universidad o similar para que quien esté interesado en descargar estos ficheros. Preferiblemente debería usarse el OneDrive de la universidad, al ser institucional. También es posible apuntar a un repositorio GitHub o similar

La estructura de directorios del fichero adjunto debe poder recoger todos los ficheros relacionados con el proyecto, clasificándolos por su propósito dentro del mismo. Los tipos más frecuentes son: ficheros fuente, ficheros de configuración, ficheros de documentación…

Se deben crear directorios para contener cada uno de los tipos de ficheros. Tener una estructura estandarizada de los directorios del proyecto es importante por varias razones:

* Ayuda a localizar la información del proyecto. Por ejemplo, los ficheros fuente siempre deben estar en la carpeta src.
* Ayuda a los desarrolladores a determinar dónde debe ir cada fichero.
* Permite crear scripts de construcción estandarizados.

|  |  |
| --- | --- |
| **Directorio** | **Contenido** |
| *./* | Contiene un fichero README.TXT explicando toda esta estructura. |
| *./<nombre\_proyecto>* | Contiene toda la estructura de directorios del proyecto para desarrollo. **Ver la tabla Recomendación de estructura de directorios de desarrollo**. <nombre\_proyecto> debe sustituirse por el nombre corto del proyecto. |
| ./instalacion | Ficheros utilizados para la instalación del proyecto o un documento explicando donde descargar el software de terceros que necesita para instalarlo, y las versiones correspondientes si fuese necesario. Es también deseable incluir un fichero con el procedimiento de instalación típico abreviado para hacer una instalación del trabajo por defecto lo más rápidamente posible. |
| ./documentacion | Contiene la documentación asociada al proyecto en PDF (este mismo documento y cualquier otro que se considere necesario). |
| ./explotacion | BD y documento explicando donde descargar el software de terceros que se necesita para ejecutar el trabajo (por ejemplo, el servidor web), y las versiones correspondientes si fuese necesario. También se deberían incluir ficheros de configuración si fuesen necesarios para que el trabajo funcione |

Estructura general del fichero anexo entregado

#### Recomendación de la Estructura de Directorios de “desarrollo”

Se muestra aquí el contenido del directorio de desarrollo de la tabla anterior, incluyendo todos los directorios que deben depender del mismo. Este ejemplo se ha elaborado suponiendo que se están usando ciertas tecnologías Java. En caso de no usarlas, buscaremos un equivalente existente (si lo hay) en la que estemos usando nosotros. Si la aplicación está destinada a dispositivos móviles, se debe adaptar el propósito de las carpetas al equivalente en las tecnologías usadas para desarrollar para ellos.

|  |  |
| --- | --- |
| **Directorio** | **Contenido** |
| ./ Directorio raíz de “desarrollo” | Contiene los ficheros de proyecto del IDE utilizado |
| ./build | Contiene el build.xml de ant (si lo usamos) o herramienta de propósito similar. Debemos situarnos dentro para poder invocarlo |
| ./conf | Contiene los diferentes ficheros de configuración del proyecto. Podría contener distintos subdirectorios, en función de la tecnología usada. En este ejemplo se muestra un ejemplo de un proyecto *Web* hecho con tecnologías *Java*:   * **web**: contiene los ficheros de configuración de la aplicación Web (por ejemplo: web.xml) * **ear**: contiene los ficheros de configuración de una aplicación empresarial (por ejemplo: application.xml) * **ejb**: contiene los descriptores de despliegue de los EJB |
| ./dist | Directorio donde se sitúan los ficheros para la distribución del proyecto. Por ejemplo: los ficheros .war o .ear |
| ./doc | Contiene los ficheros generados por herramientas de generación de documentación automática como *Javadoc* o similar |
| ./lib | Bibliotecas externas (.jar, *.dll*, …) necesarias para compilar y distribuir, de las que depende este proyecto |
| ./compile-lib | Bibliotecas externas (.jar, .dll, …) necesarias para compilar pero que no deseamos distribuir |
| ./src | Ficheros de código fuente |
| ./src/sql | Este directorio contiene los scripts de *SQL* que permiten construir y meter los datos iniciales en la base de datos del proyecto (si existe) |
| ./web | Este directorio contiene los ficheros (.JSP, .ASPX, .HTML, …) de la *Web* (si el proyecto incluyese una) |
| ./web/images | Contiene las imágenes utilizadas por los ficheros de la web del proyecto |
| ./classes | Directorio donde se guardan los ficheros compilados (como por ejemplo los .class) |
| ./test | Directorio base para todos los ficheros utilizados en la automatización del proceso de prueba |
| ./test/java | Contiene todas las pruebas unitarias utilizadas en el proceso de prueba automatizado |
| ./test/sql | Scripts *SQL* utilizados en la carga de datos de prueba |

Estructura de la carpeta “desarrollo” del fichero anexo entregado

## GNU Free Documentation License

*Version 1.3, 3 November 2008*

*Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <*[*https://fsf.org/*](https://fsf.org/)*>*

*Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.*

#### 0. PREAMBLE

*The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.*

*This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.*

*We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.*

#### 1. APPLICABILITY AND DEFINITIONS

*This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.*

*A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.*

*A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.*

*The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.*

*The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.*

*A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".*

*Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.*

*The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.*

*The "publisher" means any person or entity that distributes copies of the Document to the public.*

*A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.*

*The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.*

#### 2. VERBATIM COPYING

*You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.*

*You may also lend copies, under the same conditions stated above, and you may publicly display copies.*

#### 3. COPYING IN QUANTITY

*If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.*

*If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.*

*If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.*

*It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.*

#### 4. MODIFICATIONS

*You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:*

1. *Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.*
2. *List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.*
3. *State on the Title page the name of the publisher of the Modified Version, as the publisher.*
4. *Preserve all the copyright notices of the Document.*
5. *Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.*
6. *Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.*
7. *Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.*
8. *Include an unaltered copy of this License.*
9. *Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.*
10. *Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.*
11. *For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.*
12. *Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.*
13. *Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.*
14. *Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.*
15. *Preserve any Warranty Disclaimers.*

*If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.*

*You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.*

*You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.*

*The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.*

#### 5. COMBINING DOCUMENTS

*You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.*

*The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.*

*In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".*

#### 6. COLLECTIONS OF DOCUMENTS

*You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.*

*You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.*

#### 7. AGGREGATION WITH INDEPENDENT WORKS

*A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.*

*If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.*

#### 8. TRANSLATION

*Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.*

*If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.*

#### 9. TERMINATION

*You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.*

*However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.*

*Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.*

*Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.*

#### 10. FUTURE REVISIONS OF THIS LICENSE

*The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See* [*https://www.gnu.org/licenses/*](https://www.gnu.org/licenses/)*.*

*Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.*

#### 11. RELICENSING

*"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.*

*"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.*

*"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.*

*An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.*

*The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.*

### ADDENDUM: How to use this License for your documents

*To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:*

Copyright (C) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document

under the terms of the GNU Free Documentation License, Version 1.3

or any later version published by the Free Software Foundation;

with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU

Free Documentation License".

*If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:*

with the Invariant Sections being LIST THEIR TITLES, with the

Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

*If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.*

*If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.*