



ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

Nombre de mi aplicación: Título ilustrativo, dando a entender qué es, para qué sirve, tecnología y/o metodología

Mi nombre y apellidos

27 de diciembre de 2021



ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

Nombre de mi aplicación: Título ilustrativo, dando a entender qué es, para qué sirve, tecnología y/o metodología

- Departamento: Ingeniería Informática
- Director del proyecto: Pablo de la Torre Moreno
- Co-director del proyecto: Nombre y apellidos de mi co-tutor, o vacío
- Autor del proyecto: Mi nombre y apellidos

Mi localidad, 27 de diciembre de 2021

Fdo: Mi nombre y apellidos

DECLARACIÓN PERSONAL DE AUTORÍA

Mi nombre y apellidos, con DNI 11111111-A, estudiante del **Grado en Ingeniería Informática** en la Escuela Superior de Ingenieríade la Universidad de Cádiz, como autor de este documento académico titulado “***Nombre de mi aplicación: Título ilustrativo, dando a entender qué es, para qué sirve, tecnología y/o metodología***” y presentado como Trabajo Final de Grado,

DECLARO QUE

es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Puerto Real, a 27 de diciembre de 2021.

Resumen

Resumen del trabajo. No debe pasar de una página y debe describir, en párrafos diferentes (al menos un párrafo para cada parte): contexto, problema, propuesta y conclusiones.

Agradecimientos

Uno o varios párrafos de agradecimiento, en cursiva.

Licencia

Copyright y licencia. Si se emplea esta plantilla, debe dejarse el párrafo siguiente.

La plantilla de este documento (no su contenido) se basa parcialmente en la desarrollada por Pablo Recio Quijano, Noelia Sales Montes y Francisco Javier Vázquez Púa, la cual ha sido liberada bajo Licencia GFDL 1.3 (GNU *Free Documentation License*) y con *copyright* © 2009 Pablo Recio Quijano, y ha sido creada por Pablo de la Torre Moreno con *copyright* © 2016-2021 Pablo de la Torre Moreno, en los términos siguientes. *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”.*

Notación y formato

El presente documento utiliza un conjunto de convenios de notación de sintaxis, tal y como se describen en la Tabla ??.

| Estilo | Uso |
|----------------|---|
| negrita | Título o texto destacado. |
| <i>itálica</i> | Texto en otro idioma, destacado, citas o nombres de aplicaciones. |
| color | Enlace interno o externo. |
| monoespaciado | Código fuente. |
| subrayado | Advertencia para el lector. |
| sombreado | Meta-información. Comentarios para el alumno. |

Tabla 1: Convenios de notación y formato

La meta-información se incluye por motivos aclaratorios, bien relativos al documento en sí o a las técnicas a utilizar. No deberá aparecer meta-información en la memoria final del **TFG!** (**TFG!**) a entregar, por lo que deberá suprimirse toda meta-información de este documento (lo que incluye este párrafo) y dicha entrada de la Tabla ??.

Por otro lado, la plantilla aporta al autor el uso de dos etiquetas más: `\alert` y `\todo`. Por ejemplo, `\alert{Esto es una alerta}` aparece como **Esto es una alerta**, y `\todo{No olvides esta parte}` se muestra como **TODO: No olvides esta parte**. Dichas etiquetas son para anotaciones personales de uso exclusivo durante la elaboración del documento, pero no deben aparecer en el entregable final.

Precisamente como parte de esta meta-información, se facilita a continuación una serie de normas sintácticas que se han detectado como errores comunes en la notación y formato de diferentes memorias de **TFG!**. Las referencias relativas a ortografía y gramática pueden encontrarse publicada en libros y artículos de la Real Academia Española [?, ?, ?]:

- En castellano, las siglas no llevan plural. No es “los TFGs”, ni “los TFG’s”, sino “los TFG”.
- Los meses del año y días de la semana se escriben siempre en minúscula (enero, martes, noviembre, domingo) salvo festividades (por ejemplo, Viernes Santo).
- Los años no llevan punto de separación. No es “2.016”, sino “2016”. De hecho, no se aconseja poner ningún separador a números de cuatro cifras, sean años o no.
- Los números incluidos en párrafos no se escriben usando dígitos numéricos, salvo cuando describan resultados matemáticos, porcentajes o el texto sea grande. Así, no es correcto: “hemos definido un total de 3 iteraciones”, sino “hemos definido un total de tres iteraciones”.
- Respecto a la tilde en monosílabos (diacrítica), los únicos que lo permiten son “tú”, “él”, “mí” (todos como pronombres), “sí” (ídem y adverbio de afirmación), “té” (bebida), “dé”

(del verbo dar), “sé” (del verbo saber) y “más” (cuando NO equivale a “pero”). Un error muy común es escribir “tí” (con tilde), cuando siempre se escribe sin tilde.

- El punto y coma indica una pausa mayor que la coma, detrás se escribe en minúscula (salvo que la palabra sea nombre propio o similar) y se utiliza comúnmente para separar enumeraciones complejas, al menos alguna de las cuales suele contener una coma. Por ejemplo: “Cada grupo irá por un lado diferente: el primero, por la izquierda; el segundo, por la derecha; y, el tercero, de frente”.
- En cuanto a cuándo usar punto y cuándo usar coma, hay un sencillo truco: si puedes decir una oración de corrido sin hacer ninguna interrupción y la frase no pierde su sentido, seguro que no va un punto; como mucho, irá una coma, punto y coma, o dos puntos. Si pierde su sentido, es que va un punto allí donde deja de tenerlo.
- Antes de un signo de puntuación nunca se deja espacios. Después de un signo de puntuación siempre se deja un espacio (salvo fin de párrafo).
- Entre el sujeto y el predicado nunca se escribe coma, salvo la existencia de una subordinada. No es “la actual aplicación, ha sido desarrollada en Java”, sino “la actual aplicación ha sido desarrollada en Java”.
- De igual manera, “La motivación, el desarrollo y las conclusiones se incluyen en esta memoria” no lleva coma de separación aunque haya un enumerado. La excepción a esta norma es que el enumerado acabe en “etcétera” o similar.
- Los puntos suspensivos son siempre tres y sólo tres, y van unidos a la palabra que los precede. Pueden sustituirse por “etcétera” y por “etc.”, pero no se escriben juntos; es decir, no es correcto “Java, C, Prolog, etcétera...”, sino “Java, C, Prolog...”, o bien “Java, C, Prolog, etc.”, o incluso “Java, C, Prolog, etcetera”. Detrás de los puntos suspensivos puede haber una coma, en su caso.
- No obstante, hay que evitar los “etcétera” y similares en lo posible. Da la sensación de información incompleta.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos fuente

| | |
|---|----|
| 6.1. Java - Clase <code>UserTest</code> de pruebas unitarias para clase <code>User</code> | 54 |
|---|----|

Parte I

Prolegómeno

Capítulo 1

Introducción

El capítulo actual presenta el proyecto desde el punto de vista de la motivación, los objetivos a alcanzar y los conceptos básicos requeridos para comprender el dominio del problema. Posteriormente, se realiza un estudio del estado del arte sobre soluciones similares. Más adelante, se plantean los objetivos y sub-objetivos. Por último, se facilita la estructura del documento actual y las herramientas de ofimática utilizadas para la elaboración de esta memoria.

1.1. Motivación

Desarrollo del contexto y el problema referidos en el resumen (`abstract.tex`), con una carilla como mínimo.

1.2. Alcance

Relación de los perfiles potenciales a quienes va dirigida la aplicación, incluyendo usuarios generales, clientes específicos y/o grupos de interés.

1.3. Conceptos básicos

Se repasan en esta sección los conceptos necesarios para la comprensión del dominio del problema. Por razones prácticas, se obvian aquéllos obvios o que conforman el lenguaje consuetudinario de la ingeniería informática (como “servicio”, “servidor”, “dispositivo móvil”, etcétera).

Describir, cada uno en una sub-sección, todos los conceptos de interés relativos al dominio del problema.

Primer concepto básico

1.4. Estado del arte

Introducir aquí el estudio, y finalizar con lo siguiente.

En la Tabla ?? (pág. ??) se ofrece un cuadro comparativo de características. La comparativa presenta una pequeña parte de la oferta de productos similares, que se ha intentado que sea suficientemente representativa.

El estudio del estado del arte es trabajo de investigación, no de desarrollo, y por esa razón a veces resulta poco motivador para el alumno. No obstante, se requiere para aprender a cuidarse de no hacer trabajo repetido. Hay ocasiones en las que la adquisición de soluciones de terceros (el estado del arte no se circunscribe únicamente a aplicaciones, sino que también puede incluir bibliotecas específicas) es más óptimo que programar funciones que ya existen en el mercado. Por otra parte, se requiere poder justificar al cliente por qué debe apostar por nuestra solución, y no por otra del mercado. En el ámbito del **TFG!**, se aconseja el análisis de entre tres y cinco soluciones existentes. La profundidad del análisis y las características recogidas dependen del proyecto a presentar.

TeenSafe

TeenSafe [?] es una aplicación de pago que se instala en un dispositivo iPhone y Android, permitiendo en remoto acceder a SMS, iMessages, llamadas realizadas y contactos. También es posible comprobar la ubicación actual del teléfono. En relación a las redes sociales, el sistema permite leer mensajes de *Instagram*, *WhatsApp* y *Kik Messenger*. De igual manera, puede revisarse el historial de búsqueda y de navegación en Internet.



Evidentemente esto es un ejemplo. Debe ser sustituido por la primera de las aplicaciones a revisar, y seguido por otras, cada una en su propia sub-sección.

1.5. Objetivos

Hacer una introducción de los objetivos. Debe recordarse que los objetivos no son del alumno, sino del proyecto. Los objetivos del alumno se describen en el Capítulo ?. Posteriormente, se facilitan las tablas de objetivo, con la estructura y orden:

- Objetivo 1

- Subobjetivo 1,1
- ...
- Subobjetivo 1. m_1
- ...
- Objetivo n
 - Subobjetivo $n,1$
 - ...
 - Subobjetivo $n.m_n$

| | |
|---------------|---|
| OBJ-01 | Primer objetivo de la aplicación |
| Descripción | Descripción del objetivo. |

Tabla 1.1: OBJ-01 - Primer objetivo de la aplicación

| | |
|-----------------|---|
| OBJ-01.1 | Primer sub-objetivo del anterior |
| Descripción | Descripción. |

Tabla 1.2: OBJ-01.1 - Primer sub-objetivo del anterior

1.6. Organización del documento

Lo mostrado a continuación es un ejemplo. Deben hacerse las modificaciones pertinentes, en su caso. Debe recordarse que la introducción de cada capítulo se corresponde al mismo texto que su explicación en este apartado. Aunque esto es modificable, se aconseja dejarlo así para evitar tener en cuenta prácticamente el mismo texto en dos zonas diferentes.

El presente documento describe la elaboración del proyecto y está estructurado en las siguientes partes y capítulos:

Parte ??.

Se presenta el proyecto y su planificación, en los capítulos referidos a continuación:

Capítulo ??.

El capítulo actual presenta el proyecto desde el punto de vista de la motivación, los objetivos a alcanzar y los conceptos básicos requeridos para comprender el dominio del problema. Posteriormente, se realiza un estudio del estado del arte sobre soluciones similares. Más adelante, se plantean los objetivos y sub-objetivos. Por último, se facilita la estructura del documento actual y las herramientas de ofimática utilizadas para la elaboración de esta memoria.

Capítulo ??.

En este capítulo se detallan el ciclo de vida y la metodología de desarrollo empleados. Seguidamente, se indican las actividades previstas y su estimación temporal teniendo en cuenta los recursos disponibles. Dichas actividades son plasmadas posteriormente en un calendario de ejecución. Una vez determinados los tiempos, se estudian los riesgos críticos, se revisan y escogen las alternativas tecnológicas, se describen los recursos con los que se cuenta y se prevén los costes del proyecto. Por último, se facilita un resumen de este capítulo.

Parte ??.

Se describe en esta parte el desarrollo del proyecto en sus etapas de análisis, diseño y pruebas:

Capítulo ??.

Se incluye en este capítulo la especificación y análisis de los requisitos del cliente. En concreto, el catálogo de ??, los ??, los ?? y los ?. Los ?? incluyen las ??, sus ?? y, finalmente, un ?. Asimismo, los ?? se ilustran mediante ??, diagramas de ??, descripción de los mismos y la correspondiente ?. El capítulo se cierra con un resumen de todo el análisis.

Capítulo ??.

Se detalla en este capítulo la expansión del análisis teniendo en cuenta las decisiones técnicas y sus restricciones, constituyendo la arquitectura en la que se basará la fase de implementación. Dicha arquitectura se presenta en su vertiente de ??, ilustrándose el diagrama de equipos e interconexiones necesarias para el funcionamiento del sistema, así como sus especificaciones técnicas; y ??, donde se muestran las capas lógicas del sistema, y los servicios y aplicaciones requeridos. De igual manera, se presentan los ?? que el sistema empleará (a través de estas arquitecturas) para comunicarse con el usuario.

En cuanto a los datos, se amplía el modelo conceptual para transformarlo en un ??, en este proyecto cumpliendo los criterios de una base de datos relacional. De igual manera, los casos de uso se plasman visualmente en una serie de bocetos que ilustran la ?. Por último, se profundiza en los artefactos utilizados en el ?.

Capítulo ??.

Se describen en este capítulo los detalles de implementación. Tras la concreción de los ??, se parte de la ?? para proceder a explicar la su distribución y los elementos importantes del código: la ??, la implementación de la ??, la codificación de aspectos relativos a la ?? y la programación de las ?. Para cerrar el capítulo, se realiza un ?.

Por motivos prácticos, no se revisará todo el código, sino ejemplos representativos de éste. En todo caso, se garantiza que los ejemplos abarcan los aspectos más interesantes de la codificación, quedando fuera de esta memoria aspectos superfluos o repetitivos.

Capítulo ??.

En este capítulo se documentan los diferentes tipos de pruebas que se han llevado a cabo durante el desarrollo del sistema. En primer lugar, se describe la ?? y el ?. Posteriormente, se detallan las diferentes fases referidas: ??, que comprueba el funcionamiento independiente de los distintos componentes de la aplicación; ??, que verifica que el ensamblado entre dichos

componentes es correcto; ??, donde se confronta los requisitos del modelo con la operatividad del software; y ??, cuyo objetivo es asegurar que el sistema funciona correctamente en explotación.

Parte ??.

Como parte del proyecto, se presentan las conclusiones del mismo:

Capítulo ??.

Como capítulo final en la elaboración del proyecto, se describen los ?? del mismo y el ??. Asimismo, se comparten las ??.

Apéndices

Los apéndices facilitados al final del documento son [LOS QUE TOQUE]:

- Apéndice ??, donde se transcribe el manual del usuario.

Bibliografía y referencias

Finalmente, se enumeran las referencias bibliográficas.

Ejemplo de tabla comparativa de las aplicaciones revisadas durante el estado del arte. Las columnas deben ser modificadas para ajustarse a los criterios del estudio.

| Solución | Plataforma | Licencia | Alertas | Mensajería | Redes sociales | Llamadas | Contactos | Internet |
|-------------------------|---|----------|---------|-------------------------------------|--|----------|----------------------------|----------------------------------|
| <i>TeenSafe</i> | iPhone Android | De pago | No | SMS iMessages <i>WhatsApp</i> | <i>Instagram</i> <i>Kik</i> | Teléfono | Teléfono | Búsqueda Navegación |
| <i>Norton Family</i> | Windows Android | De pago | Sí | SMS <i>WhatsApp</i> | <i>Facebook</i> <i>Orkut</i> <i>Mixi</i> (jap.) <i>Skyrock</i> (fr.) | Teléfono | Redes sociales Teléfono | Búsqueda Navegación Vídeos |
| <i>Movistar Protege</i> | Windows Android Mac OS Kindle iOS | De pago | No | SMS | <i>Facebook</i> <i>Twitter</i> | Teléfono | Redes sociales Teléfono | Búsqueda Navegación |
| <i>Qustodio</i> | Windows Android Mac OS iOS | Limitada | Sí | SMS <i>WhatsApp</i> | <i>Facebook</i> <i>Twitter</i> <i>Instagram</i> | Teléfono | Redes sociales Teléfono | Búsqueda Navegación |
| <i>SecureTeen</i> | Windows Android iOS | De pago | No | SMS <i>WhatsApp</i> (iOS) | <i>Facebook</i> <i>Instagram</i> <i>Kik</i> (iOS) <i>Viber</i> (iOS) <i>LINE</i> (iOS) | Teléfono | Redes sociales Teléfono | Búsqueda Navegación |

Tabla 1.3: Estado del arte, comparativa

Capítulo 2

Planificación

En este capítulo se detallan el ciclo de vida y la metodología de desarrollo empleados. Seguidamente, se indican las actividades previstas y su estimación temporal teniendo en cuenta los recursos disponibles. Dichas actividades son plasmadas posteriormente en un calendario de ejecución. Una vez determinados los tiempos, se estudian los riesgos críticos, se revisan y escogen las alternativas tecnológicas, se describen los recursos con los que se cuenta y se prevén los costes del proyecto. Por último, se facilita un resumen de este capítulo.

2.1. Metodología de desarrollo

Descripción detallada del ciclo de vida y la metodología a utilizar. Debe explicarse y referenciarse (bibliografía) lo necesario para justificar la decisión. El motivo es que cualquier evaluador (y cualquier empresario) debería exigir criterios objetivos por los cuales un ingeniero ha tomado una decisión en lugar de otra. Se requiere por tanto justificar las razones por las que alternativas concretas son escogidas. Para ello es necesario demostrar: a) que la opción elegida es beneficiosa por sí misma (por ejemplo, que existe una comunidad profesional que la secunde); y, b) por comparación, que no hay otras opciones que aporten un beneficio sustancialmente mayor para el tipo de proyecto presentado. En todo caso, no se exige ningún ciclo de vida ni metodología concretos. Otros modelos como el ciclo de vida en cascada, basado en prototipos o métodos ágiles son válidos siempre y cuando se justifiquen los puntos *a* y *b* antes referidos.

2.2. Cronograma del proyecto

Atendiendo a los objetivos y a la metodología, se formaliza la planificación del proyecto, dividiéndolo en etapas o no según el ciclo de vida. Debe facilitarse un calendario del proyecto, para lo que se aconseja el uso de diagramas de Gantt.

2.3. Riesgos

Análisis de riesgos indicando referencias que han ayudado a su identificación, más actuaciones preventivas y correctivas.

2.4. Alternativas tecnológicas

A continuación se presenta el estudio y elección de diferentes alternativas tecnológicas, analizadas con objeto de satisfacer los requisitos del proyecto. Los elementos a analizar serán ??, ??, ??, ??, ??, ?? y ??.

Como ha de tenerse en cuenta en toda la memoria, los elementos a analizar dependen del proyecto, si bien se han dejado los subtítulos a modo ilustrativo. Cada uno de ellos debe explicarse y justificarse adecuadamente la decisión. No basta con decir que a uno se le da mejor, que le gusta o que le interesa aprender una tecnología específica; si bien esto puede ser lo fundamental para la elección, hace falta acompañarla de motivaciones objetivas como cuota de mercado, comunidad de soporte, innovación, etcétera.

Plataforma

Arquitectura de red

Lenguaje de programación

Entorno de desarrollo

Almacén de datos

Acceso a datos

Pruebas

2.5. Recursos

Se deja todo el texto, como ejemplo ilustrativo de la creación y uso de tablas en L^AT_EX.

Para el desarrollo del proyecto se ha contado con el autor de esta memoria como único desarrollador. La programación y edición se han realizado con un ordenador portátil, cuya información se indica en la Tabla ??.

Sobre este equipo se instalarán las aplicaciones finalmente escogidas tras el análisis de las ??.

En lo que respecta a la elaboración de esta memoria, se ha empleado el servicio *on-line* *ShareLaTeX* (<https://www.sharelatex.com/>) como editor de L^AT_EX, *Adobe Photoshop®CC 2014* para edición de imágenes (tamaño, composición, color de fondo, recorte y/o cambio de formato), *GanttProject 2.7.2* para diagramas de Gantt, y *Microsoft Visio® 2010* para técnicas de UML! (UML!).

| | |
|-------------------|---|
| Modelo | HP Pavilion g6-2210ss |
| CPU | Intel® Core™ I5-3210M @ 2.50 GHz |
| Pantalla | 39.6 cm (15.6") |
| Memoria RAM | 6 GB |
| Disco duro | ATA Hitachi HTS54505 SCSI 500 GB |
| Sistema operativo | Microsoft Windows® 7 Ultimate SP1 64 bits |

Tabla 2.1: Características de ordenador portátil, recurso

Aunque no se incluye en este **TFG!**, se insta al alumno a trabajar con un gestor de proyectos como *Planner*, *OpenProj*, *DotProject*, *Microsoft Project®*, etcétera, con objeto de adquirir experiencia profesional en el uso de herramientas para planificación de desarrollos.

2.6. Costes

Costes del proyecto y beneficios esperados, de forma rigurosa, y empleando técnicas y fuentes fiables.

2.7. Resumen de la planificación

Resumen ordenado de todos los puntos anteriores, opcionalmente añadiendo consideraciones a discutir.

Parte II

Desarrollo

Capítulo 3

Análisis

Se incluye en este capítulo la especificación y análisis de los requisitos del cliente. En concreto, el catálogo de ??, los ??, los ?? y los ??. Los ?? incluyen las ??, sus ?? y, finalmente, un ??. Asimismo, los ?? se ilustran mediante ??, diagramas de ??, descripción de los mismos y la correspondiente ??. El capítulo se cierra con un resumen de todo el análisis.

3.1. Actores

| ACT-01 | Usuario |
|-------------|--|
| Descripción | Usuario básico de la aplicación, representando al menor que usa la aplicación. |

Tabla 3.1: ACT-01 - Usuario

Completar con el resto de actores. No se aconseja la inclusión de actores secundarios, al no aportar información útil en la práctica. Como alternativa, los únicos actores a incluir son aquéllos que tienen interacción directa con el sistema a construir.

3.2. Requisitos funcionales

Se presentan, en primer lugar, las ?? del sistema. Posteriormente, se dibujan y describen los ?? y la ??.

Ojo a esto, que es un error común: los requisitos funcionales/casos de uso, son acciones que realiza el sistema, no el actor. El actor **INVOCA** a los requisitos funcionales, y el sistema los ejecuta. Así, un requisito funcional denominado *Ver Usuario* es incorrecto, porque el sistema no “ve” nada; el sistema lo que hace es “mostrar” los datos del usuario y, en consecuencia, el requisito debe denominarse *Mostrar Usuario*.

De esta forma, la manera más ordenada de concebir un requisito funcional es mediante la forma *Verbo Entidad*, donde los verbos son generalmente *Agregar*, *Modificar*, *Eliminar* y *Buscar*. Por tanto, *Buscar Contacto* significa que hay una acción que solicita el actor al sistema (que el sistema [no el actor] busque), y hay una entidad que conformará posteriormente los requisitos de información: *Contacto*.

3.2.1. Agrupaciones funcionales

Se incluye una agrupación funcional de ejemplo.

- **Gestión de Permiso**, permitiendo a los usuarios identificarse en el sistema (incluido en otras agrupaciones), así como a los supervisores dar de alta a otros usuarios. Los casos de uso contemplados son *Agregar Usuario*, *Modificar Usuario*, *Eliminar Usuario* y *Buscar Usuarios*, descritos en las Tablas ?? a ??, respectivamente. También se añade el caso de uso *Autenticación*, desplegado en la Tabla ??.
- ...

3.2.2. Casos de uso

Se presenta a continuación la descripción de cada caso de uso por cada agrupación funcional referida en el punto anterior. Para simplificar el modelo, en todo momento se considera que la acción *Buscar* conlleva a una posterior consulta de alguno de los elementos buscados.

Se incluye una agrupación funcional completa de ejemplo, que ilustra los casos de uso **CRUD!** (**CRUD!**).

Gestión de Permiso

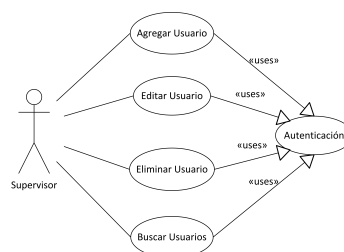


Figura 3.1: Diagrama de Casos de Uso de Gestión de Permiso

Si bien los diagramas de casos de uso son ilustrativos y permiten un vistazo general y estructurado de lo que hace el sistema, la descripción de los mismos pocas veces aporta algo al análisis, sobre todo en lo referente a los escenarios. Por ello y dado que aún sigue siendo práctica obligatoria en la carrera, se aconseja que se le dedique el menor tiempo posible al detalle de dichos escenarios, a excepción de que alguno siga una casuística más allá de las clásicas funciones de agregación, modificación, eliminación y búsqueda. Por otra parte, no es buena práctica referir explícitamente a ningún atributo de los requisitos de información. Es decir, en lugar de “El *actor* introduce el nombre, los apellidos y la edad”, es preferible “El *actor* introduce los datos”. Con eso se logran dos objetivos: en primer lugar, el no dedicarle tiempo innecesario; en segundo, y más importante, aislar en la medida de lo posible los requisitos de información de los casos de uso, de forma que una modificación en aquéllos no afecte a la descripción del proceso,

obligando a hacer doble trabajo y aumentando la probabilidad de inconsistencia por error. No obstante, si hubiera algún dato con un tratamiento especial, sí sería necesario especificarlo en la descripción del caso de uso que lo contempla.

| | |
|-------------------------|---|
| USC-01 | Autenticación |
| Descripción | El <i>actor</i> se autentifica en el sistema, previa comprobación de éste. |
| Actores | Usuario |
| Precondiciones | |
| Escenario principal | <ol style="list-style-type: none"> 1. El <i>sistema</i> comprueba que el <i>actor</i> está autenticado. 2. El <i>sistema</i> muestra el formulario. 3. El <i>actor</i> introduce los datos. 4. El <i>actor</i> acepta la acción. 5. El <i>sistema</i> comprueba que los datos son correctos. 6. El <i>sistema</i> registra al <i>actor</i> como autenticado. |
| Postcondiciones | <ul style="list-style-type: none"> ■ El <i>actor</i> queda autenticado en el sistema. |
| Escenarios alternativos | <ol style="list-style-type: none"> 1.a. El <i>actor</i> ya está autenticado <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. 5.a. Los datos no permiten autenticar. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. 5.b. El <i>sistema</i> no puede acceder el modelo. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. 6.a. El <i>sistema</i> no puede acceder el modelo. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. *.a. El <i>actor</i> cancela la acción. <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. |
| Notas | |

Tabla 3.2: USC-01 - Autenticación

| | |
|-------------------------|---|
| USC-02 | Agregar Usuario |
| Descripción | El <i>actor</i> agrega un nuevo usuario (básico o supervisor) al sistema. |
| Actores | Supervisor |
| Precondiciones | |
| Escenario principal | <ol style="list-style-type: none"> 1. Se invoca al caso de uso <i>Autenticación</i>. 2. El <i>actor</i> solicita la acción referida. 3. El <i>sistema</i> muestra el formulario. 4. El <i>actor</i> introduce los datos. 5. El <i>actor</i> acepta la acción. 6. El <i>sistema</i> realiza los cambios. |
| Postcondiciones | <ul style="list-style-type: none"> ▪ Una nueva entidad con los datos facilitados existe en el sistema. |
| Escenarios alternativos | <ol style="list-style-type: none"> 1.a. No es posible autenticar al <i>actor</i>. <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. 5.a. Los datos son incorrectos. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. 5.b. La entidad ya existe en el sistema. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. 6.a. El <i>sistema</i> no puede modificar el modelo. <ol style="list-style-type: none"> 1. El <i>sistema</i> notifica el error. 2. El <i>sistema</i> vuelve al punto 3. *.a. El <i>actor</i> cancela la acción. <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. |
| Notas | |

Tabla 3.3: USC-02 - Agregar Usuario

| | |
|-------------------------|--|
| USC-03 | Modificar Usuario |
| Descripción | El <i>actor</i> edita información sobre un usuario (básico o supervisor). |
| Actores | Supervisor |
| Precondiciones | <ul style="list-style-type: none"> La entidad existe en el sistema. |
| Escenario principal | <ol style="list-style-type: none"> Se invoca al caso de uso <i>Autenticación</i>. El <i>actor</i> solicita la acción referida. El <i>sistema</i> muestra el formulario. El <i>actor</i> introduce los datos. El <i>actor</i> acepta la acción. El <i>sistema</i> realiza los cambios. |
| Postcondiciones | <ul style="list-style-type: none"> La entidad existe en el sistema con los nuevos datos. |
| Escenarios alternativos | <ol style="list-style-type: none"> 1.a. No es posible autenticar al <i>actor</i>. <ol style="list-style-type: none"> El <i>sistema</i> cancela el caso de uso. 5.a. Los datos son incorrectos. <ol style="list-style-type: none"> El <i>sistema</i> notifica el error. El <i>sistema</i> vuelve al punto 3. 6.a. El <i>sistema</i> no puede modificar el modelo. <ol style="list-style-type: none"> El <i>sistema</i> notifica el error. El <i>sistema</i> vuelve al punto 3. *.a. El <i>actor</i> cancela la acción. <ol style="list-style-type: none"> El <i>sistema</i> cancela el caso de uso. |
| Notas | |

Tabla 3.4: USC-03 - Modificar Usuario

| | |
|-------------------------|--|
| USC-04 | Eliminar Usuario |
| Descripción | El <i>actor</i> elimina un usuario (básico o supervisor). |
| Actores | Supervisor |
| Precondiciones | <ul style="list-style-type: none"> La entidad existe en el sistema. |
| Escenario principal | <ol style="list-style-type: none"> Se invoca al caso de uso <i>Autenticación</i>. El <i>actor</i> solicita la acción referida. El <i>sistema</i> solicita confirmación. El <i>actor</i> acepta la acción. El <i>sistema</i> realiza los cambios. |
| Postcondiciones | <ul style="list-style-type: none"> La entidad queda marcada como inexistente en el sistema. |
| Escenarios alternativos | <ol style="list-style-type: none"> 1.a. No es posible autenticar al <i>actor</i>. <ol style="list-style-type: none"> El <i>sistema</i> cancela el caso de uso. 5.a. El <i>sistema</i> no puede modificar el modelo. <ol style="list-style-type: none"> El <i>sistema</i> notifica el error. El <i>sistema</i> vuelve al punto 3. *.a. El <i>actor</i> cancela la acción. <ol style="list-style-type: none"> El <i>sistema</i> cancela el caso de uso. |
| Notas | La eliminación es lógica, no física. Esto se debe a que así es posible mantener una traza de lo realizado. |

Tabla 3.5: USC-04 - Eliminar Usuario

| | |
|-------------------------|--|
| USC-05 | Buscar Usuarios |
| Descripción | El <i>actor</i> realiza una búsqueda y, posteriormente, consulta el detalle de una de las entidades resultantes. |
| Actores | Supervisor |
| Precondiciones | |
| Escenario principal | <ol style="list-style-type: none"> 1. Se invoca al caso de uso <i>Autenticación</i>. 2. El <i>actor</i> solicita la acción referida. 3. El <i>sistema</i> muestra el formulario de búsqueda. 4. El <i>actor</i> introduce los datos. 5. El <i>actor</i> acepta la acción. 6. El <i>sistema</i> muestra los resultados. 7. El <i>actor</i> solicita el detalle de uno de los resultados. 8. El <i>sistema</i> muestra el detalle. 9. El <i>actor</i> vuelve a realizar el punto 7. |
| Postcondiciones | |
| Escenarios alternativos | <ol style="list-style-type: none"> 1.a. No es posible autenticar al <i>actor</i>. <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. 6.a. El <i>sistema</i> no encuentra resultados de la búsqueda. <ol style="list-style-type: none"> 1. El <i>sistema</i> indica dicha eventualidad. 2. El <i>sistema</i> cancela el caso de uso. *.a. El <i>actor</i> cancela la acción. <ol style="list-style-type: none"> 1. El <i>sistema</i> cancela el caso de uso. |
| Notas | |

Tabla 3.6: USC-05 - Buscar Usuarios

Otra agrupación funcional, con sus casos de uso, etcétera

sectionRequisitos de información

Se presentan a continuación las ?? que conforman requisitos de información del sistema. Seguidamente, se facilitan las ??. Por último, en esta misma sección, se ilustra el ??.

Para la relación de atributos de las ?? se usará como referencia los tipos descritos en la Tabla ??. Asimismo, se incluye si el atributo es único y si es obligatorio.

Existen varias diferencias importantes entre los requisitos de información y las tablas de las bases de datos, que en ocasiones se confunden. Dichas divergencias son, fundamentalmente:

- Los requisitos de información son más parecidos a las clases que a las tablas.
- No existen “claves primarias”, “claves ajenas” ni ningún otro tipo de elemento característico de los sistemas de gestión de bases de datos relacionales (y mayormente exclusivo de ellos). Por ello, no deben incluirse campos “Id” salvo que realmente el cliente haya requerido que haya un campo identificador único, cosa que no suele ocurrir. Dichos artificios aparecerán en todo caso en la fase de ??.
- Los nombres de las entidades se indican en singular, salvo que cada entrada de dicha entidad pueda representar a más de un elemento (por ejemplo, una entidad *Ruedas* con

| Tipo de dato | Descripción |
|--------------------------------|---|
| <code>texto</code> | Campo alfanumérico. Si el tamaño es acotado, se usa <code>texto(<i>tamaño</i>)</code> . |
| <code>entero</code> | Número entero. |
| <code>natural</code> | Número natural. |
| <code>booleano</code> | Verdadero o falso. |
| <code>fecha</code> | Homónimo. |
| <code>hora</code> | Homónimo. |
| <code>fechahora</code> | Campos que incluyen los dos anteriores. |
| <code>uri</code> | Enlaces o descriptores (por ejemplo, rutas de archivos). |
| <code>binario</code> | Archivos incrustados. |
| <code>imagen</code> | Tipo particular de <code>binario</code> , que almacena una imagen. |
| <code>lista<tipo></code> | Secuencia de un tipo específico. |
| <i>Requisito</i> | Cualquier tipo definido como requisito de información. |

Tabla 3.7: Tipos de datos de análisis

los atributos *Rueda 1*, *Rueda 2*, etcétera), lo cual es muy poco habitual. Dicho criterio se mantendrá también cuando tales entidades se conviertan en tablas, en su caso.

- Deben emplearse nombres claros para los atributos.
- El nombre de los atributos no debe contener el nombre de la entidad. Para una entidad *Usuario*, no es correcto llamar a un atributo *CódigoUsuario*, sino *Código*.
- Todo atributo candidato a contener un texto dentro de un conjunto acotado de textos se concibe como una relación a una entidad débil. Por ejemplo, un atributo como *País* no debe ser especificado como tipo `texto`, ya que su valor forma parte de un conjunto acotado. En su lugar, el tipo es **País**, que deberá referenciar a una entidad *País* con (al menos) un atributo *Nombre*.

3.2.3. Entidades

Se incluye como ejemplo la entidad *Usuario*.

De acuerdo con el estudio del estado del arte

| IRQ-01 | Usuario | | | |
|---------------|----------------------------------|-----------------|-------|------|
| Descripción | Usuario básico de la aplicación. | | | |
| Atributos | Nombre | Tipo | Único | Obl. |
| | Nombre | texto(64) | | ✓ |
| | Apellidos | texto(128) | | ✓ |
| | Usuario | texto(16) | ✓ | ✓ |
| | Contraseña | texto(16) | | ✓ |
| | Contactos | lista<Contacto> | | ✓ |

Tabla 3.8: IRQ-01 - Usuario

3.2.4. Reglas de negocio

Por motivos de claridad y para simplificar el número de reglas de negocio explícitas, se ha incluido la obligatoriedad y unicidad de los atributos en el mismo detalle de las **??**. No obstante, también y a continuación se especifican de forma general las reglas de negocio respectivas, así como otras específicas.

| BRU-01 | Obligatoriedad de atributos |
|---------------|---|
| Descripción | Los campos especificados como tales en sus descripciones (ver Sección ??) deben ser obligatorios. |

Tabla 3.9: BRU-01 - Obligatoriedad de atributos

| BRU-02 | Unicidad de atributos |
|---------------|--|
| Descripción | Los campos especificados en sus descripciones (ver Sección ??) deben ser únicos en el conjunto de elementos de su entidad. |

Tabla 3.10: BRU-02 - Unicidad de atributos

| BRU-03 | Otra regla |
|---------------|-------------------------------|
| Descripción | Descripción de la otra regla. |

Tabla 3.11: BRU-03 - Otra regla

3.2.5. Diagrama conceptual

Se facilita un ejemplo para ilustrar la inclusión de figuras en L^AT_EX.

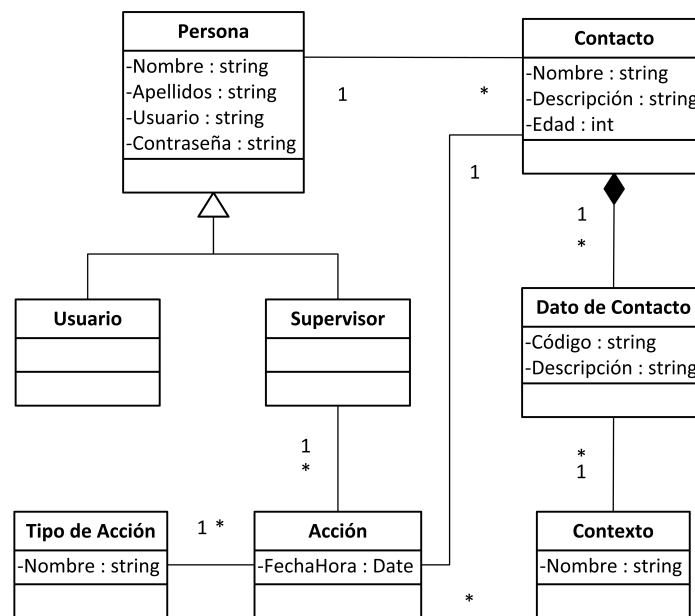


Figura 3.2: Diagrama conceptual

El diagrama conceptual atañe exclusivamente a los requisitos de información, no a los requisitos funcionales. La idea general es “si no se almacena en el sistema, no existe en el diagrama conceptual”. Todas las entidades y atributos de los requisitos de información deben aparecer en el diagrama conceptual, aunque en éste se permiten estrategias como la generalización o las entidades de relación; es decir, es posible que en el diagrama conceptual aparezca más información que en los requisitos de información, pero no a la inversa.

Además de lo indicado, como criterios generales se deben tener en cuenta los siguientes:

- Las líneas no expresan acciones, sino relaciones.
- Las relaciones con otras entidades no se incluyen como atributos en este modelo (posteriormente sí es frecuente), sino que se indica mediante relación.
- Los extremos de las relaciones deben ser cualificados (es decir, tener nombre) si y sólo si el nombre de dicha cualificación difiere del nombre de la entidad. De otro modo, no sólo no es necesario, sino que resulta contraproducente, al incluir en la vista elementos redundantes. Un ejemplo de cualificación necesaria es cuando una entidad tiene más de una relación con otra entidad.
- De igual manera, la propia relación será cualificada cuando no sea evidente (es decir, casos diferentes de “tiene”, “contiene”, “tipifica”, etcétera)..
- La composición se emplea cuando un elemento “se construye” a partir de otro.
- Debe confiarse en la generalización siempre que sea posible.
- Hay que perseguir una estética elegante y legible del modelo, si bien en los diagramas grandes puede resultar más complicado.

- Si no cabe en vertical, se debe rotar la imagen usando `\begin{landscape}` (del paquete `landscape`) en vez de `\begin{figure}[H]`.

3.2.6. Matriz de trazabilidad

A continuación y en la Tabla ?? (pág. ??) se presenta la matriz de trazabilidad entre los requisitos de información y los casos de uso.

| | | IRQ-01 | IRQ-02 | |
|--------|------------------|---------|----------------|--------------------|
| | | Usuario | [Otra entidad] | [Añadir entidades] |
| USC-01 | Autenticación | ✓ | ✓ | |
| USC-02 | Agregar Usuario | ✓ | | |
| USC-03 | Editar Usuario | ✓ | | |
| USC-04 | Eliminar Usuario | ✓ | | |
| USC-05 | Buscar Usuarios | ✓ | | ✓ |

Tabla 3.12: Matriz de trazabilidad

3.3. Requisitos no funcionales

No basta con mencionar los requisitos no funcionales. Debe explicarse la forma en la que van a tratarse, lo cual dependerá del tipo de proyecto. De igual modo, deber referenciarse de dónde se ha obtenido la relación de requisitos no funcionales.

Ejemplos de requisitos no funcionales pueden encontrarse en las normas IEEE Std. 830 [?] e ISO/IEC 25010 (SQuaRE) [?]. De igual modo, aspectos relativos a la legislación y al estilo de programación deben ser descritos.

Texto introductorio.

Requisito no funcional 1

...

Requisito no funcional 2

...

3.4. Resumen del análisis

Resumen ordenado de todos los puntos anteriores, opcionalmente añadiendo consideraciones a discutir.

Capítulo 4

Diseño

Se detalla en este capítulo la expansión del análisis teniendo en cuenta las decisiones técnicas y sus restricciones, constituyendo la arquitectura en la que se basará la fase de implementación. Dicha arquitectura se presenta en su vertiente de ??, ilustrándose el diagrama de equipos e interconexiones necesarias para el funcionamiento del sistema, así como sus especificaciones técnicas; y ??, donde se muestran las capas lógicas del sistema, y los servicios y aplicaciones requeridos. De igual manera, se presentan los ?? que el sistema empleará (a través de estas arquitecturas) para comunicarse con el usuario.

En cuanto a los datos, se amplía el modelo conceptual para transformarlo en un ??, en este proyecto cumpliendo los criterios de una base de datos relacional. De igual manera, los casos de uso se plasman visualmente en una serie de bocetos que ilustran la ?. Por último, se profundiza en los artefactos utilizados en el ??

4.1. Diseño de arquitectura

En esta sección se destalla la ?? y la ?. La primera facilita la descripción técnica del único equipo que la conforma. En cuanto a la ??, se especifican el patrón de arquitectura y el modelo de componentes.

4.1.1. Arquitectura física

Debe facilitarse aquí el esquema detallado y explicativo de los distintos componentes físicos que conforman el sistema, así como sus características relativas a componentes y conectividad.

Los diagramas de arquitectura pueden ser muy complejos, en función del sistema en el que se vaya a utilizar la aplicación. Implican enrutadores, cortafuegos, acceso a Internet, servidores de aplicaciones, servidores de datos, internet, etcétera. Pueden encontrarse varios ejemplos en las imágenes de Google, buscando por “diagrama de arquitectura”, “arquitectura de red” y “*network architecture*”.

4.1.2. Arquitectura lógica

Incluir la arquitectura lógica, tanto a nivel de servicios como en cuanto al patrón de desarrollo, en su caso.

4.2. Códigos de mensajes

Con objeto de que lógica de negocio reconozca los mensajes enviados por el sistema de gestión de bases de datos, deben especificarse aquí los códigos de error de las restricciones automáticas relativos a las ??.

| Regla | Código | Exp. | Mensaje |
|---------|--------|------|--|
| BRU-001 | 2001 | ✓ | El campo '?1' no puede estar vacío. |
| BRU-002 | 1062 | | Entrada duplicada '?1' para la clave '?2'. |

Tabla 4.1: Códigos de mensajes de las reglas de negocio

4.3. Modelo lógico

A continuación, se facilita el diseño de datos a través del modelo lógico, el cual, tras la elección de las ?? (pág. ??) [...]

Indicar las tecnologías escogidas relativas a los datos, así como aspectos como la implementación de la herencia, en su caso, y todo aquello que se considere de interés.

4.3.1. Relación de tablas

A continuación se muestra la relación de tablas del modelo. La descripción incluye el nombre de la tabla, el/los requisito/s de información que representa y los campos que contiene. Para cada campo se indica el nombre, si es clave primaria (si aparece subrayado), el tipo físico, el campo de referencia en su caso, y si es único y obligatorio.

Se incluye una tabla de ejemplo. Con motivo de establecer una serie de pautas comunes que permita acelerar y facilitar la comprensión del modelo, para la descripción de las tablas se respetarán los siguientes criterios:

- Deben eliminarse tildes y espacios, usando la notación Pascal [?].
- El primer elemento en aparecer será la clave primaria.
- Posteriormente, se indican las claves ajenas por orden alfabético.
- Por último, se incluye el resto de los campos en agrupación de significado.

| Tabla | Usuario | | | | |
|-----------|------------------------------|--------------|--------------------|-------|------|
| Requisito | Usuario (Tabla ??) [pág. ??] | | | | |
| Campos | Nombre | Tipo | Campo referenciado | Único | Obl. |
| | <u>Id</u> | int32 | | ✓ | ✓ |
| | EsSupervisor | boolean | | | ✓ |
| | Nombre | varchar(64) | | | ✓ |
| | Apellidos | varchar(128) | | | ✓ |
| | Usuario | varchar(16) | | ✓ | ✓ |
| | Contrasenna | varchar(16) | | | ✓ |

Tabla 4.2: Tabla Usuario

4.3.2. Diagrama lógico/físico de datos

Se presenta a continuación el diagrama que presenta el modelo lógico / físico como una variante simplificada del Diagrama Entidad-Relación [?]. Dicha variante ha sido obtenida de la notación ERwin/ERX2.0 [?, p. 29] y **SSADM!** (SSADM!) [?, p. 193], así como del modo en el que lo generan sistemas de gestión de bases de datos como *Microsoft SQL Server* [?, p. 30], orientado a facilitar la lectura sin pérdida de semántica.

No hay problema en utilizar el diagrama original de Chen o cualquier otro, siempre que esté justificado.

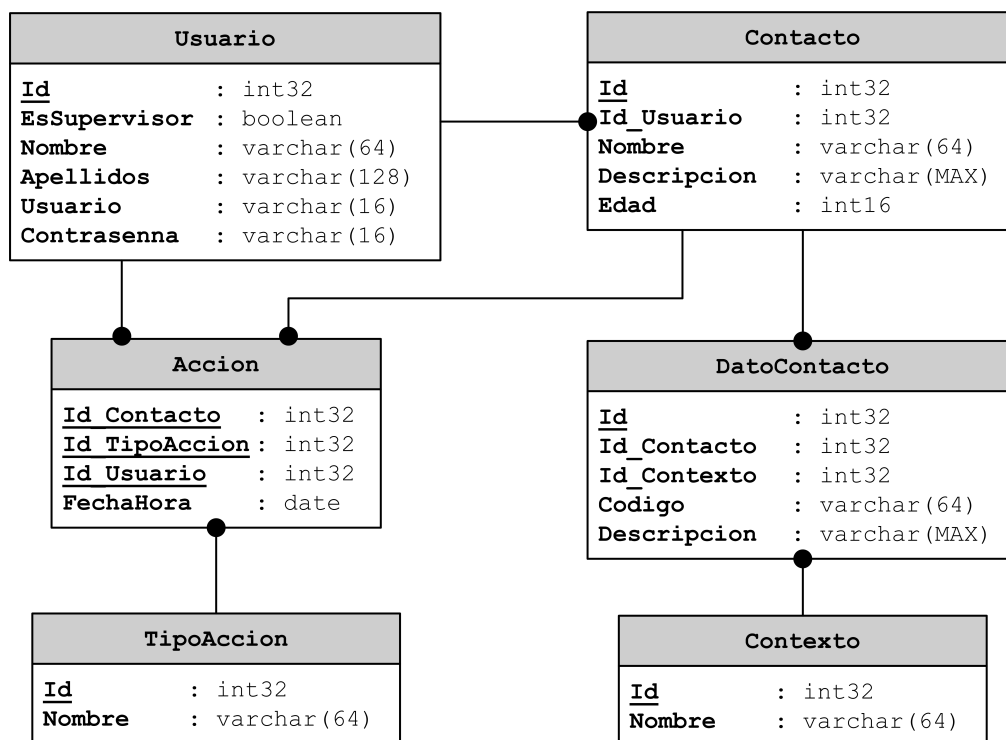


Figura 4.1: Diagrama Entidad-Relación

En caso de bases de datos relaciones, es interesante justificar la forma normal en la que se encuentra, sin entrar para ello en demasiado detalle sobre teoría de bases de datos.

4.4. Interfaz de usuario

A continuación, se muestran los bocetos de la interfaz de usuario. En concreto, se han elaborado los formularios de ?? [...]

Incluir todos los bocetos necesarios, con el programa de composición preferido.

Autenticación

La figura ?? muestra el formulario de autenticación, elaborado a través del diseño esperado a través de la solicitud de usuario y contraseña, así como un botón de acceso.

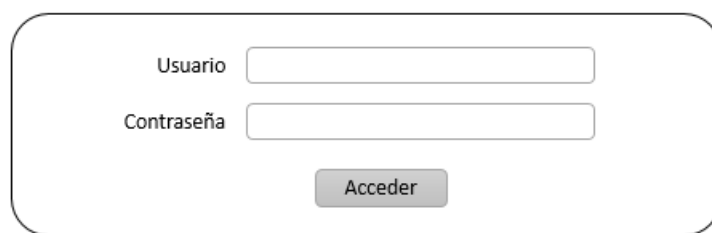
Un boceto de interfaz de autenticación dentro de un recuadro con esquinas redondeadas. Contiene dos campos de entrada de texto: el superior está etiquetado como 'Usuario' y el inferior como 'Contraseña'. Debajo de estos campos hay un botón rectangular con el texto 'Acceder'.

Figura 4.2: Boceto de interfaz de autenticación

4.5. Diseño de componentes

Inclusión del diagrama de componentes y explicación de éstos. En la práctica, el diseño de componentes es la distribución de paquetes y archivos (clases, interfaces, archivos de configuración...).

4.6. Resumen del diseño

El alumno debe indicar aquí la arquitectura, descripción de la lógica, tablas nuevas que aparecen, número de interfaces facilitados y todo aquellos elementos que ayuden a comprobar el tamaño y esfuerzo del diseño.

Capítulo 5

Implementación

Se describen en este capítulo los detalles de implementación. Tras la concreción de los ??, se parte de la ?? para proceder a explicar la su distribución y los elementos importantes del código: la ??, la implementación de la ??, la codificación de aspectos relativos a la ?? y la programación de las ??. Para cerrar el capítulo, se realiza un ??.

Por motivos prácticos, no se revisará todo el código, sino ejemplos representativos de éste. En todo caso, se garantiza que los ejemplos abarcan los aspectos más interesantes de la codificación, quedando fuera de esta memoria aspectos superfluos o repetitivos.

Como se indica, en este capítulo no debe explicarse todo el código, sino debe detallarse la implementación con carácter general. No es necesario ni oportuno plasmar ni desglosar cada clase una a una. También es interesante mencionar aspectos que hayan resultado interesantes o especialmente complicados.

Se observa en este proyecto carencias estéticas como iconos o formularios más profesionales, que es lo que se espera de un trabajo de este tipo. Debido al carácter ilustrativo de este proyecto, se ha obviado esta faceta, cosa que no debe hacer el alumno en su entregable.

5.1. Criterios generales de implementación

Esto en el caso de que se emplee la notación húngara, lo cual debe indicarse en los requisitos no funcionales. De no ser así, deberán indicarse otros criterios, incluyendo la disyuntiva *camel case*/Pascal.

Para ayudar al reconocimiento de la semántica y tipología de las variables en cualquier punto de la codificación se programado respetando un criterio de prefijos basado en notación húngara [?], que se facilita en la Tabla ??.

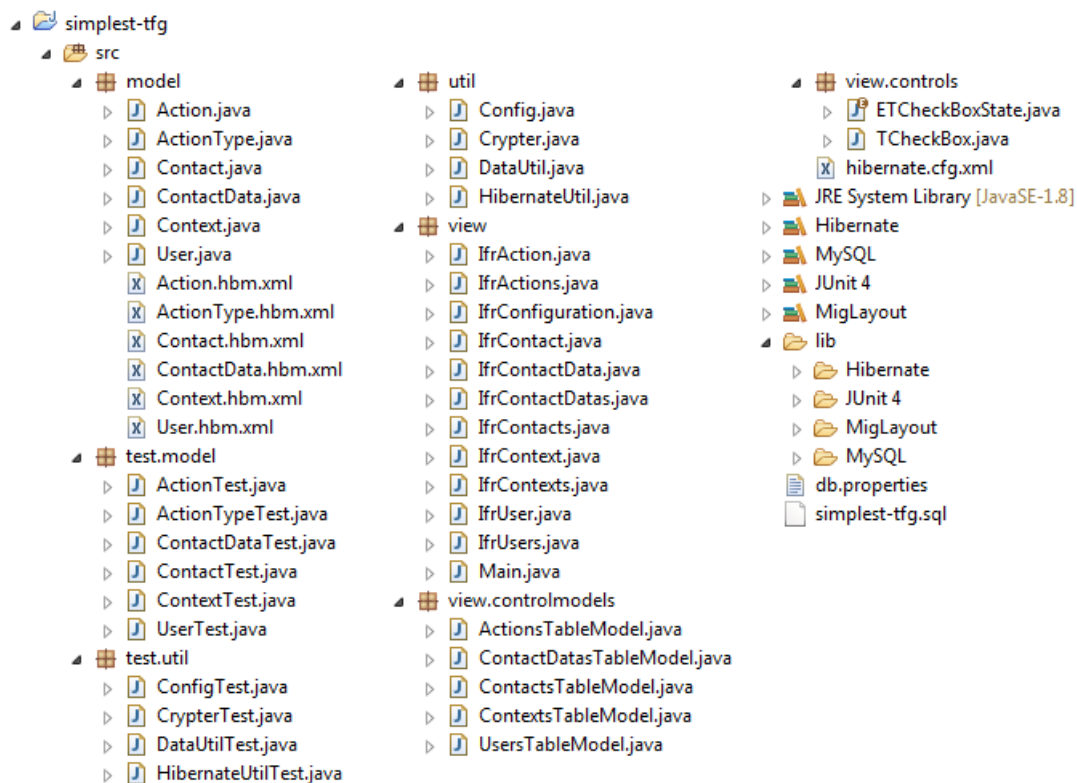
En cuanto a los métodos de acceso, se sigue el criterio habitual de usar el prefijo `get` para lectura y `set` para escritura. Por ejemplo, `String getName()` y `void setName(String sValue)`, respectivamente para la variable `String _sName`. A excepción de estos métodos y del prefijo en notación húngara, se empleará la notación Pascal para nombrar métodos y clases.

| Prefijo | Tipo | Ejemplo |
|---------|-----------------------------------|--|
| i | entero | int iAge; |
| f | decimal | float fMean; |
| d | doble precisión | double dSpeed; |
| b | booleano | boolean bIsValid; |
| s | cadena | String sName; |
| t | fecha/hora | Date tNow; |
| y | byte | byte yLow; |
| | objeto específico | User user; |
| a | vector (cualquier implementación) | List<int> aiAge; |
| _ | variable privada | private int _iId; |
| S | variable estática | private static boolean _SbTest; |
| C | constante (sobrescribe S) | private final static int _CiMaxUsers = 16; |

Tabla 5.1: Notación húngara utilizada

5.2. Estructura del código

La Figura ?? presenta la estructura del código del proyecto [Nombre del proyecto] visto en el entorno de desarrollo *[entorno]*.

Figura 5.1: Estructura del proyecto en *Eclipse*

Debe describirse brevemente la estructura. Asimismo, se aconseja editar la imagen original para que quede similar a la del ejemplo. El motivo es porque las estructuras de carpetas suelen mostrarse de arriba a abajo, ocupando mucho espacio vertical y apenas horizontal.

5.3. Configuración del sistema

El alumno ha de detallar la configuración del sistema debida a las tecnologías subyacentes. Esto incluye código **XML!** (**XML!**), configuración Java o cualquier otro código fuente de parametrización.

5.4. Base de datos

Se ilustra a continuación el código de implementación del modelo de datos en el gestor de bases de datos obtenido del archivo de creación de la base de datos `simplest-tfg.sql`. En concreto, se explican ejemplos de tablas de entidades, relaciones, disparadores y, por último, tablas del sistema.

Para la denominación de índices y disparadores, se ha seguido el criterio mostrado en la Tabla ???. En el caso de los disparadores, $\{B/A\}$ representa *begin* o *after* (uno de los dos), y $\{I/U/D\}$ indica que el disparador es de inserción, actualización o eliminación.

| Elemento | Formato | Ejemplo |
|---------------------|------------------------------------|----------------------------------|
| Índice | <code>IX_Tabla__Campo</code> | <code>IX.Contact__Id_User</code> |
| Índice único | <code>UX_Tabla__Campo</code> | <code>UX.User__Login</code> |
| Integridad | <code>RS_Tabla__Campo</code> | <code>RS.Contact__Id_User</code> |
| Disparador | <code>TR_Tabla_{B/A}{I/U/D}</code> | <code>TR.User_BU</code> |

Tabla 5.2: Criterio de nomenclatura para elementos de la base de datos

5.4.1 Tablas de entidades y relaciones

Mostrar código SQL y explicar ejemplo ilustrativo de un par de tablas referenciadas entre sí. La forma de mostrar código es: `\lstinputlistingContentlanguage=MySQL,caption=MySQL - Tabla \code[Nombre de tabla],firstline=[línea en la que empieza],lastline=[línea en la que termina],firstnumber=137` [archivo SQL en la carpeta LaTeX correspondiente]

5.4.2 Disparadores

Mostrar código SQL de algún disparador de inserción y otro de actualización, relativo a las entidades/reglas de negocio que representan.

5.4.3 Tablas del sistema

Referir y mostrar código SQL de alguna tabla de sistema (por ejemplo, que contenga datos de configuración de la aplicación).

5.5. Seguridad

Debe incorporarse y detallarse aquí información sobre el modelo de seguridad físico y lógico que garantiza la fiabilidad del sistema en cuanto a intrusión.

5.6. Interfaz de usuario

En consecuencia con la Sección ?? del Capítulo ??, se incluirán las implementaciones de cada tipo de formulario. Esto incluye no sólo el aspecto visual, sino la inclusión y descripción del código de la vista y la interfaz con el controlador, en caso de seguir el **MVC!** (**MVC!**), o cualquier otro código de interés.

5.7. Resumen de la implementación

El alumno debe indicar aquí número de clases, líneas de código, archivos de configuración, etcétera, añadiendo que se ha explicado de forma detallada el funcionamiento e implementación general, de seguridad, de configuración, etcétera, y emplazándose al código fuente entregado con la memoria para el resto del código.

Capítulo 6

Pruebas

En este capítulo se documentan los diferentes tipos de pruebas que se han llevado a cabo durante el desarrollo del sistema. En primer lugar, se describe la ?? y el ??. Posteriormente, se detallan las diferentes fases referidas: ??, que comprueba el funcionamiento independiente de los distintos componentes de la aplicación; ??, que verifica que el ensamblado entre dichos componentes es correcto; ??, donde se confronta los requisitos del modelo con la operatividad del software; y ??, cuyo objetivo es asegurar que el sistema funciona correctamente en explotación.

6.1. Estrategia de pruebas

Introducción a las pruebas (con las referencias necesarias) y explicación de la estrategia a seguir.

6.2. Entorno de pruebas

El mismo entorno y equipo de desarrollo descrito en la etapa de ?? (pág. ??) se ha empleado para las ?? y ??. Para las ?? y ??, se empleó el equipo del usuario final.

6.3. Pruebas unitarias

6.3.1. Base de datos

Los párrafos añadidos son, como se ha dicho en alguna ocasión propios del autor y del proyecto. Deberán ser modificados cuando y como corresponda.

Se obvian las comprobaciones directas sobre posibles tipos erróneos, campos nulos no permitidos e incumplimiento de integridad referencial, ya que forman parte de la propia gestión y configuración de la base de datos, no de la programación. De otra forma, se requeriría probar por cada campo y acción (inserción, actualización y eliminación) todos los tipos posibles, generando una explosión innecesaria de casos de pruebas. Por tanto, una vez comprobada la adecuada configuración de las tablas, sólo se realizarán pruebas sobre código implementado en

desencadenadores y procedimientos almacenados. Dichas pruebas se han realizado con las tablas de entidades vacías. En caso de que tengan información, los resultados pueden diferir de los presentados por conflicto de datos¹.

6.3.2 Tabla User

| TST-01 | Regla | - | Tabla | User | Acción | C | Validado | ✓ |
|-------------|--|---|-------|------|--------|---|----------|---|
| Descripción | Una inserción con valores válidos se realiza adecuadamente. | | | | | | | |
| Código | SELECT COUNT(*) FROM User; INSERT INTO User (IsSupervisor, Name, Surname, Login, Password) VALUES (TRUE, 'Bruce', 'Wayne', 'batm', '12345'); SELECT COUNT(*) FROM User; SELECT * FROM User ORDER BY Id DESC LIMIT 1; | | | | | | | |
| Resultado | 0 1 1, TRUE, 'Bruce', 'Wayne', 'batm', '12345' | | | | | | | |

Tabla 6.1: TST-01 - Inserción válida en tabla User

A título ilustrativo, se presentan únicamente un caso de prueba de **User**. En un entregable final, deben incluirse todos los casos para todas las tablas.

6.3.3. Aplicación

En este ejemplo se presenta únicamente la clase de prueba de **User** como representante de otras clases de la lógica.

Código 6.1: Java - Clase **UserTest** de pruebas unitarias para clase **User**

```

1 package test.model;
2
3 import java.sql.SQLException;
4 import java.util.List;
5
6 import org.hibernate.Query;
7 import org.hibernate.Session;
8 import org.hibernate.Transaction;
9 import org.junit.AfterClass;
10 import org.junit.Assert;
11 import org.junit.BeforeClass;
12 import org.junit.FixMethodOrder;
13 import org.junit.Test;
14 import org.junit.runners.MethodSorters;
15
16 import model.Contact;
```

¹Por ejemplo, si se ejecuta dos veces seguidas la prueba *TST-001*, el gestor de bases de datos generará un error debido a que el campo **Login** de la tabla **User** se configuró como índice único.

```

17 import model.User;
18 import util.HibernateUtil;
19
20 @SuppressWarnings({"unchecked", "rawtypes", "deprecation"})
21 @FixMethodOrder(MethodSorters.NAME_ASCENDING)
22 public class UserTest {
23
24     private static User _Suser = new User(false, "Peter", "Parker", "spid",
25         "67890");
26     private static Session _ShibernateSession;
27
28     @BeforeClass
29     public static void setUpBeforeClass() throws Exception {
30         _ShibernateSession = HibernateUtil.OpenSession(false);
31     }
32
33     @AfterClass
34     public static void tearDownAfterClass() throws Exception {
35         String sHql = "from User where Login = '" + _Suser.getLogin() + "'";
36         Query query = _ShibernateSession.createQuery(sHql);
37         List<User> aUser = query.setMaxResults(1).list();
38
39         if (!aUser.isEmpty()) {
40             Transaction tx = _ShibernateSession.beginTransaction();
41             _ShibernateSession.delete(aUser.get(0));
42             tx.commit();
43         }
44         _ShibernateSession.close();
45     }
46
47     @Test
48     public void t1_UserNotFound() {
49
50         Session session = _ShibernateSession;
51
52         String sHql = "from User where Login = '" + _Suser.getLogin() + "'";
53         Query query = session.createQuery(sHql);
54         List<User> aUser = query.list();
55
56         Assert.assertEquals(0, aUser.size());
57         Assert.assertEquals(true, aUser.isEmpty());
58     }
59
60     @Test
61     public void t2_InsertUser() {
62
63         Contact contactCheck = new Contact(_Suser, "Gwen", "First girlfriend.", 22);
64         _Suser.getContacts().add(contactCheck);
65
66         Session session = _ShibernateSession;
67
68         Transaction tx = session.beginTransaction();
69         // session.save(contactCheck) isn't needed, as it's configured as
70         // cascade="all" in User.hbl.xml.
71         session.save(_Suser);
72         tx.commit();

```

```
73     String sHql = "from User where Login = '" + _Suser.getLogin() + "'";
74     Query query = session.createQuery(sHql);
75     List<User> aUser = query.list();
76
77     Assert.assertEquals(1, aUser.size());
78     Assert.assertEquals(_Suser, aUser.get(0));
79     Assert.assertEquals(_Suser.getContacts(), aUser.get(0).getContacts());
80 }
81
82 @Test
83 public void t3_InsertDuplicateUser() {
84
85     Session session = _ShibernateSession;
86     Transaction tx = null;
87
88     User user = new User(_Suser.getIsSupervisor(), _Suser.getName(),
89         _Suser.getSurname(), _Suser.getLogin(), _Suser.getPassword());
90
91     try {
92         tx = session.beginTransaction();
93         session.save(user);
94         tx.commit();
95     } catch (javax.persistence.PersistenceException ee) {
96         SQLException eeSql =
97             ((org.hibernate.JDBCException)ee.getCause()).getSQLException();
98         Assert.assertEquals(1062, eeSql.getErrorCode());
99         tx.rollback();
100     } catch (Exception ee) {
101         if (tx != null) tx.rollback();
102     }
103 }
```

La Figura ?? (pág. ??) muestra el resultado de la ejecución del conjunto de pruebas unitarias del archivo `UserTest.java`.

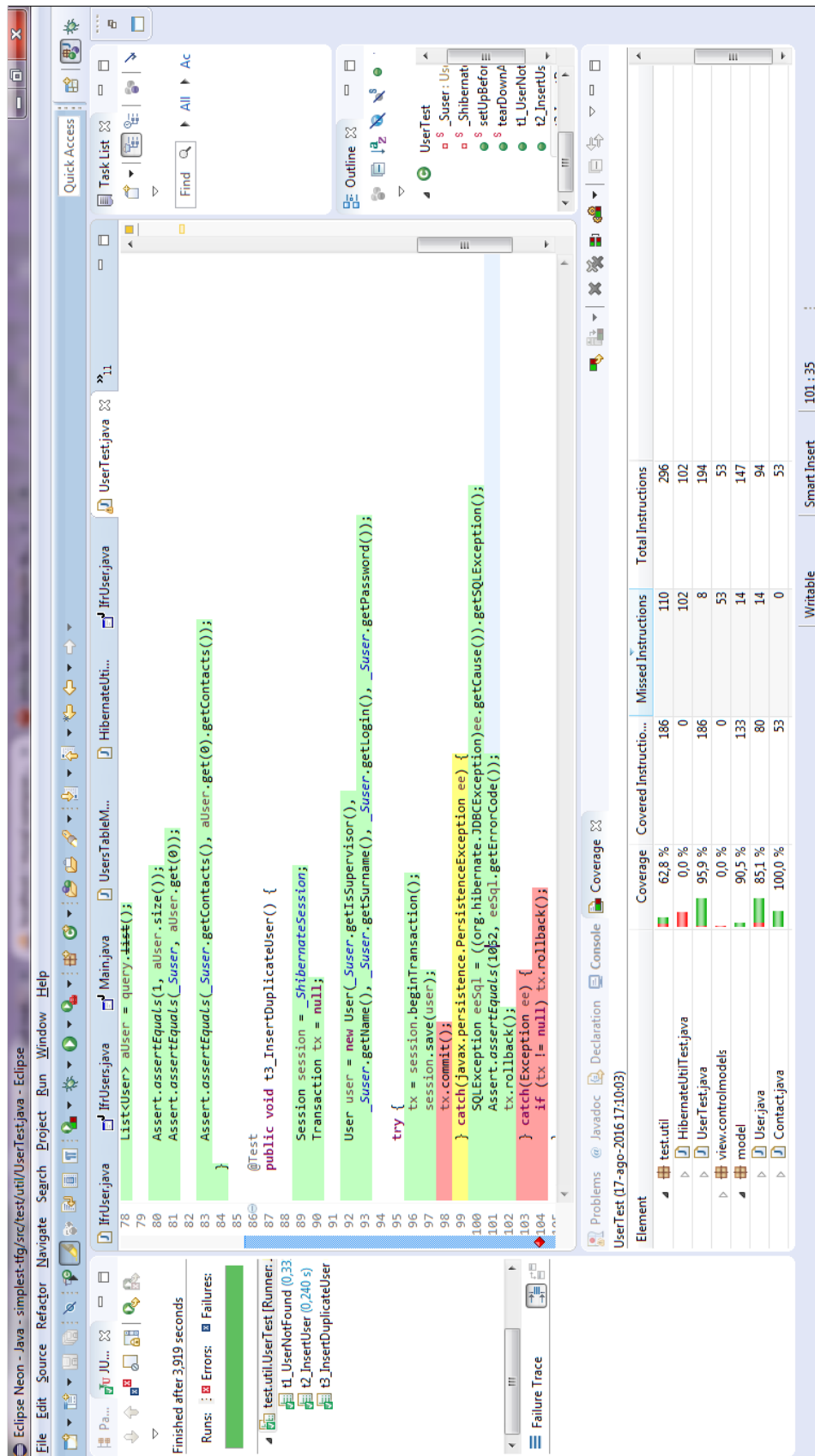


Figura 6.1: Ejemplo de pruebas unitarias con cobertura de código sobre la clase User

6.4. Pruebas de integración

Plasmar de forma similar a otras pruebas o justificar que no son necesarias.

6.5. Pruebas de validación

Especificar número de pruebas, cuántos participantes, cuál fue el guión, cuántas iteraciones, qué porcentaje de pruebas fueron validadas, qué percepción tenían los clientes de la herramienta, qué cambios habían propuesto, si los cambios suponían mantenimiento correctivo o ampliación de requisitos...

6.6. Pruebas de sistema

Plasmar de forma similar a las pruebas de validación.

6.7. Resumen de las pruebas

El alumno debe indicar aquí número de pruebas de cada tipo, número de clases, usuarios que realizaron las pruebas de validación y resultados generales.

Parte III

Epílogo

Capítulo 7

Conclusiones

Como capítulo final en la elaboración del proyecto, se describen los ?? del mismo y el ??. Asimismo, se comparten las ??.

7.1. Resultados

Para la elaboración de este capítulo, se aconseja acudir a las Secciones ?? y ?? del Capítulo ??, y referir la consecución de éstos. Igualmente, debe añadirse un breve resumen de los procesos y tecnologías empleadas, para lo cual puede usarse (mediante la correspondiente edición) el resumen de cada capítulo.

7.2. Trabajo futuro

Deben incluirse potenciales mejoras del sistema, ya sea por nuevas funcionalidades o por optimización del código existente.

7.3. Lecciones aprendidas

Sección personal que depende de las intenciones, conocimientos previos y experiencia del alumno. Además de las reflexiones sobre lo referido, generalmente suele remitirse la satisfacción por la finalización del desarrollo. También se espera alguna mención a la formación académica recibida, detallándose en lo posible el conocimiento que mejor ha influido en la consecución del **TFG!**.

Apéndices

Apéndice A: Manual de usuario

El alumno deberá incluir aquí el manual de usuario en términos comprensibles para el cliente.