



**UNIVERSIDAD DE CASTILLA-LA MANCHA  
ESCUELA SUPERIOR DE INFORMÁTICA**

**GRADO EN INGENIERÍA INFORMÁTICA**

**Tecnología específica de computación**

**TRABAJO FIN DE GRADO**

**Desarrollo de un Asistente Virtual para Simulación de  
Conducción Deportiva:  
Un Enfoque en Telemetría Comparativa**

**Adrián Ramos Rodríguez-Palmero**

**julio, 2024**





**UNIVERSIDAD DE CASTILLA-LA MANCHA  
ESCUELA SUPERIOR DE INFORMÁTICA**

**Departamento de tecnologías  
y sistemas de información**

**Tecnología específica de computación**

**TRABAJO FIN DE GRADO**

**Desarrollo de un Asistente Virtual para Simulación de  
Conducción Deportiva:  
Un Enfoque en Telemetría Comparativa**

Autor: Adrián Ramos Rodríguez-Palmero

Tutor: Luis Jiménez Linares

julio, 2024



Asistente Virtual para Simulación de Conducción Deportiva  
© Adrián Ramos Rodríguez-Palmero, 2024

Este documento se distribuye con licencia CC BY-NC-SA 4.0. El texto completo de la licencia se puede obtener en <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

La copia y distribución de esta obra está permitida en todo el mundo, sin regalías y por cualquier medio, siempre que esta nota sea preservada. Se concede permiso para copiar y distribuir traducciones de este libro desde el español original a otro idioma, siempre que la traducción sea aprobada por el autor del libro y tanto el aviso de copyright como esta nota de permiso, sean preservados en todas las copias.

Este texto ha sido preparado con la plantilla L<sup>A</sup>T<sub>E</sub>X para Trabajo Fin de Estudios en Ingeniería Informática para la UCLM publicada por [Jesús Salido](#) en el repositorio público Zenodo, DOI: [10.5281/zenodo.4561708](https://doi.org/10.5281/zenodo.4561708), como parte del curso «[L<sup>A</sup>T<sub>E</sub>X esencial para preparación de TFG, tesis y otros documentos académicos](#)» impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha. Si la empleas para preparar tu TFG, te agradeceré que la cites [8] e incluyas en tus referencias como se indica en Zenodo con el DOI suministrado para todas las versiones.





**TRIBUNAL:**

Presidente: \_\_\_\_\_

Vocal: \_\_\_\_\_

Secretario(a): \_\_\_\_\_

**FECHA DE DEFENSA:** \_\_\_\_\_

**CALIFICACIÓN:** \_\_\_\_\_

PRESIDENTE

VOCAL

SECRETARIO(A)

Fdo.:

Fdo.:

Fdo.:



*A mi familia  
Por su apoyo incondicional*



## **Asistente Virtual para Simulación de Conducción Deportiva**

Adrián Ramos Rodríguez-Palmero  
Ciudad Real, julio 2024

### **Resumen**

Los simuladores han experimentado una evolución significativa, pasando de ser modelos simplificados de la realidad a sistemas complejos que reflejan con precisión los sistemas que simulan. Esta capacidad de imitar casi a la perfección los sistemas simulados convierte a estos simuladores en herramientas eficaces para el aprendizaje y el entrenamiento.

Un campo en el que la popularidad de estos simuladores ha crecido de manera notable es en la conducción deportiva, también conocida como simracing. En estos sistemas, el piloto utiliza un conjunto de hardware que simula el vehículo, la pista y las leyes físicas que rigen las interacciones, permitiendo realizar sesiones de entrenamiento a un coste muy reducido.

El proceso de entrenamiento y aprendizaje puede ser autónomo, donde el piloto observa las diferentes telemetrías que proporciona el simulador e intenta mejorarlas, o puede ser asistido por un entrenador que aporta el conocimiento y la experiencia necesarios para indicar las mejoras a realizar.

Este trabajo se sitúa en este contexto con el objetivo de definir y construir un asistente o entrenador virtual de simracing. Este asistente será capaz de interpretar de manera comparativa la telemetría de un experto frente a la del piloto en entrenamiento, proporcionando planes de mejora y realizando un seguimiento de dicha mejora.

**Palabras clave:** TFG, Simulación, Telemetría, Simracing, UCLM



## **Virtual Assistant for Sports Driving Simulation**

Adrián Ramos Rodríguez-Palmero  
Ciudad Real, July 2024

### **Abstract**

Simulators have undergone a significant evolution from simulated models of reality to complex systems that accurately reflect the systems they simulate. This ability to mimic simulated systems almost perfectly makes these simulators effective tools for learning and training.

One area where the popularity of these simulators has grown significantly is in sports driving, also known as simracing. In these systems, the driver uses a set of hardware that simulates the vehicle, the track and the physical laws that govern the interactions, allowing training sessions to be carried out at a very low cost.

The training and learning process can be autonomous, where the driver observes the different telemetries provided by the simulator and tries to improve them, or can be assisted by a trainer who provides the necessary knowledge and experience to indicate the improvements to be made.

This work is placed in this context with the objective of defining and building a virtual simracing assistant or trainer. This assistant will be able to interpret in a comparative way the telemetry of an expert versus that of the driver in training, providing improvement plans and monitoring the improvement.

**Keywords:** TFG, Simulation, Telemetry, Simracing, UCLM



# Agradecimientos

---

Aunque es un apartado opcional, haremos bueno el refrán «*es de bien nacidos, ser agradecidos*» si empleamos este espacio como un medio para agradecer a todos los que, de un modo u otro, han hecho posible que el trabajo realizado *llegue a buen puerto*. Esta sección es ideal para agradecer a directores, profesores, mentores, familiares, compañeros, amigos, etc.

Estos agradecimientos pueden ser tan personales como se desee e incluir anécdotas y chascarrillos, pero recuerda que *no deberían ocupar más de una página*.

Adrián Ramos Rodríguez-Palmero  
Ciudad Real, 2024



# Notación y acrónimos

---

## NOTACION

Ejemplo de lista con notación (o nomenclatura) empleada en la memoria del TFG.<sup>1</sup>

$A, B, C, D$	:	Variables lógicas
$f, g, h$	:	Funciones lógicas
$\cdot$	:	Producto lógico (AND), a menudo se omitirá como en $AB$ en lugar de $A \cdot B$
$+$	:	Suma aritmética o lógica (OR) dependiendo del contexto
$\oplus$	:	OR exclusivo (XOR)
$\bar{A}$ o $A'$	:	Operador NOT o negación

## LISTA DE ACRÓNIMOS

Ejemplo de lista *ordenada alfabéticamente* con los acrónimos empleados en el texto.<sup>2</sup>

CASE	:	Computer-Aided Software Engineering
CTAN	:	Comprehensive TeX Archive network
IDE	:	Integrated Development Environment
ECTS	:	European Credit Transfer and Accumulation System
OOD	:	Object-Oriented Design
PhD	:	Philosophiae Doctor
RAD	:	Rapid Application Development
SDLC	:	Software Development Life Cycle
SSADM	:	Structured Systems Analysis & Design Method
TFE	:	Trabajo Fin de Estudios
TFG	:	Trabajo Fin de Grado
TFM	:	Trabajo Fin de Máster
UML	:	Unified Modeling Language

---

<sup>1</sup>Se incluye únicamente con propósito de ilustración, ya que el documento no emplea la notación aquí mostrada.

<sup>2</sup>Se pueden omitir aquellos acrónimos que son reconocidos en el contexto académico (p. ej., PhD), aunque aquí se han incluido a efectos ilustrativos.



# Índice general

---

<b>Resumen</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>Agradecimientos</b>	<b>ix</b>
<b>Notación y acrónimos</b>	<b>xi</b>
<b>Índice de figuras</b>	<b>xv</b>
<b>Índice de tablas</b>	<b>xvii</b>
<b>Índice de listados</b>	<b>xix</b>
<b>Índice de algoritmos</b>	<b>xxi</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Contexto disciplinar y tecnológico . . . . .	1
1.3. Estructura del documento . . . . .	2
<b>2. Objetivo</b>	<b>3</b>
2.1. Cómo plantear el objetivo de un TFG en ingeniería . . . . .	3
<b>3. Plan de gestión del trabajo</b>	<b>5</b>
3.1. Guía rápida de las metodologías de desarrollo de software . . . . .	5
3.2. Otros aspectos del plan de gestión . . . . .	7
3.3. Tecnologías . . . . .	8
<b>4. Desarrollo</b>	<b>11</b>
4.1. Requisitos y análisis del sistema . . . . .	11
4.2. Diseño del sistema . . . . .	11
4.3. Implementación del sistema . . . . .	12
4.4. Pruebas del sistema . . . . .	12
4.5. Despliegue . . . . .	12
<b>5. Conclusiones</b>	<b>13</b>
5.1. Objetivos alcanzados . . . . .	13
5.2. Justificación de competencias adquiridas . . . . .	13
5.3. Trabajos derivados y futuros . . . . .	13
5.4. Valoración personal . . . . .	13

<b>Bibliografía</b>	<b>15</b>
<b>A. Sobre la Bibliografía</b>	<b>19</b>
<b>B. Breve introducción a L<sup>A</sup>T<sub>E</sub>X</b>	<b>21</b>
B.1. Listas . . . . .	21
B.2. Ecuaciones matemáticas . . . . .	21
B.3. Tablas . . . . .	22
B.4. Figuras . . . . .	23
B.5. Algoritmos y listados de código fuente . . . . .	27
B.6. Menús, paths y teclas con el paquete menukeys . . . . .	28

# Índice de figuras

---

B.1.	Ejemplo de figura . . . . .	23
B.2.	Ejemplo de subfiguras . . . . .	23
B.3.	Gráfico girado . . . . .	25
B.4.	Gráfico apaisado . . . . .	26



# Índice de tablas

---

B.1. Ejemplo de uso de la macro <code>cline</code> . . . . .	22
B.2. Ejemplo de tabla con especificación de anchura de columna . . . . .	22



# Índice de listados

---

B.1. Código fuente en Java . . . . .	28
B.2. Ejemplo de código fuente en lenguaje C . . . . .	28
B.3. Ejemplo de script en Matlab . . . . .	28



# **Índice de algoritmos**

---

B.1. Cómo escribir algoritmos . . . . .	27
---	----



---

# CAPÍTULO 1

## Introducción

---

Este capítulo aborda la motivación del trabajo. Se trata de señalar la necesidad que lo origina, su actualidad y pertinencia. Puede incluir también un estado de la cuestión (o estado del arte) en el que se revisen estudios o desarrollos previos, explicando en qué medida sirven de base al trabajo que se presenta. La estructura del capítulo podría quedar reflejada en las secciones que se indica.

### 1.1. MOTIVACIÓN

Responde a la pregunta sobre la necesidad o pertinencia del trabajo.

En particular para este documento, se ha tenido presente que durante la realización de la memoria del TFG es muy importante respetar la guía de estilo de la institución [4]. Por tanto, el empleo de plantillas para un sistema de procesamiento de textos (p. ej., Word o L<sup>A</sup>T<sub>E</sub>X) puede requerir su adaptación cuando la plantilla mencionada no haya sido suministrada por la institución a la que se dirige el trabajo.

Es muy posible que durante la exposición en este capítulo sea preciso enunciar de un modo conciso el objetivo general del trabajo. Sin embargo, este breve enunciado no debe evitar la necesidad de incluir una sección o incluso un capítulo –como en esta plantilla– dedicado a explicar en mayor detalle el objetivo general y los secundarios (también denominados específicos) del TFG.

### 1.2. CONTEXTO DISCIPLINAR Y TECNOLÓGICO

También se puede denominar «Antecedentes» o «Estado del arte» cuando se trata de comentar trabajos relacionados que han abordado la cuestión u objetivo planteado.

En esta sección se debería introducir el *contexto disciplinar y tecnológico* en el que se desarrolla el trabajo de modo que ayude a entender con facilidad su ámbito y alcance.

Puesto que un TFG no tiene que ser necesariamente un trabajo con aportes novedosos u originales, solo es necesario la inclusión de *estado del arte* cuando este contribuya a aclarar aspectos clave del TFG o se deseé justificar la originalidad del trabajo realizado.

Para redactar un trabajo académico de modo efectivo se recomienda seguir pautas para obtener un resultado final claro y de lectura fácil, como las expuestas en el blog de Leonor Zozaya [15] o el apartado de comunicación eficaz del Departamento de Lengua y Estilo de la UOC [12].

A la hora de redactar el texto se debe poner especial atención para evitar el plagio respetando los derechos de propiedad intelectual [10]. En particular, merece gran atención la inclusión de gráficos e imágenes procedentes de Internet que no sean de elaboración propia. En este sentido se sugiere la consulta del manual de la Universidad de Cantabria [11]. Dicho documento explica de modo conciso cómo incluir imágenes en un trabajo académico de modo apropiado.

### 1.3. ESTRUCTURA DEL DOCUMENTO

Este capítulo suele finalizar con una sección en la que se indica la estructura (capítulos) del documento y el contenido de cada una de las partes en que se divide. Veamos a continuación cómo sería esta sección para este documento en concreto.

A lo largo de los capítulos que componen esta guía se muestran ejemplos de elementos de organización del texto en un documento preparado con  $\text{\LaTeX}$ . Todos estos ejemplos se explican en detalle durante el curso de  $\text{\LaTeX}$  esencial para TFG impartido por J. Salido [7]. Los ejemplos mencionados, así como los recogidos en obras de referencia, se pueden emplear para adaptar este documento a las necesidades particulares [5, 14]. Entre las obras de consulta disponibles sobre  $\text{\LaTeX}$  se recomienda el uso de las obras gratuitas en español [6, 2] y las guías disponibles en la página web de [Overleaf](#) (en inglés).

En esta plantilla de TFG se ha optado por seguir la estructura que debería presentar un TFG en la ESI-UCLM. Esta estructura consta de los capítulos siguientes:

1. **Introducción.** Donde se trata la motivación y se justifica la pertinencia del trabajo. Prosigue con el enunciado conciso del propósito material del trabajo y la descripción del contexto disciplinar y técnico de su abordaje.
2. **Objetivo.** En el que se detalla el alcance del objetivo general y los secundarios que se persiguen con la realización del trabajo.
3. **Plan de gestión del trabajo.** Describe todos los aspectos relacionados con la estrategia para abordar las distintas fases del trabajo.
4. **Desarrollo.** Donde se explica cómo se han llevado a cabo las fases del trabajo cumpliendo el plan previsto.
5. **Conclusiones.** En el que se realiza una discusión sobre los objetivos alcanzados y la justificación de la aplicación de las competencias adquiridas durante los estudios culminados con el TFG. También puede incluir una explicación sobre los trabajos derivados y futuros, así como una breve valoración personal.
6. **Bibliografía.** Lista de las referencias bibliográficas citadas en el texto.
7. **Anexos.** Contenidos auxiliares que complementan del trabajo.

---

## CAPÍTULO 2

# Objetivo

---

Este es un capítulo en el que se determina de modo claro el objetivo general del trabajo descrito que se puede desglosar en objetivos secundarios cuando el objetivo principal admite una descomposición en módulos o componentes.

Es muy importante definir el objetivo de modo apropiado. Debe concretar y exponer detalladamente el problema a resolver, el entorno de trabajo, la situación y qué se pretende obtener. También puede contemplar las limitaciones y condicionantes a considerar para la resolución del problema (lenguaje de construcción, equipo físico, equipo lógico de base o de apoyo, etc.).

### 2.1. CÓMO PLANTEAR EL OBJETIVO DE UN TFG EN INGENIERÍA

Una de las tareas más complicadas al proponer un TFG es plantear su Objetivo. La dificultad deriva de la falta de consenso respecto de lo que se entiende por *objetivo* en un trabajo de esta naturaleza. En primer lugar, se debe distinguir entre dos tipos de objetivo:

- (A) La *finalidad específica* del TFG que se plantea para resolver una problemática concreta aplicando los métodos y herramientas adquiridos durante la formación académica. Por ejemplo, «*Desarrollo de una aplicación software para gestionar reservas hoteleras on-line*».
- (B) El *propósito académico* que la realización de un TFG tiene en la formación de un/a graduado/a. Por ejemplo, demostración de la adquisición de las competencias *específicas de la especialización* cursada y de aquellas *competencias transversales* ligadas a la realización del TFG.

En el ámbito de la memoria del TFG se tiene que definir el primer tipo de objetivo, mientras que el segundo tipo es el que se añade al elaborar la propuesta de un TFG presentada ante un comité para su aprobación. *Este segundo tipo de objetivo no se debe incluir en la memoria y en todo caso hacerse en la sección de conclusiones finales.*<sup>1</sup>

Un objetivo bien planteado debe estar determinado en términos del «*producto final*» esperado que resuelve un problema específico. Por tanto, debería quedar determinado por un sustantivo *concreto y medible*. El objetivo planteado puede pertenecer a una de las categorías que se indica a continuación:

- *Diseño y desarrollo de «artefactos»* (habitual en las ingenierías). Por la naturaleza de los programas informáticos (software), los trabajos que implican su diseño suelen contemplar también el desarrollo o implementación de prototipos. Esto es menos frecuente en otras áreas de ingeniería en las que claramente se separa la fase de diseño o realización de un proyecto, frente a la ejecución del mismo (p. ej., ingeniería civil, arquitectura, etc.).
- *Estudio* que ofrece información novedosa sobre un tema (usual en las ramas de ciencias y humanidades).
- *Validación de una hipótesis* de partida (propio de los trabajos científicos y menos habitual en el caso de los TFG).

---

<sup>1</sup>En algunas titulaciones es obligatorio que la memoria explique las competencias específicas alcanzadas con la realización del trabajo.

Estas categorías no son excluyentes, de modo que es posible plantear un trabajo cuyo objetivo sea el diseño y desarrollo de un «artefacto» y este implique un estudio previo o la validación de alguna hipótesis para guiar el proceso. En este caso y cuando el objetivo sea lo suficientemente amplio puede ser conveniente su descomposición en elementos más simples hablando de *objetivos secundarios* o *subobjetivos*. Por ejemplo, un programa informático se puede descomponer en módulos o requerir un estudio previo para plantear un nuevo algoritmo que será preciso validar. La descomposición de un objetivo principal en subobjetivos u objetivos secundarios debería ser natural (no forzada), bien justificada y sólo pertinente en los trabajos de gran amplitud.

Junto con la definición del objetivo del trabajo se puede especificar los *requisitos* que debe satisfacer la solución aportada. Estos requisitos especifican *características* que debe poseer la solución y *restricciones* que acotan su alcance. En el caso de un trabajo cuyo objetivo es el desarrollo de un «artefacto» los requisitos pueden ser *funcionales* y *no funcionales*.

Al redactar el objetivo de un TFG se puede confundir los medios con el fin. De este modo es posible encontrarse con objetivos definidos en términos de las *acciones* (verbos) o *tareas* que será preciso realizar para alcanzar el resultado deseado. Aunque deben evitarse en la definición del objetivo del TFG, a la hora de planificar el desarrollo del trabajo, si es apropiado descomponer su elaboración en *hitos* y estos en *tareas* que faciliten su *planificación*.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria del trabajo. De este modo, en el caso de estudios y validación de hipótesis, el apartado de conclusiones debería incluir los resultados de experimentación y los comentarios de cómo estos validan o refutan la hipótesis planteada.

---

## CAPÍTULO 3

# Plan de gestión del trabajo

---

En este capítulo se debe detallar todos los aspectos relacionados con el plan de gestión del trabajo que incluyen:

- la metodología de desarrollo,
- las tecnologías y recursos necesarios,
- la gestión de la configuración y el aseguramiento de la calidad,
- la planificación del trabajo, y
- la estimación de costes y el análisis de riesgos.

### 3.1. GUÍA RÁPIDA DE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE

El **proceso de desarrollo de software** se denomina también **ciclo de vida del desarrollo del software** (*SDLC, Software Development Life-Cycle*) y cubre las siguientes actividades:

- 1.- **Obtención y análisis de requisitos** (*requirements analysis*). Es la definición de la funcionalidad del software a desarrollar. Suele requerir entrevistas entre los ingenieros de software y el cliente para obtener el ‘QUÉ’ y ‘CÓMO’. Permite obtener una *especificación funcional* del software.
- 2.- **Diseño** (*SW design*). Consiste en la definición de la arquitectura, los componentes, las interfaces y otras características del sistema o sus componentes.
- 3.- **Implementación** (*SW construction and coding*). Es el proceso de codificación del software en un lenguaje de programación. Constituye la fase en que tiene lugar el desarrollo de software.
- 4.- **Pruebas** (*testing and verification*). Verificación del correcto funcionamiento del software para detectar fallos lo antes posible. Persigue la obtención de software de calidad. Consisten en pruebas de *caja negra* y *caja blanca*. Las primeras comprueban que la funcionalidad es la esperada y para ello se verifica que ante un conjunto amplio de entradas, la salida sea correcta. Con las segundas se comprueba la robustez del código sometiéndolo a pruebas cuya finalidad es provocar fallos de software. Esta fase también incorpora la *pruebas de integración* en las que se verifica la interoperabilidad del sistema con otros existentes.
- 5.- **Documentación** (*documentation*). Persigue facilitar la mejora continua del software y su mantenimiento.
- 6.- **Despliegue** (*deployment*). Consiste en la instalación del software en un entorno de producción y puesta en marcha para explotación. En ocasiones implica una fase de *entrenamiento* de los usuarios del software.
- 7.- **Mantenimiento** (*maintenance*). Su propósito es la resolución de problemas, mejora y adaptación del software en explotación.

#### 3.1.1. Metodologías de desarrollo software

*Las metodologías son el modo en que las fases del proceso de desarrollo de software se organizan e interaccionan para conseguir que dicho proceso sea reproducible y predecible para aumentar la productividad*

y la calidad del software.

Una metodología es una colección de:

- A. **Procedimientos** (indican cómo hacer cada tarea y en qué momento),
- B. **Herramientas** (ayudas para la realización de cada tarea), y
- C. **Ayudas documentales**.

Cada metodología es apropiada para un tipo de proyecto dependiendo de sus características técnicas, organizativas y del equipo de trabajo. En los entornos empresariales es obligado, a veces, el uso de una metodología concreta (p. ej., para participar en concursos públicos). El estándar internacional ISO/IEC 12270 describe el método para seleccionar, implementar y monitorear el ciclo de vida del software [13, 1].

Mientras que unas metodologías intentan sistematizar y formalizar las tareas de diseño, otras aplican técnicas de gestión de proyectos para dicha tarea. Las metodologías de desarrollo se pueden agrupar dentro de varios enfoques según se señala a continuación.

1. **Metodología de Análisis y Diseño de Sistemas Estructurados** (*SSADM, Structured Systems Analysis and Design Methodology*). Es uno de los paradigmas más antiguos. En esta metodología se emplea un modelo de desarrollo en cascada (*waterfall*). Las fases de desarrollo tienen lugar de modo secuencial. Una fase comienza cuando termina la anterior. Es un método clásico poco flexible y adaptable a cambios en los requisitos. Hace especial hincapié en la planificación derivada de una exhaustiva definición y análisis de los requisitos. Son metodologías que no lidian bien con la flexibilidad requerida en los proyectos de desarrollo software. Derivan de los procesos en ingeniería tradicionales y están enfocadas a la reducción del riesgo. Emplea tres técnicas clave:

- Modelado lógico de datos (*Logical Data Modelling*),
- Modelado de flujo de datos (*Data Flow Modelling*), y
- Modelado de Entidades y Eventos (*Entity Event Modelling*).

2. **Metodología de Diseño Orientado a Objetos** (*OOD, Object-Oriented Design*). Está muy ligada a la OOP (*Programación Orientada a Objetos*) en que se persigue la reutilización del código. A diferencia de la anterior, en este paradigma los datos y los procesos se combinan en una única entidad denominada *objetos* (o clases). Esta orientación pretende que los sistemas sean más modulares para mejorar la eficiencia, calidad del análisis y el diseño. Emplea extensivamente el *Lenguaje Unificado de Modelado* (UML) para especificar, visualizar, construir y documentar los artefactos de los sistemas software y también el modelo de negocio. UML proporciona una serie de diagramas básicos para modelar un sistema:

- Diagrama de Clase (*Class Diagram*). Muestra los objetos del sistema y sus relaciones.
- Diagrama de Caso de Uso (*Use Case Diagram*). Plasma la funcionalidad del sistema y quién interacciona con él.
- Diagrama de secuencia (*Sequence Diagram*). Muestra los eventos que se producen en el sistema y como este reacciona ante ellos.
- Modelo de Datos (*Data Model*).

3. **Desarrollo Rápido de Aplicaciones** (*RAD, Rapid Application Development*). Su filosofía es sacrificar calidad a cambio de poner en producción el sistema rápidamente con la funcionalidad esencial. Los procesos de especificación, diseño e implementación son simultáneos. No se realiza una especificación detallada y se reduce la documentación de diseño. El sistema se diseña en una serie de pasos, los usuarios evalúan cada etapa en la que proponen cambios y nuevas mejoras. Las interfaces de usuario se desarrollan habitualmente mediante sistemas interactivos de desarrollo. En vez de seguir un modelo de desarrollo en cascada sigue un modelo en espiral (Boehm). La clave de este modelo es el desarrollo continuo que ayuda a minimizar

los riesgos. Los desarrolladores deben definir las características de mayor prioridad. Este tipo de desarrollo se basa en la creación de prototipos y realimentación obtenida de los clientes para definir e implementar más características hasta alcanzar un sistema aceptable para despliegue.

4. **Metodologías Ágiles.** «[...] envuelven un enfoque para la toma de decisiones en los proyectos, que se refiere a métodos de ingeniería de software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así, el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo. Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. Teniendo gran importancia el concepto de “Finalizado” (Done), ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de “software que funciona” (sin errores). Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. [...]»<sup>1</sup>

### 3.1.2. Proceso de testing

1. *Pruebas modulares* (pruebas unitarias). Su propósito es hacer pruebas sobre un módulo tan pronto como sea posible. Las *pruebas unitarias* comprueban el correcto funcionamiento de una unidad de código. Dicha unidad elemental de código consistiría en cada función o procedimiento, en el caso de la programación estructurada y cada clase, para la programación orientada a objetos. Las características de una prueba unitaria de calidad son: *automatizable* (sin intervención manual), *completa, reutilizable, independiente y profesional*.
2. *Pruebas de integración*. Pruebas de varios módulos en conjunto para comprobar su interoperabilidad.
3. *Pruebas de caja negra*.
4. *Beta testing*.
5. *Pruebas de sistema y aceptación*.
6. *Training*.

## 3.2. OTROS ASPECTOS DEL PLAN DE GESTIÓN

Además de la metodología se deben abordar los aspectos siguientes relacionados con el plan de gestión del proyecto:

- **Recursos.** En este apartado se describirán los recursos hardware y software empleados. También debería quedar aclarado el número y papel de los integrantes del equipo de proyecto.
- **Gestión de la configuración y aseguramiento de la calidad.** Durante el desarrollo de proyectos de software es esencial definir la estrategia de control de versiones y las diferentes *releases*. En esta sección además se describirán las actividades y tareas que garantizan la calidad durante el proceso de desarrollo del software, incorporando los estándares, prácticas y normas de aplicación. También se deben documentar los distintos tipos de revisiones, verificaciones y validaciones que se realizarán, los criterios para la aprobación o rechazo de cada producto y los procedimientos para llevar a cabo acciones correctivas o preventivas.
- **Planificación.** Detallará la estimación de la evolución temporal del proyecto, marcando sus iteraciones e hitos básicos. Para ello, se emplearán diagramas Gantt y debería incluir una comparación cuantitativa del tiempo y el esfuerzo realmente invertido frente al estimado.
- **Costes y análisis de riesgos.** Análisis y presupuesto del coste de los recursos (humanos y materiales) necesarios para el proyecto. El cálculo de costes de personal debe tener en cuenta la realidad del mercado laboral en España. Dicho cálculo se puede hacer en persona/mes, y luego hacer la correspondencia al coste monetario. En esta sección se debería incluir una enumeración de los riesgos del proyecto, indicando su posible impacto (efecto que la ocurrencia

---

<sup>1</sup>Fuente: Wikipedia

del citado riesgo tendría en el desarrollo del proyecto) y la probabilidad de ocurrencia. Una vez se identifican los riesgos, se deben priorizar para definir los planes necesarios que reduzcan su impacto o incluso su probabilidad de ocurrencia.

### 3.3. REVISIÓN DE TECNOLOGÍAS Y HERRAMIENTAS CASE (*COMPUTER AIDED SOFTWARE ENGINEERING*)

Además de los recursos humanos y de hardware necesarios en el trabajo, en la sección de recursos software se deberían enumerar las herramientas software previstas. A continuación se realiza una revisión rápida no exhaustiva de las mismas.

Las herramientas CASE están destinadas a facilitar una o varias de las tareas implicadas en el ciclo de vida del desarrollo de software. Se pueden dividir en las siguientes categorías:

1. Modelado y análisis de negocio.
2. Desarrollo.
3. Verificación y validación.
4. Gestión de configuraciones.
5. Métricas y medidas.
6. Gestión de proyecto (gestión de planes, asignación de tareas, planificación, etc.).

#### 3.3.1. IDE (Integrated Development Environment)

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>■ <a href="#">Notepad++</a></li> <li>■ <a href="#">Visual Studio Code</a></li> <li>■ <a href="#">Atom</a></li> <li>■ <a href="#">GNU Emacs</a></li> <li>■ <a href="#">NetBeans</a></li> </ul> | <ul style="list-style-type: none"> <li>■ <a href="#">Eclipse</a></li> <li>■ <a href="#">Qt Creator</a></li> <li>■ <a href="#">jEdit</a></li> <li>■ <a href="#">IntelliJ IDEA</a></li> </ul> |
|--|---|

#### 3.3.2. Depuración

- [GNU Debugger](#)

#### 3.3.3. Testing

- |   |   |
|---|---|
| <ul style="list-style-type: none"> <li>■ <a href="#">JUnit</a>. Entorno de pruebas para Java.</li> <li>■ <a href="#">CUnit</a>. Entorno de pruebas para C.</li> </ul> | <ul style="list-style-type: none"> <li>■ <a href="#">PyUnit</a>. Entorno de pruebas para Python.</li> </ul> |
|---|---|

#### 3.3.4. Repositorios y control de versiones

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>■ <a href="#">Git</a></li> <li>■ <a href="#">Mercurial</a></li> <li>■ <a href="#">Github</a></li> </ul> | <ul style="list-style-type: none"> <li>■ <a href="#">Bitbucket</a></li> <li>■ <a href="#">SourceTree</a></li> </ul> |
|--|---|

**3.3.5. Documentación**

- [L<sup>A</sup>T<sub>E</sub>X](#)
- [Markdown](#)
- [Doxygen](#)
- [DocGen](#)
- [Pandoc](#)

**3.3.6. Gestión y planificación de proyectos**

- [Trello](#)
- [Jira](#)
- [Asana](#)
- [Slack](#)
- [Basecamp](#)
- [Teamwork Projects](#)
- [Zoho Projects](#)



---

## CAPÍTULO 4

# Desarrollo

---

En esta sección se describirá las diferentes fases del ciclo de desarrollo de software de acuerdo al plan de gestión expuesto en el capítulo 3.

### 4.1. REQUISITOS Y ANÁLISIS DEL SISTEMA

En este apartado se presentarán los objetivos y el catálogo de requisitos del proyecto: funcionales, no funcionales, de información, reglas de negocio, etc.

Una vez catalogados los requisitos del sistema se procederá con su análisis empleando el lenguaje de modelado UML. El análisis mencionado incluirá:

- **Modelo conceptual.** En él se identifican clases, atributos, relacionales, etc.
- **Modelo de casos de uso.** Representan las interacciones entre los actores y el sistema bajo estudio.
- **Modelo de interfaz de usuario.** Puede consistir en un prototipo de baja fidelidad de la interfaz de usuario.

#### 4.1.1. Análisis de Formatos de Telemetría en Simuladores de Carreras

Dentro del análisis del sistema, es crucial comprender los formatos de ficheros de telemetría utilizados por las plataformas de simuladores de carreras. Estos formatos presentan particularidades que afectan la extracción y gestión de datos. A continuación, se detallan algunos aspectos relevantes:

- **Formatos Propietarios y Herramientas de Extracción.** La mayoría de los simuladores de carreras utilizan formatos binarios propietarios para sus ficheros de telemetría, cuya especificación no es de dominio público. Esto implica dificultades para acceder y comprender estos datos, ya que se requiere de herramientas específicas para su extracción y análisis.
- **Elección de iRacing y su Formato .ibt.** iRacing destaca como una plataforma de sim-racing ampliamente utilizada, lo que ha motivado el desarrollo de herramientas libres para la gestión de sus ficheros de telemetría en formato .ibt. Este formato binario contiene información detallada sobre parámetros como posición en pista, velocidad, aceleración, frenado, entradas del piloto, entre otros.
- **Herramientas de Extracción y Análisis.** Dentro de la comunidad de sim-racing, existen diversas herramientas de código abierto que permiten la extracción y análisis de los ficheros .ibt de iRacing. Estas herramientas facilitan la visualización de datos de rendimiento, la comparación de sesiones y la mejora del desempeño en pista.

### 4.2. DISEÑO DEL SISTEMA

En esta sección se define la arquitectura lógica general del sistema. Para describirla se puede emplear un modelo como C4 [3], el cual permite representar la arquitectura de un sistema software mediante varios diagramas a distintos niveles de abstracción.

### 4.3. IMPLEMENTACIÓN DEL SISTEMA

En este apartado se describirá la organización del código fuente y *scripts*, describiendo el propósito de los distintos ficheros y su distribución en paquetes y directorios. Puede ser conveniente incluir alguna porción significativa de código fuente que sea de especial relevancia por su funcionalidad.

En el desarrollo de proyectos software cobra especial importancia el empleo de sistemas de control de versiones junto a repositorios en línea. Estas herramientas se convierten en esenciales para disponer de un registro histórico del desarrollo que también puede ayudar a evaluar el trabajo realizado. Por este motivo, en la memoria del proyecto se debe indicar la dirección de los repositorios empleados (p. ej., Github).

### 4.4. PRUEBAS DEL SISTEMA

En esta sección se describirá el plan de pruebas del sistema incluyendo todos los tipos llevados a cabo. El desarrollo de la sección debería tratar los aspectos siguientes:

- **Estrategia.** Donde se indica el alcance de las pruebas y los procedimientos.
- **Pruebas unitarias.** Destinadas a localizar errores en cada nuevo módulo software desarrollado antes de su integración con el resto del sistema.
- **Pruebas de integración.** Su objetivo es localizar errores en subsistemas completos analizando la interacción entre varios módulos de software.
- **Pruebas de sistema.** Contempla las pruebas funcionales con las que se realiza el análisis del buen funcionamiento de la implementación de los casos de uso del sistema. Además, en estas pruebas se comprueba el funcionamiento respecto a los requisitos no funcionales: eficiencia, seguridad, etc.
- **Pruebas de aceptación.** Su intención es demostrar, con la participación del cliente, que el producto está listo para su puesta en funcionamiento en el entorno producción.

### 4.5. DESPLIEGUE

Esta sección recoge la arquitectura física propuesta del sistema, las instrucciones para su despliegue, operación y mantenimiento del nivel de servicio.

Es muy importante que todas las justificaciones aportadas se sustenten no solo en juicios de valor sino en evidencias tangibles como: historiales de actividad, repositorios de código y documentación, porciones de código, trazas de ejecución, capturas de pantalla, demos, etc.

---

## CAPÍTULO 5

# Conclusiones

---

### 5.1. OBJETIVOS ALCANZADOS

En este capítulo se realizará un juicio crítico y discusión sobre el objetivo general y objetivos secundarios alcanzados.

### 5.2. JUSTIFICACIÓN DE COMPETENCIAS ADQUIRIDAS

Es muy importante recordar que según la normativa vigente en la ESI-UCLM, el capítulo de conclusiones debe incluir *obligatoriamente* un apartado destinado a justificar la aplicación en el TFG de las competencias específicas (una o más) adquiridas en la tecnología específica cursada, como se indica a continuación:

En el TFG se han aplicado las competencias correspondientes a la Tecnología Específica de [*poner lo que corresponda*]:

**Código de la competencia 1:** [*Texto de la competencia 1*]. Explicación de cómo se ha aplicado en el TFG.

... (otras más si las hubiera).

### 5.3. TRABAJOS DERIVADOS Y FUTUROS

Si es pertinente se puede incluir información sobre trabajos derivados como publicaciones o ponencias en preparación, así como trabajos futuros (*solo si estos están iniciados o planificados en el momento que se redacta el texto*).

Se recomienda evitar la inclusión de una lista de posibles mejoras. Contrariamente a lo que se pueda pensar, pueden transmitir la impresión de que el trabajo se encuentra en un estado incompleto o inacabado.<sup>1</sup>

### 5.4. VALORACIÓN PERSONAL

En esta sección final se realizará un rápido análisis de las lecciones aprendidas en las que se pueden incluir tanto buenas prácticas adquiridas (tecnológicas y procedimentales) como cualquier otro aspecto de interés. También se puede resumir cuantitativamente el tiempo y esfuerzo dedicados al proyecto a lo largo de su desarrollo.

---

<sup>1</sup>En cualquier caso se debe reflexionar sobre este aspecto por si los miembros del comité evaluador preguntan sobre estas posibles mejoras durante la defensa del trabajo.



# Bibliografía

---

- [1] Brook Appelbaum. Top 6 software development methodologies. URL: <https://blog.planview.com/top-6-software-development-methodologies>, September 2022. Último acceso 26 feb. 2024.
- [2] Alexánder Borbón and Walter Mora. *Edición de textos científicos con LATEX. Composición, diseño editorial, gráficos, Inkscape, Tikz y presentaciones Beamer*. Instituto Tecnológico de Costa Rica, 2 edition, 2021.
- [3] Simon Brown. The C4 model for visualising software architecture. URL: <https://c4model.com/>, 2022. Último acceso: 19 feb. 2024.
- [4] Escuela Superior de Informática, Universidad de Castilla-La Mancha. Guía de estilo y formato para Trabajos Fin de Grado. URL: <https://pruebasaluclm.sharepoint.com/sites/esicr/tfg/SiteAssets/SitePages/Inicio/20190304-GuiaEstiloFormatoTFG.pdf>, March 2019. Último acceso: sep. 2021.
- [5] Leslie Lamport. *LATEX: A document preparation system*. Addison-Wesley, second edition, June 1994. ISBN: 978-0201529838.
- [6] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *La introducción no-tan-corta a LATEX2e*, 2014. URL: <http://www.ctan.org/tex-archive/info/lshort/spanish/>.
- [7] Jesús Salido. Curso: LATEX esencial para preparación de tfg, tesis y otros documentos académicos. URL: [http://visilab.etsii.uclm.es/?page\\_id=1468](http://visilab.etsii.uclm.es/?page_id=1468), 2010. Último acceso: sep. 2021.
- [8] Jesús Salido. Plantilla LaTeX para Trabajo Fin de Estudios en Ingeniería Informática - UCLM (España). Zenodo, DOI: <https://doi.org/10.5281/zenodo.4561708>, 2019.
- [9] Servicio de publicaciones UCLM. Propiedad intelectual. Documentos elaborados por el Grupo de Gestión del Conocimiento y Propiedad Intelectual de la Universidad de Castilla-La Mancha. URL: <https://e.uclm.es/servicios/doc/?id=UCLMDOCID-12-739>. Último acceso: feb. 2024.
- [10] Universidad Carlos III de Madrid. Guía temática del TFG. URL: <https://uc3m.libguides.com/TFG>, 2021. Último acceso: sep-2021.
- [11] Universidad de Cantabria. Cómo usar imágenes en trabajos. Artículo técnico disponible en URL: [https://web.unican.es/buc/Documents/Formacion/guia\\_imagenes.pdf](https://web.unican.es/buc/Documents/Formacion/guia_imagenes.pdf), 2018. Último acceso: sep. 2021.
- [12] Universitat Oberta de Catalunya. Comunicación eficaz y redacción. URL: <https://www.uoc.edu/portal/es/servei-linguistic/redaccio/10-recomanacions/index.html>. Último acceso: sep. 2021.
- [13] Leo R. Vijayasarathy and Charles W. Butler. Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software*, 33(5):86–94, September 2016. DOI: <https://doi.org/10.1109/ms.2015.26>.
- [14] WikiMedia. LATEX Wikibook. URL: <http://en.wikibooks.org/wiki/LaTeX>, 2010. Último acceso: sep. 2021.
- [15] Leonor Zozaya. Redacción de textos. recomendaciones para presentar trabajos académicos. Blog disponible en URL: <http://redaccion.hypotheses.org/>, 2017. ISSN: 2444-8885.



## **ANEXOS**



---

## ANEXO A

# Sobre la Bibliografía

---

En los anexos se incluirá, de modo opcional, material suplementario que podrá consistir en manuales de usuario, listados seleccionados de código fuente, esquemas, planos y en general aquel contenido que complementa a la memoria. Se recomienda que no sean excesivamente voluminosos, aunque su extensión no está sometida a la regulación por normativa, ya que esta afecta únicamente al texto principal de la memoria.

En esta plantilla hemos decidido incluir dos anexos. En el primero de ellos se hacen algunos comentarios adicionales sobre la bibliografía. En el segundo se aporta una breve introducción a L<sup>A</sup>T<sub>E</sub>X cuya información puede servir de ejemplo de inclusión de ciertos elementos en la preparación de la memoria del TFG.

Todo el material de terceros se debe citar convenientemente sin contravenir los términos de las licencias de uso y distribución de dicho material. Esto se extiende al uso de diagramas y fotografías. El incumplimiento de la legislación vigente en materia de protección de la propiedad intelectual es responsabilidad exclusiva del autor, independientemente de la cesión de derechos que este haya convenido.

La sección de *Bibliografía*, que si se prefiere se puede titular *Referencias*, incluirá un listado ordenado preferentemente por orden alfabético (primer apellido del autor principal), con todas las obras citadas en el texto. En la lista de referencias se especificará para cada obra: autores, título, editorial y año de publicación. Este formato se conseguirá en L<sup>A</sup>T<sub>E</sub>X mediante el uso del estilo estándar plain o cualquier otro derivado con estilo de citación numérica. En algunas titulaciones se obliga a una ordenación por orden de cita en el texto que con Bib<sub>T</sub>E<sub>X</sub> se puede obtener mediante los estilos estándar ieeetr (estilo para los IEEE *transactions*) y unsrt (estilo *unsorted*).

Es muy importante tener presente que en esta sección solo se debe incluir las referencias bibliográficas citadas expresamente en el documento. Si se desea incluir fuentes consultadas, pero no citadas, se puede confeccionar con ellas una sección denominada *Material de consulta*, aunque estas referencias se pueden incluir opcionalmente a lo largo del documento como notas a pie de página.

En las titulaciones técnicas se empleará estilo de citación numérico con el número de la referencia entre corchetes. La cita podrá incluir el número de página concreto de la referencia que se desea citar. El uso correcto de la citación implica dejar claro al lector cuál es el texto, material o idea citado. Las obras referenciadas sin mención explícita o implícita al material concreto citado se deberían considerar material de consulta y, por tanto, ser agrupadas como *Material de consulta*, distinguiéndolas claramente de aquellas otras en las que sí se recurre a la citación.

En las titulaciones que requieren un estilo de citación de tipo autor-año (no numérico), se puede incluir el paquete L<sup>A</sup>T<sub>E</sub>X apacite (con la opción natbibapa) y especificar este mismo estilo en la sección de bibliografía en el argumento del comando bibliographystyle.

Cuando se deseé incluir referencias a páginas genéricas de la Web sin mención expresa a un artículo con título y autor definido, dichas referencias se pueden incluir como notas al pie de página o como un apartado de fuentes de consulta dedicado a *Direcciones de Internet*. Por el contrario, los

documentos electrónicos publicados en Internet se pueden incluir empleando el tipo de entrada `misc` con el comando `url` como se muestra en la bibliografía que acompaña esta plantilla. Observarás que el campo `note` se emplea para añadir información adicional como la fecha de la última consulta de fuentes publicadas en Internet, y para la inclusión del DOI de algunas obras para su rápida recuperación. Sin embargo, ten cuidado porque esta estrategia puede necesitar una adaptación con un estilo de citación autor-año.

---

## ANEXO B

# Breve introducción a L<sup>A</sup>T<sub>E</sub>X

---

El contenido del trabajo final de estudios se organiza en capítulos que se subdividen en secciones. Con L<sup>A</sup>T<sub>E</sub>X este tipo de organización se realiza de modo inmediato mediante la generación automática de los estilos correspondientes a los títulos de cada sección y su inclusión en la tabla de contenidos. Los ajustes relativos a la generación del formato y estilos asociados a secciones del documento se realizan con el paquete `titlesec` empleado en esta plantilla.

En las secciones siguientes se comenta la inclusión con L<sup>A</sup>T<sub>E</sub>X de distintos elementos de organización de información junto a ejemplos que facilitan su utilización en la memoria del trabajo.<sup>1</sup>

### B.1. LISTAS

Existen dos tipos de listas: enumeraciones y listas con viñetas. En el primer tipo los elementos de la lista se preceden de una clave numérica o alfabética, mientras que en el segundo tipo se emplea una viñeta. En ambos casos los elementos se pueden anidar para crear una jerarquía entre ellos. En L<sup>A</sup>T<sub>E</sub>X se recomienda la inclusión del paquete `enumitem` que permite personalizar fácilmente las listas de un documento. A continuación se muestran algunos ejemplos:

Ejemplo de lista con viñetas personalizadas.

- pera
- manzana
- naranja

Ejemplo de lista condensada con separación mínima, en varias columnas y configuración de la etiqueta.

- |             |              |
|-------------|--------------|
| (1) pera    | (4) patata   |
| (2) manzana | (5) calabaza |
| (3) naranja | (6) fresa    |

Además del texto, los documentos pueden incluir elementos que enriquecen su contenido facilitando su exposición y comprensión. En las secciones siguientes tratamos brevemente dichos elementos.

### B.2. ECUACIONES MATEMÁTICAS

Para escribir ecuaciones matemáticas con L<sup>A</sup>T<sub>E</sub>X se recomienda incluir los paquetes siguientes en el documento: `amsmath`, `amsfonts`, `amssymb`.

La composición de ecuaciones requiere el uso de comandos especializados. Por tanto, para facilitar dicha tarea se aconseja el empleo de programas especializados como MathType o asistentes como

---

<sup>1</sup>Las explicaciones de este anexo forman parte del contenido del curso «L<sup>A</sup>T<sub>E</sub>X esencial para preparación de TFG y otros documentos académicos» de la [ESI-UCLM](#).

el incluido en editores como TeXstudio<sup>2</sup> o herramientas en línea.<sup>3</sup> Es muy sencillo incluir fórmulas matemáticas sencillas en el mismo texto en el que se escribe. Por ejemplo,  $h^2 = a^2 + b^2$  que podría ser la ecuación representativa del teorema de Pitágoras (ver también ec. B.1).

Las fórmulas también se pueden separar del texto para que aparezcan destacadas, así:

$$c^2 = \int (a^2 + b^2) \cdot dx$$

Pero si se desea, las ecuaciones pueden ser numeradas de forma automática e incluso utilizar referencias cruzadas a ellas:

$$h^2 = b^2 + c^2 \quad (\text{B.1})$$

### B.3. TABLAS

A continuación se incluyen algunos ejemplos de tablas elaboradas con L<sup>A</sup>T<sub>E</sub>X mediante el empleo de paquetes dedicados. Para la realización de tablas más complejas se recomienda la consulta de [2] y el empleo de asistentes o herramientas en línea.<sup>4</sup>

Se debe observar que el título de las tablas se ubica en la parte superior de la tabla. Puesto que el contenido de la tabla es texto, tiene sentido leer primero el título para contextualizar el contenido de la tabla antes de su lectura.

**Tabla B.1:** Ejemplo de uso de la macro `cline`

7C0	hexadecimal
3700	octal
11111000000	binario
1984	decimal

Ejemplo de tabla en la que se controla el ancho de la celda.

**Tabla B.2:** Ejemplo de tabla con especificación de anchura de columna

Día	Temp Mín (°C)	Temp Máx (°C)	Previsión
Lunes	11	22	Día claro y muy soleado. Sin embargo, la brisa de la tarde puede hacer que las temperaturas desciendan
Martes	9	19	Nuboso con chubascos en muchas regiones. En Cataluña claro con posibilidad de bancos nubosos al norte de la región
Miércoles	10	21	La lluvia continuará por la mañana, pero las condiciones climáticas mejorarán considerablemente por la tarde

<sup>2</sup><https://www.texstudio.org/>

<sup>3</sup><https://latex.codecogs.com/>, <http://www.sciweavers.org/free-online-latex-equation-editor>

<sup>4</sup><https://www.tablesgenerator.com/>

#### B.4. FIGURAS

A diferencia de lo que sucede en las tablas, el título de las figuras aparece en la parte inferior de estas. Para la inclusión de las figuras se debe tener en cuenta que su contenido se encuentra en un fichero individual con el formato y resolución apropiados para garantizar la calidad del resultado final.

En esta sección se añaden ejemplos de muestra para la inclusión de figuras simples y otras compuestas de subfiguras mediante el empleo del paquete `subcaption`.



**Figura B.1:** Fotografía a color (Fuente: J. Salido, CC BY-NC-ND)

Ejemplo de figura compuesta por dos subfiguras incluidas mediante paquete `subcaption`. A través del uso de etiquetas (`\label`) es posible incluir referencias cruzadas a subfiguras como la fotografía en blanco y negro de la Fig. B.2b.



(a) Fotografía a color



(b) Fotografía en blanco y negro

**Figura B.2:** Ejemplo de inclusión de subfiguras en un mismo entorno (Fuente: J. Salido, CC BY-NC-ND)

En los trabajos académicos la inclusión de imágenes y figuras que no son propiedad del autor suscitan bastante controversia, ya que con frecuencia se incumple inadvertidamente la ley vigente de propiedad intelectual. Respecto a este hecho se recomienda, tanto a estudiantes como tutores, consultar documentación informativa sobre el uso correcto de figuras en documentos académicos [9, 11]. Entre las «incorrectas» más habituales en los documentos académicos, se observa:

- *Abuso del derecho de cita.* Se produce al incluir, con fines exclusivamente decorativos o ilustrativos de la explicación, una figura sujeta a derechos de uso restringido invocando el derecho de cita (incluso con correcta atribución de la obra).
- *Incorrecta atribución de la obra.* Es habitual confundir al autor de la obra con la fuente de origen de la misma. La fuente es precisa cuando se cita la obra original. Sin embargo, la licencia de muchas obras exige la atribución al autor y la inclusión de la licencia bajo la que se distribuye o hace uso de la misma (véase como ejemplo cómo se realiza una correcta atribución en las Fig. B.1 y B.2 mencionando al autor y la licencia Creative-Commons<sup>5</sup> bajo la que se rige el uso de la imagen y el mecanismo de título alternativo para que dicha atribución no aparezca en el índice de figuras usando título opcional).
- *Supresión de los detalles de la licencia de uso.* Al incluir obras de terceros debemos tener presente los términos de distribución de la misma e incluirlos junto a la atribución de su legítimo autor.

La inclusión de material de *dominio público*, sin restricciones de uso o con permiso, hace innecesaria la atribución al autor, pero se recomienda incluir una nota de agradecimiento.<sup>6</sup>

Cuando se presenta la necesidad de incluir un gráfico demasiado grande para el tamaño de la página, una opción muy apropiada es la impresión del gráfico en modo girado en una página aparte. Este efecto se consigue con el entorno `sidewaysfigure` proporcionado por el paquete `rotating`. La Fig. B.3 muestra un ejemplo del entorno citado con un gráfico PDF.

---

<sup>5</sup><https://creativecommons.org>

<sup>6</sup>Incluyendo un texto como: «Por cortesía de ...»

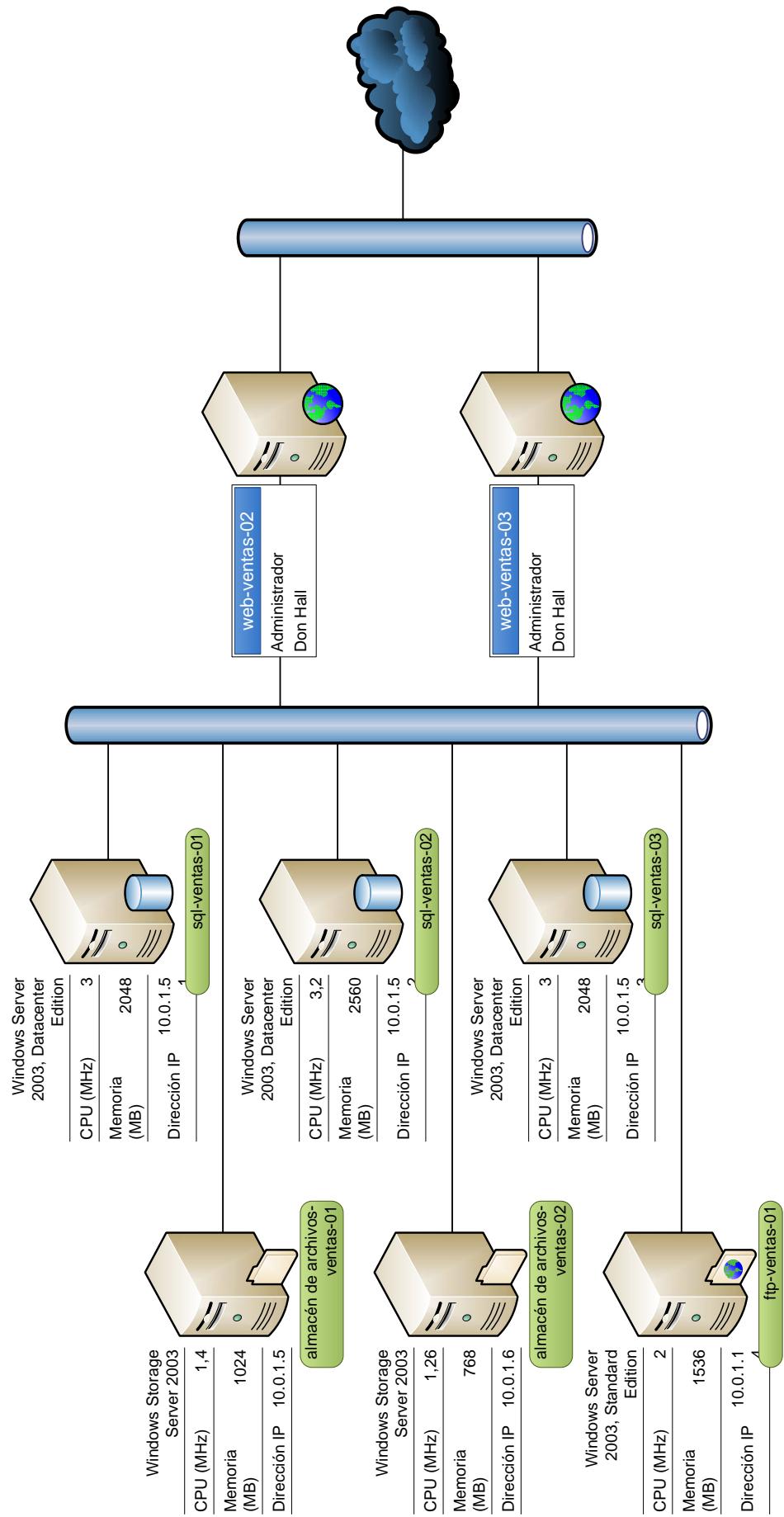
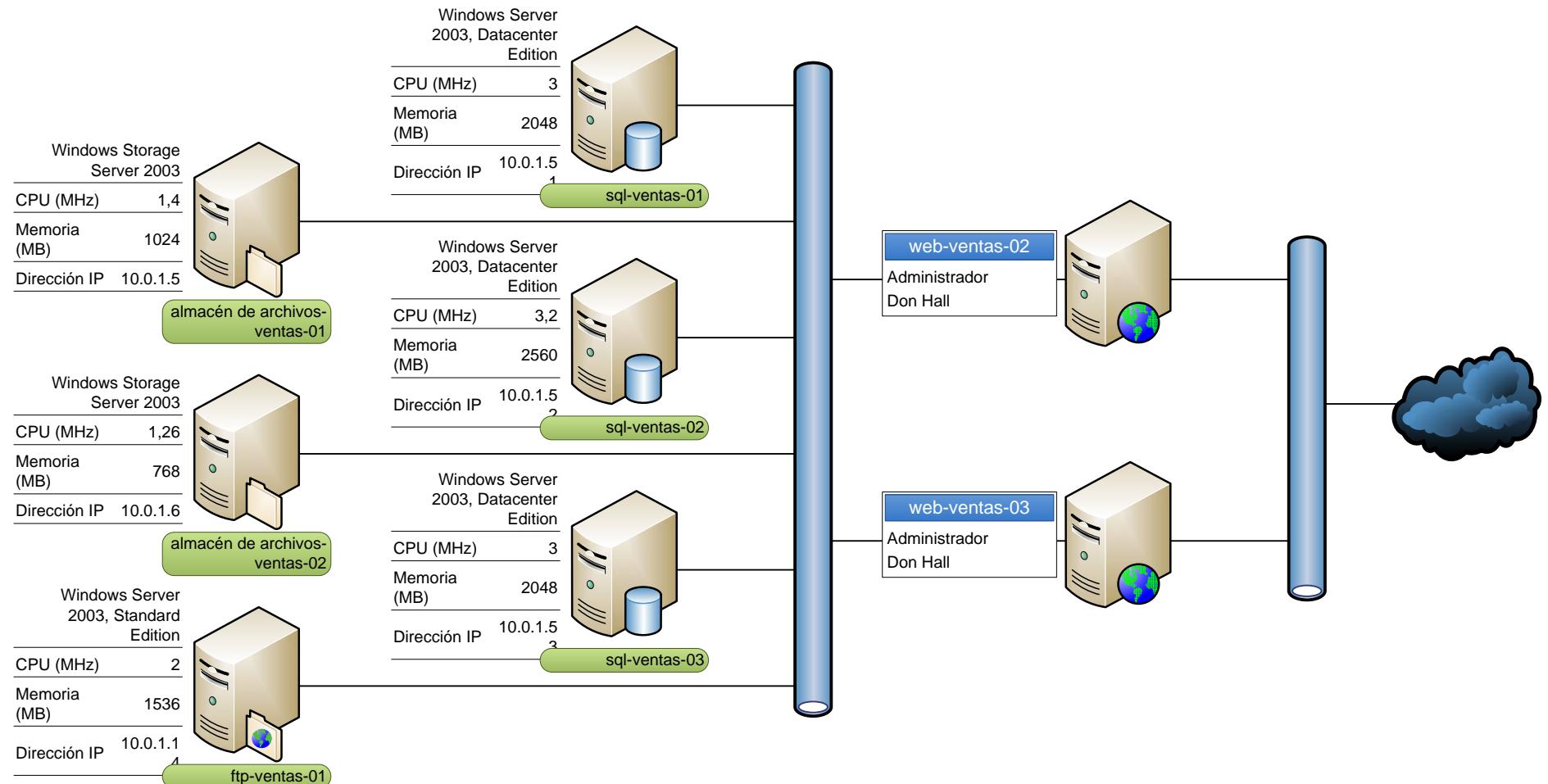


Figura B.3: Figura vectorial con impresión girada

También es posible imprimir una página en formato apaisado cuando contiene una figura muy ancha. Este efecto se consigue con el paquete `pdf1scape` y el entorno `landscape` proporcionado. Además, en este caso se han suprimido tanto la cabecera como el pie de página. La figura B.4 se muestra apaisada a modo de ejemplo.



**Figura B.4:** Figura vectorial con vista en página apaisada

### B.5. ALGORITMOS Y LISTADOS DE CÓDIGO FUENTE

En los textos científicos relacionados con las TIC<sup>7</sup> (Tecnologías de la Información y Comunicaciones) suelen aparecer porciones de código en los que se explica alguna función o característica relevante del trabajo que se expone. Muchas veces lo que se quiere ilustrar es un algoritmo o método con el que se resuelve un problema abstrayéndose del lenguaje de implementación. El paquete `algorithm2e` proporciona un entorno `algorithm` para la impresión apropiada de algoritmos, tratándolos como objetos flotantes y con mucha flexibilidad de personalización, como se observa en el algoritmo B.1 del ejemplo.

#### Algoritmo B.1: Cómo escribir algoritmos

```
Datos :este texto
Resultado:como escribir algoritmos con LATEX2e
1 inicialización;
2 while no es el fin del documento do
3   leer actual;
4   if comprendido then
5     ir a la siguiente sección;
6     la sección actual es esta;
7   else
8     ir al principio de la sección actual;
9   end
10 end
```

<sup>7</sup>Por supuesto, en un TFG (Trabajo Fin de Grado) o tesis de un centro superior de Informática.

La inclusión de porciones de código fuente se puede formatear de modo sencillo en L<sup>A</sup>T<sub>E</sub>X mediante el uso del paquete *listings*. A continuación, se muestran varios ejemplos de porciones de código correspondientes a distintos lenguajes de programación.

**Listado B.1:** Ejemplo de código fuente en lenguaje Java

```

1 // @author www.javadb.com
2 public class Main {
3 // Este método convierte un String a un vector de bytes
4
5 public void convertStringToByteArray() {
6
7 String stringToConvert = "This\u2014String\u2014is\u201415";
8 byte[] theByteArray = stringToConvert.getBytes();
9 System.out.println(theByteArray.length);
10 }
11
12 public static void main(String[] args) {
13 new Main().convertStringToByteArray();
14 }
15 }
```

**Listado B.2:** Ejemplo de código fuente en lenguaje C

```

1 // Este código se ha incluido tal cual está en el fichero LATEX
2 #include <stdio.h>
3
4 int main(int argc, char* argv[]) {
5 puts("¡Hola\u2014mundo!");
6 }
```

**Listado B.3:** Ejemplo de script en Matlab

```

1 function f = fibonacci(n)
2 % FIBONACCI Fibonacci sequence
3 % f = FIBONACCI(n) generates the first n Fibonacci numbers.
4 % Copyright 2014 Cleve Moler
5 % Copyright 2014 The MathWorks, Inc.
6
7 f = zeros(n,1);
8 f(1) = 1;
9 f(2) = 2;
10 for k = 3:n
11 f(k) = f(k-1) + f(k-2);
12 end
```

## B.6. MENÚS, PATHS Y TECLAS CON EL PAQUETE MENUKEYS

Cada vez es más usual que los trabajos en ingeniería exijan el uso de software. Para poder especificar de modo elegante el uso de menús, pulsaciones de teclas y directorios, se recomienda el uso del paquete *menukeys*.<sup>8</sup> Este paquete nos permite especificar el acceso a un menú, por ejemplo:

Herramientas ▶ Órdenes ▶ PDFLaTeX

También un conjunto de teclas. Por ejemplo: **Ctrl** + **↑** + **T**

O un directorio: **█ C: ▶ user ▶ LaTeX ▶ Ejemplos**

Aunque este paquete permite muchas opciones de configuración de los estilos aplicados, esto no es necesario para obtener unos resultados muy elegantes.

<sup>8</sup><https://osl.ugr.es/CTAN/macros/latex/contrib/menukeys/menukeys.pdf>