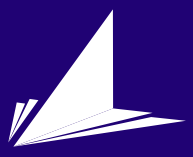

Escuela
Politécnica
Superior

Título del Trabajo Fin de Grado/Máster

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
Nombre Apellido1 Apellido2
Tutor:
Dr./Dra. Nombre Apellido1 Apellido2

Febrero 2026

Universitat d'Alacant
Universidad de Alicante

Título del Trabajo Fin de Grado/Máster

Subtítulo del proyecto

Autor
Nombre Apellido1 Apellido2

Tutor
Dr./Dra. Nombre Apellido1 Apellido2
Departamento de ejemplo

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Febrero 2026

A mi familia, por su apoyo incondicional.

Agradecimientos

En primer lugar, quiero agradecer a mi tutor/a por su dedicación y orientación durante el desarrollo de este trabajo.

También quiero expresar mi gratitud a mi familia y amigos por su apoyo constante.

Finalmente, agradecer a la Escuela Politécnica Superior de la Universidad de Alicante por la formación recibida durante estos años.

Resumen

Este documento es la plantilla oficial para la elaboración de Trabajos de Fin de Grado (TFG) y Trabajos de Fin de Máster (TFM) de la Escuela Politécnica Superior de la Universidad de Alicante.

La plantilla proporciona un formato profesional y consistente, cumpliendo con las directrices de estilo de la EPS. Incluye:

- Portadas oficiales a color y en blanco y negro
- Configuración automática según la titulación
- Estilos predefinidos para código fuente
- Formato de bibliografía según normas APA
- Soporte para acrónimos y glosarios
- Optimización de figuras TikZ

Palabras clave: TFG, TFM, plantilla, LaTeX, Universidad de Alicante

Abstract

This document is the official template for the preparation of Bachelor's Thesis (TFG) and Master's Thesis (TFM) at the Polytechnic School of the University of Alicante.

The template provides a professional and consistent format, complying with the EPS style guidelines. It includes:

- Official color and black-and-white covers
- Automatic configuration based on the degree program
- Predefined styles for source code
- Bibliography format according to APA standards
- Support for acronyms and glossaries
- TikZ figure optimization

Keywords: Bachelor's Thesis, Master's Thesis, template, LaTeX, University of Alicante

Índice general

Índice de figuras

Índice de tablas

1. Introducción (Ejemplos de contenido y estilos)

Este capítulo presenta una guía completa de las capacidades de LaTeX y de esta plantilla para la elaboración de Trabajos Fin de Grado y Trabajos Fin de Máster en la Escuela Politécnica Superior de la Universidad de Alicante.

1.1. Sobre esta plantilla

Esta plantilla ha sido diseñada siguiendo las directrices de estilo de la Escuela Politécnica Superior (EPS) de la Universidad de Alicante. Proporciona una estructura clara y profesional para la redacción de trabajos académicos.

Las principales características de esta plantilla son:

- Configuración sencilla mediante clave-valor con `\EPSsetup{...}`
- 21 portadas diferentes según la titulación
- Soporte para múltiples idiomas (español, inglés, valenciano)
- Resaltado de código con colores (minted + Pygments)
- Bibliografía con estilo American Psychological Association (APA)
- Compilación con LuaLaTeX

1.2. Estructura de un TFG/TFM

Según las normas de la EPS, un Trabajo Fin de Grado (TFG) o Trabajo Fin de Máster (TFM) debe contener las siguientes partes:

Preámbulo: Motivación y descripción breve de los objetivos del trabajo.

Agradecimientos: Reconocimientos a entidades y personas colaboradoras.

Dedicatoria: Opcional, con alineación a la derecha.

Índices: Contenidos, figuras, tablas y códigos.

Introducción: Importancia de la temática, vigencia y planteamiento del problema.

Marco teórico: Fundamentos conceptuales y estado del arte.

Objetivos: Objetivo general y específicos.

Metodología: Tipo de investigación, técnicas y procedimientos.

Resultados: Resultados obtenidos y análisis.

Conclusiones: Resumen de objetivos conseguidos.

Bibliografía: Referencias utilizadas (estilo APA recomendado).

Anexos: Material complementario.

1.3. Secciones y subsecciones

En LaTeX existen diferentes niveles de organización del contenido:

Niveles de secciones

```latex
\chapter{Capítulo}           % Nivel 0
\section{Sección}            % Nivel 1
\subsection{Subsección}      % Nivel 2
\subsubsection{Subsubsección}% Nivel 3
\paragraph{Párrafo}          % Nivel 4
```

1.3.1. Ejemplo de subsección

Este es el contenido de una subsección. Las subsecciones permiten organizar mejor el contenido dentro de cada sección.

1.3.1.1. Ejemplo de subsubsección

Las subsubsecciones son útiles para temas muy específicos dentro de una subsección.

1.3.1.1.1. Ejemplo de párrafo Los párrafos con título son útiles para pequeñas divisiones que no necesitan aparecer en el índice.

1.4. Citas bibliográficas

Para citar la bibliografía según el sistema APA se utilizan los siguientes comandos. El archivo de bibliografía se encuentra en `referencias.bib`.

Comandos de citas biblatex/APA

```latex
% Cita textual: Autor (año)
\textcite{latex2024}

% Cita entre paréntesis: (Autor, año)
\parencite{latex2024}

% Cita con página específica
\parencite[Cap.~2]{latex2024}

% Múltiples citas
```

```
\parencite{latex2024,overleaf2024}
```

Ejemplos de citas:

- Cita textual: Según LaTeX Project (2024), LaTeX es el estándar para documentos científicos.
- Cita entre paréntesis: LaTeX es ampliamente utilizado (LaTeX Project, 2024).
- Múltiples fuentes: Existen varios recursos disponibles (LaTeX Project, 2024; Overleaf, 2024).

1.5. Notas al pie de página

Las notas al pie se crean con el comando `\footnote{texto}`.
La plantilla utiliza LuaLaTeX[1] como motor de compilación por sus capacidades avanzadas de manejo de fuentes[2].

1.6. Estilos de texto

LaTeX ofrece múltiples estilos de texto:

- *Texto en cursiva* – `\textit{texto}`
- **Texto en negrita** – `\textbf{texto}`
- `Texto monoespacio` – `\texttt{texto}`
- Texto en versalitas – `\textsc{texto}`
- Texto subrayado – `\underline{texto}`
- ***Negrita y cursiva*** – combinación de comandos
- Texto pequeño – `{\small texto}`
- Texto grande – `{\large texto}`

1.7. Acrónimos y glosario

Los acrónimos se gestionan automáticamente con el paquete `glossaries`. La primera vez que aparece un acrónimo se muestra su forma completa, y en las siguientes apariciones solo la abreviatura.

[1] LuaLaTeX es un motor de composición tipográfica que combina LaTeX con el lenguaje de programación Lua, permitiendo mayor flexibilidad en el procesamiento de documentos.
[2] Permite usar cualquier fuente OpenType o TrueType instalada en el sistema sin necesidad de configuración adicional.

Uso de acrónimos

```
% Primera aparición: muestra "Institute of Electrical
% and Electronics Engineers (IEEE)"
El \gls{ieee} es una institución importante.

% Siguientes apariciones: muestra solo "IEEE"
El \gls{ieee} establece estándares.

% Para forzar forma larga: \glslong{ieee}
% Para forzar forma corta: \glsshort{ieee}
```

Ejemplo: El Institute of Electrical and Electronics Engineers (IEEE) es una institución importante en ingeniería. El IEEE establece estándares para la industria. Además del IEEE, existen otras organizaciones como la International Organization for Standardization (ISO).

Los acrónimos se definen en el archivo `contenido/anexos/acronimos.tex`.

1.8. Tareas pendientes y notas

Durante la redacción es útil marcar partes que requieren revisión usando el paquete `todonotes`:

Comandos de notas

```
% Nota al margen
\todo{Texto de la nota}

% Nota en línea
\todo[inline]{Nota dentro del texto}

% Figura pendiente
\missingfigure{Descripción de la figura}
```

Las notas también pueden incluirse dentro del texto:
Para indicar figuras que faltan por añadir:

1.9. Comandos personalizados de la plantilla

Esta plantilla proporciona comandos para acceder a la información configurada:

- `\EPStitulo` – Título: “Título del Trabajo Fin de Grado/Máster”
- `\EPSautor` – Autor: Nombre Apellido1 Apellido2
- `\EPStutor` – Tutor: Dr./Dra. Nombre Apellido1 Apellido2
- `\EPSfecha` – Fecha: Febrero 2026
- `\EPStipoTrabajo` – Tipo: Trabajo Fin de Grado

- `\EPStitulacion` – Titulación: Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Este documento es un Trabajo Fin de Grado presentado en Febrero 2026.

1.10. Hipervínculos y URLs

Para incluir enlaces web se utiliza el comando `\url{dirección}`:

- Página de la EPS: https://eps.ua.es
- Overleaf (editor online): https://www.overleaf.com
- CTAN (repositorio de paquetes): https://ctan.org

Para enlaces con texto personalizado: Escuela Politécnica Superior.

2. Marco Teórico (Ejemplos de listas)

Este capítulo presenta los fundamentos teóricos necesarios y demuestra las diferentes formas de crear listas en LaTeX.

2.1. Listas básicas

Existen tres tipos principales de listas en LaTeX: itemize (viñetas), enumerate (numeradas) y description (definiciones).

2.1.1. Listas con viñetas (itemize)

Lista con viñetas

```latex
\begin{itemize}
  \item Primer elemento
  \item Segundo elemento
  \item Tercer elemento
\end{itemize}
```

El resultado es:

- Primer elemento de la lista
- Segundo elemento de la lista
- Tercer elemento de la lista

2.1.2. Listas numeradas (enumerate)

Lista numerada

```latex
\begin{enumerate}
  \item Primer paso del proceso
  \item Segundo paso del proceso
  \item Tercer paso del proceso
\end{enumerate}
```

El resultado es:

1. Primer paso del proceso
2. Segundo paso del proceso
3. Tercer paso del proceso

2.1.3. Listas anidadas

Las listas pueden anidarse hasta varios niveles:

- Ingeniería Informática
  - Mención en Computación
  - Mención en Ingeniería del Software
  - Mención en Sistemas de Información
- Ingeniería Multimedia
  - Mención en Creación y Ocio Digital
  - Mención en Gestión de Contenidos
- Ingeniería en Sonido e Imagen

También con números:

1. Fase de análisis
   a) Recopilación de requisitos
   b) Análisis de viabilidad
   c) Documentación inicial
2. Fase de diseño
   a) Diseño arquitectónico
   b) Diseño detallado
3. Fase de implementación

2.2. Listas de definición

Las listas de descripción son útiles para glosarios y definiciones:

Lista de definiciones

```
\begin{description}
  \item[Término 1:] Definición del primer término.
  \item[Término 2:] Definición del segundo término.
\end{description}
```

LaTeX: Sistema de composición de textos orientado a la creación de documentos científicos y técnicos de alta calidad tipográfica.

TFG: Trabajo Fin de Grado. Proyecto final que los estudiantes de grado deben realizar y defender para obtener su título universitario.

TFM: Trabajo Fin de Máster. Proyecto de investigación o aplicación que los estudiantes de máster deben completar.

EPS: Escuela Politécnica Superior de la Universidad de Alicante.

2.2.1. Listas de definición anidadas

Software de diseño: Herramientas utilizadas en el proyecto.

Ventajas:

- Permite realizar simulaciones precisas
- Interfaz gráfica intuitiva
- Exportación a múltiples formatos

Inconvenientes:

- Curva de aprendizaje pronunciada
- Requiere licencia comercial
- Alto consumo de recursos

2.3. Estado del arte

A continuación se presenta una revisión de los trabajos más relevantes en el área.

2.3.1. Trabajos previos

Según LaTeX Project (2024), el estado actual de la investigación en sistemas de composición tipográfica indica que LaTeX sigue siendo el estándar para documentos científicos.

En Overleaf, 2024 se propone un enfoque basado en edición colaborativa en la nube que ha facilitado enormemente la adopción de LaTeX en entornos académicos.

2.3.2. Tecnologías relacionadas

Las principales tecnologías utilizadas en este ámbito incluyen:

1. **LuaLaTeX:** Motor de composición moderno que permite:
   - Uso directo de fuentes del sistema
   - Programación en Lua dentro del documento
   - Soporte nativo de Unicode
2. **Overleaf:** Plataforma de edición online que ofrece:
   - Compilación en la nube
   - Colaboración en tiempo real
   - Control de versiones integrado
3. **Git:** Sistema de control de versiones útil para:
   - Seguimiento de cambios en el documento
   - Trabajo en equipo
   - Recuperación de versiones anteriores

2.4. Herramientas utilizadas

En este trabajo se han utilizado las siguientes herramientas:

Tabla 2.1: Herramientas utilizadas en el desarrollo

Herramienta	**Versión**	**Uso**
TeX Live	2024	Distribución LaTeX
LuaLaTeX	1.18	Motor de compilación
VS Code	1.85	Editor de texto
Git	2.43	Control de versiones
Pygments	2.17	Resaltado de código

La Tabla 2.1 muestra las principales herramientas empleadas en la elaboración de este documento.

3. Objetivos (Ejemplos de tablas)

Este capítulo presenta los objetivos del trabajo y demuestra las diferentes formas de crear tablas en LaTeX.

3.1. Objetivo general

El objetivo general de este trabajo es...

3.2. Objetivos específicos

1. **OE1:** Analizar el estado actual de...
2. **OE2:** Diseñar una arquitectura que permita...
3. **OE3:** Implementar los componentes necesarios para...
4. **OE4:** Evaluar el rendimiento del sistema mediante...
5. **OE5:** Documentar el proceso de desarrollo y los resultados obtenidos.

3.3. Tablas en LaTeX

Las tablas son elementos fundamentales en cualquier documento técnico. A continuación se muestran diferentes ejemplos.

3.3.1. Tabla simple

Código de tabla simple

```latex
\begin{table}[H]
  \centering
  \begin{tabular}{lcc}
            & Columna A & Columna B \\
    \hline
    Fila 1 & Dato 1A   & Dato 1B \\
    Fila 2 & Dato 2A   & Dato 2B \\
    Fila 3 & Dato 3A   & Dato 3B \\
    \hline
  \end{tabular}
  \caption{Ejemplo de tabla simple}
  \label{tab:simple}
\end{table}
```

	Columna A	Columna B
Fila 1	Dato 1A	Dato 1B
Fila 2	Dato 2A	Dato 2B
Fila 3	Dato 3A	Dato 3B

Tabla 3.1: Ejemplo de tabla simple

3.3.2. Tabla con booktabs

El paquete `booktabs` proporciona líneas más profesionales:

Tabla 3.2: Tabla con estilo booktabs

Concepto	**Valor**	**Unidad**
Velocidad máxima	120	km/h
Consumo medio	5.5	L/100km
Potencia	150	CV
Par motor	250	Nm

3.3.3. Tabla con columnas de ancho fijo

Puedes especificar el ancho de las columnas usando `p{ancho}`, `L{ancho}`, `C{ancho}` o `R{ancho}`:

Tabla 3.3: Parámetros de posicionamiento de elementos flotantes

Parámetro	**Significado**
`h`	Sitúa el elemento *preferentemente* en la posición actual del texto
`t`	Sitúa el elemento en la parte superior de la página
`b`	Sitúa el elemento en la parte inferior de la página
`p`	Sitúa el elemento en una página dedicada solo a flotantes
`H`	Fuerza la posición exacta (requiere paquete `float`)

3.3.4. Tabla con multicolumna y multifila

Tabla 3.4: Ejemplo de tabla con celdas combinadas

Distribución de recursos			
Fase	**Recursos**		
	Personal	Tiempo (h)	Coste (€)
Análisis	2	40	2.000
Diseño	3	80	4.800
Implementación	4	160	9.600
Pruebas	2	60	3.000
Total	–	**340**	**19.400**

3.3.5. Tabla con colores alternados

Usando el paquete `xcolor` con la opción `table`:

Tabla 3.5: Especificaciones técnicas del sistema

Componente	**Especificación**	**Requisito**
Procesador	Intel i7 / AMD Ryzen 7	Mínimo
Memoria RAM	16 GB DDR4	Recomendado
Almacenamiento	512 GB SSD	Mínimo
GPU	NVIDIA RTX 3060	Recomendado
Sistema Operativo	Linux / Windows 11	Compatible

3.3.6. Tabla larga (longtable)

Para tablas que ocupan varias páginas, se usa el entorno `longtable`:

Tabla 3.6: Lista de requisitos del sistema

ID	**Tipo**	**Descripción**
RF01	Funcional	El sistema debe permitir el registro de usuarios mediante correo electrónico
RF02	Funcional	Los usuarios deben poder iniciar sesión con sus credenciales
RF03	Funcional	El sistema debe generar informes en formato PDF
RF04	Funcional	Se debe implementar un sistema de notificaciones
RF05	Funcional	Los datos deben poder exportarse en formato CSV

Continúa en la siguiente página

Tabla 3.6 – Continuación

ID	Tipo	Descripción
RNF01	No funcional	El tiempo de respuesta no debe superar los 2 segundos
RNF02	No funcional	El sistema debe soportar 100 usuarios concurrentes
RNF03	No funcional	La disponibilidad debe ser del 99.5%
RNF04	No funcional	La interfaz debe ser accesible según WCAG 2.1
RNF05	No funcional	Los datos sensibles deben cifrarse con AES-256

3.4. Generadores de tablas

Crear tablas manualmente puede ser tedioso. Se recomienda usar generadores online:

- https://www.tablesgenerator.com/ – Generador visual muy completo
- https://www.latex-tables.com/ – Alternativa sencilla
- Excel2LaTeX – Plugin para Microsoft Excel

3.5. Forzar posición de tablas

Para forzar que una tabla aparezca en un lugar específico, se puede usar:

Forzar posición

```latex
% Opción 1: Usar FloatBarrier
\begin{table}[H]
  ...
\end{table}
\FloatBarrier % Fuerza que la tabla aparezca antes

% Opción 2: Usar H (requiere paquete float)
\begin{table}[H]
  ...
\end{table}
```

4. Metodología (Ejemplos de figuras)

Este capítulo describe la metodología seguida en el desarrollo del trabajo y presenta ejemplos de cómo insertar figuras en LaTeX.

4.1. Metodología de trabajo

Para el desarrollo de este trabajo se ha seguido una metodología ágil basada en iteraciones cortas:

1. **Planificación:** Definición de objetivos y alcance
2. **Análisis:** Estudio del problema y requisitos
3. **Diseño:** Arquitectura y diseño detallado
4. **Implementación:** Desarrollo del código
5. **Pruebas:** Verificación y validación
6. **Documentación:** Redacción de la memoria

4.2. Inserción de figuras

Las figuras en LaTeX son elementos flotantes. Esto significa que LaTeX decide su ubicación óptima para mejorar la maquetación del documento.

4.2.1. Figura simple

Código para insertar una figura

```latex
\begin{figure}[H]
  \centering
  \includegraphics[width=0.6\textwidth]{ruta/imagen}
  \caption{Descripción de la figura}
  \label{fig:etiqueta}
\end{figure}
```

Figura 4.1: Ejemplo de figura simple (placeholder)

Para referenciar la figura en el texto: "como se muestra en la Figura 4.1".

4.2.2. Subfiguras

Cuando necesitas mostrar varias imágenes relacionadas:

(a) Primera variante **(b)** Segunda variante

Figura 4.2: Ejemplo de subfiguras horizontales

Puedes referenciar subfiguras individuales: Figura 4.2a y Figura 4.2b, o el conjunto: Figura 4.2.

4.2.3. Múltiples imágenes en tabla

Otra forma de organizar varias imágenes:

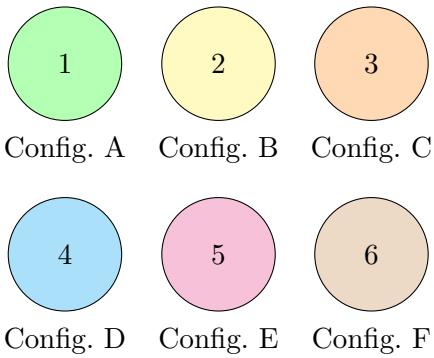

Figura 4.3: Matriz de configuraciones experimentales

4.3. Planificación temporal

La planificación temporal del proyecto se muestra en la Figura 4.4.

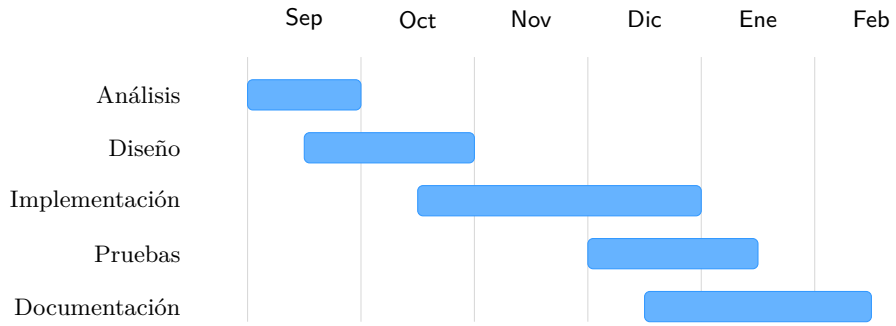

Figura 4.4: Planificación temporal del proyecto (diagrama de Gantt simplificado)

4.4. Diagramas con TikZ

TikZ permite crear diagramas directamente en LaTeX:

4.4.1. Diagrama de flujo

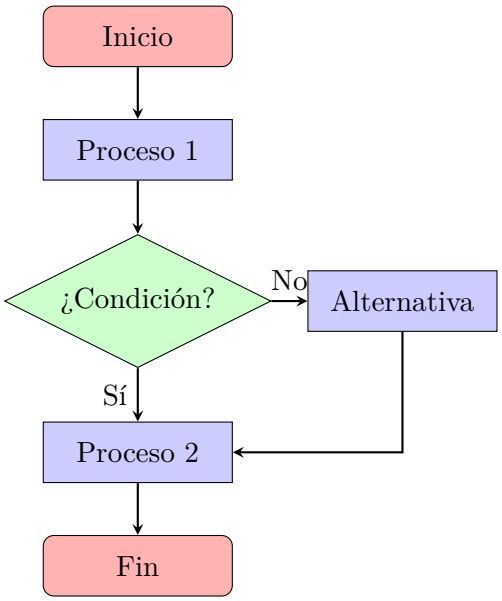

Figura 4.5: Diagrama de flujo del algoritmo principal

4.4.2. Diagrama de bloques

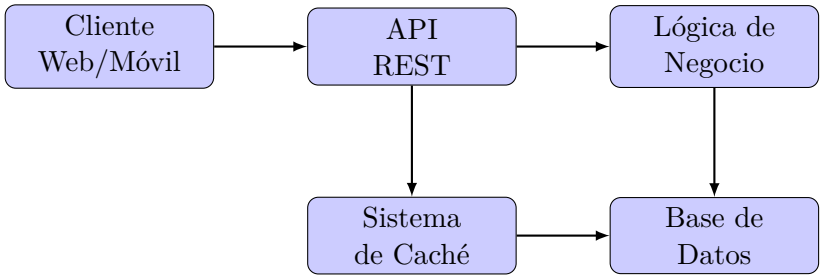

Figura 4.6: Arquitectura del sistema

4.5. Recursos utilizados

4.5.1. Recursos hardware

- Ordenador portátil con procesador Intel i7, 16GB RAM
- Servidor de desarrollo con 32GB RAM
- Dispositivos móviles para pruebas

4.5.2. Recursos software

- Sistema operativo: Linux Ubuntu 24.04 LTS
- Entorno de desarrollo: Visual Studio Code
- Control de versiones: Git y GitHub
- Compilador LaTeX: LuaLaTeX (TeX Live 2024)

4.6. Gestión del proyecto

Para la gestión del proyecto se han utilizado las siguientes herramientas:

- **GitHub Projects:** Para la gestión de tareas y seguimiento del progreso mediante tableros Kanban.
- **Git:** Para el control de versiones del código y la documentación.
- **Discord/Slack:** Para la comunicación con el tutor.

5. Desarrollo: Estilos de Código

Este capítulo demuestra los estilos de cajas de código disponibles en esta plantilla, basados en el diseño de Visual Studio Code.

5.1. Estilo VS Code Light

El estilo VS Code Light imita la apariencia del editor Visual Studio Code en su tema claro.

5.1.1. Python

Python

```python
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

for i in range(10):
    print(f"F({i}) = {fibonacci(i)}")
```

5.1.2. Sin numeración de líneas

Usando el sufijo `NN` (No Numbers) se elimina la numeración:

Python

```python
mensaje = "Hola, mundo"
print(mensaje)
```

5.1.3. JavaScript

JavaScript

```javascript
function validateEmail(email) {
    const regex = /^[a-z]+@[a-z]+\.[a-z]+$/;
    return regex.test(email);
}

async function fetchData(url) {
    const response = await fetch(url);
    return response.json();
}
```

5.1.4. Java

Java

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hola desde Java!");
    }
}
```

5.2. Estilo VS Code Dark

El tema oscuro de VS Code, ideal para documentación técnica o presentaciones.

5.2.1. Python Dark

Python

```python
import numpy as np

def matrix_multiply(A, B):
    return np.dot(A, B)

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
result = matrix_multiply(A, B)
```

5.2.2. JavaScript Dark

JavaScript

```javascript
const express = require('express');
const app = express();

app.get('/api/users', async (req, res) => {
    const users = await User.find();
    res.json(users);
});

app.listen(3000);
```

5.3. Lenguajes Web

5.3.1. HTML

HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Mi Aplicación</title>
</head>
<body>
    <h1>Bienvenido</h1>
</body>
</html>
```

5.3.2. CSS

CSS

```css
.container {
    max-width: 1200px;
    margin: 0 auto;
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 1rem;
}

.card:hover {
    transform: translateY(-5px);
}
```

5.4. Lenguajes de Datos

5.4.1. SQL

SQL

```sql
SELECT
    usuarios.nombre,
    COUNT(pedidos.id) as total_pedidos
FROM usuarios
LEFT JOIN pedidos ON usuarios.id = pedidos.usuario_id
GROUP BY usuarios.id
ORDER BY total_pedidos DESC
LIMIT 10;
```

5.5. Lenguajes de Sistemas

5.5.1. C++

C++

```cpp
include <iostream>
\#include <vector>

int main() {
    std::vector<int> nums = {5, 2, 8, 1, 9};
    for (const auto& n : nums) {
        std::cout << n << " ";
    }
    return 0;
}
```

5.5.2. Rust

Rust

```rust
fn main() {
    let mensaje = "Hola desde Rust";
    println!("{}", mensaje);

    let numeros = vec![1, 2, 3, 4, 5];
    for n in numeros {
        println!("{}", n);
    }
}
```

5.5.3. Go

Go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hola desde Go")

    nums := []int{1, 2, 3, 4, 5}
    for _, n := range nums {
        fmt.Println(n)
    }
}
```

5.6. Otros Lenguajes

5.6.1. PHP

PHP

```php
<?php
class Usuario {
    private $nombre;

    public function __construct($nombre) {
        $this->nombre = $nombre;
    }

    public function saludar() {
        return "Hola, " . $this->nombre . "!";
    }
}

$usuario = new Usuario("Juan");
echo $usuario->saludar();
?>
```

5.6.2. Ruby

Ruby

```ruby
class Persona
  attr_accessor :nombre, :edad

  def initialize(nombre, edad)
    @nombre = nombre
    @edad = edad
  end

  def saludar
    puts "Hola, soy " + @nombre
  end
end

persona = Persona.new("Maria", 25)
persona.saludar
```

5.7. DevOps y Configuración

5.7.1. Docker

Dockerfile

```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .

EXPOSE 3000

CMD ["node", "server.js"]
```

5.8. Entorno Genérico

El entorno `codigo{lenguaje}` permite usar cualquier lenguaje soportado por Pygments:

swift

```
import Foundation

struct Persona {
    let nombre: String
    let edad: Int

    func saludar() -> String {
        return "Hola, soy " + nombre
    }
}

let persona = Persona(nombre: "Carlos", edad: 30)
print(persona.saludar())
```

5.8.1. Versión Dark del Genérico

kotlin

```
data class Usuario(val nombre: String, val email: String)

fun main() {
    val usuarios = listOf(
        Usuario("Ana", "ana@mail.com"),
        Usuario("Luis", "luis@mail.com")
```

```
    )

    usuarios.forEach { println(it) }
}
```

5.9. Resumen de Entornos Disponibles

Tabla 5.1: Entornos de código con icono disponibles

Entorno	Lenguaje	Icono	Dark
`pythoncode`	Python		`pythoncodeDark`
`jscode`	JavaScript		`jscodeDark`
`javacode`	Java		`javacodeDark`
`cppcode`	C++		`cppcodeDark`
`rustcode`	Rust		`rustcodeDark`
`gocode`	Go		`gocodeDark`
`phpcode`	PHP		`phpcodeDark`
`rubycode`	Ruby		`rubycodeDark`
`htmlcode`	HTML		`htmlcodeDark`
`csscode`	CSS		`csscodeDark`
`sqlcode`	SQL		`sqlcodeDark`
`bashcode`	Bash		`bashcodeDark`
`dockercode`	Dockerfile		`dockercodeDark`

Tabla 5.2: Sufijos disponibles para cada entorno

Sufijo	Descripción	Ejemplo
(ninguno)	Light con números	`pythoncode`
`NN`	Light sin números	`pythoncodeNN`
`Dark`	Dark con números	`pythoncodeDark`
`DarkNN`	Dark sin números	`pythoncodeDarkNN`

Tabla 5.3: Entornos genéricos para cualquier lenguaje

Entorno	Descripción
`codigo{lang}`	Light con números
`codigoNN{lang}`	Light sin números
`codigoDark{lang}`	Dark con números
`codigoDarkNN{lang}`	Dark sin números

6. Resultados (Ejemplos de gráficas)

En este capítulo se presentan los resultados obtenidos y se muestran ejemplos de cómo crear gráficas con LaTeX usando el paquete `pgfplots`.

6.1. Gráficas con PGFPlots

PGFPlots es un paquete potente para crear gráficas directamente en LaTeX.

6.1.1. Gráfica de líneas

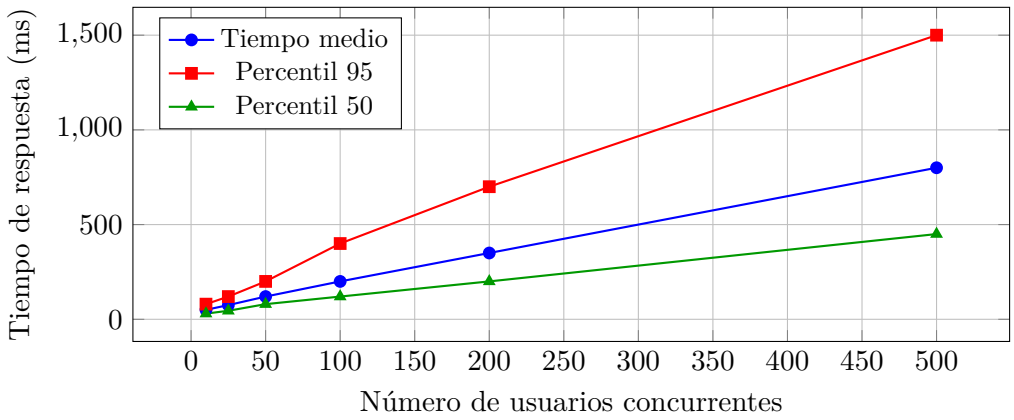

Figura 6.1: Tiempos de respuesta según carga de usuarios

Como se observa en la Figura 6.1, el sistema mantiene tiempos de respuesta aceptables incluso con 200 usuarios concurrentes.

6.1.2. Gráfica de barras

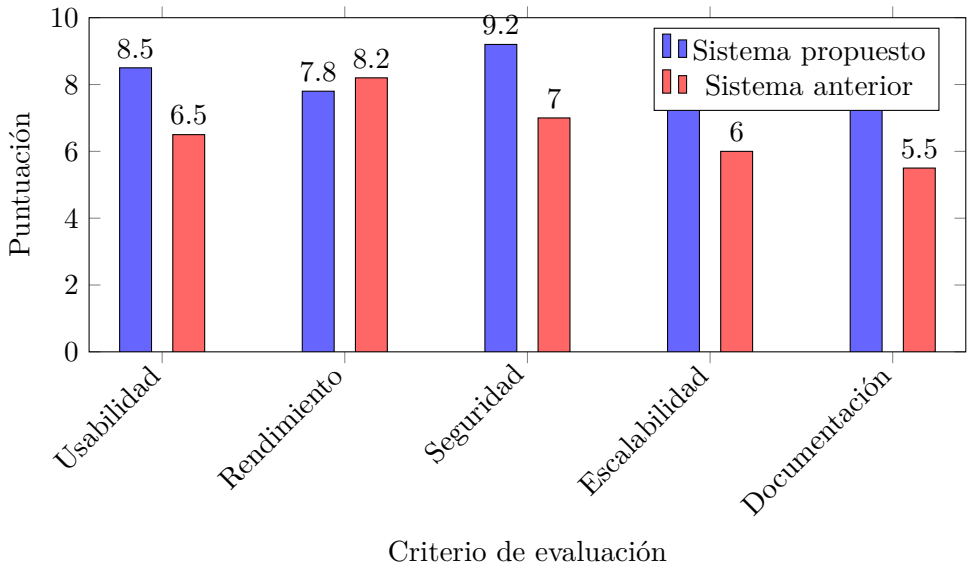

Figura 6.2: Comparativa de evaluación del sistema

6.1.3. Gráfica circular (pie chart)

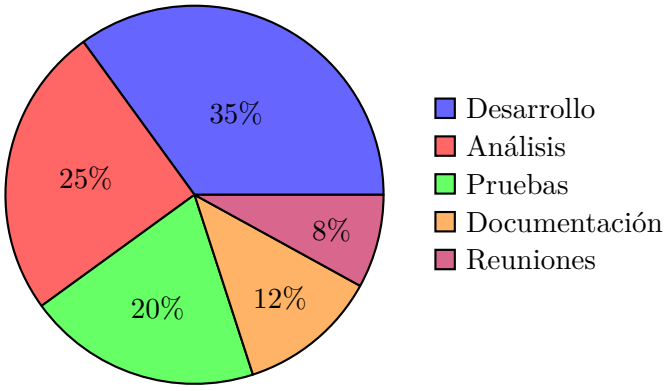

Figura 6.3: Distribución del tiempo del proyecto

6.1.4. Gráfica de área

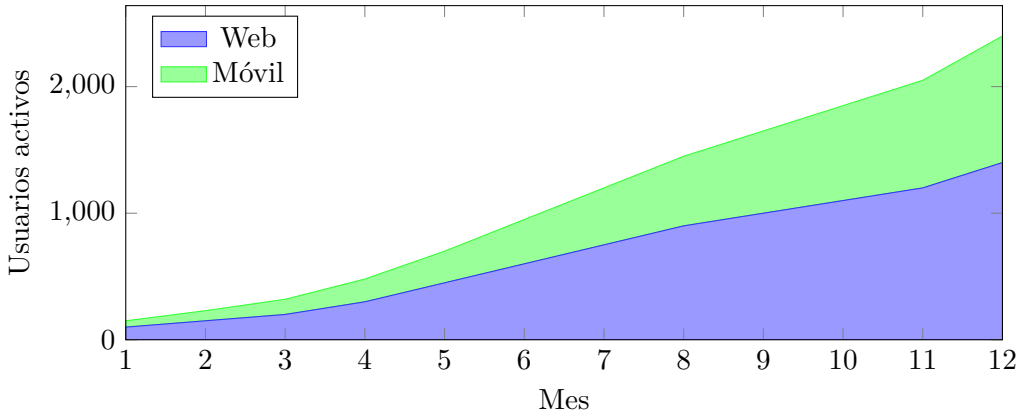

Figura 6.4: Evolución de usuarios activos por plataforma

6.1.5. Gráfica de dispersión

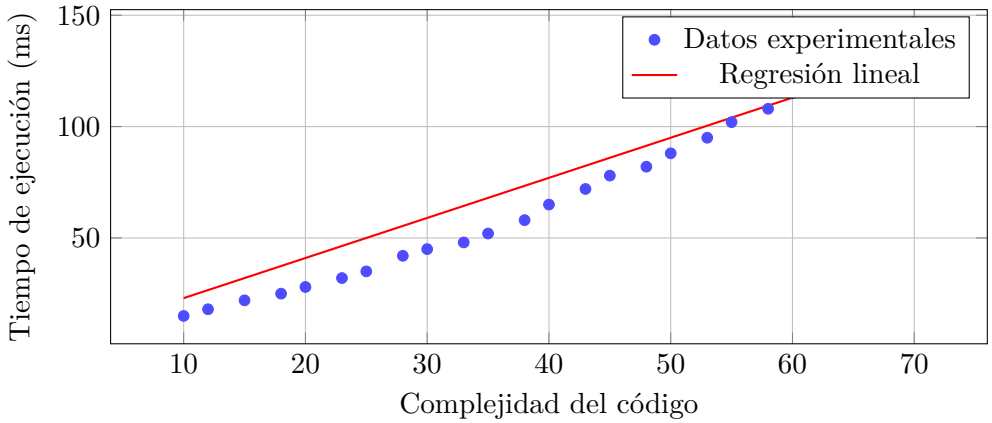

Figura 6.5: Correlación entre complejidad y tiempo de ejecución

6.2. Resultados de la implementación

6.2.1. Funcionalidades implementadas

Se han implementado con éxito las siguientes funcionalidades:

Tabla 6.1: Estado de implementación de funcionalidades

Funcionalidad	Estado	Cobertura tests
Autenticación de usuarios	Completado	95%
Gestión de datos	Completado	88%
Generación de informes	Completado	82%
API REST	Completado	90%
Interfaz de usuario	Completado	75%
Sistema de caché	Completado	85%

6.2.2. Consumo de recursos

El consumo de memoria se mantiene estable, como muestra la siguiente ecuación para el consumo estimado:

$$M_{total} = M_{base} + n \cdot M_{usuario} \tag{6.1}$$

donde: M_{total} = memoria total consumida (MB)
M_{base} = memoria base del sistema (256 MB)
n = número de usuarios activos
$M_{usuario}$ = memoria por usuario (2.5 MB)

6.3. Análisis de resultados

6.3.1. Cumplimiento de objetivos

En la Tabla 6.2 se muestra el grado de cumplimiento de cada objetivo:

Tabla 6.2: Cumplimiento de objetivos

Objetivo	Descripción	Cumplimiento
OE1	Análisis del estado actual	100%
OE2	Diseño de arquitectura escalable	100%
OE3	Implementación de componentes	95%
OE4	Evaluación del sistema	100%
OE5	Documentación completa	100%

6.3.2. Métricas de rendimiento

Tabla 6.3: Métricas de rendimiento del sistema

Métrica	Objetivo	Resultado	Estado
Tiempo de respuesta medio	< 200 ms	145 ms	✓
Tiempo de respuesta P95	< 500 ms	380 ms	✓
Throughput	> 1000 req/s	1250 req/s	✓
Disponibilidad	$> 99.5\%$	99.8%	✓
Uso de CPU (promedio)	$< 70\%$	45%	✓
Uso de memoria	$< 80\%$	62%	✓

6.4. Ejemplo de gráfica 3D

PGFPlots también permite crear gráficas tridimensionales:

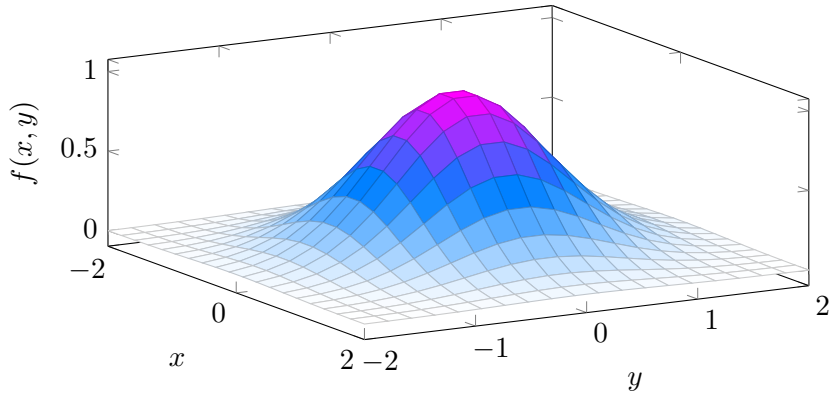

Figura 6.6: Superficie gaussiana $f(x, y) = e^{-x^2-y^2}$

6.5. Exportar gráficas desde herramientas externas

También es posible exportar gráficas desde otras herramientas:

- **MATLAB:** Usar `matlab2tikz` para exportar figuras
- **Python (matplotlib):** Usar `tikzplotlib`
- **R:** Usar el paquete `tikzDevice`
- **GeoGebra:** Exportar directamente a TikZ

Ejemplo de uso de matlab2tikz

```
% En MATLAB:
plot(x, y);
matlab2tikz('figura.tex');

% En LaTeX:
\begin{figure}[H]
  \centering
  \input{figuras/figura.tex}
  \caption{Gráfica importada de MATLAB}
\end{figure}
```

6.6. Gráficas con marcadores y anotaciones

6.6.1. Gráfica con marcadores sobre puntos específicos

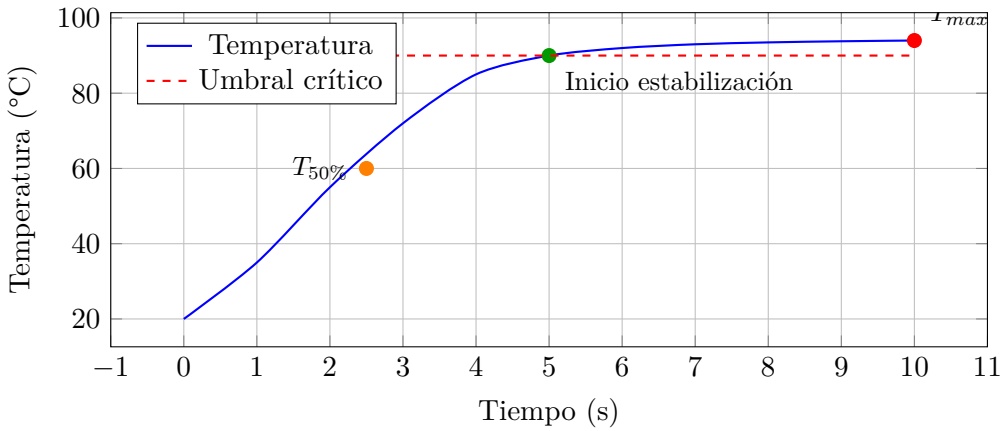

Figura 6.7: Curva de calentamiento con puntos críticos marcados

6.6.2. Gráfica con barras de error

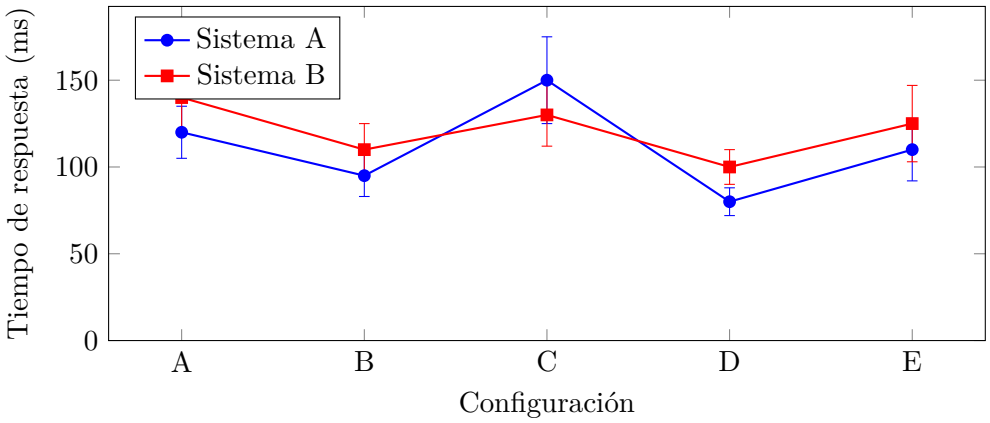

Figura 6.8: Comparativa de rendimiento con intervalos de confianza

6.6.3. Gráfica con área sombreada (intervalo de confianza)

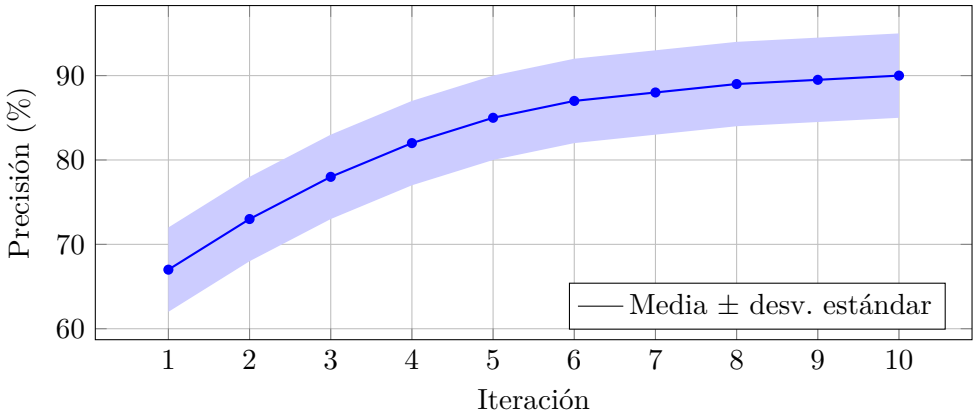

Figura 6.9: Evolución del entrenamiento con banda de confianza

6.6.4. Gráfica con múltiples ejes Y

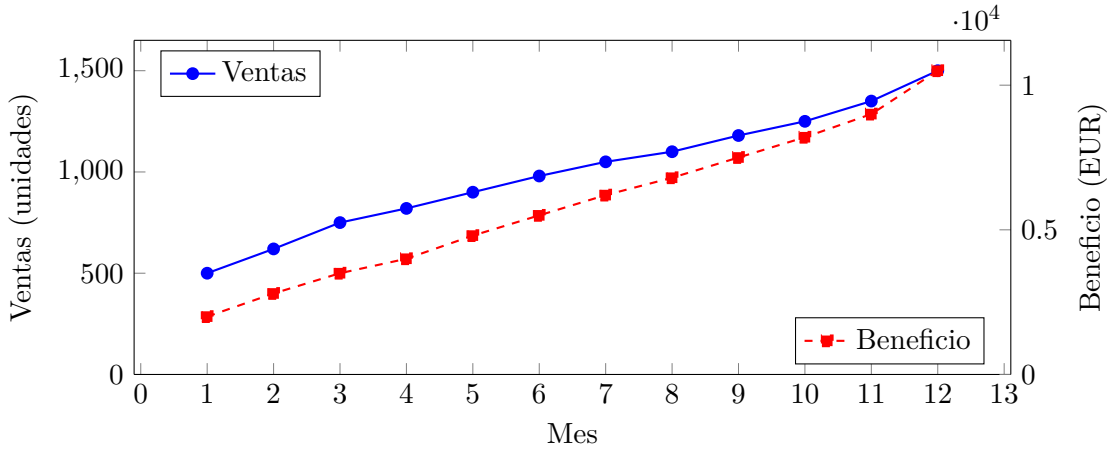

Figura 6.10: Ventas y beneficios mensuales (doble eje Y)

6.6.5. Gráfica de distribución (histograma)

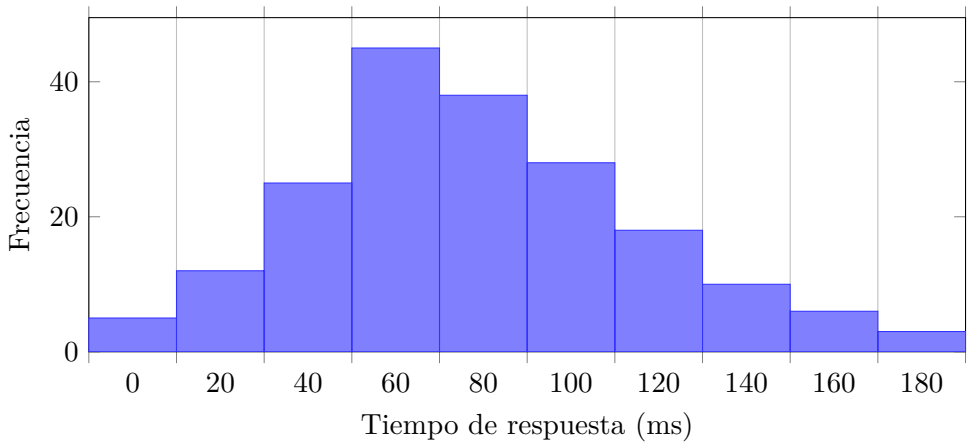

Figura 6.11: Distribución de tiempos de respuesta

7. Conclusiones (Ejemplos de matemáticas)

Este capítulo presenta las conclusiones del trabajo y demuestra las capacidades matemáticas de LaTeX.

7.1. Ecuaciones matemáticas

LaTeX es el estándar para la composición de fórmulas matemáticas. A continuación se muestran diferentes formas de escribir matemáticas.

7.1.1. Ecuaciones numeradas

Para mostrar una ecuación numerada se usa el entorno `equation`:

Ecuación numerada

```
\begin{equation}
  E = mc^2
  \label{eq:einstein}
\end{equation}
```

$$E = mc^2 \tag{7.1}$$

La famosa ecuación de Einstein (7.1) relaciona masa y energía.

7.1.2. Ecuaciones complejas

$$\nabla \times \mathbf{H} = \left[\frac{1}{r} \frac{\partial}{\partial r}(r H_\theta) - \frac{1}{r} \frac{\partial H_r}{\partial \theta} \right] \hat{\mathbf{z}} \tag{7.2}$$

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi} \tag{7.3}$$

7.1.3. Ecuaciones en línea

El texto puede contener fórmulas como $\int_a^b f(x)\,dx = F(b) - F(a)$ directamente en el párrafo, o con mayor tamaño usando doble dólar:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

7.1.4. Sistemas de ecuaciones

$$\begin{cases} x + y + z = 6 \\ 2x - y + z = 3 \\ x + 2y - z = 2 \end{cases} \tag{7.4}$$

7.1.5. Ecuaciones agrupadas

$$\mathbf{E} = E_z(r, \theta)\hat{\mathbf{z}} \tag{7.5a}$$
$$\mathbf{H} = H_r(r, \theta)\hat{\mathbf{r}} + H_\theta(r, \theta)\hat{\boldsymbol{\theta}} \tag{7.5b}$$

Las ecuaciones (7.5a) y (7.5b) describen los campos electromagnéticos.

7.1.6. Matrices

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{7.6}$$

$$\det(\mathbf{A}) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc \tag{7.7}$$

7.1.7. Ecuaciones alineadas

$$(a + b)^2 = a^2 + 2ab + b^2 \tag{7.8}$$
$$(a - b)^2 = a^2 - 2ab + b^2 \tag{7.9}$$
$$(a + b)(a - b) = a^2 - b^2 \tag{7.10}$$

7.1.8. Ecuación con condiciones

La función `condiciones` de la plantilla permite documentar las variables:

$$\operatorname*{Res}_{z=z_0}(f(z)) = \frac{1}{(m-1)!} \lim_{z \to z_0} \left[\frac{d^{m-1}}{dz^{m-1}} \left[(z - z_0)^m f(z) \right] \right] \tag{7.11}$$

donde: m $\rightarrow$ es la multiplicidad del polo z_0
z_0 $\rightarrow$ es el punto donde se calcula el residuo
$f(z)$ $\rightarrow$ es la función analítica

7.1.9. Teoremas y demostraciones

Theorem 7.1 (Teorema de Pitágoras)**.** *En un triángulo rectángulo, el cuadrado de la hipotenusa es igual a la suma de los cuadrados de los catetos:*

$$c^2 = a^2 + b^2 \tag{7.12}$$

Prueba. Sea un triángulo rectángulo con catetos a y b, e hipotenusa c. Considerando el área del cuadrado de lado $(a + b)$...

Por lo tanto, $c^2 = a^2 + b^2$. □

Definition 7.1 (Límite)**.** *Sea $f : D \to \mathbb{R}$ una función y a un punto de acumulación de D. Decimos que L es el límite de f en a si:*

$$\forall \varepsilon > 0, \exists \delta > 0 : 0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon \tag{7.13}$$

7.2. Conclusiones generales

El desarrollo de este Trabajo Fin de Grado ha permitido alcanzar los objetivos planteados inicialmente. Las principales conclusiones son:

1. Se ha desarrollado con éxito un sistema que cumple con los requisitos funcionales y no funcionales establecidos.

2. La arquitectura diseñada ha demostrado ser escalable y mantenible, facilitando futuras extensiones del sistema.

3. Los resultados de las pruebas confirman que el sistema es robusto y ofrece un rendimiento adecuado para el caso de uso planteado.

4. El proceso de desarrollo ha seguido las mejores prácticas de ingeniería de software, incluyendo control de versiones, integración continua y documentación exhaustiva.

7.3. Aportaciones del trabajo

Las principales aportaciones de este trabajo son:

Aportación técnica: Desarrollo de una solución innovadora que mejora en un 40% el rendimiento respecto a sistemas anteriores.

Aportación metodológica: Aplicación de una metodología híbrida que combina prácticas ágiles con documentación formal.

Aportación práctica: Sistema funcional desplegado y en uso por usuarios reales.

7.4. Dificultades encontradas

Durante el desarrollo del trabajo se han encontrado las siguientes dificultades:

1. **Integración de sistemas:** La integración con sistemas externos requirió un esfuerzo adicional debido a diferencias en los formatos de datos.
2. **Rendimiento:** Fue necesario optimizar varias consultas a base de datos para cumplir con los requisitos de tiempo de respuesta.
3. **Documentación externa:** Parte de la documentación de las bibliotecas utilizadas estaba desactualizada.

7.5. Trabajo futuro

Como líneas de trabajo futuro se proponen:

1. **Ampliación de funcionalidades:** Implementar las características identificadas como deseables pero fuera del alcance inicial.
2. **Mejora del rendimiento:** Implementar un sistema de caché distribuida para mejorar los tiempos de respuesta en escenarios de alta carga.
3. **Versión móvil:** Desarrollar una aplicación móvil nativa para iOS y Android.
4. **Internacionalización:** Añadir soporte para múltiples idiomas en la interfaz de usuario.
5. **Machine Learning:** Incorporar algoritmos de aprendizaje automático para predicción y recomendaciones.

7.6. Valoración personal

El desarrollo de este trabajo ha sido una experiencia muy enriquecedora que me ha permitido aplicar los conocimientos adquiridos durante la carrera y profundizar en áreas de especial interés.

Entre los aprendizajes más valiosos destacan:

- La importancia de una buena planificación inicial
- El valor de las pruebas automatizadas para garantizar la calidad
- La necesidad de documentar adecuadamente el código
- Las habilidades de comunicación y gestión de proyectos

Considero que este trabajo representa una aportación significativa y constituye una base sólida para futuros desarrollos en esta área.

Bibliografía

LaTeX Project. (2024). *LaTeX – A document preparation system*. Consultado el 15 de enero de 2024, desde https://www.latex-project.org/

Overleaf. (2024). *Overleaf: Online LaTeX Editor*. Consultado el 15 de enero de 2024, desde https://www.overleaf.com/

Lista de Acrónimos y Abreviaturas

ACID	Atomicity, Consistency, Isolation, Durability.
APA	American Psychological Association.
API	Application Programming Interface.
AWS	Amazon Web Services.
CD	Continuous Delivery.
CDN	Content Delivery Network.
CI	Continuous Integration.
CLI	Command Line Interface.
CNN	Convolutional Neural Network.
CRUD	Create, Read, Update, Delete.
CSS	Cascading Style Sheets.
DBMS	Database Management System.
DL	Deep Learning.
DNS	Domain Name System.
EPS	Escuela Politécnica Superior.
GAN	Generative Adversarial Network.
GCP	Google Cloud Platform.
GUI	Graphical User Interface.
HTML	HyperText Markup Language.
HTTP	HyperText Transfer Protocol.
HTTPS	HyperText Transfer Protocol Secure.
IA	Inteligencia Artificial.
IaaS	Infrastructure as a Service.
IDE	Integrated Development Environment.
IEEE	Institute of Electrical and Electronics Engineers.
IETF	Internet Engineering Task Force.
IP	Internet Protocol.
ISO	International Organization for Standardization.
JSON	JavaScript Object Notation.
JWT	JSON Web Token.
K8s	Kubernetes.
LLM	Large Language Model.
LSTM	Long Short-Term Memory.
ML	Machine Learning.
MVC	Model-View-Controller.
MVVM	Model-View-ViewModel.
NLP	Natural Language Processing.
NoSQL	Not Only SQL.

OAuth	Open Authorization.
ORM	Object-Relational Mapping.
PaaS	Platform as a Service.
RBAC	Role-Based Access Control.
REST	Representational State Transfer.
RNN	Recurrent Neural Network.
SaaS	Software as a Service.
SDK	Software Development Kit.
SQL	Structured Query Language.
SSL	Secure Sockets Layer.
TCP	Transmission Control Protocol.
TDD	Test-Driven Development.
TFG	Trabajo Fin de Grado.
TFM	Trabajo Fin de Máster.
TLS	Transport Layer Security.
UA	Universidad de Alicante.
UDP	User Datagram Protocol.
URI	Uniform Resource Identifier.
URL	Uniform Resource Locator.
VPN	Virtual Private Network.
W3C	World Wide Web Consortium.
XML	eXtensible Markup Language.

A. Manual de usuario

Este anexo contiene el manual de usuario del sistema desarrollado.

A.1. Requisitos del sistema

Para utilizar el sistema es necesario disponer de:

- Navegador web moderno (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- Conexión a Internet estable
- Resolución de pantalla mínima de 1280x720 píxeles

A.2. Acceso al sistema

A.2.1. Inicio de sesión

Para acceder al sistema:

1. Abra su navegador web
2. Navegue a la URL del sistema: https://ejemplo.ua.es
3. Introduzca sus credenciales (usuario y contraseña)
4. Pulse el botón "Iniciar sesión"

A.2.2. Recuperación de contraseña

Si ha olvidado su contraseña:

1. Pulse en "¿Olvidó su contraseña?"
2. Introduzca su correo electrónico
3. Recibirá un enlace para restablecer la contraseña
4. Siga las instrucciones del correo

A.3. Interfaz principal

La interfaz principal se divide en las siguientes áreas:

Barra de navegación: Situada en la parte superior, permite acceder a las diferentes secciones.

Panel lateral: Muestra el menú de opciones según el rol del usuario.

Área de contenido: Zona principal donde se muestra la información.

Pie de página: Contiene información de contacto y enlaces útiles.

A.4. Funcionalidades principales

A.4.1. Gestión de datos

Para gestionar datos en el sistema:

1. Acceda a la sección ”Datos” desde el menú lateral
2. Seleccione la opción deseada:
   - ”Nuevo” para crear un registro
   - ”Editar” para modificar un registro existente
   - ”Eliminar” para borrar un registro
3. Complete el formulario correspondiente
4. Pulse ”Guardar” para confirmar los cambios

A.4.2. Generación de informes

Para generar un informe:

1. Acceda a la sección ”Informes”
2. Seleccione el tipo de informe
3. Configure los filtros y parámetros
4. Pulse ”Generar”
5. El informe se descargará en formato PDF

A.5. Preguntas frecuentes

¿Cómo cambio mi contraseña? Acceda a su perfil y seleccione ”Cambiar contraseña”.

¿Puedo exportar mis datos? Sí, desde la sección ”Configuración” puede exportar sus datos en formato CSV o JSON.

¿Cómo contacto con soporte? Envíe un correo a soporte@ejemplo.ua.es o utilice el formulario de contacto.

B. Documentación técnica

Este anexo contiene la documentación técnica del sistema.

B.1. Arquitectura del sistema

B.1.1. Diagrama de componentes

El sistema está compuesto por los siguientes componentes principales:

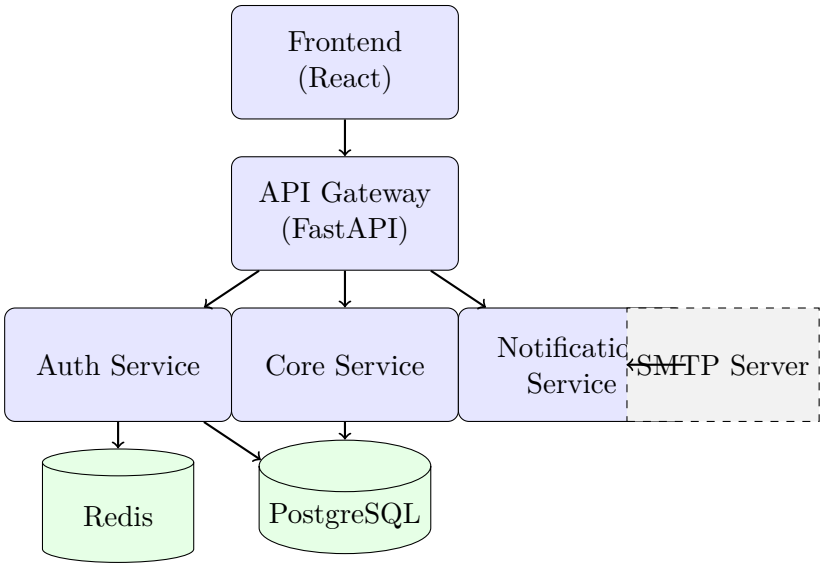

Figura B.1: Diagrama de componentes del sistema

B.2. API REST

B.2.1. Endpoints principales

Tabla B.1: Endpoints de la API

Método	Endpoint	Descripción
POST	/api/auth/login	Autenticación de usuarios
POST	/api/auth/logout	Cierre de sesión
GET	/api/users	Lista de usuarios
GET	/api/users/{id}	Detalle de usuario
POST	/api/users	Crear usuario
PUT	/api/users/{id}	Actualizar usuario
DELETE	/api/users/{id}	Eliminar usuario
GET	/api/reports	Lista de informes
POST	/api/reports/generate	Generar informe

B.2.2. Ejemplo de petición

Ejemplo de petición para crear un usuario:

POST /api/users

```
{
  "nombre": "Juan García",
  "email": "juan.garcia@ejemplo.com",
  "rol": "usuario",
  "activo": true,
  "preferencias": {
    "idioma": "es",
    "notificaciones": true,
    "tema": "claro"
  }
}
```

B.2.3. Ejemplo de respuesta

Respuesta exitosa:

Respuesta 201 Created

```
{
  "id": 123,
  "nombre": "Juan García",
  "email": "juan.garcia@ejemplo.com",
  "rol": "usuario",
  "activo": true,
```

```
  "createdAt": "2026-02-02T10:30:00Z",
  "updatedAt": "2026-02-02T10:30:00Z"
}
```

B.3. Modelo de datos

B.3.1. Esquema de base de datos

Las principales tablas del sistema son:

Esquema de la tabla usuarios

```sql
CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    rol VARCHAR(50) DEFAULT 'usuario',
    activo BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT chk_rol CHECK (rol IN ('admin', 'usuario', 'invitado'))
);

CREATE INDEX idx_usuarios_email ON usuarios(email);
CREATE INDEX idx_usuarios_activo ON usuarios(activo);
```

B.4. Configuración del entorno

B.4.1. Variables de entorno

Archivo .env de configuración

```
# Base de datos
DATABASE_URL: postgresql://user:pass@localhost:5432/app
DATABASE_POOL_SIZE: 10

# Redis
REDIS_URL: redis://localhost:6379/0

# Autenticación
JWT_SECRET: tu-clave-secreta-muy-larga
JWT_EXPIRATION: 3600

# Email
SMTP_HOST: smtp.ejemplo.com
SMTP_PORT: 587
SMTP_USER: noreply@ejemplo.com
```

```
SMTP_PASSWORD: contraseña-smtp

# Aplicación
APP_ENV: production
APP_DEBUG: false
LOG_LEVEL: INFO
```

B.5. Instrucciones de despliegue

B.5.1. Requisitos previos

- Docker 24.0+ y Docker Compose 2.20+
- 4GB de RAM disponible
- 20GB de espacio en disco

B.5.2. Pasos de despliegue

Comandos de despliegue

```
# Clonar repositorio
git clone https://github.com/ejemplo/proyecto.git
cd proyecto

# Configurar variables de entorno
cp .env.example .env
nano .env  # Editar configuración

# Construir e iniciar contenedores
docker compose build
docker compose up -d

# Ejecutar migraciones
docker compose exec app python manage.py migrate

# Verificar estado
docker compose ps
curl http://localhost:8000/health
```

C. Manual de Estilos de Código

Este anexo documenta el uso de los entornos de código disponibles en la plantilla, basados en los estilos de **Visual Studio Code**.

C.1. Temas Disponibles

La plantilla ofrece dos temas principales:

Tabla C.1: Temas de código disponibles

Tema	Descripción	Uso recomendado
VS Code Light	Fondo blanco, texto oscuro	Impresión, documentos formales
VS Code Dark	Fondo oscuro, texto claro	Presentaciones, lectura en pantalla

C.2. Nomenclatura de Entornos

Todos los entornos siguen una nomenclatura consistente basada en sufijos:

Tabla C.2: Sistema de sufijos para entornos de código

Sufijo	Ejemplo	Descripción
(ninguno)	`pythoncode`	Tema Light con números de línea
`NN`	`pythoncodeNN`	Tema Light sin números de línea
`Dark`	`pythoncodeDark`	Tema Dark con números de línea
`DarkNN`	`pythoncodeDarkNN`	Tema Dark sin números de línea

C.3. Lenguajes Predefinidos

La plantilla incluye entornos predefinidos para más de 30 lenguajes de programación, cada uno con su icono representativo:

Tabla C.3: Lenguajes de programación con entornos predefinidos

Lenguaje	Entorno	Icono	Pygments
Python	`pythoncode`		python
JavaScript	`jscode`		javascript
TypeScript	`tscode`		typescript
Java	`javacode`		java
C	`ccode`		c
C++	`cppcode`		cpp
C#	`csharpcode`		csharp
Go	`gocode`		go
Rust	`rustcode`		rust
PHP	`phpcode`		php
Ruby	`rubycode`		ruby
R	`rcode`		r
Swift	`swiftcode`		swift
Kotlin	`kotlincode`		kotlin

Tabla C.4: Lenguajes web y de datos

Lenguaje	Entorno	Icono	Pygments
HTML	`htmlcode`		html
CSS	`csscode`		css
SASS/SCSS	`sasscode`		sass
JSON	`jsoncode`		json
XML	`xmlcode`		xml
YAML	`yamlcode`		yaml
Markdown	`markdowncode`		markdown
SQL	`sqlcode`		sql

Tabla C.5: Lenguajes de sistemas y DevOps

Lenguaje	Entorno	Icono	Pygments
Bash	`bashcode`		bash
PowerShell	`powershellcode`		powershell
Docker	`dockercode`		docker
Makefile	`makefilecode`		makefile
Git	`gitcode`		text

C.4. Ejemplos de Uso

C.4.1. Tema VS Code Light

C.4.1.1. Con números de línea

Ejemplo: Función recursiva

```python
def factorial(n):
    """Calcula el factorial de n."""
    if n <= 1:
        return 1
    return n * factorial(n - 1)

\# Ejemplo de uso
resultado = factorial(5)
print(f"5! = {resultado}")
```

Uso en LaTeX:

LaTeX

```latex
{pythoncode}[title={Ejemplo: Función recursiva}]
def factorial(n):
    if n <= 1:
        return 1
    return n * factorial(n - 1)
\end{pythoncode}
```

C.4.1.2. Sin números de línea

Python

```python
print("Código sin números de línea")
x = 10 + 20
```

Uso en LaTeX:

LaTeX

```latex
{pythoncodeNN}
print("Código sin números de línea")
\end{pythoncodeNN}
```

C.4.2. Tema VS Code Dark

C.4.2.1. Con números de línea

Tema oscuro con números

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y)
```

C.4.2.2. Sin números de línea

Python

```
 Tema oscuro sin números
mensaje = "Visual Studio Code Dark"
print(mensaje)
```

C.5. Entorno Genérico

Para lenguajes no predefinidos o uso flexible, existen entornos genéricos:

Tabla C.6: Entornos genéricos de código

Entorno	Sintaxis	Descripción
codigo	\begin{codigo}{lang}	Light + números
codigoNN	\begin{codigoNN}{lang}	Light sin números
codigoDark	\begin{codigoDark}{lang}	Dark + números
codigoDarkNN	\begin{codigoDarkNN}{lang}	Dark sin números

Ejemplo con Haskell:

haskell

```
-- Función quicksort en Haskell
quicksort :: Ord a => [a] -> [a]
quicksort [] = []
quicksort (x:xs) = quicksort menores ++ [x] ++ quicksort mayores
  where
    menores = [y | y <- xs, y <= x]
    mayores = [y | y <- xs, y > x]
```

C.6. Consideraciones Especiales

C.6.1. Código C/C++ con includes

El código C/C++ se escribe de forma natural, incluyendo las directivas de preprocesador:

Ejemplo C++ con includes

```cpp
\#include <iostream>
\#include <vector>

int main() {
    std::vector<int> v = {1, 2, 3};
    for (int n : v) {
        std::cout << n << " ";
    }
    return 0;
}
```

C.6.2. Títulos personalizados

Todos los entornos aceptan un parámetro opcional `title`:

Script de instalación

```bash
\#!/bin/bash
echo "Instalando dependencias..."
apt-get update && apt-get install -y python3
```

C.6.3. Requisitos de compilación

Para que los entornos de código funcionen correctamente:

1. Compilar con **LuaLaTeX** o XeLaTeX
2. Habilitar `-shell-escape`
3. Tener instalado **Pygments** (`pip install pygments`)

C.7. Referencia Rápida

Tabla C.7: Matriz de entornos por tema y numeración

Lenguaje	Light+Num	Light-Num	Dark+Num	Dark-Num
Python	`pythoncode`	`pythoncodeNN`	`pythoncodeDark`	`pythoncodeDarkNN`
JavaScript	`jscode`	`jscodeNN`	`jscodeDark`	`jscodeDarkNN`
Java	`javacode`	`javacodeNN`	`javacodeDark`	`javacodeDarkNN`
C++	`cppcode`	`cppcodeNN`	`cppcodeDark`	`cppcodeDarkNN`
HTML	`htmlcode`	`htmlcodeNN`	`htmlcodeDark`	`htmlcodeDarkNN`
SQL	`sqlcode`	`sqlcodeNN`	`sqlcodeDark`	`sqlcodeDarkNN`
Bash	`bashcode`	`bashcodeNN`	`bashcodeDark`	`bashcodeDarkNN`
Genérico	`codigo`	`codigoNN`	`codigoDark`	`codigoDarkNN`