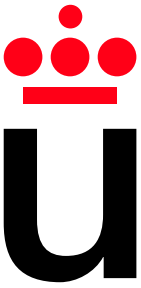


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

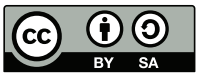
Grado en Ingeniería XXX

Curso 202X-202Y

Trabajo Fin de Grado

TÍTULO DEL TRABAJO DE FIN DE GRADO

Autor: Nombre Apellido1 Apellido2
Tutor: NombreTutor Apellido1 Apellido2



Agradecimientos

Breves agradecimientos o dedicatoria.

Resumen

Una página resumiendo el contenido de la memoria. Esta página se escribe cuando ya se ha escrito toda la memoria para poder resumirla mejor. El último párrafo del resumen suele indicar los capítulos que contiene el documento y qué hay en cada uno.

Palabras clave:

- Python
- Ciberseguridad
- Aprendizaje automático (pueden ser varias)
- ...

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción y objetivos

Tutor: Extensión del capítulo: 4/8 páginas

Contexto en el que se enmarca el proyecto y la justificación de este.

1.1. Contexto

Contexto en el que se enmarca el proyecto y la justificación de este. Esta sección explica la utilidad de la aplicación, el contexto en el que sería usada y las funcionalidades principales que tendría y por qué es relevante su implementación, etc. No se especifican aspectos técnicos.

Tutor: Debe crearse nueva

1.2. Estado del arte

Descripción de otras aplicaciones y plataformas web que ofrecen funcionalidad parecida a la que se ha implementado y se destaca en qué aspectos es diferente tu página.

Tutor: Debe crearse nueva

1.3. Objetivos

Objetivos que se plantearon cuando se inició este trabajo.

Tutor: Sección "Objetivos" de la documentación de repositorio

1.3.1. Objetivos funcionales

Un párrafo resumiendo los objetivos funcionales que se plantearon al comenzar el desarrollo de la aplicación.

Una lista de 3-10 funcionalidades detallando un poco más el párrafo de objetivos funcionales.

- Objetivo funcional 1
- Objetivo funcional 2
- Objetivo funcional 3
- ...

1.3.2. Objetivos técnicos

Un párrafo resumiendo los aspectos tecnológicos que se plantearon al comenzar el desarrollo de la aplicación.

Una lista de 3-10 aspectos técnicos detallando un poco más el párrafo de objetivos técnicos.

- Objetivo técnico 1
- Objetivo técnico 2
- Objetivo técnico 3
- ...

2 Metodología

Tutor: Extensión del capítulo: 1/2 páginas

Se describirá en alto nivel cómo se ha desarrollado el trabajo.

Tutor: Sección "Metodología" de la documentación de repositorio

Fases:

- Fase 1: Definición de funcionalidades. Se indicará las secciones concretas en la que se describe la funcionalidad general y detallada.
- Fase 2: Configuración de las tecnologías y herramientas de desarrollo con controles de calidad que se realizan de forma periódica. Se indicará la sección concreta en la que se describe la tecnología usada.
- Fases 3, 4, 5: Desarrollo iterativo e incremental de la aplicación de forma iterativa e incremental. Al final de cada fase se ha publicado una versión (release). Se indicará la sección concreta en la que se describe el proceso de desarrollo empleado.
- Fase 6: Escritura de la memoria.
- Fase 7: Preparación de la presentación.

Se incluirán las fechas de inicio y fin de cada fase.

Se elaborará un diagrama de Gantt en el que se muestren gráficamente estas fases.

No se mencionará ningún otro aspecto técnico (no se mencionará git, CI/CD, las tecnologías utilizadas...).

3

Descripción funcional

Tutor: Extensión del capítulo: 5/10 páginas.

Funcionalidades principales de la aplicación desarrollada.

Tutor: Sección funcionalidades de la documentación de repositorio

- Su contenido es el mismo que la sección "Funcionalidades" de la documentación del repositorio de código.
- Se incluirá el enlace al vídeo de YouTube.
- Se indicará que el listado detallado de funcionalidades se encuentra en el Anexo 1. Puedes referenciarlo así: Anexo A.
- Se indica que las instrucciones para ejecutar la aplicación están en el Anexo 2. Puedes referenciarlo así: Anexo B.

4

Descripción Informática

Contenido del capítulo

Este capítulo recoge todos los aspectos informáticos de la aplicación.

Tutor: Sección "Guía de desarrollo"de la documentación de repositorio salvo el subcapítulo 4.4, que debe crearse nuevo

Comenzar indicando dónde se puede encontrar el repositorio con el código de la aplicación (enlace a GitHub)

Párrafo indicando la arquitectura de despliegue de la aplicación web. Habitualmente se especificará que se trata de una aplicación web con arquitectura SPA (y habrá que explicar en qué consiste) y las partes que tiene (cliente, servidor y base de datos). Si es una aplicación distribuida se hará una descripción de alto nivel de este aspecto.

Resumen de esta descripción en formato tabla/lista:

- Tipo: web MVC, Web SPA, API REST, microservicios...
- Tecnologías: lenguajes, librerías, servicios adicionales.
- Herramientas: IDEs empleados y herramientas auxiliares.
- Control de calidad: Qué controles de calidad se aplican y qué tecnologías/herramientas se usan.
- Despliegue: Cómo se empaqueta, distribuye y despliega la aplicación (Docker, Kubernetes). Entorno de despliegue (si aplica en este TFG).
- Proceso de desarrollo: iterativa e incremental, git, DevOps (CI/CD).

Párrafo describiendo cada una de las secciones en las que se divide el capítulo (que se indican a continuación)

4.1. Tecnologías

Tutor: Extensión de la sección: 2/3 páginas. (salvo que haya alguna tecnología de partes optativas que necesite describirse en detalle)

Tecnologías que usa la aplicación para su ejecución (no las herramientas usadas para su desarrollo).

- Sólo se profundizará si se consideran poco conocidas. Si no, serán de un párrafo.
- Si no es evidente se indicará para qué se usan en el proyecto.
- En todas se indicará la URL oficial.

4.2. Herramientas

Tutor: Extensión de la sección: 2/3 páginas

IDEs empleados y herramientas auxiliares.

- Sólo se profundizará si se consideran poco conocidas. Si no, serán de un párrafo.
- Si no es evidente se indicará para qué se usan en el proyecto.
- En todas se indicará la URL oficial.

4.3. Arquitectura

Tutor: Extensión de la sección: 5/6 páginas

4.3.1. Despliegue

Arquitectura de despliegue (indicando procesos independientes y protocolos de comunicación)

4.3.2. Modelo del dominio

Entidades persistentes de la aplicación, sus atributos y sus relaciones.

4.3.3. API REST

Se describirá en alto nivel la API REST como un listado de los recursos y sus operaciones. Para mantener la documentación clara se recomienda dedicar una subsección a cada recurso disponible, indicando:

- Formato JSON del recurso (respuestas de consulta o detalles del recurso).
- Operaciones disponibles (endpoints) agrupadas por método HTTP.

Recurso Books

```
{
  "id": 1,
  "title": "Clean Architecture",
  "author": "Robert C. Martin",
  "isbn": "9780134494166"
}
```

Operaciones

- **GET /api/books/{id}** - Obtiene la información de un libro específico por su identificador.
- **GET /api/books** - Obtiene el listado completo de libros.
- **POST /api/books** - Crea un nuevo libro en el sistema.
- **PUT /api/books/{id}** - Actualiza la información completa de un libro existente.
- **PATCH /api/books/{id}** - Actualiza parcialmente la información de un libro existente.
- **DELETE /api/books/{id}** - Elimina un libro del sistema.

Tutor: Los alumnos deben documentar todos los recursos de la API siguiendo este esquema. Si existen más recursos (usuarios, préstamos, pedidos, etc.), se repetirá la estructura para cada uno.

4.3.4. Arquitectura del servidor

Diagrama de clases del servidor reflejando su separación por capas. Responsabilidades de cada capa (Controladores, servicios, repositorios...).

4.3.5. Arquitectura del cliente

Diagrama de clases del cliente reflejando su separación por capas. Responsabilidades de cada capa (Componentes, servicios...).

4.4. Implementación

Tutor: Extensión de la sección: 5/12 páginas

Aspectos relevantes de la implementación.

Tutor: Esta sección debe crearse nueva

Por ejemplo, se podría incluir alguno de los siguientes aspectos:

- Algoritmo complejo que se haya tenido que desarrollar.
- Integración entre librerías con la que hayan surgido problemas.
- Resolución de algún bug que haya sido especialmente problemático.
- Estudio que haya sido necesario para elegir una tecnología.
- Focalizar en alguna parte de la lógica de negocio que haya sido especialmente compleja y describirla en más detalle.

En esta sección se pueden incluir fragmentos de código fuente (ver Anexo D para ejemplos de cómo incluirlos).

4.5. Control de calidad

Tutor: Extensión de la sección: 3/4 páginas

Descripción de los controles de calidad que se han realizado.

Descripción de las pruebas automáticas de cliente y servidor:

- Tipos de pruebas
- Descripción de qué funcionalidades se prueban. Ya que están numeradas en el anexo, conviene que haya algún tipo de trazabilidad entre la prueba y la funcionalidad.
- Estadísticas de las pruebas (número, cobertura, etc.). Mostrar captura de pantalla de su ejecución.

Herramientas de análisis estático de código (si se ha usado):

- Captura de pantalla de los resultados de los análisis de código finales o a lo largo del tiempo.
- Métricas del tamaño del código (número de clases, número de líneas de código, separadas por tecnologías...)

4.6. Despliegue

Tutor: Extensión de la sección: 2/3 páginas

Cómo se realiza el empaquetado, distribución y despliegue

- Empaquetado y distribución: una única imagen Docker para cliente y servidor, docker compose para coordinar, uso de Kubernetes, serverless...). Se indicará la URL para acceder al artefacto de la aplicación (DockerHub).
- Despliegue: Computación en la nube (AWS, Azure...), tecnología Infraestructura como código (Cloud Formation...).

4.7. Proceso de desarrollo

Tutor: Extensión de la sección: 4/5 páginas

Descripción de los aspectos técnicos del proceso de desarrollo (de las fases 2 a 5).

Proceso iterativo e incremental, que sigue los principios del manifiesto ágil y se apoya en algunas de las buenas prácticas de Programación Extrema (XP) y Kanban. No se puede decir que se ha aplicado Scrum, porque no se aplica.

4.7.1. Gestión de tareas

GitHub Issues, GitHub Projects y gestión visual (tablero).

4.7.2. Git

Se describe que se ha usado un repositorio git y la estrategia de ramas utilizada. Métricas de uso de git: Número de commits, número de ramas, etc.

4.7.3. Integración y entrega continua

Se indican las tareas que realizan los flujos automáticos del sistema de integración continua (workflows de GitHub Actions).

4.7.4. Despliegue continuo y/o versionado

Descripción del procedimiento de lanzamiento de versiones (releases), fechas en las que se ha publicado cada versión, descripción de alto nivel de las funcionalidades que incluye, etc. Si se ha usado Despliegue continuo, debería definirse aquí y cómo se relaciona con las versiones. Se debería mencionar el impacto en los usuarios cuando se produce el despliegue.

5

Conclusiones y trabajos futuros

Tutor: Extensión del capítulo: 1/2 páginas.

5.1. Conclusiones

Reflexión sobre el trabajo realizado:

- Reflexión sobre los objetivos funcionales y cómo se han materializado en la aplicación.
- Reflexión sobre los objetivos técnicos y cómo se han materializado en la aplicación.
- Conclusiones personales sobre el trabajo realizado.

5.2. Trabajos futuros

Qué aspectos funcionales y técnicos podrían mejorarse para versiones futuras de la aplicación.

Bibliografía

[1] Autores, "Cómo escribir tfg," *IEEE Informática*, 3081. [Online]. Available: http://una.url

Anexos

A

Funcionalidades detalladas

Tutor: Sección funcionalidades detalladas de la documentación del repositorio de código

B

Instrucciones para ejecutar la aplicación

Tutor: Sección "Ejecución" de la documentación del repositorio de código

C

Ejecución y edición de código

Tutor: Sección "Guía de desarrollo - Ejecución y edición de código" de la documentación del repositorio de código

D

Documentación LaTeX

Tutor: Este anexo DEBE borrarse al terminar la memoria, solo contiene ejemplos de uso de LaTeX

D.1. Ejemplo de funcionamiento de LaTeX

Esto es una referencia bibliográfica [1].

Podemos definir una nota a pie de página.[1]

D.1.1. Tablas y figuras

Las tablas y figuras deben presentarse en el texto, referenciadas y numeradas. La descripción de una figura debe ir posicionada debajo de la misma. Las descripciones de tablas pueden aparecer encima o debajo de las mismas (pero de forma consistente en todo el documento).

En las tablas se recomienda evitar líneas verticales y usar pocas horizontales.

La figura D.1 se utiliza en la portada. LaTeXubica automáticamente las tablas y figuras. Para ello emplea reglas basadas en la experiencia de profesionales de la edición de textos. Podemos forzar su ubicación, pero en general es recomendable

[1] Esta es una nota a pie de página de ejemplo.

Tabla D.1: Título de la tabla.

	Subs.	Students	A	PE	WA	RE	CTE	IF	TLE	All
Ex. 1	104	44	1.27	0	0.55	0.23	0.20	0.11	0	2.36
Ex. 2	118	37	0.92	0	0.92	0.27	0.49	0.59	0	3.19
Ex. 3	100	28	1.21	0.39	1.18	0.54	0.14	0.07	0.04	3.57
Ex. 4	78	25	1.08	0.84	0.52	0.40	0.24	0.04	0	3.12
Ex. 5	116	31	1.48	0.10	0.77	0.32	0.42	0.19	0.45	3.74
Ex. 6	213	32	1.06	0.34	3.81	0.56	0.69	0.06	0.13	6.66
Ex. 7	116	34	1.35	0.38	0.38	0.68	0.62	0	0	3.41
Average	120.7	33	1.20	0.26	1.14	0.42	0.40	0.16	0.08	3.66

Figura D.1: Logo de la Universidad.

Tabla D.2: Tabla rotada. Factor groupings for the Mooshak questionnaire.

Factor	**Interpretation** / Items* (loadings)	Median	Mode
1	**Students' perception of Mooshak towards its helpfulness in learning**		
(21.17 %)	m10. Mooshak has forced me to implement programs more carefully (0,849)	4	4
$\alpha = 0.922$	m6. Mooshak has helped me improve as a programmer (0,819)	3	4
	m5. Mooshak has made me more aware of the need to write correct code (0,781)	3	3
	m1. Mooshak has forced me to program more responsibly (0,713)	3	3
	m15. The specifications regarding the exercises used with Mooshak are adequate (0,687)	3	3
	m18. Mooshak helps to measure my current programming skills (0,680)	2.5	3
2	**Disposition towards using Mooshak**		
(17.93 %)	m24. I would be willing to participate in a programming contest using Mooshak, with similar exercises to the ones	2	1
$\alpha = 0.897$	seen throughout the course (0,807)		
	m13. Using Moohak in the final exams is a good idea (0,748)	2	1
	m14. I would like to use Mooshak or a similar tool in the future (0,734)	3	1
	m17. Knowing Mooshak can motivate me to take part in a programming contest (0,655)	2	1
	m9. It would have been useful to use Mooshak from the first programming course (0,527)	2.5	1
	m16. Using Mooshak in the course has been interesting (0,522)	3	4
3	**Effect of Mooshak's feedback in the tool's usefulness**		
(14.84 %)	m12. Mooshak's feedback is adequate (0,832)	2	1
$\alpha = 0.836$	m3. Using Mooshak has increased my workload considerably (0,693)	4	4
	m7. If Mooshak does not accept my code I feel motivated to find and fix the errors (0,691)	2	3
	m8. In general, using Mooshak has been a good idea (0,666)	3	4
4	**Mooshak's effect on persistence**		
(11.20 %)	m23. When Mooshak does not accept my code I get discouraged and I abandon the exercise (0,848)	3	3
$\alpha = 0.705$	m22. Mooshak has been a waste of time (0,597)	2	2
	m25. Once a program has passed Mooshak's tests, I rewrite it in order to enhance it (0,559)	2	2
5	**Students' perception of Mooshak's features**		
(10.87 %)	m20. Even if it is not related to the grade, I feel satisfied if I am one of the first students to complete an exercise (0,729)	2	2
$\alpha = 0.742$	m19. I value the fact that a tool like Mooshak returns feedback in real time about the correction of my programs (0,650)	3.5	4

*Measured on a 5-point Likert scale (1: strongly disagree; 2: disagree; 3: neutral; 4: agree; 5: strongly agree).

numeric literals	integers	in decimal	`8743`
		in octal	`0o7464`
			`0O103`
		in hexadecimal	`0x5A0FF`
			`0xE0F2`
	fractionals	in decimal	`140.58`
			`8.04e7`
			`0.347E+12`
			`5.47E-12`
			`47e22`
char literals			`'H'`
			`'\n'`
			`'\x65'`
string literals			`"bom dia"`
			`"ouro preto\nmg"`

Tabla D.3: Tabla con "multicolumnas" y "multifilas".

usar la ubicación sugerida por el sistema LaTeX. Usad gráficos vectoriales siempre que podáis.

D.2. Segunda sección

Normalmente no tendremos que insertar saltos de página, salvo para forzar que los capítulos empiecen en páginas impares, con

```
\blankpage
```

En cualquier caso, podemos introducir un salto de página con el comando

```
\newpage
```

.

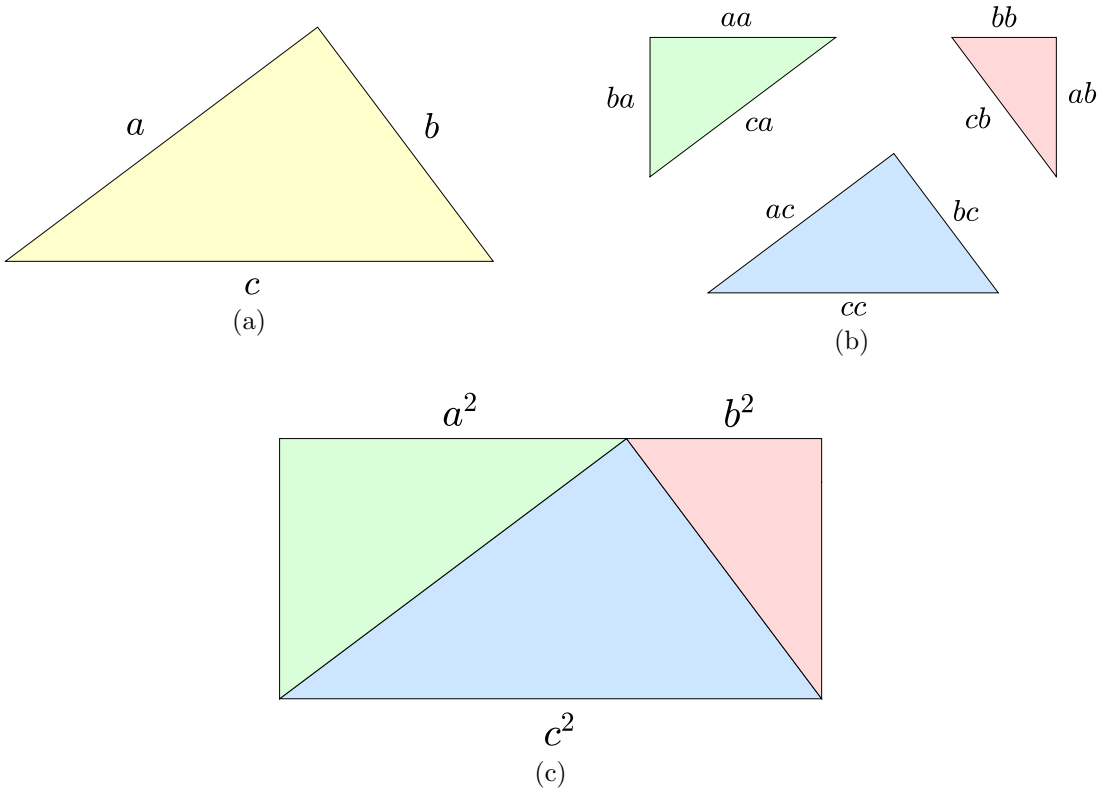


Figura D.2: Ejemplo con varias figuras. Demostración visual del teorema de Pitágoras. En (a) tenemos un triángulo rectángulo con hipotenusa c y catetos a y b. En (b) se muestra tres copias escaladas del mismo triángulo. El verde se ha escalado por a, el rojo/rosa por b, y el azul por c. En (c) se juntan los triángulos de (b) para formar un rectángulo cuya base es c^2, pero también $a^2 + b^2$. Por tanto, $a^2 + b^2 = c^2$.

Algoritmo 1 *Additional Louvain* **input**=$(A,\ \mathcal{M})$ **output**=P

$\forall i \in V$, let i be an isolated community
$o = permutation(V)$
for $k \in o$ **do**
  search in A all the neighbours of k, j
  $\forall j$, calculate $\Delta Q_k(j)$ in matrix $\mathcal{M}$
  $j^* = \{\ j \mid \Delta Q_k(j^*) = \text{máx}_j\{Q_k(j)\}\ \}$
  if $\Delta Q_k(j^*) > 0$ **then**
    Move node k to j^* 's community
  else
    k remains in its community
  end if
end for

D.2.1. Código

```python
def sum_list_limits_1(a, lower, upper):
    if lower > upper:
        return 0
    else:
        return a[upper] + sum_list_limits_1(a, lower, upper - 1)
```

Código D.1: Titulo del algoritmo/código.

El código D.1 es un ejemplo en Python.

```java
@GetMapping("/{id}")
public ResponseEntity<Book> getBook(@PathVariable long id) {
  // Comment to explain the code
  Optional<Book> op = service.findById(id);
  log.info("Book with id {} found", id);
  if (op.isPresent()) {
    Book book = op.get();
    return new ResponseEntity<>(book, HttpStatus.OK);
  } else {
    return new ResponseEntity<>(HttpStatus.NOT_FOUND);
  }
}
```

Código D.2: Endpoint REST para recuperar un libro por su id

El código D.2 es un ejemplo en Java.

En el algoritmo 1 aparece un ejemplo en pseudocódigo.

D.2.2. Comentarios

Esto es un ejemplo de texto

Tutor: Esto es un comentario del tutor

Alumno: Esto es un comentario del alumno