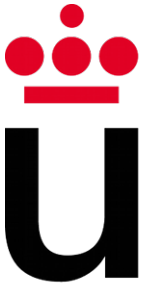

Universidad
Rey Juan Carlos

TITULACIÓN EN MAYÚSCULAS

Curso Académico 20XX/20XX

Trabajo Fin de Grado/Máster

UN TÍTULO DE PROYECTO LARGO EN DOS LÍNEAS

Autor/a : Nombre del Alumno/a
Tutor/a : Dr. Nombre del Profesor/a

Trabajo Fin de Grado/Máster

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

Autor/a : Nombre del Alumno/a
Tutor/a : Dr. Nombre del profesor/a

La defensa del presente Proyecto Fin de Grado/Máster se realizó el día 3 de de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Móstoles/Fuenlabrada, a de de 20XX

Aquí normalmente
se inserta una dedicatoria corta

Agradecimientos

Aquí vienen los agradecimientos...

Hay más espacio para explayarse y explicar a quién agradeces su apoyo o ayuda para haber acabado el proyecto: familia, pareja, amigos, compañeros de clase...

También hay quien, en algunos casos, hasta agradecer a su tutor o tutores del proyecto la ayuda prestada...

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Índice de fragmentos de código

Capítulo 1

Introducción

La robótica es un campo en constante evolución con aplicaciones en múltiples sectores como la industria, la medicina, la agricultura y el transporte. Su enseñanza, sin embargo, enfrenta grandes desafíos debido a la complejidad de los entornos de programación, la necesidad de hardware especializado y la dificultad de acceso a laboratorios físicos. Los estudiantes que desean aprender sobre robótica deben instalar y configurar diversos programas, como ROS2 (Robot Operating System 2), simuladores como Gazebo, y librerías adicionales que pueden ser complejas de integrar correctamente. Esto representa una barrera de entrada significativa, especialmente para aquellos sin experiencia previa en entornos de desarrollo avanzado.

Además, la enseñanza tradicional de la robótica requiere acceso a robots físicos, lo que no siempre es viable debido a los altos costos de los equipos y las limitaciones de espacio y recursos en las instituciones educativas. Como resultado, los estudiantes tienen menos oportunidades para experimentar con la programación y prueba de robots en escenarios reales o simulados.

Para abordar estos problemas, han surgido diversas plataformas de aprendizaje en línea que buscan facilitar el acceso a herramientas de programación robótica. No obstante, muchas de ellas presentan limitaciones en términos de flexibilidad, compatibilidad y accesibilidad. Es en este contexto donde Unibotics se posiciona como una solución innovadora y accesible para la enseñanza de la robótica en entornos educativos.

Unibotics es una plataforma web diseñada con el objetivo de simplificar y mejorar el aprendizaje práctico de la robótica. Proporciona a los estudiantes un entorno de trabajo donde pueden acceder a ejercicios interactivos y simulaciones sin la necesidad de instalar ni configurar software complejo en sus equipos locales. A través de esta plataforma, los usuarios pueden programar robots en un entorno seguro y controlado, con acceso a desafíos diseñados para desarrollar habilidades en navegación, percepción y control au-

tónomo. Además, Unibotics permite a los estudiantes trabajar con una variedad de robots simulados, desde vehículos autónomos hasta drones y robots de servicio, lo que amplía significativamente las posibilidades de aprendizaje.

Uno de los aspectos clave de Unibotics es su enfoque en la accesibilidad y la facilidad de uso. Al eliminar la necesidad de instalaciones locales y proporcionar un entorno de programación basado en la web, la plataforma permite que cualquier estudiante con acceso a un navegador pueda comenzar a aprender robótica sin preocuparse por problemas de compatibilidad o configuración. Esto la convierte en una herramienta ideal tanto para principiantes como para usuarios avanzados que desean perfeccionar sus habilidades en robótica.

A pesar de las ventajas que ofrece Unibotics, aún existen áreas de mejora en la gestión y visualización de los datos generados en la plataforma. Actualmente, la información sobre los ejercicios, el rendimiento de los estudiantes y las métricas de uso de la plataforma están almacenadas en PostgreSQL, pero no se presentan de manera visual ni se explotan para obtener información valiosa sobre el aprendizaje de los estudiantes.

Este proyecto tiene como objetivo principal el desarrollo de una API que permita extraer, procesar y analizar los datos generados en Unibotics, proporcionando una interfaz para su consumo y análisis. Además, se diseñarán dashboards interactivos que faciliten la visualización de la información, permitiendo a profesores y administradores obtener insights sobre el uso de la plataforma y el progreso de los estudiantes.

Con la implementación de esta API y dashboards, se pretende mejorar la experiencia de los usuarios al proporcionarles herramientas avanzadas para el análisis de datos, optimizando así el proceso de enseñanza y aprendizaje de la robótica.

1.1 Objetivos

Esto es una sección, que es una estructura menor que un capítulo.

Por cierto, a veces me comentáis que no os compila por las tildes. Eso es un problema de codificación. Al guardar el archivo, guardad la codificación de “ISO-Latin-1” a “UTF-8” (o viceversa) y funcionará.

1.1.1 Objetivo general

Aquí vendría el objetivo general en una frase: Mi Trabajo Fin de Grado/Master consiste en crear de una herramienta de análisis de los comentarios jocosos en repositorios de software libre alojados en la plataforma GitHub.

Recuerda que los objetivos siempre vienen en infinitivo.

1.1.2 Objetivos específicos

Los objetivos específicos se pueden entender como las tareas en las que se ha desglosado el objetivo general. Y, sí, también vienen en infinitivo.

Lo mejor suele ser utilizar una lista no numerada, como sigue:

- Un objetivo específico.
- Otro objetivo específico.
- Tercer objetivo específico.
- ...

1.2 Planificacion

1.3 Estructura del documento

Es conveniente que incluyas una descripción de lo que te ha llevado realizar el trabajo. Hay gente que añade un diagrama de GANTT. Lo importante es que quede claro cuánto tiempo has consumido en realizar el TFG/TFM (tiempo natural, p.ej., 6 meses) y a qué nivel de esfuerzo (p.ej., principalmente los fines de semana).

1.4 Estructura de la memoria

Por último, en esta sección se introduce a alto nivel la organización del resto del documento y qué contenidos se van a encontrar en cada capítulo.

- En el primer capítulo se hace una breve introducción al proyecto, se describen los objetivos del mismo y se refleja la planificación temporal.
- En el siguiente capítulo se describen las tecnologías utilizadas en el desarrollo de este TFM/TFG (Capítulo 2).
- En el capítulo 3 Se describe el proceso de desarrollo de la herramienta ...
- En el capítulo 4 Se presentan las principales pruebas realizadas para validación de la plataforma/herramienta...(o resultados de los experimentos efectuados).
- Por último, se presentan las conclusiones del proyecto así como los trabajos futuros que podrían derivarse de éste (Capítulo **??**).

Capítulo 2

Tecnologías

Descripción de las tecnologías que utilizas en tu trabajo. Con dos o tres párrafos por cada tecnología, vale. Se supone que aquí viene todo lo que no has hecho tú.

Puedes citar libros, como el de Bonabeau et al., sobre procesos estigmérgicos [**bonabeau:_swarm**]. Me encantan los procesos estigmérgicos. Deberías leer más sobre ellos. Pero quizás no ahora, que tenemos que terminar la memoria para sacarnos por fin el título. Nota que el~añade un espacio en blanco, pero no deja que exista un salto de línea. Imprescindible ponerlo para las citas.

Citar es importantísimo en textos científico-técnicos. Porque no partimos de cero. Es más, partir de cero es de tontos; lo suyo es aprovecharse de lo ya existente para construir encima y hacer cosas más sofisticadas. ¿Dónde puedo encontrar textos científicos que referenciar? Un buen sitio es Google Scholar[1]. Por ejemplo, si buscas por "stigmergy libre software" para encontrar trabajo sobre software libre y el concepto de *estigmergia* (¿te he comentado que me gusta el concepto de estigmergia ya?), encontrarás un artículo que escribí hace tiempo cuyo título es "Self-organized development in libre software: a model based on the stigmergy concept". Si pulsas sobre las comillas dobles (entre la estrella y el "citado por ...", justo debajo del extracto del resumen del artículo, te saldrá una ventana emergente con cómo citar. Abajo a la derecha, aparece un enlace BibTeX. Púlsalo y encontrarás la referencia en formato BibTeX, tal que así:

[1] http://scholar.google.com

Uno	2	3
Cuatro	5	6
Siete	8	9

Tabla 2.1: Ejemplo de tabla. Aquí viene una pequeña descripción (el *caption*) del contenido de la tabla. Si la tabla no es autoexplicativa, siempre viene bien aclararla aquí.

```
n+nc@inproceedingspn+nlrobles2005selfp,
  n+natitlep=l+sSelforganized development in libre software:
l+s         a model based on the stigmergy conceptp,
  n+naauthorp=l+sRobles, Gregorio and Merelo, Juan Julian
l+s          and GonzalezBarahona, Jesus M.p,
  n+nabooktitlep=l+sProSim05p,
  n+nayearp=l+s2005
p
```

Copia el texto en BibTeX y pégalo en el fichero `memoria.bib`, que es donde están las referencias bibliográficas. Para incluir la referencia en el texto de la memoria, deberás citarlo, como hemos hecho antes con [**bonabeau:_swarm**], lo que pasa es que en vez de el identificador de la cita anterior (bonabeau:_swarm), tendrás que poner el nuevo (robles2005self). Compila el fichero `memoria.tex` (`pdflatex memoria.tex`), añade la bibliografía (`bibtex memoria.aux`) y vuelve a compilar `memoria.tex` (`pdflatex memoria.tex`)...y *voilà* ¡tenemos una nueva cita [**robles2005self**]!

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página del GSyC[2].

2.1 Sección 1

Hemos hablado de cómo incluir figuras, pero no se ha descrito cómo incluir tablas. A continuación se presenta un ejemplo de tabla, la Tabla 2.1 (fíjate en cómo se introduce una referencia a la tabla).

[2] http://gsyc.es

2.2 Entorno de desarrollo: PyCharm

PyCharm es un Integrated Development Enviroment (Entorno de Desarrollo Integrado) (IDE) dedicado concretamente a la programación en Python y desarrollado por la compañía checa JetBrains.

Proporciona análisis de código, un depurador gráfico, una consola de Python integrada, control de versiones y, además, soporta desarrollo web con Django. Todas estas características lo convierten en un entorno completo e intuitivo, idóneo para el desarrollo de proyectos académicos como el que nos ocupa.

2.3 Redacción de la memoria: LaTeX/Overleaf

LaTeX es un sistema de composición tipográfica de alta calidad que incluye características especialmente diseñadas para la producción de documentación técnica y científica. Estas características, entre las que se encuentran la posibilidad de incluir expresiones matemáticas, fragmentos de código, tablas y referencias, junto con el hecho de que se distribuya como software libre, han hecho que LaTeX se convierta en el estándar de facto para la redacción y publicación de artículos académicos, tesis y todo tipo de documentos científico-técnicos.

Por su parte, Overleaf es un editor LaTeX colaborativo basado en la nube. Lanzado originalmente en 2012, fue creado por dos matemáticos que se inspiraron en su propia experiencia en el ámbito académico para crear una solución satisfactoria para la escritura científica colaborativa.

Además de por su perfil colaborativo, Overleaf destaca porque, pese a que en LaTeX el escritor utiliza texto plano en lugar de texto formateado (como ocurre en otros procesadores de texto como Microsoft Word, LibreOffice Writer y Apple Pages), éste puede visualizar en todo momento y paralelamente el texto formateado que resulta de la escritura del código fuente.

Capítulo 3

Arquitectura

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

3.1 Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a "vista de pájaro").

Por ejemplo, puedes verlo en la Figura 3.1. LaTeX pone las figuras donde mejor cuadran. Y eso quiere decir que quizás no lo haga donde lo hemos puesto...Eso no es malo. A veces queda un poco raro, pero es la filosofía de LaTeX: tú al contenido, que yo me encargo de la maquetación.

Recuerda que toda figura que añadas a tu memoria debe ser explicada. Sí, aunque te parezca evidente lo que se ve en la Figura 3.1, la figura en sí solamente es un apoyo a tu texto. Así que explica lo que se ve en la Figura, haciendo referencia a la misma tal y como ves aquí. Por ejemplo: En la Figura 3.1 se puede ver que la estructura del *parser* básico, que consta de seis componentes diferentes: los datos se obtienen de la red, y según el tipo de dato, se pasará a un *parser* específico y bla, bla, bla...

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

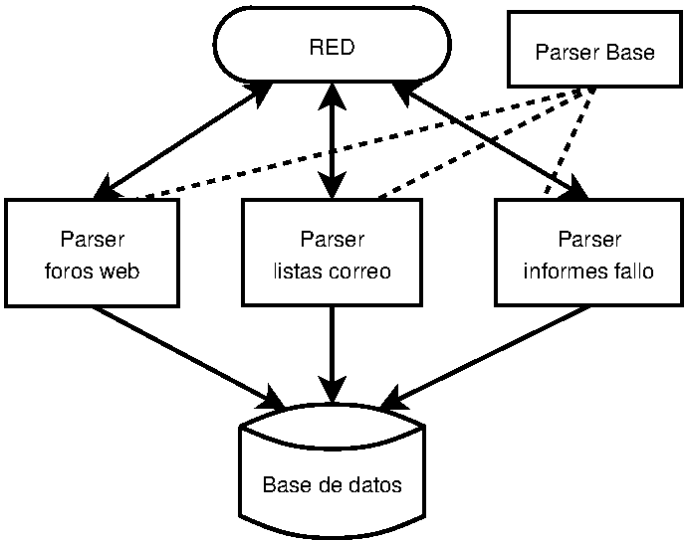

Figura 3.1: Estructura del parser básico.

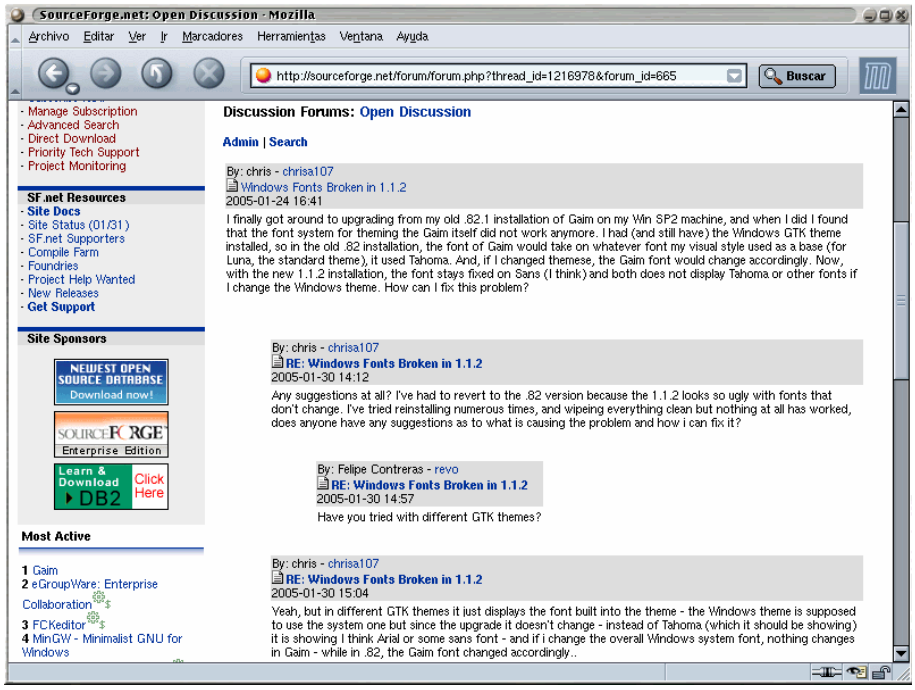

Figura 3.2: Página con enlaces a hilos

Capítulo 4

Experimentos y validación

Atención: Este capítulo se introdujo como requisito en 2019.

Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

4.1 Incorporación de código en la memoria

Es bastante habitual que se reproduzcan fragmentos de código en la memoria de un TFG/TFM. Esto permite explicar detalladamente partes del desarrollo que se ha realizado que se consideren de especial interés. No obstante, tampoco es conveniente pasarse e incluir demasiado código en la memoria, puesto que se puede alargar mucho el documento. Un recurso muy habitual es subir todo el código a un repositorio de un servicio de control de versiones como GitHub o GitLab, y luego incluir en la memoria la URL que enlace a dicho repositorio.

Para incluir fragmentos de código en un documento LaTeXse pueden combinar varias herramientas:

- El entorno `kbeginn+nblistingn+nb[]...kendn+nblistingn+nb` permite crear un marco en el que situar el fragmento de código (parecido al generado cuando insertamos una tabla o una figura). Podemos insertar también una descripción (*caption*) y una etiqueta para referenciarlo luego en el texto.
- Dentro de este entorno, se puede utilizar el paquete `minted` [1], que utiliza el paquete

[1] https://es.overleaf.com/learn/latex/Code_Highlighting_with_minted