UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE MÁSTER

GRADO EN TECNOLOGÍAS PARA LA SOCIEDAD DE LA INFORMACIÓN

El lenguaje de programación Z

Autor: Arturito Belesquele Tiruriru

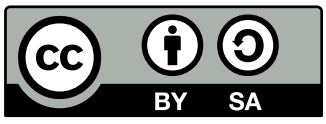
Directora: Michaela Faraday de Jesús

Madrid, junio 2021

Arturito Belesquele Tiruriru

El lenguaje de programación Z
Proyecto Fin de Máster, jueves 17 junio, 2021
Directora: Michaela Faraday de Jesús

E.T.S. de Ingeniería de Sistemas Informáticos
Campus Sur UPM, Carretera de Valencia (A-3), km. 7
28031, Madrid, España



Yo, **Arturito Belesquele Tiruriru**, estudiante de la titulación **Grado en Tecnologías para la Sociedad de la Información** de la de **E.T.S. de Ingeniería de Sistemas Informáticos** de la **Universidad Politécnica de Madrid**, como **autor** del Proyecto Fin de Máster titulado:

El lenguaje de programación Z

DECLARO QUE

Este proyecto es una obra original y que todas las fuentes utilizadas para su realización han sido debidamente citadas en el mismo. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la memoria presentada de conformidad con el ordenamiento jurídico vigente.

Madrid, a jueves 17 junio, 2021

Fdo.: Arturito Belesquele Tiruriru
Autor del Proyecto Fin de Máster

Resumen

El resumen de un proyecto de fin de grado, máster o de una tesis condensa en tres o cuatro párrafos el contenido de la memoria. Se debe dar por sentado que el lector podrá tener una idea clara de lo que trata, y suele ser la primera barrera donde decide si continúa leyendo o no el texto.

Condensado no quiere decir incompleto. Debe contener la información más destacable. Lo ideal es que ocupe entre media y una cara de un folio A4. Comenzará por el propósito y principales objetivos de la memoria. Luego hablaremos sobre los aspectos más destacables de la metodología empleada, seguido de los resultados obtenidos. Por último se presentarán las conclusiones de forma condensada.

Debe tener un estilo claro y conciso, sin ambigüedades de ningún tipo. Además, al ser un resumen de todo el contenido, ni que decir tiene que deberá ser lo último que elaboraremos, y deberá mantener una absoluta fidelidad con el contenido de la memoria.

Palabras clave: Cuatro o cinco; Expresiones clave; Que resuman; Nuestro proyecto o; Investigación

Abstract

The summary of a bachelor's or master's degree project or thesis condenses the content of the report into three or four paragraphs. It should be assumed that the reader will have a clear idea of what it is about, and it is usually the first barrier to deciding whether or not to continue reading the text.

Condensed does not mean incomplete. It should contain the most important information. Ideally, it should take up half to one side of an A4 sheet of paper. Start with the purpose and main objectives of the report. Then we will talk about the most important aspects of the methodology used, followed by the results obtained. Finally, the conclusions will be presented in a condensed form.

It should have a clear and concise style, without ambiguities of any kind. Furthermore, as it is a summary of all the content, it goes without saying that it should be the last thing we will produce, and it should be absolutely faithful to the content of the report.

Keywords: Four or five; Key Expressions; Summarising; Our Project or; Research

Agradecimientos

Pues aquí agradecimientos y esas cosas

Índice general

Índice de figuras

Índice de cuadros

Índice de listados

1. Introducción

La introducción a un Proyecto Fin de Grado (PFG), Proyecto Fin de Máster (PFM) o Tesis Doctoral (TD) el el punto de entrada a todo el trabajo realizado y es considerada la más importante tras el abstract, que es quien resume el trabajo entero. En ella habría que dejar claro qué es el trabajo que se ha realizado, por qué es importante y qué es lo que aporta como resultados.

La introducción generará expectativas, y por tanto hay que intentar venderla bien. Un gancho típico en los trabajos suele ser el de aportar un dato relevante o controvertido para discutir sobre él o plantear una pregunta relevante para el contexto en el que se está trabajando.

Dentro del capítulo, tras introducir el trabajo realizado de forma genérica, se suelen incluir las siguientes secciones para establecer bien el alcance y las limitaciones del mismo: motivación, objetivos, suposiciones/limitaciones y, a veces, estructura de la memoria.

Ni que decir tiene que esta estructura planteada, tanto del capítulo como de la memoria en si es únicamente un ejemplo o propuesta. Cada proyecto es único y a veces es más cómodo escribirlo de otro modo.

1.1 Motivación

Esta sección[1] describe qué factores han hecho al estudiante decantarse por trabajar en éste y no en otro tema.

Lo más indicado en este caso es apoyarse en datos de fuentes contrastables en lugar de en expresiones tipo "ampliar mis conocimientos". Información extraída de revistas especializadas (i.e. científicas), periódicos, organismos de estandarización, el Instituto Nacional de Estadística (INE), etcétera se suele presuponer contrastada y fiable, y por tanto una buena base sobre la que partir.

[1] A veces esta sección se denomina *justificación*, aunque *motivación* está más extendido.

1.2 Objetivos

El objetivo de un PFG, PFM y TD es una de las piezas clave a plantear, y a su vez una de las más complicadas. Se considera **la finalidad** del proyecto en cuestión a realizar y suele encajar dentro de una de las siguientes categorías:

- **Contraste** o validación de una hipótesis. Este es típico de TDs, aunque algunos PFMs y (muy raramente) PFGs pueden caer dentro de esta categoría.
- **Desarrollo** o diseño de algo (e.g. Software, hardware, sistema, edificio). Suele ser el más común en la rama de la ingeniería, tanto PFMs como PFGs.
- **Estudio** de un tema que deduce o descubre nuevo conocimiento. Éste suele ser más común en las ramas de las ciencias puras y humanidades, tanto PFMs como PFGs.

Decimos que es una pieza clave porque sirve como primer indicador de la consecución del proyecto. Si nos planteamos un objetivo, en las conclusiones podemos indicar si se ha cumplido o no el objetivo planteado. Por eso es necesario que el objetivo esté bien definido, porque si se acepta como objetivo válido en un proyecto, y éste se concluye como cumplido, el proyecto habrá sido ejecutado correctamente.

Ahora bien, ¿cómo determinamos que el objetivo se ha cumplido? pues intentando definirlo para que se pueda cumplir, es decir, intentando que sea:

- **Acotado en el tiempo**, así es más fácil establecer un marco temporal para su realización y programar temporalmente las partes de las que se compone.
- **Medible**, para saber cómo de lejos estamos de llegar a un resultado aceptable.
- **Específico**, de manera que esté bien acotado y sea difícil embarcarse en tareas que no nos acerquen a su consecución.
- **Alcanzable**, porque si no lo es, por mucha intención y esfuerzo que le pongamos no se va a terminar.
- **Relevante**, porque si, en un PFG para Ingeniería del Software, desarrollamos un producto mecánico para sexar pollos, pues por muy importante que sea, poco tiene que ver con lo que se ha estudiado durante todos estos años.

Y sí, para acordarnos de cuáles son estas características podemos usar el acrónimo *AMEAR*.

1.3 Estructura de la memoria

Esta memoria está estructurada de la siguiente forma: CUANDO TERMINE LA ESTRUCTURA, PONERLA AQUÍ

2. Configuración de la memoria

2.1 Ficheros y directorios

La estructura de ficheros es la siguiente:

`./appendices/` Los fuentes de los capítulos de apéndices.

`./chapters/` Los fuentes de los capítulos que forman parte del cuerpo de la memoria.

`./figures/` Las figuras (imágenes, diagramas) que se usarán en la memoria.

`./fonts/` Las fuentes que se usan en la memoria. Lo mismo que antes, si se piensa en ampliar la plantilla, es otro de los sitios donde tocar.

`./prefrontmatter/` Los fuentes de todo aquello que se incluye antes del cuerpo de la memoria, como por ejemplo los glosarios.

`./logos` Los logos que se usan en la configuración de la memoria. Es de esperar que no se toquen, aunque si se está trabajando en ampliar la plantilla, este es uno de los sitios donde tocar.

`./sources` Ficheros con fuentes que se incluyen dentro de listados en el documento.

`./firma.png` El fichero con la imagen de la firma del estudiante que ha desarrollado la memoria. Se usa en la hoja de declaración de autoría.

`./references.bib` Los fuentes en bibtex de la bibliografía referenciada en la memoria.

`./report.tex` El fichero con el código fuente principal, el cual será necesario tocar para incluir el resto de fuentes.

`./upm-report.cls` El fichero con la descripción de la memoria, con todas las opciones de configuración y demás.

2.2 ¿Cómo empiezo a escribir la memoria?

Con cuidado. Esto quiere decir que habría que empezar por el principio, es decir, con el fichero `report.tex`. La primera línea del fichero tiene la siguiente forma:

Listado 2.1: Primeras líneas del fichero `report.tex`

```
\documentclass[%
    school=etsisi,%
    type=pfm,%
    degree=61TI,%
    authorsex=m,%
    directorsex=f,%
]{upm-report}
```

En este punto es donde se configura gran parte de la plantilla. Los parámetros y sus opciones son las siguientes:

`school` La escuela a la que pertenece el estudiante. La idea de la plantilla es que se use a lo largo de todas las escuelas de la UPM, y que cada una de ellas tenga su propia configuración. La escuela determinará, entre otras cosas, direcciones y colores principales. Las opciones se describen en el apéndice A.

`type` El tipo de memoria. Modifica algunos textos, incluida la portada. Puede tomar los valores `pfg` (Proyecto Fin de Grado), `pfm` (Proyecto Fin de Máster) o `phd` (Tesis Doctoral)

`degree` El grado al que aspira el estudiante. De momento sólo están definidos los grados que se imparten en la ETSISI.

`authorsex` Puede ser `m` (masculino) o `f` (femenino), y sirve para modificar algunos textos relacionados con el sexo del estudiante. Si no se especifica, se usará el genérico masculino.

`directorsex` Similar al parámetro `authorsex`, pero para el director/tutor del proyecto.

Tras esta configuración, se incluye el fichero de referencias bibliográficas:

Listado 2.2: Inclusión del fichero de referencias bibliográficas `references.bib`

```
\addbibresource{references.bib}
```

El tema de las referencias bibliográficas se explica en el capítulo 5, sección ??. En principio no habría que tocar nada, pero si las referencias se tienen en otro fichero, bastaría con cambiar el nombre al de dicho fichero.

Los tres siguientes comandos indican el nombre del autor del proyecto, su título y el tutor/director. La verdad es que no tiene mucho más misterio.

Listado 2.3: Configurando autor, título del proyecto y director

```
\author{Arturito Belesquele Tiruriru}
\title{El lenguaje de programación Z}
\director{Michaela Faraday de Jesús}
```

Tras ello, empieza el cuerpo del proyecto propiamente dicho. Los primeros tres comandos `include` incluyen el contenido de tres ficheros, el referente al *glosario*, al *abstract* y a los *agradecimientos*, ficheros que se encuentran bajo el directorio `prefrontmatter`

Listado 2.4: Insertando los ficheros de agradecimientos, abstract y glosario

```
    Condensado no quiere decir incompleto. Debe contener la información
       más destacable. Lo ideal es que ocupe entre media y una cara de
       un folio A4. Comenzará por el propósito y principales objetivos
       de la memoria. Luego hablaremos sobre los aspectos más
       destacables de la metodología empleada, seguido de los resultados
        obtenidos. Por último se presentarán las conclusiones de forma
       condensada.
```

Sobre el glosario se habla con algo más de detalle en el capítulo 5, sección 5.4. El fichero de `abstract.tex` incluye, en realidad, dos capítulos, uno para el resumen en español y otro para

el resumen en inglés (y sí, hay que hacer los dos). Por último, el fichero `agradecimientos.tex` sirve para añadir los agradecimientos, que esto siempre gusta a las abuelas.

Tras ello, pasamos a la parte frontal de la memoria: los índices, glosario y acrónimos. Se autogeneran a partir del contenido de la memoria, pero hay que declararlos. Son los siguientes:

Listado 2.5: Generando índices y listados

```
\abstract{english}{
    The summary of a bachelor's or master's degree project or thesis
       condenses the content of the report into three or four paragraphs
       . It should be assumed that the reader will have a clear idea of
       what it is about, and it is usually the first barrier to deciding
        whether or not to continue reading the text.

    Condensed does not mean incomplete. It should contain the most
       important information. Ideally, it should take up half to one
       side of an A4 sheet of paper. Start with the purpose and main
       objectives of the report. Then we will talk about the most
       important aspects of the methodology used, followed by the
       results obtained. Finally, the conclusions will be presented in a
        condensed form.
```

Realmente indispensable no hay ninguno, pero por lo menos estaría bien mantener la tabla de contenidos (comando `tableofcontents`). Los demás se refieren a la lista de figuras, de cuadros[1], de listados de fuentes y glosario más acrónimos respectivamente.

Tras la parte frontal se pasa al cuerpo donde, normalmente, tendremos un fichero por capítulo, así tenemos la memoria bien organizada. Estos capítulos se incluyen con el comando `include` y esta plantilla tiene unos cuantos para que se vea su uso.

El último elemento que se incluye es la bibliografía (comando `printbibliography`) tras la cual vienen los apéndices. Éstos se incluirán igual que el resto de ficheros, con el comando `include`,

[1] Lo llamamos cuadros y no tablas porque un cuadro es un concepto más genérico que una tabla, y tabla es un *false friente* del inglés *table*.

pero al ir declarados después del comando `appendix` su numeración será diferente.

Y ya está terminada la memoria. Resumiento, hayq ue configurar la plantilla, poner el autor, título y director del proyecto e incluir los capítulos y apéndices que queramos.

2.3 ¿Cómo estructurar la memoria?

La respuesta rápida es "como buenamente quieras/puedas". En realidad la estructura de la memoria va a depender del tipo de trabajo desarrollado.

Aún así es cierto que, con carácter general, los trabajos suelen seguir ciertas estructuras. En esta sección comentamos algunas de éstas en función del tipo de memoria que se esté desarrollando.

Un PFG es un trabajo cuyo propósito es demostrar que se han llegado a adquirir las competencias asociadas con la titulación cursada. Con esto queremos decir que, a diferencia de otros tipos de trabajo académico, en éste no es necesario realizar aportaciones originales al estado de la cuestión.

Una estructura típica es al siguiente:

1. Contenido inicial
2. Estado de la cuestión
3. Metodología
4. Resultados y Discusión
5. Conclusiones
6. Referencias bibliográficas
7. Glosario
8. Apéndices
9. Índice

Un PFM, a diferencia de un PFG trata de profundizar más en un campo concreto de una disciplina, por lo que tiene a ser más extenso y mucho más específico.

En términos generales, la estructura es similar. Sin embargo es de esperar que el nivel de exigencia sea mayor, ya que el estudiante que lo realiza debe demostrar que es un titulado superior. Esto se nota más en la fase de documentación, ya que al tratar de profundizar en un tema más específico, el trabajo de contextualizar y argumentar es más tedioso.

Se pueden identificar dos tipos de proyectos diferenets, aquellos que podríamos catalogar de *profesionales*, con enfoque a la innovación o mejora en un área profesional comcreta, y aquellos *de investigación*, más enfocados a la búsqueda de nuevo conocimiento en el área, y que suelen ser el comienzo de la carrera investigadora.

Por último, una TD es un trabajo eminentemente de investigación. Su profundidad y complejidad es la más alta, su extensión es mucho mayor, con claro énfasis en el análisis del estado de la cuestión y la discusión de resultados. Por lo demás, su estructura es similar a la presentada anteriormente.

3. Componentes de la plantilla

Aquí tenemos que hablar de los componentes que tenemos disponibles y cómo se utilizan: labels para referencias cruzadas, bibliografía, tablas, imágenes, columnas, cuadros, enlaces de hipertexto, código fuente, algoritmos, fórmulas etcétera.

4. Figuras, tablas, ecuaciones y fuentes

En este capítulo vamos a hablar de los elementos denominados "flotantes" o *floats*. Estos elementos son bloques de contenido que "flotan" por la página hasta que LaTeX los coloca donde considera a través de ciertos algoritmos.

Lo general es declarar el elemento flotante inmediatamente **después** del párrafo donde se ha referenciado, y después dejar que LaTeX elija el mejor sitio. Y si después de haber escrito todo el documento hay algo que no cuadre, pues ahí modificarlo.

Una pregunta típica es la siguiente: "¿oye, y si no hago referencia a un elemento flotante, dónde lo pongo?" La respuesta es sencilla: no se pone. Generalmente, si hay algo que no merece ser nombrado, tampoco tiene mucho sentido que aparezca. Remarco eso de "generalmente"; lo mismo existe algún caso donde sí tiene sentido, pero así, para empezar, es muy raro.

Aunque más adelante veremos los diferentes tipos de *floats*, en caso de que queramos modificar su comportamiento todos tienen los especificadores de posición indicados en el cuadro 4.1:

Cuadro 4.1: Opciones para los elementos flotantes de LaTeX

Especificador	**Acción**
H	Colocar exactamente en el sitio indicado
h	Colocar aproximadamente en el sitio indicado
t	Colocar al comienzo de la página
b	Colocar al final de la página
p	Colocar en una página exclusiva para elementos "flotantes"
!	Forzar las indicaciones y obviar los mecanismos internos de LaTeX

4.1 Figuras

El código siguiente (listado 4.1) renderizará una imagen.

Listado 4.1: Inserción de una figura

```
\begin{figure}[H]
    \centering
    \caption{\label{fig:img-vault-boy} Vault Boy approves that}
    \includegraphics[width=0.25\textwidth]{figures/vault-boy.png}
\end{figure}
```

La sintaxis es bastante autoexplicativa. El entorno `figure` es el que delimita el contenido de la figura. El comando `centering` determina que se tiene que centrar y, aunque hay varias formas de hacerlo, creemos que esta es la más sencilla. Tras éstos, el comando `caption` determina el pie de imagen, el cual además incluye una etiqueta (comando `label`) que sirve para referenciar. Por último, se incluye la imagen con el comando `includegraphics`.

En definitiva, la imagen (figura 4.1) se mostrará con un ancho igual a la mitad del ancho que ocupa el texto, centrada, con un pie de foto y una etiqueta para referenciar.

Figura 4.1: Vault Boy approves that

El comando `includegraphics` puede importar los formatos típicos de imagen, como jpeg, png o pdf. También admite una serie de opciones como rotación, alto, ancho (éste le hemos especificado con `width`), etcétera. ¡Ojo! Siempre que se pueda, hay que intentar insertar imágenes vectoriales. De esta manera, se mantiene la calidad de la imagen. Si no, puede ocurrir que se pixele y no quede nada bien.

Subfiguras

¿Y qué pasa cuando queremos incluir múltiples imágenes dentro de una figura? Bueno, pues aquí hay que usar el entorno `subfigure`. En el listado 4.2 vemos un ejemplo de cómo se manejan.

Listado 4.2: Inserción de varias subfiguras

```
\begin{figure}[H]
  \centering
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\linewidth]{figures/vault-boy.png}
    \caption{\label{fig:subfigure-1}Vault Boy 1}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 2}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 3}
  \end{subfigure}
  \caption{\label{fig:subfigures}Todos los Vault Boy}
\end{figure}
```

En realidad cada subfigura se trata como una figura normal, pero en relación con el *float* contenedor. Cuando a una subfigura se le especifica un ancho, se le está diciendo al compilador de qué ancho es esa subfigura en concreto (en nuestro caso 0.3 veces el ancho de la línea). Sin embargo, a la imagen se le da un ancho total de `linewidth`, porque al estar dentro de su espacio de subfigura, el ancho ha cambiado. El resultado es el que se observa en la figura 4.2. Por cierto, también podemos referenciar a los pies de las subfiguras (e.g. Así: 4.2a).

(a) Vault Boy 1 (b) Vault Boy 2 (c) Vault Boy 3

Figura 4.2: Todos los Vault Boy

4.2 Ecuaciones

La facilidad de composición de ecuaciones es una de las cosas que más atrae de LaTeX a muchos autores. LaTeXmantiene dos renderizadores diferentes, uno para el texto y otro para las ecuaciones, denominados modo párrafo y modo matemático[1]. El modo párrafo es el modo por defecto y no se le llama explícitamente. Al modo matemático, sin embargo, se le invoca de varias maneras diferentes.

Modo en párrafo

La forma más común es la forma "en línea", donde el texto para el modo matemático se encierra entre dos signos $. Por ejemplo, veamos la frase del listado 4.3.

Listado 4.3: Ejemplo de inserción de fórmulas en linea

```
El pequeño teorema de Fermat dice que si $p$ es un número primo, entonces
    , para cada número natural $a$, con $a>0$, $a^p \equiv a (\mod p)$
```

La frase quedaría como sigue:

El pequeño teorema de Fermat dice que si p es un número primo, entonces, para cada número natural a, con $a > 0$, $a^p \equiv a(mod p)$

[1] Existe un tercer modo, denominado *LR mode* o *left-to-right mode*, raramente utilizado y que no trataremos aquí

Ecuaciones en bloque

Cuando en lugar de poner una ecuación dentro de un párrafo existente la queremos insertar en su propio espacio independiente hacemos uso de los entorno `equation` o `align`, dependiendo de si queremos una o más ecuaciones en el bloque, respectivamente. Por ejemplo, en el caso de una única ecuación, sería similar al ejemplo siguiente:

```
\begin{equation}
  \int_a^b f'(x) \, dx=f(b)-f(a)
\end{equation}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a) \tag{4.1}$$

Sin embargo, en el caso de que quisiésemos más de una ecuación en el mismo bloque haríamos uso del carácter & para indicar en qué punto se alinean las ecuaciones; por ejemplo:

```
\begin{align}
    u  &= \arctan x \\
    du &= \frac{1}{1 + x^2}dx
\end{align}
```

$$u = \arctan x \tag{4.2}$$

$$du = \frac{1}{1+x^2}dx \tag{4.3}$$

Por último, si no tenemos por qué referenciarlas en el texto, podemos hacer uso de los entornos `equation*` y `align*`

```
\begin{equation*}
  \int_a^b f'(x) \, dx=f(b)-f(a)
\end{equation*}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a)$$

4.3 Cuadros y tablas

Los cuadros son una forma muy eficaz de presentar información. En los resultados de casi cualquier trabajo existen cuadros de algún tipo para que los datos se comprendan de un único vistazo (o para que al menos sea más fácil identificarlos.

¿Por qué "cuadro" en lugar de "tabla"?
Tal y como se indica en el FAQ de CervanTex, *table* (inglés) y *tabla* (español) son falsos amigos; el inglés *table* tiene un sentido más general que el español *tabla*, cuyo uso es únicamente para aquellos cuadros dedicados a la disposición de números (e.g. tabla de multiplicar o tabla de logaritmos).

Sin embargo, las tablas suelen ser bastante complicadas en LaTeX. Para no escribir demasiado, la respuesta para casi toda maquetación de tabla está en https://www.tablesgenerator.com/. En serio, no perdáis el tiempo si no es estrictamente necesario. La maquetáis visualmente, la generáis (con estilo *booktabs*) y a correr.

4.4 Código fuente

Para la gestión de los listados de código fuente se utiliza el paquete `listings`. El estilo usado es una modificación[2] de Solarized desarrollado por Ethan Schoonover.

Existen muchas formas diferentes de incluir listados de código en una memoria. Aquí introducimos los más comunes.

Código en párrafo

Bloques de código

Insertar código en párrafos no es tan común como insertar bloques enteros. Para ello haremos uso del entorno `lstlisting`. Por ejemplo:

```
\begin{lstlisting}
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

[2] En realidad la modificación es cambiar el fondo de crema a blanco

Nos daría el siguiente resultado:

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Una cosa que hay que tener en cuenta es que dentro de un entorno `lstlisting` se ignoran todos los comandos de LaTeX [3] y el texto se imprime tal y como se ha introducido. Esto incluye los tabuladores y espacios de principio de línea.

Al igual que en el código de párrafo, también podemos especificar en qué lenguaje está escrito el código para que se resalten en éste las palabras reservadas. Por ejemplo:

```
\begin{lstlisting}[language=python]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Código directamente desde fichero

Esta forma es muy común, ya que se usa tanto para hacer referencia al código fuente de la aplicación directamente, como a código separado en ficheros para mantener el tamaño de la memoria

[3] En realidad no todos, y si no mira en estos fuentes cómo hemos metido el fin de bloque `lstlisting` dentro del propio bloque.

manejable.

Suponiendo que tenemos el fichero `sources/snippets.py`, para incluirlo entero basta con usar el comando `lstinputlisting`:

```
\lstinputlisting[language=python]{sources/snippets.py}
```

Con este comando conseguiríamos el siguiente resultado:

```
def all_unique(l):
    return len(l) == len(set(l))

def is_palindrome(l):
    return l == l[::-1]
```

En caso de que se desease importar sólo parte del fichero, se pueden indicar las filas que delimitan el trozo de código. Por ejemplo:

```
\lstinputlisting[
    language=python,
    firstline=4,
    lastline=5
]{sources/snippets.py}
```

Esto haría que sólo se imprimiesen las filas 4 y 5, correspondientes a la segunda función:

```
def is_palindrome(l):
    return l == l[::-1]
```

Ambas opciones son opcionales, y los valores por defecto de `firstline` y `lastline` serán el principio y el final del fichero respctivamente.

Etiquetando bloques de código

Al igual que con el resto de bloques *float* (e.g. figuras o tablas), se pueden[4] (**deben**) añadir pies de texto a los bloques de código, lo cual hace más legible su cometido.

Para ello basta con añadir el argumento `caption` a las opciones del bloque. Por ejemplo:

```
\begin{lstlisting}[language=python, caption=Función para determinar
   cuando una palabra \textt{w1} es anagrama de otra palabra \textt{w2}]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

Listado 4.4: Función para determinar cuando una palabra `w1` es anagrama de otra palabra `w2`

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

4.5 Referencias cruzadas: etiquetas (*labels*)

Las etiquetas (*label*) son una herramienta muy útil en el proceso de composición tipográfica. Se puede pensar en ellas como punteros a zonas de interés del documento, de tal manera que se les pueda referenciar sin necesidad de conocer su posición final en la composición.

Por ejemplo, lo normal es que en un libro, a la hora de referenciar una figura, aparezca una frase del estilo "[…] como muestra la Figura 3 […]". Lo que es bastante raro son las frases del estilo "[…]

[4] Pongo *pueden* porque es opcional, pero en realidad se **deben** poner, porque si no los listados de fuentes quedan horribles, como el de esta plantilla por ejemplo (échale un vistazo si no lo has hecho antes).

como muestra la Figura de los moñecos amarillos […]" o "[…] como muestra la siguiente Figura […]"[5].

Una de las propiedades más útiles y, en ocasiones, infravaloradas de LaTeXes la facilidad y potencia de su sistema de etiquetado. Este sistema permite referenciar tablas, listados de código fuente, ecuaciones, capítulos, secciones, etc., con facilidad y flexibilidad. Además, LaTeXlas numera y referencia automáticamente, cambiando la numeración en función de las adiciones y supresiones sin que el autor tenga que hacer nada.

Para referenciar un elemento, lo primero que hay que crear es una etiqueta **después** del elemento a referenciar. Esto ya lo hemos visto anteriormente, por ejemplo en el listado 4.1. Si nos fijamos, se declara una etiqueta justo después de la etiqueta `caption` con el nombre `fig:img-vault-boy`. De esta manera, podemos referenciar varios indicadores de la figura, como se muestra en el listado 4.5

Listado 4.5: Referenciando una figura y su página

```
Mira la Figura~\ref{fig:img-vault-boy} en la página~\nameref{fig:img-
    vault-boy}.
```

Dicho listado daría el siguiente resultado:

«Mira la Figura 4.1 titulada Vault Boy approves that en la página 12.»

¿Por qué a mí me aparece el símbolo ?? en lugar de una referencia? Pues lo más seguro es que sea un error a la hora de escribir la etiqueta, un type. Menos común, pero también puede pasar, es que el documento no se haya compilado bien. Hay que tener en cuenta que LaTeX fue creado en una época donde las máquinas tenían poca (¡poquísima!) RAM, y para funcionar lo que se hacían eran varias compilaciones sobre el documento, almacenando los valores temporales en ficheros. Y como nadie quiere perder tiempo en cambiar y *debuggear* algo que funciona estupendamente bien, no se reimplementa. De todas formas, si te animas, ahí tienes un buen proyecto que si lo sacas adelante te va a hacer muy famoso.

[5] Sí, bueno, quizá la segunda frase no es tan rara, pero siempre es preferible referenciar directamente a dar posiciones relativas.

5. Referencias, glosarios e índices

En este capítulo mostraremos cómo crear nuevas referencias bibliográficas, entradas en el glosario de términos y acrónimos y palabras para el índice.

5.1 Referencias bibliográficas

Hay muchas formas diferentes de gestionar las referencias bibliográficas, así que aquí hemos decidido elegir una de ellas por considerarla la más cómoda y simple, que es mediante el paquete *biblatex*.

El fichero de referencias, `references.bib`, incluirá una entrada por cada una de las referencias que se citan durante la memoria. Luego, en el cuerpo del texto, se podrán hacer referencias a dichas entradas y será LaTeX después quien se encargue de indexar correctamente, crear los hipervínculos y maquetar automáticamente.

El fichero `references.bib` puede tener muchas más de las referencias que se citan en el cuerpo del texto. Sin embargo, sólo aparecerán las referencias que se citen en el texto.

No has dicho en ningún momento *bibliografía* Sí. Las referencias bibliográficas, también conocidas como lista de referencias o simplemente referencias, son todas aquellas fuentes bibliográficas que han sido citadas a lo largo del documento. La bibliografía, también conocida como referencias externas, es simplemente una lista de recursos utilizados, citados o no. Como generalmente los no referenciados no se usan para dar soporte a un texto científico se suelen descartar.

¿Cómo creamos nuevas referencias?

El fichero `references.bib` contará cero o más entradas con la estructura mostrada en el listado ??.

Listado 5.1: Estructura general de una referencia

```
@tipo{id,
    author = "Autor",
    title = "Título de la referencia (libro, artículo, enlace, ...)",
    campo1 = "valor",
    campo2 = "valor",
    \ldots
}
```

En esta entrada, `@tipo` indica el tipo de elemento (p. ej. `@article` para artículos o `@book` para libros) e `id` es un identificador **único en todo el documento** para el elemento. El resto de campos dependerán del tipo de la referencia, aunque generalmente casi todos los tipos comparten los campos de `author`, `title` o `year`.

AQUÍ CONTAR CÓMO SE AÑADEN ENTRADAS, LAS POSIBLES OPCIONES Y EL ENLACE A DONDE SE DESCRIBE TODO EN PROFUNDIDAD

¿De qué manera puedo citar las referencias?

AQUÍ CONTAR LAS DIFERENTES OPCIONES PARA REFERENCIAR ARTÍCULOS. PONGO LA BÁSICA, QUE ES [1], PARA QUE ME APAREZCA EL CAPÍTULO DE REFERENCIAS.

5.2 Referencias cruzadas

5.3 Referencias a recursos externos

5.4 Glosario y acrónimos

Para gestionar el glosario y los acrónimos se hace uso del paquete `glossaries`. Es, quizá, algo complejo de configurar ya que permite muchas opciones diferentes.

Aquí proponemos una configuración por defecto para que lo único que haya que hacer sea añadir y referenciar las entradas.

Definiendo los términos del glosario

Un listado aquí todo majo:

```
\newglossaryentry{ex}{
    name={sample},
    description={an example}
}
```

5.5 Índice

6. Licencia

Cuando se publica la obra en el archivo digital, por defecto lo hace con la licencia de CC Reconocimiento - Sin obra derivada - No comercial. Aunque usar cc guay, esta licencia no es una licencia libre y a algunos nos parece algo malo.

Considero que todo el conocimiento generado en una universidad pública ha de ser público y libre. Por ello, aunque por defecto está puesto ese, recomiendo usar `documentclass{upm-report}`, de tal manera que mientras se respete que se comparte igual, todo el mundo puede hacer de esta obra el uso que quiera.

De todas formas, indican que si el autor quiere usar otra licencia de Creative Commons deberá ponerse en contacto con el Administrador del Archivo Digital UPM: archivo.digital@upm.es.

Google

Bibliografía

[1] W. S. McCulloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity», *The bulletin of mathematical biophysics*, vol. 5, nº 4, págs. 115-133, 1943.

A. Escuelas y títulos

A continuación se describen todas las opciones de grados y títulos disponibles en la memoria.

A.1 Escuelas

Las escuelas disponibles se describen en el cuadro A.1.

Clave	Valor
`etsii`	E.T.S. de Ingenieros Industriales
`etsisi`	E.T.S. de Ingeniería de Sistemas Informáticos

Cuadro A.1: Relación entre el código de la plantilla y la escuela a la que se refiere

De momento no están todas, así que si te apetece añadir la tuya puedes, o bien contactar con los autores, o bien modificarlo (mira el apéndice B) y también contactar con los autores, así lo podemos hacer público con la mayor cantidad de usuarios posible.

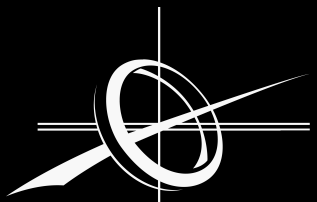
A.2 Titulaciones

RELLENAR; FALTAN, ADEMÁS DE TITULACIONES DE OTRAS ESCUELAS, LOS MÁSTERS Y LOS PROGRAMAS DE DOCTORADO

ETSII: Grados: 05TI (Grado en Ingeniería en Tecnologías), 05IQ (Grado en Ingeniería Química), 05IR (Grado en Ingeniería de Organización) y 05IE (Grado en Ingeniería de la Energía) ETSISI: Grados: 61CDIA (Grado en Ciencia de Datos e Inteligencia Artificial), 61CI (Grado en Ingeniería de Computadores), 61IW (Grado en Ingeniería del Software), 61SI (Grado en Sistemas de Información), 61TI (Grado en Tecnologías para la Sociedad de la Información)

B. ¿Cómo ampliar la plantilla?

C. Lista de paquetes incluidos


Universidad
Politécnica
de Madrid
ETSI SISTEMAS
INFORMÁTICOS