Universidad Nacional de Quilmes

Elementos de Programación y Lógica

Cuadernillo del estudiante

por Alan Rodas Bonjour

Ver. 0.4 (17/03/2019)

PUBLICADO ONLINE POR ALAN RODAS BONJOUR

HTTP://ELEMENTOSDEPROGRAMACIONYLOGICA.WEB.UNQ.EDU.AR

Primera edición, Febrero 2019

Índice general

Introducción

Imagen de la supercomputadora Thunderbird, en el Sandia National Laboratory..
Fotografía del Departamento de Energía de los Estados Unidos.

La **informática** es la disciplina o campo de estudio que abarca el **conjunto de conocimientos, métodos y técnicas referentes al tratamiento automático de la información, junto con sus teorías y aplicaciones prácticas, con el fin de almacenar, procesar y transmitir datos e información en formato digital utilizando sistemas computacionales**. Los datos son la materia prima para que, mediante su proceso, se obtenga como resultado información. Para ello, la informática crea y/o emplea sistemas de procesamiento de datos, que incluyen medios físicos en interacción con medios lógicos y las personas que los programan y los usan.

Temas a tratar

Como este campo requiere el uso de **computadoras** (en el sentido amplio de la palabra), comenzaremos en la unidad I por analizar qué es una computadora, ver como funcionan, de que partes están compuestas y una breve historia de las mismas. Se debe tener en cuenta que nos centraremos en computadoras electrónicas, también llamadas clásicas, las cuales son las que predominan en el mercado. Sin embargo, analizaremos otras variantes de computadoras cuando veamos la historia de las mismas, como las computadoras mecánicas. Así mismo, los conceptos teóricos que rigen las ciencias de la computación, auguran como posibilidades otros modelos computacionales, todavía en etapas muy inmaduras, como las computadoras cuánticas, o las biológicas, tema que tocaremos brevemente.

En la unidad II analizaremos como las computadoras almacenan **información**, y como se procesa y se interpreta la misma. Haremos una mínima introducción a lo que es el concepto de datos binarios, y veremos como con ese sistema se almacenan números, letras, colores, imágenes, videos y cualquier información que uno desee. Veremos como funcionan los editores de texto y los visualizadores de archivos, y tendremos un primer acercamiento a un lenguaje informático mediante el uso de lenguajes de marcado. Evitaremos mencionar los estándares que rigen las computadoras modernas, pues solamente nos interesará el concepto subyacente a los mismos, por lo que algunos lectores que posean conocimiento de esta temática podrán sentir que se ha tomado

una aproximación muy laxa. Tenga en cuenta el lector que la omisión es intencional.

Posteriormente en la unidad III analizaremos la **lógica**, uno los fundamentos teóricos más importantes que subyace en la disciplina. Analizaremos parte de los formalismos que rigen en esta ciencia y veremos como se aplican esos conceptos en matemática y en ciencias de la computación en general. El foco del presente no es realizar un análisis exhaustivo de esta disciplina, la cual es sumamente amplia para abarcarla en tan pocas páginas. Aparecerán entonces los conceptos de conectivas, y las utilizaremos para formular preguntas nuevas a partir de otras que ya teníamos disponibles, un concepto muy útil a la hora de programar. También hablaremos de razonamientos deductivos, de sus premisas y conclusión, y aplicaremos los conocimientos de forma práctica para introducirnos en la lógica proposicional y descubrir sus formalismos. Finalmente, veremos razonamientos que no pueden resolverse con la lógica proposicional y mojaremos nuestros pies en la superficie de la lógica de predicados, analizando muy brevemente sus formalismos.

Por último, en la unidad IV veremos como se procesa información utilizando **programación**. Crearemos nuestros primeros programas que solucionarán pequeños problemas, y analizaremos los mismos para descubrir algunas de las estructuras más comúnmente utilizadas en programación. Nuevamente, la disciplina es amplia, y por tanto solo abordaremos pequeñas cuestiones puntuales, como la secuencia de comandos para generar programas, el uso de estructuras de control para estructurar ideas y reducir la cantidad de código, así como generar soluciones más genéricas. Por sobre todo trabajaremos en planificar nuestras soluciones en forma de tareas pequeñas para simplificar la complejidad del problema a resolver, así como comunicar correcta y eficazmente nuestra solución. Los lectores que tengan conocimientos previos en esta área sentirán al comienzo que lo presentado es muy básico, y que aporta poco a su saber. Sin embargo, recomendamos no tomarse a la ligera lo presentado en esta unidad, pues el abordaje elegido difiere del seleccionado por muchos autores, y hace especial hincapié en la comunicación de la solución al problema, concepto fundamental para convertirse en un programador profesional.

Estructura del libro

El presente se estructura entonces en cuatro unidades y una únidad adicional que contiene el índice y una serie de apendices. Cada unidad a su vez, se divide en capítulos, y cada capítulo en diferentes secciones.

Cada capítulo del presente se apoya en lo visto en capítulos anteriores, y si bien pueden leerse de forma independiente, esto trae aparejado el supuesto de ciertos conocimientos ya vistos, por lo que recomendamos leer el libro de forma completa y continuada.

El texto intenta ser lo más breve y directo posible, para permitir la comprensión incluso a los menos asiduos a la lectura. Por otro lado, se intenta explicar los conceptos con definiciones más pragmáticas que enciclopédicas. Si bien esto puede no ser del agrado de todos los lectores, creemos que la comprensión del concepto es más importante que una definición, la cual en general varía de autor en autor. Si bien en algunos lugares se sacrifica precisión en pos de la comprensión, en todo lugar donde esto ocurre es deliberado, pues no se desea sobrecargar el libro en conceptos que, si bien importantes para un profesional de la industria, no son relevantes para quien busca adquirir conocimientos generales de la temática.

Al final de cada capítulo se presentan ejercicios prácticos a realizar, los cuales tienen dos propósitos. Por un lado, muchos de ellos sirven para asentar los conocimientos teóricos vistos utilizando un enfoque práctico. Por otro, algunos de ellos ocultan detalles teóricos interesantes que no pudieron entrar en estas páginas por motivos de espacio.

Los ejercicios suelen tomar unos pocos minutos en resolverse, dado que el lector haya comprendido correctamente todos los temas de la unidad, y recomendamos que se vayan realizando a medida que se avanza con la lectura.

Al final de cada unidad se presenta un conjunto de bibliografía adicional que permitirá al lector expandir sus conocimientos o abordar con mayor profundidad en las diversas temáticas abordadas aquí, así como también tener otros puntos de vista.

La última parte del libro contiene el índice con las palabras clave, y los lugares en donde dichas palabras aparecen en el libro de forma relevante. También posee una serie de apéndices, los cuales presentan información adicional sobre una temática particular, o resultan útiles de tener a mano a la hora de realizar las actividades propuestas.

Requisitos y saberes previos

Para poder comprender los contenidos de este libro no se requieren demasiados saberes previos. Se espera sin embargo, que el lector posea algunos conceptos básicos de matemática, que incluyen aritmética elemental y algo de geometría.

También se espera que el lector sepa utilizar una computadora, o se haya cruzado en algún momento con una. Si bien existen diversos tipos de computadora, y diversas formas de interactuar con las mismas, el presente toma por supuesto que el lector ha utilizado una computadora de escritorio o notebook, y que sabe lo que es un teléfono móvil inteligente (smartphone), haya o no interactuado con uno.

La unidad III, en su última sección, se vuelve más sencilla de comprender si el lector se cruzó previamente con el concepto de función (en términos de análisis matemático), aunque no es imprescindible para la correcta comprensión de los temas abordados.

Para seguir la teoría de la unidad IV, así como resolver los ejercicios propuestos puede recurrir solamente a la lectura, lápiz y papel. Sin embargo, realizar los ejercicios en la computadora facilitará una mayor comprensión de los conceptos, simplificando el proceso de aprendizaje. Para esto se espera que lector cuente con acceso a una computadora, ya sea una notebook o máquina de escritorio, y acceso a internet (ya sea continuo o esporádico).

Comentarios finales

La disciplina de las ciencias de la computación y la informática en general son campos relativamente nuevos (aunque sus orígenes datan de más de tres milenios atrás, no se comenzó a considerar la informática como una ciencia en si misma sino hasta comienzos del siglo XX). Aún así, esta disciplina ha evolucionado mucho en un muy corto período de tiempo, revolucionando de forma sustancial el mundo en el que vivimos.

Si bien las computadoras son omnipresentes en nuestro día a día, todo indica que recién estamos nadando en la superficie de las aguas de lo que es posible con la informática, y que el futuro nos depara una inmersión en las mismas, a profundidades que aún no alcanzamos a vislumbrar. Las posibilidades auguradas por las ciencias de la computación son muchas y muy amplias, y es un campo en completa expansión, que trabaja cada vez más de forma interdisciplinar con ciencias tan dispares como la física, la biología, la neurología, la lingüística, la economía, la política, e incluso la filosofía.

Por otro lado, como ya mencionamos, la informática tiene que ver con el proceso y manejo de

la información. Ser meros usuarios de tecnologías, desconociendo completamente los conceptos subyacentes más básicos de los mismos, nos vuelve vulnerables a engaños y manipulaciones por quienes controlan la información. ¿Es bueno el voto electrónico? ¿Qué datos tienen las empresas sobre mi persona y cómo pueden utilizarlos? ¿Es seguro utilizar una tarjeta de crédito en internet? Si conocemos los conceptos subyacentes a la tecnología que utilizamos, responder esas preguntas se vuelve más sencillo, y nos permite tomar mejores decisiones como ciudadanos digitales.

Muchos especialistas afirman que dentro de pocos años en el futuro, desconocer estos conceptos será equivalente a no saber leer y escribir en la sociedad actual. Por eso este libro apunta a brindar un panorama introductorio, pero lo suficientemente abarcativo como para que el lector obtenga un panorama general de los conceptos principales de la informática generalmente desconocidos.

I Computadoras

1. Computadoras

Placa base de una computadora con sus circuitos impresos.
Fotografía de Blickpixel.

Las utilizamos todos los días, rigen nuestra vida cotidiana, y el mundo moderno dejaría de funcionar sin ellas. Las computadoras son omnipresentes en nuestro día a día, ya sea que trabajemos directamente con ellas o no. Pero, ¿Qué es exactamente una computadora?, y más aún ¿Pará que sirve?.

Este capítulo intenta responder todas esas interrogantes y generar un glosario de los términos más comunes en informática. No es el fin de este capítulo centrarnos en detalles, sino obtener un panorama general de una computadora y como esta funciona. Capítulos siguientes retomarán este contenido y complementarán aportando detalles más específicos.

Comencemos entonces por definir qué es una computadora:

Definición 1.1 Una **computadora** es una máquina electrónica o electromecánica que recibe datos, los analiza, procesa y transforma, convirtiéndolos en información conveniente y útil para el posterior uso por seres humanos.[1]

Es decir, una computadora es un dispositivo cuya única función es la de procesar datos. Esos datos, también suelen ser llamados "información", y podrían ser procesados por un ser humano sin ningún problema. Lo que hace a una computadora realmente interesante es que **es capaz de procesar gran cantidad de datos en muy poco tiempo**. Así, una tarea que implique procesar grandes volúmenes de información, o procesarla de formas complejas (tarea que le podría llevar a una persona meses, o incluso años) a una computadora le puede llevar pocos segundos.[2]

Toda computadora está formada físicamente por numerosos componentes electrónicos y electromecánicos. Además, toda computadora cuenta con algún dispositivo que permite ingresar datos a la misma (por ejemplo, un teclado o un mouse), y algún otro dispositivo que permite ver los datos que la computadora procesó (por ejemplo, un monitor o una impresora).

Cuando se habla de computadora, generalmente pensamos en la típica **computadora de escritorio** (con su voluminoso gabinete, un monitor, teclado y mouse). Sin embargo, las computadoras

tienen hoy en día formas cada vez más diversas, como por ejemplo, las **notebooks**, y sus primas pequeñas las **netbooks**. Las **tablets**, los **celulares** e incluso los **relojes inteligentes** son también ejemplos de computadoras. Cada una tiene sus particularidades, por ejemplo, la forma de ingresar datos en una notebook no es la misma que en un celular. Sin embargo, todas son computadoras y comparten características comunes.

Vieja Apple II.
Exhibición del Musée Bolo, Lausana, Suiza.

De hecho, la definición de computadora que utilizamos es tan amplia, que abarca una serie de dispositivos que en general no clasificaríamos como computadoras, pero que, por sus características, lo son. Además de las anteriormente mencionadas podríamos incluir también:

- **Consolas de videojuegos**
- **Juguetes electrónicos**
- **Sistemas de domótica (IoT)**
- **Computadoras de abordo (en autos, barcos y aviones)**
- **Robots**
- **Calculadoras graficadoras programables**
- **Microcontroladores**

Muchos otros dispositivos también podrían ser consideradas computadoras, o incluyen computadoras como un componente fundamental para poder funcionar.[3]

En realidad
Técnicamente cualquier dispositivo electrónico que realice cálculos (cómputos) es una computadora. Sin embargo, en general nos referimos como computadora a aquellos dispositivos que son "programables", es decir, que pueden configurarse para llevar a cabo distintas tareas por parte del usuario. Está distinción no nos resulta relevante.

Una computadora está constituida por dos partes fundamentales: El **hardware**, que comprende a la **parte física** de una computadora, y el **software**, es decir, la parte **parte lógica**. A continuación trataremos más en detalle ambas partes.

1.1 Hardware (parte física)

Comencemos por definir a qué nos referimos cuando hablamos de hardware:

Definición 1.2 El **hardware** comprende todos los elementos físicos que componen a la computadora. Es decir, el conjunto de circuitos, cables, perillas, palancas, botones, luces, displays, dispositivos de impresión, motores, imanes, placas metálicas, etc.[1]

Una definición más pragmática sería: **si no funciona y lo puedo patear, es hardware**.

Los componentes físicos que tenga la computadora, serán dependientes del tipo de computadora con la que contemos. Así por ejemplo, algunas tendrán un teclado, mientras que otras contarán con una pantalla táctil; unas tendrán un monitor, mientras otras tendrán un display indicador; etc.

Muchos de estos dispositivos trabajan con elementos que no son puramente eléctricos, sino físicos o mecánicos. A estos se los denomina **dispositivos analógicos**. En general, los datos que emiten o reciben tienen forma de onda (son valores continuos sobre un rango).

Los dispositivos que no emplean partes mecánicas, sino solamente electrónicas, suelen emitir o

recibir datos discretos (valores puntuales, que no conforman un espectro). Este tipo de dispositivos se denominan **dispositivos digitales**.

La mayoría de las computadoras de hoy en días (aunque no siempre es así, ni tampoco fue así siempre) contienen una serie de partes relativamente estándar entre las que destacan:

Fuente de alimentación (Fuente de poder) Se trata de un transformador de electricidad que modifica la corriente que le llega (por ejemplo, desde un enchufe en la pared, con 220V, o desde una batería de litio), al voltaje que requiere la máquina para funcionar (por ejemplo, 12V, 5V, etc.). Como las computadoras son aparatos electrónicos, todas requieren electricidad de alguna forma. Este dispositivo se encarga entonces de darle energía a todos los otros componentes de la máquina.

Motherboard (Placa base o tarjeta madre) Consiste en una tarjeta o placa, que contiene los circuitos principales de la computadora impresos en un material conductivo (cobre, oro, etc.) sobre su superficie. La placa cuenta con ranuras (la mayoría estandarizadas) en donde se conectan los diversos componentes del equipo (aunque a veces los mismos pueden estar soldados directamente a la placa). Sus circuitos se encargan de conectar y a los diversos componentes entre sí, permitiendo que la electricidad fluya de uno a otro componente.

CPU (Central Processing Unit o Procesador) Es lo que se conoce como un circuito integrado. Está compuesto de millones de transistores microscópicos, que son capaces de realizar miles de millones de cálculos por segundo. Además el CPU se encarga de coordinar al resto de los componentes, manejar los datos que entran, los que salen etc.

Memoria RAM Otro importante circuito integrado, pero que no realiza cálculos, sino que almacena información. Es el lugar donde el equipo almacena los datos temporales de las cuentas y acciones que va realizando (no debe confundirse con el lugar en donde se guardan datos a largo plazo). La memoria RAM solamente almacena información mientras la computadora esté encendida. Una vez la máquina se apaga, la información se pierde.

Estos componentes son lo que en general llamamos **computadora** en sí misma. El resto de los componentes son conocidos como periféricos. Los periféricos se conectan a la **computadora** para agregar capacidades de entrada y salida de información.[4]

En el lenguaje coloquial, usamos el término **computadora** para referirnos tanto a los componentes principales como a todos los periféricos conectados a los mismos. En general utilizamos el término con esta última acepción pero al hablar de periféricos el término computadora suele referir solo a los componentes principales.

1.1.1 Periféricos

Los **periféricos** son componentes de hardware adicionales que se acoplan a los componentes principales de la computadora. El nombre deriva de que los mismos se colocan en torno a estos (en la periferia).

Definición 1.3 Un **periférico** es un dispositivo de hardware que se acopla a los componentes centrales de una computadora y que permiten el ingreso y/o egreso de información a la misma.[5]

Los periféricos solo tienen por finalidad el permitir ingresar datos o información a los componentes principales de la computador, o por el contrario, extraer la información que los componentes principales hayan generado para presentársela al usuario. Algunos de estos dispositivos cumplen ambas funciones. El almacenamiento a largo plazo de información también puede considerarse como entrada y salida de datos, y por eso los dispositivos que almacenan información son considerados

Diversos componentes de hardware de una computadora de escritorio.
Ilustración de GregoryJCL.

también periféricos.

Los periféricos se **clasifican en 4 categorías**, según si permiten ingresar datos, extraerlos, ingresarlos y extraerlos al mismo tiempo, o almacenar los datos a largo plazo.

Periféricos de Entrada

Los periféricos de entrada son aquellos que utilizamos para ingresar datos a la computadora. El dispositivo lee lo que el usuario ingresa de forma analógica (por ejemplo, que palancas accionó, o que botones presionó) y traduce esa información a impulsos eléctricos que serán enviados a la computadora para que esta procese la información ingresada.[5]

Ejemplos comunes de este tipo de periféricos son:

- **Teclado**
- **Mouse**
- **Trackpad**
- **Micrófonos**
- **Cámaras Digitales**
- **Webcams**
- **Joysticks**
- **Scanners**

Fuera de los tradicionales, hoy en día los dispositivos móviles cuentan con una amplia cantidad de periféricos de entrada en forma de sensores. Además, computadoras que comprenden sistemas más específicos, como de automatización industrial, pueden contar con sensores más especializados aún. Algunos ejemplos incluyen:

- **Giroscopios**
- **Potenciómetros**
- **GPS**
- **Sensores de luz**
- **Sensores de temperatura**
- **Detectores de humo/CO_2**
- **Perillas, palancas y botones**
- **Sensores de movimiento**
- **Sensores de humedad**
- **Lectores de tarjetas magnéticas**
- **Lectores RFID**
- **Lectores de tarjetas perforadas**

La gran mayoría de los periféricos de entrada suelen ser sensores analógicos cuya señal es transformada en digital por algún componente para ser luego enviada al CPU. También existen por supuesto casos de periféricos completamente digitales.

Periféricos de Salida

Los dispositivos de salida incluyen todos aquellos dispositivos mediante los cuales la computadora nos muestra indicaciones de los datos que procesó, o que se encuentra procesando.[5] Algunos ejemplos de estos dispositivos incluyen:

- **Pantallas o monitores**
- **Impresoras**
- **Displays digitales**
- **Luces (LEDs)**
- **Parlantes**
- **Auriculares**

Nuevamente podemos encontrar sistemas con casos más específicos, como puede ser:

- **Proyectores**
- **Dispositivos de lectura braille**
- **Fax**
- **Plotter**
- **Motores**
- **Impresoras 3D**

Al igual que en los periféricos de entrada, dentro de los periféricos de salida podemos encontrar algunos completamente digitales y otros analógicos. El detalle de cada dispositivo queda sujeto a la curiosidad del lector y no pertinente a este libro.

Periféricos de Entrada/Salida

En esta categoría entran todos los dispositivos que son mixtos, es decir, que permiten tanto el ingreso de datos, como la salida de los mismos. Algunos autores catalogan también aquí los dispositivos de almacenamiento.[5] Entre este tipo de dispositivos encontramos:

- **Pantallas táctiles / multitáctiles**
- **Impresoras multifunción**
- **Cascos virtuales**

Un teclado Ascom BEG 100 de Ascom Bankensysteme AG, creado específicamente para funcionar con el sistema de información financiera Reuters Dealing 2000. El teclado en este caso es un periférico de entrada y de salida. Fotografía de DAFlippers.

La realidad es que muchos dispositivos que tradicionalmente eran considerados como solo de entrada, o solo de salida, actualmente están entrando cada vez más en esta categoría. Pueden tomarse como ejemplo los joysticks de consolas de videojuegos. Inicialmente considerados dispositivos de

entrada, hoy la mayoría incluye sistemas de estímulo al jugador, como luces o vibraciones. Las tabletas graficadoras son otro ejemplo de dispositivo que ahora incluye en forma de pantalla táctil, o en forma de tinta electrónica, selectores de herramientas que varían dependiendo de la aplicación, o muestras del dibujo realizado.

> **Sabías qué**
> Los **Joysticks** de la consola de videojuegos **Nintendo 64** permitían acoplarle un dispositivo llamado **Rumble Pack**, que transmitía vibraciones a las manos del jugador de acuerdo a lo que estuviera sucediendo en el juego. Sony incluyó la idea en sus controles **DualShock** de la consola **PlayStation** y pronto se transformó en un estándar en la industria. Así, la mayoría de los joysticks actuales ya no son solo dispositivos de entrada, sino de entrada y salida, algo a lo que los jugadores de videojuegos ya se han acostumbrado.

Periféricos de Almacenamiento

Por último, los periféricos de almacenamiento son dispositivos que permiten almacenar datos para su uso en un futuro. Algunos permiten escribir datos múltiples veces, y leer en muchas ocaciones. Otros permiten una única escritura, y muchas lecturas. Algunos ejemplos incluyen:

- **Discos rígidos magnéticos**
- **Discos de estado sólido**
- **Discos ópticos (CDs/DVDs/Bluerays)**
- **Discos magnéticos extraíbles (Floppys)**
- **Memorias flash (pendrives/tarjetas de memoria)**
- **Cintas magnéticas (cassetes/VHSs)**
- **Tarjetas perforadas**

En general, muchos de estos dispositivos se pueden separar en dos partes, el medio de almacenamiento en sí, y dispositivo que oficia de lector y/o escritor de los datos sobre el medio. En algunas otras ocaciones el medio y el dispositivo que lee o escribe están acoplados en un mismo elemento, no pudiendo separarse, y por tanto considerado un único elemento.

Como mencionamos, muchos de estos medios requieren de un lector especial para poder leer su información. Estos dispositivos pueden ser considerados dispositivos de entrada. También suele ser necesario un dispositivo para escribir en dichos medios, considerado en general como un dispositivo de salida. Cuando el mismo dispositivo se utiliza tanto para leer como para escribir información (por ejemplo una **lecto-grabadora de CDs**) estos suelen considerarse de entrada y salida. El medio en sí (por ejemplo, el CD) no suele ser considerado un periférico.

Algunos autores separan el medio de almacenamiento de su dispositivo de lectura y escritura, y catalogan solo a estos últimos como periféricos de entra, de salida, o de entrada y salida, según el caso. Otros prefieren incluir al medio como una parte necesaria del dispositivo, y agrupan a ambos como un dispositivo de entrada y salida. Finalmente, muchos autores han optado por colocar dichos dispositivos en una categoría completamente propia, como se ha hecho en este caso.

1.1.2 Redes

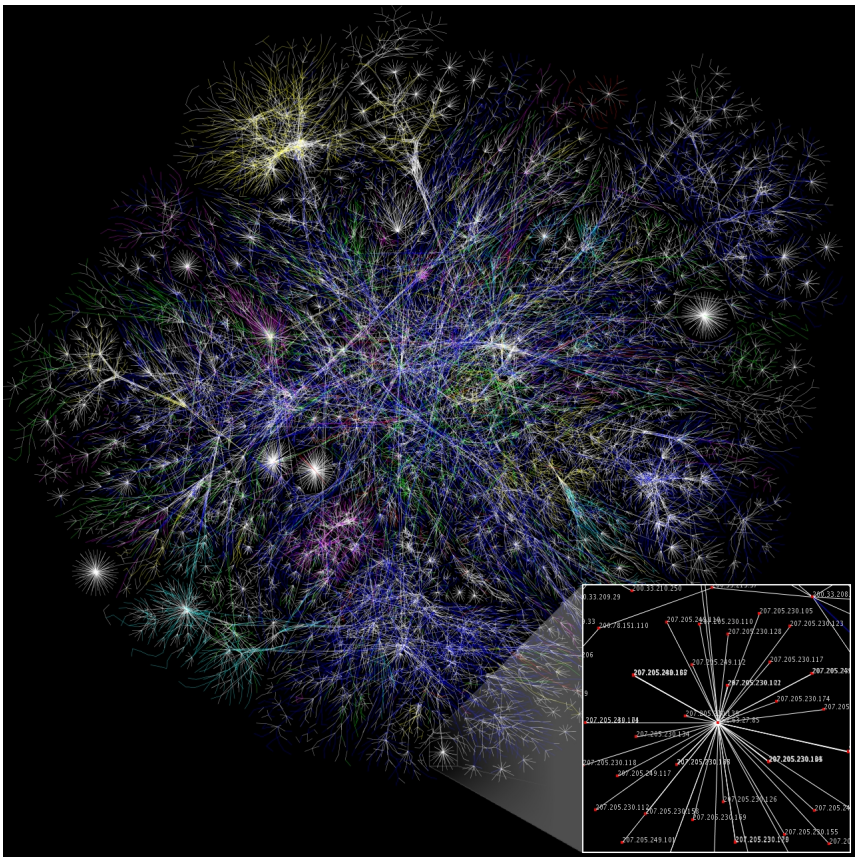
Ahora que sabemos qué es una computadora, resulta interesante saber que dos o más computadoras pueden conectarse entre sí, formando una **red**.

Una **red** no es más que una serie de computadoras conectadas entre sí a través de algún periférico, por ejemplo, una **tarjeta de red cableada**, o una **tarjeta de red inalámbrica**.

Definición 1.4 Una **red de computadoras** es un conjunto de equipos conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.[6]

Los equipos pueden conectarse directamente unos con otros, o utilizar alguna computadora particular que oficia de gestor de comunicaciones. En general, la computadora utilizada suele ser una que esté específicamente diseñada para tal fin, como un **router** o un **switch**.

Cuando una serie de computadoras se comunican entre sí, es posible que desde una máquina se acceda a los recursos de otra (archivos, programas, periféricos, e incluso al procesador y memoria). La forma en la que se configura la red permite que esta actúe de formas diversas, por ejemplo, como una única gran computadora, o como diversas máquinas que trabajan de forma independiente pero utilizando una serie de recursos en común, entre otras opciones.



Un mapa de Internet a principios de 2005. Cada línea representa la unión entre dos equipos, y el largo expresa el tiempo medio de comunicación entre ellos. Los colores representan la ubicación de los elementos.
Mapa creado por The Opte Project.

A su vez **una red puede estar conectada a otra red**, formando redes más grandes. Por ejemplo, una computadora puede estar conectada mediante una red pequeña a los dispositivos que tiene cerca, como impresora inalámbrica, celular y otros, utilizando por ejemplo, tecnología Bluetooth (red conocida como **red de área personal** o **PAN**). A su vez, esa red puede formar parte de una red más amplia que conecta todas las computadoras de la casa, y a la que se conectan también televisores y

otros dispositivos, generalmente mediante cables o WiFi, utilizando un router como dispositivo que maneja las comunicaciones (red conocida como **red de área local** o **PAN**). A su vez el router puede estar conectado con el servidor de un **proveedor de servicios de Internet** (conocido como **ISP** por las siglas en inglés), que permite conectarnos a servidores a redes que están más allá de nuestro router. Estos servidores conformas así una red muy amplia (conocida como **red de área ancha** o **WAN**) que a su vez se conecta a otras para formar una red aún más grande, **Internet**.

Para que las máquinas puedan conectarse, estas deben entender como comunicarse entre sí. Para eso, existen protocolos específicos que dictaminan como deben hacerlo, y existen diversos protocolos para diversas funcionalidades. Por ejemplo, el protocolo IP determina como una computadora puede identificar a otras (y a sí misma) en la red, el protocolo HTTP dictamina como dos equipos pueden compartir recursos de hipertexto, por mencionar solo algunos de ellos.

Definición 1.5 **Internet** (a veces llamado el internet o la internet) es un conjunto. descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos de internet (a veces llamadas TCP/IP), lo cual garantiza que las redes físicas heterogéneas que la componen, formen una red lógica única de alcance mundial.[7]

1.1.3 Relación del hardware y el software

Lo que hemos aprendido hasta ahora es que el hardware corresponde a toda la parte física de la computadora, es decir, a todos los circuitos electrónicos, chips, cables, etc.

Sin embargo, no hemos mencionada nada sobre la electricidad que fluye por dichos circuitos. Y es así porque el hardware, efectivamente no incluye a la electricidad, la cual, es considerada parte del software. Es decir, que **el hardware no sirve para nada sin software**. Si no hay electricidad corriendo por los circuitos de la computadora, la misma es solo una pila de metales y plástico.

Cabe destacar que, en general, no pensamos al software como electricidad, pero, la realidad es que todo lo que no sea físico en la computadora (en general denominado lógico) termina siendo nada más que eso, un flujo de electrones que se mueven con cierta energía por los diferentes elementos del hardware.

1.2 Software (parte lógica)

Mientras que el hardware corresponde a toda la parte física, el **software** corresponde a toda la **parte lógica**. Es decir, **toda señal eléctrica que recorre los circuitos**. Aunque en general no nos referimos al software como electricidad, sino que hablamos de **programas**, **archivos**, **directorios**, etc.

Definición 1.6 El **software** es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.[1]

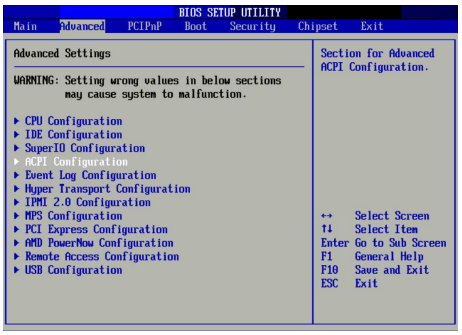
Nuevamente una definición pragmática sería **si no anda y solo lo puedo insultar pero no golpear, entonces es software**.

Una computadora sin software no sirve para nada, pues sin electricidad, simplemente no funciona. Más aún, no solo debe haber electricidad, sino que debe haber electricidad en los lugares correctos en el momento correcto para que el equipo funcione de la forma esperada.

Si bien los programas son software, el software no son solo programas, como ya mencionamos. Sin embargo, en ocasiones la palabra software se utiliza muchas veces por diversos autores o en el uso cotidiano como sinónimo de "programa", omitiendo los otros elementos que están incluidos en el software. El lector debe ser cuidadoso al consultar diversa bibliografía para identificar con que significado se está utilizando el término.

1.2.1 Firmware

Como una computadora no hace nada sin software, **toda computadora viene de fábrica con algún software mínimo que permite al menos encender la misma y determinar cual es el tarea que la misma debe realizar**. Este sistema se conoce como **firmware**.[8]



Pantalla que muestra la configuración avanzada de la BIOS AMI. Fotografía de Richard Masoner.

En las computadoras de escritorio y notebooks, el **firmware** incluye al sistema **BIOS**, que permite al usuario del equipo configurar cosas como:

- Determinar si, al iniciar, el equipo debe ejecutar lo que encuentre en el disco rígido, en la lectora de CD/DVD o un pendrive
- Optar por sobreexigir al procesador para que funcione más rápido (a pesar de que esto puede dañar el equipo)
- Visualizar datos de control como la temperatura interna de la máquina.
- Elegir si se desea habilitar o nos los puertos USB del equipo.

Estas son solo algunas de las muchas tareas que permite realizar, las cuales varían además dependiendo del fabricante y de equipo a equipo.

Además ese sistema realiza un chequeo de todos los componentes internos cada vez que se inicia la computadora, y notifica a los usuarios si algo está fallando, ya sea mediante mensajes en la pantalla, o mediante una serie de notificaciones sonoras.

Definición 1.7 El **firmware** es un programa informático que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Está fuertemente integrado con la electrónica del dispositivo, es el software que tiene directa interacción con el hardware, siendo así el encargado de controlarlo para ejecutar correctamente las instrucciones externas.[57]

El firmware puede venir de fabrica en un circuito que no puede modificarse, conocidos como **ROM**, o venir en circuitos que son "re-escribibles' conocidos como **EEPROM**, lo cual posibilita actualizar el firmware por nuevas versiones del mismo (que en general corrigen errores o agregan nuevas características).

No solo las computadoras de escritorio traen un firmware. También muchos periféricos como **tarjetas gráficas** o **tarjetas de sonido** suelen incluir su propio firmware. Otros elementos como **routers** o **modems** suelen traer un firmware complejo (o incluyen también un pequeño sistema operativo al que consideran parte del firmware) que se encarga de administrar las computadoras conectadas en red a estos dispositivos.

Algunos grupos de desarrolladores incluso crean **firmware alternativo** (en oposición al oficial que es ofrecido por el fabricante del dispositivo). Este firmware alternativo puede incluir características adicionales a las ofrecidas por el fabricante, aumentando en algunos casos de forma significativa las posibilidades del aparato al exprimir todas las características del hardware.

Sabías qué
DD-WRT y **OpenWrt** son firmwares alternativos y libres para diversos routers inalámbricos. En general el sistema amplifica las posibilidades ofrecidas por los fabricantes. Diversas marcas y modelos de routers son soportados, como Linksys, Buffalo, Belkin, ASUS, Mitsubishi, Motorola, Siemens y D-Link, entre otros.

El proyecto es lo que se conoce como **software libre**, lo que permite a cualquier programador contribuir al proyecto, y está basado en **Linux**, uno de los sistemas operativos libres más populares con una gran base de usuarios.

1.2.2 Sistemas Operativos

El firmware hace muy poco, casi nada a la vista de un usuario actual de una computadora. Salvo en máquinas muy especificas, como pueden ser sistemas de automatización industrial, automotores, u otros, el usuario final requiere de un programa adicional que se encargue de manejar la mayor parte del equipo, y permita al mismo realizar tareas comunes de forma sencilla. Este programa se conoce como **sistema operativo**.

El sistema operativo se encarga de coordinar los diversos componentes del equipo para que todos funcionen, así como de proveer una capa común a los programas que utilizará el usuario final para que estos realicen sus tareas de forma segura y eficiente. El sistema operativo es el encargado de determinar como se lee un CD, como se guardan datos en el disco rígido, o que programa debe ejecutar en que momento, y por cuanto tiempo, entre otras varias tareas. También tiene la potestad de determinar que aplicaciones pueden acceder a que archivos, o terminar programas cuando lo considere necesario, con la finalidad de evitar que una aplicación pueda realizar acciones maliciosas en el equipo. Si bien el sistema operativo es también un programa, este tiene mayores "privilegios" que el resto de las aplicaciones.

Definición 1.8 Un **sistema operativo** es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes.[9,10]

La mayoría de los sistemas operativos modernos, proveen adicionalmente una serie de herramientas que posibilitan al usuario utilizar los periféricos del equipo sin demasiada complejidad, o realizar tareas comunes, como crear y editar archivos. Aunque esas herramientas no son parte en sí mismo del sistema operativo, suelen mencionárselas como tales. Por eso, se suele hablar del **núcleo del sistema operativo** (es decir, la parte importante encargada de manejar el hardware), y del resto del sistema (las herramientas, iconos, imágenes, sonidos, etc. que vienen junto con el sistema para permitir una mejor experiencia al usuario). La distinción es sutil, y la mayoría de las veces no es necesaria. Sin embargo, pasa a ser importante en sistemas operativos como Linux, donde precisamente, Linux refiere solo al núcleo, mientras que el sistema en su conjunto se conoce como distribución. Así, existe un único "sistema operativo" Linux, pero cientos de "distribuciones" (o distros) Linux.

Existen muchos sistemas operativos, algunos para uso general en computadoras de escritorio, otros pensados para dispositivos móviles, otros pensados con fines más específicos, como servidores

de internet, o router, robots, etc. También hay sistemas operativos que son desarrollados y vendidos por empresas, y otros que son hechos por comunidades de desarrolladores autoconvocados. Algunos tienen fines comerciales, otros educativos, otros de investigación, y algunos son simplemente por entretenimiento.

En las computadoras de escritorio y notebook actuales es común encontrar instalado alguna versión del sistema operativo **Windows**. Windows es desarrollado y vendido por **Microsoft Corporation**, una de las empresas dedicadas al desarrollo de software más grandes del mundo. El sistema se volvió el más popular en las computadoras de escritorio gracias a las prácticas monopólicas que la empresa lleva adelante, practicando acuerdos con fabricantes de hardware para que el sistema venga pre-instalado, y utilizando lobby político para conseguir acuerdos con instituciones educativas y entidades gubernamentales para la enseñanza de sus sistemas como currícula informática. En Argentina es común que la enseñanza de informática solamente incluya en el programa escolar las nociones del uso de Windows y del paquete de aplicaciones de oficina **Microsoft Office**. Casos análogos se dan en muchos países, no solo latinoamericanos, sino también africanos, europeos y asiáticos.[11]

Linux es un sistema operativo que merece una mención aparte. Hoy en día Linux ha avanzado mucho desde sus inicios en 1991, cuando un estudiante finés, **Linus Torvalds**, decidió crear su propio sistema operativo y compartirlo con el mundo. Linus adhirió prontamente a las ideas de **Richard Stallman**, un desarrollador que, cansado de un modelo de software en donde los usuarios quedan prisioneros de las políticas y reglas que imponga la empresa, comenzó un movimiento alternativo en donde son los usuarios lo que tienen el poder y las decisiones sobre el software, el movimiento de **software libre**. Stallman también desarrollo una fundación para la promoción del software libre, e inició el **Proyecto GNU**, un proyecto destinado a crear herramientas de software totalmente libres. Por ser software libre, Linux puede ser modificado por cualquier usuario con los conocimientos técnicos necesarios para adaptarse a una gran variedad de necesidades y situaciones.

Escritorio del sistema operativo Huayra Linux 3.2
Fotografía de Conectar Igualdad.

Así, diferentes **distribuciones** de Linux existen para solucionar diversos problemas de múltiples usuarios, o para satisfacer diferentes gustos y preferencias. Algunas de las distribuciones más populares para computadoras de escritorio son **Ubuntu**, **Linux Mint**, **Debian**, **Fedora**, **RedHat Enterprise Linux**, **SuSe Linux**, **Arch**, **Manjaro** y **Zorin OS**, pero la lista es inmensa. Existen además distribuciones que se enfocan en casos muy específicos: sistemas para servidores de internet, routers, celulares, smart TVs, etc. Como gran cantidad de distribuciones se ofrecen de forma gratuita, es cada vez más común encontrarlo instalado de fábrica en equipos de escritorio o notebooks, ya que el fabricante minimiza costos de venta final al no tener que pagar el software de Microsoft. Además, Linux se ha convertido en un sistema robusto y confiable, tanto así que el 70 % de todos los servidores de internet utilizan Linux como sus sistema operativo.

Y habla de distribuciones Linux, la más popular hoy en día es casi sin lugar a dudas una versión modificada y adaptada para funcionar sobre teléfonos celulares inteligentes y tablets, **Android**. Android tiene una cuota de mercado del 80 %; es decir, 8 de cada 10 celulares o tablets utilizan Android. El sistema surgió de la mano de una pequeña empresa respaldada por **Google** en 2007, realizando acuerdos con la Open Handset Alliance (un conjunto de empresas de hardware, software

y telecomunicaciones). Al ser libre, modificable y altamente personalizable, los distintos fabricantes pueden modificar la apariencia del sistema para adaptarlo a su marca y estilo, pero permitiéndolo interactuar con el gran ecosistema de desarrolladores y aplicaciones de la plataforma, y gozando de los avances generales que se llevan adelante en el sistema.

1.2.3 Programas

Pantalla donde se listas las aplicaciones del equipo en un sistema Windows 8.1.
Fotografía de India7 Network.

Los programas o aplicaciones que utilizamos a diario en la computadora son también parte del software. Programas como **Word**, **Excel** o **PowerPoint**para ofimática, **Photoshop**, **GIMP** para diseño gráfico, **iTunes** o **Windows Media Player** para reproducción de música y video, **Super Mario Brothers**, **Pacman**, **Angry Birds** u otros juegos, **todos son software**.

Llamamos a este software **programa** o **aplicación**. Estos programas son los que brindan funcionalidades específicas al usuario final, como una planilla de cálculo o un reproductor de video. Se "paran" sobre el sistema operativo, y por tanto, quienes lo desarrollan pueden abstraerse del hardware subyacente del equipo. Por ejemplo, el programa de reproducción de música simplemente le dice al sistema operativo "quiero reproducir este sonido", pero desconoce completamente si el sonido va a salir por los parlantes de la computadora, los auriculares u otro; es el sistema operativo quien se encarga de ello.

Definición 1.9 Una **aplicación** es un programa informático diseñado como herramienta para permitir a un usuario realizar uno o varios tipos de tareas.[12]

El desacoplar el sistema operativo de las aplicaciones permite a los desarrolladores de software generar más aplicaciones, con más funcionalidades, en menos tiempo, pues no tienen que preocuparse por detalles que, a los fines del programa no son relevantes. No solo eso, sino que al desentenderse del hardware subyacente, las aplicaciones pueden correr en cualquier equipo, sin importar el hardware que posea, por lo que son más genéricos y menos susceptibles a cambios en el hardware.

Banner publicitario de CorelDRAW Graphic Suite 2018 con su precio de lista
Captura de pantalla del sitio oficial de Corel.

Las aplicaciones son una parte fundamental del mercado informático, tanto en aplicaciones de escritorio como en celulares y tablets. Tanto es así que las tiendas de aplicaciones de Android y iOS facturaron en 2018 un total de 34.400 millones de dólares. Las posibilidades de este mercado en las plataformas móviles generan un circulo virtuoso en donde los desarrolladores generan constantemente nuevas y mejores aplicaciones, ampliando la oferta, mientras que los usuarios esperan de forma constante nuevas apps, descartando las viejas. Mientras que en el mercado de celulares y tablets hay una idea de "aplicaciones para lo que sea", en los sistemas operativos de escritorio la oferta suele estar más enfocada a aplicaciones para profesionales y de alta calidad. Así, aplicaciones como Photoshop o Microsoft Office se venden a cientos de dólares.

También hay un gran mercado para aplicaciones gratuitas que basan sus ganancias en publicidad o en venta de datos y patrones de uso de los usuarios, como Google o Facebook. Muchas veces los usuarios desconocen el verdadero costo de las aplicaciones y servicios que utilizan, no entendiendo que sus datos pueden ser vendidos a terceros por la empresa que les ofrece la aplicación de forma gratuita.

Las aplicaciones de software libre son también una moneda corriente, habiendo en general una alternativa libre a casi cualquier aplicación comercial, las cuales, en general se ofrecen de forma gratuita. La calidad final de estas aplicaciones varía enormemente, habiendo desde aquellas en donde la versión libre supera incluso a las versiones comerciales (Un ejemplo es el reproductor de video VLC), y casos en donde la versión comercial está tan por arriba de la libre que optar por esta última se vuelve muy difícil si uno utiliza el software con motivos comerciales (Por ejemplo Adobe Illustrator o CorelDraw están muy por arriba de sus contrapartidas libres).

Finalmente hay un gran mercado de aplicaciones a medida, generadas específicamente para solucionar el problema puntual de una persona u organización. Este software brinda una solución ad-hoc. Estas aplicaciones suelen ser en ocaciones mucho más costosas que aplicaciones genéricas, pues solo se venden a un único usuario, quien debe cargar con la totalidad de los costos de desarrollo.

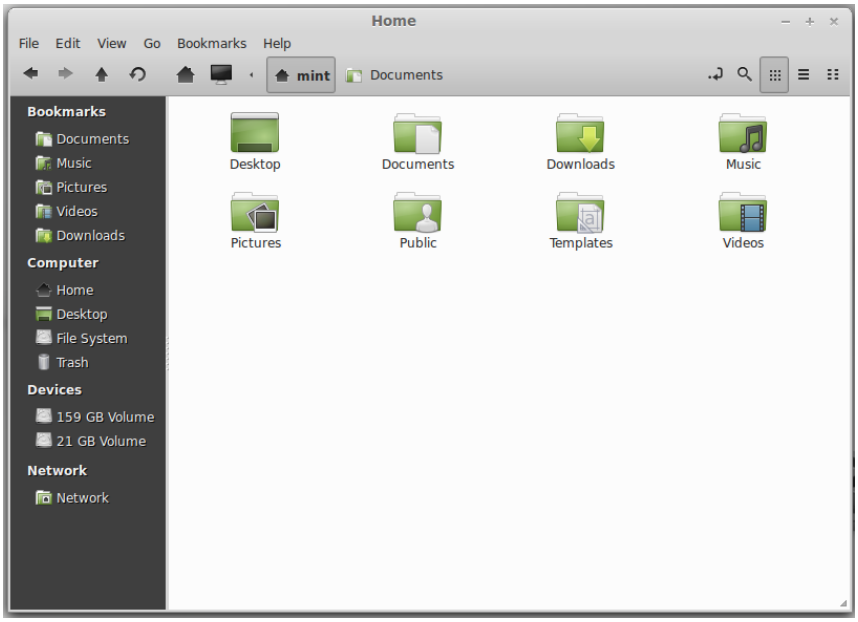
Definición 1.10 **Ad-hoc** es una locución latina que significa literalmente "para esto". Generalmente se refiere a una solución específicamente elaborada para un problema o fin preciso y, por tanto, no generalizable ni utilizable para otros propósitos.

1.2.4 Archivos y Directorios

Un **archivo** digital no es más que la representación lógica de un archivo físico (Por ejemplo, una hoja con texto escrito, una foto, una tarjeta, etc.). El término surge en base a una analogía con los elementos en papel que se manipulaban en una oficina previo a la era de la digitalización. Sin embargo, un archivo también puede representar elementos que uno no encontraría en dichas oficinas (o al menos no como papel), como música, videos, etc. En las computadoras modernas, los archivos son guardados a bajo nivel como **bits**, un término que discutiremos en la unidad II.

Cada archivo posee un **nombre**, que lo identifica, así como un **tamaño** (la cantidad de bits que se utiliza para almacenarlo), y un **tipo** (que dictamina al sistema operativo de que forma deberá procesar ese archivo, por ejemplo, si representa una foto, o un video, etc.). Además, dependiendo del sistema operativo y otros detalles técnicos pueden o no tener más características, como permisos (quien puede acceder al archivo y para hacer qué cosa), marcas temporales (por ejemplo, la fecha en la que fue creado el archivo, la fecha de la última modificación, etc.), por mencionar algunas.

En general (aunque depende del sistema operativo) los archivos no se encuentran "sueltos" en el equipo, sino que se organizan en **directorios** (llamadas también carpetas). Un directorio no es más que la representación virtual de una carpeta física (nuevamente basado en la analogía de la



Captura de Pantalla de Nemo, el manejador de archivos de Linux Mint.
Imagen de Gaba P.

oficina), en donde uno puede agrupar varios archivos. Generalmente, un directorio pueden contener a su vez, otros directorios, formando lo que se conoce como una **jerarquía de directorios**.

Para manipular un archivo informático es importante saber también en que carpeta se encuentra almacenado, ya que su nombre y la carpeta son lo que lo identifica de forma única en todo el sistema.

Definición 1.11 Un **archivo informático** es un conjunto de bits que son almacenados en un dispositivo. Un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. Un archivo puede representar cualquier tipo de dato, ya sea texto, imágenes, audio, video, programas u otros.[1]

Toda la información que se almacena en los discos rígidos, CDs, y otros periféricos de almacenamiento suele estar organizada en carpetas (en la mayoría de los sistemas operativos), y suelen haber carpetas estándar en donde se almacenan determinados archivos que el sistema requiere para funcionar.

Otras carpetas, en cambio, son manejadas por los usuarios, pudiendo crear nuevas, agregar o quitar elementos a la misma, modificar su nombre, moverlas a otra carpeta, copiarlas a otro dispositivo, etc. La carpeta le sirve al usuario para organizar sus archivos por algún criterio que considere apropiado para sus necesidades.

Definición 1.12 Un **directorio** es un contenedor virtual en el que se almacenan una agrupación de archivos informáticos y otros subdirectorios, atendiendo a su contenido, a su propósito o a cualquier criterio que decida el usuario.[1]

Sabías qué
La mayoría de los sistemas operativos modernos incluyen algún programa que permite ver cuales son los archivos y carpetas del sistema. Este programa se conoce en general como **Navegador de Archivos** o **Explorador de Archivos**.

Los sistemas operativos más antiguos también lo hacían, aunque de una forma menos intuitiva, utilizando la **terminal** o **consola**.

1.2.5 Otros elementos de software

Mencionamos algunos de los elementos más relevantes que son considerados software, como son el **firmware**, el **sistema operativo**, las **aplicaciones**, los **archivos** y los **directorios**.

Existen sin embargo otros elementos que podrían mencionarse, como las **utilidades**, que consisten en programas diseñados para realizar tareas de mantenimiento del sistema, **antivirus**, que consisten en un programa que verifica que los archivos y programas de la computadora no contengan código perjudicial para el usuario, **herramientas de desarrollo**, que consisten en programas pensados para programadores, para que estos puedan crear nuevos programas, etc.

Como ya mencionamos, a fin de cuentas estas categorizaciones son solamente desde un punto de vista del usuario, pues para el hardware, un programa se almacena en un archivo, al igual que una foto o un video, y todo termina siendo información pasando por cables, o para ser más específico, electricidad. Como usuarios de una computadora hablamos por supuesto de archivos, carpetas y aplicaciones, y no de cables y electricidad, realizando un proceso de abstracción. Retomaremos este último concepto sobre el final de la unidad II.

Vale la pena recordar que, el hardware no sirve para nada sin software que lo maneje, mientras que el software no existiría si no hubiera hardware. Ambos elementos forman una simbiosis, y si bien uno puede centrarse en el estudio de uno u otro con mayor o menor intensidad, es necesario comprender ambos elementos para tener una idea clara del funcionamiento de una computadora.

1.3 Actividades

A continuación se presentan una serie de actividades orientadas a reforzar los conceptos vistos en este capítulo. Realice las actividades investigando en Internet si lo cree conveniente.

Ejercicio 1.1 Mire el cartel a continuación

Determine.
¿Cuáles de los elementos mencionados corresponden a software?
¿Cuáles de los elementos mencionados corresponden a hardware? ■

Ejercicio 1.2 ¿Conoce otros sistemas operativos además de los mencionados? ¿Conoce algo más sobre los sistemas mencionados?
Investigue en internet para descubrir nuevos sistemas y ampliar su conocimiento de los vistos. ■

Ejercicio 1.3 ¿Se le ocurren otros dispositivos de hardware que use de forma cotidiana? ¿Son periféricos de entrada, de salida, de entrada y salida, o de almacenamiento? ■

2. Historia de las computadoras

Museo de Historia de las Computadoras en Mountain View, California, EE.UU..
Fotografía de Michael Kappel.

La historia de las computadoras está directamente ligada a las historias de la matemática, la lógica, la gramática y la ingeniería mecánica y electrónica. Así, los seres humanos han utilizado dispositivos que los ayuden a computar (calcular) diversas cosas desde hace milenios, así como desarrollado técnicas que hoy en día se utilizan ampliamente en ciencias de la computación.

Este capitulo busca realizar un pequeño recorrido histórico por el desarrollo de las ciencias de la computación, no solo desde una perspectiva del hardware, sino también del software y de las teorías que sustentan a la disciplina. En el transcurso de este viaje se espera que el lector no solo se lleve una serie de anécdotas, sino que pueda apreciar el estado de las ciencias de la computación en la actualidad como parte de un proceso evolutivo que continúa en desarrollo, así como descubrir algunos conceptos que verá de forma recurrente si opta por desarrollarse en la disciplina.

Pero no solo veremos el pasado de la computación, también haremos un pequeño análisis del presente, para cerrar el capítulo con un vistazo a lo que depara el futuro.

2.1 Los precursores de las computadoras modernas

Pensar que la computación es un desarrollo que se ha realizado solamente en los últimos años del siglo XX es una idea ingenua. Los seres humanos hemos tenido la necesidad de contar y calcular (**computar**) cosas desde hace mucho, mucho tiempo; tanto tiempo como la prehistoria.

Desde el desarrollo de pequeños utensilios para asistir en los trabajos de computo, pasando por el desarrollo de conceptos matemáticos y trabajos manuales de trabajadores y trabajadoras cuyo esfuerzo nunca fue reconocido ni remunerado, hasta el desarrollo de maquinarias con complejos engranajes que permitieran asistir en el trabajo computacional. Todos terminan por ser precursores, de una u otra forma, de un proceso mental que se fue generando en la humanidad; la necesidad de contar con computadoras.

Y es que, a medida que avanza la ciencia y la técnica, también avanza la necesidad de realizar cálculos cada vez más complejos, y cada vez, en menor tiempo. Las computadoras no son una

casualidad, son una necesidad intrínseca al proceso de desarrollo científico-tecnológico que ha desarrollado la humanidad.

2.1.1 La prehistoria de la computación

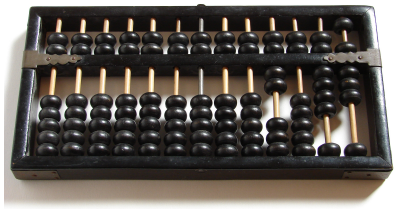
Los registros cuentan que el ser humano parece haber comenzado a contar desde al menos el año 20.000 AC. Por ejemplo, varios científicos, antropólogos e historiadores nos comentan sobre la utilidad del **hueso de Ishango**, un utensilio de hueso que data del paleolítico superior (20.000 AC.), y que habría servido para realizar conteos y cuentas. Incluso pareciera que ya en esa época, los seres humanos tenían alguna noción sobre números primos. Obviamente al no haber registros escritos, es imposibles saberlo con seguridad.[13]

Pero los verdaderos avances en materia de cuentas comenzaron en la antigua mesopotamia con la invención del **ábaco sumerio**, al rededor del 2500 AC. El ábaco consiste en un cuadro de madera con barras paralelas por las que corren una serie de bolas movibles. Este aparato permitía realizar cálculos sencillos (sumas y restas). Los chinos pronto en el 190 AC. mejorarían la idea con la creación del **suanpan**, conocido también como el **ábaco chino**. Este incluía las características del ábaco sumerio, pero las ampliaba permitiendo realizar también multiplicaciones y divisiones gracias a su innovador diseño.[14]

> **Sabías qué**
> El ábaco todavía se utiliza hoy en día para enseñarle operaciones sencillas a los niños, e incluso el suanpan es utilizado ampliamente por mercaderes, comerciantes y contadores en áreas donde el uso de tecnología es poco práctico.

En el 910 AC se crea también en China el **carruaje que apunta al sur**, un dispositivo que actúa de forma similar a una brújula, y que permitía a los mercaderes orientarse en los caminos. Este dispositivo es el primero en utilizar un engranaje, una de las piezas fundamentales de las **computadoras mecánicas**.

Pero si de engranajes hablamos, el **Mecanismo de Anticitera**, datado del 150 AC, es ya una completa **computadora mecánica**. Habría sido construido por científicos griegos con la intención de calcular eventos astronómicos y calendáricos, así como determinar la fecha exacta de los juegos olímpicos y otros certámenes deportivos de la antigüedad. La complejidad de este aparato no fue equiparada hasta siglos después, con la invención de los primeros relojes mecánicos.[15]



Un suanpan. Su diseño permite contar en decimal y hexadecimal.
Archivo de David R. Tribble.

En el área de las matemáticas, los avances de los chinos, utilizando el número cero, y los números negativos, así como los aportes del matemático Indio Pingala desarrollando el **sistema de numeración binario**, el cual es la base de todas las computadoras modernas sentarían las bases para la matemática moderna y por tanto para las ciencias de la computación.

El matemático Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī, conocido en general como al-Juarismi, quien fue un prestigioso matemático, astrónomo y geógrafo persa alrededor del año 800. Dentro de sus numerosas obras se le atribuye el haber propuesto por primera vez la idea de que, mediante una especificación clara y concisa de pasos a realizar, se podían calcular sistemáticamente, y por tanto se podrían desarrollar dispositivos mecánicos similares al ábaco que realizarán trabajaran sobre este para realizar las cuentas en lugar

de tener el usuario que emplear las manos. Este concepto es el precursor de lo que hoy en día se conoce como **algoritmo** (llamado así como una deformación del nombre al-Juarismi).

Por supuesto que solamente mencionamos algunos de los hitos más destacados del mundo antiguo, el cual estuvo lleno de avances científicos y matemáticos que aportaron de alguna forma u otra a las ciencias de la computación.

2.1.2 Las computadoras humanas

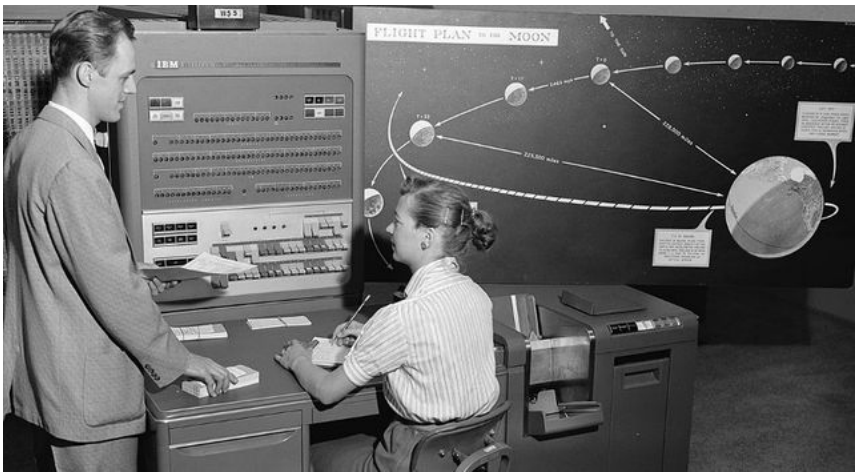


Imagen de una mujer operando una de las primeras computadoras, una IBM 704, en el Jet Propulsion Laboratory de la NASA.
Imagen de NASA/JPL-Caltech.

A medida que la ciencia avanzaba, se volvía necesario muchas veces realizar gran cantidad de cálculos para lograr comprender el universo que nos rodeaba. Por ejemplo, las áreas de astronomía y de física, solían requerir una gran cantidad de cuentas para arribar a conclusiones sobre las observaciones realizadas sobre diversos fenómenos.

Así, era común que los científicos contrataban a personas que realizaban los cálculos, mientras ellos se centraban en otros experimentos u observaciones. En ocaciones en los que los cálculos tardarían semanas o meses para realizarse, contrataban varias personas trabajando día y noche.

Las personas contratadas debían poseer sin embargo una serie de conocimiento (saber leer y escribir, ser capaz de realizar cálculos matemáticos, etc.) que no eran comunes en la población general. Esto y la necesidad de abaratar costos llevó a que se contraten en general mujeres, generalmente de clases sociales acomodadas y que hayan recibido educación. En un contexto donde la mujer no trabajaba (salvo en el hogar) la remuneración ofrecida podía ser baja.[16, p23]

A estas personas se las conocía como **computadoras**. El término que hoy usamos para los dispositivos, deviene de esas mujeres que, en la mayoría de los casos, la historía ha omitido de sus páginas.

Incluso luego de la invención de las primeras computadoras electromecánicas, las computadoras humanas se seguían empleando, pues eran consideradas más fiables, e incluso más económicas en algunos casos que comprar una computadora.

Sabías qué
Alexis Clairaut, Jérôme Lalande, y Nicole-Reine Lepaute, los astrónomos que calcularon la fecha del retorno del cometa Halley en 1750, habían contratado a un equipo de computadoras humanas para realizar el cálculo. En ese equipo se encontraba Nicole-Reine Etable de la Brière Lepaute, una de las primeras mujeres de las cuales se tiene registro que oficiaron de computadoras.

2.1.3 Las calculadoras mecánicas

En 1642 **Blaise Pascal**, matemático y físico francés, cansado de realizar cálculos manualmente, desarrolló la primer calculadora mecánica de la cual se tengan registros, la **Pascalina**.

Era una pequeña caja de madera que tenía en la tapa una hilera de discos numerados, con los agujeros para introducir los dedos y hacerlos girar. Funcionaba con engranajes, de forma que cuando una rueda daba una vuelta completa, avanzaba la otra rueda situada a su izquierda. Esta calculadora solo podía realizar sumas y restas. En 1672, **Gottfried Leibniz** perfeccionaría el dispositivo, creando el **Cilindro de Leibniz**, que permitía además multiplicar y dividir.[17]

Sabías qué
El Cilindro de Leibniz fue la base de numerosas calculadoras mecánicas creadas hasta la década de 1970, cuando finalmente se reemplazaron por calculadoras electrónicas.

Pero el invento tal vez más ambicioso fue el ideado por el matemático británico **Charles Babbage**. Babbage creo la **Máquina Diferencial**, una calculadora mecánica capaz de calcular gran cantidad de polinomios. Su máquina funcionaba con engranajes metálicos conectados, como una gran pieza de relojería. Babbage consiguió fondos de la Royal Astronomical Society para la construcción de su máquina. Sin embargo nunca logró hacerla funcionar. Los engranajes de la época no eran lo suficientemente buenos, y los materiales se rompían o generaban vibraciones que desarmaban el equipo. Desesperado, Babbage modificaba constantemente su diseño. Finalmente, la Royal Astronomical Society cortaría el financiamiento y el proyecto sería abandonado.

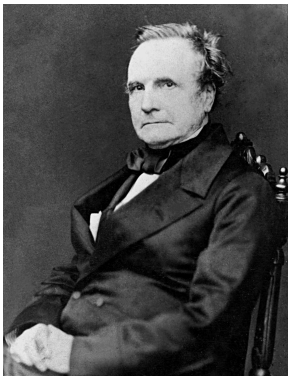
2.1.4 La primer computadora mecánica

Años después, en 1833, Charles Babbage volvería a intentarlo, pero esta vez, con una idea todavía más radical, la **Máquina Analítica**. Este equipo tenía la característica de ser **programable**. Es decir, se podía configurar para realizar distintas tareas en distintos momentos. Los planos diseñados por Babbage definían un complejo equipo capaz de realizar diversos cálculos, almacenar los resultados en una memoria, y hasta imprimirlos con un dispositivo similar a una impresora.

Sobre hombros de gigantes
Babbage no inventó todo de cero. Su sistema de engranajes se basaba en gran parte en el Cilindro de Liebniz. Como característica adicional su máquina permitía ser programada mediante tarjetas perforadas.

El sistema fue copiado de un telar creado en 1801 por **Joseph Marie Jaquard**, un tejedor y comerciante francés. El telar de Jaquard era **programable** a través de un sistema de **tarjetas perforadas**, y permitía a una persona diseñar complejos patrones en el tejido. Esta idea revolucionaria permitía reducir costos al permitir diseñar tejidos complejos con un muy bajo esfuerzo y reducido personal.

El invento de Babbage hubiera estado dotado de la mejor tecnología de la época.



Charles Babbage (1860). Dominio Público.

Charles Babbage buscó desesperadamente financiamiento para su dispositivo, pero tras el fracaso anterior, el gobierno y los inversores privados eran escépticos. Sin embargo, una joven aristócrata y matemática se acercó a Babbage por sus diseños. Se trataba de Augusta Ada Byron, Condesa de Lovelace, conocida más simplemente como **Ada Lovelace**, hija del famoso poeta Lord Byron. Ada reconoció el potencial en los diseños de Babbage, y comenzó a diseñar programas para el dispositivo. Por este motivo **Ada Lovelace se considera la primer programadora de la historia**.[18]

Los deficientes materiales y técnicas de construcción de la época hicieron que todos los esfuerzos de Babbage por construir la máquina analítica sean un fracaso. Así, el proyecto cayó en el olvido. Sin embargo, muchos años después, su máquina sería recreada utilizando los planos originales, y se demostraría que esta funcionaba correctamente. Por esto, **Charles Babbage es considerado el padre de la computadora moderna, así como de la impresora**. Además, realizó numerosas contribuciones a las áreas de la matemática y de la criptografía.

2.2 El despertar de la computación

La computación se empieza a plantar como una realidad cada vez más inminente con llegada del nuevo siglo (1900). Las ideas de Babbage serían reflotadas, ya en una época en donde los avances en la técnica metalúrgica permitía crear mejores dispositivos mecánicos y en donde los descubrimientos en materia de electricidad planteaba nuevas posibilidades.

Muchos científicos y matemáticos comienzan a plantearse las posibilidades y los límites de dispositivos que computen, o realicen pruebas matemáticas automatizadas, y grandes avances se realizan en estos términos, dando lugar a las ciencias de la computación como una disciplina nueva dentro de la rama de las matemáticas, pero que interactúa en gran medida con la electrónica, la mecánica y la ingeniería.

La Primera Guerra Mundial, y el advenimiento de una segunda gran guerra, en donde contar con una superioridad tecnológica podía ser clave para la victoria, llevó a que gobiernos tanto del eje como aliados financiaran serios desarrollos e investigaciones en el área.

2.2.1 La computación como teoría

En 1920 la matemática atravesaba una fuerte crisis. Los nuevos conceptos y abstracciones desarrollados en la disciplina a finales del siglo XIX traían aparejadas una serie de paradojas e inconsistencias en los fundamentos básicos de la matemática. En ese contexto, **David Hillbert**, un matemático alemán, propuso un proyecto de investigación conocido como “programa de Hillbert”, que intentaba solucionar el problema mediante la reformulación de la matemática sobre bases solidas y completamente lógicas. Hillbert creía que, en principio, esto podía lograrse, mostrando dos cosas:

1. Toda la matemática se sigue de un sistema finito de axiomas escogidos correctamente.
2. Se puede probar que tal sistema axiomático es consistente.

En 1931, **Kurt Gödel** publicaría su **teoréma de la incompletitud**, demostrando que no se podía probar la completitud de ningún sistema formal no contradictorio que fuera suficientemente amplio para incluir al menos la aritmética, sólo mediante sus propios axiomas. Así probó que el

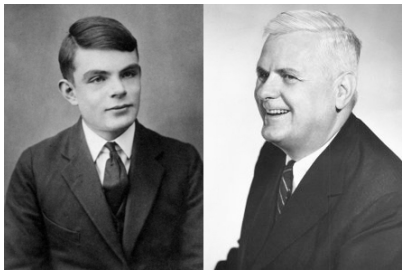
ambicioso plan de Hilbert era imposible tal como se planteaba.[19]

La necesidad de comprender las implicancias del trabajo de Gödel llevaron pronto al desarrollo de la **teoría de la computabilidad** y de allí a la **lógica matemática** como disciplina autónoma en los años 30. Esto sería la base para los desarrollos teóricos en ciencias de la computación realizados por Alonzo Church y Alan Turing.

La **teoría de la computabilidad** comenzó a estudiar que problemas pueden ser resueltos mediante un **algoritmo** lo cual permite de alguna forma saber que problemas pueden ser resueltos mediante una computadora, y cuales no.

Definición 2.1 Un **algoritmo** es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permiten llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad.

En Inglaterra, en 1936, **Alan Turing** desarrolla un modelo matemático conocido como **Máquina de Turing**. Si bien se conoce como "máquina", no hay ninguna construcción física involucrada, sino que se trata de un dispositivo hipotético, capaz de llevar adelante un algoritmo con el fin de solucionar un problema.



Alan Turing (izq.) y Alonzo Church (der.).
Fotografía de Dominio Público. Universidad de Princeton.

El dispositivo imaginado por un Turing consiste en un cabezal que manipula símbolos sobre una tira de cinta que se extiende infinitamente, basándose en una tabla de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de computadora y es particularmente útil en la explicación de las funciones del CPU dentro de una computadora.

El modelo formal de computadora de Turing le permitió demostrar que **existían problemas que una computadora no podía resolver, sin importar que tan poderosa fuera la misma**. Demostró que es imposible resolver el "Entscheidungsproblem", un problema matemático planteado por Liebniz en el siglo XVII, y retomado por Hillbert al plantear su propuesta de refundar la matemática.[19]

Al mismo tiempo pero del otro lado del Océano Atlántico, en Estados Unidos, **Alonzo Church** desarrolla un modelo matemático conocido como **cálculo lambda**, un sistema formal diseñado para investigar la definición de función, la noción de aplicación de funciones y la recursión.

Se puede considerar al cálculo lambda como el lenguaje universal de programación. Consiste en una regla de transformación simple (sustitución de variables) y un esquema simple para definir funciones. Hoy en día este modelo teórico es la base para el análisis formal de problemas computacionales, así como la creación de nuevos lenguajes de programación.

El cálculo lambda permitía probar, de forma similar a las máquinas de Turing, que **hay problemas que no pueden solucionarse con un una computadora**. Así, de forma independiente y mediante métodos distintos, dos científicos habían llegado a la misma conclusión.

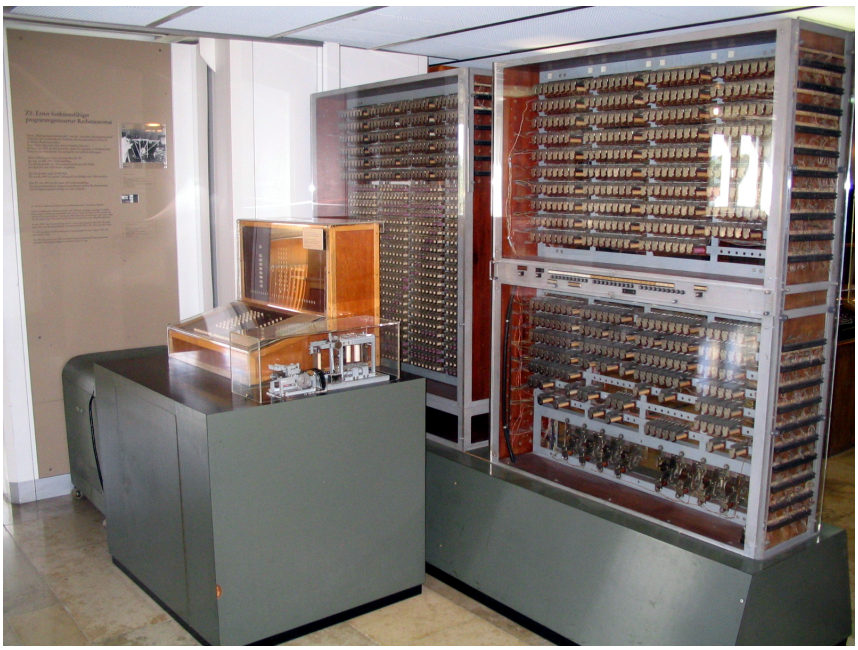
Posteriormente se formularía la **tesis de Church-Turing**, la cual formula hipotéticamente la equivalencia entre los conceptos de función computable y máquina de Turing, que expresado en lenguaje corriente vendría a ser "todo algoritmo es equivalente a una máquina de Turing". Esto quiere decir que el cálculo lambda y las máquinas de Turing, así como otros formalismos creados para analizar las posibilidades de la computación, son todas equivalentes.[19]

Lo interesante de estos desarrollos es que por primera vez se demostraron, no las utilidades prácticas de contar con computadoras, sino los límites de las mismas, pero también sus enormes posibilidades. Los formalismos desarrollados por estos científicos siguen presentes actualmente y son la base sobre la que se sustentan muchos de los avances en ciencias de la computación.

2.2.2 Primeras computadoras electromecánicas

Ya llegada la mitad de la década de 1930 nos encontramos en un contexto en donde los trabajos de Babbage comienzan a hacer soñar a muchos científicos e ingenieros. Además, los nuevos trabajos teóricos de Turing y Church auguraban grandes posibilidades para la computación.

Considerado hoy el inventor de las computadoras modernas, Konrad Zuse diseño la **Z1** en 1935, y terminó de construirla en 1938. La Z1 era una máquina completamente mecánica, era sumamente lenta y solo funcionaba unos pocos minutos cada vez sin necesidad de reajustes. Sin embargo, era la primer computadora realmente funcional. Más tarde realizó una segunda versión de su computadora utilizando **relés** (una especie de interruptor operado mediante magnetismo), bautizada la **Z2**.



Réplica de la Z3 de Zuse, la primer máquina completamente automática y digital (electromecánica) en el Deutsches Museum en Alemania.
Archivo del Deutsches Museum.

Esperando conseguir financiamiento para sus proyectos, presentó la Z2 al Deutsche Versuchsanstalt für Luftfahrt (Laboratorio Alemán para la Aviación) en 1940, siendo la presentación una de las pocas veces en que la Z2 funcionará realmente. De está forma consiguió el financiamiento del gobierno para crear la **primer computadora digital, automática y programable del mundo, la Z3**. Completada en 1941, la Z3 fue construida utilizando 2.600 relés y operaba a una velocidad de 4 o 5 Hz. Se debía inicializar la computadora manualmente, y ejecutaba programas que se almacenaban en cintas perforadas. La Z3 no fue considerada de vital importancia por el gobierno alemán durante la Segunda Guerra Mundial, por lo que operó poco tiempo antes de terminar destruida tras los bombardeos aliados a Berlin. Por haberse desarrollado en Alemania bajo el gobierno de Hitler, la historia omitió durante años el trabajo de Zuse.[3]

En el bando de los aliados, en Estados Unidos, comienzan a aparecer proyectos de desarrollos de computadoras, apoyadas por el estado (generalmente por la rama militar), universidades o empresas privadas.

Los avances en la tecnología y la reducción de costos permitían reemplazar relés (que contenían piezas mecánicas) con **tubos de vacío**, un dispositivo que actuaba de la misma forma, pero completamente electrónico, sin piezas móviles. Como contrapartida los tubos de vacío se iluminaban y generaban gran cantidad de calor, y en sus primeras versiones requerían de gran cantidad de espacio.

Ya en 1944 aparece **ENIAC**, financiada por el Ejercito de los Estados Unidos, y utilizada para el cálculo de trayectorias balísticas. La computadora ocupaba un sótano entero en la Universidad de Pensilvania. Físicamente, la ENIAC tenía 17468 tubos de vacío, 7200 diodos de cristal, 1500 relés, 70000 resistencias, 10000 condensadores y cinco millones de soldaduras. Pesaba 27 Toneladas, medía $2,4m$x$0,9m$x$30m$ ($167m^2$), requería la operación manual de unos 6000 interruptores, y su programa o software, cuando requería modificaciones, demoraba semanas de instalación manual.[20]

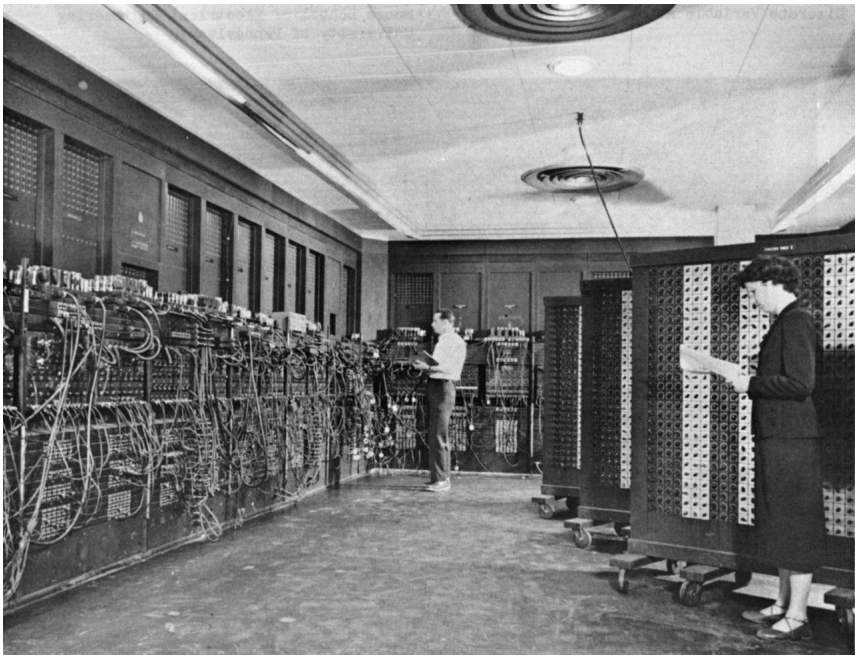
No poco después aparece la primer computadora construida por **IBM**, una de las empresas que más aportaría a la computación mundial. La computadora en cuestión, el **IBM Automatic Sequence Controlled Calculator (ASCC)**, más conocido como **Harvard Mark I**, fue una computadora electromecánica, construida por IBM y enviado a Harvard en 1944. Tenía 760.000 ruedas y 800 kilómetros de cable y se basaba en la máquina analítica de Charles Babbage. Medía unos 15,5 metros de largo, unos 2,40 metros de alto y unos 60 centímetros de ancho, pesaba aproximadamente unas cinco toneladas y poseía unas cubiertas de cristal que dejaban que se admirara toda la maquinaria de su interior.[20]

Sabías qué
ENIAC fue creada por **John Presper Eckert** y **John William Mauchly**, quienes se llevaron el crédito principal, pero su programación fue realizada exclusivamente por 6 mujeres que fueron generalmente omitidas de los libros de historia: **Betty Snyder Holberton**, **Jean Jennings Bartik**, **Kathleen McNulty Mauchly Antonelli**, **Marlyn Wescoff Meltzer**, **Ruth Lichterman Teitelbaum** y **Frances Bilas Spence**.

Al ENIAC le sucedió **EDVAC**, utilizado también para balística, mientras que los ingleses desarrollaron las máquinas **Colossus** (creadas con intención de descifrar los códigos alemanes). Otras como **UNIVAC** (la primera en ser vendida en los Estados Unidos) y la **Z4** de Zuse les sucedieron. Las computadoras eran todavía artefactos muy costosos (EDVAC por ejemplo tuvo un costo aproximado de unos US$ 100.000) y utilizados en principio solo con fines militares, pero pronto expandiéndose hacia la investigación científica y los negocios.

Sabías qué
Los tubos de vacío atraían insectos que se quedaban en ellos, causando el mal funcionamiento de los equipos. Lo técnicos debían entrar y remover los insectos para que la computadora vuelva a funcionar. A este proceso se le conoce como **debuggear** (del inglés quitar los bichos). El término quedó, y hoy en día se conoce con ese nombre al proceso de arreglar los errores en un programa.

A medida que aparecían más y más computadoras, también se comenzó a desarrollar una idea más seria sobre **arquitectura de computadoras**. Es decir, se comenzaron a desarrollar modelos y descripciones funcionales de los requerimientos y las implementaciones de diseño para varias partes de un computadora, en especial en la forma en que el procesador trabaja internamente y



Glen Beck y Betty Snyder programan la ENIAC en el edificio 328 de la Ballistic Research Laboratory (Filadelfia, Pensilvania).
Fotografía del Ejercito de los Estados Unidos.

accede a las direcciones de memoria.

Definición 2.2 La **arquitectura de computadoras** es el diseño conceptual y la estructura operacional fundamental de un sistema de computadoras.

La Mark I sentó las bases para lo que hoy se conoce como **arquitectura de Harvard**, que involucra una computadora en donde las pistas de almacenamiento y de señal se encuentran físicamente separadas para las instrucciones y para los datos. Por otro lado **John Von Neumann**, un matemático húngaro-estadounidense desarrolló el primer borrador sobre la arquitectura de EDVAC, describiendo detalles arquitecturales que simplificaban la arquitectura de Harvard, además de optimizar costos. La arquitectura pasó a ser conocida como **Arquitectura de Von Neumann**, y se transformó en un estándar, al punto de que casi todas las computadoras modernas la utilizan.[21]

2.2.3 IBM, Bell y grandes computadoras

Ya terminada la guerra, el boom de las computadoras estaba en el aire, y el potencial de estos dispositivos en otras áreas distintas a la guerra comenzaban a vislumbrarse.

Una empresa venía ya vendiendo dispositivos electrónicos para grandes compañías y fábricas, como máquinas tabuladoras y relojes de fichada. Se trataba de **IBM**. Con la experiencia del desarrollo del **Harvard Mark I**, y tras terminada la guerra, IBM crea la **IBM 701** (1953), considerada la primer computadora comercial basada en tubos de vacío. Se siguieron pronto otros modelos, como la **IBM 702** y la **IBM 650**.

Sabías qué
En 1956, **Arthur L. Samuel**, del laboratorio de IBM en Poughkeepsie, Nueva York, programó un IBM 704 para jugar a las damas utilizando un método por el que la máquina podía "aprender" a partir de su propia experiencia. Se cree que este es el primer programa de "auto-aprendizaje", una demostración del concepto de **inteligencia artificial**.

IBM se transformó rápidamente en la empresa líder en todo lo referente a computadoras. Invirtiendo gran cantidad de dinero en investigación y desarrollo (I+D). En 1957 IBM desarrolló el primer sistema de almacenamiento informático basado en disco, llamado el IBM 305 RAMAC. El **RAMAC ATOGA** es el predecesor de los discos duros actuales y estaba formado internamente por cincuenta discos.[22]

Sabías qué
IBM también desarrolló el lenguaje de programación científico **FORTRAN** (FORmula TRANslation) para facilitar la creación de programas. También fueron los inventores de otras tantas tecnologías ampliamente utilizadas hoy en día, como, **la memoria RAM**, **las tarjetas magnéticas** y **los códigos de barras**.

IBM no fue la única empresa que incursionó en tecnología. Otra de las grandes empresas que lo haría sería **AT&T** quien controlaba los teléfonos de Estados Unidos de forma monopólica, y principalmente su subsidiaria, **Bell Labs** (que gracias al monopolio contaba con uno de los presupuestos más altos del mundo en términos de investigación en tecnología).

Entre las patentes y descubrimientos más importantes de Bell Labs destacan **la libreta de un solo uso**, **el transistor**, **el láser**, **la fibra óptica**, **la tecnología DSL**, **la telefonía móvil**, **los satélites de comunicaciones**, **el sistema operativo Unix**, **el lenguaje de programación C** y **el lenguaje de programación C++**. Además, once de sus investigadores han sido galardonados con Premios Nobel.[23]

Las computadoras sin embargo, seguían siendo costosas, grandes, pesadas, y requerían de personal altamente capacitado para su instalación y armado. Por eso solamente se encontraban disponibles en universidades, dependencias del estado o grandes empresas privadas, que podían solventar los costos de estos aparatos.

Cabe destacar que la operación de las computadoras era tarea compleja. **Programarlas requería de expertos, que supieran como estaba armada la computadora (como estaban diagramados los cables, los circuitos, etc)** y por tanto, solamente ingenieros electrónicos o matemáticos eran capaces de operarlas. Adicionalmente, **como cada computadora era distinta, un programa escrito para una computadora no funcionaba en otra**, y un profesional que supiera operar una computadora tenía que volver a aprender todo de cero para poder operar otra. Comprar una computadora era realizar un compromiso a largo plazo con el fabricante, y por eso las pequeñas empresas no podían competir contra compañías como IBM, que podían ofrecer profesionales capacitados, mantenimiento y muchos otros servicios al comprador de sus equipos.

2.2.4 Los lenguajes de programación

Las primeras computadoras no venían con software para el usuario final, sino que solamente permitían ser programadas. Es decir, todo usuario de la computadora debía primero transformarse en un programador. El proceso de programar implicaba conocer la arquitectura del equipo, y pasar las ideas que tuviera el usuario a complejos códigos en tarjetas perforadas (es decir, había que aprender en que lugares perforar para lograr el comportamiento deseado del equipo, algo que

Ken Thompson (sentado) y Dennis Ritchie (parado), creadores de UNIX, trabajando en Bell Labs en una PDP-11.
Fotografía de Peter Hamer.

además variaba de máquina a máquina).

Los primigenios programadores, debían lidiar constantemente con la incompatibilidad de los programas que escribían. Así, un científico en Harvard no podía simplemente darle su programa a alguien en Princeton, pues las computadoras eran distintas, y por tanto, también los programas que se escribían para ellas. Tampoco era sencillo actualizar una computadora simplemente comprando una nueva, pues los programas ya escritos dejaban de funcionar.

Un programa escrito en tarjetas perforadas. Las marcas en la parte superior de la pila de cartas permitía identificar el programa y visualizar sus diversas partes.
Fotografía de Arnold Reinhold.

Con la intención de solventar estos problemas, y reducir adicionalmente los costos asociados a la programación de las computadoras, es que aparecieron los primeros **lenguajes de programación**. Estos permitían programar la computadora mediante un conjunto de instrucciones fáciles de recordar por los programadores.[3]

A medida que avanzaba la técnica, también avanzaban los lenguajes de programación. Comenzaron a crearse lenguajes que funcionaban en múltiples computadoras. Así, el usuario podía escribir un programa en FORTRAN o COBOL, que funcionara en todos los equipos que soportaran dicho lenguaje de programación. Esto era un gran avance, no solo en términos de expansión del conocimiento, sino también en la reducción de tiempos, y por tanto costos, en los procesos de desarrollo. Adicionalmente, el desarrollo de software dejaba gradualmente de ser un trabajo solo de científicos y matemáticos, para pasar a ser algo que

podía realizar cualquier persona con algo de formación.

Los lenguajes más avanzados permiten a los programadores expresar mejor sus ideas, y cuanto menor el trecho entre la idea y la aplicación en el equipo, menor el tiempo que toma llevar la idea adelante.

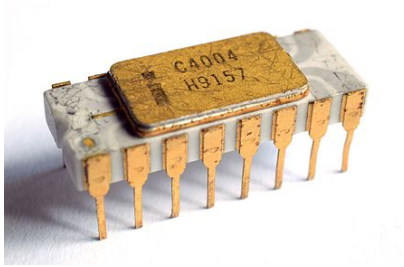
2.2.5 La revolución de los transistores y los circuitos integrados

La verdadera revolución en el campo de las computadoras surge con la invención del **transistor**. Este dispositivo reemplazaba al tubo de vacío, siendo mucho más fiable, económico, duradero, pero por sobre todo, pequeño que el tubo.

Los transistores permitían disminuir significativamente el tamaño, el peso y la complejidad de una computadora. Además, a medida que avanzaba la técnica, los mismos se podían hacer más y más pequeños.

Sabías qué
El transistor moderno que utilizamos en nuestras computadoras ronda los 14nm (nanómetros). Además, se han creado exitosamente transistores más pequeños, de tan solo 5 y 6 nm, e incluso de 1nm (aunque son todavía experimentales).

Para tener una comparación, el virus del VIH mide 100nm, un glóbulo rojo aproximadamente 1000nm, y un grano de arena mide entre 63000nm y 200000nm.



Un procesador Intel 4004.
Fotografía de Thomas Nguyen.

En 1959 el ingeniero **Jack S. Kilby** mientras trabajaba para la empresa de tecnología **Texas Instruments** desarrolló el primer **circuito integrado** (también conocido como **chip** o **microchip**). Consistía en la integración de seis transistores en un mismo dispositivo. En el año 2000 Kilby fue galardonado con el Premio Nobel de Física por la enorme contribución de su invento al desarrollo de la tecnología.[24]

La capacidad de producción masiva de circuitos integrados, su confiabilidad y la facilidad de agregarles complejidad, llevó a su estandarización, reemplazando circuitos completos con diseños que utilizaban transistores discretos, y además, llevando rápidamente a la obsolescencia a las válvulas o tubos de vacío.

Definición 2.3 Un **chip** o circuito integrado (CI) o microchip, es una estructura de pequeñas dimensiones de material semiconductor, normalmente silicio, sobre la que se fabrican circuitos electrónicos mediante fotolitografía y que está protegida dentro de un encapsulado de plástico o de cerámica.

Gracias a este avance, las computadoras ya no requerían enormes cuartos para su instalación, ni pesaban toneladas. Asimismo, el bajo costo de los chips volvía a la computadora un dispositivo asequible por "cualquier" persona (seguían siendo equipos caros, pero el costo ahora era equiparable al de un vehículo automotor).

2.3 La modernidad de las computadoras

En la década de 1970 surge un quiebre en la historia de las computadoras. La invención del chip, como elemento de bajo costo, hace que entusiastas y empresas sean capaces de armar sus propios equipos.

Con la reducción de costos, la estandarización del código gracias a lenguajes de programación, y una economía en auge, decenas de empresas comenzaron a producir computadoras (o en su defecto componentes y software para computadoras). Muchas de esas empresas hoy son líderes en la industria informática.

Por otro lado, el desarrollo de ideas democratizadoras de tintes socialistas llevó al desarrollo de software por parte de comunidades auto-convocadas, sin fines lucrativos, sino sociales.

Todos estos elementos son los que llevan al estado actual de las computadoras y de la informática, transformando a estos dispositivos en una parte fundamental de las sociedades modernas.

2.3.1 Las primeras microcomputadoras

A mediados de la década de 1970, comenzaron a aparecer las primeras **microcomputadoras**, equipos de tamaño sumamente reducido en comparación a las grandes computadoras que dominaban la escena de la época (aunque grandes para los estándares actuales).

La primer microcomputadora fue la japonesa **SMP80/08** de Sord Computer Corporation (1972), a lo que siguió la francesa **Micral N**. En 1974 aparece en los Estados Unidos el **Altair 8800**, que era vendido como un conjunto que debía ser ensamblado por el usuario.

> **Sabías qué**
> La Altair 8800 solamente contaba con unos pocos interruptores e indicadores lumínicos, y no tenía ni pantalla, ni teclado, ni nada parecido.

La revolución de los circuitos integrados y sus reducidos costos hizo que comenzaran a aparecer clubes de entusiastas y amantes de la electrónica y la tecnología, que se juntaban para compartir piezas, intercambiar planos, conocimiento, diseños y software. Aparece en esta época el **microprocesador**. Sumamente económicos, consistían en un complejo circuito integrado que cumplía múltiples funciones, antes realizadas por varios chips. Los **Intel 8080**, los **Z80** y los **Motorola 6800** serían los procesadores que potenciarían toda un generación de computadoras y consolas de videojuegos, transformándose en emblemas de la era.[3]

> **Definición 2.4** Un **microprocesador** es el circuito integrado central más complejo de un sistema informático. Es el encargado de ejecutar los programas, desde el sistema operativo hasta las aplicaciones de usuario. Puede contener una o más unidades centrales de procesamiento (CPU).

El **Homebrew Computer Club**, uno de estos clubes de entusiastas más influyentes y conocidos surgió en 1975 en **Silicon Valley, California**. Fue considerado como "el crisol de una industria entera", pues de allí surgieron influyentes personajes, e incluso empresas. En particular, **Apple** surge de los trabajos realizados en ese club por parte de **Steve Jobs** y **Steve Wozniak**.

También sería el lugar a donde **Bill Gates**, fundador de **Microsoft** enviaría una famosa carta justificando la privatización y comercialización del software mediante el actual modelo de licencias. Gates alegaba que el software no debía ser compartido (práctica habitual en esa época) sino que

Steve Jobs (parado) y Steve Wozniak (sentado) miembros del Homebrew Computers Club diseñando su primer computadora, la Apple I.
Fotografía de Apple Computer.

los usuarios debían pagar un permiso (licencia) para tener derecho a su uso. La carta fue tomada con gran aceptación y permitió que el desarrollo de software se transformara en una industria millonaria.

En los 80s llega el boom de las computadoras comerciales y decenas de microcomputadoras aparecen en el mercado, como la **Apple II**, la **Commodore 64**, la **BBC Micro**, o **TRS 80**. También aparecen en la escena las primeras consolas de videojuegos, como la **Atari 2600**, la **ColecoVision**, la **Intellivision** o la **Magnavox Odyssey**.

El público comenzaría a acostumbrarse a estos dispositivos, y a demandar mayor producción y más características. Las computadoras pasarían a ser un elemento fundamental en muchas industrias y negocios, así como un valioso recurso en instituciones educativas.

2.3.2 Los primeros sistemas operativos

La mayoría de las primeras microcomputadoras no permitían al usuario hacer otra cosa que no fuera de programar. La mayoría traía incluido un "manual de usuario" que enseñaba a los usuarios a programar en el equipo, el cual, generalmente incluía el lenguaje de programación BASIC. Esto hacía que los usuarios requirieran aprender una serie de conocimientos para nada sencillos solo para poder operar la unidad. Eso si, los conocimientos aprendidos permitían realizar cualquier cosa con el equipo, librado solo a la imaginación del usuario y a las capacidades técnicas del hardware.

CP/M corriendo sobre un monitor de fósforo verde en una Sanco 8001.
Fotografía de Mspetch.

Sin embargo, este modelo traía aparejado la problemática de un bajo número de usuarios, y una resistencia por parte de empresas e instituciones de adoptar computadoras debido al alto costo de capacitación del personal. Con la intención de solventar ese problema surge la idea de los **sistemas operativos**.

Un **sistema operativo** consistía en un programa que corría de forma constante en la computadora, y permitía al usuario escribir instrucciones que requiriera de la máquina, y esta las llevaría adelante. Así, un usuario podía solicitarle que realice un cálculo, ejecute un juego, o lo que requiriera del equipo, sin necesidad de aprender a programar. Esto requería por supuesto que la computadora incluyera software capaz de llevar adelante muchas tareas genéricas. Así, aparecen los primeros programas como procesadores de documentos, planillas de cálculo y bases de datos.[3]

En las grandes computadoras que existían en las universidades y entidades gubernamentales, el sistema operativo por defecto era **UNIX**, creado por **Bell Labs (AT&T)**. El sistema se distribuía de forma libre en un inicio, pero luego AT&T decidió comenzar a vender el producto. Esto llevó a que algunas universidades buscaran reemplazar UNIX con otra cosa.

En las microcomputadoras aparecen sistemas como **CP/M**, creado por **Gary Kildall**, y **DOS** (Disk Operating System), que en realidad no es un sistema operativo, sino un conjunto de sistemas, pues cada fabricante solía crear el suyo propio, y comercializarlo junto con su computadora (Y no eran necesariamente compatibles entre si). Tal vez el más popular sea de estos últimos sea **MS-DOS** de **Microsoft**. **MS-DOS** supo ganarse al mercado gracias a que era el sistema por defecto de las computadoras **PC** (Personal Computer). PC era una gama de modelos de computadoras vendidas por **IMB**, de gran éxito comercial por poseer **partes fácilmente intercambiables**, y por licenciar el diseño de componentes a diversas empresas que creaban clones económicos de muchos componentes.

Apple lanzaría en 1984 la **Apple Macintosh**, un modelo de computadora que fue el primero en venir de serie con un sistema operativo que incluía un entorno gráfico y mouse. Si bien no fue el primer sistema operativo gráfico (ya se habían hecho varios experimentos en diversos sistemas), al incluirlo de serie, puso la idea en el aire de que ese era el futuro, y pronto fue copiado por otros. Por su parte, la primera versión de **Windows**, el sistema operativo de **Microsoft** que hoy sigue siendo comercializado, vería la luz en 1985.

Sabías qué

Apple tomó la idea del mouse y de la interfaz gráfica de experimentos que se realizaban en el Centro de Investigación de Palo Alto de Xerox, conocido como **Xerox PARC**. Este centro fue la cuna de la computadora personal como hoy la conocemos, además de otros inventos como el mouse, la interfaz gráfica, la metáfora de escritorio, la impresión láser, las redes ethernet, la programación orientada a objetos, la computación ubicua, entre otras tantas cosas.[25]

Para mediados de los 80 prácticamente no se comercializaban computadoras que no tuvieran un sistema operativo pre-instalado. Algunas venían incluso con más de uno, permitiendo al usuario elegir su sistema de preferencia. Ningún sistema mantenía la hegemonía total, por lo que todavía había un mercado para el desarrollo de nuevos sistemas operativos. Así surgen sistemas como **Amiga OS**, o **BeOS**, entre otros.

2.3.3 El Software Libre

Una consecuencia directa de la masificación del uso de computadoras fue la **privatización del software**. El software pasa a considerarse una obra intelectual, de forma similar a un libro, una canción o una película. Es decir, **el software pasa a tener derechos de autor (copyright)**.

Ademas, por las características técnicas del software, tales como la posibilidad de replicar el mismo en tantas máquinas como se quiera, o de utilizarlo con fines que al autor no considera apropiados, hacen que el software no se comporte como los libros, música o películas de la época.

Así, **el software no se vende, se licencia**. Es decir, uno compra la posibilidad de utilizar el software de una forma determinada (una cantidad específica de equipos, usarlo con determinados fines, compartirlo de determinada forma, etc.).

Definición 2.5 Una **licencia de software** es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribución) y el licenciatario (usuario consumidor, profesional o empresa) del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Esta práctica pasa a ser el estándar en el desarrollo de software de la época. Sin embargo, en 1985, un programador llamado **Richard Stallman**, cansado de problemas asociados al copyright del software, inicia un movimiento que involucra tanto ideas de izquierda como cuestiones técnicas asociadas al software, el **movimiento de software libre**. Poco tiempo después fundaría la **Free Software Foundation** y el **proyecto GNU**, con la intención de promover la creación y el uso del software libre.[26]

Stallman acuña el término **copyleft**, como una alternativa al **copyright**. En este caso, una obra intelectual sigue teniendo derechos de autor, pero el autor manifiesta al momento de su creación de forma explicita su voluntad de permitir a otros utilizar su obra para lo que sea, siempre y cuando esta se mantenga libre (es decir, que los derivados de dicha obra compartan la misma característica de poder ser utilizados como se desee).

Richard Stallman en una charla en Madrid (2016).
Fotografía de Rubén Ojéda.

Esto supone un cambio radical en la industria del software. Involucra el compartir el **código fuente** de los programas, y permitir que cualquiera trabaje sobre el. Stallman sostiene que esto permite que la sociedad en su conjunto se beneficie de las mejoras que se vayan realizando.

Las empresas en principio juzgaron fuertemente al movimiento, pero las bondades del mismo pronto se hicieron evidentes. Al tener cientos de programadores que participan de forma voluntaria en un producto, el mismo termina por contar con mayor calidad y siendo más seguro. Así, aparece la **Open Source Initiative**, una iniciativa que comparte las prácticas del movimiento del software libre, pero, cuyos fines no son los mismos; mientras el primero persigue fines altruísticos, el segundo solo busca una mejora en la rentabilidad y beneficios comerciales.

Se dice en general que el software es **libre** si sus creadores adhieren a las políticas propuestas por la **Free Software Foundation**, y que es **open source** si simplemente abren su código pero no comparten la filosofía. El resto del software se denomina **privativo**. En muchas ocaciones no se realiza la distinción entre los términos libre y open source, considerando a todo el software de código abierto como libre.

Hay que tener en cuenta que el software libre es general, pero no necesariamente, gratuito. Esto es uno de los principales incentivos para los usuarios, pero a esto se le debe adicional la confiabilidad en el software, pues su código ha sido visto por muchos desarrolladores que no tienen interés comercial en el mismo de forma directa, y que denunciarían cualquier acción que pudiera perjudicar al usuario final.

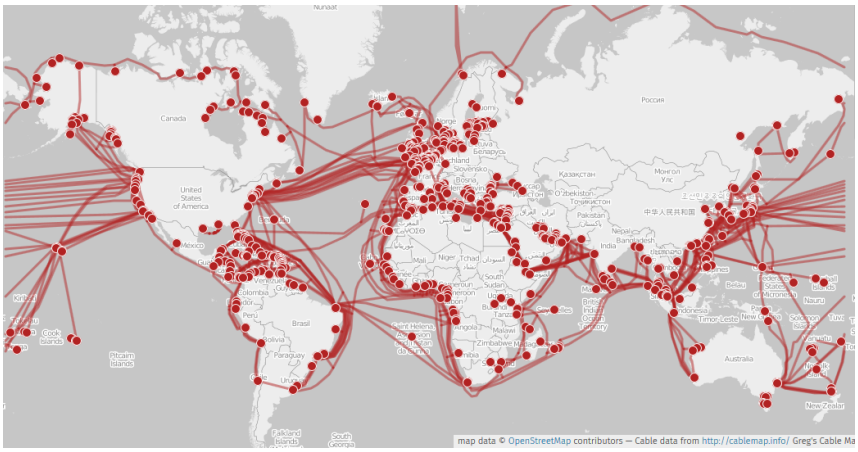
Hoy en día, la mayoría de los programas comerciales cuentan con una alternativa “libre”. En muchas ocaciones, la alternativa libre es superior a la “privativa”, pero en otras ocaciones, la falta de financiamiento y prácticas monopólicas hacen que las alternativas libres no estén a la altura de sus contrapartes comerciales.

Este movimiento se ha vuelto tan grande que incluso empresas como Microsoft o IBM ahora desarrollan y apoyan el software libre u open source. Muchas empresas se dedican incluso a la venta de servicios de soporte o desarrollo de soluciones de software libre, por lo que el software libre no representa una amenaza a la industria del desarrollo de software.

2.3.4 Las redes, Internet, y la globalización

No tardó mucho tiempo en que los usuarios quisieran compartir información entre computadoras ubicadas en distintos lugares físicos. Ya en 1950, el el sistema de radares militares de Estados Unidos, conocido como **SAGE**, se puso en línea, conectando computadoras de diversos puntos del país. Por su parte, en la Unión Soviética en 1959 se propone un plan de defensa que incluye una red de computadoras ubicadas en lugares estratégicos.

Sin embargo, el verdadero hito en materia de redes se produciría en 1969, con la creación de **ARPANET** (Advance Research Projects Agency Network). Nacida como un proyecto del Departamento de Defensa de los Estados Unidos, ARPANET sería la primer red en utilizar un protocolo estandarizado para comunicar diversas computadoras. Empezó uniendo los equipos de diversas universidades en la costa oeste, para expandirse velozmente hacia la costa este y centro del país.



Mapa de las conexiones submarinas de Internet.
Datos de Greg Mahlknecht sobre mapa de OpenStreetMap.

En Europa, diversas organizaciones generaban sus propias redes, algunas de las cuales colaboraban con la ARPANET adoptando sus protocolos, como **NORSAR** en Noruega, o redes en las universidades de Inglaterra. ARPANET evolucionaría rápidamente para convertirse en lo que hoy llamamos **Internet**.[27]

Sabías qué
Internet se basa en una red de computadoras conectadas por cables. Hay cables submarinos que conectan todos los lugares del mundo, así como cables enterrados bajo la tierra en diversos países que no limitan con algún océano.

Muchas personas creen que Internet funciona mediante conexiones satelitales. Si bien es cierto que existen conexiones de estas características, las velocidades de conexión y los altos costos de mantener ese servicio, lo vuelven inviables para un servicio tan masivo.

2.4 La actualidad y el futuro

Avanzamos mucho desde esas primeras microcomputadoras. Hoy, es normal que en los países desarrollados y emergentes la mayoría de la población cuente con una computadora. Además, la penetración de internet en la población de dichos lugares suele ser elevada o encontrarse en constante aumento, tal es el caso de Argentina.

Sabías qué
En algunos países el acceso a internet es considerado un derecho constitucional. En esos lugares entienden que el acceso a la información debe ser irrestricto, y por tanto, el estado debe garantizar alguna forma de que las personas accedan a Internet.

Los dispositivos se han reducido en tamaño y aumentado en potencia, pero siguen basados en los mismos principios que las microcomputadoras de los 70. Un celular moderno es incluso cientos de veces más potente que esas primigenias computadoras que, no obstante, llevaron al hombre a la luna.

Nuevas tecnologías como el GPS, las redes inalámbricas, las conexiones de datos móviles, entre otras, nos posicionan en mundo donde la tecnología es ubicua. Además, nuevos avances, aún en desarrollo, pero ya una realidad, nos plantean oportunidades, a la vez que desafíos de cara a un futuro donde la tecnología estará cada vez más integrada en nuestro día a día.

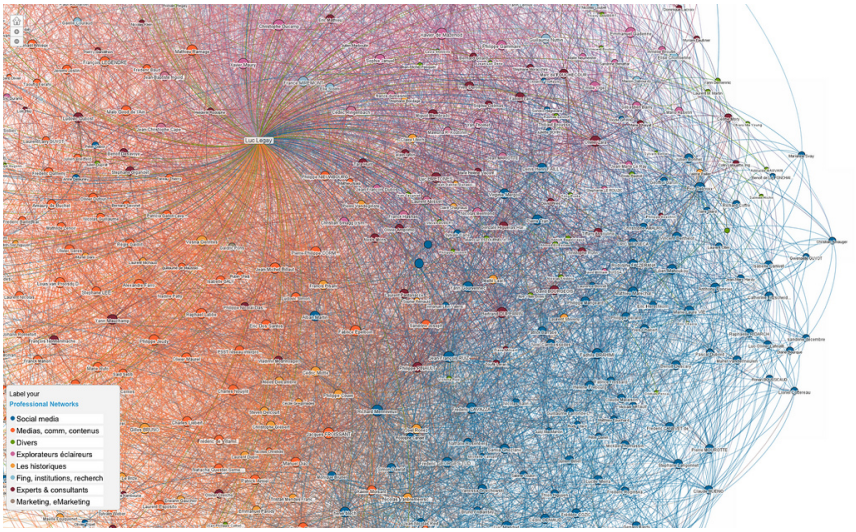
La concepción de la información como un elemento fundamental del universo, inherente a la física o a la biología, plantea nuevos modelos computacionales que presentan inicialmente cambios radicales en la forma en la que pensamos en computación, así como también auguran avances en áreas como la física, la astronomía o la medicina.

2.4.1 La internet de las cosas y Big Data

Uno de los escenarios en donde hay cada vez más desarrollo y más trabajos en ciencias de la computación tiene que ver con el concepto de **"Internet de las cosas" (IoT)**. Es un concepto que se refiere a una interconexión digital de objetos cotidianos con internet.[28]

Esto implica que virtualmente todo objeto tenga algún tipo de identificador de radiofrecuencia que le permita al dispositivo conectarse a Internet. Con objeto nos referimos a autos, heladeras, tostadoras, lámparas, libros, paquetes, envoltorios, etc. Virtualmente todo. Kevin Ashtor, quien acuñó el término expresa los siguiente:

"Las computadoras actuales —y, por tanto, internet— son prácticamente dependientes de los seres humanos para recabar información. Una mayoría de los casi 50 petabytes de datos disponibles en internet fueron inicialmente creados por humanos, a base de teclear, presionar un botón, tomar una imagen digital o escanear un código de barras. Los diagramas convencionales de internet, dejan fuera a los routers más importantes de todos: las personas. El problema es que las personas tienen un tiempo, una atención y una precisión limitados, y no se les da muy bien conseguir información sobre cosas en el mundo real. Y eso es un gran obstáculo. Somos cuerpos físicos, al igual que el medio que nos rodea. No podemos comer bits, ni quemarlos para resguardarnos del frío, ni meterlos en tanques de gas. Las ideas y la información son importantes, pero las cosas cotidianas



Visualización de las conexiones del usuario Luc Legay en la plataforma LinkedIn.
Imágen de Luc Legay.

tienen mucho más valor. Aunque, la tecnología de la información actual es tan dependiente de los datos escritos por personas que nuestras computadoras saben más sobre ideas que sobre cosas. Si tuviéramos computadoras que supieran todo lo que tuvieran que saber sobre las 'cosas', mediante el uso de datos que ellas mismas pudieran recoger sin nuestra ayuda, nosotros podríamos monitorizar, contar y localizar todo a nuestro alrededor, de esta manera se reducirían increíblemente gastos, pérdidas y costes. Sabríamos cuándo reemplazar, reparar o recuperar lo que fuera, así como conocer si su funcionamiento estuviera siendo correcto. El internet de las cosas tiene el potencial para cambiar el mundo tal y como hizo la revolución digital hace unas décadas. Tal vez incluso hasta más." Ashton [29]

La internet de las cosas no es un escenario futurista lejano, pues distintas estimaciones proyectan que para 2020 ya habrá aproximadamente unos 20 mil millones de dispositivos conectados, siendo la cifra actual de unos 9 mil millones.[30]

Este tipo de interacciones con dispositivos tendría implicancias prácticas en áreas como la **domótica** (automatización de hogares), el **marketing**, y la **logística**. Sin embargo, a medida que la tendencia crece surgen diversas preocupaciones y problemáticas. Por un lado, aspectos relacionados a la seguridad de estos dispositivos, pues se ha comprobado que muchos de ellos pueden ser vulnerados con facilidad, algo que sucedio incluso con una cámara de monitoreo para bebés, pudiendo el atacante tener acceso a la filmación. Por otro lado, la cantidad de datos a manejar representa un volumen significativo, lo que trae aparejados desafíos tanto a nivel de ingeniería (servidores, energía, etc.) como a nivel de desarrollo de software (manejando grandes volumenes de datos).

Así, esta área va de la mano con el desarrollo de **Big Data**, un concepto que hace referencia a conjuntos de datos tan grandes y complejos que requieren de aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente. El análisis de los datos permite elaborar complejas estadísticas o predicciones.[31]

Pero el big data no solo se produce a partir del internet de las cosas, sino que otros datos, por ejemplo, aquellos generados por seres humanos al utlizar tecnología también son sujetos de esta disciplina. Hoy en día empresas como Google o Facebook recolectan y venden una gigantesca

cantidad de datos de sus usuarios, que involucran sus gustos, sus conocidos, su ubicacion en distintos momentos, entre otras muchas cosas, y que se utilizan para determinar la publicidad más adecuada para cada segmento, así como para hacer predicciones sobre su comportamiento y consumo. No solo es publicidad, sino que también son utilizados esos datos para manipular las elecciones en diferentes paises.[32]

2.4.2 La inteligencia artificial y la automatización

Otra de las áreas en donde hay un especial interés dado su amplio potencial es en la **inteligencia artificial**. Coloquialmente, el término se aplica cuando una máquina imita las funciones "cognitivas" que los humanos asocian con otras mentes humanas, como por ejemplo: "aprender" y "resolver problemas". Técnicamente puede definirse como la capacidad de un sistema para interpretar correctamente datos externos, para aprender de dichos datos y emplear esos conocimientos para lograr tareas y metas concretas a través de la adaptación flexible.[33]

La inteligencia artificial permite **automatizar** procesos previamente realizados por humanos. Dentro de las aplicaciones prácticas ya destacan actualmente la participación de robots automátizados en procesos industriales, o de inteligencias artificiales en el mercado de valores, con resultados que superan ampliamente a los que podrían obtener cualquier grupo de seres humanos. También se utilizan sistemas de inteligencia artificial en procesos de filtrado de spam en correos electrónicos, o generación de listas musicales, así como en videojuegos. Hoy en día incluso hay empresas que ofrecen para empresas de noticias, servicios de inteligencia artificial con autómatas capaces de redactar notas, por ejemplo, de eventos deportivos, basandose solamente en las estadísticas del juego. También existen sistemas que permiten eliminar puestos de gerencia media, al identificar el trabajo realizado por estos y reemplazarlos por sistemas automáticos. Ya existen también, autos capaces de manejarse solos, y de tomar la mejor decisión de conducción ante situaciones inprevistas. Se espera cada vez más y más intervención de este tipo de tecnologías, en transporte, en madicina, en telecomunicaciones, en política, etc. Su incorporación será tan grande que se habla de una "**cuarta revolución industrial**", la cual cambiará significativamenteel panorama económico y laboral del planeta, al punto de que se estima que el un alto porcentaje de la población quedará sin empleo, aunque también se pronostica la aparición de nuevos puestos de trabajo asociados a estas tecnologías.[34]

2.4.3 La computación cuántica y la computación biológica

El último area de interés que mencionaremos está relacionado a nuevos modelos computacionales que vienen a romper el paradigma de computadora actual. En ese sentido hay dos grandes modelos que llenan de emoción a los cientificos.

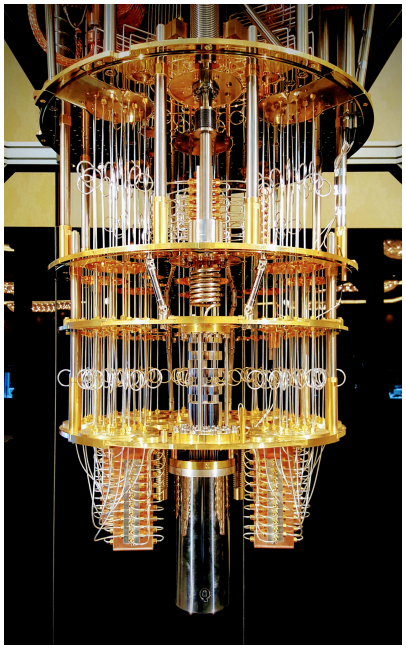
El primero, la **computación cuántica**, sería capaz de resolver de forma eficiente algunos problemas que no pueden resolverse con computadoras clásicas, permitiendo la implementación de nuevos algoritmos. Los avances tanto en la parte formal de esta ciencia, así como en la ingeniería de la misma, auguran un gran futuro para este tipo de computadoras. Las mismas utilizan principios de la física cuántica, tales como la superposición o el enlazamiento cuántico para manipular información, la cual está codificada como alguna propiedad cuántica de una particula (por ejemplo la polarización de un fotón, o el spin de un electrón).[35]

Las primigenias computadoras cuánticas ya están entre nosotros. Compañías como IBM, Microsoft y Google ya tienen desarrollos sobre está tecnología. Sin embargo, todavía son propensas a errores, y lo que ofrecen puede realizarse mediante simulaciones en computadoras clásicas, por lo que todavía no ofrecen mejoras sustanciales. Sin embargo, la tecnología avanza rapidamente y se

espera que en pocos años pasen a ser una realidad.[36]

El segundo modelo es el de las **computadoras biológicas**. Este tipo de computadoras se hacen posibles gracias a la nanobiotecnología, y permitirían manipular proteínas y ADN para realizar calculos computacionales. Este tipo de computadoras permitiría manipular elementos biologicos, generando complejas interacciones, lo cual podría resultar en nuevos medicamentos o tratamientos para enfermedades.[37] No solo eso, sino que la capacidad de los sistemas biologicos de autoreplicarse permitiría crear computadoras con un costo sumamente bajo, lo cual sería una ventaja económica sustancial frente a sistemas que requieren procesos manuales para su fabricación.

Estos nuevos modelos computacionales surgen de la combinación de las ciencias de la computación con otras disciplinas (física y biología respectivamente). Este enfoque nos demuestra que las ciencias de la computación están todavía en una etapa embrionaria, pues la interpretación de cualquier elemento como "información" da como resultado estas tecnologías que nos obligan a repensar seriamente las posibilidades y los límites de la computación. Por esto muchos cientificos se emocionan con las posibilidades que auguran estas nuevas tecnologías, aunque muchas de ellas sean todavía teóricas.



Parte de la computadora cuántica Q de IBM.
Fotografía de Lars Plougmann.

2.5 Actividades

Esta sección presenta una serie de ejercicios y preguntas para que realice. En este caso, se pide investigue en Internet para poder contestar las preguntas a continuación:

Ejercicio 2.1 ¿Cuál fue la primer computadora en argentina? ¿Para que se usaba? ¿Quién la trajo al país? ■

Ejercicio 2.2 ¿Qué es la ley de Moore? ¿Se cumple actualmente? ■

Ejercicio 2.3 ¿De quién son los cables submarinos de Internet? ■

Ejercicio 2.4 Sabía que en Argentina se desarrollaron varias distribuciones de Linux. Averigue el nombre y la historia de algunas de ellas. ■

Ejercicio 2.5 ¿Conoce alguna de las microcomputadoras emblemática? ¿Cuál? Si no conoce ninguna, investigue que computadoras marcaron esa época. ■

Ejercicio 2.6 ¿Qué lenguajes de programación conoce? ¿Sabe cuales son los más populares? ■

Referencias

[1] Dan Gookin. *PCs For Dummies*. 10ª ed. For dummies. Hobooken, Nueva Jersey: Wiley, 2005 (véanse páginas 13, 14, 20, 26).

[2] J Clark Scott. *But How Do It Know? - The Basic Principles of Computers for Everyone*. John C. Scott, 2009 (véase página 13).

[3] Paul E. Ceruzzi. *A History of Modern Computing*. 2ª ed. London, England; Cambridge, Massachusetts: MIT Press, 2003 (véanse páginas 14, 35, 39, 41, 43).

[4] Dan Gookin. *Buying a PC For Dummies*. 2006 ed. For dummies. Hobooken, Nueva Jersey: Wiley, 2006 (véase página 15).

[5] Philip A. Laplante. *Dictionary of Computer Science, Engineering and Technology*. CRC Press, 2000 (véanse páginas 15-17, 21).

[6] Doug Lowe. *Networking for Dummies*. 7ª ed. For dummies. Wiley, 2004 (véase página 19).

[7] Douglas Downing y col. *Dictionary of computer and Internet terms*. 10ª ed. Barron's business guides. Hauppauge, Nueva York: Barron's Educational Series, 2009 (véanse páginas 20, 21).

[8] Firmware. *Merriam-Webster's dictionary*. `https://www.merriam-webster.com/dictionary/firmware`, 2018 (véase página 21).

[9] Andrew S. Tanenbaum. *Modern Operating System*. 2ª ed. Prentice Hall, 2007 (véase página 22).

[10] Andrew S. Tanenbaum y Albert S. Woodhull. *Operating Systems: Design and Implementation*. 3ª ed. Prentice Hall, 2006 (véase página 22).

[11] Thomas M. Lenard. «Competition, Innovation and the Microsoft Monopoly: Antitrust in the Digital Marketplace». En: *Proceedings of a conference held by The Progress & Freedom Foundation in Washigton, DC February 5, 1998*. Editado por Thomas M. Lenard y Jeffrey Eisenach. Springer Netherlands, 1999 (véase página 23).

[12] Aplicación. *Diccionario de la lengua española*. Madrid, España, 2004 (véase página 24).

[13] Jonas Bogoshi, Kevin Naido y John Webb. «The oldest mathematical artifact». En: *The Mathematical Gazette* 71.458 (1987), página 294 (véase página 30).

[14] Carl B. Boyer y col. *A history of mathematics*. 2ª ed. Wiley, 1991 (véase página 30).

[15] Jian-Liang Lin y Hong-Sen Yan. *Decoding the Mechanisms of Antikythera Astronomical Device*. Springer, 2016 (véase página 30).

[16] Clair L. Evans. *Broad Band: The Untold Story of the Women Who Made the Internet*. Penguin, 2018 (véase página 31).

[17] George C. Chase. «History of Mechanical Computing Machinery». En: *IEEE Annals of the History of Computing* 2.3 (1980), páginas 198-226 (véase página 32).

[18] Fuegi J. y Francis J. «Lovelace & Babbage and the creation of the 1843 'notes'». En: *IEEE Annals of the History of Computing* 25.4 (oct. de 2003), páginas 16-26 (véase página 33).

[19] Charles Petzold. *Annotated Turing*. Wiley, 2008 (véase página 34).

[20] Paul E. Ceruzzi. *Computing: A Concise History*. Cambridge, Massachusetts: MIT Press, 2012 (véase página 36).

[21] David A. Patterson y John L. Hennessy. *Computer Architecture: A Quantitative Approach*. 3.ª edición. Morgan Kaufmann, 2002 (véase página 37).

[22] Emerson W. Pugh. *Building IBM: Shaping an Industry and Its Technology*. London, England; Cambridge, Massachusetts: The MIT Press, 1996 (véase página 38).

[23] John Gertner. *The Idea Factory: Bell Labs and the Great Age of American Innovation*. Penguin, 2013 (véase página 38).

[24] Reid T.R. *The Chip : How Two Americans Invented the Microchip and Launched a Revolution*. HarperBusiness, 2001 (véase página 40).

[25] michael A. Haltzik. *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age*. HarperBusiness, 2000 (véase página 43).

[26] Peter H. Salus. *The Daemon, the Gnu, and the Penguin*. Reed Media Services, 2008 (véase página 44).

[27] Katie Hafner y Mathew Lyon. *Where Wizards Stay Up Late: The Origins Of The Internet*. Simon & Schuster, 1998 (véase página 45).

[28] Christian Floerkemeier y col., edicíon. *The Internet of Things: First International Conference, IOT 2008, Zurich, Switzerland, March 26-28, 2008. Proceedings*. Lecture Notes in Computer Science 4952 : Information Systems and Applications, incl. Internet/Web, and HCI. Springer-Verlag Berlin Heidelberg, 2008 (véase página 46).

[29] Kevin Ashton. *That 'Internet of Things' Thing*. https://www.rfidjournal.com/articles/view?4986. Jul. de 2009 (véase página 47).

[30] Gartner Press. *Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016*. https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016. Egham, U.K., feb. de 2017 (véase página 47).

[31] Pete Warden. *Big Data Glossary*. O'Reilly Media, 2011 (véase página 47).

[32] «Facebook: Cambridge Analytica habría manipulado las elecciones en Argentina». En: *Perfil* (mar. de 2018). URL: https://www.perfil.com/noticias/internacional/facebook-cambridge-analytica-habria-manipulado-las-elecciones-en-argentina.phtml (véase página 48).

[33] Andreas Kaplan y Michael Haenlein. «Siri, Siri in my Hand, who's the Fairest in the Land? On the Interpretations, Illustrations and Implications of Artificial Intelligence». En: *Business Horizons* 62 (2019), páginas 15-25 (véase página 48).

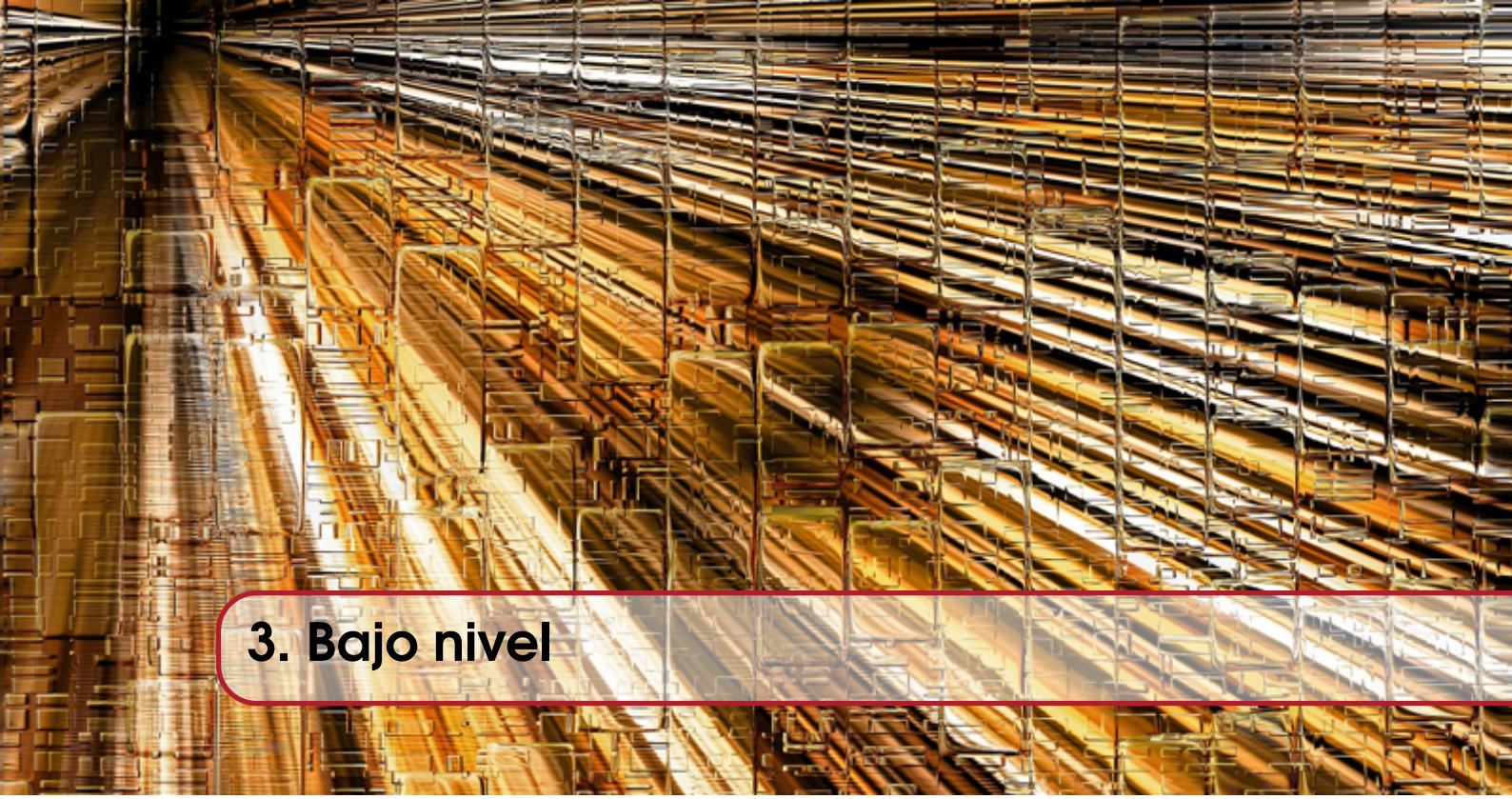
[34] Klaus Schwab. «The Fourth Industrial Revolution: what it means, how to respond». En: *World Economic Forum* (ene. de 2016). URL: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/ (véase página 48).

[35] Michael A. Nielsen e Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary. Cambridge University Press, 2011 (véase página 48).

[36] Will Knight. «Serious quantum computers are finally here. What are we going to do with them?» En: *MIT technology Review* (feb. de 2018). URL: https://www.technologyreview.com/s/610250/serious-quantum-computers-are-finally-here-what-are-we-going-to-do-with-them/ (véase página 49).

[37] Robert A. Freitas. *Nanomedicine Volume I: Basic Capabilities*. Austin, Texas: Landes Bioscience, 1999, páginas 349-351 (véase página 49).

II Información



3. Bajo nivel

Fotografía artistica que representa información.
Fotografía de Pixabay.

En este capítulo analizaremos como la computadora transmite y almacena información, y como procesa los datos que tiene almacenados. También veremos como la concepción de "información", independientemente del tipo de información que se trate y que tan compleja sea, puede abstraerse a una interpretación sencilla capaz de ser procesada rápida y eficazmente por una computadora.

También veremos como las computadoras solamente trabajan con electricidad, y como esta puede ser utilizada para representar todo tipo de información mediante una codificación.

3.1 Representación de Información

¿Cómo hacen las computadoras para hacer todo lo que hacen? ¿Dónde guardan las cosas? ¿Cómo sáben? Son preguntas que en general el usuario promedio de computadoras, no versado en la temática, suele hacerse en algún momento.

En esta sección intentaremos responder algunas de esas preguntas, analizando en primer lugar la pregunta "¿Qué es la información?" desde la prespectiva de una computadora. Tras contestar eso veremos como estas complejas máquinas pueden procesar esa información para realizar diversas tareas.

Esto nos llevará a sumergirnos aún más en el mundo del software y en su interacción con el hardware en el nivel más sencillo, para luego acender en un proceso espiral en los distintos niveles de abstracción que existen en el software.

3.1.1 Códigos y comunicación

Olvidemonos de las computadoras solo por un momento y empecemos por analizar un poco la comunicación humana. Digamos que queremos comunicarnos con alguien, por ejemplo, hablando. Nosotros seríamos entonces el **emisor** de la comunicación, y la persona con la que hablamos el **receptor**. Además, la comunicación se produce siempre mediante un **canal**, es decir, un medio

físico capaz de transferir la información desde el emisor hasta el receptor (en el caso del habla, el canal es el aire, capaz de propagar las ondas sonoras). Por el canal circula el **mensaje**, que es el elemento de la comunicación que se transfiere desde el emisor hacia el receptor (en el caso del habla, las ondas sonoras).

Ahora bien, no haga trampa y no se adelante en el texto, lea solo lo que tiene inmediatamente a continuación, y responda ¿Puede descifrar el siguiente mensaje alienigena?

Lo más probable es que la respuesta sea que no. Aquí va una pista, cada símbolo que aparece en el mensaje corresponde a una letra. ¿Puede descifrarlo ahora?. ¿Todavía no? Muy bien, la respuesta era:

¿Lo intentamos una vez más? Intente descifrar ahora este mensaje:

¿Lo logró? Debería haberle sido más sencillo. El mensaje era “amiga mía”. El truco estaba en buscar el mismo símbolo en el mensaje anterior, y luego ir reemplazando cada símbolo con la letra correspondiente.

Si colocáramos cada símbolo junto a su correspondiente letra, y lo hacemos para todas las letras del abecedario, entonces tendríamos una simple tabla que nos permitiría no solo leer cualquier mensaje que use esos símbolos, sino también escribir mensajes con los mismos.

Esta práctica es algo común que muchos realizamos cuando éramos chicos, como parte de un juego que nos permitía enviarle mensajes secretos a otra persona, sin que terceros (por ejemplo, nuestros padres, o las maestras) pudieran leerlos. El truco consistía en que la tabla de símbolos solo fuera conocida por nosotros y por la persona a quien le íbamos a enviar los mensajes.

Imaginemos ahora dos barcos que se cruzan en altamar en la noche. Los barcos se encuentran lo suficientemente lejos como para que el sonido de la voz de sus capitanes viaje de un barco a otro, pero sin embargo están lo suficientemente cerca como para ver las luces del otro barco. Así, si los capitanes desean comunicarse, pueden usar la luz (por ejemplo, de una linterna) como canal de comunicación. Un capitan puede enviar mensajes mediante una serie de parpadeos lumínicos que genera con su linterna (los cuales pueden ser largos o cortos) y que el otro capitan podría entender. Pero, para poder comunicarse efectivamente, ambos capitanes deben tener una serie de reglas o convenciones en común que le permitan saber que significan dichos parpadeos. Un parpadeo corto y uno largo, fácil, la letra “A”, uno corto, uno largo y luego dos cortos, la letra “L”.

Este sistema ideado por Samuel F.B. Morse en 1836, para ser utilizado en sistemas telegráficos.

En general las personas que utilizan código Morse no hablan de parpadeos lumínicos largos y cortos, sino de “rayas” y “puntos”, pues es una convención sencilla para escribir esos parpadeos. Si deseamos comunicarnos basta tener la siguiente tabla a mano:

letra	código	letra	código	letra	código
A	·—	J	·———	S	···
B	—···	K	—·—	T	—
C	—·—·	L	·—··	U	··—
D	—··	M	——	V	···—
E	·	N	—·	W	·——
F	··—·	O	———	X	—··—
G	——·	P	·——·	Y	—·——
H	····	Q	——·—	Z	——··
I	··	R	·—·		

En el caso de los mensajes alienigenas y el juego que realizabamos cuando chicos era que el sistema de reglas que usabamos para leer y escribir el mensaje es secreto, solo nosotros y alguien más lo conocía. En el caso del código Morse en cambio, el código no debe ser secreto, sino todo lo contrario (todos los capitantes de barcos deberían saberlo para que podamos comunicarnos efectivamente).[1]

Definición 3.1 Un **código** es un sistema de reglas que permiten convertir información dada por el emisor de un mensaje en alguna forma (letras, símbolos, gestos, parapedos lumínicos, sonidos u otros) en información relevante para el receptor (probablemente en otra forma).

Pero, que tiene todo esto que ver con las computadoras. Para eso debemos mantener esta idea en la cabeza unos minutos mientras analizamos un poco el hardware de la computadora.

3.1.2 Electricidad y cables

Las computadoras modernas son dispositivos principalmente electrónicos, por tanto solamente pueden manipular la electricidad que fluye por sus cables (principalmente de cobre, como los de las instalaciones electricas, pero de tamaños mucho más reducidos y con mucha menos carga eléctrica). La computadora posee una serie de dispositivos que bloquean o no el paso de corriente eléctrica en determinados casos (similar a un interruptor eléctrico, pero que no requiere de intervención mecánica) llamados transistores. También existen en la computadora otra serie de dispositivos electrónicos como resistencias, capacitores, osciladores, etc. Estos se combinan de intrincadas formas para permitir o no el paso de electricidad en determinados cables ante ciertas condiciones, formando lo que conocemos como **circuitos**.

Definición 3.2 Un **circuito electrónico** consiste en un conjunto de componentes electrónicos, tales como resistencias, transistores, capacitores, inductores y diodos, conectados mediante cables o trazas conductivas a través de los cuales fluye corriente eléctrica.

Todas las cuentas y cálculos que la computadora realiza, todas las imágenes que se muestran en pantalla, todo es solamente electricidad. Así **los cables de la computadora son el canal por el que fluye la comunicación**, mientras que **la electricidad que fluye por los cables conforma el mensaje**.

Si queremos comunicarnos con una computadora debemos pensar en el proceso comunicacional

que ocurre. El emisor y el receptor alternan entre usuario y computadora, ya sea que el usuario ingrese datos a esta, esta le brinde información al usuario. Más importante es pensar en el canal y en el mensaje de dicha comunicación.

El único canal existente son los cables que mencionamos, y el mensaje solo puede estar dado por pulsos electricos. Muy similar al ejemplo de las linternas, donde solo teníamos "puntos" o "rayas", aquí solo tenemos electricidad enviada por un cable (o no enviada). Es decir, en un único cable, hay solo dos posibles mensajes en un mismo momento.

Podemos decir que nuestros mensajes están limitados al canal en donde se transmite. Si consideramos más cables, veremos que podemos enviar electricidad por algunos de ellos, mientras que no lo hacemos por otros, dando lugar a una mayor cantidad de combinaciones, que aumenta exponencialmente a medida que agregamos cables. Esto permite enviar combinaciones más complejas, y por tanto mayor cantidad de mensajes en un mismo momento. Nos centraremos más en esto en las siguientes secciones, pero tenga en cuenta el lector que en general, no nos interesa analizar un único cable, sino varios de ellos, pues es cuando se combinan de diversas formas que efectivamente se tiene información interesante.

Siguiendo con los circuitos, los mismos **pueden tener formas sumamente complejas que permiten realizar operaciones de lo más variado, como sumas, restas, multiplicaciones y demás**. De esta forma, la información que ingresamos puede ser procesada para transformarse en información que la computadora nos brindará luego.[2]

Uno de los principales problemas que tienen los circuitos hechos con transistores y cables, es que solo funcionan mientras hay electricidad. Si por ejemplo, se cortara el suministro eléctrico todo lo que hay en la computadora se pierde. Sin embargo, es deseable almacenar información para un uso posterior (por ejemplo, guardar una imagen o un video digital). Así, existen dispositivos que utilizan otros elementos para almacenar información, conocidos como **periféricos de almacenamiento**. Todos estos dispositivos son capaces de tomar la electricidad que circula por un cable especifico (conjunto de cables en realidad, aunque no nos centraremos aquí en el funcionamiento detallado de los dispositivos) y almacenarla de alguna forma tal que luego pueden leer y transformar nuevamente en pulsos eléctricos que viajan por dicho cable.

Los discos rígidos son un ejemplo de periférico de almacenamiento que utilizan una serie de imanes microscópicos que pueden estar orientados hacia alguna de dos direcciones posibles. El disco contiene un cabezal que puede leer la dirección a la que apunta un imán particular, o alterarla. Si lee una cierta dirección, el dispositivo es capaz de mandar un pulso eléctrico (usando energía que proviene de la fuente de alimentación) por alguno de los cables relevantes.

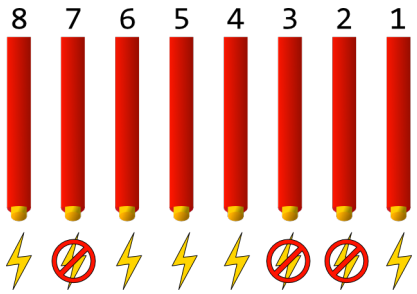
Los discos ópticos (CD/DVD) operan de forma similar, pero con una superficie que refracta la luz, un láser y sensores ópticos en lugar de imanes y un cabezal. Dependiendo del lugar hacia donde se refracte la luz, el dispositivo enviará pulsos electricos en determinados cables.

Las tarjetas perforadas hacen algo parecido con punciones sobre una superficie de cartón o papel, para que luego un cabezal intente pasar por un determinado lugar de la tarjeta, trasmitiendo electricidad si se encontraba perforada en ese lugar, pero bloqueando el paso de la misma si no lo estaba.

De esta forma, podemos decir que, **todo la información en una computadora (es decir, el software) no es más que electricidad pasando por cables, transistores y resistencias que se unen en formas extremadamente complejas, formando circuitos, que permiten procesar datos y comunicarse entre usuario y máquina**.[3]

3.1.3 Código Binario

Imaginemos un conjunto de ocho (8) cables. El pasaje o no pasaje de electricidad por dichos cables hace finalmente al mensaje. Probablemente nos interese mencionar en algún momento, por qué cables pasa electricidad y por cuales no, para determinar cuál es el mensaje que estamos, o bien enviando a la máquina, o bien recibiendo de ella. Podríamos graficar nuestros cables de la siguiente forma:

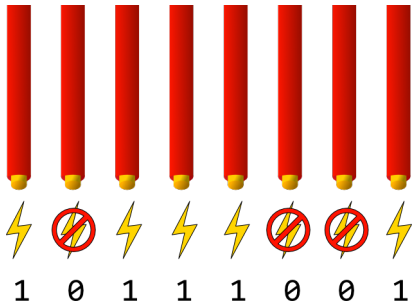


En este caso numeramos los cables del 0 al 7 de derecha a izquierda (podríamos haber numerado del 1 al 8 y de izquierda a derecha. La elegida es una convención arbitraria, pero útil en términos prácticos y por tanto, muy utilizada). Como se puede predecir, cuando la cantidad de cables aumenta de unos pocos a cientos, a miles, realizar un dibujo para indicar en qué cables pasa electricidad y en cuáles no deja de ser viable.

> **En realidad**
> En realidad, los cables siempre tienen electricidad, pero el voltaje que pasa por los mismos es distinto. Los circuitos se "activan" cuando el nivel de voltaje supera un umbral (por ej. 1.5 voltios), por lo que todo cable por el que pase menos voltaje se considera "apagado" y por el que pase más "encendido".

Así es como surge la idea de utilizar alguna especie de **código** para poder hablar del mensaje que se transmite por esos cables, sin tener que realizar complejos dibujos o digramas. Esto nos permite además abstraernos de los cables y circuitos, es decir, del canal, para centrarnos en el mensaje.

El código elegido para representar el estado de los cables internos de la computadora es el **código binario**, o lo mismo el **sistema binario**. El **sistema binario** es un sistema numérico en el que solamente se utilizan dos símbolos, **cero** y **uno** (nuevamente, una convención, cualquier otro par de símbolos, ej. "A" y "B" hubiera sido perfectamente válido). Así, un **número binario** se compone de una secuencia de ceros y unos (Por ej. 10100111). Como el sistema solo permite dos números, es ideal para representar los estados de los cables en la computadora que trabajan con presencia o ausencia de electricidad. Por lo tanto, podríamos representar el estado de nuestros ocho cables con un número.

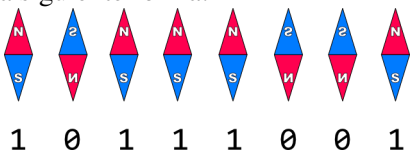


En este caso, elegimos que el dígito 1 representa la presencia de electricidad, mientras que el 0

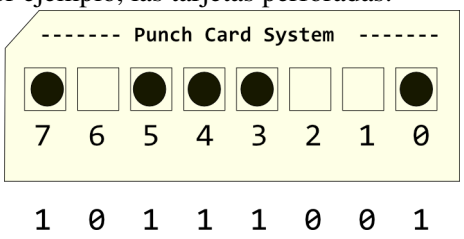
representa su ausencia, pero podría haberse elegido de forma inversa, es irrelevante, pues es solo una convención.

Definición 3.3 El **sistema binario** es un sistema numérico posicional de dos dígitos, los cuales son en general representados mediante cero y uno (0 y 1).[4]

Como ya vimos en la sección anterior, la electricidad que fluye por los cables puede eventualmente almacenarse en un dispositivo adicional, como un disco rígido, que utiliza imanes para representar la presencia o ausencia de electricidad. Si almacenamos en un disco, podríamos representarlo gráficamente de la siguiente forma:



Como vemos, el número binario que representa al valor almacenado es el mismo que el que representa a la electricidad en los cables. Podríamos ver que esto aplica también en otros dispositivos de almacenamiento, como por ejemplo, las tarjetas perforadas:



El sistema binario es lo suficientemente versátil para permitirnos hablar acerca de lo que vamos a almacenar o transmitir por un cable, sin importar el medio físico subyacente. Así, nos permite abstraernos del dispositivo y pensar acerca de la información que fluye en la computadora de forma matemática.

El sistema binario fue estudiado por primera vez en el siglo III A.C. por el matemático indio Pingala, coincidiendo con el descubrimiento del concepto del número cero. El matemático alemán Leibniz documentó en su totalidad el sistema en el siglo XVII, en su artículo “Explication de lÁrithmétique Binaire”. En 1854, el matemático británico George Boole publicó un artículo que marcó un antes y un después, detallando un sistema de lógica que terminaría denominándose “Álgebra de Boole”. La elección de este sistema para representar los datos responde entonces también a la gran cantidad de documentación y trabajos previos sobre dicho sistema numérico.

3.1.4 Codificación de información

Vamos a realizar un juego, similar al que hicimos dos secciones atrás. ¿Se anima a descifrar este mensaje?

72 79 76 65 32 77 85 78 68 79

Por supuesto que a priori es imposible identificar el mensaje, pero si seguimos la misma lógica de antes, podríamos asignar un número a cada letra, y obtendríamos una tabla que nos permita realizar la conversión de uno a otro. Es decir, sabemos que para poder entender ese mensaje, necesitamos un **código**. Dado el código en la siguiente tabla:

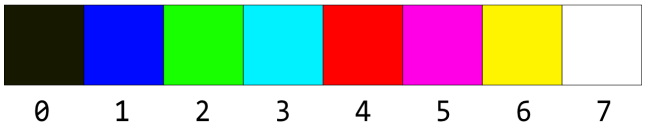
letra	número	letra	número	letra	número
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90
I	73	R	82	espacio	32

¿Ahora puede identificar el mensaje dado? Claro que si, es cuestión de reemplazar uno a uno los número por las letras dadas.

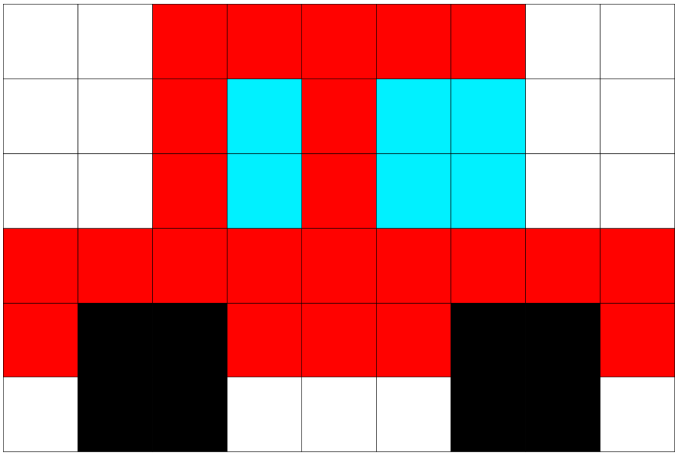
```
72 79 76 65 32 77 85 78 68 79
H  O  L  A     M  U  N  D  O
```

Imagínese ahora que la tabla no la tenemos solo dos personas, sino que es una tabla que todo el mundo conoce, y a la que todo el mundo tiene acceso. Si ese fuera el caso, entonces cualquier persona podría escribir un mensaje en forma de números, y cualquiera podría leerlo.

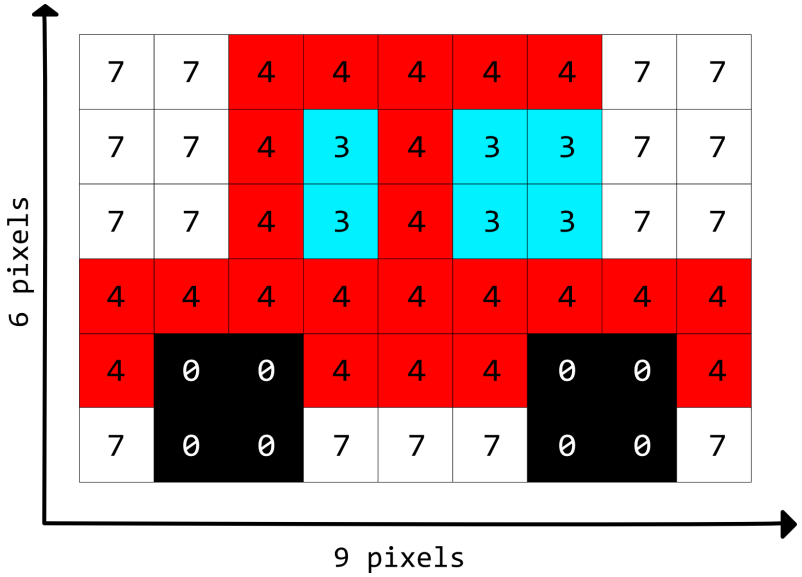
Más interesante aún, podríamos representar otras cosas que no sean letras como números, o como grupos de números, por ejemplo, colores:[5]



Teniendo colores, podríamos representar una imagen como una secuencia de números.



Para representar una imagen, la misma debe ser dividida en una grilla cuadriculada en donde cada celda tiene un único posible color. Luego podemos determinar el color de cada celda haciendo un reemplazo uno a uno con la representación de colores que vimos anteriormente, para obtener una secuencia de números. Además deberíamos indicar el alto y el ancho de la imagen, y elegir el orden en el que vamos a leer la secuencia (por ejemplo, de izquierda a derecha y de arriba a abajo).



Si tomamos los dos primeros dos dígitos como ancho y alto respectivamente, vemos que se indica que la imagen tiene 9 celdas de ancho, por 6 de alto. El resto corresponde a cada uno de las celdas de la grilla, indicando para cada una, que color tiene.

```
9 6
7 7 4 4 4 4 4 7 7
7 7 4 3 4 3 3 7 7
7 7 4 3 4 3 3 7 7
4 4 4 4 4 4 4 4 4
4 0 0 4 4 4 0 0 4
7 0 0 7 7 7 0 0 7
```

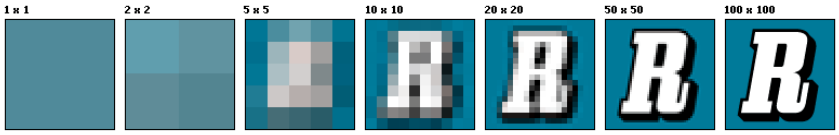
Dispusimos los números en forma de filas y columnas, similar a la imagen, para que sea fácil de leer, pero esto no es necesario. Más aún, como cada color se compone solo de un dígito, podemos eliminar completamente los espacios entre un dígito y otro, transformando nuestra imagen en un gran número.

```
967744444777743433777743433774444444444004440047007770 07
```

Note que el ancho y el alto no deben ser superiores a 10, para que efectivamente sea entendible por el receptor.

Definición 3.4 Se conoce como **codificación** al proceso mediante el cual una cierta información (por ejemplo, una imágen, un color, una letra) es transformada en algún otro elemento que la representa mediante la utilización de un código. El proceso inverso se conoce como

decodificación.



Ejemplo de como la calidad de imagen aumenta a medida que aumenta su resolución.
Imagen de dominio público.

Sabías qué

Las pantallas de los monitores, celulares y otros dispositivos electrónicos están compuestas de cientos de miles de pequeñísimos puntos conocidos como **pixeles**. Cada **pixel** puede prenderse de un único color (o estar apagado y verse negro). Las imágenes digitales entonces se almacenan guardando la información de que color debe tener cada pixel.

Los sensores de las cámaras digitales funcionan igual. La cantidad de sensores se mide en **megapixeles** y tiene que ver con la cantidad de puntos que van a conformar la imagen.

Las imágenes digitales entonces se almacenan basados en los pixeles, aunque como la cantidad de sensores de la cámara y la cantidad de puntos en la pantalla no siempre coincide, a veces un único pixel de una imagen puede corresponder a varios puntos de la pantalla. La cantidad de pixeles que tiene una imagen es lo que se conoce como **resolución** de la imagen. A mayor resolución, más nítida y de mejor definición, a costa de que ocupa mayor espacio de almacenamiento.

Es decir, cualquier tipo de elemento del que podamos querer hablar es suceptible de ser codificado mediante números.[3] Pero eso es solo el principio de un proceso que va más allá, pues todo número puede ser representado mediante binario.

3.1.5 Representando números con binario

El poder representar todo con números es una excelente idea, pero surge un problema al querer guardar esos números en una computadora. Como ya vimos, la computadora solo maneja unos y ceros (o electricidad y no electricidad), no números naturales como los que necesitamos para representar nuestras letras, colores e imágenes.

Por suerte, los números naturales pueden expresarse en binario con poco esfuerzo. El truco consiste en elegir una representación binaria distinta para cada número, de forma similar a lo que hicimos con las letras. Debemos para ello partir de la cantidad de cables que vamos a tener, por ejemplo, ocho cables. Luego, basta analizar todos los posibles estados en los que puede llegar a estar ese conjunto de cables (por ejemplo, todos sin electricidad, todos con electricidad, todos menos el último sin electricidad, todos menos los dos últimos sin electricidad, y así siguiendo). Es decir, analizamos todos los posibles números binarios que representarían a nuestro conjunto de cables. Luego, basta asignar un número a cada binario. Por ejemplo, si tuviéramos cuatro cables tendríamos:

número	binario	número	binario
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Note como la cantidad de cables que tenemos limita la cantidad de números que podemos representar, siendo mayor los números a mayor cantidad de cables. En particular:

$$2^{\text{cant. cables}} = \text{cant. números distintos} \tag{3.1}$$

Por supuesto no es la única representación posible, podríamos haber elegido asignar los números en otro orden. Existen diferentes formas estandarizadas para representar un número en binario, que incluso sirven para números negativos o racionales, algo que no analizaremos pues escapa a los alcances del presente.

Lo importante es que **a toda representación numérica, se puede asignar una representación binaria**. Así **toda la información digital, ya sea texto, colores, o imágenes, pueden representarse con binario**.[3]

Una secuencia de números se traduce entonces directamente a una secuencia de números binarios, es decir, a una larga tira de unos y ceros. Y por tanto, cualquier información es posible de ser representada como datos binarios.

3.1.6 Bits, Bytes y otros

Cuando se utilizan los unos y ceros para representar información se habla de **bits**.

Un bit es un dígito en el sistema de numeración binario. Es decir, corresponde a un único cero o uno en una posición determinada. **Si tenemos un dato que está representado por ocho cables (o imanes, o lo que fuera) decimos que está representado por 8 bits.**

Era muy común que las antiguas computadoras trabajaran con 8 bits. Como era muy común la medida de 8 bits para representar datos, surge la palabra **octeto** para indicar precisamente esa cantidad. Adicionalmente surge la palabra **byte**. El **byte** suele representar 8 bits, pero en algunas computadoras antiguas el término se utilizaba con otra acepción (por ejemplo, 6, 7 o incluso 9 bits). Hoy en día la mayoría de las computadoras hablan de **byte para referirse a 8 bits**.

También es muy común escuchar los términos **kilobit** o **megabit**. El prefijo corresponde a una escala adicional que indica una gran cantidad de bits. Así de forma similar a como un kilometro representa mil metros, un kilobit representa mil bits.

Nombre	Prefijo	Equivalencia
Kilobit	kbit	1000 bits
Megabit	Mbit	1000 Kbits = 1.000.000 bits
Gigabit	Gbit	1000 Mbits = 1.000.000.000 bits
Terabit	Tbit	1000 Gbits = 1.000.000.000.000 bits
Petabit	Pbit	1000 Tbit = 1.000.000.000.000.000 bytes
Exabit	Ebit	1000 Pbit = 1.000.000.000.000.000.000 bytes

Los bytes también son sujeto de los mismos prefijos, siendo un Kilobyte equivalente a 1000 bytes.[4]

Nombre	Prefijo	Equivalencia
Kilobyte	kB	1000 bytes = 8000 bits
Megabyte	MB	1000 KB = 1.000.000 bytes
Gigabyte	GB	1000 MB = 1.000.000.000 bytes
Terabyte	TB	1000 GB = 1.000.000.000.000 bytes
Petabyte	PB	1000 TB = 1.000.000.000.000.000 bytes
Exabyte	EB	1000 PB = 1.000.000.000.000.000.000 bytes

Para tener una idea de lo importante que son estas medidas, basta con acercarse a un local de hardware para comprar algunos dispositivos. Las memorias de computadoras (conocidas como RAM) tienen su capacidad de almacenamiento expresadas en Gigabytes (GB), siendo hoy en día comunes las de 2, 4 y hasta 8 GB, aunque hay incluso de más. Los discos rígidos también tienen su capacidad expresadas en GB, siendo comunes los de 250, 500, 750, e incluso hay algunos que llegan a ser de 1 TB.

Al contratar un servicio de conexión a internet, la velocidad de conexión se expresa en Megabits por segundo (Es decir, cuantos bits se pueden descargar en un segundo) siendo comunes hoy en día conexiones de 2, 4, 10, 20 y hasta 50 Mbits por segundo.

A continuación se deja una idea de cuanta información (comparada con información impresa) representa cada medida.

Número de bytes	Múltiplo	Equivalencia aproximada
1	1 B	Una letra.
10	10 B	Una o dos palabras.
100	100 B	Una o dos frases.
1.000	1 kB	Una historia muy corta.
10.000	10 kB	Una página de enciclopedia con un dibujo simple.
100.000	100 kB	Una fotografía de resolución mediana.
1.000.000	1 MB	Una novela.
10.000.000	10 MB	Dos copias de la obra completa de Shakespeare.
100.000.000	100 MB	Un estante de un metro lleno de libros.
1.000.000.000	1 GB	Una camioneta llena de páginas con texto.
1.000.000.000.000	1 TB	Páginas con texto elaboradas de 50.000 árboles.
10.000.000.000.000	10 TB	La colección impresa de la biblioteca de EE.UU.
1.000.000.000.000.000	1 PB	Los datos que maneja Google cada hora.
1.000.000.000.000.000.000	1 EB	Todos los datos en Internet para finales de 2001.

3.2 Sistemas de Caja Negra

En muchas ocaciones es útil pensar los sistemas como una **caja negra**. Es decir, un elemento o proceso del cual no nos importa el funcionamiento interno, pero si sabemos que, ante cierta entrada, producirá cierta salida.

Definición 3.5 Se denomina **caja negra** a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

Puede pensarse una analogía al proceso de caja negra con un auto. Cualquier persona puede conducir un auto. Basta comprender que, ante la presión del pedal del acelerador, y habiendo colo-

Ilustración de un proceso de Caja Negra.
Ilustración propia.

cado la palanca de cambios en la posición correcta, el auto irá hacia adelante. Los funcionamientos internos del vehículo, tales como los detalles sobre el funcionamiento de un motor de combustión interna, la batería, o la caja de cambios, no son necesarios desde el punto de vista del conductor.

En informática este proceso se debe llevar a cabo todo el tiempo, y en varios niveles. Pensemos por ejemplo en el CPU, uno de los componentes de hardware más complejos presentes en las computadoras modernas. El mismo está compuesto de millones de transistores colocados e interconectados de intrincadas formas. Si lo que nos interesa es diseñar microprocesadores, probablemente sea relevante conocer los detalles sobre como un procesador está estructurado y compuesto. Sin embargo, si lo que nos interesa es es componer los grandes componentes de hardware, el procesador será un elemento más, y su interior no es relevante. Más aún, los componentes de hardware no son relevantes si lo que interesa es analizar el software de la computadora. Un último ejemplo podría ser la forma en la que los datos están codificados, como binario, que no interesa si lo que queremos es utilizar el equipo, por ejemplo, editando una imágen o escribiendo un texto.

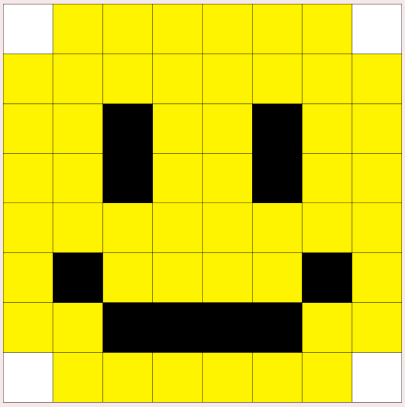
El concepto de caja negra utilizado en sistemas no debe confundirse con el "cajanegrismo"; éste es un concepto vinculado a la sociología que hace referencia al hecho de que las personas suelen olvidar el funcionamiento interno de las cosas (generalmente nuevos dispositivos tecnológicos) a medida que se familiarizan con ellos y terminan por asimilarlos como de uso cotidiano.

La principal diferencia entre ambos conceptos radica en que el **estudio de un sistema como una caja negra es un proceso de abstracción**, mientras que el "cajanegrismo" es más bien un proceso de olvido.

Por lo tanto, **concebir al hardware o al software de la computadora como una caja negra** puede ser útil desde el punto de vista de un programador o un usuario. **La concepción de caja negra es una idea que aplica de forma constante en las disciplinas informáticas**, y es un concepto recurrente ya que se deben realizar abstracciones de forma constante.

3.3 Actividades

Ejercicio 3.1 Dada la imágen a continuación, expresela como un único número utilizando la misma codificación que se aplicó en este capítulo.



■

Ejercicio 3.2 La tabla mostrada en este capítulo para representar letras como números corresponde a la codificación ASCII. Utilizando esa tabla se pide que codifique la siguiente frase como una serie de números.

SOMOS LO QUE PROGRAMAMOS ■

Ejercicio 3.3 Nuevamente, usando ASCII, se pide ahora que decodifique el siguiente mensaje expresado como una serie de números.

80 82 79 71 82 65 77 79 32 76 85 69 71 79 32 69 88 73 83 84 79 ■

Ejercicio 3.4 Si contamos con una computadora con 16 cables para representar nuestros datos, ¿Qué cantidad de números distintos pueden representarse con ellos? ■

Ejercicio 3.5 Un chiste de informáticos reza lo siguiente:

"Solo hay 10 tipos de personas en este mundo, los que entienden binario y los que no"

En qué radica la gracia del chiste. ■

4. Informática

Placa base de una computadora con sus circuitos impresos.
Fotografía de Blickpixel.

En este capítulo veremos como los conceptos del capítulo anterior se aplican a los conceptos que utilizamos día a día como usuarios de computadoras. Veremos qué son los archivos, las carpetas y los programas, y veremos como se crean los programas utilizando lenguajes de programación.

Esto nos llevará a analizar qué es exactamente un lenguaje y a ver diversos tipos de lenguajes que existen en informática. Nos centraremos luego en los lenguajes de marcado, como una forma de representar datos mediante texto.

4.1 Archivos y extensiones

Cuando utilizamos la computadora sabemos bien donde está guardada nuestra información. Archivos, ubicados en carpetas como “Mis Documentos” poseen nuestros documentos de texto y planillas de cálculo. Otros, ubicados en “Mis Imágenes” mantienen nuestros recuerdos en forma de imágenes digitales.

Pero ¿Qué es exactamente un archivo informático? ¿Y qué es un directorio? En esta sección analizaremos esas preguntas desde dos puntos de vista. En una primer instancia, lo veremos desde el punto de vista de la computadora en bajo nivel. Luego, veremos cuales son los detalles importantes desde el punto de vista del usuario final para poder administrar y manejar mejor los archivos.

También analizaremos diversos tipos de archivos, y veremos como el sistema es capaz de identificar esos tipos de archivos, y como opera con y sobre ellos.

4.1.1 Archivos informáticos

Un archivo informático no es más que la representación digital de un archivo físico que se tiene en papel. Puede contener texto, imágenes, videos, modelos tridimensionales, o cualquier otro contenido que uno quiera almacenar y manipular en la computadora.

En última instancia, un archivo informático se reduce solo a una serie de bits que la compu-

tadora es capaz de almacenar, transmitir, y manipular mediante los circuitos que posee, como ya mencionamos previamente. Pero surge la pregunta: "¿Qué es lo que hace que una serie de ceros y unos sea una imagen, y otra sea un video, audio o un modelo 3D?". La respuesta radica en el **formato de archivo**.

Cada archivo tiene un formato, que es lo que determina como la computadora debe leer e interpretar los bits que componen al mismo. El formato está relacionado a la codificación que se haya elegido. Por ejemplo, la codificación que elegimos para las imágenes en la sección anterior, es una, de muchas posibles codificaciones.

No hay un único formato para cada tipo de archivo, por ejemplo, no hay un único formato para todo lo que son imágenes. Cada formato tiene sus ventajas y desventajas, y está pensado para solucionar problemas puntuales que tienen que ver con representar ese archivo en la computadora en alguna forma particular, por ejemplo, con mayor calidad, con menor espacio posible, etc.

La información se **codifica** como binario y se **decodifíca** como información (de binario a información). Esa interpretación de los datos en ida y vuelta está dada entonces por el **formato de archivo**. Para que los archivos puedan compartirse y leerse como se espera por cualquier computadora, la forma en la que interpretamos esos ceros y unos debe ser la misma para todos, por lo que los formatos están **estandarizados**.

Definición 4.1 Un **archivo informático** es la represenación digital de un archivo físico, el cuál se encuentra almacenado en un dispositivo de almacenamiento como una cadena de bits la cual es la codificación de lo representado utilizando algún código determinado, determinando su formato. Un archivo además tiene un nombre y un tamaño (que depende de la cantidad de bits que tiene el archivo).[6]

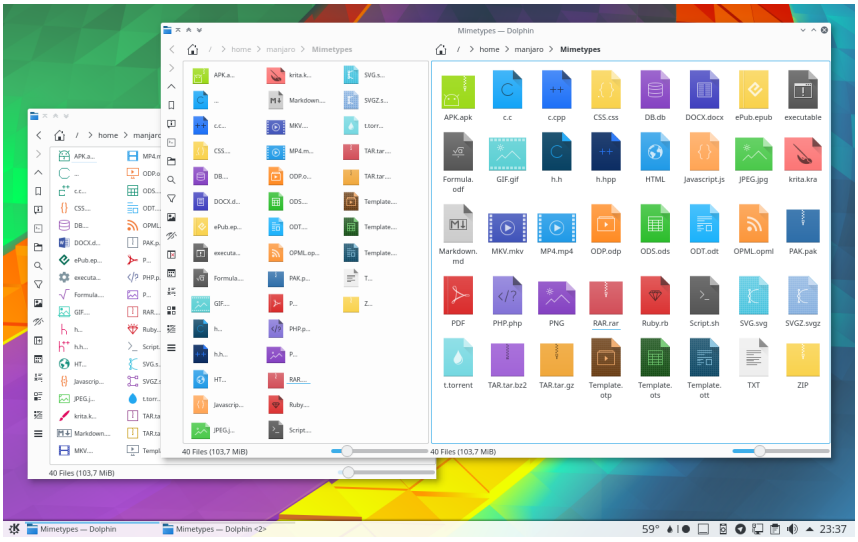
Para tener una mejor idea, analicemos un ejemplo simple. Una fotografía o imágen digital puede almacenarse en alguno de los muchos formatos disponibles para ello. Tres de los más utilizados son los formatos RAW, JPEG y PNG.

- Las imágenes en formáto RAW son aquellas en donde se almacena toda la información para cada pixel de la imágen. Es decir, se divide la imágen en una grilla gigante de pequeños puntos (pixeles), y el archivo consiste en la información de que color debe tener cada pixel. Esto otorga la mejor precisión de los colores y la mejor calidad de imagen posible, fiel al dispositivo que capturó la imágen. Como contrapartida, una sola imágen en buena calidad puede llegar a ocupar mucho tamaño en dispositivos de almacenamiento.
- Las imágenes JPG utilizan una técnica mediante la cual pueden eliminar la información de varios pixeles, los cuales pueden deducirse a partir de la información de los pixeles circundantes. Adicionalmente, se almacena la información de cada pixel mediante una técnica que permite reducir el tamaño de la información. Así, la calidad de imágen baja significativamente, pero el tamaño se reduce de forma acorde. Este formato es ideal para fotografías digitales y el estándar usado por la mayoría de las cámaras fotográficas.
- Las imágenes PNG utilizan técnicas que permiten minimizar el tamaño que ocupa la información de cada pixel, al pre-anotar los colores que aparecen en la imágen. Adicionalmente, marca grandes bloques de imágen anotando su posición relativa y tamaño, marcandolo con un único color. Este formato es ideal para imágenes que tienen pocos colores, o grandes bloques de un mismo color, por ejemplo, dibujos digitales.

Incluso los programas de la computadora son solamente archivos, cuyo formato le indica a la compu que debe ejecutarlo, y como hacerlo, algo que veremos más adelante.

Es importante saber que todo archivo tiene un **nombre**, un **tamaño** (dado en bytes) y una ubicación (un directorio en donde se encuentra). Además, dependiendo del sistema operativo y otros detalles puede contener otra información. Estos datos no componen a lo que representa el archivo en si mismo, sino que es información adicional que el mismo posee, y que se conocen como **metadatos**.

4.1.2 Extensiones de archivos



Muchos sistemas operativos utilizan iconos distintos para cada formato de archivo al mismo tiempo que ocultan la extensión del nombre de archivo. Esto permite al usuario identificarlo visualmente, a pesar de no leer el nombre completo. En la foto, una imagen del escritorio de un sistema con KDE plasma.
Imagen de KDE.org

Ahora bien, surge otra pregunta; ¿Cómo hace la computadora para saber que formato tiene cada archivo?. Si bien en realidad existen varias formas de hacerlo, lo más común es que se determine el formato mediante el nombre del archivo.

El nombre que vemos de los archivos, no es en general el nombre completo del mismo. El nombre completo se compone del nombre, y de una **extensión**, que no es más que un conjunto de letras que se anexan al final del nombre, separados del mismo por un punto.[7]

Así, por ejemplo, si tenemos una imagen con el nombre “de vacaciones”, lo más probable es que el nombre completo del archivo sea “de vacaciones.jpg”. La extensión del archivo es entonces “jpg” (o a veces, según la bibliografía también “.jpg”)

En realidad

Como ya se dijo, “la computadora”, no hace nada por si sola, es el software el que hace. En particular, el que detecta el formato de archivo y elige la forma de interpretarlo es el sistema operativo. Esta es, entre otras varias, una de las tareas que tiene el mismo.

Además, algunos sistemas operativos utilizan otras técnicas adicionales a mirar la extensión de archivo para determinar el formato, como por ejemplo, la lectura de la primer parte del archivo en búsqueda de “números mágicos’ (un número que, en general, es distinto para distintos formatos de archivo).

Existen miles de extensiones de archivo, algunos relativamente comunes (por ejemplo JPG para fotografías, o MP3 para música), y otros que son bastante más exóticos, pues son usados por formatos de archivo que sirven solo para una aplicación en particular (por ej. DWG, formato de archivo de AutoCAD).

A continuación se presenta un listado con algunos de los formatos de archivo más comunes que uno puede encontrar en el uso cotidiano:

Fotos e imágenes:
- JPG
- JPEG
- PNG
- APNG
- BMP
- TIFF
- GIF
- SVG

Audio:
- MP3
- OGG
- WAV
- 3GP
- M4A
- FLAC
- AIFF

Video:
- MP4
- AVI
- DIVX
- XVID
- MOV
- WMV
- FLV
- MKV

Documentos:
- DOC
- DOCX
- ODT
- XLS
- XLSX
- ODS
- PPT
- PPTX
- ODP
- PDF

Archivos comprimidos:
- ZIP
- ZIPX
- RAR
- TAR
- GZ
- ISO

Texto plano:
- TXT
- MD
- XML
- HTML
- JSON
- JS
- C
- CSS
- JAVA

Una categorización muy amplia y genérica, pero sumamente útil es la que agrupa los archivos en los siguientes tres grupos:

Archivos ejecutables Son los programas de la computadora. Programas como Word, Excel, Paint, aplicaciones de celulares, editores de fotos, etc.

Archivos de datos binarios Son los archivos que solamente pueden ser leídos por programas específicamente diseñados para tal fin: documentos de Word, imágenes, videos, audio, etc. Su codificación puede ser estándar, o puede estar restringida a ser solo conocida por la aplicación que maneja el archivo.

Archivos de texto plano Son archivos que usan una codificación estándar y en donde su contenido representa pura y exclusivamente texto. Pueden ser leídos por un **editor de texto**. La gracia de los archivos de texto es que la interpretación de sus bits representa siempre lo mismo, texto.

Un detalle más

Más allá de que los archivos representen lo mismo (por ej. una imagen) cambiar la extensión de archivo no hace que cambie el formato en el que está codificado, por lo que transformar una foto en .jpg en una en .png requiere del uso de un programa que sea capaz de cambiar todos los unos y ceros del archivo en otros que representen lo mismo, pero con otra codificación. Los archivos de texto pueden ser cambiados de extensión tantas veces como se quiera, y siguen pudiendo abrirse con el mismo programa, pues siguen representando siempre texto, y la codificación es la misma para todos los archivos de texto.

Dentro de cada uno de estas categorías se encuentran a su vez sub-categorías que terminan por definir de forma más granular la forma en la que se interpreta cada archivo. Por ejemplo, las imágenes pueden estar codificadas de diversas formas (diferentes a las que vimos), cada una con ventajas y desventajas, y por tanto la interpretación de los ceros y unos del archivo, cambia según el formato.

4.2 Organización de archivos, directorios y rutas

Todos los archivos se deben almacenar en algún lado para poder utilizarlos cuando se desee. Esto puede ser, en un disco rígido, en un disco extraíble (CD, DVD, Pendrive, Tarjeta de memoria, etc). La forma en la que se estructuran los archivos en el disco se conoce como **sistema de archivos**.

Uno de los primeros sistemas de almacenamiento para guardar archivos era el disco flexible (En inglés Floppy), un disco magnético extraíble, recubierto de una carcasa plástica de protección. Los primeros sistemas operativos que soportaban estos discos como espacio de almacenamiento, solo permitían guardar en ellos unos pocos archivos. Las personas agrupaban archivos relacionados en un mismo disco, y en caso de que la cantidad de archivos fuera mayor a la del disco, agrupaban discos fisicamente.

Sabías qué
Los primeros sistemas de archivos utilizaban una cantidad preestablecida de bits para representar el nombre y extensión de cada archivo. Por eso, en sistemas operativos antiguos, el nombre de un archivo no podía superar los 8 caracteres, y la extensión no podía superar los 3. Está característica quedó, y por eso la mayoría de las extensiones hoy cuentan con solo 3 caracteres, más allá de que esa limitación ya no existe.

A medida que aumentaba la capacidad de los discos, y la cantidad de archivos que podían almacenar, el organizar archivos separándolos en distintos discos se vuelve ineficiente. Así surgen los primeros sistemas de archivos con **directorios** (también llamadas carpetas).

Windows utiliza la anología de archivadores físicos, carpetas y archivos para su sistema operativo. Imagen propia.

Los directorios son el equivalente lógico de una carpeta física, permitiendo agrupar archivos (en base a alguna cualidad que el usuario considere oportuna).

Además, una carpeta puede estar dentro de otra carpeta, generando una estructura de árbol. En estas estructuras siempre hay una carpeta que contiene a todas las otras, conocida como carpeta raíz.

Definición 4.2 Un **directorio** es la represenación digital de una carpeta física. La misma se encuentra en algúna ubicación en algún dispositivo de almacenamiento, posee un nombre, y puede contener ninguno o varios archivos y/o directorios en su "interior". El tamaño, en bytes, de un directorio es la suma del tamaño de todos los directorios y archivos que contiene.[6]

Cada sistema operativo tiene su propio sistema de archivos, y su forma particular de organizar las carpetas. Por ejemplo, Windows tiene una carpeta raíz por cada disco rígido que tenga conectada

la computadora. Linux y macOS utilizan una única carpeta raíz que contiene como sub-carpetas los diversos discos conectados.

Los sistemas de archivos basados en directorios permitían cosas muy interesantes, como por ejemplo, tener varios archivos con el mismo nombre (nombre completo, es decir, nombre y extensión) siempre y cuando se encuentren en distintas carpetas. Pero también traían aparejados otros problemas, como que ahora los archivos no son identificables mediante solo su nombre, sino mediante el nombre y la carpeta en la que se ubican.

Los sistemas de archivos que permiten directorios que contienen a otros directorios y archivos se denominan sistemas de archivos jerárquicos.

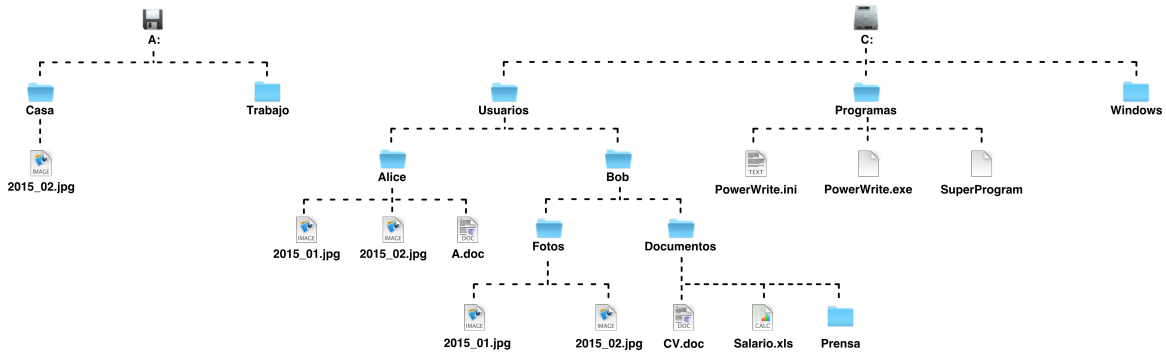
4.2.1 Rutas absolutas

En los sistemas operativos cuyo sistema jerarquicos (lo cuál es hoy en día prácticamente la totalidad de los mismos, aunque hay excepciones) la ruta es la forma de identificar univocamente un archivo en el equipo. Así, todo archivo tiene asociada una ruta absoluta en el equipo.

Definición 4.3 Una **ruta absoluta a un archivo** es el camino de carpetas que hay que seguir en el árbol de directorios para llegar **desde el directorio raíz hasta el archivo en cuestión**.[8]

La ruta suele variar entre un sistema operativo y otro, pues como ya mencionamos, cada sistema tiene su propia forma de estructurar las carpetas, pero el concepto subyacente es el mismo.

A continuación se muestra un detallado ejemplo con algunos directorios en un sistema Windows para anlizar rutas absolutas de varios archivos.



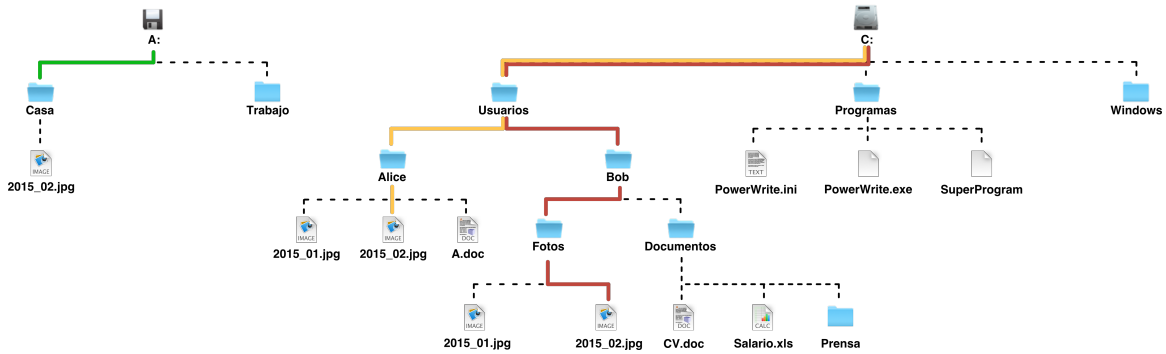
En este sistema hay dos dispositivos de almacenamiento. Un disco extraible “floppy” que Windows ha identificado con el nombre “A:”, y un disco rígido, el cual se identifica con “C:”. Es decir, “A:”, y “C:” son las carpetas raices de cada uno de los dispositivos de almacenamiento presentes en el sistema.

Dentro del disco rígido hay tres carpetas, “Usuarios”, “Programas” y “Windows”. La carpeta “Windows” está **vacía**. Un directorio se denomína vacío cuando no contiene otras carpetas o archivos en su interior. La carpeta “Programas” contiene solamente tres archivos en su interior, “PowerWrite.ini”, “PowerWrite.exe” y “SuperProgram”. La carpeta “Usuarios” tiene dentro dos carpetas, “Alice” y “Bob”, las cuales a su vez contienen otras carpetas y archivos en su interior.

Surgen varias cosas de este ejemplo. En primer lugar podemos ver que hay más de un archivo con el nombre “2015_02.jpg”. ¿Significa esto que todos son el mismo archivo? La respuesta es no, solo son archivos con el mismo nombre, pero el contenido del mismo podría ser completamente distinto. Podemos así modificar uno de esos archivos sin afectar el contenido de los otros. También

surge la pregunta ¿Cómo sabe el sistema de cuál de los archivos estámos hablando? La respuesta es, mediante la ruta absoluta del archivo.

Cuando se trabaja sobre un archivo en el equipo, y aunque transparente para el usuario en la mayoría de los casos, se debe indicar la ruta completa al archivo, para que el sistema lo identifique de forma unívoca. Tomemos el ejemplo del archivo "2015_02.jpg"; dicho nombre no es único, sino que existen tres archivos con el mismo en úbicaciones distintas en el equipo. Si deseamos trabajar con cualquiera de ellos, deberíamos dar al sistema la ruta absoluta de aquel archivo que nos interesa. Para obtenerla uno debe imáginar una especie de camino que parte desde la carpeta raíz (por ejemplo, "C:") y luego continúa por las lineas punteadas hasta arribar en el archivo, pasando en el camino por una serie de directorios intermedios (siempre moviendose hacia abajo y a los laterales, obteniendo así el camino más corto). En el gráfico siguiente se muestran tres caminos distintos que llevan a los distintos archivos con el nombre "2015_02.jpg", marcados cada uno con un color distinto.



Cada línea corresponde entonces a la ruta absoluta para cada uno de los archivos con el nombre "2015_02.jpg". La primer línea, la verde, parte desde el disco floppy "A:", pasa por el directorio "Casa", para luego arribar en el archivo. Así, vamos a decir que la ruta a ese archivo es:

■ Ejemplo 4.1 Ruta roja
A: » Casa » 2015_02.jpg ■

En el segundo caso, la ruta amarilla, parte desde el disco rígido "C:", y pasa luego por la carpeta "Usuarios", para continuar por "Alice" previo a arribar al archivo. Los pasos serían entonces:

■ Ejemplo 4.2 Ruta amarilla
C: » Usuarios » Alice » 2015_02.jpg ■

Finalmente, la ruta roja parte igualmente desde el disco rígido "C:", y pasa luego por la carpeta "Usuarios", pero continua ahora por "Bob", luego en "Fotos" para arribar ahora si, al archivo:

■ Ejemplo 4.3 Ruta roja
C: » Usuarios » Bob » Fotos » 2015_02.jpg ■

Como vemos, tres archivos con el mismo nombre tienen rutas completamente distintas. Podemos obtener varias conclusiones de este ejemplo. Por un lado vemos que los sistemas de archivos con directorios permiten que múltiples archivos se llamen de la misma forma, ya sea que efectivamente sean el mismo archivo, o que no lo sean. Por otro lado, no pueden existir nunca dos archivos con el mismo nombre en un mismo directorio.

Miremos con atención el caso de "PowerWrite". Existen dos archivos con ese nombre en la misma carpeta, sin embargo, tienen distinta extensión. Tenga presente el lector que siempre que hablamos del nombre de un archivo, en general nos referimos al nombre y su extensión.

De la experiencia que usted pueda tener con el uso de computadoras, sabrá que los archivos pueden copiarse, y moverse a diferentes directorios. ¿Qué ocurre si muevo un archivo de lugar? Lo que sucede es que su ruta absoluta cambia. Todo esto puede parecer algo menor, pero las rutas absolutas son utilizadas por los programadores para hacer referencia a archivos que deben estar presentes en el equipo para que un determinado sistema funcione. Mover un archivo de lugar puede hacer que todo un sistema falle, pues el archivo no se encuentra en la ruta que el sistema esperaba. Así también, modificar configuraciones de un sistema que involucran rutas absolutas sin mover de forma acorde los archivos puede ocasionar el mismo efecto. Por tanto, entender qué y cómo funcionan las rutas es una parte fundamental para cualquier persona que quiera comprender un poco más en profundidad los sistemas informáticos, cualquiera sea su fin.

Es menester mencionar algunos detalles sobre nomenclatura de rutas en diversos sistemas. Note que en los ejemplos utilizamos el símbolo "»" para separar cada uno de los elementos de una ruta. En realidad cada sistema operativo puede utilizar una símbolo distinto para separar los elementos de una ruta. Windows utiliza por convención la barra invertida (\). Linux, macOS, y otros sistemas (aquellos que utilizan la convención de los sistemas básados en UNIX, y que en general se denominan sistemas basado en UNIX, o sistemas POSIX) separan los elementos con barra símple (/). Así, las rutas de los ejemplos de esta sección se escribirían en Windows:

- A:\Casa\2015_02.jpg
- C:\Usuarios\Alice\2015_02.jpg
- C:\Usuarios\Bob\Fotos\2015_02.jpg

La convención en sistemas UNIX, es también la utilizada por muchos lenguajes de programación y de marcado, como veremos más adelante. Utilizando esta convención las rutas anteriores quedarían.

- A:/Casa/2015_02.jpg
- C:/Usuarios/Alice/2015_02.jpg
- C:/Usuarios/Bob/Fotos/2015_02.jpg

Como detalle final, cabe mencionar que los sistemas UNIX no utilizan una letra para cada uno de los dispositivos, sino que suelen tener un sistema de archivos que comienza en una carpeta virtual denominada "/" ("raíz" o "root"). Los diversos dispositivos son simplemente carpetas dentro de alguna de las carpetas que tiene "/", por ejemplo, la carpeta "mnt" o "mount". Las rutas absolutas en dichos sistemas comienzan entonces con la barra símple, para indicar que parten desde la raíz. Las rutas anteriores podrían verse algo como:

- /mount/floppy1/Casa/2015_02.jpg
- /mount/hd0/Usuarios/Alice/2015_02.jpg
- /mount/hd0/Usuarios/Bob/Fotos/2015_02.jpg

Las rutas absolutas son un concepto simple de entender, pero complejo de dominar sin práctica en su uso. Más dificil aún es el concepto de rutas relativas.

4.2.2 Rutas relativas

Historicamente, los primeros sistemas funcionaban indicandole a la computadora comandos a realizar, mediante la introducción de las mismas utilizando el teclado. El usuario solamente visualizaba una pantalla con texto, y el mouse no existía. En ese contexto, indicar la ruta absoluta de forma constante para cualquier archivo que el usuario quisiera manipular era sumamente engorroso. Para solucionar esa molestia surgen dos ideas, la primera es el concepto de **directorio de trabajo actuál**, y la segunda las **rutas relativas**.

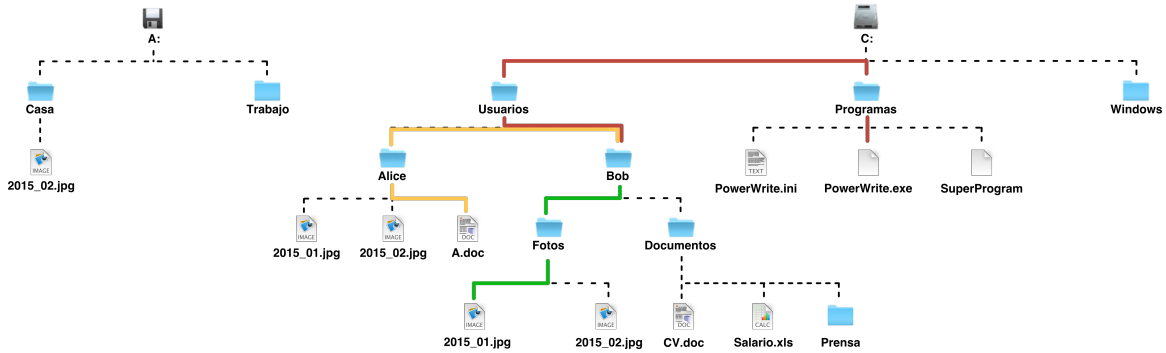
Definición 4.4 El **directorio de trabajo actuál** es un directorio de un sistema de archivos jerarquico el cual se encuentra asociado de forma dinámica a un proceso o tarea del sistema.

En estos sistemas antiguos, el usuario se encontraba siempre trabajando en un directorio, el cual era denominado el directorio de trabajo actual. El sistema operativo mantenía un registro del directorio de trabajo actual, ya que el mismo podía cambiar a lo largo del tiempo, pudiendo ser modificado por el usuario mediante la utilización de uno o más comandos. Este concepto viene de la mano con la idea de **rutas relativas**.

Definición 4.5 Una **ruta relativa** hacia un archivo consiste en la ruta a realizar desde un directorio cualquiera para dar con dicho archivo. Si el directorio elegido es el directorio raíz del dispositivo en donde se encuentra el archivo, la ruta relativa y la ruta absoluta son la misma.[9]

Una ruta relativa puede indicar donde se encuentra un archivo sin tener que conocer todas las carpetas del equipo. En particular, en estos primeros sistemas operativos, uno podía mencionar un archivo con la ruta relativa desde el directorio de trabajo actual hacia el mismo. Así, sin importar en que carpeta uno estuviera trabajando, si se deseaba hablar de un archivo en esa carpeta bastaba solo con el nombre, incluso si la ruta absoluta era larga y compleja.

Veamos un ejemplo en donde el usuario ha decidido que el directorio de trabajo actual sea la carpeta “Bob”, dentro de la carpeta “Usuarios” en el disco rígido “C:”. Es decir, todas las rutas serán relativas a la carpeta “C: » Usuarios » Bob”.



Analicemos primero la ruta verde. Bob quiere acceder al archivo en “C: » Usuarios » Bob » Fotos » 2015_01.jpg”. Como el directorio de trabajo ya es “C: » Usuarios » Bob”, basta con poner la ruta desde dicha carpeta (sin incluirla) hasta el archivo.

■ **Ejemplo 4.4 Ruta verde**
Fotos » 2015_01.jpg ■

Es decir, desde la carpeta “Bob” basta ir a la carpeta “Fotos” para encontrar allí el archivo “2015_01.jpg”. Fijesé como la ruta siempre va hacia “abajo”, es decir, que nos adentramos más en la jerarquía de carpetas, siempre yendo hacia la carpeta que está dentro de la anterior, y nunca saliendo de la misma.

En el caso de la ruta amarilla esto no es así. Bob quiere acceder al archivo de Alice, pero dichos archivos no se encuentran dentro de una carpeta en “Bob”, es decir, se debe “subir” para poder acceder a ellos. Es decir, la ruta relativa debe dejar constancia de que, en primer lugar, hay que ir de la carpeta “C: » Usuarios » Bob” a “C: » Usuarios” en lo que podríamos denominar “subir un nivel”. La ruta quedaría algo como:

■ **Ejemplo 4.5 Ruta amarilla**
^ » Alice » A.doc ■

Es decir, partiendo de la carpeta "Bob", vaya a la carpeta inmediatamente arriba (llamada **padre**). Luego, desde ahí, vaya a la carpeta "Alice", una vez allí encontrará el archivo "A.doc".

En el caso de la ruta roja el proceso es el mismo, aunque en este caso no basta con subir un nivel, sino que tenemos que subir varias veces.

■ **Ejemplo 4.6 Ruta roja**
^ » ^ » Programas » PowerWrite.exe ■

En este caso la ruta absoluta hubiera sido más corta y sencilla que la ruta relativa, sin embargo es interesante ver como una ruta relativa puede aplicar en cualquier punto de la jerarquía de carpetas, e independientemente de la carpeta.

Nuevamente el símbolo "^" que utilizamos para indicar "ir un nivel hacia arriba" se representa distinto en distintos sistemas. Sin embargo, los sistemas más populares usan todos la misma convención: "..". Las rutas anteriores quedarían en Windows:

- Fotos\2015_01.jpg
- ..\Alice\A.doc
- ..\..\Programas\PowerWriter.exe

Y en los sistemas UNIX

- Fotos/2015_01.jpg
- ../Alice/A.doc
- ../../Programas/PowerWriter.exe

Una propiedad interesante de las rutas relativas es que, dado un directorio como directorio de trabajo actual, todas las rutas relativas desde el mismo hacia archivos o carpetas que este contenga son iguales independientemente de donde esté ubicado el directorio en el equipo. Esto resulta muy util a las personas de sistemas, pues un archivo de configuración o programa que utilice esta técnica para hacer mención de sus archivos puede ser copiado o movido a otro lado sin perjudicar su funcionamiento, dado que no se copie solo el archivo, sino todos los archivos que están en la carpeta del programa. Veremos más este detalle en secciones futuras.

4.2.3 Universal Resource Identifier

Imaginemos ahora una serie de computadoras conectadas entre si. Es decir, una red de computadoras. Las redes permiten compartir recursos, los cuales, entre otras cosas, incluyen archivos y directorios. Por tanto, es importante poder identificar de forma única un archivo, ya no en un equipo, sino en toda una serie de equipos. Si la cantidad de máquinas es grande, nombrar con una única letra cada dispositivo de la red se vuelve imposible. Para solucionar este problema surgen el **identificador universal de recursos**, o en inglés **universal resource identifier (URI)**. Pero antes de entender que son los URIs, veamos un momento como se identifica un equipo en una red.

Cuando se cuenta con diversas computadoras conectadas a una red, se debe poder hacer mención a una máquina en particular, diferenciandola del resto de las de la red. Así, los sistemas de software de red (por ejemplo, aquellos incluidos en un sistema operativo o incluidos en dispositivos como routers) asignan a cada equipo un número único, conocido como **dirección IP**.

Otra técnica para hablar de un equipo único en una red consiste en asignarle un **nombre de dominio**. Los nombres de dominio son identificadores en texto, en lugar de numéricos, por lo que

son más fáciles de recordar por los seres humanos. Un sistema especializado mantiene registro de los nombres asignados para que no existan dos equipos con el mismo nombre, además de recordar la asociación entre el nombre de dominio y la dirección IP a la que corresponde.

Tal vez haya visto en algún momento estas formas de identificar computadoras al navegar por Internet, pues Internet no es más que una gran red donde el usuario simplemente solicita archivo a diversos equipos. Una dirección IP se compone en realidad de 4 números entre 0 y 255 cada uno, los cuales se escriben separados por púntos (ej. “125.65.92.135”). En cambio los nombres de dominio suelen consistir en dos partes, de las cuales la primera suele ser un nombre que identifica a una empresa u organización, y la segunda, que consiste en un conjunto de 2 y/o 3 letras que identifican el tipo de organización (empresa, ONG, entidad educativa, etc.) y/o el país al que pertenece (ej. “google.com.ar”, “wikipedia.org”).

Ahora bién, una URI se compone entonces de diversas partes. En primer lugar un protocolo, que le indica a los equipos de que forma se espera que se transfiera el archivo de una máquina a la otra, seguido por alguna forma de identificador de equipo (ya sea la dirección IP o un nombre de dominio) y luego una ruta hacia el archivo en cuestión. Como por motivos de seguridad las computadoras no comparten en la red todos los archivos sino una carpeta específica del equipo, la ruta de la URI no es una ruta absoluta, sino relativa a la carpeta compartida.[10]

Un URI también puede ser llamado URL (Universal Resource Locator). Si bien hay pequeñas diferencias técnicas entre uno y otro concepto, los términos hoy en día se suelen utilizar de forma prácticamente intercambiable. Además, tanto los URI como los URL pueden contener otros elementos, como sub-dominios, puerto, querys, y fragmentos, que no analizaremos aquí.

Definición 4.6 Un **identificador de recursos universal** es una cadena de texto capaz de identificar un recurso en un equipo específico dentro de una red de computadoras.

El concepto de URI no es nada más que el mismo concepto de ruta expandido hacia las redes, de forma tal que cualquier archivo (que sea accesible) tenga una forma única de identificarse.

Cabe destacar como detalle adicional que el símbolo utilizado en una URI para separar los elementos de la ruta sigue el estándar de los sistemas UNIX, es decir, con barra símble (/). Esto quiere decir que, cuando se accede a un archivo en un sistema Windows desde Internet, las barras invertidas de la ruta al archivo en Windows deberán ser reemplazadas con barras símples.

4.2.4 Programas, Editores y Visualizadores

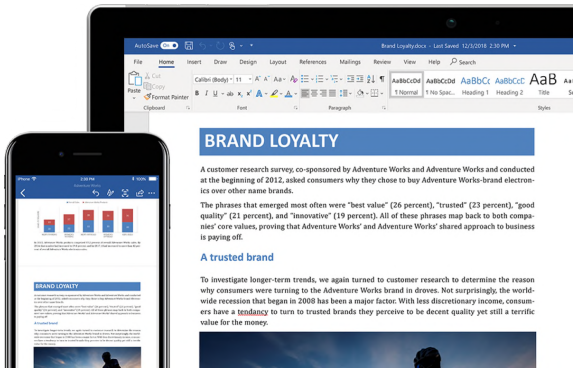
Los programas son uno de los elementos de software que más utilizamos. Son lo que provee a nuestra computadora de diferentes funcionalidades espcíficar, y nos permite realizar tareas sumamente diversas con un mismo equipo.

A diferencia del resto de los archivos del equipo, un programa es un archivo ejecutable. Es decir, al pedirle al sistema operativo que abra el archivo de un programa, lo que este hará será ejecutarlo.

Ejecutar un programa consiste simplemente en llevar adelante una serie de pasos que el programa define. Estos pasos pueden ser muchos, y muy diversos, dando lugar a programas

completamente distintos. Además, pueden incluir elementos visuales, como botones, imágenes y otros, para simplificar el uso para el usuario final.

Los programas pueden agruparse en categorías según diversos criterios. Por ejemplo, podríamos decir que hay programas que son "accesorios", elementos prácticos que el usuario puede requerir y conviene tener a manos, como una calculadora o un anotador. También hay programas que podrían entrar en una categoría "reproductores de video", que permiten al usuario ver videos en uno o más formatos, asi como controlar factores como el sonido y otros.



Banner publicitario de Microsoft Word con su aplicación para PC y teléfonos móviles.
Imagen de Microsoft.

Una categorización interesante que nos interesa mencionar aquí tiene que ver con como tratan los programas a los archivos que manipulan. En particular, podemos determinar dos grandes categorías, **visualizadores** y **editores**.

Un visualizador permite simplemente ver los datos que se encuentran almacenados en un archivo, pero no modificarlos. Por ejemplo, un reproductor de video permite ver una pelicula digital, pero no modificarla. Un navegador de internet permite ver páginas web, pero nuevamente, no pueden modificarse usando ese programa.

Definición 4.7 Un **visualizador** es un programa que permite ver el contenido de archivos que tienen algún formato específico.

Los editores en cambio permiten manipular la información del archivo, alterando sus contenidos. Por ejemplo, un editor de sonido, o un manipulador de imágenes permiten editar audio o fotografías digitales, y no son los programas que el usuario utilizaría si lo que le interesa es escuchar música o ver fotografias.

Definición 4.8 Un **editor** es un programa que permite crear y modificar el contenido de archivos que tienen algún formato específico.

Existen programas que cumplen ambas funciones al mismo tiempo. Por ejemplo Microsoft Word es el programa que se usa tanto para editar archivos de documentos, como para visualizar su contenido. Los documentos de Word no pueden (en principio) verse con otros programas que no sean Word, y tampoco editarse con otros programas.

Una mención especial merecen los **editores de texto**. Un editor de texto permite editar el contenido de un archivo de texto sin formato. Como veremos más adelante, algunos formatos de archivo no codifican la información en binario, sino como texto. Para estos archivos, un editor de texto permite modificar su contenido, independientemtente si codifican imágenes, documentos, u otros.

4.3 Programas y Lenguajes

Un programa es el resultado del trabajo de un desarrollador de software (dentro de otros elementos), por lo que para realizar un programa se requiere de una serie de conocimientos técnicos específicos que permiten desarrollarlo.

Resulta interesante tener una noción básica de la forma en la quese realiza desarrolla un programa (sin internalizar en los conceptos específicos). Para encarar esta tarea es prioritario comprender el concepto de lenguaje, desde un punto de vista técnico.

4.3.1 Lenguajes

El lenguaje nos posibilita comunicarnos con quienes nos rodean, pero también con otras personas a lo largo del tiempo y el espacio. Gracias al lenguaje es que hoy tenemos registro de lo que pasó en tiempos distantes, en lugares remotos.

Definición 4.9 Un **lenguaje** es un sistema de comunicación, el cual se encuentra definido y estructurado, y que posee un contexto de uso (es decir, se puede establecer una semántica a determinados elementos).

Banderas de diversos paises ondeando en la plaza de los Juegos Panamericanos en Rio de Janeiro. Imagen de Wilson Dias / Agência Brasil

Puede sonar complejo en la definición, pero utilizamos el lenguaje todo el tiempo. La forma más común de lenguajes son los idiomas, como el español o el inglés. Cada uno de los idiomas define un conjunto de símbolos que son válidos, por ejemplo, el español cuenta con la letra “Ñ”, que no está presente en el inglés. Cualquier palabra que use dicho símbolo es inválida en inglés, y por tanto no forma parte de ese lenguaje, mientras que en cambio, si puede formar parte del nuestro. También las reglas de un idioma (que son extremadamente complejas) determinan que textos son válidos y cuales no. En primer lugar, la regla más sencilla determina palabras que existen en el lenguaje (dadas muchas veces por los diccionarios oficiales). La gramática determina luego como unir esas palabras para que cobren sentido.

No solo hay idiomas que han surgido y evolucionado naturalmente, sino que pueden existir lenguajes creados de forma artificial con fines específicos. Por ejemplo, algunas series de televisión han creado lenguajes con fines artisticos, como la serie de ciencia ficción Star Trek y el lenguaje Klingon, o Game of Thrones (basada en los libros de George R. R. Martin) y el Idioma Dothraki.[11]

Pero también han habido intentos más nobles, como el Esperanto. Creado por L. L. Zamenhof, un oftalmologo poláco, el Esperanto surge con la idea de que se transforme en un idioma capaz de ser facilmente aprendido y hablado por cualquier persona en el mundo. La intención última era unir a la humanidad bajo un mismo lenguaje universal (como segunda lengua, sin reemplazar los idiomas oficiales que ya tenían las distintas naciones). El Esperanto es la lengua planificada más expandida y utilizada en el mundo hoy en día, y es reconocida por instituciones como la UNESCO.

Todo lenguaje está computesto de dos partes, una **sintaxis** y una **semántica**. La sintaxis determina los símbolos y las reglas que es posible utilizar en el lenguaje, mientras que la semántica determina que significado le asociamos a una determinada sintaxis. Por ejemplo, las expresiones “Me baño en el río” y “Me río en el baño” son muy similares en cuanto a sintaxis se refiere, pero la

semántica asociada, es decir, el significado de esas dos frases, es completamente distinto.

Pensemos también en una frase como "lo voy a tomar". ¿Cuál es la semántica de dicha frase?. A priori es dificil saberlo pues la palabra "tomar" tiene muchos significados distintos dependiendo del contexto. Puede que se esté hablando de un cuarto de un hotel, o de un transporte público. También podría ser que la frase se refiera a una bebida, o a un helado. Incluso la palabra tomar puede referirse a agarrar un objeto. Decimos entonces que la frase es **ambigua**, pues su significado no queda claro si no se cuenta con el contexto subyacente.

La ambiguedad es una problemática que a los humanos no nos molesta demasiado, pues somos capaces de analizar contextos y desambiguar rapidamente una frase basandonos en la información que tenemos. Las máquinas por otro lado, no tienen esa capacidad. Por tanto, es dificil (imposible prácticamente) comunicarse con una computadora utilizando los idiomas que hablamos a diario. Por eso surgen lenguajes especialmente diseñados para tal fin, y que tienen como principal caracteristica el no presentar ambiguedades. Estos lenguajes se conocen como **lenguajes formales**.

Definición 4.10 Un **lenguaje formal** es un lenguaje cuyos símbolos y reglas para unir esos símbolos están formalmente especificados. Al conjunto de los símbolos primitivos se le llama el **alfabeto**, y al conjunto de las reglas se lo llama la **gramática formal**.[12]

Dentro de los lenguajes formales podemos identificar dos grandes categorías: los **lenguajes de programación**, cuya finalidad es poder escribir programas de computadora, y los **lenguajes de marcado**, cuyo fin es describir datos que un programa puede luego interpretar.

4.3.2 Lenguajes de programación

Los **lenguajes de programación** consisten en un conjunto de comandos (llamadas a veces instrucciones) que la computadora es capaz de entender. Mediante la secuenciación de estas instrucciones se pueden lograr complejos programas que realizan diversas tareas. Así, puede procesar datos que provienen por sensores, o que son ingresador por el usuario, pueden imprimir información en pantalla, o transmitir información por la red, dependiendo de qué es lo que el programador haya determinado.

Cabe destacar que, en general, los comandos incluidos en un lenguaje suelen realizar tareas sencillas y bien puntuales. Por ejemplo, es casi seguro que un lenguaje de programación no va a incluir un comando para calcular la hipotenusa de un triangulo sabiendo los catetos del mismo. Sin embargo, una persona con conocimientos del teorema de Pitágoras puede llegar al mismo resultado utilizando una serie de comandos más simples (multiplicación, radicación y sumas). A continuación hay un ejemplo de programa que calcula la hipotenusa de un triangulo dados sus catétos.

■ **Ejemplo 4.7 código de ejemplo para el cálculo de hipotenusa de un triangulo**

1. Leer del teclado el número que el usuario ingresa como C1
2. Leer del teclado el número que el usuario ingresa como C2
3. CSQ1 ← C1 * C1
4. CSQ2 ← C2 * C2
5. HSQ ← CSQ1 + CSQ2
6. H ← raiz de HSQ
7. Informar al usuario en pantalla el valor H

■

Definición 4.11 Un **lenguaje de programación** es un lenguaje formal que especifica una serie de instrucciones para que una computadora procese y produzca diversas clases de datos. Los lenguajes de programación se usan para crear programas de computadora.[13]

En este código "*" se utiliza como símbolo para multiplicar, y "←" indica que el resultado de la cuenta debe ser recordado con un nombre (que figura a la izquierda de la flecha) para poder ser utilizado posteriormente.

No existe un único lenguaje de programación, sino que hay cientos de ellos, cada uno con sus propias características, y diseñados para satisfacer necesidades específicas. Así, **no existe un "mejor lenguaje"**, sino que existen lenguaje más o menos apropiados para solucionar un determinado problema. Por otro lado, muchos lenguajes distintos permiten solucionar los mismos problemas, pero con diferentes enfóques, por lo que el **programador puede elegir aquel que más se adapte a su necesidad, pero también a sus preferencias personales**.

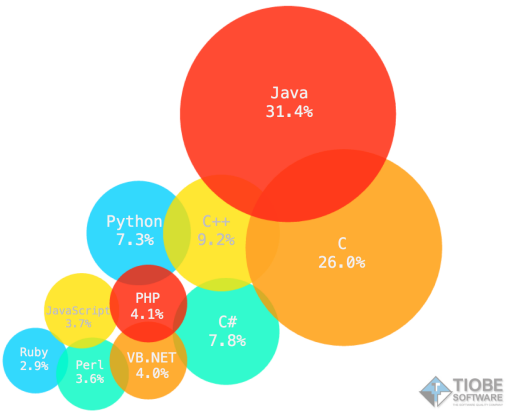


Gráfico de burbuja que muestra la popularidad de diferentes lenguajes de programación según el Indice TIOBE.
Imagen de Untapt con datos de TIOBE.

Pero, si las computadoras trabajaban con binario, ¿Cómo hacen para entender ese programa?. La respuesta radica en utilizar un programa. El programador genera un archivo con texto, similar al expuesto arriba, y lo guarda en el equipo. Luego, le indica a un programa especial, llamado **compilador**, que procese ese archivo para transformar el texto que contiene en una secuencia de unos y ceros que correspondan a un programa que realiza las tareas descriptas en el archivo. Al proceso mencionado, que realiza el compilador, se lo conoce como **compilar**.

El compilador transforma entonces una secuencia de unos y ceros que representan texto, en una secuencia de unos y ceros que representan un programa

El archivo que contiene el texto que describe el programa se conoce como **código fuente** del programa, mientras que el programa resultante se conoce como **código objeto** o **código compilado**.

Definición 4.12 Un **compilador** es un programa que transforma un archivo con código fuente en un archivo con código objeto, capaz de ser ejecutado en una computadora.

También existen algunos lenguajes que no trabajan de esta forma, sino que un programa va leyendo cada una de las líneas del programa, y lleva adelante la acción descripta para esa línea previo a leer la siguiente linea. Esto se conoce como **interpretación**, y el programa que realiza esa tarea se conoce como **interpete**.

Definición 4.13 Un **interprete** es un programa que ejecuta paso a paso cada uno de los comandos descriptos en un archivo con código fuente.

Simplificación
La realidad es que el funcionamiento de los compiladores e interpetes es bastante más compleja. Hoy en día muchos interpretes realizan pasos de pre-compilación, y muchos compiladores funcionan con técnicas JIT. También se está obviando el tema de máquinas virtuales, lo cual representa un componente muy importante en los procesos de ejecución de programas actuales.

Estos temas merecen un trabajo mucho más en profundidad, así como un bagaje de conocimientos mucho mayor al que el presente intenta proporcionar.

Es importante a tener en cuenta que, ni los compiladores ni los interpretes trabajarán con cualquier texto, sino que esperan texto escrito en un lenguaje específico. Es decir, el compilador del lenguaje C, no funcionará si se intenta procesar con el un archivo con código Python. Así, cada lenguaje tiene su propio compilador, o su propio interprete (o ámbos).

El lector puede o podría encontrar en diversa bibliografía terminos como "**lenguajes compilados**" o "**lenguajes interpretados**". Esta categorización responde simplemente al hecho de que, en general, el autor del lenguaje provee asociado a la defínción su lenguaje (gramática, semántica, etc.), la herramienta que permite generar programas en dicho lenguaje (la cuál suele ser o bien un compilador, o bien un interpete). Sin embargo, nada evita que un lenguaje que era inicialmente distribuido con un interprete pueda ser luego distribuido con un compilador, sin alterar el lenguaje en si. Esto quiere decir que la categorización anterior es incorrecta, pues ser compilado o interpretado no es una caracteristica del lenguaje, sino de la herramienta que lo acompaña.

Otra categorización que suele realizarse es la que diferencia lenguajes entre aquellos que son de **bajo nivel** y los que son de **alto nivel**, distinguiendo incluso en ocaciones con un **medio nivel**. Los lenguajes de bajo nivel suelen proveer comandos que están asociados a las operaciones que puede realizar el hardware del equipo. Así, para cada comando, seguramente haya un circuito específico dentro del equipo. En cambio los lengujes de alto nivel proveen construcciones más complejas, que no necesariamente se traducen a circuitos, y que están más ligadas a ideas o conceptos que el desarrollador puede querer expresar. Cuando el desarrollador realiza el programa, primero piensa la idea, la cual está expresada en lenguaje natural. Sin embargo, luego debe plasmar esa idea en el lenguaje de programación específico. Cuanto más parecido sea el lenguaje de programación a las ideas mentales del programador, más fácil será ese proceso.[14]

Una última categorización tiene que ver con la forma en la que el lenguaje permite estructurar ideas. Esto se conoce como el **paradigma** del lenguaje. Los primeros lenguajes de programación masivos utilizaban el **paradigma imperativo**, que consiste en secuencias de instrucciones dadas al equipo una tras otra. Existen otros paradigmas, como el **paradigma funcional**, más ligado a los desarrollos de teoría de la computación formal desarrollados por Alonzo Church, o la **programación orientada a objetos**, que utiliza procesos de abstracción conocidos como objetos para modelar los datos, y la comunicación entre ellos para modelar las transformaciones de esos datos. También existen otros paradigmas, más especificos y menos difundidos, y lenguajes que permiten escribir programas usando más de un paradigma al mismo tiempo.

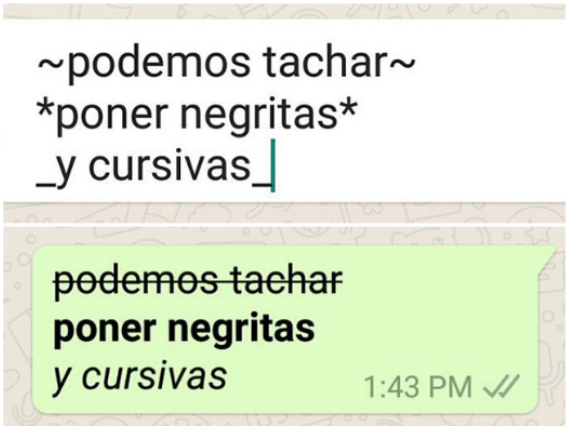
4.3.3 Lenguajes de marcado

Otro tipo de lenguajes formales son los **lenguajes de marcado**. Estos lenguajes tienen como función describir datos, donde "datos" puede ser cualquier cosa. Por ejemplo, existen lenguajes de marcado que describen posteos en foros, otros que describen páginas de wikis (como Wikipedia), otros que describen documentos, otros para imágenes, otros para páginas web, etc.

Recordemos que todas estas cosas podrían ser facilmente representadas mediante archivos con

diferente formato, esto es, con una codificación distinta del binario que respresenta el dato en el archivo. En los lenguajes de marcado, no se requieren diferentes formatos de archivo, sino que siempre se trabaja con texto plano, por lo que los datos **pueden ser editados con un editor de textos**.

La gracia es que el **texto contiene una serie de símbolos especiales, llamadas marcas** (de ahí el nombre de "lenguajes de marcado"). Estas marcas representan diferentes elementos semánticos dentro del documento, indicando por ejemplo, que una determinada parte se debe ver en negrita o en itálica, o que un determinado texto debería ser considerado un título, o que se debería insertar una tabla de datos en determinado lugar, o simplemente dando información semántica sobre la parte "marcada". Estas marcas pueden luego ser interpretadas luego por un visualizador, que las mostrará como elementos visuales, o ser utilizadas por un programa para realizar tareas especificas de acuerdo a las mismas.



Whatsapp trae incluido un lenguaje de marcas, que permite utilizar símbolos de texto simple al momento en el que el emisor redacta el mensaje para que el receptor visualice porciones del texto recibido con un formato particular.
Imagen propia.

Existen cientos de lenguajes de marcado, cada uno con un propósito distinto, o un enfoque distinto ante mismos propósitos. Cada lenguaje define entonces que símbolos representan marcas, y como deben ser utilizados en el texto, así como la asociación semántica de cada marca.[15]

Definición 4.14 Un **lenguaje de marcado** consiste en una codificación que incorporta en un archivo de texto plano que junto con el texto incorpora una serie de marcas o etiquetas que contienen información sobre la semántica, la estructura o la forma de presentar el texto.

Sabías qué

Un archivo de texto solo contiene información del texto que contiene, es decir de las letras que están escritas. No maneja ninguna otra información, como si las letras están en negrita, itálica o de algún color en particular. Sin embargo, muchos editores de texto diseñados para programar ofrecen una caracteristica conocida como "syntax highlighting" o "code styling", que colorea partes específicas del código, lo que le permite al desarrollador tener un panorama visual más ameno, facilitando su tarea.

Este libro utiliza coloreo en los lugares en donde se muestra código para hacer más fácil la comprensión del mismo.

Analicemos un ejemplo sencillo. El lenguaje de marcado de los **archivos de inicialización de Windows**, o archivos **.ini** consiste en texto que debe estructurarse en una serie de líneas. Cada línea consiste en un par de elementos, donde el primero, el de la izquierda, es conocido como "clave", y el segundo como "valor". Ambos elementos se encuentran separados con un sígno "=" entre ambos. Adicionalmente, un conjunto de líneas con pares clave valor pueden estar antecedidas con un nombre, colocado entre corchetes en la línea inmediatamente superior al conjunto, de forma

tal de agruparlas bajo un mismo identificador. Finalmente, cualquier línea que comience con el símbolo “’;” (punto y coma) corresponde a un comentario, y el texto de dicha línea no representa datos relevantes para el archivo, sino información para el desarrollador.[16]

Todas estas reglas pueden parecer complejas de entender, pero en realidad es uno de los lenguajes de marcado más sencillos que existe, aunque algunos autores consideran que es un lenguaje de configuración, más que de marcado.

Pero ¿Para qué sirve un archivo de inicialización de Windows? Bien, tanto el sistema operativo como diversas aplicaciones pueden utilizar el archivo para establecer preferencias del usuario. Por ejemplo, el código ficticio que se presenta a continuación podría utilizarse para configurar un navegador web:

■ **Ejemplo 4.8 contenido hipotético de un archivo .INI**

```
[Red]
; Poner UsarProxy=1 si no hay cortafuegos
UsarProxy=1
Proxy=192.168.0.1

[Preferencias]
PaginaInicio=https://unq.edu.ar
Maximizar=1
```

■

Las marcas en un archivo INI consisten en el símbolo “;”, que marca comentarios, o los caracteres “[” y “]” que determinan el nombre de un grupo, así como el signo “=” que separa la clave del valor en cada línea. El salto de línea, aunque no es un caracter visible, también es considerado una marca, porque brinda información (termina un par clave valor y comienza otro).

Los archivos de configuración de Windows son un formato de archivo sencillo, pero hoy en día ya obsoleto.

Otros lenguajes de marcado incluyen **JSON** o **XML**, los cuales son utilizados para intercambios de datos en la web o configuración de aplicacones más modernas. A continuación se deja un ejemplo sencillo de **XML**:[17]

■ **Ejemplo 4.9 Una receta de cocina codificada en un archivo XML**

```
<Receta>
    <Nombre> Banana con Dulce de Leche </Nombre>
    <Descripcion> El mejor postre </Descripcion>
    <Ingredientes>
        <Ingrediente> Banana </Ingrediente>
        <Ingrediente> Dulce de Leche </Ingrediente>
    </Ingredientes>
    <Utencilios>
        <Utencilio> Cuchara </Utensilio>
    </Utencilios>
    <Pasos>
        <Paso>Pelar la banana</Paso>
        <Paso>Destapar el frasco de dulce de leche</Paso>
        <Paso>Colocar la cuchara en el frasco</Paso>
```

```
15          <Paso>Tomar una gran cucharada de dulce de leche</Paso>
16          <Paso>Untar el dulce de leche en la banana</Paso>
17          <Paso>Comer la banana</Paso>
18          <Paso>Repetir el proceso si fuera necesario</Paso>
19          <Paso>Cerrar frasco y lavar cuchara</Paso>
20      </Pasos>
21  </Receta>
```

■

El archivo anterior describe una receta de cocina de un típico postre argentino. El lenguaje XML contiene como marcas un elemento conocido como **etiqueta**. La etiqueta consiste en un nombre colocado entre los caracteres "<" y ">", si es de apertura, o "</' y ">" si es de cierre. Cada etiqueta puede contener, o bien texto, o bien otras etiquetas.

XML permite definir los nombres de las etiquetas, y estructurarlas como se desee. Por eso, este lenguaje es ampliamente utilizado para describir información. XML es sumamente flexible, y es considerado el padre de decenas de lenguajes de marcado. Por ejemplo, se utiliza para definir documentos y planillas de cálculo en los formatos de documento libre ODT y ODS (archivos similares a los de Word y Excel), como así también para describir imágenes en el formato SVG. La diferencia de estos lenguajes de marcado particulares con XML radica en que no cualquier etiqueta es válida, y no pueden tampoco anidarse de cualquier forma, sino que deben cumplir una serie de restricciones adicionales.

Dentro de los lenguajes que descienden de XML se encuentra **HTML**, o **HyperText Markup Language**, en español "lenguaje de marcado de hipertexto". Este es tal vez el lenguaje de marcado más popular y más vastamente utilizado de la historia. Esto se debe a que HTML es el lenguaje con el que se **describen las páginas web**. Los elementos de una página web son estándarizados, y están dados por las etiquetas que el lenguaje define. Por ejemplo, la etiqueta "<a>" define un enlace a otro sitio web, o la etiqueta "<img>" permite insertar una imágen.

Luego, un programa especializado, conocido como **browser** o **navegador web**, actúa como visualizador de estos archivos, mostrando los elementos visuales que el autor haya escrito, sin mostrar las etiquetas.

La combinación de las distintas etiquetas, estructuradas de diversas formas, puede dar lugar a las más diversas y creativas páginas web que uno pueda desear. HTML se combina con otros lenguajes, como CSS, y JavaScript, para lograr sitios web interactivos y con grandes caracteristicas visuales. Ejemplos y detalles de este lenguaje se pueden ver en el apendice B.

> **Sabías qué**
> Casi todos los **browsers** permiten ver el **código fuente** de la página web que el usuario está navegando. No solo eso, sino que algunos incluyen herramientas que sirven a los desarrolladores para analizar, escribir, y reparar las páginas que escriben, permitiendo incluso modificar el código de páginas remotas de forma local y temporal.

Markdown es otro interesante lenguaje de marcado. Es ideal para escribir pequeños documentos informativos, como puede ser documentación, o correos electrónicos. La ventaja fundamental de Markdown radica en que no se requiere de un visualizador especializado para comprender el documento, sino que el mismo código fuente permite visualizar el contenido de forma clara. Un pequeño resumen del lenguaje se puede ver en el apendice A.

4.4 Modelos de comercialización de software

4.4.1 Copyright

Debe comprenderse que el software hoy en día goza de **derechos de autor** o **copyright**. Es decir que el autor de un programa goza de todos los derechos legales de explotación del mismo. Es decir, el autor puede determinar cosas como la forma en la que el software se vende, alquila o licencia, o cosas como para que fines puede ser utilizado su programa, si puede ser utilizado como parte de otros programas, si puede ser utilizado comercialmente, etc.

Definición 4.15 El **derecho de autor** (o **copyright**, por su voz inglesa) consiste en un conjunto de normas juridicas que defienden el derecho moral de los autores de obras literarias, artisticas, musicales, cientificas, didacticas y de software, ya sea publicada o inédita, para explotar dicha obra comercialmente en cualquier forma que considere por un periodo de tiempo determinado.[18]

El modelo actual de comercialización de software nace en la época de 1970, con la aparición de las primeras microcomputadoras. Los fabricantes de estas primeras computadoras instalaban software de fabrica para venderlo como un valor agregado a sus equipos. Sin embargo, estos programas solo funcionaban en la máquina del fabricante, y por tanto uno quedaba atado a este para mantener compatibilidad de los productos. Además esos productos solían ser de baja calidad.

Con la masividad de las computadoras, se vuelve posible la venta de software como un producto o servicio de terceros, separado del fabricante. Microsoft es una de las empresas que promocionó este nuevo modelo de negocio, ofreciendo software de calidad a sus usuarios. Así, Microsoft Basic se volvió uno de los programas practicamente inprescindibles a tener en un equipo a comienzos de 1980.

Oficinas de Microsoft en Barcelona.
Imagen de T Magazine (The New York Times Style Magazine: España).

Si bien un desarrollador puede optar por vender su software, el modelo de comercialización más común hoy en día es el modelo de **licenciamiento**. Es decir, quien adquiere un sistema, no lo compra de la misma forma que se adquiere un elemento físico, sino que lo que se adquiere es el **derecho a uso del sistema**. En ese sentido, el autor, llamado el **licenciante**, le otorga al comprador,

el **licenciatario**, el derecho de uso del programa. La **licencia** consiste entonces en un contrato legal entre estas partes, que especifica cosas como la forma en la que se puede utilizar el sistema, la forma en la que puede distribuirse, etc.

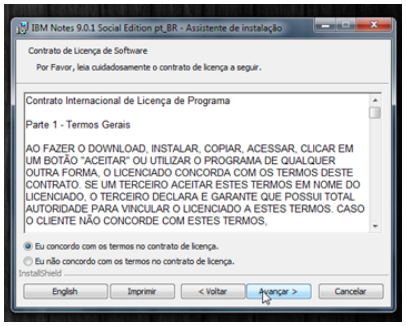
Definición 4.16 Una **licencia de software** es un contrato legal entre quien tiene los derechos de explotación del software (**licenciante**) y quien suscribe para un uso determinado del mismo (**licenciante**).

Romper el contrato de licencia, por ejemplo, usar un programa para un fin para el cual no se tiene permiso, o descargarlo de un sitio que no está oficialmente habilitado para su distribución, es **ilegal**. La descarga y distribución de programas de forma ilegal se conoce como **piratería**. Por ejemplo, descargar y utilizar Microsoft Windows sin haber pagado los canones de licencia oficiales es considerado **piratería**.

Definición 4.17 La **pirateria de software** consiste en el incumplimiento del contrato legal que establece el propietario de los derechos de autor del software, tal como el uso indebido, descarga o adquisición en lugares no autorizados, redistribución del programa sin el consentimiento apropiado, prácticas de ingeniería inversa, modificaciones o alteraciones al programa original, entre otras.

En general, el licenciatario paga un canon o tarifa al momento de adquirir la licencia. Al pagarlo, adquiere no solo el derecho a uso del sistema, sino en general el derecho a recibir las actualizaciones que aparezcan del mismo (muchas veces dentro de cierto rango de tiempo).

Algunos licenciantes ofrecen el software de forma gratuita para uso dentro de cierto límite de tiempo (por ejemplo, 10 días de prueba del software). Esto permite a los potenciales compradores evaluar el producto previo a su compra, a la vez que posiciona potencialmente al desarrollador de una mejor forma frente al mercado. También puede suceder que la versión ofrecida de esta forma tenga limitaciones en su utilización, recortando caracteristicas escenciales. Este tipo de software se conoce como **shareware**.



Contrato de licencia de IBM Notes 9.0.1. Al presionar "Aceptar" el usuario está firmando un documento con validez legal, y cuyo incumplimiento puede llevarlo a que la contraparte lo demande. Imagen de Lhreis.

Definición 4.18 El **shareware** consiste en software que se ofrece al usuario final de forma gratuita por un periodo de tiempo, o con características límitadas, con la intención de que el usuario pueda rpobar el producto previo a la adquisición del mismo.[19]

Otros licenciantes pueden ofrecer el software de forma completamente desinteresada, y por tanto gratuita. Este tipo de software se denomina **freeware**. No hay que confundir este término con software libre, que es un concepto completamente distinto. El freeware puede ofrecerse bajo ciertas restricciones, por ejemplo, es gratuito, pero no si se usa comercialmente, o es gratuito, pero no puede distribuirse de otra forma que no sea mediante la tienda oficial del autor.

Definición 4.19 El **freeware** consiste en software que se ofrece al usuario final de forma gratuita, bajo un licencia que puede o no contener restricciones.[20]

En general, el desarrollador solo ofrece el programa final en forma de **código objeto**, o dicho de otra forma, se ofrece el programa compilado, pero no su código fuente. Aquellos programas que adicionalmente ofrecen acceso al código fuente suelen denominarse **open source** o **software libre**, algo que trataremos en la siguiente sección.

Con la masificación de Internet, es cada vez más común encontrar sistemas que funcionan en línea, desde un navegador web. El licenciante puede optar por ofrecer el servicio con un modelo de alquiler, cobrandole al licenciante de forma mensual o anual por el servicio, o ofrecerlo de forma gratuita y solventando los gastos mediante la inserción de publicidad. Este modelo se conoce como **software como servicio (SaaS)**.

Otra modalidad muy común es el **software a medida**, donde el desarrollador no realiza un producto generico para distribuir al mercado, sino que realiza un sistema especialmente diseñado para un cliente particular, a pedido de este. Dependiendo del contrato el mismo puedo incluir solamente el código compilado, o también el código fuente, manejandose en general costos más elevados para este último. Junto con ello, suelen pautarse jornadas de capacitación para los usuarios del sistema, manuales y un periodo de soporte o garantía que puede pagarse junto con el desarrollo, o pagarse mensualmente luego de finalizado el mismo.[21]

4.4.2 Software Libre

Previo al actual modelo comercial del software, este era algo que se compartía entre los cientificos e ingenieros en las universidades. La historia cuenta que Richard Stallman, un desarrollador de software se encontró ya a fines de los 70 trabajando en una empresa donde múltiples equipos estaban conectados en red con una única impresora. La impresora tenía el problema de trabarse de forma habitual, sin notificar al usuario del problema y este solo se se enteraba de que había ocurrido un error cuando se acercaba a la impresora, la cual se encontraba en la otra punta de la oficina. Stallman quería solucionar el problema, y solicitó al fabricante de la impresora que le proporcionara el software de la misma. La empresa se negó a entregar el software, incluso cuando Stallman ofreció trabajar gratis para mejorar las caracteristicas del mismo.

Stallman entendió que **el modelo de licenciamiento y comercialización de la época limitaba la libertad de los usuarios, que quedaban atados a la voluntad del fabricante**. Esto, según él, limitaba el progreso del softare en general, y atentaba contra la libertad de las personas y la sociedad en su conjunto. Añorando el modelo de los inicios del software, anunció en 1983 que iba a comenzar a trabajar en desarrollar un sistema operativo completamente libre, en donde los usuarios no estuvieran limitados a ataduras por parte del licenciante. En 1985 funda la **Free Software Foundation**, una fundación sin fines de lucro con la intención de llevar adelante su proyecto.

También enuncia el concepto de **software libre**. Un software, para Stallman, es libre si cumple con cuatro características que el denomína “libertades”.

- Libertad 0: Ejecutar el programa comos e quiera y para el fin que se desee.
- Libertad 1: Posibilidad de estudiar el código fuente del programa y modificarlo a medida para solucionar las necesidades propias.
- Libertad 2: Distribuir copias exactas del programa, tal cual lo distribuye el fabricante a quien se desee.
- Libertad 3: Distribuir copias modificadas del programa a quien se desee.

Para cumplir estas libertades, todo software que se considere libre **debe garantizar el acceso al código fuente** a cualquier usuario. De lo contrario no sería posible llevar adelante las libertades 1 y 3. El software que no garantiza estas propiedades se conoce como **software propietario** (porque tiene un propietario, a diferencia del libre que le pertenece a la comunidad) o **software privativo** (porque priva al usuario de sus derechos).[22]

Definición 4.20 El **software libre** es aquel que cumple con las cuatro libertades enunciadas por Richard Stallman, y que permite al usuario utilizar el programa, estudiarlo, compartirlo, modificarlo y compartir copias modificadas. Para ello el software libre debe permitir acceder a su código fuente.

Definición 4.21 El **software propietario, o software privativo** es todo aquel software que viola alguna de las libertades enunciadas por Richard Stallman.

El Festival Latinoamericano de Instalación de Software Libre (FLISoL) es un evento en el que usuarios de software libre ayudan a quienes se acercan a reemplazar el software privativo de sus equipos por software libre. El evento también cuenta con charlas de divulgación o técnicas y otras actividades. Se realiza en diversas sedes de múltiples paises al mismo tiempo una vez al año. La imágen corresponde al evento realizado en el 2017 en la ciudad de Bariloche.
Imagen de Cristian Caravá, Jose Gimenez, Mirian Ripoll, Ariel Kennedy, Andrés Grad Gut, Stefan Ronacher, Lucas Passalacqua y Pedro Wood

Estas libertades permiten teoricamente que los usuarios sean quienes tienen el poder en el sistema, en lugar del autor del software. También crea la licencia **GPL (General Public Licence)**, un modelo de licenciamiento de software que reconoce al autor, a la vez que mantiene la garantía de estas libertades. La licencia **GPL** se considera una licencia **Copyleft**, concepto también creado por Stallman en contraposición al copyright, y que estipula que toda obra derivada de un producto con licencia copyleft debe necesariamente incluir la misma licencia. Así, uno no puede simplemente tomar un programa con licencia GPL, modificarlo y comercializarlo como propio con una licencia distinta a la GPL. Esto conlleva que las modificaciones realizadas por cualquier usuario terminan siempre siendo beneficiosas a la sociedad toda.

Otras licencias aparecen a posteriori, como la licencia **MIT**, o la **BSD**, que también son

consideradas licencias libres. Sin embargo estas no son licencias copyleft, permitiendo a otros tomar el código y empaquetarlo como un producto comercial. El código fuente de **Microsoft Windows**, por ejemplo, posee código fuente original del sistema operativo **BSD**, que posee la licencia del mismo nombre.

El software libre se corresponde a una ideología, ligada al humanismo y las corrientes de izquierda socialistas. El **fin último es el avance social, y no la comercialización del software**. Sin embargo, esto no evita que diversas empresas puedan dedicarse al software libre como emprendimiento rentable. En general **el modelo productivo del software libre implica la venta de soporte a terceros que utilizan el sistema, aunque también puede venderse el software** (por ejemplo, vender la versión compilada, aunque se deja libre el código fuente. Como no todos tienen los conocimientos técnicos para compilar el código, muchas personas pagan por la comodidad de no tener que aprender a hacerlo).

El proyecto de Stallman de crear un sistema operativo completamente libre nunca logro completarse. Si bien se crearon sendas herramientas que acompañan a un sistema operativo, la parte fundamental que maneja el hardware, conocido como kernel, jamas terminó de realizarse. En 1991, un joven finés llamado **Linus Torvalds** compartió en Internet su intención de llevar adelante el desarrollo de un kernel para un sistema operativo clón de UNIX que corriera en computadoras de escritorio, y compartió libremente su código fuente. Contactado por Stallman y cenvencido del movimiento de software libre, decidió licenciar su software con GPL, bautizandolo Linux. Este sistema operativo se ha vuelto hoy en día el sistema operativo libre más popular del mundo.

4.4.3 Open Source

Muchas empresas encontraron al modelo de software libre altamente beneficioso. Al publicar el código fuente cientos de desarrolladores pueden contribuir a mejorar el programa realizado, y lo hacen de forma desinteresada y gratuita. Esto implica menores costos para las empresas que llevan adelante este proceso, pues no deben mantener el sistema costeando desarrolladores, sino que es la comunidad la que lo mantiene.

Las empresas suelen tomar el software realizado como libre y empaquetarlo para la venta, en ocaciones agregando funcionalidades adicionales que no se encuentran disponibles en la versión libre, y que vuelven al sistema en otro más potente o más apetecible por los usuarios. En general se ofrecen los sistemas con un doble licenciamiento, comercial, por un lado, y libre, por otro.

Linus Torvalds, creador de Linux, en la conferencia LinuxCon Europa 2014 en la ciudad de Düsseldorf.
Imagen de Krd.

Así, diversas empresas comenzaron a abrir su código, pero no por una visión filantropica, sino por una cuestión comercial y económica. El software que se enmarca en este modelo se conoce como **Open Source**. En la práctica, el software open source y el software libre comparten las mismas características, pero la filosofía que subyace a ambas es completamente distinta. La **Open Source Initiative** fue creada como una ONG con la intención de regular las licencias que son consideradas open source, y que incluye a las licencias libres, pero agrega otro conjunto de licencias que violan en ocaciones las libertades básicas del software libre. Así, todo software libre es open source, pero no a la inversa.

4.4.4 Demistificación de frases sobre el FOSS

Al conjunto de sobre libre con software open source se lo suele denominar **FOSS** (Free Open Source Software).

A diferencia de lo que uno podría pensar, **software libre no significa software gratuito**. Si bien es cierto que gran parte del software libre se ofrece de forma gratuita, la realidad es que existe software gratuito que no es libre. Más aún, existe software libre que no es gratuito.

Durante los años han sido varias las empresas que han trabajado en el desarrollo de software, y algunas de ellas, como Microsoft, Apple o IBM han alcanzado gran éxito comercial gracias al software privativo y a prácticas de negocio que implicaban el aprisonamiento de los usuarios a un sistema, el monopolio y el lobby político. No es de extrañar que Richard Stallman y muchos otros seguidores del movimiento software libre vean a estas empresas como "malvadas". Sin embargo estas empresas solo hacen lo que toda empresa debe hacer, intentar generar ganancia a sus inversionistas.

Sin embargo, también son varias las empresas que desarrollan y promueven el software libre o al menos el software open source. Google por ejemplo ha publicado numerosas herramientas para desarrolladores que se distribuyen como libres. En particular, el sistema operativo Android es un sistema operativo libre, basado en Linux. Facebook y Twitter también lo hacen. Los moelos de negocio de estas empresas pasan por la venta de servicios y publicidad, y no por la venta de software.

Otras empresas que han desarrollado múltiples FOSS incluyen a la ya desaparecida Sun Microsystems, Novell, Red hat y Canonical. Estas empresas proveen servicios empresariales alrededor del software libre que construyen, dandole la posibilidad al usuario de elegir si desea pagar por profesionales capacitados por la empresa para mantener el software de acuerdo a sus necesidades, o desea capacitar su propio personal. También brindan capacitaciones y certificaciones a profesionales.

Es decir, es falso que quien hace FOSS no va a ganar dinero, o va a entregar un producto gratis. Existen formas múltiples de ganar dinero con el FOSS.

Muchas veces se realizan comparaciones entre el software libre y el software privativo. En general las comparaciones suelen estar guiadas con intereses particulares de uno u otro grupo. A continuacion hay algunas de las proclamas que muchas veces se escuchan a favor y en contra del software libre en estas comparaciones, y algunas demistificaciones de las mismas:

El software libre es más barato. Si bien puede ser cierto en la mayoría de las ocaciones, algunos productos de software libre, se deben comprar. Por ejemplo, Red Hat Enterprise Linux, puede llegar incluso a ser más costoso que su contrapartida privativa en determinadas configuraciones, incluyendo soporte y otros.

El software libre corre en todas las plataformas. Completamente falso. Un programa FOSS solo corre en la plataforma para la cual fue diseñado, al igual que un sistema privativo. Grupos de desarrolladores independientes que gustan de la aplicación suelen portar la misma a diferentes plataformas, pero no es seguro que esto ocurra con cualquier programa.

El software libre es más seguro. La seguridad nada tiene que ver con lo libre o lo privativo. Se dice que Linux suele ser más seguro que Windows, y esto se usa de palanca para el alegato mencionado, pero en otros programas esto no es necesariamente así. La única ventaja realmente visible tiene que ver con el hecho de poder verificar el código fuente del softare libre, que nos permite asegurarnos qué es lo que el programa realiza realmente con nuestros datos, mientras que el software privativo no lo permite.

El software libre es de menor calidad. Esto también es falso. Es cierto que en el software priva-

tivo grandes empresas colocan mucho dinero, con la intención de obtener un buen producto que puedan comercializar, mientras que el software libre se basa en el apoyo de algunos pocos desarrolladores que lo hacen en su tiempo libre. Sin embargo, como ya mencionamos, muchas empresas lucran también con el software libre, por lo que están dispuestas a invertir recursos en este. En general proyectos bien establecidos y con grandes bases de usuarios suelen tener una muy buena calidad, incluso a veces mayor que la de sus contrapartes privativas.

El software libre no requiere actualizaciones Todo el software tiene, y muchas veces requiere actualizaciones. A medida que el hardware se actualiza, el software debe actualizarse para soportar dicho hardware, lo mismo ocurre con programas que corren en un determinado sistema operativo, si este último cambia, puede que el programa deba cambiar. Además los desarrolladores actualizan sus programas dotandolos de nuevas funcionalidades, corrección de fallos y otros. Libre o privativo nada tiene que ver en el asunto.

El software libre no brinda soporte técnico En parte esto es cierto. En general el software libre se distribuye bajo un concepto "as is" (como está), que significa que el autor no se hace responsable por fallos o errores que tenga el programa, ni por los problemas que este pueda ocacionar. Sin embargo, el soporte existe, ya sea a traves de foros o sitios de la comunidad de desarrolladores y usuarios o a traves de servicios pagos que uno puede contratar. Los sistemas privativos en general tampoco ofrecen soporte, o se cobra un plus para poder acceder al mismo.

Probablemente estas no sean las únicas cosas que se dicen con poco criterio en el ámbito del software libre, sino que hay muchas otras. Entender como funcionan las computadoras y el software en particular permite discriminar mejor los postulados claramente erroneos de aquellos que no lo son, así como comprender los beneficios y las problemáticas de utilizar software libre.

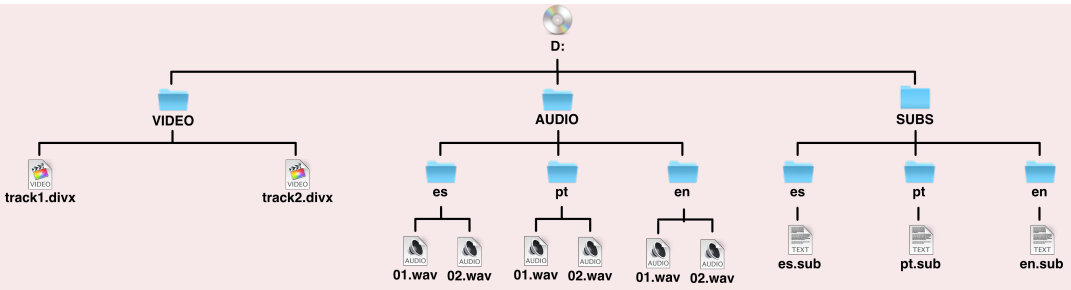
4.5 Actividades

Ejercicio 4.1 Valiendose de internet, determine para cada uno de los nombres completos de archivo a continuación, de qué formato de archivo se trata (imagen, audio, texto, programa ejecutable, etc.).

a) ACDC.mp3
b) LEEME.txt
c) 2018_06_15_221653.jpg
d) run.py
e) index.php
f) cv.docx
g) materias.xls
h) subtitles_tbbt_s01e04.zip
i) script.sh
j) showcase.odp
k) cuadernillo.pdf
l) day_of_the_tentacle.exe
m) bumbleebee.wav
n) library.c
ñ) configuration.xml
o) logo.png
p) LEEME.md
q) casa.dwg

■

Ejercicio 4.2 Teniendo en cuenta el siguiente sistema de archivos de Windows que corresponde a un DVD de una película, se pide.



Se pide que escriba las siguientes rutas absolutas:

a) Al archivo de subtitulos en inglés (en.sub)
b) Al archivo de audio de la pista 02 en español (carpeta es, archivo 02.wav)
c) Al archivo de video de la pista 01 (track1.divx)
d) Al archivo de audio de la pista 01 en portugues.

■

Ejercicio 4.3 Teniendo en cuenta el sistema de directorios del ejercicio anterior y sabiendo qué:

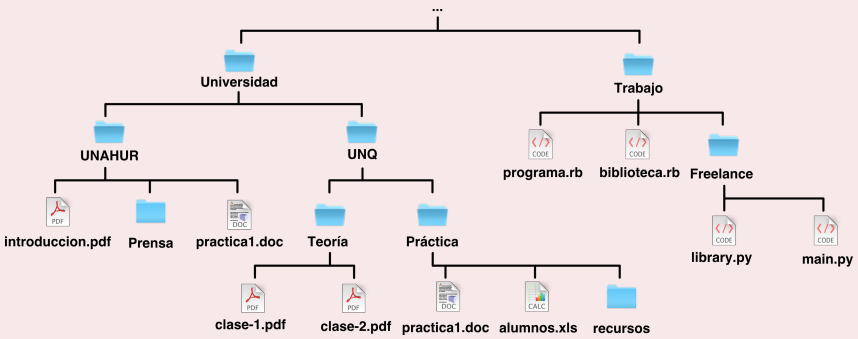
- Todo archivo de subtitulos pesa 500 KB.
- Los archivos de audio de la pista 01 pesan 5 MB.
- Los archivos de audio de la pista 02 pesan 25 MB, menos el de idioma portugués que pesa 27 MB.
- El archivo de video de la pista 01 pesa 340 MB
- El archivo de vide de la pista 02 pesa 660 MB.

Se pide determine las siguientes:

a) Cuanto pesa la carpeta VIDEO
b) Cuanto pesa la carpeta SUBS
c) Cuanto pesa en total el disco D:

■

Ejercicio 4.4 Teniendo en cuenta la siguiente porción de un sistema de archivos de un sistema Linux:



Describa la ruta absoluta a las carpetas que se encuentren vacías, asumiendo que la carpeta superior es la raíz del sistema. ■

Ejercicio 4.5 Utilizando el mismo sistema de archivos del ejercicio anterior, escriba las siguientes rutas relativas:

a) Desde la carpeta más externa de la jerarquía hasta el archivo main.py
b) Desde la carpeta más externa de la jerarquía hasta el archivo alumnos.xls
c) Desde la carpeta UNQ hasta el archivo clase-2.pdf
d) Desde la carpeta UNAHUR al archivo introduccion.pdf
e) Desde la carpeta UNQ al archivo introduccion.pdf
f) Desde la carpeta Teoría en UNQ al archivo programa.rb

■

Ejercicio 4.6 Un programa "hola mundo" consiste en código que simplemente imprime en la pantalla las palabras "hola mundo". Este tipo de programa se ha hecho popular como una forma de poder dar a los programadores una vista minima y rápida de la sintaxis de un lenguaje, y en cursos que se enfocan en la enseñanza de lenguajes de programación en lugar de sus conceptos, es muy común encontrarlo como una de las primeras muestras de código o actividades a realizar.

Se pide entonces que busque en internet el código de un programa "hola mundo" para los siguientes lenguajes de programación. Puede encontrar una gran colección en el sitio http://helloworldcollection.de

a) Python
b) Lisp
c) C
d) Java
e) Assembly ARM
f) Assembly z80, Console

¿Qué reflexiones puede realizar despues de ver los diferentes códigos? ■

Ejercicio 4.7 El sitio web https://alternativeto.net permite a sus usuarios buscar un programa por nombre, y brinda alternativas a ese programa con funcionalidades equivalentes o similares. Dentro de los resultados, se puede filtrar por sistema operativo y por el tipo de licencia (comercial, gratuita o libre).

Se pide que busque al menos una alternativa que sea software libre y/o open source para cada uno de los siguientes programas:

a) Windows 10
b) Microsoft Office Suite
c) Adobe Acrobat Reader
d) Internet Explorer
e) Adobe Photoshop
f) Autodesk AutoCAD
g) Age of Empires II

■

Referencias

[1] Charles Petzold. *Code: The Hidden Language of Computer Hardware and Software*. Microsoft Press, 2000 (véase página 57).

[2] Noam Nisan y Shimon Schocken. *The Elements of Computing Systems: Building a Modern Computer from First Principles*. MIT Press, 2005 (véase página 58).

[3] Ron White y Tim Downs. *How Computers Work: The Evolution of Technology*. 10.ª edición. How it works. Que Publishing, 2015 (véanse páginas 58, 63, 64).

[4] Philip A. Laplante. *Dictionary of Computer Science, Engineering and Technology*. CRC Press, 2000 (véanse páginas 60, 65).

[5] ASCII Corporation y Nippon Gakki Co. LTD. *V9938 MSX Video: Technical Data Book Programmer's Guide*. Yamaha Nippon Gakki Co. LTD., 1985 (véase página 61).

[6] Dan Gookin. *PCs For Dummies*. 10ª ed. For dummies. Hobooken, Nueva Jersey: Wiley, 2005 (véanse páginas 70, 73).

[7] Linfo Team. «Filename Extension Definition». En: *The Linux Information Project* (2018). URL: `http://www.linfo.org/filename_extension.html` (véase página 71).

[8] FOLDOC. «Absolute Path». En: *Free Online Dictionary of Computing* (2018). URL: `http://foldoc.org/absolute+path` (véase página 74).

[9] FOLDOC. «Relative Pathname». En: *Free Online Dictionary of Computing* (2018). URL: `http://foldoc.org/relative+pathname` (véase página 77).

[10] M. Nottingham. «RFC 7320». En: *Internet Engineering Task Force (IETF)* (2014). URL: `https://tools.ietf.org/html/rfc7320` (véase página 79).

[11] Richard Littaur. *The Dothraki Language Dictionary*. Web published, 2017 (véase página 81).

[12] K. V. N. Sunitha y N. Klyani. *Formal Languages and Automata Theory*. Pearson Education India, 2015 (véase página 82).

[13] Robert W. Sebesta. *Concepts of programming languages*. 7.ª edición. Addison Wesley, 2005 (véase página 83).

[14] Richard L. Wexelblat. *History of programming languages*. ACM monograph series. Nueva York: Academic Press, 1981 (véase página 84).

[15] James H. Coombs, Allen H. Ranear y Steven J. DeRose. «Markup Systems and the Future of Scholarly Text Processing». En: *Communications of the ACM*. Universidad de Michigan, Detroit, Estados Unidos: ACM, 1987 (véase página 85).

[16] Microsoft. «GetPrivateProfileString function». En: *Microsoft Developer Network* (2018). URL: `https://msdn.microsoft.com/en-us/library/windows/desktop/ms724353(v=vs.85).aspx` (véase página 86).

[17] Beneit Marshal. *XML by example*. QUE, 2000 (véase página 86).

[18] Copyright. *Oxford's dictionary*. `https://en.oxforddictionaries.com/definition/copyright`, 2018 (véase página 88).

[19] Shareware. *Merriam-Webster's dictionary*. `https://www.merriam-webster.com/dictionary/shareware`, 2018 (véase página 89).

[20] Freeware. *Merriam-Webster's dictionary*. `https://www.merriam-webster.com/dictionary/freeware`, 2018 (véase página 90).

[21] Kul Bhushan C. Saxena, Swanand J. Deodhar y Mikko Ruohonen Ruohonen. *Business Model Innovation in Software Product Industry: Bringing Business to the Bazaar*. Management for Professionals. Springer India, 2017 (véase página 90).

[22] Richard M. Stallman y Joshua Gay. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Paramount, California: CreateSpace Independent Publishing Platform, 2009 (véase página 91).

III

Lógica

IV

Programación

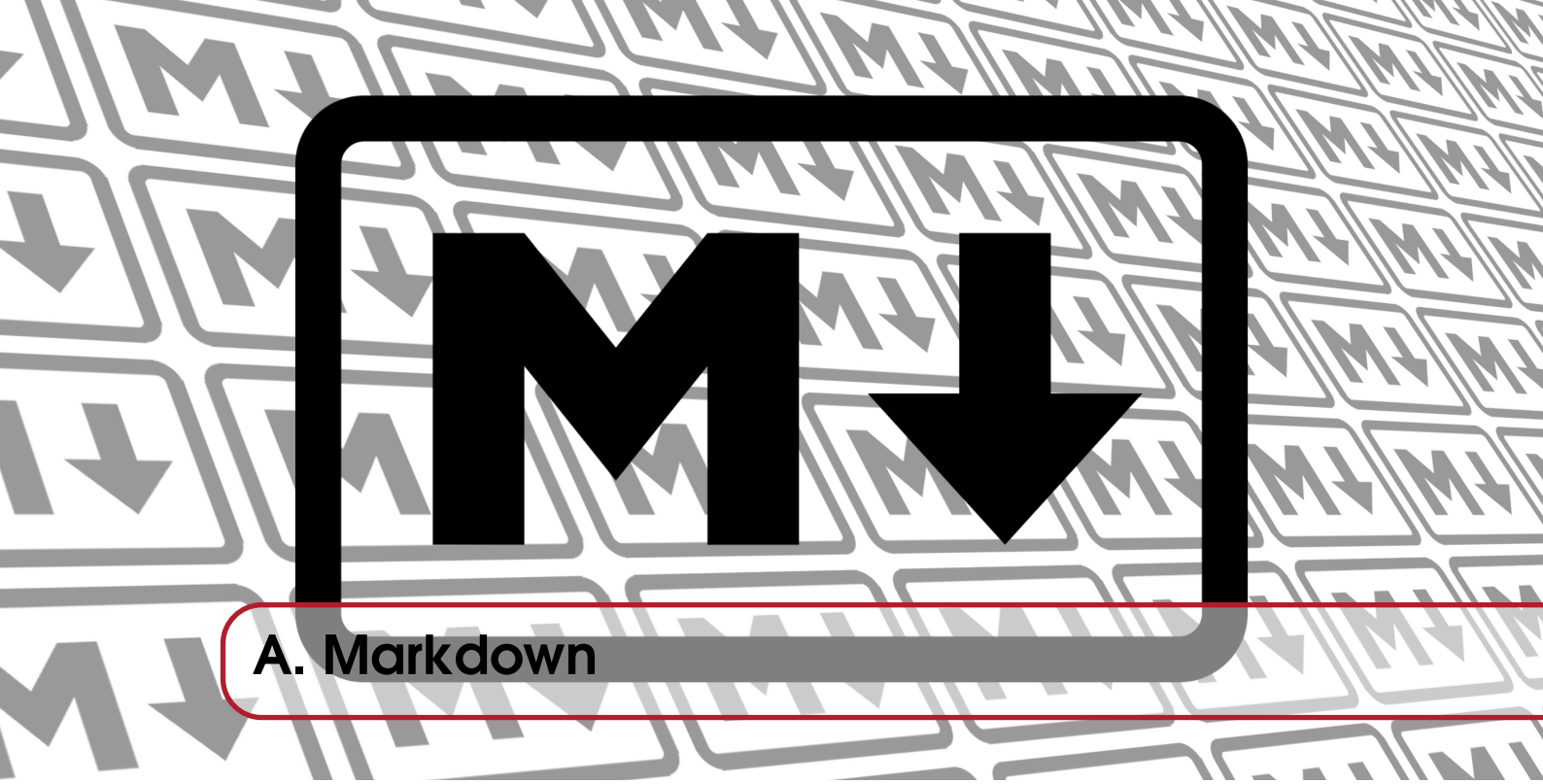
V

Índice y Apendices

Índice alfabético

Imagen de la supercomputadora Thunderbird, en el Sandia National Laboratory..
Fotografía del Departamento de Energía de los Estados Unidos.

Appendices



A. Markdown

Logo de Markdown sobre fondo con logos tridimencionales.
Diseño propio.

Markdown es un lenguaje de marcado ligero creado por **John Gruber**, con ayuda de **Aaron Swartz**, que trata de conseguir la máxima legibilidad y facilidad de publicación tanto en su forma de entrada como de salida. Gruber deseaba que la gente pudiera escribir un documento de forma sencilla, y que pudiera comprenderlos, sin necesidad de saber demasiado sobre el lenguaje, y aún así producir documentos válidos que pudieran ser exportados luego a documentos HTML válidos.

Si bien su lanzamiento oficial fue en el 2004, se inspira en una gran cantidad de convenciones existentes para marcar mensajes de correo electrónico en texto plano, muy utilizadas en la década de 1980 y principios de 1990. Markdown es un lenguaje libre, y también lo son sus herramientas, que se distribuyen bajo una **licencia BSD**. La versión original fue creada en el lenguaje de programación Perl, pero con el tiempo se ha portado a muchos otros lenguajes.

No hay un estándar definido para Markdown, aparte de la implementación original de John Gruber, que algunos consideran obsoleto. Esto ha ocasionado una gran fragmentación, con diversas implementaciones que agregan funciones, o remueven otras.

Markdown se ha posicionado como un importante lenguaje en diversos ámbitos, por ejemplo, varios sistemas de blogs permiten escribir articulos utilizando Markdown.

En el área de programación es ampliamente utilizado para escribir documentación de los programas, habiendose convertido en el estándar de facto. Por ejemplo, sitios que albergan repositorios de código, como **GitHub**, **GitLab** y **Bitbucket** entre otros, incluyen visualizadores para Markdown que muestran la documentación del programa de forma automática. Cómo estos sitios albergan proyectos de software libre, leer la documentación de estos programas es una buena forma de aprender Markdown, ya que se puede ver fácilmente la visualización y su código fuente.

También cabe destacar que, si bien se puede escribir código Markdown con cualquier editor de texto simple, existen editores especializados o plugins para editores que permiten visualizar la salida como HTML, resaltan la sintaxis y proveen funciones de autocompletado. También existen editores con visualizadores que corren directamente online, sin necesidad de instalar nada en el equipo, como el sitio web http://dillinger.io/.

Los archivos que contienen código markdown suelen guardarse con la extensión **.md**, o con extensión **.markdown**, según la herramienta que se utilice para producirlos.

Cabe destacar que Markdown permite exportar el contenido a HTML, un lenguaje de marcado que se utiliza para diseñar páginas web. Sin embargo, sus caracteristicas están muy reducidas con respecto a este último, por lo que el diseño de páginas no es su función. En cambio, está pensado para redactar correos electrónicos, articulos simples y documentación.

A.1 Sintaxis y Semántica

Párrafos y texto simple

Todo texto escrito en un documento Markdown que no contenga marcas específicas aparece tal cual, como un párrafo. Un salto de línea en el texto original no se ve reflejado en el resultado visualizado.

Los saltos de línea en la salida se dan cuando se insertan dos espacios consecutivos en la entrada (donde un salto de línea es considerado un espacio).

```
Esto es un párrafo único, incluso
si hay un salto de línea en el medio.

Esto aparece en otro párrafo, pues hay dos
saltos de línea desde el párrafo anterior.
```

Énfasis

Existen dos formas de enfatizar una palabra en Markdown, un énfasis simple (que en general se visualiza como texto en itálica), y un énfasis doble (para palabras o frases un poco más importantes que las anteriores, y que termina visualizandose en negrita).

El énfasis consiste en colocar una marca (para énfasis simple) o dos (para énfasis doble) al inicio del texto a resaltar, y la misma cantidad al final de dicho texto. Pueden utilizarse tanto guines bajos como asteriscos para marcar, pero en general se estila utilizar guión bajo para énfasis simple, y asteriscos para negrita.

```
Esto texto _tiene estas palabras con énfasis simple_,
pero estas que están a continuación **Presentan énfasis doble**
```

La salida del texto anterior se visualizará de forma similar a lo siguiente:

Esto texto *tiene estas palabras con énfasis simple*,

pero estas que están a continuación **Presentan énfasis doble**

Títulos

Los titulos permiten delimitar secciones bien diferenciadas en el texto, y ya son por si mismos texto resaltado, por lo que no deben rodearse de marcas de énfasis.

Los títulos vienen en diversos niveles, de acuerdo a la importancia del mismo. Por ejemplo, los títulos de primer nivel son los más importantes, y por tanto deberían ser utilizados solo como el título general del documento. Los títulos de nivel dos actuan como subtitulo, y permiten diferenciar secciones grandes del documento. Titulos de tercer, cuarto, quinto y sexto nivel pueden ser utilizados para marcar secciónes más especificas.

Los títulos, si bien tienen un estilo destacado, tienen una connotación semántica clara. No deberían ser utilizados para enfatizar palabras.

Los títulos comienzan con uno o más signos de numeral, llamado también almohadilla, o hash (#). La cantidad de almohadillas expresa el nivel del título, siendo una almohadilla los títulos de primer nivel, dos almohadillas de segundo nivel, y así siguiendo.

```
1 # Título de primer nivel
2 ## Título de segundo nivel
3 ### Título de tercer nivel
4 #### Título de cuarto nivel
5 ##### Título de quinto nivel
6 ###### Título de sexto nivel
```

Lo que se termina visualizando como:

Título de primer nivel

Título de segundo nivel

Título de tercer nivel

Título de cuarto nivel

Título de quinto nivel

Título de sexto nivel

También existe una forma distinta de escribir títulos de primer y segundo nivel, que consiste en "subrayar" las palabras que conforman el texto. Un subrayado doble indica títulos de primer nivel, y se logra utilizando signos de igual (=) para realizar el subrayado. Un subrayado simple indica títulos de segundo nivel, y se logra utilizando guiones medios, o signos de resta (-). El subrayado debe tener al menos dos signos consecutivos, pero lo ideal para maximizar la lectura en la entrada es que abarque tanto como el texto que está marcando.

```
Título de primer nivel
======================

Título de segundo nivel
-----------------------
```

Listas no ordenadas

Las listas no ordenadas marcan cada uno de sus elementos como viñetas. Se pueden utilizar asteriscos (*), signos de suma (+), o guiones o signos de resta (-) para marcar los elementos de la lista. Incluso pueden combinarse, aunque esto atenta contra la lectura en la entrada y no es recomendado.

```
+ Item 1
+ Item 2
+ Item 3
```

Resulta en:

- Item 1
- Item 2
- Item 3

También pueden colocarse listas dentro de otras listas, tabulando los elementos

```
+ Item 1
+ Item 2 tiene los siguientes elementos
    * Item 2A
    * Item 2B
+ Item 3
```

Resultando en:

- Item 1
- Item 2 tiene los siguientes elementos
    - Item 2A
    - Item 2B
- Item 3

Listas ordenadas

Las listas ordenadas, también llamadas listas numeradas, presentan números en lugar de viñetas. Se logran colocando un número al comienzo de la línea y siguiendolo de un punto.

```
1. Primer elemento
2. Segundo elemento
3. Tercer elemento
```

1. Primer elemento
2. Segundo elemento
3. Tercer elemento

De igual forma que con las listas desordenadas, estas se pueden anidar. Más aún, se pueden anidar listas ordenadas con no ordenadas en diversas formas.

Enlaces

Una caracteristica interesante es la posibilidad de insertar enlaces en cualquier punto del texto, que apuntan a otros documentos o sitios web. Un elnace se compone de dos partes: la dirección a la cual se quiere acceder, y el texto que se le mostrará al lector. El texto se coloca entre corchetes, los cuales van seguidos de unos parentesis que contienen la dirección a la cual se enlazará.

```
Lo siguiente es un enlace: [Ir a Google](https://google.com)
```

Lo siguiente es un enlace: Ir a Google

Imágenes

En ocasiones es necesario agregar algún tipo de elemento visual, en particular, imágenes. Las imágenes funcionan de forma similar a un enlace, pero se debe anteponer un signo de admiración (!) a los corchetes. El texto que se coloca entre corchetes corresponde con la descripción de la imágen, mientras que la dirección entre parentesis corresponde con la ubicación de la misma.

```
![Logo de Markdown Here](http://cor.to/mdhere)
```

Código

Por haber sido pensado para escribir documentación de código de programación, Markdown incluye una forma de mostrar código. Para ello se utilizan comillas agudas, llamadas también backsticks (‘). El texto entre backsticks se corresponde a código fuente. Si el mismo tiene varias líneas, se deben incluir tres backsticks en lugar de uno, pudiendo colocar al final de los backsticks de apertura, en el inicio y entre corchetes, el lenguaje en el que se está escribiendo el código.

```
‘let x = 5‘

‘‘‘ [JavaScript]
let x = 5;
let y = 7;
let z = x + y;
‘‘‘
```

Lo cual se vería así:

`let x = 5`

```javascript
let x = 5;
let y = 7;
let z = x + y;
```

Separaciones

También es posible colocar líneas de separación entre diferentes secciones del texto. Una línea de separación se ve como una fina línea horizontal que abarca todo el ancho del documento, y puede hacerse con una serie de signos de resta o guiones (-) colocados de forma consecutiva.

También pueden utilizarse asteriscos o signos de igual para la separación.

```
Esta sección se encuentra separada de la que le sigue.

---------------------------------------------------

La línea superior marca esa separación de forma clara.
```

A.2 Actividades

Ejercicio A.1 Dado el siguiente código en Markdown, modifiquelo de forma tal de dar énfasis simple a los périfericos que aparecen mencionados en él y énfasis doble a las palabras informática, software y hardware en todos los lugares en donde aparece.

```
La informática es la disciplina que que estudia el
tratamiento automatizado de información por parte
de las computadoras, las cuales se componen de un
conjunto de hardware y software.

El hardware se compone de una placa base, una
fuente de alimentación, memoria, un microprocesador,
y un conjunto de periféricos tales como mouse,
teclado, pantalla, parlantes, impresora, microfono
y placas de expansión.

El software consiste en los programas que corremos
en el equipo, como Word, GIMP, Age of Empires y otros.
```

■

Ejercicio A.2 Redacte en Markdown una receta para hacer tallarines caseros, utilizando una lista no ordenada para los mencionar los ingredientes y una lista ordenada para los pasos a seguir en la preparación. Puede valerse de Internet para buscar una receta, en caso de que desconozca los pasos a seguir. ■

Ejercicio A.3 Buscque en Internet imágenes de tres lugares del mundo que desearía visitar. En un documento Markdown, inserte las imágenes de forma consecutiva, colocando previo a cada una el nombre del lugar como titulo de nivel 3, y el país en el que se encuentra como subtitulo de nivel 5. El título debe consistir en un enlace a la página de Wikipedia sobre ese lugar. ■

Ejercicio A.4 El siguiente código Markdown tiene algunos errores, tanto semánticos como de sintaxis.

```
##### Sobre Markdown

## El lenguaje

El lenguaje Markdown fue creado en 2004 por John Gruber.
Es un lenguaje de marcado ligero diseñado para maximizar la
facilidad de escritura y lectura tanto en la entrada como
en la salida.

-----------------------------------------------

## _La sintaxis_

La sintaxis del lenguaje es sencilla, y consiste en signos
como asteriscos, almohadillas, y corchetes para lograr
estructurar un documento. En general se pueden escribir
los elementos de la siguiente lista:

Titulos
Palabras con énfasis
Listas
Enlaces
Código

-----------------------------------------------

Copyright 2019
```

Se pide entonces que detecte los errores y proponga un código que los corrija. ■

Ejercicio A.5 Elabore un documento Markdown que funcione como su curriculum vitae. Para ello, coloque su nombre completo como título principal, y siga al mismo con un subtitulo de nivel dos indicando su profesión. Agregue una foto de su persona, y seccione el contenido en datos personales, trabajos, formación y otros elementos utilizando titulos de tercer nivel en adelante y separadores según lo considere necesario. Utilice una lista no ordenada para sus datos personales.

No hace falta que el CV sea real, basta con que sea verosímil, si no cuenta con experiencia laboral o estudios, puede inventarlos para realizar esta actividad. ■

Ejercicio A.6 Realice en Markdown una breve carta de presentación donde usted solicita un puesto de “Desarrollador de Software” en una empresa. Utilice títulos, énfasis y listas de forma apropiada en los lugares que crea conveniente. ■

Imágen de un diseñador web pensando en código.
Diseño de Mudassar Iqbal con retoques propios.

HyperText Markup Language (HTML) es un lenguaje de marcado que sirve para describir páginas web, por lo que se ha transformado en el lenguaje de marcado más extendido del mundo.

HTML es un lenguaje basado en XML, por lo que su principio fundamental son las etiquetas, que actúan de marcas para indicar distintos tipos de contenidos en la página web que describen.

Un archivo de HTML es simplemente un archivo de texto, que en general se identifican con la extensión **.html**, y que puede editarse con cualquier editor de texto. Para visualizar el contenido como un sitio web, se debe utilizar un programa visualizador especial, que en general se conoce como **browser** o **navegador web**.

B.1 Browsers

El browser tiene una doble funcionalidad, por un lado, permite conectarse a dispositivos remotos, los cuales se identifican con una URL específica, y descargar de este uno o más archivos. Por otro lado permite visualizar los archivos HTML, interpretando su contenidos y dando una muestra visual de los contenidos (Esto incluye la interpretación de CSS y de JavaScript). Cuando uno coloca una URL como “’’, el browser ingresa a la máquina identificada como “google.com” y descarga por defecto el archivo “index.html” del mismo. Otras URLs, como “’’ descargan el archivo específico “archivo.html”.

Existen muchos browsers en el mercado, algunos privativos, otros libres. Entre los más destacados están:

- Internet Explorer
- Edge
- Google Chrome
- Mozilla Firefox
- Opera

B.2 W3C y estándares de la web

A diferencia de otros lenguajes, HTML se encuentra claramente específicado en lo que se conoce como el estándar HTML, el cual está definido por el **World Wide Web Consortium (W3C)**, una organización encargada de definir estándares para la web, y promover su correcto uso.

Logos de diferentes browsers. Existe una amplia diversidad, con navegadores que se enfocan en mercados sumamente específicos.
Logos del proyecto github.com/alrra/browser-logos.

Cabe destacar que HTML define una estructura para el contenido, y son los navegadores los que determinan de que forma mostrar dicho contenido (el estilo). Si se desea especificar estilos más avanzados, debe recurrirse al uso de otro lenguaje de la web, conocido como "Hojas de Estilo en Cascada", o en inglés **Cascade Style Sheets (CSS)**. Este lenguaje complementa a HTML, permitiendo definir cosas como los colores de texto, tamaños de fuentes, ubicación de imágenes, colores de fondo, distribución de contenidos en la página, entre una gran serie de otros elementos. CSS también está especificado por la W3C.

Finalmente, si lo que se desea es lograr una interactividad en el sitio (por ejemplo, que se muestre cierto contenido cuando el usuario selecciona una opción, o cuando presion un botón), se debe recurrir a un lenguaje de programación conocido como **JavaScript** (o **ECMAScript**), el cual permite manipular las etiquetas del documento HTML, agregando o quitando contenido, atributos, e incluso nuevas etiquetas en cualquier lugar del documento. JavaScript también está definido por la W3C, complementandose así los tres lenguajes.

La W3C también define otros estándares que permiten la definición de imágenes escalables para la web, o formatos de transmisión de datos entre equipos, entre otra serie de cosas.

A lo largo del tiempo se han definido varias versiones de HTML, siendo las primeras bastante primitivas, permitiendo poca interacción con el usuario, mientras que actualmente se puede definir aplicaciones enteras usando HTML, CSS y JavaScript. La versión actual de HTML es la versión 5, y es una versión "viva", es decir, no hay una versión 5 final, sino que el estándar se va actualizando a medida que surge la necesidad de utlizar nuevas etiquetas. Así, algunas etiquetas clásicas desaparecen, y otras nuevas aparecen. Junto con HTML5 aparecen nuevos estándares para CSS (CSS 3, también una versión "viva"), y JavaScript (En versiones cada vez más modernas adhiriendo al

estándar ECMAScript que publica una nueva versión cada año).

Por ser tan longevo, y por las cualidades de ser un lenguaje vivo, la documentación que uno puede encontrar en Internet sobre HTML puede estar desactualizada y haberse vuelto obsoleta. A esto se debe sumar el hecho de que mucha gente aprende por su cuenta, y luego hace público su conocimiento, el cuál no siempre es correcto. Por eso es importante para los desarrolladores web seguir los estándares de la W3C y asegurarse de que su código siga dichos estándares.

¿Qué ocurre si nuestro código no sigue el estándar? En principio, nada. Cualquier "buen browser" (entiendase casi todos los navegadores web modernos, a excepción de Internet Explorer) garantiza que los códigos que siguen el estándar de la W3C se verán correctamente. Sin embargo, cada uno tiene su propia implementación, y su propia idea de "que hacer" ante código que no sigue la norma. Así, código que funciona en Google Chrome puede no verse bien en Firefox o en Internet Explorer.

Si nos encontramos realizando un sitio que va a ser utilizado por el público en general, debemos seguir necesariamente el estándar, pues no podemos garantizar cual es el navegador web que va a utilizar la persona que ingrese. Internet Explorer, por gozar de una situación monopólica en la mayoría de los equipos domésticos, pudo no seguir los estándares. Esto hacía que cualquier desarrollador que hiciera una página, debía realizar dos versiones de la misma, una para Internet Explorer, y otra para el resto de los navegadores. La situación se mantuvo hasta la salida de Edge por parte de Microsoft, reemplazando a Internet Explorer, forzada por las decisiones de múltiples empresas y desarrolladores de dejar de dar soporte a Internet Explorer.

B.3 Sistema de etiquetas como marcas

En los lenguajes basados a etiquetas, la etiqueta es el elemento fundamental que permite marcar una porción de texto, y es, en si mismo, un elemento marcable por otras etiquetas.

Toda etiqueta está dada por una marca que comienza por el signo "<" (menor), se sigue de un **nombre**, y cierra con el signo ">" (mayor). El nombre de la etiqueta consiste en una única palabra, que puede contener letras, números, guiones bajos o guiones medios (no puede tener espacios).

Además, la mayoría de las etiquetas consisten en dos marcas, una como la mencionada anteriormente, conocida como marca de "**apertura**", y otra que es conocida como marca de "**cierre**", y que a diferencia de la anterios utiliza "</" (menor seguido de barra) para el comienzo de la marca.

Así, la siguiente es la marca de apertura:

```
1 <nombre_de_etiqueta>
```

Y la siguiente una etiqueta de cierre:

```
1 </nombre_de_etiqueta>
```

El nombre de la etiqueta debe ser el mismo tanto en la marca de apertura como en la de cierre. La etiqueta marca el texto que se encuentra entre la marca de apertura y la de cierre, llamado "**contenido**" de la etiqueta.

```
<nombre_etiqueta> Contenido </nombre_etiqueta>
```

El contenido de una etiqueta pueden ser otras etiquetas:

```
<etiqueta1>
    <etiqueta2>
        Contenido de la etiqueta
    </etiqueta2>
</etiqueta1>
```

También puede darse el caso de que se requiera información adicional para marcar un texto que el solo nombre de la etiqueta. Esa información adicional puede ser brindada a la etiqueta mediante un "**atributo**". Un atributo consiste en un par de elementos, conocidos como "**clave**" (el nombre del atributo) y "**valor**" (el valor del atributo). Estos se colocan únicamente en la etiqueta de apertura, separando la clave del valor mediante un signo igual (=), y donde el valor siempre debe escribirse entre comillas.

```
<nombre_de_etiqueta atributo="valor">
```

En una misma etiqueta pueden colocarse tantos atributos como se desee, sepando cada atributo del otro con un espacio. Note como el primer atributo también se separa del nombre de la etiqueta mediante un espacio.

```
<nombre attr1="val1" attr2="val2">
```

Finalmente, en ocasiones es necesario agregar contenido que sea ignorado por el visualizador. Este contenido sirve para que el mismo autor del código pueda estructurar sus ideas, o dejar mensajes a otros posibles lectores del código fuente, y se conocen como "**comentarios**".

Un comentario comienza cuando se encuentran los signos "<!–" y termina cuando se encuentran los signos "–>". Un comentario puede abarcar varias etiquetas, e incluso varias líneas de texto.

```
<!--
    Esto es un comentario,
    puede ponerse lo que se desee aquí,
    incluso etiquetas como <hola> o </hola>
    y serán ignoradas
-->
```

Además, vale aclarar que en HTML, así como en otros lenguajes de etiquetas, un espacio, multiples espacios, una o más tabulaciones, o incluso uno o más saltos de línea son equivalentes a un único espacio. Así, estas cuatro etiquetas son exactamente equivalentes:

```
<etiqueta attr1="val1" attr2="val2">
<etiqueta       attr1="val1"       attr2="val2">
<etiqueta attr1="val1"
```

```
          attr2="val2">
<etiqueta
    attr1="val1"
    attr2="val2"
>
```

Notese como el signo “>” de cierre de la marca, puede también estar en otra línea. También vale para el de apertura.

Por último, considere el siguiente código:

```
<etiqueta1><etiqueta2>Contenido de la etiqueta
</etiqueta2><etiqueta2 attr="val1"><etiqueta3>Contenido de la
   etiqueta 3
</etiqueta3>
</etiqueta2></etiqueta1>
```

Observe como se vuelve complicado identificar que etiqueta contiene a que otra, y qué marca tiene cada uno. Considere ahora la siguiente versión con el mismo código, pero estructrado de una forma diferente:

```
<etiqueta1>
    <etiqueta2>
        Contenido de la etiqueta
    </etiqueta2>
    <etiqueta2 attr="val1">
        <etiqueta3>
            Contenido de la etiqueta 3
        </etiqueta3>
    </etiqueta2>
</etiqueta1>
```

Observesé como esta forma de estructurar el contenido fácilita su lectura y su entendimiento. Esto se conoce en programación como “**indentación del código**”, es decir, tabular y estructurar el código de forma tal que sea fácil identificar las partes del mismo. La palabra “indentación” viene de la palabra inglesa “indent’, y su equivalente en español sería “tabulado”. Es deseable que todo el código se encuentre correctamente indentado. En los lenguajes de etiquetas, cada marca de cierre debería colocarse a la misma altura horizontal que su marca de cierre. Luego, el contenido dentro de una etiqueta se debe colocar en una nueva línea y siempre tabulado dos o cuatro espacios (un ancho de tabulación) más a la derecha que la etiqueta que lo contiene.

B.4 Etiquetas básicas de HTML

HTML, como lenguaje de etiquetas, sigue la misma estructura mencionada en la sección anterior, pero el conjunto de etiquetas que pueden utilizarse y la forma en la que deben estructurarse las mismas. También, el efecto que produce cada etiqueta al ser visualizada está definido en la especificación del lenguaje.

Existen etiquetas para una gran cantidad de elementos, y este apunte no pretende ser una guia comprensiva de los mismos, así como tampoco estar necesariamente actualizado. Por esto se

recomienda tener a mano los siguientes enlaces para aprender sobre nuevas etiquetas o validar la información sobre las aquí expuestas.

- https://developer.mozilla.org/es/
- https://www.w3schools.com

Párrafos y énfasis

La étiqueta más sencilla que permite definir un párrafo de texto es "**<p>**". Dentro de "<p>' solamente debería haber texto, el cuál puede o no estár enfatizado en algun lugar.

```
<p>
    El texto aquí presente compone un párrafo, el
    cuál empieza en la marca de apertura y termina
    en la marca de cierre.
</p>
<p>
    Al abrir una nueva etiqueta, el contenido de
    esta se encuentra en otro párrafo.
</p>
```

Cada párrafo se encuentra separado del siguiente por un pequeño espacio, cuyo tamaño inicial es delimitado por el browser. Si se requiere una mayor distancia entre un párrafo y el siguiente (u otros elementos que lo rodean) se debe utilizar CSS.

También se puede dar énfasis a palabras o frases enteras dentro de un párrafo. Para ello existen dos etiquetas, "****" que da un énfasis simple, y "****", el cuál da un énfasis doble. El énfasis simple suele ser mostrado por defecto en itálica, mientras que el doble suele ser mostrado en negrita. Nuevamente, este comportamiento puede variarse con CSS.

```
<p>
    En este párrafo <em>éste texto está siendo
    enfátizado</em>, y el que sigue también,
    <strong>pero con un énfasis aún mayor</strong>
</p>
```

Si bien es posible colocar una etiqueta "" dentro de una etiqueta "", esto no es recomendado, ya que solo se obtiene un efecto visual, el cuál debería ser logrado con CSS, semánticamente la frase ya está siendo más relevante que otras al encontrarse con énfasis.

Títulos

Para los títulos se usa el grupo de etiquetas "H", el cuál abarca seis diferentes nivel, desde "**<h1>**" hasta "**<h6>**". La etiqueta "<h1>" representa el título principal del documento, y por tanto no debería haber más de una en un mismo sitio. Etiquetas "**<h2>**" pueden ser utilizadas para subtítulos, o títulos de secciones o de artículos. Los "**<h3>**" pueden ser utilizados para subtítulos dentro de una sección o artículo. Así siguiendo el mísmo criterio, se debería utilizar títulos para marcar aquellos textos que identifícan partes importantes del sitio.

```
<h1>Título de primer nivel</h1>
<h2>Título de segundo nivel</h2>
<h3>Título de tercer nivel</h3>
<h4>Título de cuarto nivel</h4>
<h5>Título de quinto nivel</h5>
<h6>Título de sexto nivel</h6>
```

Listas

HTML también soporta listas, tanto ordenadas (se muestran los elementos como una serie de elementos numerados) como no ordenadas (se muestran los elementos con viñetas). Para declarar una lista se debe "abrir la lista", luego declarar los elementos como contenido de la lista, y finalmente "cerrar la lista". Para abrir la lista se utiliza la etiqueta "**<ol>**" (si se trata de una lista ordenada) o "**<ul>**" (si es una lista no ordenada). Se utiliza la misma etiqueta para el cierre, como marca de cierre. Los elementos de la lista se declaran con la etiqueta "**<li>**", la cuál se marca con apertura y cierre.

```
<h4>La siguiente es una lista ordenada</h4>
<ol>
    <li>Primer elemento</li>
    <li>Segundo elemento</li>
    <li>Tercer elemento</li>
    <li>Cuarto elemento</li>
</ol>

<h4>Y ahora una lista no ordenada</h4>
<ul>
    <li>Primer elemento</li>
    <li>Segundo elemento</li>
    <li>Tercer elemento</li>
    <li>Cuarto elemento</li>
</ul>
```

Dentro de la etiqueta "<li>" puede colocarse algo más que texto, por ejemplo, etiquetas de énfasis, e incluso algunas otras etiquetas, pero no debería contener párrafos enteros o títulos.

Dentro de las etiquetas "<ol>" y "<ul>" solo pueden haber etiquetas "<li>", es decir, no puede haber un párrafo o título en medio de la lista de elementos. Tampoco pueden existir elementos "<li>" fuera de una estructura de lista.

Tablas

Las tablas son una forma de generar información tabulada, o de doble entrada (fila y columna). Es una buena forma de mostrar datos que dependen de dos elementos, por ejemplo, paises con algunas de sus propiedades geográficas o políticas, como muestra la tabla a continuación:

País	Población	Superficie	Densidad
Argentina	44.494.502 hab.	2.780.400 km^2	16 hab/km^2
Uruguay	3.519.014 hab.	176.215 km^2	19,97 hab/km^2
Paraguay	7.152.703 hab.	406.752 km^2	17,58 hab/km^2
Bolivia	11.383.094 hab.	1.098.58 km^2	10,36 hab/km^2

Las tablas deben definirse con la etiqueta "**<table>**", y dentro de ésta declarar las filas, y en cada fila, las celdas (las columnas terminan siendo declaradas en base a las celdas).

La tabla enterior puede ser generada con el siguiente código HTML:

```
<table>
    <tr>
        <td>País</li>
        <li>Población</li>
        <li>Superficie</li>
        <li>Densidad</li>
    </tr>
    <tr>
        <li>Argentina</li>
        <td>44.494.502 hab.</li>
        <li>2.780.400 km$^2$</li>
        <li>16 hab/km$^2$</li>
    </tr>
    <tr>
        <li>Uruguay</li>
        <td>3.519.014 hab.</li>
        <li>176.215 km$^2$</li>
        <li>19,97 hab/km$^2$</li>
    </tr>
    <tr>
        <li>Uruguay</li>
        <td>7.152.703 hab.</li>
        <li>406.752 km$^2$</li>
        <li>17,58 hab/km$^2$</li>
    </tr>
    <tr>
        <li>Uruguay</li>
        <td>11.383.094 hab.</li>
        <li>1.098.58 km$^2$</li>
        <li>10,36 hab/km$^2$</li>
    </tr>
</table>
```

Como se puede apreciar, la etiqueta "**<tr>**" define una nueva fila, y va ubicada dentro de la etiqueta "<table>". Por otro lado, la etiqueta "**<td>**" se ubica dentro de la etiqueta "<tr>", y define una celda de la tabla. Los estilos como bordes, dimensiones y otros detalles deben ser definidos utilizando CSS.

Otros casos de uso más complejos pueden requerir que un conjunto de celdas se encuentren agrupadas de alguna forma, algo que puede lograrse mediante el uso de diversos atributos en las etiquetas "<tr>" o "<td>", dependiendo de cuál sea el agrupamiento deseado.

Enlaces

El nombre de HTML significa en español "lenguaje de marcado de hipertexto". El concepto de **hipertexto** remite a la idea de poder navegar el texto de una forma no lineal, mediante la utilización de enlaces, que pueden llevarnos delante y atrás en un conjunto de documentos.

Precisamente, uno de los elementos más importantes es entonces la capacidad de agregar

enlaces a otros documentos. Un enlace siempre debe componerse de dos partes fundamentales: el texto a mostrarle al lector y la dirección al cuál se desea enlazar.

Por esto, la etiqueta "**<a>**" (llamado así por "anchor", áncla en inglés), utilizada para definir enlaces, requiere de forma obligatoria un atributo llamado "**href**" (de hiper-referencia). Mientras que el texto que se le mostrará al lector del documento al visualizarlo en un browser será el contenido de la etiqueta, el lugar al que enlaza será la dirección colocada como valor en el atributo "href".

```
1 <a href="https://google.com">Ir a Google</a>
```

Es importante destacar que el enlace puede ser una URL cualquiera, pero también pueden utilizarse rutas relativas para enlazar a otros documentos en la misma máquina (es decir, del mismo sitio web).

```
1 <a href="archivo/2015_05.html">
2     Ver las noticias de Mayo del 2015
3 </a>
```

Imágenes

Las imágenes en HTML están dadas por la etiqueta "**<img>**", una de las etiquetas más extrañas de HTML, pues tiene la particularidad de requerir únicamente su marca de apertura, no teniendo que ser cerrada. Es decir, es una etiqueta que no posee contenido alguno.

Lleva sin embargo, de forma obligatoria, dos atributos. El primero es **src**, el cuál espera como valor una URL a la ubicación de la imágen (también podría ser la ruta a un archivo local). El otro atributo es **alt**, que espera como valor un texto, el cual se muestra en caso de que la imágen no pueda ser cargada, o cuando el usuario deja el puntero del mouse sobre la imagen. Este texto consiste en la "descripción" de la imágen, y también puede ser utilizado por visualizadores específicos para no videntes como información adicional.

```
1 <img src="imagenes/fotografias/banner.png"
2      alt="Publicidad de la empresa">
```

El tamaño de la imágen se debe determinar por CSS.

Audio y Video

Las etiquetas "**audio**" y "**video**" se introdujeron en HTML5 con la intensión de soportar contenido multimedia avanzado (algo que antes de esta versión solo podía lograrse mediante complementos externos del navegador, como Adobe Flash o Microsoft Silverlight). Las etiquetas mencionadas pueden incluir un atributo controls, que no lleva valor alguno, y que le indica al navegador que debe mostrarle los controles de reproducción al usuario.

El contenido de esas etiquetas debe ser una o más etiquetas "**<source>**", las cuales mediante el atributo **src** para indicar la ubicación del archivo, y el atributo **type** para indicar el formato en que

se encuentra (En los sitios web, muchos archivos pueden publicarse sin sus extensiones, por lo que indicar el formato resulta importante).

```
<audio controls>
    <source src="caballo.ogg" type="audio/ogg">
    <source src="caballo.mp3" type="audio/mpeg">
</audio>

<audio controls>
    <source src="caballo.ogg" type="video/ogg">
    <source src="caballo.mp4" type="video/mp4">
</audio>
```

Se debe tener en cuenta que estas etiquetas solamente son soportadas en navegadores web modernos, y no se visualizarán correctamente en máquinas antiguas no actualizadas.

Contenido embebido

Una etiqueta también interesante y muy utilizada es "**iframe**". Ésta permite incrustar contenido de otro sitio web, y hacer que el mismo se encuentre disponible en nuestro documento. Un ejemplo muy común es el caso de los videos de plataformas como YouTube, los cuales se incrustan utilizando esta etiqueta.

En general uno no suele incluir el contenido de otros sitios, y cuando lo hace, utiliza una porción de código predefinida, con atributos especiales ya completados por quien provee el código.

```
<iframe
    width="560"
    height="315"
    src="https://www.youtube.com/embed/dQw4w9WgXcQ"
    frameborder="0"
    allow="accelerometer; autoplay; encrypted-media; gyroscope;
   picture-in-picture"
    allowfullscreen>
</iframe>
```

Etiquetas semánticas de HTML

HTML define todo un conjunto de etiquetas que permiten agrupar contenido de forma semántica. Así, por ejemplo, un título, una serie de párrafos e imágenes que pertenezcan a una misma agrupación conceptual pueden ser marcadas e identificadas como tales.

Estas etiquetas no tienen una contraparte visual, y solamente permiten definir bloques lógicos. Sin embargo, herramientas de procesamiento automático de datos, como los "robots" de Google o Bing utilizan este agrupamiento para "comprender" la información del documento y brindar resultados de búsqueda de calidad para los usuarios (Y un diseñador siempre quiere que su sitio sea bien entendido por estas empresas para aparecer en los primeros lugares de los resultados de búsqueda).

Las agrupaciones conceptuales, si bien suman un nivel semántico no visual, pueden tener una contrapartida visual si se utiliza CSS para darles estilo.

Dentro de las etiquetas de agrupamiento encontramos:

- **<header>**, utilizada para la cabecera de un sitio (por ejemplo, la parte que contiene el título principal del sitio, el logo, algúnos posibles datos adicionales y tal vez algún menú de navegación principal).
- **<main>**, utilizada para agrupar todo el contenido principal del sitio, es decir, la parte relevante que interesa leer a los usuarios.
- **<footer>**, utilizada para definir el pie de página (una sección al final del sitio que suele contener datos como el copyright, enlaces a redes sociales, enlaces al mapa del sitio, y otros datos del dueño del sitio o del mismo sitio).
- **<section>**, utilizada para agrupar lógicamente secciones grandes del sitio que se muestran en un mismo documento. Cada sección tiene que contener de forma obligatoria un título, de "<h2>" en adelante.
- **<article>**, utilizada para agrupar contenido que lógicamente conforma un articulo, y que podría leerse de forma independiente del resto del contenido, y aún así, tener sentido.
- **<nav>**, utilizada para definir barras de navegación, en general con enlaces a uno o más documentos del sitio.
- **<aside>**, utilizada para agrupar elementos que no refieren a una parte importante del sitio. Por ejemplo, publicidad, o elementos de barras laterales suelen agruparse bajo ésta etiqueta.
- ** **, utilizada para agrupar elementos que no entran semánticamente en ninguna de las etiquetas anteriores.

Cada una de estas etiquetas abre cuando se comienza a agrupar y cierra cuando se deja de agrupar. A continuación se da una muestra de dos secciones de un sitio, que a su vez contienen articulos.

```
<section>
    <h2>Noticias deportivas</h2>

    <article>
        <h3>La preocupación de los argentinos</h3>
        <p>
            Todos los argentinos siguen preocupados
            por la indecisión de los expertos sobre
            quién es el mejor jugador de futbol del
            mundo. <strong>Maradona</strong> y
            <strong>Messi</strong> son dos de los
            cantidatos más votados.
        </p>
    </article>
    <article>
        <h3>Se puede jugar al básquet con 1.50?</h3>
        <p>
            El básquet siempre ha sido un deporte
            dominado por personas de gran altura,
            pero una nueva óla de jugadores de baja
            estatura y alta performance están dando
            muestra de que la altura no es un
            condisionante a la hora de jugar.
        </p>
    </article>
</section>
<section>
```

```
28      <h2>Noticias del espectáculo</h2>
29
30      <article>
31          <h3>Mar del Plata o Villa Carlos Paz</h3>
32          <p>
33              Cuáles son las obras de teatro que
34              podemos encontrar este verano en los
35              puntos turisticos?
36          </p>
37      </article>
38      <article>
39          <h3>Una nueva de Ricardo Darín</h3>
40          <p>
41              El actor nos cuenta de su nuevo
42              trabajo, una película que no sabemos
43              de que se trata, pero que seguro es
44              buena, porque actúa Darín, y si es
45              mala, no importa, porque actúa Darín.
46          </p>
47      </article>
48  </section>
```

B.5 Documentos HTML válidos

Hasta ahora vimos la forma en la que las etiquetas se escriben y pueden anidarse, y también vimos algunas etiquetas básicas de HTML. Sin embargo, un documento HTML no consiste en etiquetas sueltas en el texto, sino que debe contar con una estructura protocolar obligatoria.

Todo documento HTML debe comenzar con una pseudo-etiqueta, llamada "declaración del doctype". Esta declaración es leída por el navegador web previo a procesar cualquier elemento del documento, y le indica al browser qué es lo que debe esperar del mismo. Nosotros escribiremos lo siguiente:

```
1  <!DOCTYPE html>
```

Dicha sentencia le indica al browser que el documento se trata de un archivo HTML en su versión 5. Sentencias similares para HTML en versiones anteriores también existen, así como otros estándares como XHTML o XML, que también pueden ser comprendidos por el navegador, pero trabajados de forma distinta por el mismo.

Luego, puede comenzar el contenido del sitio, el cual debe estar siempre dentro de una etiqueta que engloba a la totalidad de los elementos, y que se llama "**html**". Dicha etiqueta no puede contener cualquier cosa, sino que solamente dos etiquetas pueden estar dentro de ella, las etiquetas "**head**" y la etiqueta "**body**". Es decir, la declaración de un documento debería ir quedando algo como:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4    </head>
```

```
  <body>
  </body>
</html>
```

No debería haber ninguna otra etiqueta fuera de html, ni directamente bajo esta, sino que toda otra etiqueta debería estar dentro de las etiquetas “<head>” o “<body>”.

La etiqueta “**<head>**” sirve para darle información al navegador sobre caracteristicas adicionales del documento. Estos datos son utilizados de diferentes formas por el navegador, y también por los robots de Google o Bing. En particular hay dos etiquetas importantes a colocar dentro de “<head>”, la etiqueta “**<title>**” que indica el título del sitio (el navegador suele mostrarlo como título de la ventana o pestaña) y la etiqueta “**<meta>**” que carga datos adicionales al navegador, pudiendo haber más de una etiqueta meta en el head.

Cada etiqueta meta define, mediante atributos, información adicional para el navegador. Uno de los atributos más importantes es el “**charset**”, el cuál indica al browser que vamos a utilizar símbolos “especiales” (es decir, caracteres que no son del idioma inglés, como acentos o la letra eñe). Para esto, el atributo “charset” debe tener el valor “utf-8”.

También es recomendable agregar como atributo el idioma en el cuál va a estar escrito nuestro documento. Para ello se utiliza el atributo “**lang**”, el cuál se coloca directamente sobre la etiqueta “<html>”. Para español se debe utilizar el valor “**es**”.

Adicionalmente podemos agregar etiquetas meta con información como el autor del sitio, descripción, palabras clave, e incluso información sobre como debería mostrarse el documento en un dispositivo. Para ello, la etiqueta “<meta>” soporta los “**name**” y “**content**”, que deben llenarse con valores específicos en cada ocasión. Nuestro documento HTML comienza a verse así:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Título del sitio</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
            initial-scale=1.0">
        <meta name="author" content="Autor del sitio">
        <meta name="description"
            content="Un sitio de prueba para los que se inician en
    HTML">
        <meta name="keywords"
            content="HTML, Web, CSS, JavaScript">
    </head>
    <body>
    </body>
</html>
```

Ahora si podemos centrarnos en el contenido, el cuál estará declarado en la etiqueta “<body>”, la cuál contendrá todos los elementos visibles del sitio. Por ejmplo, un header con el titulo principal del sitio, una etiqueta main con el contenido, y un footer con información de copyright.

A continuación se muestra un ejemplo de una página HTML básica.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <title>Noticias</title>
        <meta charset="utf-8">
        <meta name="viewport"
            content="width=device-width, initial-scale=1.0">
        <meta name="author" content="Juan de los Palotes">
        <meta name="description"
            content="Un sitio de noticias para todos y todas">
        <meta name="keywords"
            content="Noticias, Espectáculo, Deportes, Economía">
    </head>
    <body>
        <header>
            <h1>El sitio de las noticias</h1>
        </header>
        <main>
            <section>
                <h2>Noticias deportivas</h2>

                <article>
                    <h3>La preocupación de los argentinos</h3>
                    <p>
                        Todos los argentinos siguen preocupados
                        por la indecisión de los expertos sobre
                        quién es el mejor jugador de futbol del
                        mundo. <strong>Maradona</strong> y
                        <strong>Messi</strong> son dos de los
                        cantidatos más votados.
                    </p>
                </article>
                <article>
                    <h3>Se puede jugar al básquet con 1.50?</h3>
                    <p>
                        El básquet siempre ha sido un deporte
                        dominado por personas de gran altura,
                        pero una nueva óla de jugadores de baja
                        estatura y alta performance están dando
                        muestra de que la estatura no es un
                        condicionante a la hora de jugar.
                    </p>
                </article>
            </section>
            <section>
                <h2>Noticias del espectáculo</h2>

                <article>
                    <h3>Mar del Plata o Villa Carlos Paz</h3>
                    <p>
                        ¿Cuáles son las obras de teatro que
                        podemos encontrar este verano en los
                        puntos turisticos?
                    </p>
                </article>
                <article>
```

```
57                      <h3>Una nueva de Ricardo Darín</h3>
58                      <p>
59                          El actor nos cuenta de su nuevo
60                          trabajo, una película que no sabemos
61                          de que se trata, pero que seguro es
62                          buena, porque actúa Darín, y si es
63                          mala, no importa, porque actúa Darín.
64                      </p>
65                  </article>
66              </section>
67          </main>
68          <footer>
69              <p>
70                  Copyright© 2019 - Juan de los Palotes
71              </p>
72          </footer>
73      </body>
74  </html>
```

Por supuesto que “<body>” admite cualquier combinación compleja de etiquetas de contenido, y no solo la aquí expuesta. El contenido dependerá, en última instancia, de lo que se desea mostrar.

Para saber si un documento es o no es válido, la W3C, encargada de regular el estándar de HTML, provee a los desarrolladores una página web que permite validar sus archivos, para detectar si estos efectivamente cumplen con el estándar. La dirección es:

https://validator.w3.org/nu/#file

Hay que tener en cuenta que el validador de la W3C solamente analiza si las etiquetas utilizadas son correctas, y si se cumple con todos los campos protocolares, así como si las etiquetas están anidadas correctamente. No valida, por otro lado, cuestiones de estilo, como la indentación, ni el uso correctamente semántico de las etiquetas. Por esto es que el validador es útil, pues nos brinda una idea clara de si nuestro documento cumple al menos con una estructura correcta, y por tanto, va a ser procesado correctamente por los navegadores, pero no nos asegura que el código sea “bueno”.

B.6 CSS

Es interesante comprender que en general, un sitio web no se compone únicamente de código HTML, sino que es un conjunto de archivos, que incluye tanto a uno como más archivos HTML, como también a las imágenes del sitio, íconos, fuentes, estilos, y código para interactividad local. Es decir, cuando uno realiza un sitio web, no realiza un archivo, sino un directorio lleno de archivos.

Las rutas de los archivos HTML siempre deberían ser rutas relativas. Esto permite que el directorio que involucra el proyecto, y que contiene a todos los archivos, pueda ser copiado de una máquina a otra, e incluso subido a un servidor para ser compartido.

Uno de estos tipos de archivo resulta particularmente interesante, el CSS. Los estilos por defecto de HTML resultan sumamente básicos, y prácticamente no existe sitio hoy en día que no tenga CSS (Aunque si puede carecer de otros) El código CSS consiste en una serie de reglas que se le brindan a los elementos HTML. Estas reglas incluyen detalles como:

- El tamaño de fuente.
- El estilo del texto (itálica, negrita, subrayado, etc).
- El color del texto y del fondo de los elementos.

- Bordes, con sus tamaños, colores, estilo, redondeo de bordes, etc.
- Tamaños de las imágenes, videos, reproductores de audio, etc.
- Imágenes de fondo y transparencias.
- Ubicación de los elementos, tamaños y formas.
- Distancia del elemento a otros elementos y a su contenido interno.

En las últimas versiones de CSS se pueden definir incluso cosas como animaciones, transformaciones de contenido (Por ejemplo, poner el texto en mayúsculas), e incluso estilos que se aplican únicamente en ciertas situaciones (por ejemplo, si ve el contenido en un celular, o si se va a imprimir el sitio).

Aprender CSS y todas sus particularidades es un trabajo complejo, y hay gente que se especializa en esta área, conocidos como "Diseñadores web". Un diseñador web se diferencia de un programador. El primero nada sabe de lenguajes de programación, y solamente se enfoca en HTML y CSS. El segundo debe tener una idea de HTML y CSS, pero no es su fuerte, ni debe conocerlos en profundidad.

El código CSS consiste en una serie de "reglas", las cuales se aplican una tras otra sobre el los elementos del documento. El navegador web aplica las reglas, y tras hacerlo, muestra el resultado. Cada regla consiste en un "selector" (el o los elementos sobre el cuál se aplicará la regla) y una serie de "propiedades" (que son las cualidades a dar al elemento sobre el cuál se aplica la regla), cada una con su correspondiente "valor".

```
selector {
  propiedad1: valor1;
  propiedad2: valor2;
  propiedad3: valor3;
}
```

Cada propiedad se separa de las que le siguen mediante un punto y coma, que no debe olvidarse. El objetivo aparece al comienzo, y las propiedades que se aplicarán iran entre llaves.

Como selector, puede colocarse el nombre de cualquier etiqueta HTML. Las propiedades que acepta una regla dependen del tipo de elemento a dar estilo. Por ejemplo, el siguiente código coloca el color de texto de todos los títulos h2 en rojo.

```
h2 {
  color: red;
}
```

También pueden definirse más de un selector mediante la separación de los mismos utilizando comas. Por ejemplo, el siguiente código pone en rojo sobre fondo azul tanto los títulos de nivel dos, como los de nivel tres y cuatro.

```
h2, h3, h4 {
  color: red;
  background-color: blue;
}
```

Existen formas de apuntar a un elemento específico del documento HTML, en lugar de “a todos los elementos h1”, así como de apuntar a un grupo de elementos particular (por ejemplo, el grupo de elementos que deberían tener texto en rojo). Para ello, el HTML debe cumplir ciertos requisitos, agregando atributos a los elementos a dar estilo (Los atributos “class” y “id”). Tmabién existen formas de apuntar a los conocidos como pseudo-elementos (por ejemplo, los enlaces que tienen el cursor del mouse sobre ellos), dandoles estilo.

El código CSS se incluye en la etiqueta “<head>”, pues si bien tiene resultados visibles, no involucran contenido a disponer en la página. Así, dentro de “<head>” se puede agregar código CSS utilizando la etiqueta “<style>”, que abre cuando inicia el código CSS y cierra cuando termina.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>...</title>
  <style>
    h2, h3, h4 {
      color: red;
      background-color: blue;
    }
  </style>
</head>
<body>
  ...
```

También es posible colocar el estilo en un archivo externo, y adjuntarlo al documento HTML mediante la etiqueta “<link>”, algo que no trataremos aquí. Versiones anteriores de HTML permitían incluir estilos directamente en el HTML, como atributos del elemento a dar estilo. Esta práctica se encuentra hoy en día obsoleta, y se desalienta su uso en HTML5, pues se removerá la posibilidad de hacerlo en el futuro.

B.7 Formularios y respuestas del servidor

Si ha navegado por Internet, habrá notado sitios como Gmail, Facebook o Twitter, que solicitan al usuario que suministre un correo electrónico y contraseña, previo a dejarlo visualizar la página principal. ¿Cómo hace HTML para validar que el usuario y la contraseña son correctos? La respuesta es, no lo hace.

Un servidor espcializado, encargado de validar la información recibe los datos que el usuario ingresó en el HTML y luego determina si la contraseña coincide con los registros de la empresa o no, mostrandole al usuario el contenido que solicitaba, o indicandole que verifique sus credenciales y vuelva a intentarlo.

HTML provee funcionalidad para realizar este tipo de acciones, es decir, enviar información a un servidor. Para ello, utiliza una etiqueta llamada “<form>”, y etiquetas que van dentro de esta, como “<input>”, “<select>”, “<textarea>” y “<button>”.

Una etiqueta “<form>” debe contener dos atributos, “method” y “action”. El atributo “method’ puede tener un valor de “GET” o de “POST”, y le indica al navegador la forma en la que los datos serán enviados al servidor (El servidor deberá estar configurado para esperar lo mismo que el navegador envía). La etiqueta action, por su parte, espera como valor una URL, que es la dirección a donde se enviarán los datos.

Todo formulario debe contar con alguna forma en que el usuario confirme los valores ingresados, y mande la información al servidor de forma efectiva. Esta acción se conoce como “submit”, y suele ser un botón el que provee tal caracteristica.

Los botones se crean con la etiqueta “<button>”, y muestran como texto del mismo, lo que tengan en su contenido. Además, suelen llevar un atributo llamado “type” que indica qué es lo que se espera que haga el botón. Existen dos types que el browser trata de forma especial: “submit” y “reset”. Si el usuario da clic sobre un botón cuyo “type” es “submit”, la información cargada en el formulario será enviada al servidor. Si da clic sobre un botón con “type” igual a “reset”, limpiará toda la información cargada al momento en el formulario, dejandolo en blanco.

Existen diversos elementos que pueden aparecer en un formulario. La mayoría de ellos se ven representados en la etiqueta “<input>”, qué posee dos atriutos obligatorios. El primero “type” indica que tipo de input se trata, pudiendo ser:

- **text** Un campo de texto genérico
- **password** Un campo de texto, pero donde el contenido no se muestra o aparece como puntos o asteriscos.
- **checkbox** Un campo de marca de selección, en donde el usuario puede tildar, o destildar la opción.
- **radio** Un campo de selección entre varias opciones. Si el usuario marca una opción, las otras se desmarcan.
- **hidden** Un campo que no se muestra en el formulario, y que puede ser utilizado para datos de control.
- **file** Un campo en el que el usuario puede seleccionar un archivo de su máquina para la carga.

Desde HTML5, tipos de input con valores más semánticos se han creado, lo que permite una mayor diferenciación sobre los datos que se están manipulando, dando lugar por ejemplo, a herramientas capaces de completar automáticamente formularios enteros. Así “type” también puede ser:

- **color** Permite al usuario elegir un color mediante un selector de colores.
- **date** El usuario puede ingresar una fecha, con un selector de fechas estilo calendario (sin ingresar hora).
- **email** Un campo de texto, pero donde se espera que el usuario agregue un correo electrónico.
- **number** Un campo de texto, pero donde se espera que el usuario ingrese un número.
- **tel** Un campo de texto, pero donde se espera que el usuario ingrese un número de teléfono.
- **url** Un campo de texto en donde se espera una dirección URL válida.

También existen otras opciones, para horas, meses, rangos de elementos, campos de búsqueda, etc.

El otro atributo importante de los input es “name”, el nombre. Este atributo es la forma de identificar al valor de forma inequivoca por parte del servidor. Por tanto, cada input debe tener un valor distinto. El valor consiste en un nombre, que puede estar formado por letras, números, guiones bajos y guiones medios, pero que no puede contener espacios.

Un caso particular es el de los inputs de tipo “radio”. En este caso, se espera que el nombre sea el mismo para aquellas opciones que conformen un conjunto (es decir, las opciones tales que si el usuario selecciona una, las otras se des-seleccionan).

A continuación hay un ejemplo de un pequeño formulario que solicita al visitante su correo electrónico y contraseña.

```
<form action="servidor/procesar_ingreso" method="POST">
  <label> Correo electrónico:
    <input type="email" name="correo_de_usuario">
  </label>
  <label> Contraseña:
    <input type="password" name="password_de_usuario">
  </label>
  <button type="submit">
    Ingresar
  </button>
</form>
```

Esto resulta en algo como lo siguiente:

Note el uso de la etiqueta “<label>”. Esta se utiliza para englobar a los elementos “<input>”, permitiendo mostrar un texto asociado a los mismos que el visitante verá en el sitio, dandole indicaciones sobre qué es lo que se espera que ingrese. Pueden usarse en su lugar párrafos de texto simple, pero “<label>” brinda una ventaja adicional de usabilidad, cuando el usuario da clic sobre el texto, el “<input>” se selecciona, y el usuario ya puede comenzar a escribir, esto no ocurre con un párrafo. En particular es una caracteristica muy útil cuando se trabaja con elementos de tipo “radio” o “checkbox”.

Las etiquetas “<input>” son muy útiles, pero en ocaciones no son suficientes. Si se desea por ejemplo, que el usuario ingrese texto muy largo, se debe recurrir a la etiqueta “<textarea>”. Esta etiqueta presenta un cuadro de texto de múltiples líneas, el cual se agranda automáticamente a medida que el usuario escribe. Al igual que los “<input>”, espera un atributo “name” para poder ser procesado, pero no lleva “type”.

También puede que se desee elegir de una serie de opciones de una lista, algo que se logra con la etiqueta “<select>” (también espera un “name” como atributo), la cual lleva dentro una o más etiquetas “<option>”. Esta etiqueta muestra una lista desplegable, en donde el usuario puede seleccionar una de las opciones (dadas cada una por un “<option>” distinto).

A continuación hay un formulario que muestra el uso de etiquetas “<textarea>” y “<select>”. También presenta un “checkbox” y un “radio” El formulario permite al usuario ingresar un reclamo en el sitio de una empresa proveedora de internet.

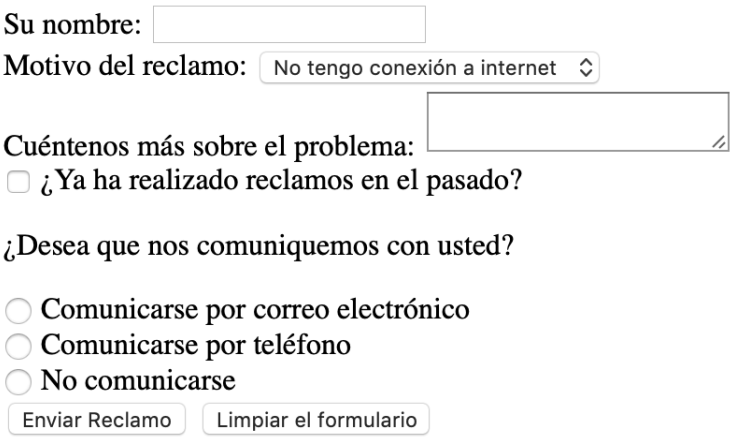
```
<form action="servidor/procesar_ingreso" method="POST">
  <div>
    <label for="nombre">Su nombre:</label>
    <input type="text" name="nombre" id="nombre">
  </div>

  <div>
    <label for="motivo">Motivo del reclamo:</label>
    <select name="motivo" id="motivo">
      <option>No tengo conexión a internet</option>
      <option>Mi conexión anda lenta</option>
      <option>Mi conexión es intermitente</option>
      <option>Otro tipo de reclamo</option>
    </select>
```

```
15    </div>
16
17    <div>
18      <label for="problema">Cuéntenos más sobre el problema:</label>
19      <textarea name="problema" id="problema"></textarea>
20    </div>
21
22    <div>
23      <input type="checkbox" name="reclamos_previos"
          id="reclamos_previos">
24      <label for="reclamos_previos">¿Ya ha realizado reclamos en el
          pasado?</label>
25    </div>
26
27    <p>¿Desea que nos comuniquemos con usted?</p>
28    <div>
29      <input type="radio" name="comunicacion" id="comunicacion1">
30      <label for="comunicacion1"></label>Comunicarse por correo
          electrónico</label>
31    </div>
32    <div>
33      <input type="radio" name="comunicacion" id="comunicacion2">
34      <label for="comunicacion2"></label>Comunicarse por
          teléfono</label>
35    </div>
36    <div>
37      <input type="radio" name="comunicacion">
38      <label for="comunicacion3">No comunicarse</label>
39    </div>
40
41    <div>
42      <button type="submit">
43        Enviar Reclamo
44      </button>
45      <button type="reset">
46        Limpiar el formulario
47      </button>
48    </div>
49  </form>
```

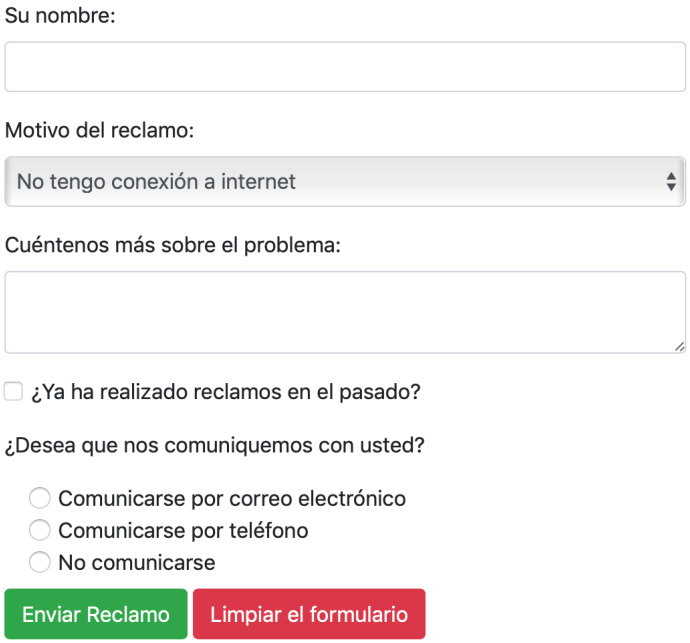
Note como en este ejemplo las etiquetas “<label>” no envuelven al campo que denotan, sino que se usa otra técnica, la cuál consiste en utilizar un atributo “for” en “<label>” cuyo valor coincide con el valor del atributo “id” del “<input>” que se encuentra marcando.

También se envuelve cada conjunto de “<label>” y “<input>” en una etiqueta “ ” que los agrupa semánticamente. Esto también logra el efecto de mostrar un elemento debajo de otro en el formulario, ya que por defecto, los “ ” se muestran uno bajo el otro, mientras que los “<label>” y “<input>” se muestran todos en el mismo renglón.

El formulario anterior se vería algo así:



Los formularios pueden ebellecerse con CSS. Existen incluso conjuntos de código pre-hechos por terceros que uno puede utilizar para embellecer sus formularios (y sus sitios en general). Estos códigos se agrupan en lo que se llama “biblioteca’. Con un poco de CSS el formulario anterior puede verse de la siguiente forma:



Existen también otros atributos y particularidades que hacen a los formularios, así como al procesamiento de los mismos. Estos detalles no son parte del presente, y se recomienda al lector retomar estos temas cuando posea un conocimiento más amplio de programación y de redes de computadoras.

B.8 Actividades

Los siguientes ejercicios se realizar de forma continuada, es decir, para realizar el ejercicio 2 deberá realizar el ejercicio 1 previamente. Recomendamos hacerlos en orden y no saltearse ninguno.

Ejercicio B.1 Cree un documento HTML un sitio web que actuará como portal de noticias. Acuerdesé de que el documento debe tener todas las partes protocolares necesarias. Luego agregue los siguientes elementos a su sitio:

a) Un título principal con el texto "Últimas Noticias".
b) Una primer sección bajo el título "Noticias deportivas".
c) Una segunda sección bajo el título "Noticias del espectáculo".
d) Un párrafo descriptivo en cada sección indicando el tipo de noticias que el lector podrá encontrar.

El resultado debería verse algo como:

Últimas Noticias

Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

Noticias del Espectáculo

Los famosos, la téle, el teatro, y todo el chisme.

■

Ejercicio B.2 Agregue dos artículos de noticias a cada sección del sitio anterior. Puede buscar noticias en un medio de su preferencia, o inventarlas. Cada noticia debe contar con un título (de menor jerarquía que los de las secciones), y un copete (resumen de la noticia) en donde aparezcan algunas palabras o frases clave con énfasis. El resultado debería ser similar al siguiente:

Últimas Noticias

Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

La preocupación de los argentinos

Todos los argentinos siguen preocupados por la indecisión de los expertos sobre quién es el mejor jugador de futbol del mundo. **Maradona** y **Messi** son dos de los cantidatos más votados.

Se puede jugar al básquet con 1.50?

El básquet siempre ha sido un deporte dominado por *personas de gran altura*, pero una nueva óla de jugadores de **baja estatura y alta performance** están dando muestra de que la estatura no es un condicionante a la hora de jugar.

Noticias del Espectáculo

Los famosos, la téle, el teatro, y todo el chisme.

Mar del Plata o Villa Carlos Paz

¿Cuáles son las obras de teatro que podemos encontrar este verano en los *puntos turisticos*?

Una nueva de Ricardo Darín

El actor nos cuenta de su nuevo trabajo, una película que no sabemos de que se trata, pero que seguro es buena, porque **actúa Darín**, y si es mala, no importa, porque **actúa Darín**.

■

Ejercicio B.3 Añada el link de referencia a los sitios de donde tomo las noticias (o coloque un enlace a sitios de su preferencia si las inventó). Los enlaces deben leerse con el texto "Leer más". El resultado debería ser similar a este:

Últimas Noticias

Noticias Deportivas

Toda la información del fútbol, basquet, y mucho más.

La preocupación de los argentinos

Todos los argentinos siguen preocupados por la indecisión de los expertos sobre quién es el mejor jugador de futbol del mundo. **Maradona** y **Messi** son dos de los cantidatos más votados.

Se puede jugar al básquet con 1.50?

El básquet siempre ha sido un deporte dominado por *personas de gran altura*, pero una nueva óla de jugadores de **baja estatura y alta performance** están dando muestra de que la estatura no es un condicionante a la hora de jugar.

Noticias del Espectáculo

Los famosos, la téle, el teatro, y todo el chisme.

Mar del Plata o Villa Carlos Paz

¿Cuáles son las obras de teatro que podemos encontrar este verano en los *puntos turisticos*?

Una nueva de Ricardo Darín

El actor nos cuenta de su nuevo trabajo, una película que no sabemos de que se trata, pero que seguro es buena, porque **actúa Darín**, y si es mala, no importa, porque **actúa Darín**.

■

Ejercicio B.4 Sume ahora, antes del título del diario, en una cabecera, una imágen que actúe de logo del sitio. Puede tomar de internet una imágen que guste como logo. Incluya también un pie de página con la información de copyright del sitio. La misma debe contener el nombre del autor y el año en que realizó el sitio. Tenga en cuenta que lo que antes eran posiblemente simples secciones deberían ser ahora el contenido principal del sitio. ■

Ejercicio B.5 Si realizó los ejercicios hasta ahora correctamente (es decir, ha colocado todas las etiquetas de estructura semánticas que corresponde) debería ser posible que su sitio luzca como la siguiente imagen:

Para ello necesitaremos agregar en el “<head>” del sitio la siguiente definición de CSS dentro de una etiqueta “<style>”:

```
header, footer {
  background-color: #222;
  color: lightgray;
}
footer {
  height: 40px;
}
header {
  height: 85px;
}
header, footer p {
  padding: 10px;
}
header h1, header h2, header h3,
header h4, header h5, header h6 {
  display: none;
}
header img {
  height: 35px;
  width: auto;
}
header nav {
  width: 100%;
  background-color: #222;
  margin-bottom: 20px;
}
header li {
  display: inline-block;
  padding: 10px;
}
header li:hover {
  background-color: #555;
  cursor: pointer;
}
header a {
  color: white;
  text-decoration: none;
}
main {
  margin-top: 10px;
  display: flex;
  flex-direction: row;
}
section {
  justify-content: space-between;
  width: 40%;
  margin-left: 5%;
  border: 3px solid rgb(150, 25, 25);
  border-radius: 10px;
  padding: 20px;
```

```
}
section>h1, section>h2, section>h3,
section>h4, section>h5, section>h6,
section>p {
  color: rgb(150, 25, 25);
}
article p {
  text-align: justify;
}
article a {
  padding: 10px 20px;
  background-color: rgb(150, 25, 25);
  border-radius: 5px;
  color: white;
  text-decoration: none;
}
article a:hover {
  background-color: rgb(200, 25, 25);
  text-decoration: none;
}
aside {
  text-align: center;
  background-color: cornflowerblue;
  color: darkred;
  width: 50%;
  margin: 0 auto;
  font-weight: bolder;
}
aside h1, aside h2, aside h3, aside h4,
aside h5, aside h6 {
  animation: blinker 2s linear infinite;
}
@keyframes blinker {
  50% { opacity: 0; }
}
```

■

Ejercicio B.6 Agregue en la cebecera una barra de navegación, que contenga una lista no ordenada de enlaces a las secciones del sitio. Las mismas son: Politica, Policiales, Deportes y Espectáculos. El resultado debe ser como el siguiente:

Ejercicio B.7 Agregue justo sobre el pie de página una publicidad. Coloque en ella un título o dos títulos de nivel dos en adelante, y un párrafo de texto. El resultado se verá also así:

Ejercicio B.8 Valide el código que ha generado con el validador oficial de la W3C. Si el código pasa exitosamente las pruebas, significa que el código no viola los estándares (aunque no significa que siga buenas prácticas de estilo, por lo que no necesariamente es un buen código).

De encontrarse con errores al momento de válidar, corrija su código hasta que este sea válido. Puede encontrar el validador en:

https://validator.w3.org/nu/#file

Ejercicio B.9 Se pide que diseñe una página web que contenga un formulario web capaz de cargar ordenes médicas para pacientes que requieren reposo laboral. El formulario debe contener los siguientes campos:

- Un campo de texto para el nombre del médico interviniente.
- Un campo de texto para el número de matricula del médico interviniente.
- Un campo de selección radial que permita elegir si el número de matricula corresponde a la matricula nacional o provincial.
- Un campo de texto para el nombre del paciente.
- Un campo de selección que permita elegir la cantidad de horas de reposo. Deben haber 4

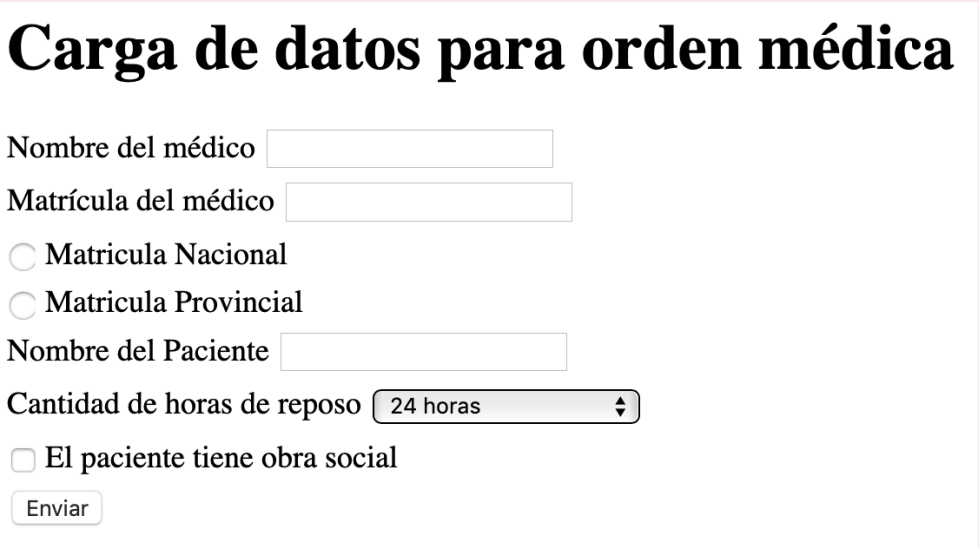
opciones: “24 horas”, “48 horas”, “72 horas”, “hasta alta médica”.
- Un campo de selección (checkbox) que se pueda marcar si el paciente tiene obra social, y destildar en caso negativo.
- Labels para cada uno de los campos.
- Un botón para enviar el formulario.

El formulario deberá utilizar el método GET, y su acción deberá indicar como valor “#” (solo el signo numeral).

Puede utilizar el siguiente código CSS:

```
label {
  display: block;
  margin: 5px 0;
}
```

Esto logrará que los elementos aparezcan como en la imagen a continuación:



Ejercicio B.10 Diseñe una página web que contenga un título principal con el texto “Mundiales de la FIFA”, y luego una tabla de doble entrada, donde las filas corresponden cada una a un mundial distinto. Deben haber tres columnas, la pimera indicando en que lugar del mundo se realizó el evento, y la segunda y tercer columna indicando el campeón y subcampeón respectivamente. Incluya al menos los mundiales de los siguientes años:

- 2002
- 2006
- 2010
- 2014
- 2018

Puede utilizar el siguiente código CSS para que la tabla quede más fácil de visualizar:

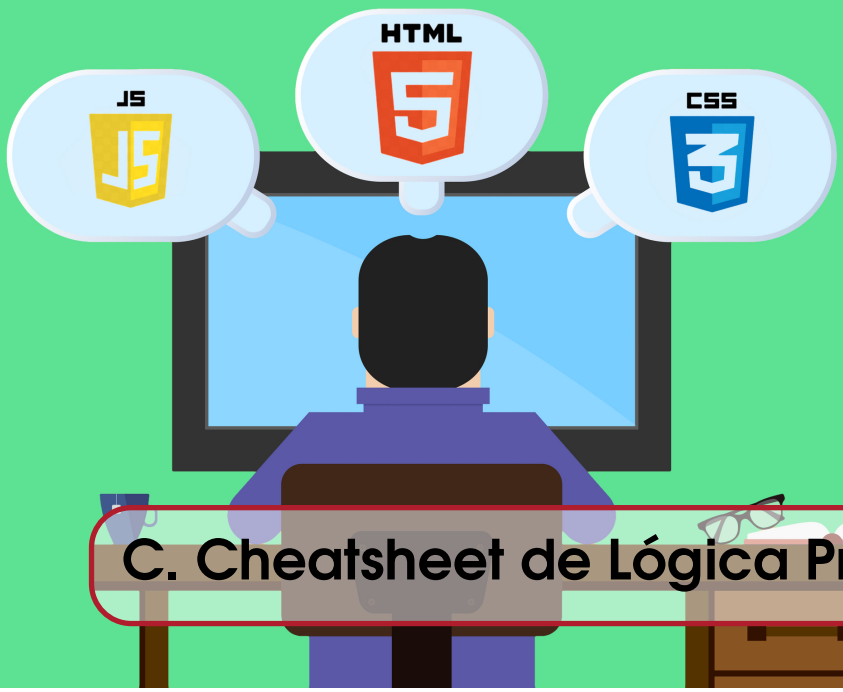
```
table {
```

```
  border: 3px solid rgb(150, 25, 25);
  border-collapse: collapse;
  border-spacing: 0;
}
table tr:nth-child(odd) {
  background-color: rgb(255, 175, 175);
}
table tr:nth-child(1) {
  background-color: rgb(150, 25, 25);
  color: white;
}
table td {
  padding: 3px 5px;
}
```

El resultado de la página debería ser el siguiente.

Mundiales de la FIFA

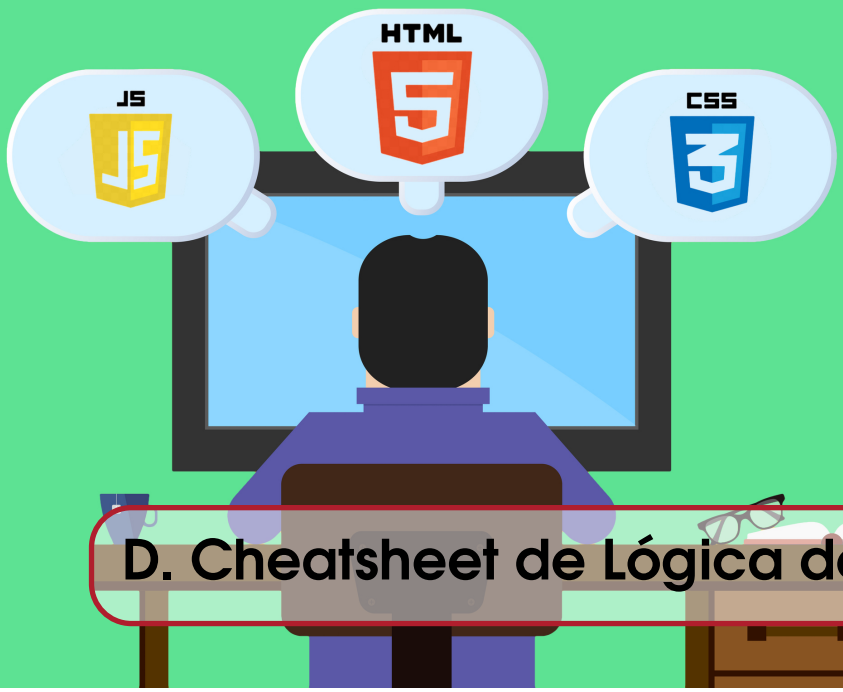
Año	Sede	Campeón	Subcampeón
1930	Uruguay	Uruguay	Argentina
1934	Italia	Italia	Checoslovaquia
1938	Francia	Italia	Hungría
1950	Brasil	Uruguay	Brasil
1954	Suiza	Alemania	Hungría
1958	Suecia	Brasil	Suecia
1962	Chile	Brasil	Italia
1966	Inglaterra	Inglaterra	Alemania
1970	México	Brasil	Italia
1974	Alemania	Alemania	Paises Bajos
1978	Argentina	Argentina	Paises Bajos
1982	España	Itália	Alemania
1986	México	Argentina	Alemania
1990	Italia	Alemania	Argentina
1994	Estados Unidos	Brasil	Italia
1998	Francia	Francia	Brasil
2002	Corea del Sur/Japón	Brasil	Alemania
2006	Alemania	Italia	Francia
2010	Sudafrica	España	Paises Bajos
2014	Brasil	Alemania	Argentina
2018	Rusia	Francia	Croacia



C. Cheatsheet de Lógica Proposicional

Imágen de un diseñador web pensando en código.
Diseño de Mudassar Iqbal con retoques propios.

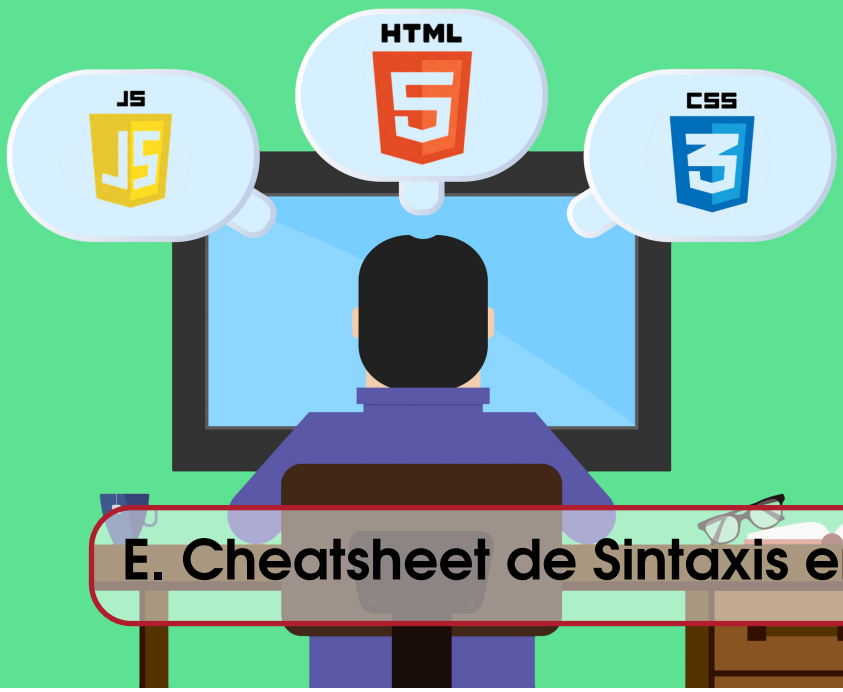
A ESCRIBIR



D. Cheatsheet de Lógica de Predicados

Imágen de un diseñador web pensando en código.
Diseño de Mudassar Iqbal con retoques propios.

A ESCRIBIR



E. Cheatsheet de Sintaxis en Programación

Imágen de un diseñador web pensando en código.
Diseño de Mudassar Iqbal con retoques propios.

A ESCRIBIR