

ONNA BUGEISHA

GUERRERAS DE LA CIENCIA

HACKERS & DEVELOPERS™ PRESS

HD HACKERS &
DEVELOPERS™

ONNA BUGEISHA

HACKERS & DEVELOPERS™ PRESS

ONNA BUGEISHA

GUERRERAS DE LA CIENCIA



HACKERS & DEVELOPERS™ PRESS
2020

ONNA BUGEISHA – GUERRERAS DE LA CIENCIA

© 2019 por Andrea Cristina Navarro Moreno, Bibiana Mollinedo Rivadeneira, Fernanda Eugenia Bahit, Fernanda Hernández González, Florencia Paula Vilardel Tignanelli, Indira Luz Burga Menacho, Jessica Sena, Josefina Monsegur, Lorena Santamaría Jiménez, Marta Macho Stadler, Milagros Mora y Montserrat Ortega Gallart.

Excepto donde se indique lo contrario, el contenido de este libro está distribuido bajo una **Licencia Creative Commons Atribución 4.0 Internacional (CC BY 4.0)**. Se permite la reproducción total o parcial de esta obra, su incorporación a un sistema informático, su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin necesidad de autorización previa de los titulares de los derechos de autor.

Licencia completa: <https://creativecommons.org/licenses/by/4.0/legalcode.es>

El diseño gráfico de esta obra se encuentra restringido a una licencia CC BY-NC-ND 4.0 (Algunos derechos reservados). Puede distribuirse, copiarse, y almacenarse con cualquier fin, excepto fines comerciales y no puede utilizarse para realizar obras derivadas. La licencia libre aplica al contenido de los artículos, pero no al diseño y maquetación.

Licencia completa para el diseño gráfico: <https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

ISBN 978-987-86-3950-5

Editado por Eugenia Bahit

1^a edición, marzo de 2020. Avellaneda, Buenos Aires. Rep. Argentina.

Hecho el depósito que marca la Ley 11.723.

Publicado por Hackers & Developers™ Press.
483 Green Lanes. N13 4BS, London. United Kingdom.
<https://www.hdmag.press/>

Compilación de glosario: Maiara Beatriz Cercasi

Revisión de artículos: A. Navarro, E. Bahit, F. Hernández, J. Monsegur y M. Mora.

Diseño de portada: © 2019 Eugenia Bahit. CC BY-NC-ND 4.0

Diseño y maquetación interior: © 2019 Eugenia Bahit. CC BY-NC-ND 4.0

*A todas las mujeres que dedican su vida
a la ciencia, luchando por la igualdad y
la equidad en el mundo.*

RESUMEN DE CONTENIDOS

PRIMERA PARTE: LÓGICA Y EPISTEMOLOGÍA

P. 71 | Hallazgo y determinación del conocimiento científico

En este apartado, la autora expone las bases epistemológicas sobre las que se sustenta el conocimiento científico.

P. 91 | Lógica y razonamiento deductivo

En este artículo, la autora profundiza en los temas esenciales de la lógica inherentes al razonamiento deductivo, como base para comprender el método utilizado por las ciencias para validar el conocimiento.

P. 121 | Introducción al método científico

En este artículo, la autora explica en qué consiste el método científico empleado por las ciencias formales (como las matemáticas) y el empleado por las ciencias ciencias fácticas o experimentales.

P. 132 | Las Ciencias Informáticas y su relación con las ciencias físicas y la matemáticas

Un breve resumen introductorio a la segunda parte del libro.

SEGUNDA PARTE: FUNDAMENTOS TEÓRICOS

FÍSICA

P. 137 | Fundamentos físicos de las Ciencias Computacionales

En este artículo la autora realiza un resumen de los principales procesos físicos y teorías sobre los que se fundamentan las Ciencias de la Computación.

COMUNICACIONES

P. 151 | Introducción a la comunicación de datos

En este artículo, la autora realiza una repaso sobre los fundamentos teóricos en los que se basa la comunicación de los datos, abarcando desde la codificación hasta la transmisión de los mismos.

P. 165 | Introducción a las comunicaciones ópticas

En este documento la autora explica los fundamentos físicos y tecnológicos sobre los que se apoyan las transmisiones de datos por fibra óptica.

TERCERA PARTE: TEMAS DE INFORMÁTICA APLICADA

BASES DE DATOS

P. 183 | Enmascaramiento de datos

En este artículo, la autora realiza una introducción al contexto teórico sobre el enmascaramiento de datos para efectuar un análisis de casos prácticos.

P. 193 | Generación de datos de prueba

En este documento, la autora realiza un análisis minucioso de las principales herramientas de código abierto que pueden emplearse para la generación de datos falsos con fines de prueba, y ofrece algunos ejemplos en SQL, de generación directa sobre MariaDB/MySQL.

DESARROLLO WEB

P. 205 | Generación de hojas de estilo escalables

En este artículo, una especialista en desarrollo Web y UX, realiza un compendio de los principales factores a tener en cuenta para el desarrollo de hojas de estilo (CSS) que puedan evolucionar con el tiempo y continuar siendo comprensibles.

P. 211 | Variables nativas en CSS

Complementando la «Guía de procedimiento para la generación de hojas de estilo escalables», la autora explica qué son las variables nativas en CSS, su uso y funcionalidad, y cómo implementarlas.

P. 215 | Introducción a los Sistemas de Diseño

En este documento la autora realiza un repaso sobre los elementos que componen un sistema de diseño, y su utilidad en el Diseño Gráfico.

P. 221 | Mapas de viaje para la identificación de problemas

En este artículo, la autora propone la generación de diagramas de ruta del usuario, siguiendo la línea cronológica de los procedimientos que este sigue durante el uso de la aplicación, para detectar los momentos críticos que impiden que el usuario complete su objetivo.

DESARROLLO MÓVIL

P. 229 | CouchDB para el manejo de datos fuera de línea

En este artículo la autora propone utilizar CouchDB para evitar la pérdida de información y/o falta de funcionamiento de una aplicación móvil cuando el dispositivo subyacente pierde temporalmente la conexión a Internet. En el documento, la autora explica cómo utilizar PouchDB como enlace entre CouchDB y el desarrollo basado en Ionic.

GESTIÓN DE PROYECTOS

P. 229 | Scrum en empresas de desarrollo de Software

La autora explica en qué consiste este marco de trabajo para el desarrollo de Software, y finalmente proponer una guía de procedimiento para su implementación en empresas.

HARDWARE

P. 251 | Mantenimiento preventivo y correctivo de impresoras

En este artículo, la autora realiza una descripción de los principales componentes de las impresoras láser y de inyección, sus averías habituales y sugiere un plan de acción en cada caso.

P. 259 | Mantenimiento preventivo de ordenadores

En este documento, la autora realiza un breve repaso sobre los principales componentes que pueden encontrarse tanto en ordenadores de sobremesa como portátiles para finalmente explicar cómo proceder a la limpieza de dichos componentes.

INTELIGENCIA ARTIFICIAL Y MINERÍA DE TEXTO

P. 265 | Introducción a las Redes Neuronales

En este documento, la autora realiza una introducción a los fundamentos biológicos en los que se sustentan las redes neuronales artificiales, así como un repaso de los conceptos generales para explicar cómo se aplican las redes neuronales artificiales al aprendizaje supervisado.

P. 285 | Análisis de Texto sobre redes sociales

En este trabajo de investigación, la autora busca abordar los procedimientos necesarios para aplicar análisis de textos no estructurados, con el fin de entender las reacciones de usuarios a partir de los comentarios realizados a un video institucional.

PROGRAMACIÓN

P. 309 | Guía de referencias del lenguaje C

Se trata de una guía completa, comentada y explicada, de referencias del lenguaje de programación C, destinada para servir por un lado, de fuente de consulta a programadores de otros lenguajes, y por el otro, al aprendizaje de los fundamentos de la programación de Sistemas Operativos.

SEGURIDAD DE LA INFORMACIÓN

P. 347 | Prevención de ciberataques en el sector sanitario

Una especialista en ciberseguridad del sector sanitario, explica la problemática particular con los datos en dicho sector para finalmente exponer una serie de recomendaciones normalizadas, que sirvan para prevenir dicha problemática.

SEGURIDAD INFORMÁTICA

P. 361 | Tratamiento decimal de caracteres Unicode

En este trabajo, la autora propone una aproximación a un modelo teórico para el tratamiento decimal de datos a alto nivel, como mecanismo de validación de datos y prevención de vulnerabilidades.

P. 393 | Autenticación por credenciales criptográficas disociadas

En este trabajo de investigación, la autora propone un mecanismo de autenticación para la generación de credenciales de usuario en tiempo de ejecución, mediante el uso de algoritmos criptográficos, que permita la validación de identidades sin necesidad de almacenar nombres de usuario o contraseñas.

SISTEMAS DE INFORMACIÓN GEOGRÁFICA

P. 415 | Entorno virtual para la centralización de datos

En esta guía, la autora propone la virtualización de un sistema basado en Ubuntu, PostgreSQL y la extensión PostGIS, como medio para la centralización de los datos en sistemas de información geográfica.

P. 423 | Diseño de mapas geográficos

En este artículo, una especialista en el desarrollo de sistemas de información geográfica, explica cómo diseñar e implementar funciones de geocodificación y geolocalización mediante el uso de bibliotecas JavaScript gestadas para la interacción con mapas geográficos.

P. 429 | Trazado de mapas mediante mosaicos vectoriales

En este documento, la autora complementa su artículo de «Diseño de mapas geográficos aplicados al desarrollo Web y móvil», explicando en detalle qué son los mosaicos vectoriales y los fundamentos del trazado de mapas mediante el uso de tecnología Web.

CUARTA PARTE: MATEMÁTICA TEÓRICA Y DIVULGACIÓN CIENTÍFICA

P. 441 | El teorema de los cuatro colores

En este documento la autora explica en qué consiste este teorema matemático que plantea que para colorear cualquier mapa geopolítico plano de tal manera que dos países con frontera común sean de distinto color, basta con emplear cuatro colores, y la influencia que han tenido los grafos y el uso de ordenadores en su demostración.

PARTE FINAL

P. 461 | Glosario lexicográfico

P. 477 | Índice general

P. 489 | Propiedad intelectual

P. 490 | Agradecimientos

ÍNDICE DE
AUTORAS

ARTÍCULOS DE LA AUTORA ANDREA NAVARRO MORENO

Enmascaramiento de Datos

Generación de datos de prueba

Introducción a las Redes Neuronales aplicadas al Aprendizaje

Supervisado



ANDREA NAVARRO MORENO

Autora

(Argentina, 1989)

Ingeniera en Informática por la Universidad de Mendoza, Argentina. Especializada en Inteligencia Artificial, se desempeña como codirectora del Laboratorio de Inteligencia Artificial de la sede San Rafael de la Facultad de Ingeniería de la UM, donde también ejerce como docente de las cátedras de Programación, e Inteligencia Artificial. En 2016, cofundó su propia empresa de formación, Junco TIC, donde además de impartir cursos, se encuentra a cargo de la redacción de artículos técnicos y educativos.

ARTÍCULOS DE LA AUTORA BIBIANA MOLLINEDO RIVADENEIRA

Introducción a la Comunicación de Datos

Introducción a las Comunicaciones Ópticas



BIBIANA MOLLINEDO RIVADENEIRA

Autora

(Argentina, 1992)

Ingeniera en Telecomunicaciones por la Universidad de Río Cuarto, Córdoba. Actualmente desarrolla su carrera como docente Universitaria, ejerciendo como ayudante rentada de la cátedra de Introducción a la Ing. en Telecomunicaciones II, de su Universidad. Se desempeña como docente del Plan Ceibal (Fundación Sadosky), impartiendo clases de Pensamiento Computacional. Es instructora Python y actualmente se encuentra trabajando en un proyecto de I+D para el prototipado de soluciones IoT.

ARTÍCULOS DE LA AUTORA EUGENIA BAHIT

Hallazgo y determinación del conocimiento científico

Nociones de Lógica y las Formas del Razonamiento

Introducción al método científico

Guía de Referencias del Lenguaje C

Manipulación decimal de caracteres Unicode con codificación UTF-8

Sistema de Autenticación por Credenciales Criptográficas Disociadas



EUGENIA BAHIT

Autora y Editora

(Argentina, 1978)

Informática Teórica y Técnica Profesional en Neuropsicología.

Miembro de la *European Association for Theoretical Computer Science* (EATCS), se encuentra especializada en lógica matemática aplicada a la Seguridad Informática, Teoría de Conjuntos y Teoría de Objetos. Actualmente se encuentra trabajando en Modelos Axiomáticos para la postulación de una Teoría de Objetos en Estado Puro. Como docente se ha especializado en Neurociencias aplicadas al aprendizaje de la programación informática. En el campo de la Informática Aplicada, se especializó en Ingeniería Inversa de Código y Seguridad por Diseño sobre entornos GNU/Linux y OpenBSD.

ARTÍCULOS DE LA AUTORA FERNANDA HERNÁNDEZ GONZÁLEZ

Fundamentos Físicos de las Ciencias Computacionales

La autora participó además, en la revisión técnica del artículo «Introducción a las Comunicaciones Ópticas» de la Ing. Bibiana Mollinedo Rivadeneira



FERNANDA HERNÁNDEZ GONZÁLEZ

Autora

(México, 1995)

Física. Graduada por la Universidad Nacional Autónoma de México (UNAM). Interesada por la mecánica cuántica, desarrolló su tesis de grado sobre la solución de la *Ecuación Estacionaria Unidimensional de Schrödinger* por medio de matrices de dispersión. Actualmente, se encuentra cursando una Maestría en Ciencias Físicas en la UNAM.

ARTÍCULOS DE LA AUTORA FLORENCIA PAULA VILARDEL TIGNANELLI

Prevención de Ciberataques en el Sector de Salud



FLORENCIA PAULA VILARDEL TIGNANELLI

Autora

(Argentina, 1978)

Magister en tecnologías de la información por la Universidad CAECE. Licenciada en Sistemas de Información por la Universidad de Buenos Aires, cuenta además con una certificación ITIL®. Se encuentra especializada en Seguridad de la Información, y en el diseño, desarrollo e implementación de mejoras en el área de la ciberseguridad, que aseguren el cumplimiento de diversos marcos normativos tales como CMMI®, CobiT®, ITIL® v3, PMI®, ISO®/IEC® 20000, e ISO®/IEC® 27000, entre otros.

ARTÍCULOS DE LA AUTORA INDIRA LUZ BURGA MENACHO

Análisis de Texto sobre Redes Sociales



INDIRA BURGA MENACHO

Autora

(Perú, 1987)

Ingeniera de Sistemas. Máster en Ciencia de Datos e Ingeniería de Computadores. Obtuvo el grado de Ingeniería en 2009 por la Universidad de Ingeniería y Tecnología de Perú, y el Máster de especialización en Ciencia de Datos y Tecnologías Inteligentes en 2016, en la Universidad de Granada, España. Se desempeña actualmente como Líder Técnica de Inteligencia de Negocios y *Data Science* en la UTEC, dirigiendo un proyecto que tiene por objetivo la transformación de la Universidad en una empresa orientada a datos, capaz de lograr el incremento de la tasa de graduaciones mediante el uso de la Ciencia de Datos.

ARTÍCULOS DE LA AUTORA JESSICA SENA

CouchDB para el manejo de datos fuera de línea en aplicaciones basadas en Ionic

Diseño de mapas geográficos aplicados al desarrollo web y móvil

Introducción al trazado de mapas Web mediante mosaicos vectoriales



JESSICA SENA

Autora

(España, 1985)

Ingeniera de Software por la Universitat Politècnica de Catalunya.

Posgraduada de la Universitat Oberta de Catalunya. Realizó su posgrado en Diseño de Experiencia de Usuario (UX). Especializada en Ingeniería de Sistemas de Información Geográfica (SIG), integró el equipo de Innovación, Geostart, del Instituto Cartográfico y Geológico de Cataluña hasta 2019, donde se desempeñó como Ingeniera de Software desde el año 2013. En 2017 participó del proyecto *The Canopy Health Monitoring* (CanHeMon) de la Comisión Europea del Centro Común de Investigación, ejerciendo como *Mobile Software Engineer*.

ARTÍCULOS DE LA AUTORA JOSEFINA MONSEGUR

Guía de Procedimiento para la Generación de Hojas de Estilo Escalables

Variables nativas en CSS

Sistemas de Diseño

Mapas de Viaje para la Identificación de Problemas



JOSEFINA MONSEGUR

Autora

(Argentina, 1978)

Licenciada en Artes Visuales por el Instituto Nacional del Arte (UNA) de Argentina. Especializada en Diseño de Interfaces y Experiencia de Usuario (UX), su carrera se ha enfocado en la comprensión de los procesos computacionales a bajo nivel, con el fin de optimizar los mecanismos de diseño propios de su especialidad. Se desempeña como *Lead Designer* en *Booking Holdings Inc.*, en su sede central de Ámsterdam, donde se encuentra trabajando actualmente, en un proyecto I+D cuyo objetivo persigue hallar formas de cuantificación de métricas cualitativas que puedan ser aplicadas a la reducción de brechas entre el producto y los procesos de diseño.

ARTÍCULOS DE LA AUTORA LORENA SANTAMARÍA JIMÉNEZ

Mantenimiento preventivo y correctivo de impresoras láser y de inyección de tinta

Mantenimiento preventivo de ordenadores



LORENA SANTAMARÍA JIMÉNEZ

Autora

(España, 1980)

Técnica Superior en Administración de Sistemas Informáticos en Red. Graduada del Instituto FOC de Granada, se desempeña actualmente como Técnica de Sistemas y Microinformática del *Agora International School* de Madrid, donde tiene a su cargo el mantenimiento y seguridad de los equipos informáticos y redes de datos, así como la gestión de la *Intranet*. Paralelamente, se encuentra trabajando en un proyecto privado de recreación de un ordenador antiguo para juegos, a gran escala, y aspira a especializarse en la administración de servidores en la nube.

ARTÍCULOS DE LA AUTORA MARTA MACHO STADLER

El teorema de los cuatro colores: un problema topológico demostrado con ayuda de un ordenador



MARTA MACHO STADLER

Autora

(España, 1962)

Doctora en Matemáticas por la Université Claude Bernard Lyon

(UBCL), Francia. Topóloga especializada en Teoría Geométrica de Foliaciones y Geometría no conmutativa, ejerce actualmente como Profesora Agregada del Departamento de Matemáticas de la Universidad del País Vasco, donde además se desempeña como editora del *blog* «Mujeres con Ciencia» de la Cátedra de Cultura Científica. Galardonada con numerosos premios entre los que se destacan la Medalla de la Real Sociedad Matemática Española y el Premio Emakunde, en reconocimiento a su labor como Divulgadora Científica y su compromiso con la igualdad de género en el ámbito social y académico.

ARTÍCULOS DE LA AUTORA MILAGROS MORA

Implementación de Scrum en una empresa de desarrollo de Software



MILAGROS MORA

Autora

(Argentina, 1983)

Analista Universitaria en Sistemas por la Universidad Tecnológica Nacional, Argentina. Con una Diplomatura en Gestión Ágil de Proyectos, se desempeña actualmente como *Scrum Master* y Analista Funcional, habiéndose destacado además, su labor como programadora de base de datos SQL. En el marco del Laboratorio de Innovación Ciudadana, Argentina 2018, de la Secretaría General Iberoamericana, formó parte del equipo a cargo del diseño y desarrollo de la plataforma «*Ruta Trans*», la primera aplicación que ha permitido compartir información a nivel internacional, sobre espacios y organizaciones de apoyo a personas transgénero y transexuales.

ARTÍCULOS DE LA AUTORA MONTSERRAT ORTEGA

GALLART

Configuración de un entorno virtual para la centralización de datos en Sistemas de Información Geográfica (SIG)



MONTSERRAT ORTEGA GALLART

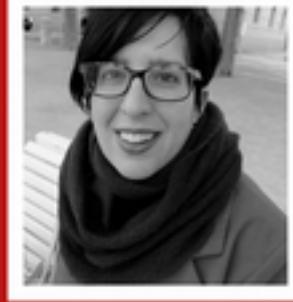
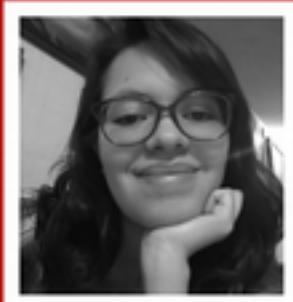
Autora

(España, 1982)

Ingeniera Informática por la Universitat Oberta de Catalunya.

Especializada en Sistemas de Información Geográfica (SIG), cuenta con un posgrado en Geografía y Cartografía, y desempeña actualmente como Analista Programadora del Instituto Cartográfico y Geológico de Cataluña, donde se ha destacado por su trabajo como desarrolladora de la plataforma «Instamaps», la principal plataforma cartográfica de la *Generalitat de Catalunya*, destinada a la exploración geográfica de la Comunidad Autónoma.

AUTORAS DE HACKERS & DEVELOPERS™



PRÓLOGO



Convención sobre la eliminación de todas las formas de discriminación contra la mujer¹ de Naciones Unidas

Adoptada y abierta a la firma y ratificación, o adhesión, por la Asamblea General de Naciones Unidas en su resolución 34/180, de 18 de diciembre de 1979. Entrada en vigor: 3 de septiembre de 1981, de conformidad con el artículo 27 (1)

Los Estados Partes en la presente Convención

Considerando que la Carta de las Naciones Unidas reafirma la fe en los derechos humanos fundamentales, en la dignidad y el valor de la persona humana y en la igualdad de derechos de hombres y mujeres,

Considerando que la Declaración Universal de Derechos Humanos reafirma el principio de la no discriminación y proclama que todos los seres humanos nacen libres e iguales en dignidad y derechos y que toda persona puede invocar todos los derechos y libertades proclamados en esa Declaración, sin distinción alguna y, por ende, sin distinción de sexo,

1 Disponible en línea en <https://www.ohchr.org/sp/professionalinterest/pages/cedaw.aspx>

Considerando que los Estados Partes en los Pactos Internacionales de Derechos Humanos tienen la obligación de garantizar a hombres y mujeres la igualdad en el goce de todos los derechos económicos, sociales, culturales, civiles y políticos,

Teniendo en cuenta las convenciones internacionales concertadas bajo los auspicios de las Naciones Unidas y de los organismos especializados para favorecer la igualdad de derechos entre el hombre y la mujer,

Teniendo en cuenta asimismo las resoluciones, declaraciones y recomendaciones aprobadas por las Naciones Unidas y los organismos especializados para favorecer la igualdad de derechos entre el hombre y la mujer,

Preocupados, sin embargo, al comprobar que a pesar de estos diversos instrumentos las mujeres siguen siendo objeto de importantes discriminaciones,

Recordando que **la discriminación contra la mujer viola los principios de la igualdad de derechos y del respeto de la dignidad humana**, que **dificulta** la participación de la mujer, en las mismas condiciones que el hombre, en la vida política, social, económica y cultural de su país, que **constituye un obstáculo** para el aumento del bienestar de la sociedad y de la familia y que **entorpece** el pleno desarrollo de las posibilidades de **la mujer para prestar servicio a su país y a la humanidad**,

Preocupados por el hecho de que **en situaciones de pobreza la mujer tiene un acceso mínimo a la alimentación, la salud, la enseñanza, la capacitación y las oportunidades de empleo, así como a la satisfacción de otras necesidades,**

Convencidos de que el establecimiento del nuevo orden económico internacional basado en la equidad y la justicia contribuirá significativamente a la promoción de la igualdad entre el hombre y la mujer,

Subrayado que la eliminación del apartheid, de todas las formas de racismo, de discriminación racial, colonialismo, neocolonialismo, agresión, ocupación y dominación extranjeras y de la injerencia en los asuntos internos de los Estados es indispensable para el disfrute cabal de los derechos del hombre y de la mujer,

Afirmando que el fortalecimiento de la paz y la seguridad internacionales, el alivio de la tensión internacional, la cooperación mutua entre todos los Estados con independencia de sus sistemas sociales y económicos, el desarme general y completo, en particular el desarme nuclear bajo un control internacional estricto y efectivo, la afirmación de los principios de la justicia, la igualdad y el provecho mutuo en las relaciones entre países y la realización del derecho de los pueblos sometidos a dominación colonial y extranjera o a ocupación extranjera a la libre determinación y la independencia, así como el respeto de la soberanía nacional y de la integridad territorial, promoverán el progreso social y el desarrollo y, en consecuencia, contribuirán al logro de la plena igualdad entre el hombre y la mujer,

Convencidos de que la máxima participación de la mujer en todas las esferas, en igualdad de condiciones con el hombre, es indispensable para el desarrollo pleno y completo de un país, el bienestar del mundo y la causa de la paz,

Teniendo presentes el gran aporte de la mujer al bienestar de la familia y al desarrollo de la sociedad, hasta ahora no plenamente reconocido, la importancia social de la maternidad y la función tanto del padre como de la madre en la familia y en la educación de los hijos, y conscientes de que el papel de la mujer en la procreación no debe ser causa de discriminación, sino que **la educación de los niños exige la responsabilidad compartida entre hombres y mujeres** y la sociedad en su conjunto,

Reconociendo que para lograr la plena igualdad entre el hombre y la mujer es necesario modificar el papel tradicional tanto del hombre como de la mujer en la sociedad y en la familia,

Resueltos a aplicar los principios enunciados en la Declaración sobre la eliminación de la discriminación contra la mujer y, para ello, a adoptar las medidas necesarias a fin de suprimir esta discriminación en todas sus formas y manifestaciones,

Han convenido en lo siguiente:

Parte I

Artículo 1

A los efectos de la presente Convención, la expresión «discriminación contra la mujer» denotará **toda distinción, exclusión o restricción** basada en el sexo que **tenga por objeto o resultado menoscabar o anular el reconocimiento, goce o ejercicio por la mujer**, independientemente de su estado civil, sobre la base de la igualdad del hombre y la mujer, de los derechos humanos y las libertades fundamentales en las esferas política, económica, social, cultural y civil o en cualquier otra esfera.

Artículo 2

Los Estados Partes condenan la discriminación contra la mujer en todas sus formas, convienen en seguir, por todos los medios apropiados y sin dilaciones, una política encaminada a eliminar la discriminación contra la mujer y, con tal objeto, **se comprometen** a:

- a) **Consagrarse**, si aún no lo han hecho, **en sus constituciones nacionales** y en cualquier otra legislación apropiada el principio de la **igualdad del hombre y de la mujer** y asegurar por ley u otros medios apropiados la realización práctica de ese principio;
- b) Adoptar medidas adecuadas, legislativas y de otro carácter, con las sanciones correspondientes, que prohíban toda discriminación contra la mujer;

- c) Establecer la **protección jurídica de los derechos de la mujer** sobre una base de igualdad con los del hombre y garantizar, por conducto de los tribunales nacionales competentes y de otras instituciones públicas, la protección efectiva de la mujer contra todo acto de discriminación;
- d) Abstenerse de incurrir en todo acto o práctica de discriminación contra la mujer y velar por que las autoridades e instituciones públicas actúen de conformidad con esta obligación;
- e) **Tomar todas las medidas apropiadas para eliminar la discriminación contra la mujer practicada por cualesquiera personas, organizaciones o empresas;**
- f) Adoptar todas las medidas adecuadas, incluso de carácter legislativo, para modificar o derogar leyes, reglamentos, usos y prácticas que constituyan discriminación contra la mujer;
- g) **Derogar todas las disposiciones penales nacionales que constituyan discriminación contra la mujer.**

Artículo 3

Los Estados Partes tomarán en todas las esferas, y en particular en las esferas política, social, económica y cultural, todas las medidas apropiadas, incluso de carácter legislativo, para asegurar el pleno desarrollo y adelanto de la mujer, con el objeto de garantizarle el ejercicio y el goce de los derechos humanos y las libertades fundamentales en igualdad de condiciones con el hombre.

Artículo 4

1. La adopción por los Estados Partes de medidas especiales de carácter temporal encaminadas a acelerar la igualdad de facto entre el hombre y la mujer no se considerará discriminación en la forma definida en la presente Convención, pero de ningún modo entrañará, como consecuencia, el mantenimiento de normas desiguales o separadas; estas medidas cesarán cuando se hayan alcanzado los objetivos de igualdad de oportunidad y trato.
2. La adopción por los Estados Partes de medidas especiales, incluso las contenidas en la presente Convención, encaminadas a proteger la maternidad no se considerará discriminatoria.

Artículo 5

Los Estados Partes tomarán todas las medidas apropiadas para:

- a) **Modificar los patrones socioculturales de conducta de hombres y mujeres, con miras a alcanzar la eliminación de los prejuicios y las prácticas consuetudinarias y de cualquier otra índole que estén basados en la idea de la inferioridad o superioridad de cualquiera de los sexos o en funciones estereotipadas de hombres y mujeres;**
- b) Garantizar que la educación familiar incluya una comprensión adecuada de la maternidad como función social y el **reconocimiento de la responsabilidad común de hombres y mujeres en cuanto a la educación y al desarrollo de sus hijos**, en la inteligencia de que el interés de los hijos constituirá la consideración primordial en todos los casos.

Artículo 6

Los Estados Partes tomarán todas las medidas apropiadas, incluso de carácter legislativo, para **suprimir todas las formas de trata de mujeres y explotación de la prostitución de la mujer.**

Parte II

Artículo 7

Los Estados Partes tomarán todas las medidas apropiadas para **eliminar la discriminación contra la mujer en la vida política y pública del país** y, en particular, garantizarán a las mujeres, en igualdad de condiciones con los hombres, el derecho a:

- a) Votar en todas las elecciones y referéndums públicos y ser elegibles para todos los organismos cuyos miembros sean objeto de elecciones públicas;
- b) Participar en la formulación de las políticas gubernamentales y en la ejecución de éstas, y ocupar cargos públicos y ejercer todas las funciones públicas en todos los planos gubernamentales;
- c) Participar en organizaciones y en asociaciones no gubernamentales que se ocupen de la vida pública y política del país.

Artículo 8

Los Estados Partes tomarán todas las medidas apropiadas para garantizar a la mujer, en igualdad de condiciones con el hombre y sin discriminación

alguna, la oportunidad de representar a su gobierno en el plano internacional y de participar en la labor de las organizaciones internacionales.

Artículo 9

1. Los Estados Partes otorgarán a las mujeres iguales derechos que a los hombres para adquirir, cambiar o conservar su nacionalidad. Garantizarán, en particular, que ni el matrimonio con un extranjero ni el cambio de nacionalidad del marido durante el matrimonio cambien automáticamente la nacionalidad de la esposa, la conviertan en ápatriada o la obliguen a adoptar la nacionalidad del cónyuge.
2. Los Estados Partes otorgarán a la mujer los mismos derechos que al hombre con respecto a la nacionalidad de sus hijos.

Parte III

Artículo 10

Los Estados Partes adoptarán todas las medidas apropiadas para eliminar la discriminación contra la mujer, a fin de asegurarle la igualdad de derechos con el hombre en la esfera de la educación y en particular para asegurar, en condiciones de igualdad entre hombres y mujeres:

- a) **Las mismas condiciones de orientación en materia de carreras y capacitación profesional**, acceso a los estudios y obtención de diplomas en las instituciones de enseñanza de todas las categorías, tanto en zonas rurales como urbanas; esta igualdad deberá asegurarse en la enseñanza preescolar,

general, técnica, profesional y técnica superior, así como en todos los tipos de capacitación profesional;

b) Acceso a los mismos programas de estudios, a los mismos exámenes, a personal docente del mismo nivel profesional y a locales y equipos escolares de la misma calidad;

c) **La eliminación de todo concepto estereotipado de los papeles masculino y femenino en todos los niveles y en todas las formas de enseñanza**, mediante el estímulo de la educación mixta y de otros tipos de educación que contribuyan a lograr este objetivo y, en particular, mediante la modificación de los libros y programas escolares y la adaptación de los métodos de enseñanza;

d) **Las mismas oportunidades para la obtención de becas y otras subvenciones para cursar estudios;**

e) Las mismas oportunidades de acceso a los programas de educación permanente, incluidos los programas de alfabetización funcional y de adultos, con miras en particular a reducir lo antes posible toda diferencia de conocimientos que exista entre hombres y mujeres;

f) La reducción de la tasa de abandono femenino de los estudios y la organización de programas para aquellas jóvenes y mujeres que hayan dejado los estudios prematuramente;

g) Las mismas oportunidades para participar activamente en el deporte y la educación física;

h) Acceso al material informativo específico que contribuya a asegurar la salud y el bienestar de la familia, incluida la información y el asesoramiento sobre planificación de la familia.

Artículo 11

1. Los Estados Partes adoptarán todas las medidas apropiadas para eliminar la discriminación contra la mujer en la esfera del empleo a fin de asegurar a la mujer, en condiciones de igualdad con los hombres, los mismos derechos, en particular:

a) El derecho al trabajo como derecho inalienable de todo ser humano;

b) **El derecho a las mismas oportunidades de empleo, inclusive a la aplicación de los mismos criterios de selección en cuestiones de empleo;**

c) **El derecho a elegir libremente profesión** y empleo, el derecho al ascenso, a la estabilidad en el empleo y a todas las prestaciones y otras condiciones de servicio, y el derecho a la formación profesional y al readiestramiento, incluido el aprendizaje, la formación profesional superior y el adiestramiento periódico;

d) **El derecho a igual remuneración, inclusive prestaciones, y a igualdad de trato con respecto a un trabajo de igual valor, así como a igualdad de trato con respecto a la evaluación de la calidad del trabajo;**

- e) El derecho a la seguridad social, en particular en casos de jubilación, desempleo, enfermedad, invalidez, vejez u otra incapacidad para trabajar, así como el derecho a vacaciones pagadas;
 - f) El derecho a la protección de la salud y a la seguridad en las condiciones de trabajo, incluso la salvaguardia de la función de reproducción.
2. A fin de impedir la discriminación contra la mujer por razones de matrimonio o maternidad y asegurar la efectividad de su derecho a trabajar, los Estados Partes tomarán medidas adecuadas para:
- a) Prohibir, bajo pena de sanciones, el despido por motivo de embarazo o licencia de maternidad y la discriminación en los despidos sobre la base del estado civil;
 - b) Implantar la licencia de maternidad con sueldo pagado o con prestaciones sociales comparables sin pérdida del empleo previo, la antigüedad o los beneficios sociales;
 - c) Alentar el suministro de los servicios sociales de apoyo necesarios para permitir que los padres combinen las obligaciones para con la familia con las responsabilidades del trabajo y la participación en la vida pública, especialmente mediante el fomento de la creación y desarrollo de una red de servicios destinados al cuidado de los niños;
 - d) Prestar protección especial a la mujer durante el embarazo en los tipos de trabajos que se haya probado puedan resultar perjudiciales para ella.

3. La legislación protectora relacionada con las cuestiones comprendidas en este artículo será examinada periódicamente a la luz de los conocimientos científicos y tecnológicos y será revisada, derogada o ampliada según corresponda.

Artículo 12

1. Los Estados Partes adoptarán todas las medidas apropiadas para eliminar la discriminación contra la mujer en la esfera de la atención médica a fin de asegurar, en condiciones de igualdad entre hombres y mujeres, el acceso a servicios de atención médica, inclusive los que se refieren a la planificación de la familia.

2. Sin perjuicio de lo dispuesto en el párrafo 1 supra, los Estados Partes garantizarán a la mujer servicios apropiados en relación con el embarazo, el parto y el período posterior al parto, proporcionando servicios gratuitos cuando fuere necesario, y le asegurarán una nutrición adecuada durante el embarazo y la lactancia.

Artículo 13

Los Estados Partes adoptarán todas las medidas apropiadas para eliminar la discriminación contra la mujer en otras esferas de la vida económica y social a fin de asegurar, en condiciones de igualdad entre hombres y mujeres, los mismos derechos, en particular:

a) El derecho a prestaciones familiares;

- b) El derecho a obtener préstamos bancarios, hipotecas y otras formas de crédito financiero;
- c) El derecho a participar en actividades de esparcimiento, deportes y en todos los aspectos de la vida cultural.

Artículo 14

1. Los Estados Partes tendrán en cuenta los problemas especiales a que hace frente la mujer rural y el importante papel que desempeña en la supervivencia económica de su familia, incluido su trabajo en los sectores no monetarios de la economía, y tomarán todas las medidas apropiadas para asegurar la aplicación de las disposiciones de la presente Convención a la mujer en las zonas rurales.
2. Los Estados Partes adoptarán todas las medidas apropiadas para eliminar la discriminación contra la mujer en las zonas rurales a fin de asegurar en condiciones de igualdad entre hombres y mujeres, su participación en el desarrollo rural y en sus beneficios, y en particular le asegurarán el derecho a:
 - a) Participar en la elaboración y ejecución de los planes de desarrollo a todos los niveles;
 - b) Tener acceso a servicios adecuados de atención médica, inclusive información, asesoramiento y servicios en materia de planificación de la familia;
 - c) Beneficiarse directamente de los programas de seguridad social;

- d) Obtener todos los tipos de educación y de formación, académica y no académica, incluidos los relacionados con la alfabetización funcional, así como, entre otros, los beneficios de todos los servicios comunitarios y de divulgación a fin de aumentar su capacidad técnica;
- e) Organizar grupos de autoayuda y cooperativas a fin de obtener igualdad de acceso a las oportunidades económicas mediante el empleo por cuenta propia o por cuenta ajena;
- f) Participar en todas las actividades comunitarias; g) Obtener acceso a los créditos y préstamos agrícolas, a los servicios de comercialización y a las tecnologías apropiadas, y recibir un trato igual en los planes de reforma agraria y de reasentamiento;
- h) Gozar de condiciones de vida adecuadas, particularmente en las esferas de la vivienda, los servicios sanitarios, la electricidad y el abastecimiento de agua, el transporte y las comunicaciones.

Parte IV

Artículo 15

1. Los Estados Partes reconocerán a la mujer la igualdad con el hombre ante la ley.
2. Los Estados Partes reconocerán a la mujer, en materias civiles, una capacidad jurídica idéntica a la del hombre y las mismas oportunidades para el ejercicio de esa capacidad. En particular, le reconocerán a la mujer iguales

derechos para firmar contratos y administrar bienes y le dispensarán un trato igual en todas las etapas del procedimiento en las cortes de justicia y los tribunales.

3. Los Estados Partes convienen en que todo contrato o cualquier otro instrumento privado con efecto jurídico que tienda a limitar la capacidad jurídica de la mujer se considerará nulo.

4. Los Estados Partes reconocerán al hombre y a la mujer los mismos derechos con respecto a la legislación relativa al derecho de las personas a circular libremente y a la libertad para elegir su residencia y domicilio.

Artículo 16

1. Los Estados Partes adoptarán todas las medidas adecuadas para eliminar la discriminación contra la mujer en todos los asuntos relacionados con el matrimonio y las relaciones familiares y, en particular, asegurarán en condiciones de igualdad entre hombres y mujeres:

a) El mismo derecho para contraer matrimonio;

b) El mismo derecho para elegir libremente cónyuge y contraer matrimonio sólo por su libre albedrío y su pleno consentimiento;

c) Los mismos derechos y responsabilidades durante el matrimonio y con ocasión de su disolución;

- d) Los mismos derechos y responsabilidades como progenitores, cualquiera que sea su estado civil, en materias relacionadas con sus hijos; en todos los casos, los intereses de los hijos serán la consideración primordial;
- e) **Los mismos derechos a decidir libre y responsablemente el número de sus hijos** y el intervalo entre los nacimientos y a **tener acceso a la información, la educación y los medios que les permitan ejercer estos derechos;** f) Los mismos derechos y responsabilidades respecto de la tutela, curatela, custodia y adopción de los hijos, o instituciones análogas cuando quiera que estos conceptos existan en la legislación nacional; en todos los casos, los intereses de los hijos serán la consideración primordial;
- g) Los mismos derechos personales como marido y mujer, entre ellos el derecho a elegir apellido, profesión y ocupación;
- h) Los mismos derechos a cada uno de los cónyuges en materia de propiedad, compras, gestión, administración, goce y disposición de los bienes, tanto a título gratuito como oneroso.

2. No tendrán ningún efecto jurídico los esponsales y el matrimonio de niños y se adoptarán todas las medidas necesarias, incluso de carácter legislativo, para fijar una edad mínima para la celebración del matrimonio y hacer obligatoria la inscripción del matrimonio en un registro oficial.

Parte V

Artículo 17

1. Con el fin de examinar los progresos realizados en la aplicación de la presente Convención, se establecerá un Comité para la Eliminación de la Discriminación contra la Mujer (denominado en adelante el Comité) compuesto, en el momento de la entrada en vigor de la Convención, de dieciocho y, después de su ratificación o adhesión por el trigésimo quinto Estado Parte, de veintitrés expertos de gran prestigio moral y competencia en la esfera abarcada por la Convención. Los expertos serán elegidos por los Estados Partes entre sus nacionales, y ejercerán sus funciones a título personal; se tendrán en cuenta una distribución geográfica equitativa y la representación de las diferentes formas de civilización, así como los principales sistemas jurídicos.
2. Los miembros del Comité serán elegidos en votación secreta de un lista de personas designadas por los Estados Partes. Cada uno de los Estados Partes podrá designar una persona entre sus propios nacionales.
3. La elección inicial se celebrará seis meses después de la fecha de entrada en vigor de la presente Convención. Al menos tres meses antes de la fecha de cada elección, el Secretario General de las Naciones Unidas dirigirá una carta a los Estados Partes invitándolos a presentar sus candidaturas en un plazo de dos meses. El Secretario General preparará una lista por orden alfabético de todas las personas designadas de este modo, indicando los Estados Partes que las han designado, y la comunicará a los Estados Partes.

4. Los miembros del Comité serán elegidos en una reunión de los Estados Partes que será convocada por el Secretario General y se celebrará en la Sede de las Naciones Unidas. En esta reunión, para la cual formarán quórum dos tercios de los Estados Partes, se considerarán elegidos para el Comité los candidatos que obtengan el mayor número de votos y la mayoría absoluta de los votos de los representantes de los Estados Partes presentes y votantes.
5. Los miembros del Comité serán elegidos por cuatro años. No obstante, el mandato de nueve de los miembros elegidos en la primera elección expirará al cabo de dos años; inmediatamente después de la primera elección el Presidente del Comité designará por sorteo los nombres de esos nueve miembros.
6. La elección de los cinco miembros adicionales del Comité se celebrará de conformidad con lo dispuesto en los párrafos 2, 3 y 4 del presente artículo, después de que el trigésimo quinto Estado Parte haya ratificado la Convención o se haya adherido a ella. El mandato de dos de los miembros adicionales elegidos en esta ocasión, cuyos nombres designará por sorteo el Presidente del Comité, expirará al cabo de dos años.
7. Para cubrir las vacantes imprevistas, el Estado Parte cuyo experto haya cesado en sus funciones como miembro del Comité designará entre sus nacionales a otro experto a reserva de la aprobación del Comité.
8. Los miembros del Comité, previa aprobación de la Asamblea General, percibirán emolumentos de los fondos de las Naciones Unidas en la forma y

condiciones que la Asamblea determine, teniendo en cuenta la importancia de las funciones del Comité.

9. El Secretario General de las Naciones Unidas proporcionará el personal y los servicios necesarios para el desempeño eficaz de las funciones del Comité en virtud de la presente Convención.

Artículo 18

1. Los Estados Partes se comprometen a someter al Secretario General de las Naciones Unidas, para que lo examine el Comité, un informe sobre las medidas legislativas, judiciales, administrativas o de otra índole que hayan adoptado para hacer efectivas las disposiciones de la presente Convención y sobre los progresos realizados en este sentido:

a) En el plazo de un año a partir de la entrada en vigor de la Convención para el Estado de que se trate;

b) En lo sucesivo por lo menos cada cuatro años y, además, cuando el Comité lo solicite.

2. Se podrán indicar en los informes los factores y las dificultades que afecten al grado de cumplimiento de las obligaciones impuestas por la presente Convención.

Artículo 19

1. El Comité aprobará su propio reglamento.

2. El Comité elegirá su Mesa por un período de dos años.

Artículo 20

1. El Comité se reunirá normalmente todos los años por un período que no exceda de dos semanas para examinar los informes que se le presenten de conformidad con el artículo 18 de la presente Convención.
2. Las reuniones del Comité se celebrarán normalmente en la Sede de las Naciones Unidas o en cualquier otro sitio conveniente que determine el Comité.

Artículo 21

1. El Comité, por conducto del Consejo Económico y Social, informará anualmente a la Asamblea General de las Naciones Unidas sobre sus actividades y podrá hacer sugerencias y recomendaciones de carácter general basadas en el examen de los informes y de los datos transmitidos por los Estados Partes. Estas sugerencias y recomendaciones de carácter general se incluirán en el informe del Comité junto con las observaciones, si las hubiere, de los Estados Partes.
2. El Secretario General de las Naciones Unidas transmitirá los informes del Comité a la Comisión de la Condición Jurídica y Social de la Mujer para su información.

Artículo 22

Los organismos especializados tendrán derecho a estar representados en el examen de la aplicación de las disposiciones de la presente Convención que correspondan a la esfera de las actividades. El Comité podrá invitar a los organismos especializados a que presenten informes sobre la aplicación de la Convención en las áreas que correspondan a la esfera de sus actividades.

Parte VI

Artículo 23

Nada de lo dispuesto en la presente Convención afectará a disposición alguna que sea más conducente al logro de la igualdad entre hombres y mujeres y que pueda formar parte de:

- a) La legislación de un Estado Parte; o
- b) Cualquier otra convención, tratado o acuerdo internacional vigente en ese Estado.

Artículo 24

Los Estados Partes se comprometen a adoptar todas las medidas necesarias en el ámbito nacional para conseguir la plena realización de los derechos reconocidos en la presente Convención.

Artículo 25

1. La presente Convención estará abierta a la firma de todos los Estados.
2. Se designa al Secretario General de las Naciones Unidas depositario de la presente Convención.
3. La presente Convención está sujeta a ratificación. Los instrumentos de ratificación se depositaran en poder del Secretario General de las Naciones Unidas.
4. La presente Convención estará abierta a la adhesión de todos los Estados. La adhesión se efectuará depositando un instrumento de adhesión en poder del Secretario General de las Naciones Unidas.

Artículo 26

1. En cualquier momento, cualquiera de los Estados Partes podrá formular una solicitud de revisión de la presente Convención mediante comunicación escrita dirigida al Secretario General de las Naciones Unidas.
2. La Asamblea General de las Naciones Unidas decidirá las medidas que, en caso necesario, hayan de adoptarse en lo que respecta a esa solicitud.

Artículo 27

1. La presente Convención entrará en vigor el trigésimo día a partir de la fecha en que haya sido depositado en poder del Secretario General de las Naciones Unidas el vigésimo instrumento de ratificación o de adhesión.

2. Para cada Estado que ratifique la Convención o se adhiera a ella después de haber sido depositado el vigésimo instrumento de ratificación o de adhesión, la Convención entrará en vigor el trigésimo día a partir de la fecha en que tal Estado haya depositado su instrumento de ratificación o de adhesión.

Artículo 28

1. El Secretario General de las Naciones Unidas recibirá y comunicará a todos los Estados el texto de las reservas formuladas por los Estados en el momento de la ratificación o de la adhesión.

2. No se aceptará ninguna reserva incompatible con el objeto y el propósito de la presente Convención.

3. Toda reserva podrá ser retirada en cualquier momento por medio de una notificación a estos efectos dirigida al Secretario General de las Naciones Unidas, quien informará de ello a todos los Estados. Esta notificación surtirá efecto en la fecha de su recepción.

Artículo 29

1. Toda controversia que surja entre dos o más Estados Partes con respecto a la interpretación o aplicación de la presente Convención que no se solucione mediante negociaciones se someterá al arbitraje a petición de uno de ellos. Si en el plazo de seis meses contados a partir de la fecha de presentación de solicitud de arbitraje las partes no consiguen ponerse de acuerdo sobre la forma del mismo, cualquiera de las partes podrá someter la controversia a la

Corte Internacional de Justicia, mediante una solicitud presentada de conformidad con el Estatuto de la Corte.

2. Todo Estado Parte, en el momento de la firma o ratificación de la presente Convención o de su adhesión a la misma, podrá declarar que no se considera obligado por el párrafo 1 del presente artículo. Los demás Estados Partes no estarán obligados por ese párrafo ante ningún Estado Parte que haya formulado esa reserva.
3. Todo Estado Parte que haya formulado la reserva prevista en el párrafo 2 del presente artículo podrá retirarla en cualquier momento notificándolo al Secretario General de las Naciones Unidas.

Artículo 30

La presente Convención, cuyos textos en árabe, chino, español, francés, inglés y ruso son igualmente auténticos, se depositarán en poder del Secretario General de las Naciones Unidas.

En testimonio de lo cual, los infrascritos, debidamente autorizados, firman la presente Convención.

LÓGICA Y
EPISTEMOLOGÍA

PARTE I: EPISTEMOLOGÍA

HALLAZGO Y DETERMINACIÓN DEL CONOCIMIENTO CIENTÍFICO

Eugenia Bahit

Desde la perspectiva científica, no todo el conocimiento es válido. Las bases sobre las que se define qué es aquello que se considera válido como conocimiento desde el punto de vista de la ciencia, son establecidas por disciplinas metacientíficas como la epistemología y la lógica.

El objetivo de este apartado es exponer a modo sintético y resumido, los principales conceptos, derivados de estas disciplinas, sobre los que se fundamenta el análisis del conocimiento para determinar su validez.

Dado el carácter sintético y resumido de esta sección, de ningún modo puede considerarse como la única base de estudio necesaria para la validación del conocimiento.

A fin de alcanzar una mayor fiabilidad en el análisis del conocimiento obtenido, cada disciplina y concepto aquí explicado, podría ser estudiado en

profundidad, más allá de lo que a continuación se expone a modo meramente orientativo.

Validez del conocimiento

Desde la perspectiva científica, la validez del conocimiento se determina por su tipo, y el tipo, por el método que haya sido aplicado para su hallazgo. Según el método empleado, es el tipo de conocimiento que se obtiene. En este sentido, es posible diferenciar dos métodos:

1. El método científico, mediante el cual se obtiene conocimiento científico.
2. Un método no científico o natural, por el cual se obtiene conocimiento ordinario o cotidiano. Esta categoría encierra a cualquier método distinto al método científico.

La clasificación anterior deriva en dos categorías contrarias, es decir, dos categorías directamente opuestas. Para evitar confundir estas dos categorías con dos categorías contradictorias (ver diferencia entre contrario y contradictorio), se emplearán los términos «**método científico**», en referencia al método epistemológicamente aceptado para el hallazgo de conocimiento científico válido, y «**método no científico**», en referencia a cualquier otro método.

A fin de lograr una comprensión integral de lo que el método científico representa, se llegará a su definición partiendo de los conceptos fundamentales sobre los que se concibe, a fin de ofrecer un marco de orientación

introductorio de aquellos elementos implicados en los mecanismos de producción y validación del conocimiento científico.

Metadisciplinas que definen los criterios de validación del conocimiento

Las metadisciplinas son aquellas disciplinas que tienen por objeto el estudio de la propia disciplina. Así, una metaciencia, es una metadisciplina que por objeto tiene el estudio de la ciencia.

Epistemología

La epistemología es la rama de la filosofía que tiene por objeto el estudio del conocimiento científico, y define los criterios a los que debe someterse toda investigación científica.

Como metadisciplina, hace uso de la lógica para determinar las características que debe cumplir el lenguaje científico, así como la estructura, forma y método que debe seguirse para la validación de enunciados.

Lógica

La lógica es la disciplina que tiene a su cargo el estudio de los métodos que permiten establecer la validez de un razonamiento, entendiendo como tal al proceso mental que, partiendo de ciertas premisas, deriva en una conclusión inferida sobre la base de éstas. Se hace preciso comprender que como disciplina, la lógica se centra únicamente en el establecimiento de la validez

argumental y no así en su veracidad (véase «*Diferencia entre verdad y validez*»).

Semiotica

La semiotica (o «Teoría General de los Signos» de C. S. Peirce²) es la metadisciplina que se encarga del estudio de los signos, y de entender la relación de los signos entre sí, su significado y uso.

Contraste de conceptos

Algunos términos suelen confundirse debido a su similitud fonética, su utilización poco precisa, u otros factores. La siguiente lista pretende esclarecer el significado de conceptos frecuentes en el estudio del conocimiento, a fin de evitar interpretaciones ambiguas.

Diferencia entre divulgación científica y comunicación científica

El concepto de «*divulgación científica*» hace referencia a aquel que busca poner el conocimiento científico al alcance del público en general, mientras que el de «*comunicación científica*», busca transmitir el conocimiento científico dentro de la propia comunidad científica.

2 Ver Charles Sanders Pierce en «The Stanford Encyclopedia of Philosophy»
<https://plato.stanford.edu/entries/peirce/>

Ambos conceptos buscan transmitir el conocimiento científico, y se diferencian por el público destinatario al que apuntan, y por lo tanto, en la forma del discurso utilizado.

Se trata de dos conceptos contradictorios pero no contrarios, es decir, que no son directamente opuestos.

Diferencia entre contrario y contradictorio

En el análisis lógico, existe una diferencia conceptual entre contrario y contradictorio, entendiéndose por «afirmaciones contrarias» a proposiciones directamente opuestas, y por «afirmaciones contradictorias», a aquellas que de forma contrastada se invalidan mutuamente.

Cuando se habla de «términos directamente opuestos», se hace referencia a la negación del propio término. Por ejemplo, lo opuesto a algo «blanco» es algo «no blanco» y lo opuesto a «no blanco» es «no (no blanco)» (debe considerarse el principio de doble negación por el cual de una doble negación se obtiene una afirmación). Sin embargo, si se afirma que un objeto es negro, y que el mismo objeto es gris, se incuraría en una contradicción puesto que afirmar que el objeto es gris, contradice la afirmación de que el objeto es negro.

La importancia de esta diferenciación, a nivel de evaluación del conocimiento científico, radica en la categorización y agrupamiento de la información. A modo de ejemplo, se plantean dos conjuntos opuestos: uno, formado por todos los objetos de color blanco, y el otro, el opuesto. Este último, incluiría todos

los objetos que NO resultasen verdaderos a la afirmación de que son de color blanco. Sin embargo, si se formaran conjuntos de objetos de colores contradictorios, podría existir un conjunto para cada color, e incluso, un conjunto por cada tonalidad o matiz de un mismo color.

Diferencia entre verdad y validez

La veracidad de un enunciado se reduce a la capacidad de establecer como cierta o no cierta, una afirmación. Son verdaderas (o falsas) las afirmaciones que se consideran ciertas debido a la evidencia o a la naturaleza de las mismas. No pueden considerarse verdaderos o falsos, enunciados que no afirman o nieguen algo. Esto implica que, entre otras, una pregunta, exclamación, u orden imperativa, no podrían considerarse verdaderas o falsas, dado que no niegan ni afirman nada.

La validez, sin embargo, representa la medida de análisis de lo correcto o incorrecto desde el punto de vista lógico. Esto implica que dado un argumento cuyas premisas se presumen verdaderas, este será válido si su forma es correcta, o inválido en caso contrario.

Cabe mencionar que la lógica es el estudio de la forma de un argumento, y por lo tanto, a través de ella, puede determinarse la validez del mismo.

Por ejemplo, la afirmación «el perro es el mejor amigo del hombre» puede considerarse tanto verdadera como falsa. El hecho de que sea susceptible a ser considerada verdadera o falsa, es lo que la califica como proposición válida. Es decir que es válida como premisa, aunque pueda no ser verdadera.

Diferencia entre verdad formal y verdad fáctica

Una *verdad formal* es una consecuencia lógica de proposiciones previamente establecidas como verdaderas. Por ello, una verdad formal es una verdad lógica.

Una *verdad fáctica*, en cambio, es toda afirmación confirmada por experimentación a través de un número limitado de casos.

Diferencia entre razonamiento y pensamiento

Mientras que por *razonamiento* se entiende al proceso mental que partiendo de ciertas premisas deriva en una conclusión inferida sobre la base de estas, por *pensamiento* se hace referencia al conjunto de todos los procesos mentales de cualquier índole y naturaleza, pudiendo abarcar desde un proceso imaginativo, e incluso un recuerdo hasta una asociación libre y al propio razonamiento.

A modo de ejemplo, se propone el siguiente razonamiento deductivo:

Todas las hojas de aquel libro son hojas de color tiza.

Todas las hojas de color tiza me resultan agradables.

Luego, todas las hojas de aquel libro me resultan agradables.

La imagen mental que pueda producirse en el cerebro durante la lectura del razonamiento anterior, sería un ejemplo de pensamiento que no constituye un razonamiento. Algo similar sucedería con la orden imperativa «*préstame el libro que tiene hojas de color tiza*» o con la pregunta «*¿te resultan agradables los libros cuyas hojas son de color tiza?*».

Cuando un razonamiento válido se componga solo de premisas verdaderas se lo denominará *razonamiento sólido*. Los razonamientos sólidos, son los que establecen la veracidad lógicamente válida, de su conclusión.

Diferencia entre deducción e inducción

Una *deducción* es aquella conclusión que se desprende necesariamente de las proposiciones de un razonamiento válido. A este tipo de razonamiento se lo denomina *razonamiento deductivo* y es el único proceso de razonamiento que puede ofrecer pruebas determinantes sobre la conclusión a la que se alcanza. En palabras de Irving Copi, «*un razonamiento deductivo es aquel cuyas premisas se pretende que suministren pruebas concluyentes para afirmar la verdad de su conclusión*» (Introducción a la lógica, pág. 167, Editorial EUDEBA, 2014).

La *inducción*, sin embargo, es aquella que implica un grado de *probabilidad* aparentemente suficiente sobre su conclusión, pero donde esta, no se deduce necesariamente de sus premisas.

Diferencia entre argumento y explicación

En este texto se habla de argumento en cuanto su primera acepción. Mediante esta, se denomina «*argumento*» a aquel razonamiento deductivo que ofrece en sus premisas las pruebas de su inferencia (es decir, de su conclusión). Sin embargo, una explicación solo establece relaciones causales, temporales (u otras no lógicas) entre el enunciado y su conclusión.

En las conexiones causales puede ser habitual encontrar explicaciones tales como «La aplicación “A” es más segura que la aplicación “B” puesto que “A” fue desarrollada con “X” marco de trabajo». En esta explicación, se expone que la seguridad de la aplicación “A” es *«a causa»* del uso de “X”, sin embargo, no se ofrecen pruebas concluyentes de que así lo sea.

Algo similar sucede al decir que «desde que “A” emplea “X” marco de trabajo, sus aplicaciones han resultado ser más seguras». La conexión que se realiza entre el uso de “X” y la seguridad de las aplicaciones desarrolladas por “A”, es una conexión temporal, dado que se expone que la seguridad de las aplicaciones desarrolladas por “A” mejoró *«en el mismo momento»* en el que “A” comenzó a utilizar “X”. Es decir, se establece una coincidencia en el tiempo entre dos hechos, pero no se ofrece una prueba contundente de que el uso de “X” sea la razón de que “A”, genere aplicaciones más seguras.

Diferencia entre objetivo y subjetivo

Más adelante se hablará del concepto de *«objetividad científica»* el cual extiende el concepto explicado en los párrafos siguientes. A continuación, se abarcan los conceptos de objetividad y subjetividad, desde una perspectiva general, menos específica que la que se tratará más adelante.

Cuando se habla de *objetividad* en el discurso, se hace referencia a la independencia de los pensamientos, emociones y sentimientos propios de quien expone, con respecto al objeto que expone. Esto se refiere a la capacidad de exponer un objeto determinado sin que en la exposición interfieran cualidades

intrínsecas de quien expone, tales como sus propios pensamientos, sus creencias, ideologías, sentimientos y emociones con respecto al objeto.

Por ejemplo, en la frase *«la ventaja de usar “X” herramienta es que permite corregir el código fuente en tiempo de programación»*, el objeto es *«“X” herramienta»*, y una de sus características es que *«se puede activar la corrección de código fuente en tiempo de programación»*. Por lo tanto, la frase original no es objetiva, ya que describe las características del objeto desde la percepción personal de quien relata, al transmitir su propia impresión de dicha característica como una ventaja, es decir, como algo beneficioso. Sería posible estar o no de acuerdo con que dicha característica es o no una ventaja. Sin embargo no es el objetivo de este párrafo. El objetivo de este párrafo es demostrar que la frase *«la ventaja de usar “X” herramienta es que permite corregir el código fuente en tiempo de programación»* no es una frase objetiva por incluir la percepción personal de quien la expone. El mismo concepto, expuesto de forma objetiva, y teniendo en cuenta que dicha característica pudiese tanto activarse como desactivarse, podría ser similar a *«una característica de “X” herramienta es la opción de activar o desactivar la corrección de código fuente en tiempo de programación»*. De esta forma, se estaría describiendo objetivamente, sin omitir que se trata de una opción que puede utilizarse tanto como descartarse. Por lo tanto, no se la expone ni como ventaja ni desventaja, y se deja tal análisis a las personas receptoras del discurso.

La *subjetividad*, representa exactamente lo opuesto a la objetividad. Es cuando el discurso va más allá del propio objeto, y la exposición de éste, es

interferida por los pensamientos, ideologías, creencias, sentimientos, emociones, etc. de la persona que expone, es decir que no se centra en el objeto sino en el sujeto que lo expone. La frase del ejemplo anterior, por lo tanto, constituiría una frase subjetiva.

Diferencia entre «bueno y malo», «correcto e incorrecto»

Desde el punto de vista metacientífico, el conocimiento no puede calificarse ni clasificarse de forma ambigua. El concepto del bien y el mal o de lo bueno y lo malo, es algo intrínseco de una persona, y responde a cuestiones culturales, éticas y/o morales, inherentes a cada ser, lo que convierte a ambos conceptos, en ambiguos, dado que variarán acorde a quien los interprete.

Desde el punto de vista lógico y epistemológico, el conocimiento solo puede clasificarse como «correcto o incorrecto» sobre la base de su concepción racional. Esto significa, que un conocimiento concebido a partir de un proceso de razonamiento deductivo válido, se considera correcto desde el punto de vista lógico, y cualquier otro conocimiento, no.

No debe asociarse el concepto de «correcto» del que hace uso la lógica, al concepto de «bueno» tal y como se desprende de aspectos relacionados a la moral, la ética de cualquier índole, lo espiritual y/o religioso. En lógica, *correcto* significa que algo es válido desde el punto de vista formal. Esto significa explícitamente, que algo correcto implica que su forma argumental es válida.

Desde el punto de vista epistemológico, lo correcto es toda afirmación que desde la perspectiva científica se considera válida.

Desde un punto de vista psicosocial, lo bueno y lo malo tiene una relación directa con el impacto emocional que un determinado conocimiento o información, genera en la persona receptora de dicho conocimiento. Pero no es el conocimiento en sí mismo el que desde lo psicosocial se cataloga como bueno o malo, sino el efecto que este genera en cada persona, sobre la base de sus propias convicciones y/o creencias.

Por ejemplo, la frase «*el aborto es una práctica médica*», desde un punto de vista metacientífico, puede ser catalogada como verdadera o falsa, considerándose «correcta» solo si la frase se cataloga como «verdadera». Para diferentes personas, con diferentes creencias y convicciones, esta frase podría impactar emocionalmente de formas diferentes, pudiendo resultar en una emoción agradable, desagradable, o apática. Sin embargo, el conocimiento por sí solo, únicamente puede considerarse verdadero, sí y solo sí, se ofrecen pruebas científicamente válidas, o falso, sí y sólo sí, no se ofrecen pruebas científicamente válidas. La forma del conocimiento, será la que se considere válida, o no.

Desde el punto de vista metacientífico, será un conocimiento correcto, aquel conocimiento verdadero que posea una forma válida, independientemente del impacto que pueda o no, generar su contenido.

Diferencia entre falacia y sesgo cognitivo

En el estudio de la lógica, se denomina falacia a todo aquel argumento psicológicamente persuasivo, que sin ser válido lo aparenta. En «Introducción a la lógica» Irving Copi, define dos grandes tipos de falacias: las *falacias de atinencia* (aquellas cuyas premisas carecen de relación lógica con respecto a su conclusión), y las *falacias de ambigüedad*, donde el razonamiento presenta palabras o frases ambiguas que van, sutilmente, modificando su significado a lo largo de todo el razonamiento. Dentro de esta clasificación, puede hallarse una subdivisión de tipos de falacias más concreta.

En psicología cognitiva, se denomina *sesgo cognitivo* a un error sistemático de «pensamiento», que conduce a afirmaciones falsas, formuladas a partir de procesos heurísticos. Esto es, que se trata de procesos de pensamiento basados en juicios subjetivos realizados a partir del conocimiento parcial de uno o más eventos auto experimentados, de la intuición, suposiciones y creencias individuales.

Tanto la falacia como el sesgo cognitivo, conducen a afirmaciones falsas, por distintos motivos y de forma diferente. No obstante, una falacia podría ser producto de un sesgo cognitivo o ser psicológicamente persuasiva por apelar a un sesgo cognitivo para ser creíble. Tienen en común, que en ambos casos, se trata de afirmaciones falsas.

Diferencia entre dato anecdótico y evidencia científica

La anécdota es un dato curioso y circunstancial, generalmente no reproducible, que depende de una circunstancia concreta. Por el contrario, la evidencia es una demostración clara y manifiesta más allá de toda duda razonable. La anécdota no explica racionalmente el proceso mediante el cuál, se llega a inferir que lo que se afirma es verdadero; mientras que la evidencia, expone dicho proceso de forma manifiesta.

Cuando a partir de una anécdota, es decir, de un hecho circunstancial, se extraen datos que resultan útiles o convenientes para apoyar una afirmación de la que no se tiene evidencia científica, se obtiene lo que se conoce como *dato anecdótico*.

Por lo tanto, mientras que la evidencia científica es una demostración formal o empírica, clara y manifiesta obtenida a partir de un método válido desde una perspectiva epistemológica, el dato anecdótico puede ser cualquier información obtenida a partir de cualquier hecho circunstancial pasado.

Diferencia entre dogma y explicación científica

El dogma es una proposición acabada (es decir, no es flexible a cambios) y que se acepta como cierta e innegable, sin posibilidad de ser sometida a ensayo. Esto implica que como tal, no puede ser refutada.

La explicación científica es una hipótesis provisoria sujeta a falsabilidad. Esto implica, que aquello que se afirma puede ser refutado si se lo somete a prueba empírica, y nunca es final ni definitivo.

Diferencia entre creencia y convicción

Una creencia es un acto voluntario en el que se da crédito absoluto y se asume como cierta una afirmación o hecho determinado, sin mediar un análisis racional ni constatación (formal o empírica) de aquello que se afirma. Se califica como irracional por el hecho de ser asumida como verdad sin proceso de razonamiento previo. En este sentido, no debe confundirse el término «irracional» con el de «demencial». Solo es irracional porque su análisis no se ciñe a un proceso de razonamiento.

Se entiende por convicción al acto de convencerse sobre una verdad que a partir de las pruebas presentadas, no puede ser negada razonadamente.

Una creencia irracional puede ser temporal (permanecer como idea solo un tiempo hasta que quien la posea, la abandone por diferentes circunstancias), o bien permanente (generalmente, cuando la creencia es dogmática, y sobrevive a cualquier argumento racional que pueda demostrar su falsedad).

La convicción, sin embargo, siempre es provisoria (se acepta como verdad hasta que existan pruebas que la contradigan). Una convicción permanente, se convertiría en creencia dogmática, puesto que no estaría sujeta a cambios incluso cuando se demostrara su falsedad.

No obstante, un mismo hecho o afirmación, incluso cuando verdaderos, pueden aceptarse como válidos, tanto por convicción como por fe. En este sentido, puede decirse que sin importar la veracidad o falsedad de una afirmación, el mecanismo de aceptación y su temporalidad, determinan si

quién acepta la proposición, la incorpora por fe o por convicción. Por lo tanto, la aceptación no modifica la validez del enunciado (ni su veracidad), sino que solo afecta individualmente a quién lo acepta.

Diferencia entre ciencia formal y ciencia fáctica

Una ciencia formal es aquella ciencia cuya validación se realiza por medio de «la forma del razonamiento». Las Ciencias Matemáticas son ciencias puramente formales.

Las ciencias fácticas son aquellas ciencias que requieren de demostración empírica, es decir, que demandan probar sus teorías mediante la experimentación. De esta forma, toda teoría afirma una verdad provisoria que está sujeta a ser falsada por los hechos. En este sentido, para demostrar una teoría se necesita experimentar de forma repetida con una muestra suficientemente significativa, mientras que solo se necesita un experimento fallido para falsarla.

Diferencia entre axioma y postulado

Un *axioma* es un principio indemostrable que se acepta como punto de partida para el desarrollo de una teoría formal. Un *postulado* es una proposición que se acepta como verdadera sin necesidad de demostración.

Tienen en común su aceptabilidad sin la exigencia de pruebas. Sin embargo, la diferencia fundamental, radica en dos aspectos:

1. Falsabilidad. Mientras el axioma es indemostrable, el postulado puede demostrarse y por lo tanto, ser falsado.
2. Contexto. Mientras que el axioma sirve como principio fundamental sobre el cual construir una teoría matemática, el postulado sirve de premisa para razonamientos ulteriores, pero no para la elaboración de teorías.

Diferencia entre observación y demostración

La observación es el acto de examinar un hecho o suceso, y describirlo de forma cuantitativa o cualitativa.

La demostración, en cambio, es la acción de explicar y comprobar, mediante métodos específicos, reproducibles y falsables, la forma en la que se produce un determinado hecho o suceso.

A modo de ejemplo, se puede observar que en N casos examinados, toda vez que sucede “A”, se produce “B”, pero no se explica cómo “A” produce “B”, por lo tanto, solo «se observa “B” luego de “A”, pero no se demuestra que “A” sea la causa de “B”».

Por ello, se observan hechos o sucesos no necesariamente atinentes, y se demuestran teorías necesariamente atinentes al hecho o suceso que se observa.

Diferencia entre ciencia y pseudociencia

Se habla de ciencia y pseudociencia en el ámbito de las ciencias no formales. En este contexto, ciencia se refiere a las ciencias fácticas, descritas anteriormente.

Una ciencia es una disciplina que por medio de un método estudiado, aceptado y reproducible, emplea el razonamiento y la observación, para producir conocimiento demostrable mediante la experimentación, y a partir del cual pueden inferirse principios y leyes generales capaces de predecir sucesos futuros, y teorías falsables.

Pseudo- (o *seudo*), es un elemento compositivo del idioma español, que significa *falso*. *Pseudo-* más *ciencia*, significa entonces, «falsa ciencia». En este sentido, puede definirse una pseudociencia como una disciplina que en apariencia puede parecer o proclamarse como científica, pero que no presenta una o más de las características (enunciadas en el párrafo precedente), presentes en las disciplinas científicas.

Diferencia entre escepticismo científico y pensamiento crítico

El escepticismo, en términos generales, es la duda o desconfianza que se tiene sobre una afirmación. En ese sentido, podría ser correcto definir el escepticismo científico como la «duda o desconfianza que se tiene sobre una afirmación, desde una perspectiva científica». Es decir, no es dudar de la ciencia sino, dudar haciendo uso de ella.

El pensamiento crítico es una cualidad cognitiva que hace referencia a la capacidad de analizar ideas, hechos, situaciones hasta incluso afirmaciones y teorías, mediante el uso simultáneo de diferentes formas de pensamiento, que abarcan desde el análisis racional hasta el pensamiento creativo, abstracto y lateral.

En este sentido, mientras que el escepticismo científico plantea de forma concreta, la necesidad de hacer uso del análisis racional y la constatación científica para aceptar como válida y verdadera una afirmación, el pensamiento crítico se refiere a una cualidad más abstracta y de ámbito más amplio.

Diferencia entre ciencia teórica y ciencia aplicada

Una ciencia aplicada es aquella que enfoca el estudio en razón de su utilidad, mientras que la ciencia teórica (también llamada ciencia básica o pura), es aquella que se enfoca en la obtención del conocimiento en sí mismo, con independencia de cualquier aplicación.

Diferencia entre teoría, teorema, tesis e hipótesis

Una *hipótesis* es la suposición que se establece de forma provisoria como base para una investigación científica y tiene por objetivo la confirmación o desestimación de algo que se supone posible.

La *tesis* es una conclusión confirmada obtenida a partir del razonamiento. En este sentido, la tesis supone la confirmación de la hipótesis.

La *teoría* es una hipótesis provisoria de algo que aún no ha sido falsado.

Falsar una teoría, significa desmentirla mediante pruebas empíricas.

A diferencia de una teoría, el *teorema* es una proposición solo demostrable a través de la lógica, que no necesita ser falsada.

NOCIONES DE LÓGICA Y LAS FORMAS DEL RAZONAMIENTO

Eugenia Bahit

Introducción

La lógica es la disciplina que determina el modo correcto de razonar. Razonar de la forma correcta es lo que permitirá evitar las falacias y reconocer los sesgos cognitivos. Esto implica, en otras palabras, reconocer un porcentaje de errores de pensamiento de distinta clase. Por regla general, todo lo que parta de un error de pensamiento, conducirá a una conclusión equivocada, bien sea esta una mera afirmación o supuesto hallazgo, o bien, la solución a un problema.

Entre los elementos principales que la lógica otorga, se propone el estudio de la forma de los enunciados. Conocer y estudiar la forma de los enunciados, es lo que permitirá categorizar los componentes del enunciado, mientras que la categorización de dichos componentes, permitirá establecer la forma en la que

los componentes del enunciado se relacionan entre sí. Esta relación es la que permite saber cómo los elementos de un componente se distribuyen con respecto al otro. Esto es en definitiva, lo que evitará cometer errores de distribución, es decir, abarcar más de lo que abarcan los enunciados, algo que conduciría a efectuar afirmaciones falsas.

Se describirán por lo tanto:

- los dos tipos de enunciados posibles (categóricos e hipotéticos) ;
- la forma típica de los enunciados y la distribución de los términos;
- la naturaleza formal de los silogismos como mecanismo para verificar la validez de enunciados y argumentos;
- la forma de traducir el lenguaje cotidiano en afirmaciones categóricas (válidas como premisas), para evitar errores de razonamiento en los silogismos.

Enunciados categóricos

Los enunciados categóricos son afirmaciones compuestas por dos términos, sujeto y predicado, donde la categoría del primer término, se incluye o excluye, total o parcialmente, de la categoría del segundo término.

Forma típica de un enunciado categórico

En su forma típica, un enunciado categórico se compone de:

Elemento	Descripción	Valores posibles
<i>Cuantificador</i>	Cuantificador lógico	Todos, algunos, ninguno
<i>S</i>	Término sujeto	Cualquier clase
<i>Verbo</i>	Verbo copulativo, cuya única función sea unir a S y P	ser, estar
<i>P</i>	Término predicado	Cualquier clase

<cuantificador> <S> <verbo> <P>

Los términos sujeto y predicado pueden ser cualquier categoría formada por un sustantivo e incluyendo cualquier modificador directo o indirecto.

Todos_{cuantificador} los árboles del parque_S son_{verbo} álamos_P.

Los enunciados categóricos pueden no tener todos sus componentes de forma explícita.

Por ejemplo, en este enunciado, el cuantificador lógico «todos» está implícito:

*Los árboles del parque son álamos.
¿Cuántos árboles del parque son álamos? Todos.
[TODOS] los árboles del parque son álamos.*

Y aquí, el sustantivo del término predicado, «árboles», también está implícito:

Todos los árboles del parque son de color azul.

¿Quiénes son de color azul? Los árboles.

Todos los árboles del parque son [ÁRBOLES] de color azul.

Y a veces, más de un componente puede estar tácito, aparecer en otro orden, o no estar el verbo copulativo:

En el desierto las arenas se mueven solas.

[TODAS] las arenas en el desierto [SON ARENAS QUE] se mueven solas.

Si bien, como se ha visto, los enunciados pueden no aparecer en una forma sintáctica explícita, sí pueden ser traducidos para encajar en dicha forma, y permitir su correcta categorización. Los enunciados ya traducidos, pueden entonces, encontrarse en cuatro formas diferentes, divididas en dos grandes grupos:

- **Formas universales:** las que incluyen los cuantificadores lógicos todos y ninguno.
- **Formas particulares:** las que utilizan el cuantificador lógico algunos.

Cada uno de estos grupos, tendrá dos formas:

- Forma afirmativa
- Forma negativa

Logrando así un total de cuatro formas típicas:

Forma típica			Esquema
Universal	afirmativa	A	Todos los <i>S</i> son <i>P</i> .
	negativa	E	Ningún <i>S</i> es <i>P</i> .
Particular	afirmativa	I	Algunos <i>S</i> son <i>P</i> .
	negativa	O	Algunos <i>S</i> no son <i>P</i> .

Distribución

Ahora, según la forma de un enunciado (A, E, I u O), es posible establecer la relación existente entre sujeto y predicado, para determinar cómo una categoría se distribuye con respecto a otra.

Se dice que un término está distribuido cuando se hace referencia al total de la categoría. Por ejemplo, al decir «todos los patos» se está haciendo referencia al total de la categoría patos, mientras que si se modifica el cuantificador por «algunos», la categoría «patos» ya no estaría distribuida, porque solo se haría mención a la categoría parcial, y no total.

*

La distribución de los términos sujeto y predicado, según la forma del enunciado, se define en la siguiente tabla:

Forma típica			Esquema	Términos distribuidos
Universal	afirmativa	A	Todos los <i>S</i> son <i>P</i> .	S
	negativa	E	Ningún <i>S</i> es <i>P</i> .	S / P
Particular	afirmativa	I	Algunos <i>S</i> son <i>P</i> .	-
	negativa	O	Algunos <i>S</i> no son <i>P</i> .	P

Para entender mejor la distribución de un término con respecto al otro, a continuación se expone una explicación con ejemplos.

Distribución del sujeto en la forma «A»

En el mundo existe un total *N* de personas menores de 20 años. A su vez, en el curso de inglés del instituto «Y», todos los estudiantes del curso son menores de 20 años. Por lo tanto, afirmar que *«todos los estudiantes de inglés del instituto “Y” son personas menores de 20 años»* sería también equivalente a decir que *«todos los estudiantes de inglés del instituto “Y” son [ALGUNAS DE LAS] personas menores de 20 años»*.

Distribución del sujeto y el predicado en la forma «E»

En la forma universal negativa de este mismo enunciado, se estaría afirmando que *«Ningún estudiante del curso de inglés del instituto “Y” es una persona*

menor de 20 años». Esta afirmación, sería equivalente a decir que «*Ninguna persona menor de 20 años es estudiante del curso de inglés del instituto “Y”*».

Distribución del predicado en la forma «O»

En la forma particular negativa del enunciado, es válido para el predicado, lo dicho al ejemplificar la distribución del predicado en la forma «E», puesto que al afirmar que «*Algunos estudiantes del curso de inglés del instituto “Y” no son menores de 20 años*», se está confirmando que «*Ninguna persona menor de 20 años es estudiante del curso de inglés del instituto “Y”*».

No distribución del sujeto ni del predicado en la forma «I»

En la forma particular afirmativa, en cambio, siempre se hace referencia a conjuntos parciales. Al afirmar que «*[solo] algunos de los estudiantes del curso de inglés del instituto “Y” son personas menores de 20 años*», se hace tan imposible asegurar que «*Ninguna persona menor de 20 años es estudiante del curso de inglés del instituto “Y”*» —pues algunas sí lo son—, como asegurar que «*Todas las personas menores de 20 años son estudiantes del curso de inglés del instituto “Y”*» —pues solo se sabe que algunos sí lo son, pero podría ser que algunos no lo fueran—.

Solo los términos «todos» y «ninguno» incluyen y excluyen respectivamente, elementos del total de una categoría. Por eso son universales, ya que hacen referencia al universo de dicha categoría. Sin embargo, el cuantificado «algunos», hace una referencia parcial.

Enunciados categóricos compuestos

Los enunciados categóricos compuestos, son enunciados categóricos simples concatenados, que pueden surgir a partir de una traducción del lenguaje cotidiano. «Algunas manzanas son rojas y otras son verdes» es la conjunción de «algunas manzanas son manzanas rojas» y «algunas manzanas son manzanas verdes». La composición de enunciados concatenados por la conjunción «y» da origen a los denominados *enunciados conjuntivos*.

La misma conjunción, aunque implícita, se da con los *enunciados exceptivos* (que hacen excepciones), cuyos cuantificadores no son lógicos, como sucede en «todas las manzanas son rojas excepto las verdes», que sería equivalente al ejemplo planteado en el párrafo anterior.

Otros enunciados categóricos compuestos, son los *enunciados disyuntivos*. Estos enunciados plantean opciones exclusivas, como en «las manzanas son o bien rojas, o bien verdes» (pero no pueden ser rojas y verdes al mismo tiempo), o inclusivas, como «el cajón de manzanas es un cajón de manzanas rojas o verdes, o ambas».

Los enunciados disyuntivos exclusivos plantean dos o más opciones de las cuales solo una puede ser verdadera, mientras que los enunciados disyuntivos inclusivos plantean dos o más opciones, más una combinación de éstas, donde una más de las mismas puede ser verdadera.

Para ejemplificarlo, suponiendo que «a» y «b» sean dos variables, serían verdaderos:

- Enunciado conjuntivo: $(a \text{ AND } b)$.
- Enunciado disyuntivo exclusivo: $(a \text{ OR } b) \text{ AND NOT } (a \text{ AND } b)$.
- Enunciado disyuntivo inclusivo: $(a \text{ OR } b) \text{ OR } (a \text{ AND } b)$.

En el conjuntivo, tanto «a» como «b» deben ser verdaderas. En el disyuntivo exclusivo, «a» o «b» deben ser verdaderas pero no ambas, y en el disyuntivo inclusivo, «a», «b» o ambas, deben ser verdaderas.

Enunciados hipotéticos

Los enunciados hipotéticos son enunciados condicionales, es decir que afirman una consecuencia que se cumple toda vez que una condición sea verdadera.

Si <condición> entonces <consecuencia>

Los enunciados hipotéticos son, hoy en día, la base del método científico de las ciencias fácticas.

La condición de un enunciado hipotético es un antecedente que, de ser verdadero, su consecuencia será indudablemente verdadera. En otras palabras, un enunciado hipotético predice consecuencias que se cumplen frente a ciertas condiciones.

Si el valor aportado por el usuario no es un número real mayor que 0.01, entonces un mensaje de error es generado por el sistema.

El enunciado anterior afirma que:

- Cuando algo (un valor)

- NO es el esperado (mayor a 0.01)
- Entonces, se lanza un evento (mensaje de error)

Pero de ningún modo afirma que si se lanza un evento (mensaje de error) es porque algo (un valor), NO ha sido el esperado (mayor a 0.01). Esto es debido a que las consecuencias son producto de una condición, que podría no ser la condición conocida (por ejemplo, un mensaje de error podría ser lanzado por cualquier motivo diferente al evaluado).

Diferencia entre enunciado hipotético y enunciado bicondicional

Mientras que el enunciado hipotético es de la forma «*si p entonces q*», un enunciado bicondicional, es el que lleva la forma «*p sí, y solo si q*».

Cuando se dice «*si p entonces q*», es equivalente a decir que «*si p es verdadera, q también lo es*». Pero no podría decirse que si *q* es verdadera, *p* también lo sea, porque se incurriría en la falacia de afirmar el consecuente.

Sin embargo, al decir «*p sí y solo si q*», equivale a decir que «*p es verdadera si q es verdadera*». Por lo tanto, «*si q (es verdadero) entonces p (es verdadero), y p (es verdadero) si q (es verdadero)*». El «*sí y solo sí*» indica bicondicionalidad.

Función de verdad

En los enunciados compuestos el valor de verdad está determinado por los valores de verdad de sus enunciados componentes. Cuando un enunciado

compuesto cuyo valor de verdad pueda determinarse absolutamente por el valor de verdad de sus enunciados componentes, se denomina *función de verdad*.

Función de verdad en enunciados conjuntivos

El valor de verdad de un enunciado conjunto está dado por el valor de verdad de todos sus enunciados componentes. Esto implica que todos sus componentes deben ser verdaderos para que el enunciado sea función de verdad. Para “A y B”, “A” debe ser verdadero y “B” debe ser verdadero para que la conjunción “A y B” también lo sea.

Función de verdad en enunciados disyuntivos exclusivos

El valor de verdad de un enunciado disyuntivo exclusivo estará dado por el valor de verdad de uno solo de sus enunciados componentes. Para “A o B (pero no ambos)”, “A” debería ser verdadero y “B” falso o “B” verdadero y “A” falso, para que “A o B (pero no ambos)” sea verdadero.

Función de verdad en enunciados disyuntivos inclusivos

El valor de verdad de un enunciado disyuntivo inclusivo estará dado por el valor de verdad de al menos uno de sus enunciados componentes, es decir, que bastará con que uno de sus componentes sea verdadero para que el enunciado lo sea. Sin embargo, también será función de verdad si todos sus componentes son verdaderos. Para “A o B (o ambos)”, será verdadero si “A” y “B” son verdaderos, si “A” es verdadero y “B” falso o si “B” es verdadero y “A” falso.

Función de verdad en enunciados hipotéticos

El valor de verdad de un enunciado hipotético estará dado por el valor de verdad del antecedente y también del consecuente. Esto significa que ningún enunciado hipotético con consecuente falso puede ser verdadero. Para “A entonces B”, será verdadero solo si “A” es verdadero y “B” es verdadero.

Silogismos

Un silogismo es un razonamiento deductivo en el que a partir de dos premisas, se infiere una conclusión.

Silogismo categórico

El *silogismo categórico* es aquel que emplea tres proposiciones categóricas, dos de ellas como premisa y la tercera, como conclusión.

Términos de un silogismo

A lo largo de un silogismo categórico, pueden encontrarse tres **términos**:

Término menor	<i>S</i>	Aparece en el término sujeto de la conclusión
Término mayor	<i>P</i>	Aparece en el término predicado de la conclusión
Término medio	<i>M</i>	Aparece en las dos premisas pero no en la conclusión

La conclusión, siempre contendrá en el término sujeto, al término menor (S), y en el término predicado, al término mayor (P), por lo que siempre tendrá la forma S – P:

$$\begin{array}{ccc} ? & - & ? \\ ? & - & ? \\ \textbf{S} & - & \textbf{P} \end{array}$$

El término mayor, debe aparecer siempre en la premisa mayor (como sujeto o predicado) e irá acompañado del término medio:

PRIMERA OPCIÓN:

$$\begin{array}{ccc} \textbf{P} & - & \textbf{M} \\ ? & - & ? \\ \textbf{S} & - & \textbf{P} \end{array}$$

SEGUNDA OPCIÓN:

$$\begin{array}{ccc} \textbf{M} & - & \textbf{P} \\ ? & - & ? \\ \textbf{S} & - & \textbf{P} \end{array}$$

Con el término menor sucederá lo mismo, solo que en vez de aparecer en la premisa mayor, aparecerá en la premisa menor:

PRIMERA OPCIÓN:

$$\begin{array}{ccc} ? & - & ? \\ \textbf{S} & - & \textbf{M} \\ \textbf{S} & - & \textbf{P} \end{array}$$

SEGUNDA OPCIÓN:

$$\begin{array}{ccc} ? & - & ? \\ \textbf{M} & - & \textbf{S} \\ \textbf{S} & - & \textbf{P} \end{array}$$

A las premisas que poseen los términos menor y mayor, se las denomina «*premisa menor*» y «*premisa mayor*», respectivamente.

Forma de los silogismos

La **forma** de un silogismo está determinada por el orden y cantidad en el que sus proposiciones aparecen. Así, un *silogismo categórico de forma típica* es

aquel posee tres proposiciones categóricas: premisa mayor, premisa menor y conclusión.

	M		P
(A)	<i>Todo lenguaje de programación es un lenguaje informático.</i>		
(B)		<i>PHP es un lenguaje de programación.</i>	
	S		M
(C)		<i>Luego, PHP es un lenguaje informático.</i>	
	S		P

(A) Premisa mayor | (B) Premisa menor | (C) Conclusión

La figura de un silogismo

El orden en el que se presentan los términos menor, mayor y medio, a lo largo del silogismo, determina su **figura**, pudiendo encontrarse un total de cuatro combinaciones:

Primera Figura	Segunda Figura	Tercera Figura	Cuarta Figura
M – P	P – M	M – P	P – M
S – M	S – M	M – S	M – S

El **modo** de un silogismo estará determinado por la forma de los enunciados y la disposición de sus términos, y será parte necesaria para determinar la **validez de un silogismo categórico**.

En el siguiente ejemplo, se presenta un silogismo con la forma **AEE-2**:

- (A) *Todo caballo_P es herbívoro_M.*
- (E) *Ningún mosquito_S es herbívoro_M.*
- (E) *Luego, ningún mosquito_S es caballo_P.*

Conocer la forma de un enunciado, permite determinar cómo se distribuyen sus términos.

Conocer la distribución de los términos y el orden en el que estos se presentan a lo largo de un silogismo, es lo que permitirá determinar la validez del mismo, mediante una serie de reglas.

Reglas de validación de un silogismo categórico

Hay seis reglas que un silogismo categórico de forma típica debe cumplir para ser válido. Se detalla cada una a continuación.

Regla N°1: el silogismo debe tener solo tres términos utilizados cada uno, en el mismo sentido a lo largo del argumento.

Si bien el uso de un mismo término en sentidos contradictorios puede apreciarse en cualquier ámbito, podría ser más habitual que se lo utilice con la finalidad de manipulación de la información, en el ámbito político.

Un ejemplo imaginario de esto podría darse entre dos partidos políticos opuestos. Uno liberal, y otro conservador, para los cuáles el concepto de «Nación segura», tendría significados opuestos.

En este ejemplo, el partido conservador considera que incrementar la seguridad de una Nación, consiste en permitir el uso civil de armas de fuego, mientras que el partido liberal, considera que incrementar la seguridad de una

Nación, consiste en lo opuesto, es decir, en prohibir el uso civil de armas de fuego.

Partiendo de este ejemplo, se podría elaborar el siguiente silogismo en apariencia válido, pero violando la primera regla:

Los liberales tienen por objetivo crear una nación más segura.

Una Nación más segura es aquella que tiene por objetivo permitir a todos sus ciudadanos civiles, portar armas de fuego.

Por lo tanto, los liberales tienen por objetivo permitir a todos sus ciudadanos civiles, portar armas de fuego.

Dado que la definición de «Nación más segura» es diferente para ambos partidos, el silogismo no es válido, pues el término medio está siendo utilizado en más de un sentido.

Regla N°2: el término medio debe estar distribuido en al menos una de las premisas.

En el siguiente ejemplo, el término medio se encuentra en el predicado de dos premisas universales afirmativas (forma A), por lo tanto, nunca se distribuye:

(A) Todos los números racionales son números reales.

(A) Todos los números irracionales son números reales.

(A) Por lo tanto, todos los números irracionales son números racionales.

Regla N°3: la conclusión no puede distribuir términos no distribuidos en las premisas.

En el siguiente ejemplo, el término menor (animales) no está distribuido en la premisa menor, ya que la misma tiene forma I, forma esta que no distribuye ninguno de sus términos:

- (A) *Todos los mamíferos son vertebrados.*
- (I) *Algunos animales son mamíferos.*
- (A) *Por lo tanto, todos los animales son vertebrados.*

Sin embargo, la conclusión tiene una forma universal afirmativa, la cual distribuye el término sujeto, es decir, el término menor, no distribuido en la premisa.

Regla N°4: no puede tener dos premisas negativas (al menos una debe ser afirmativa).

En el siguiente silogismo, la conclusión es falsa porque parte de dos premisas negativas:

- Ningún mamífero es invertebrado.*
- Ningún equinodermo es mamífero.*
- Por lo tanto, ningún equinodermo es invertebrado.*

Regla N°5: si una premisa es negativa la conclusión también lo es.

En el siguiente ejemplo se presenta una premisa negativa, por lo que su conclusión debería ser negativa:

- Ningún invertebrado es mamífero.*
- Todos los equinodermos son invertebrados.*
- Por lo tanto, todos los equinodermos son mamíferos.*

La conclusión válida del silogismo, debería haber sido «*Ningún equinodermo es mamífero*».

Regla N°6: una conclusión particular no puede basarse en dos premisas universales.

En el siguiente ejemplo, la conclusión es particular a partir de dos premisas universales. Debería ser universal para ser válida:

Todos los mamíferos son vertebrados.

Todos los caballos son mamíferos.

Por lo tanto, algunos caballos son vertebrados.

Silogismo disyuntivo

El *silogismo disyuntivo* es aquel que como premisa, tiene un enunciado disyuntivo.

En cuanto a su forma, mientras que la de un silogismo categórico sigue la sucesión: *Premisa mayor, Premisa menor, Conclusión*, la forma de un silogismo disyuntivo, se presenta de la siguiente manera:

Enunciado disyuntivo

Premisa categórica

Conclusión

En el silogismo disyuntivo, la negación de una de las disyunciones, afirma la otra:

Juan no ha venido al trabajo, o bien porque se siente mal, o bien porque tiene examen.

Juan no se siente mal.

Por lo tanto, Juan tiene examen.

Sin embargo, la afirmación de una de las disyuntivas, no niega la otra:

Juan no ha venido al trabajo, o bien porque se siente mal, o bien porque tiene examen.

Juan tiene examen.

Por lo tanto, Juan no se siente mal. (INCORRECTO)

El hecho de que Juan tuviese un examen, no lo exime de sentirse mal. Más allá de la cuestión empírica, la explicación lógica de este radica en que debido a que un enunciado disyuntivo puede ser verdadero tanto si una sola premisa es verdadera como si ambas lo son, la única forma de deducir cuál es verdadera, es sabiendo que una de ellas es falsa. Pero no puede deducirse cuál de ellas es falsa sabiendo cuál es verdadera, puesto que ambas podrían ser verdaderas, y el enunciado continuar siendo verdadero.

Silogismo hipotético

El *silogismo hipotético* es aquel en el cual su premisa es un enunciado condicional. Este tipo de silogismos conformarán una de las bases del método científico empírico.

La **forma** de un silogismo hipotético es similar a la de uno disyuntivo, solo que en vez de un enunciado disyuntivo tiene uno condicional:

Enunciado condicional

Premisa categórica

Conclusión

La premisa categórica *puede* ser:

- La afirmación del antecedente
- La negación del consecuente

La premisa categórica *no puede* ser:

- La afirmación del consecuente
- La negación del antecedente

En estos últimos dos casos, se estaría incurriendo en falacia.

Para **validar un silogismo** hipotético, existen solo dos modos posibles:

1. *Modus ponens* (modo afirmativo).
2. *Modus tollens* (modo negativo).

En el modo afirmativo, la premisa afirma el antecedente, y la conclusión el consecuente:

Si los datos están corruptos, entonces hay un fallo en la validación de los datos.

Los datos están corruptos.

Por lo tanto, hay un fallo en la validación de los datos.

En el modo negativo, la premisa niega el consecuente, y la conclusión el antecedente.

Si los datos están corruptos, entonces hay un fallo en la validación de los datos.

No hay un fallo en la validación de los datos.

Por lo tanto, los datos no están corruptos.

Falacias

Las falacias se emplean a menudo con el fin de persuadir psicológicamente. Si bien su estudio no es necesario para analizar la validez de un argumento, conocer las formas habituales de persuasión podría ofrecer una alternativa para detectar formas argumentales no válidas.

No existe una clasificación universalmente aceptada de falacias³ pero sí pueden ser agrupadas en dos grandes grupos:

1. Falacias formales
2. Falacias no formales

Las falacias formales son las que guardan relación con un error de forma, mientras que las no formales, conservan una forma válida pero con conclusiones falsas, bien sea por falta de atinencia lógica, o bien, por ambigüedad.

³ Irving M. Copi, Introducción a la lógica (Eudeba, 2014). Pág. 81.

Falacias formales

La falacia formal es toda aquella que viole al menos una de las reglas de cualquiera de los tipos de silogismos.

En los *silogismos hipotéticos*, son falacias:

- Falacia de negación del antecedente: se da cuando la premisa de un silogismo hipotético niega el antecedente del enunciado para negar el consecuente en conclusión. Se la considera una falacia, ya que la premisa de un silogismo hipotético solo tiene dos formas posibles, afirmar el antecedente (que deriva en la afirmación del consecuente como conclusión), o negar el consecuente (que deriva en la negación del antecedente como conclusión).
- Falacia de afirmación del consecuente: se da cuando la premisa de un silogismo hipotético afirma el consecuente del enunciado y afirma el antecedente como conclusión. Al igual que en el caso anterior, se considera una falacia ya que los únicos modos posibles, son el modo afirmativo (afirmar el antecedente) o el modo negativo (negar el consecuente).

En los *silogismos disyuntivos*:

- Falacia de afirmación del silogismo disyuntivo: esta falacia se da cuando en un silogismo disyuntivo, la premisa afirma uno de los componentes y deduce la negación del siguiente. Se considera una falacia puesto que el único modo posible del silogismo disyuntivo, es

que la premisa niegue uno de los componentes del enunciado para afirmar el otro en la conclusión. Esta falacia se comprueba mediante una tabla de verdad.

Y, en los *silogismos categóricos*:

- Incurren en falacias non sequitur, todos aquellos argumentos formalmente inválidos y/o que violen una o más reglas silogísticas. Un caso habitual es el de la falacia del término medio no distribuido.

Falacias no formales

Como se comentó anteriormente, no existe una clasificación universalmente aceptada. Por lo tanto, a continuación se listarán algunas de las falacias de atinencia y ambigüedad consideradas por autores como Irving Copi como las falacias no formales más habituales. Se omiten algunas falacias de forma intencionada.

Entre las **falacias de atinencia** lógica, es posible encontrar:

- Argumentación ad baculum: se da cuando quien argumenta, en vez de ofrecer pruebas sobre aquello que afirma, apela a la intimidación para lograr la aceptación. La intimidación puede entenderse como cualquier acto que infunda miedo.
- Argumentación ad hominem: este tipo de falacia puede darse de dos formas:

- *Forma ofensiva*: se da cuando en vez de ofrecer evidencias de aquello que se afirma, se ataca a la persona que ofrece un argumento opuesto.
 - *Forma circunstancial*: se da cuando en vez de ofrecer evidencias de aquello que se afirma, se apela a creencias circunstanciales, no probatorias. Este tipo de argumentación *ad hominem* suele estar amparada en cuestiones de índole dogmática o ideológica.
- Argumentación ad ignorantiam: se da cuando se afirma que algo es verdadero porque no existen pruebas de lo contrario. «*La ausencia de pruebas*» no es una prueba.
 - Argumentación ad populum: se da cuando en vez de ofrecer pruebas de aquello que se afirma, se apela a ganar la mayor cantidad de adeptos posibles por medio de la simpatía con lo que se afirma, procurando generar una reacción emocional.
 - Argumentación ad verecundiam: se da cuando en vez de ofrecer pruebas de aquello que se afirma, se apela a la autoridad de quien argumenta. Suele ser el caso contrario a la argumentación *ad hominem* en su forma ofensiva.
 - Petitio principii (petición de principio): se da al utilizar como premisa a la propia conclusión, aunque no directamente utilizando las mismas palabras.

- Falacia de la pregunta compleja: esta falacia es la que en una pregunta da por sentado que previamente se ha respondido «si» a una pregunta no realizada, partiendo así de una proposición falsa. En la pregunta «¿por qué “A”?» se da por sentado que «A» existe y es verdadera, por lo que se presume que se ha respondido «sí» a la pregunta «¿Es “A” verdadera?». Este tipo de preguntas presuponen verdaderas, afirmaciones de las cuáles en realidad, no se ha establecido previamente su veracidad o falsedad.
- Ignoratio elenchi: se da cuando para justificar una conclusión, se emplean premisas destinadas a argumentar otra, y por lo tanto, carece de atinencia.

Las **falacias de ambigüedad** se dan por polisemia, anfibología o énfasis.

La *polisemia* se produce cuando una misma expresión lingüística tiene más de un significado, como sucede en “banco”, “estado”, o “cuña”, mientras que la “anfibología”, consiste en expresar una misma frase de forma tal que pueda tener una doble interpretación (o doble sentido). Lo anterior da origen a dos tipos de falacias:

- Falacia del equívoco: se da cuando se emplea un mismo término en sentidos polisémicos.
- Falacia de anfibología: se da cuando se emplea una premisa en su interpretación verdadera, y la conclusión conduce a su interpretación falsa.

Las falacias de énfasis, están directamente relacionadas con la entonación y las pausas de las que se haga uso. Por ejemplo, en «no, puedo» y «no puedo», el uso de una pausa entre el «no» y el «puedo» o la ausencia de dicha pausa, cambia por completo el sentido de la frase. La falacia de énfasis se comete cuando se utiliza la entonación para generar ambigüedad en aquello que se afirma o arguye.

Lógica matemática

La lógica matemática, conocida también como *lógica simbólica*, es aquella que emplea el uso de un lenguaje de símbolos abstracto —denominado *lenguaje formal*—, como base de un sistema.

En la lógica matemática que será abarcada a continuación, los signos especificados se encuentran ya definidos. El estudio particular de estos corresponde a la *semiótica* pero no se profundizará aquí en su estudio.

A continuación se abarcará el sistema simbólico mínimo sin el cuál, no podrían entenderse los enunciados lógicos básicos.

Elementos básicos de la lógica simbólica

Variables sentenciales

Para la sustitución de enunciados se utilizan las letras medias del alfabeto, “*p*”, “*q*”, “*r*”, “*s*”, tal que un enunciado hipotético podría expresarse en la forma:

Si p entonces q

Conjunciones

Para unir de forma conjuntiva puede emplearse el punto medio · o el símbolo \wedge . Tanto el enunciado $p \cdot q$ como el enunciado $p \wedge q$, son conjuntivos.

Negación y agrupación

Para la negación se pueden utilizar los símbolos “~” y “¬”.

Negación de una variable: $\sim q$ o $\neg q$

Para la negación de un enunciado, se requiere agrupar el enunciado. La agrupación se realiza con paréntesis:

Negación de la conjunción p y q : $\sim(p \wedge q)$ o $\neg(p \wedge q)$

Disyunciones

Para expresar enunciados disyuntivos se utiliza la letra “v”. $p \vee q$

Para la disyunción exclusiva será necesario expresar la exclusión de forma literal, es decir, que para expresar que “ p o q pero no ambas” se requerirá el enunciado compuesto “ p o q , y no p y q ”: $(p \vee q) \cdot \sim(p \cdot q)$

En programación, mientras que $p \vee q$ es representado mayormente por la instrucción (`p or q`), algunos lenguajes de alto nivel, resuelven la expresión $(p \vee q) \cdot \sim(p \cdot q)$ mediante un *token* específico tal como (`p xor q`), evitando así la instrucción completa (`p or q`) and `not` (`p and q`). Sin embargo, a pesar de ello, internamente siempre se procesará la versión formal (`p or q`) and `not` (`p and q`).

Condicionales

Para enunciados condicionales se utilizan los símbolos herradura \supset o la flecha doble \Rightarrow . “*Si p entonces q*” podría expresarse tanto en la forma $p \supset q$ como $p \Rightarrow q$.

Validación de enunciados lógicos y argumentos

Existen diversos mecanismos para probar la validez o invalidez de un argumento. Algunos de estos mecanismos se describen a continuación, solo a modo ilustrativo.

Emulación de la forma argumental con conclusión evidentemente falsa: El mecanismo más simple es demostrar la invalidez construyendo un argumento con una forma idéntica del que se desea validar, con premisas verdaderas pero con una conclusión evidentemente falsa.

Sustitución de variables sentenciales: La sustitución de enunciados por letras, suprime la semántica del argumento permitiendo abstraerse solo a su forma.

Tablas de verdad: Para evaluar el valor de verdad de un enunciado compuesto se pueden utilizar tablas de verdad. Estas se confeccionan sobre la base de un encabezado que contiene los enunciados componentes, y una fila por cada combinación de opciones de verdad posible. Para las opciones de verdad posibles (verdadero o falso) se emplean en castellano, las letras **V** y **F** respectivamente, y **T** y **F** en inglés, correspondientes a los términos *True* (verdadero) y *False* (falso).

Para el enunciado $p \cdot q$, la tabla de verdad quería representada de la siguiente forma:

p	q	$p \cdot q$
V	V	V
.....		
V	F	F
.....		
F	V	F
.....		
F	F	F

Para armar una tabla de verdad, se descompone un enunciado obteniendo sus componentes más pequeños, hasta llegar a los enunciados compuestos más

complejos. En la tabla anterior pueden verse tres columnas, p , q y $p \cdot q$, debido a que los componentes más atómicos de $p \cdot q$ son p , q .

Luego, en cada fila, se van poniendo los resultados de la evaluación por separado. Primero, el de los componentes más pequeños y luego, el de los más grandes. Por ejemplo, para la validación de una disyunción exclusiva $(p \vee q) \cdot \sim(p \cdot q)$, la tabla de verdad quedaría representada como la siguiente:

p	q	$p \vee q$	$p \cdot q$	$\sim(p \cdot q)$	$(p \vee q) \cdot \sim(p \cdot q)$
V	V	V	V	F	F
V	F	V	F	V	V
F	V	V	F	V	V
F	F	F	F	V	F

INTRODUCCIÓN AL MÉTODO CIENTÍFICO

Eugenia Bahit

Fundamentos epistemológicos

Se comentó anteriormente que la epistemología es una disciplina metacientífica que tiene por objeto el estudio del conocimiento científico. Con el fin de establecer qué es aquello que se entiende por conocimiento científico válido, la epistemología estudia, entre otros aspectos, el lenguaje. La comprensión de las diferentes formas del discurso es la que permite definir las características que el lenguaje debe tener para ser considerado científico. En este sentido, se hace a continuación un breve resumen de los aspectos fundamentales estudiados por la epistemología.

Diferentes tipos de discurso

El lenguaje puede ser empleado con diferentes propósitos, los cuales no necesariamente implicarán propósitos puros (o aislados). Estos propósitos son:

- *Informativo*: aquel que busca comunicar información (tanto verdadera como falsa; y correcta como incorrecta).
- *Expresivo*: aquel que busca expresar emociones.
- *Directivo*: aquel que busca dar origen a una acción concreta o impedirla.

Puesto que el discurso informativo es el único que se basa en la formulación de proposiciones tanto afirmativas como negativas, es el único que puede ser considerado verdadero o falso.

La neutralidad emotiva en el lenguaje científico

El impacto emotivo en el discurso informativo puede representar un escudo a la hora de evaluar objetivamente la validez y/o veracidad del conocimiento. Epistemológicamente, se considera que el lenguaje científico debe ser tan neutro, en términos emotivos, como sea posible.

La neutralidad emotiva suele medirse en perspectiva al impacto emocional que un vocablo, expresión o giro idiomático genera en quienes reciben el mensaje. Esto implica que no necesariamente los adjetivos o adverbios posean carga emotiva. Un sustantivo podría tener carga emotiva si denota o se relaciona de alguna manera con ciertos estereotipos. Esto conlleva a que en ocasiones se prefiera el uso de la definición de un término que el término específico en sí mismo, razón por la cual, la economía lingüística tiende a ser opuesta a los requerimientos del lenguaje científico.

Un caso común de impacto emotivo se da con el uso de adverbios de cantidad tales como «muchos», «pocos» y expresiones tales como «una gran mayoría», «casi todos», «prácticamente ninguno», entre otros. Mientras que expresiones tales como «muchos» o «una gran mayoría» impactan de forma totalizadora, sus expresiones contrarias, pueden ser interpretadas como nulidad de casos. De esta forma, frases como *«la mayoría del software desarrollado hasta el momento es inseguro»* puede impactar de forma alarmante y ser interpretada como *«el 100% del software es inseguro y jamás existirá la posibilidad de desarrollar software seguro»*.

En el idioma español, el desdoblamiento del género marcado, es otra de las condiciones que hoy día, impacta emocionalmente en el discurso. Un ejemplo de ello puede estudiarse con la siguiente frase pronunciada en el discurso de asunción presidencial de Alberto Fernández en la Rep. Argentina, el día 10 de diciembre de 2019:

«Deseo dirigirme muy personalmente a cada una y a cada uno de esos argentinos que habitan esta Patria.»

La frase precedente podría generar diversas emociones, tanto negativas como positivas. Y esto podría deberse a un uso incorrecto del desdoblamiento. Una persona acorde a las ideas del emisor, podría sentirse a gusto por el uso de dicho desdoblamiento, y una persona opositora, podría sentirse a disgusto por el mismo motivo. Sin embargo, personas de opinión neutral e incluso personas de ideas tanto acordes como opuestas, podrían sentirse tanto a gusto como a disgusto con el mencionado desdoblamiento.

El desdoblamiento es una característica del lenguaje que se emplea cuando el uso del género marcado (el género masculino para el idioma español) puede dejar dudas con respecto al objeto u objetos apuntados (objeto directo). En este sentido, el desdoblamiento correcto se realiza, justamente, desdoblando el objeto que genera duda, y no el adverbio (del cual se podría prescindir):

«(...) *cada argentina y argentino* (...)» o «(...) *cada argentino y argentina* (...)»

No obstante, el desdoblamiento podría continuar apelando a emociones. De allí que si la frase anterior tuviese que emplearse en un contexto científico, o por lo menos, emocionalmente neutro, probablemente lo sería más si utilizase el término persona:

«(...) *a cada una de las personas* (...) ».

Sin embargo, fuera del contexto de los discursos intencionalmente emotivos, la frase completa, en el sentido más neutral posible, podría ser como la siguiente:

«*Deseo dirigirme muy personalmente a cada una y a cada uno de esos argentinos persona de nacionalidad argentina que habitan esta Patria este país*»

Principios de la definición

Uno de los objetivos de la definición es evitar la ambigüedad. Cuando se la emplea además como sustituto de un término, su objetivo es evitar la carga emotiva.

La definición se compone de dos elementos: el ***definiendum*** (término que se define) y el ***definiens*** (conjunto de términos empleados para explicar el *definiendum*).

Existen cinco tipos de definiciones que pueden encontrarse:

1. **Definición *estipulativa*:** aquella que se da por primera vez al introducir un término nuevo. Es frecuente en el ámbito científico y representa la única forma válida de economía lingüística.
2. **Definición *lexicográfica*:** se emplea en términos preexistentes para esclarecer a qué aspectos del mismo se hará referencia. En el ámbito científico tiende exclusivamente a evitar la ambigüedad que podría producirse al emplear un mismo término en más de un sentido, generando así un error de razonamiento (falacia).
3. **Definición *teórica*:** es aquella cuyo objetivo es el de proponer la aceptación de una teoría. Este tipo de definiciones se encuentran sujetas a cambios a medida que el conocimiento científico va avanzando y se amplía la información disponible sobre dicha teoría.

Otros dos tipos de definición son la ***definición persuasiva*** cuyo único objetivo es provocar un determinado tipo de reacción (no se considera dentro de los tipos de definición que el lenguaje científico emplea), y la ***definición aclaratoria***, que busca establecer casos límites y excepciones (utilizada en mayor medida en el ámbito jurídico).

A fin de realizar una correcta definición, ya sea esta estipulativa, lexicográfica o teórica, la definición se rige por una serie de principios que se enumeran a continuación:

1. Debe describir las propiedades esenciales que otorguen identidad al término o especie definida.
2. No puede ser circular (se refiere a que el *definiendum* no puede aparecer en el mismo sentido lexicográfico como parte del *definiens*).
3. Debe ser justa en el sentido de que el *definiens* no puede tener mayor ni menor alcance que el *definiendum*.
4. Debe poder interpretarse en un sentido literal y concreto, no figurado o ambiguo.
5. Debe afirmar lo que el término es, y no lo que el término no es.

El método científico

El método científico es el método utilizado por la ciencia para la producción de conocimiento válido. El método difiere entre el empleado en las ciencias fácticas (empíricas) y las ciencias formales (lógica, matemática).

Las ciencias formales emplean un *método axiomático* mientras que las ciencias fácticas, un *método hipotético-deductivo*. Ambos métodos son descritos a continuación.

El método axiomático de las ciencias formales

El *método axiomático* provee los principios necesarios para la producción de sistemas axiomáticos.

Los *sistemas axiomáticos* son conjuntos de razonamientos deductivos complejos a partir de los cuales se obtienen teoremas.

Los *teoremas* son afirmaciones que se demuestran formalmente mediante el uso de la lógica, partiendo de axiomas.

Los *axiomas* son verdades indemostrables que partiendo de términos primitivos, se aceptan como punto de partida para los sistemas axiomáticos.

Los *términos primitivos* son un grupo de expresiones que se presumen conocidas, por lo que su definición no es necesaria.

Propiedades

Un sistema axiomático debe satisfacer tres propiedades:

1. Consistencia
2. Independencia
3. Completitud

Un sistema consistente, es aquel que no permite que se deduzcan enunciados contrarios o contradictorios a partir del mismo conjunto de axiomas.

La independencia se refiere exclusivamente al conjunto de axiomas. Un axioma es independiente cuando no puede deducirse a partir de otros. Un axioma que puede deducirse, en realidad es un teorema.

Un sistema axiomático es completo cuando dado un enunciado y su negación, al menos uno de ellos puede ser demostrado a partir de los axiomas del sistema. Será incompleto cuando ninguno de los dos sea demostrable por los axiomas.

El método hipotético-deductivo de las ciencias fácticas

El *método hipotético-deductivo* es un método basado en hipótesis para formular teorías científicas.

Una *hipótesis* es un enunciado con carácter condicional, aún no verificado empíricamente ni refutado.

Una *teoría científica* es un conjunto de enunciados básicos a partir de los cuáles se formula una jerarquía de hipótesis interrelacionadas, donde a partir de una hipótesis general, es posible deducir otras hipótesis particulares.

Tipos de enunciados científicos

De acuerdo a la forma en la que los enunciados se fundamentan y verifican, existen tres niveles de enunciados científicos:

1. El primer nivel, corresponde a los *enunciados empíricos básicos*.

2. El segundo nivel, a las *generalizaciones y leyes empíricas*.

3. Y el tercer nivel, a los *enunciados teóricos*.

La diferencia entre estos tres tipos de enunciados, se basa en gran parte, en los términos que cada uno de ellos emplea, pudiéndose encontrar de dos tipos:

1. Los *términos empíricos*, que designan objetos observables.
2. Los *términos teóricos*, que designan construcciones teóricas no observables.

Enunciados empíricos básicos

Los enunciados empíricos básicos son enunciados de observación directa sobre ciertas características específicas de las entidades observables. Estos enunciados tienen dos propiedades que los diferencian de otros:

- Solo utilizan términos empíricos
- Hacen referencia a un número limitado y accesible de entidades observables

VERIFICABILIDAD: Un enunciado empírico básico es verdadero solo si aquello que afirma es observable en las entidades observadas.

Generalizaciones y leyes empíricas

Al igual que los enunciados empíricos básicos, se trata de enunciados de observación directa sobre características específicas de entidades observables, que también hace uso solo de términos empíricos. Se diferencian de los

anteriores, en que sus afirmaciones hacen referencia al total de una clase y no, solo a un número limitado de entidades de la misma.

Estos enunciados, a su vez, pueden diferenciarse en tres subtipos:

1. *Enunciados generales universales*: aquello que afirman hace referencia al total de entidades observables de una misma clase, sin excepción.
2. *Enunciados generales existenciales*: aquello que afirman está presente en por lo menos un ejemplar observable de toda la clase.
3. *Enunciados generales estadísticos*: aquello que afirman es observable con determinada frecuencia o probabilidad en las entidades de una clase.

VERIFICABILIDAD: Un enunciado general no es verificable puesto que:

1. En el caso de ser universal, se deberían observar entidades infinitas.
2. En caso de ser existencial o estadístico, la afirmación es sobre un número limitado de entidades observables, por lo que siempre podría aparecer una entidad en la que el enunciado no fuese verdadero.

Esto es lo que da a la ciencia fáctica, enunciados de carácter provisional.

Enunciados teóricos

Se trata de enunciados que, pudiendo ser particulares o universales, emplean al menos un término teórico.

VERIFICABILIDAD: Al tratarse de enunciados que hacen referencia a por lo menos un término teórico de entidades no observables, su verificación dependerá de la predictibilidad observable que a partir del enunciado pueda obtenerse deductivamente. Esto significa que a partir de un enunciado teórico, en combinación con enunciados empíricos, se debe poder obtener deductivamente, una predicción observable para que el enunciado sea verdadero.

Validez de una teoría científica y predictibilidad

Considerando que una teoría científica se compone de un conjunto de hipótesis fundadas en enunciados básicos, una teoría será válida en la medida que la totalidad de sus enunciados básicos no sea refutada. Esto implica que:

- Todo enunciado básico debe poder ser refutado por medio de la observación.
- La verificación de un enunciado debe hacerse sobre todas las entidades observadas sin excepción. Esto implica que si se observan 100 entidades de un total infinito de entidades observables de la clase, el enunciado deber verificarce en las 100 entidades para considerarse verdadero.
- Mientras que para verificar un enunciado se requiere hacerlo sobre el total de las entidades observadas de la clase, basta una sola entidad para falsarlo. Esto implica que si de las 100 entidades observadas, en

una sola de ellas no se ratifica la veracidad del enunciado, sería suficiente para decir que el enunciado es falso.

- Basta con que un solo enunciado sea falso, para que toda la teoría se considere inválida. Pues de la hipótesis principal basada en dicho enunciado, se desprenden el resto de hipótesis de la teoría.

Una hipótesis se basa en enunciados que afirman algo observable sobre entidades conocidas. En este sentido, a partir de una hipótesis se deben poder deducir experiencias previas y predecir, con los mismos resultados, las experiencias futuras, para que la teoría continúe siendo válida.

LAS CIENCIAS INFORMÁTICAS Y SU RELACIÓN CON LA FÍSICA Y LA MATEMÁTICA

Tanto la matemática como la física son la base de las Ciencias Informáticas. Por un lado, la física sirve de base para el desarrollo de los componentes electrónicos desde los que se conciben los equipos computacionales, y las

tecnologías basadas en el electromagnetismo. Por otro lado, las matemáticas sirven de base a todos los procesos computacionales, tanto teóricos como aplicados.

Un detalle más exhaustivo sobre las **teorías y procesos físicos sobre los que se concibe la computación**, puede encontrarse en el artículo de la física *Fernanda Hernández González* en la segunda parte de este libro.

La informática teórica es la disciplina científica que actúa como nexo entre las matemáticas y las ciencias informáticas. En este sentido, cualquier rama de las matemáticas es necesaria como sustento para cualquier teoría informática.

En la Informática Aplicada, las matemáticas son empleadas de forma continua en todas sus ramas. Es en la programación donde el uso de las matemáticas se hace más evidente. La aplicación de la lógica matemática, la aritmética, el álgebra, la probabilidad, la estadística, el análisis matemático, entre otras, son conocimientos que constantemente deben aplicarse a la programación.

Bibliografía de la primera parte

- [0] A. Gianella, «Capítulo 7» en *Lógica Simbólica y Elementos de Metodología de la Ciencia*. Ediciones Cooperativas: Argentina, 2002. pp. 201-215.
- [1] I. Copi, *Introducción a la Lógica*. Eudeba: Argentina, 2014.
- [2] D. Martínez Zorrilla, «Las falacias argumentativas» en *Una breve introducción a la argumentación* [online]. Universitat Oberta de Catalunya:

España. Disponible en

[http://openaccess.uoc.edu/webapps/o2/bitstream/10609/246/5/T
%C3%A9cnicas%20de%20expresi%C3%B3n%2C%20argumentaci%C3%B3n%20y%20negociaci%C3%B3n_M%C3%B3dulo1_Una%20breve%20introducci%C3%B3n%20a%20la%20argumentaci%C3%B3n.pdf](http://openaccess.uoc.edu/webapps/o2/bitstream/10609/246/5/T%C3%A9cnicas%20de%20expresi%C3%B3n%2C%20argumentaci%C3%B3n%20y%20negociaci%C3%B3n_M%C3%B3dulo1_Una%20breve%20introducci%C3%B3n%20a%20la%20argumentaci%C3%B3n.pdf)

[3] R. Gaeta, N. Robles, «Introducción» en *Nociones de epistemología*. Eudeba: Argentina, 1990. pp. 9-15.

FUNDAMENTOS TEÓRICOS

PARTE II. FUNDAMENTOS TEÓRICOS

FUNDAMENTOS FÍSICOS DE LAS CIENCIAS COMPUTACIONALES

Fernanda Hernández González

Definiciones previas

Física

La Física es el estudio de las propiedades y dinámica de la materia, representadas a través de variables en ecuaciones matemáticas. Estas ecuaciones constituyen un modelo de la naturaleza y sus procesos, con el cual se pueden obtener predicciones. La fiabilidad del modelo se verificará con la experimentación, de manera que una teoría física siempre debe pasar a través de un experimento que confirme lo obtenido a través del modelo.

Computación

Se entiende por computación al proceso de obtener unos valores de salida determinados (*output*) a partir de unos valores de entrada (*input*) esperados. Esto se logra a partir de procesos tales como funciones o algoritmos, que tomando los valores de entrada, a través de una serie de pasos y/o

evaluaciones, obtienen los valores de salida. Las máquinas que realizan este proceso son las computadoras.

Objetivo

El objetivo de este documento es describir algunas de las teorías y procesos físicos que permiten que una computadora pueda realizar sus funciones, mostrando así la participación de los avances científicos en el desarrollo tecnológico, en particular, la computación.

Introducción

Una de las razones que motivan el estudio de la Física es que el ser humano pueda conocer las características y propiedades de las fuerzas de la naturaleza, y hacer uso de ellas en beneficio de la humanidad.

Hasta ahora se consideran cuatro fuerzas fundamentales. En orden de intensidad relativa, estas son:

1. Fuerza nuclear fuerte.
2. Fuerza electromagnética.
3. Fuerza nuclear débil.
4. Fuerza gravitacional.

Siendo la fuerza electromagnética, la principal incumbencia en lo que a computación respecta.

La *fuerza electromagnética* es una fuerza que se presenta entre partículas cargadas, estáticas y en movimiento, de manera que afecta a protones, electrones, y piones, entre otras más, y es una combinación entre la fuerza magnética y eléctrica. Los aparatos electrónicos trabajan con energía eléctrica aprovechando esta fuerza.

A continuación se hace un desglose de las principales ramas de la Física que contribuyen como fundamento de las Ciencias de la Computación. Si bien hay divisiones entre los apartados, será frecuente que los temas tengan relación entre sí.

Teorías y procesos físicos sobre los que se concibe la computación

1 Electrónica

1.1 Circuitos integrados

1.2 Sensores y señales

1.3 Transistores y diodos

1.4 Teoría de semiconductores

2 Electromagnetismo

2.1 Tecnología inalámbrica y ondas electromagnéticas

2.2 Restricciones a la transmisión de señales

Electrónica

La electrónica es una disciplina que se encarga del estudio de los sistemas eléctricos, desde el punto de vista del electrón (Alcalde San Miguel, 2009), de manera que se busca su manipulación para que los electrones sigan las trayectorias más convenientes. Esto se logra con dispositivos como circuitos, transistores, diodos, resistencias, y condensadores, entre otros.

La electrónica se subdivide en dos tipos: la electrónica analógica y la electrónica digital.

La *electrónica analógica* es la que se encarga de analizar señales continuas como lo que se percibe con los sentidos, mientras que la *electrónica digital* se ocupa de las señales discretas que pueden tomar sólo dos estados opuestos: positivo o negativo; sí o no; abierto o cerrado. Estos estados generalmente son codificados mediante bits (0 y 1).

Circuitos integrados (CI)

Un circuito integrado es una pequeña pieza de silicona, en la que en el área de una moneda es posible reunir miles de millones de transistores, haciendo posible, entre otras cosas, la tecnología móvil. Estos circuitos representan un avance miniaturizado de los circuitos tradicionales, en los que los electrones viajan a través de cables y sus terminales se encuentran soldadas directamente a los dispositivos electrónicos, formando así complejos circuitos mediante módulos independientes.

Sensores y señales

La materia que rodea al ser humano tiene ciertas propiedades, las cuales dejan impresiones sobre células cuyo objetivo es recibir estímulos del exterior, transmitirlos para que el cerebro los interprete y actúe acorde a la situación. Con un mecanismo análogo, en una computadora hay circuitos y sensores que aprovechan las propiedades físicas de la materia para poder recibir información de su ambiente y realizar una acción. Una parte de este proceso es determinar qué es lo que recibe la computadora, pues debe poder entenderlo.

Las computadoras pueden procesar y guardar información de manera discreta o digital usando circuitos internos. La unidad de información es el *bit*, base del sistema binario.

Dado que hasta el momento, una computadora no entiende el concepto de número ni puede leerlo tal como lo hace el ser humano, para que realice cálculos se le debe proveer una señal que interprete como *bit*. Esto se puede lograr mediante pulsos de voltaje (o corriente) enviados a través de cables metálicos o pistas de circuitos.

La *relación entre voltaje y corriente* está expresada en la Ley de Ohm $V=R \cdot I$, donde V es la diferencia de voltaje, R es la resistencia del material, e I es la corriente, determinada por la cantidad de carga por unidad de tiempo (algunos materiales o circuitos no cumplen esta relación exacta, pero para voltajes pequeños es una buena aproximación).

La *resistencia* es una propiedad vinculada directamente al material y mide la oposición de este al paso de la corriente. Este valor puede variar de acuerdo a la temperatura, tamaño y forma del cable o material por el que transita la corriente, de manera que un cable largo tiene una mayor resistencia y un cable grueso, una resistencia menor que uno de delgado. Lo que se necesita es que la resistencia de los materiales disminuya, pues como se verá más adelante, existe una pérdida de energía debido a ésta.

Teoría de semiconductores

Algunas de las propiedades de los materiales a gran escala, surgen a partir de propiedades a pequeña escala. Una de ellas es la *conductividad*. Si bien hay materiales que por su naturaleza son conductores (pues tienen baja resistencia al paso de corriente, por lo que los electrones pueden pasar libremente a través del material), hay otros que son aislantes.

Sin embargo, no es una clasificación dicótoma, pues hay otros materiales que dependiendo de la temperatura a la que se encuentren, la corriente que los atraviese o el tipo de radiación electromagnética (luz) que incida sobre él, pueden funcionar como aislantes o conductores. A este tipo de materiales se los llama *semiconductores*, y tienen diversas aplicaciones desde el punto de vista electrónico, pues entre otras cosas, pueden funcionar como un interruptor, donde bajo ciertas condiciones físicas, dejen pasar corriente y cuando su estado se modifica, impedir su paso, como se explicará párrafos adelante. Con esto se pueden construir compuertas lógicas, que son la base para el desarrollo computacional.

Para explicar este fenómeno es necesario abarcar la *Mecánica Cuántica*, ya que para encontrar las energías permitidas para los electrones, hay que resolver la ecuación de *Schrödinger*⁴, es decir encontrar una función de onda que cumpla dicha ecuación. Se llega a esta solución utilizando el modelo *Kronig-Penney*⁵.

Lo primero es modelar a los átomos en un material como una sucesión periódica de pequeñas barreras, es decir repitiendo el valor del potencial en un intervalo fijo. El estado del electrón está determinado por dichos valores, y por las condiciones de frontera, esto es el valor que la función de onda toma en los extremos espaciales del material.

Una vez que se resuelve la ecuación bajo estas condiciones, se obtiene el espectro de energías, constituido por los valores de la energía que pueden tomar los electrones. Esto no tiene un análogo clásico pues en mecánica clásica las energías toman valores continuos.

Lo anterior genera ciertas bandas de energías permitidas, intercaladas con bandas de energía prohibida, y en particular para los materiales semiconductores, es posible, con aumento de temperatura o irradiación

- 4 NdE: Para profundizar en el tema de la «*Ecuación de Schrödinger*» se sugiere visitar el sitio Web del proyecto «*Hyperphysics*» (en español), del Departamento de Física y Astronomía de la Universidad del Estado de Georgia: <http://hyperphysics.phy-astr.gsu.edu/hbasees/quantum/schrcn.html>. También puede verse la sección de «*Mecánica cuántica*» del «Curso de Física en Internet» de la Universidad del País Vasco: <http://www.sc.ehu.es/sbweb/fisica3/cuantica/portada.html>
- 5 NdE: Para comprender el «*Modelo de Kronig-Penney*» se sugiere visitar la página sobre «*Potencial Periódico*» del «Curso de Física en Internet» de la Universidad del País Vasco: <http://www.sc.ehu.es/sbweb/fisica/cuantica/lineal/lineal.htm>

electromagnética de una frecuencia concreta, hacer que los electrones ganen energía y pasen a bandas más energéticas donde la conducción sea posible. Si se baja la temperatura o se deja de irradiar al material, entonces los electrones pierden excitación y vuelven a un estado menos energético, por lo que el material ya no es buen conductor.

Para fabricar este tipo de materiales se puede *dopar*⁶ con impurezas un material, es decir colocar en un material puro, un átomo con un electrón más que el del átomo del propio material de manera que tenga más electrones excitados (semiconductor *tipo n*), o en cambio, si se reemplaza un átomo con otro que tenga un electrón menos, entonces se dice que tiene vacancia de electrones (semiconductores *tipo p*). Este procedimiento mantiene al material neutro, pues no se está afectando el número de cargas positivas con respecto a las negativas.

Transistores y diodos

Si se unen dos materiales, uno *tipo p* y uno *tipo n*, en la interfaz entre estos dos materiales hay una acumulación de electrones del lado *p* y una acumulación de vacancias de electrones del lado del material tipo *n* por lo que, al transitar una corriente en un sentido tendrá una resistencia y si se invierte el sentido de la corriente se tendrá una resistencia con un orden

- 6 NdE: En electrónica, la acción de introducir impurezas en un material semiconductor, con el fin de modificar su comportamiento, se conoce como «*dopado de un semiconductor*». Fuente: «Curso de Electrónica Básica» de la Universidad del País Vasco:
http://www.sc.ehu.es/sbweb/electronica/elec_basica/tema2/TEMA2.htm
- 4 Siendo estrictos, sí toman valores discretos pero la separación de niveles de energía en el caso clásico es tan pequeño que para la escala de medición son valores continuos.

diferente, por lo que no pasará la corriente. Una de las aplicaciones de estos dispositivos, es el interruptor electrónico (*switch*) y la rectificación de corriente.

Un *transistor* consiste en la unión de tres de estos materiales. Los transistores más simples son los bipolares y existen los tipos *n-p-n* o *p-n-p*. Estos componentes pueden modificar corrientes, por lo que en la electrónica tienen aplicaciones como amplificadores, osciladores y también como interruptores electrónicos.

El uso de transistores bipolares requiere grandes cantidades de energía para su funcionamiento, por lo cual, como se verá más adelante, genera que el circuito se caliente afectando su funcionamiento. Por este motivo, actualmente se utilizan transistores tipo FET (por sus siglas en inglés, *Field-effect Transistor*) los cuales funcionan a partir de campos eléctricos.

Configuraciones particulares de transistores FET logran reproducir compuertas lógicas NOT, NOR y NAND. El posterior desarrollo de la teoría de semiconductores hace posible la miniaturización de componentes electrónicos, permitiendo, por ejemplo, caminar con un computador en el bolsillo.

Electromagnetismo

La relación entre los fenómenos electrónicos y magnéticos se establece por: a) el fenómeno explicado por *Oersted* en 1819, en el que pudo determinar que con una corriente eléctrica se puede generar un campo magnético, y b) la

expresión de las ecuaciones de *Maxwell* que describen el comportamiento acoplado de dichos campos (magnético y eléctrico).

Tecnología inalámbrica y ondas electromagnéticas

Una onda es una perturbación de un medio continuo que se propaga con una forma fija y velocidad constante. Estas dos últimas características pueden modificarse en caso de que al pasar por el medio haya absorción y dispersión.

Las ondas satisfacen la ecuación de onda:

$$\nabla^2 F = \frac{1}{v^2} \frac{\partial^2 F}{\partial t^2}$$

donde ∇^2 es el *operador laplaciano*, que en el espacio cartesiano se escribe

como $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$, v es la velocidad de propagación de la onda y

t es el tiempo.

Como se puede observar, es una ecuación diferencial parcial de segundo orden y para asegurar la unicidad de sus soluciones se deben imponer dos condiciones iniciales o *de frontera*.

Debido a que la ecuación diferencial depende del tiempo y del espacio, la función F que la satisface, también dependerá de estas dos variables. Su solución más general es la suma de ondas que viajan a la derecha y a la izquierda, pero por condiciones iniciales puede reducirse a una sola onda. Las ecuaciones de *Maxwell* contienen de manera natural a esta ecuación.

Más adelante se verá que este tipo de ondas no necesitan un medio para transmitirse, de manera que pueden ser emitidas y recibidas sin la necesidad de que haya un material de por medio, como sucedería con el sonido.

Una onda se caracteriza por su amplitud A (distancia desde cero hasta la máxima amplitud), su frecuencia f (número de oscilaciones por segundo) y su longitud de onda λ (distancia entre dos valles consecutivos), y como las electromagnéticas viajan a velocidad c , se cumple la relación $c=f\lambda$, de manera que dada una frecuencia su longitud de onda queda establecida, por lo que se tiene todo un espectro de frecuencias por explorar y que siguen transmitiéndose a velocidad c .

La luz visible (y por ende todos los colores que puede percibir el ojo humano) es solo un pequeño rango de frecuencias. En orden ascendente se tienen las de radio, microondas, infrarrojo, visible, ultravioleta, rayos X y rayos gamma, siendo estos últimos los más energéticos. En un mundo ideal (esencialmente donde no hay un medio) las ondas transmitidas podrían propagarse hasta infinito, pero en el mundo real no es así pues entre los aparatos de emisión y recepción hay materiales que interactúan con la onda, como la atmósfera, paredes, lluvia, etcétera, y provocan una disminución de la amplitud de la onda, por lo que eventualmente, ya no existirá la onda pues la absorción implica pérdida de energía, de manera que las ondas solo pueden alcanzar distancias finitas.

Restricciones a la transmisión de señales

Las ondas electromagnéticas son capaces de transmitirse en el vacío y a una velocidad constante $c=299'792'458\text{ m/s}$. Considerando que el objeto más veloz creado por el hombre, la sonda solar *Parker*, viaja a $192'227.2\text{ m/s}$, la velocidad de la luz es muy superior, conformando un límite para la velocidad de cualquier objeto, pues nada puede viajar a una velocidad mayor, ya que si fuera el caso, violaría el *principio de causalidad* que establece que los efectos podrían observarse antes que sus causas.

La teoría electromagnética está estrechamente vinculada a la relatividad, otra rama de la física. Uno de sus postulados establece que «*La velocidad de la luz tiene el mismo valor en todos los marcos de referencia inerciales*⁷», de manera que no importa si se mide la velocidad desde un punto fijo o en movimiento, la velocidad de la luz siempre será la misma. Este hecho tiene implicaciones físicas relevantes en las telecomunicaciones, pues no importa cómo se mueva la fuente el mensaje a transmitir, se tardará lo que tarda la luz o más, pero nunca menos que eso. Considerando que la velocidad de la luz es muy alta, a cortas distancias puede no ser un problema y las comunicaciones a través del planeta Tierra se llevan a cabo con menores contratiempos. Pero si la misma señal, se enviase desde otro sistema solar, el mensaje podría llegar a la Tierra años después.

⁷ Los marcos de referencia inerciales son aquellos que son estáticos con respecto al observador o se mueven a velocidad constante, es decir que no están acelerados.

Referencias

Las referencias se encuentran divididas entre la bibliografía consultada por la autora para contrastar y corroborar la información expuesta en el artículo, y las fuentes de consulta que la autora y/o la editora recomiendan a los lectores, para abordar el tema expuesto y temas relacionados, en mayor profundidad.

Bibliografía consultada por la autora

- [1] Horswil. What is computation? [online] 2008. Disponible en: <http://www.cs.northwestern.edu/~ian/What%20is%20computation.pdf>
- [2] Advanced information on the Nobel Prize in Physics 2000 [online]. The Royal Swedish Academy of Sciences: Estados Unidos, 2014. Disponible en https://www.nobelprize.org/nobel_prizes/physics/laureates/2000/advanced.html
- [3] M. Raymer, Physics, Silicon, and the “Magic” behind the Internet Age en The Silicon Web: Physics for the Internet Age. Taylor & Francis: Estados Unidos, 2009. pp. 2,7-11,337-356 .
- [4] A. Giambattista. Resistance and Resistivity en Physics. McGraw Hill: Estados Unidos, 2009 pp. 648-652.
- [5] P. Alcalde San Miguel, Electrónica Analógica y Digital en Electrónica. Paraninfo: México, 2009. pp. 1-11.

- [6] L. De la Peña, Metales y Semiconductores: Teoría de Bandas en Introducción a la Mecánica Cuántica. Fondo de Cultura Económica: México, 2014. pp. 170-178.
- [7] D. J. Griffiths, Waves in one dimension en Introduction to Electrodynamics. Prentice-Hall: Estados Unidos, 1999. pp. 364-400, 443-457.
- [8] The fastest object made by humans is just trying to slow down [online]. Tim Fernholz (Quartz): San Francisco, Estados Unidos, 2018. Disponible en <https://qz.com/1353429/the-parker-solar-probe-is-the-fastest-man-made-object-ever/>
- [9] A. Beiser, Postulates of Special Relativity en Concepts of Modern Physics. McGraw Hill: Nueva York, 2003. pp. 3.

Fuentes de consulta sugeridas

- [0] «Curso de Electrónica Básica» de la Universidad del País Vasco: http://www.sc.ehu.es/sbweb/electronica/elec_basica/tema2/TEMA2.htm
- [1] Proyecto «Hyperphysics» (en español), del Departamento de Física y Astronomía de la Universidad del Estado de Georgia: <http://hyperphysics.phy-astr.gsu.edu/hbasees/hframe.html>
- [2] Proyecto internacional de la OMS, sobre investigación científica de los efectos de los campos electromagnéticos sobre la salud, en frecuencias de 0 a 300 GHz: <https://www.who.int/peh-emf/about/WhatisEMF/es/>

INTRODUCCIÓN A LA COMUNICACIÓN DE DATOS

Bibiana Mollinedo Rivadeneira

Al hablar de comunicación de datos se hace referencia a la forma en la cual la información digital (datos), se transmite a un receptor, por medio de una señal codificada, desde una fuente de origen (transmisor), a través de un canal determinado (medio).

Introducción

Durante un proceso cualquiera de transmisión de información digital, la misma se organiza, procesa y almacena en forma de datos binarios (cuya unidad mínima es el *bit -BInary digiT-*), y se transmite en forma de pulsos o señales eléctricas. Estos datos pueden ser alfabéticos, numéricos, simbólicos o una combinación de ellos. En un sistema de comunicación de datos, tanto el emisor como el receptor, interpretan *digitalmente* la información que

intercambian. Sin embargo, el canal de comunicación (medio), puede ser de naturaleza *analógica*.

Son ejemplo de dispositivos digitales, las computadoras, los teléfonos móviles, y los cajeros automáticos, entre otros.

Codificación de los datos

Codificación de la información en sistema binario

Haciendo referencia al concepto más básico de comunicación digital en el que dos dispositivos electrónicos comparten un canal de comunicación e intercambian información entre si, dicha información se genera en el transmisor y es interpretada en el receptor en forma digital (incluso, aunque en el canal de comunicación adquiera naturaleza analógica).

El mensaje a enviar es *codificado* en sistema binario, es decir, que se lo fragmenta y a cada fragmento se le asigna un *código* binario.

Este sistema sistema de base dos con el que se codifica la información digital, se compone de dos «dígitos», el 0 y el 1, que combinados representan los elementos humanamente conocidos como caracteres alfanuméricos. Cada uno de estos dígitos se denomina *bit*. Los *bits* se agrupan formando *palabras* de 8 bits, conformando así 1 *byte*.

Codificación de la señal

Si se desea enviar el mensaje codificado en base 2, el transmisor debe generar una señal eléctrica que será transformada en pulsos de tensión no nula para el 1, y nula para el 0. Estos pulsos tienen una duración fija y predefinida, que el receptor debe poder interpretar para *entender* el mensaje.

El conjunto de *bits* (mensaje) a transmitir, podrá ser representado por diferentes formas de onda que se conforman por estos pulsos eléctricos, y que dependerán de:

- **Los niveles de tensión:** bien sean *unipolares*, donde se transmite un único nivel de voltaje (denotado por V , y siendo $+V$ un nivel de voltaje positivo, y $-V$, un nivel de voltaje negativo), distinto a 0 ($+V$ para 1, y $0V$ para 0), o *bipolares*, donde se requieren dos niveles de voltaje ($+V$ para 1, y $-V$ para 0).
- **El tiempo de bit:** intervalo de tiempo transcurrido entre dos bits, es decir, el tiempo en el que una señal denota el nivel de tensión correspondiente a un bit. Al final de este tiempo, el nivel de tensión corresponde al bit siguiente.
- **El criterio con el que conmuta** de un nivel de tensión a otro.

Para generar estos pulsos, se suelen emplear tres tipos de sistemas:

1. Los de **no retorno a cero** (NRZ o por sus siglas en inglés, NRZ), en los que el pulso se mantiene constante a lo largo de un bit, sin regresar a cero.
2. Los de **retorno a cero** (RC o por sus siglas en inglés, RZ), en los que el pulso no se mantiene constante a lo largo de un bit, sino que en la mitad, regresar a cero.
3. Los **bifásicos**, en los que al igual que en los sistemas RZ, el pulso cambia en la mitad del bit, hacia el nivel opuesto de tensión en vez de hacerlo a un nivel de tensión nula como los RZ.

Sistemas NRZ de no retorno a cero (NRZ)

NRZ-L: El nivel de tensión de la señal se mantiene constante hasta detectar un cambio de señal (es decir, cuando cambia de 1 a 0, o viceversa).

NRZ-M: El nivel de tensión mantiene el último valor de bit y no retorna a cero, es decir se mantiene constante hasta que el valor de bit tome el valor 1. El valor de tensión de la forma de onda binaria cambia al detectar un 1, y se mantiene constante al detectar un 0 (este “cambio” de un valor de tensión al otro se denomina flanco).

NRZ-S: Similar a NRZ-M con criterio de cambio al detectar un 0.

Sistema RZ de retorno a cero (RC)

En este sistema, el nivel de tensión no es constante a lo largo de un bit sino que en la mitad regresa a cero.

Los valores de tensión adquieren voltaje ($+V$) durante la mitad del tiempo de bit, y vuelven a 0 según los bits toman los valores 1 y 0 respectivamente.

Sistemas bifásicos

Bifase-L: Durante la mitad del tiempo de bit, adquiere $+V$ y vuelve a 0 para el 1, y 0 y $+V$ para el 0.

Bifase-M: Cambia de flanco al comenzar el tiempo de *bit*, luego de un tiempo igual a la mitad del tiempo de bit, cambia el flanco si detecta un 1, y mantiene su valor si detecta un 0.

Bifase-S: Similar a Bifase-M, cambia de flanco si detecta un 0, y mantiene su valor si detecta un 1.

Bipolar

Durante la primera mitad del tiempo de bit, adquiere valores de tensión $+V$ y $-V$ para el 1 y 0 respectivamente. Durante el tiempo restante de bit retorna al nivel 0 de tensión.

Transmisión de los datos

Según la cantidad de canales disponibles para transmitir los datos, es posible encontrar dos tipos de transmisión diferentes:

1. **Transmisión de datos en serie**, en la que los datos se transmiten en un solo hilo de manera secuencial, es decir, uno detrás de otro.
2. **Transmisión de datos en paralelo**, en la que los datos se transmiten de forma simultánea en varios hilos.

Transmisión de datos en serie

En una transmisión en serie (o serial), los *bits* que conforman el mensaje se envían uno a uno por un único hilo o canal de comunicación.

Velocidad de transmisión: Siendo t bit el tiempo de duración de un bit, para la transmisión de 1 byte (equivalente a 8 bits) se requiere un tiempo de transmisión $T_{tx}=8t$ bit .

Distancia: En conductores de par de cobre es posible transmitir datos en distancias de hasta 500 metros.

Tipos de transmisión en serie

Según la dirección en la que se efectúa la transmisión, es posible diferenciar tres tipos de transmisión en serie:

Simplex: La transmisión se realiza en un solo sentido, de “*Sim*” a “*plex*”, es decir que “*plex*” sólo recibe información proveniente de “*Sim*” y no le es posible responder. Ejemplo de ello es la información que el teclado le envía al microprocesador.

Semi duplex (Half Duplex): La transmisión se realiza en ambos sentidos pero no simultáneamente. Mientras “*Half*” envía información a “*Duplex*”, “*Duplex*” debe esperar disponibilidad del canal. Un ejemplo de este tipo de comunicación, son los transceptores de radio portátiles (*walkie-talkies*).

Duplex completo (Full Duplex): La transmisión se realiza en ambos sentidos y de manera simultánea. Un ejemplo de ello son las comunicaciones telefónicas.

Sincronismo

En las formas de onda NRZ, es posible observar ciertos inconvenientes al intentar diferenciar dos bits consecutivos con valor 1, así como dos mensajes recibidos simultáneamente. Esto advierte la necesidad de conocer el inicio y fin de un bit, de un byte o de un bloque, para una correcta comunicación.

Para ello existen dos formas:

- **Transmisión asíncrona:** emplea un bit al inicio y fin de cada conjunto de bits que se desea diferenciar en la detección, para lo que se requieren bits extra además de la información (por ejemplo bits en 0 al inicio de un conjunto de 8 bits, y bits en 1 al final del mismo).
- **Transmisión sincrónica:** las señales de reloj⁸ de transmisor y receptor deben estar sincronizadas, es decir, oscilar a la misma frecuencia. No se emplean bits de inicio o fin de bloques, sino algunos bits de sincronismo que, al ser detectados por el receptor, sincronizan su reloj con el del transmisor. Este método permite una velocidad de comunicación superior en comparación a la asíncrona, dado que no debe reservar bits de inicio y fin.

8 *Señal de reloj (clock):* se trata de una señal eléctrica que cambia de un estado alto (+V) a uno bajo (-V o 0) a una frecuencia (período) a la que se desea el sincronismo.

Estándar RS-232

La norma EIA RS-232C es una de las normas para transmisión de datos en serie, que define las características eléctricas, físicas y funcionales, de las interfaces de comunicación en serie así como las normas para aplicaciones específicas.

	Bit 0	Bit 1
Transmisor	+3V, +15V	-3V, -15V
Receptor	+3V, -25V	-3V, -25V

1. Tabla: Especificaciones eléctricas de voltaje del estándar RS-232

Por otra parte, la norma recomienda *conectores* de 25 pines y 21 señales, como el conector Db25. Dependiendo de la comunicación y funciones necesarias, pueden emplearse grupos de 3, 5, 7 o 9 señales. Sin embargo, un conector con una menor cantidad de pines como el conector DE9, según sus funciones, podría ser correcto.

Aplicaciones

La transmisión de datos en serie se encuentra presente en las comunicaciones entre los dispositivos actuales, como por ejemplo:

- Bus⁹ de comunicación *Serial ATA* (SATA), utilizado en ordenadores.
- *Ethernet*, en redes de información.
- USB (*Universal Serial Bus*), en periféricos como el ratón, el teclado, o la impresora.

USB (Universal Serial Bus)

Se trata de un bus de expansión externa que actualmente conecta, comunica y provee alimentación eléctrica a diversos dispositivos electrónicos. En la tabla a continuación, pueden verse las características y velocidad máxima de transferencia de datos, según las diferentes versiones.

Versión	Características	Tipo	Velocidad ¹⁰
1.0	Empleado en dispositivos de interfaz humana como ratones, teclados y cámaras.	Semidúplex	1.5 Mbps
1.1	Implementa un algoritmo de división de ancho de banda.		12 Mbps

- 9 NdE: la palabra inglesa «bus» utilizada en un contexto informático (p.e.: en «bus de datos») es una analogía al «medio de transporte digital» empleado para transferir datos.
- 10 NdE: la velocidad en la transmisión de datos se mide en cantidad de bits transportados por segundo (bps), donde M hace referencia a los «mega» bits por segundo, y G, a los «giga» bits por segundo. No debe confundirse con *Megabytes* y *Gigabytes*.

2.0	Además de las dos líneas de datos, cuenta con dos líneas de alimentación.		480 Mbps
3.0	Incluye cinco contactos adicionales, y activa una etapa de bajo consumo cuando el dispositivo se encuentra en desuso.	Dúplex completo	4.8 Gbps ¹¹

Especificaciones técnicas

Cada versión de USB corresponde a un protocolo de comunicación de datos en serie que especifica aspectos eléctricos, físicos y funcionales del mismo.

Características eléctricas

Las señales eléctricas en USB hasta la versión 2.0 inclusive se interpretan en modo diferencial, es decir que el nivel de voltaje se determina con la diferencia en voltaje entre una línea y la otra, lo que reduce el ruido. En el caso de la versión 3.0 se emplean dos líneas de transmisión adicionales para comunicación dúplex completa.

Los niveles de tensión son los siguientes:

- Bit en 0: [+0, +0.8] Voltios
- Bit en 1: [+3, +3.6] Voltios

11 La versión 3.1 del estándar, especifica velocidades de hasta 10 Gbps, y el conector «Tipo C» reversible que se usará a ambos extremos del cable. El estándar 3.2, especifica velocidades de transferencia de hasta 20 Gbps, empleando el conector «Tipo C».

Características físicas y funcionales

La información en un *Universal Serial Bus* viajan en un par trenzado¹². Los cables USB tienen una distancia limitada de aproximada de 3 metros.

Tipos de conectores

Los conectores USB se dividen en los siguientes grupos:

- Tipo A
- Tipo B

Hasta la versión 2.0, cada tipo puede subdividirse en *mini* y *micro* A y B, respectivamente. La versión 3.0 incluye una subdivisión *micro* B (correspondiente al tipo B).

Compatibilidad

- Los dispositivos USB 3.0 Tipo A, se pueden conectar en conectores USB 2.0 del mismo tipo y viceversa.
- Los dispositivos USB 2.0 Tipo B y micro-B, se pueden conectar en conectores USB 3.0 del mismo tipo. Sin embargo, no ocurre lo mismo con los dispositivos USB 3.0.

12 Se trata de un cable con dos hilos conductores entrelazados.

Redes USB

Una red USB es un conjunto de dos o más dispositivos USB conectados a un mismo nodo. Mediante un HUB (concentrador) USB, pueden ser conectados dos o más dispositivos USB a un mismo puerto hembra, y extenderse además, la longitud de los cables de 3 a 10 mts de largo.

Un HUB puede ser de dos tipos, dependiendo de la forma en la que adquiere la energía:

- *Activo*: obtiene energía eléctrica de una fuente externa para distribuir a los dispositivos USB que se concentran en él.
- *Pasivo*: obtiene toda la energía desde el puerto USB principal.

Una red USB puede extenderse a un máximo de 127 dispositivos, y funcionar correctamente con hasta 7 capas desde el HUB principal o raíz (*root*).

Transmisión de datos en paralelo

En la transmisión en paralelo se dispone de más de un hilo para la transmisión de los datos. El número de hilos es siempre potencia de 2 ($2^3=8$, $2^4=16$, $2^5=32$, etc.).

En una transmisión de datos en paralelo se envían n bits por vez (donde n es al cantidad de hilos disponibles). Si bien se dispone de n canales o hilos de transmisión sobre los que se envían datos de manera simultánea, en cada uno de ellos la transmisión es en serie.

En comparación a la transmisión en serie, la **velocidad de transmisión** en paralelo es mayor, dado que se envían n bits por vez, en lugar de 1. Siendo t bit el tiempo de duración de un bit (sea éste, 0 o 1), un sistema de transmisión de datos en paralelo puede transmitir n bits al cabo de un t bit.

El **estándar IEEE 1284** es posiblemente uno de los más empleados para la comunicación en paralelo con periféricos antiguos como impresoras.

Referencias

- [0] W. Tomasi, *Sistemas de Comunicaciones Electrónicas*. Prentice Hall: México.
- [1] Universidad Tecnológica del Perú, *Introducción a la Ingeniería de Telecomunicaciones*.

INTRODUCCIÓN A LAS COMUNICACIONES ÓPTICAS

Bibiana Mollinedo Rivadeneira

Revisión técnica de
Fernanda Hernández González

Al hablar de comunicaciones ópticas se hace referencia a un medio de transmisión que transporta la información en forma de pulsos de luz.

Dada la limitante física que postula que nada puede viajar más rápido que la luz, puede inferirse que la comunicación óptica es la más veloz conocida hasta el momento.

La fibra óptica que actúa como medio de transporte de este tipo de comunicaciones es la que mejores resultados reporta frente al ruido, dado que por un lado, se trata de un material dieléctrico (de baja conductividad eléctrica) por lo que la señal interior queda aislada de las interferencias

electromagnéticas externas, y por el otro, debido a su estructura interna, las fibras no generan inducción entre ellas (como sí sucede con cables conductores), por lo que no se produce diafonía. Finalmente, al funcionar en una frecuencia diferente que la de las ondas de radio, la fibra óptica es inmune a las interferencias de esta.

La fibra óptica frente a otros medios de transmisión

Frente a otros medios de transmisión, la fibra óptica presenta las siguientes diferencias:

- **Capacidad:** Los sistemas de comunicaciones ópticos poseen una capacidad de transmitir información significativamente superior a otros medios de transmisión, debido a la posibilidad de transportar numerosas señales de información de manera simultánea a través del mismo medio, lo que se traduce en un mayor ancho de banda.
- **Tamaño y peso:** El diámetro y peso de un cable de fibra óptica es inferior al de un cable de cobre con similar capacidad, lo que torna a la fibra más fácil de transportar e instalar.
- **Inmunidad a las interferencias:** La fibra óptica es inmune a las interferencias electromagnéticas o de radiofrecuencia y no genera interferencias por si misma.
- **Seguridad:** Un sistema de comunicaciones óptico no se puede intervenir por medio de mecanismos eléctricos convencionales, y difícilmente por medios ópticos, puesto que son fácilmente detectables.

- **Fiabilidad y mantenimiento:** Los enlaces de fibra óptica bien diseñados son inmunes a condiciones adversas de humedad y temperatura y se emplean incluso para cables subacuáticos. Se trata de enlaces de larga vida útil, y el mantenimiento requerido es menor al de otros medios de transmisión.
- **Versatilidad:** Los sistemas de comunicación ópticos son compatibles con la mayoría de los formatos de comunicaciones de datos, voz y vídeo.
- **Expansión:** Los sistemas de fibra óptica son escalables, por lo que es posible transformar un sistema de baja velocidad a uno de alta velocidad.
- **Regeneración de la señal:** Los enlaces de fibra óptica tienen un alcance mayor al de otros sistemas de comunicación antes de requerir un amplificador de señal, los que a su vez extienden significativamente dicho alcance.
- **Conversión electro-óptica:** Para transportar señales de información en un sistema óptico, las señales eléctricas deben convertirse en pulsos de luz, lo que requiere de equipamiento extra.
- **Caminos homogéneos:** Se requiere un camino físico dedicado para la instalación de enlaces de fibra óptica.
- **Instalación y mantenimiento:** Tanto para la instalación como para el mantenimiento de la fibra óptica se requieren técnicas y

equipamientos especiales, debido a la particularidad de los materiales que la componen. Por tal motivo, la reparación de un cable de fibra óptica dañado presenta una mayor complejidad de procedimiento que la reparación de un cableado convencional.

Aplicaciones

La fibra óptica se ha convertido en un medio de transmisión transversal a todas las comunicaciones, sin importar tipo de servicio, formato de datos, etc.

Se usan cables de fibra tanto para conectar distancias cortas como comunicaciones *inter-oficina*, así como grandes distancias intercontinentales, y en ambos casos a alta velocidad.

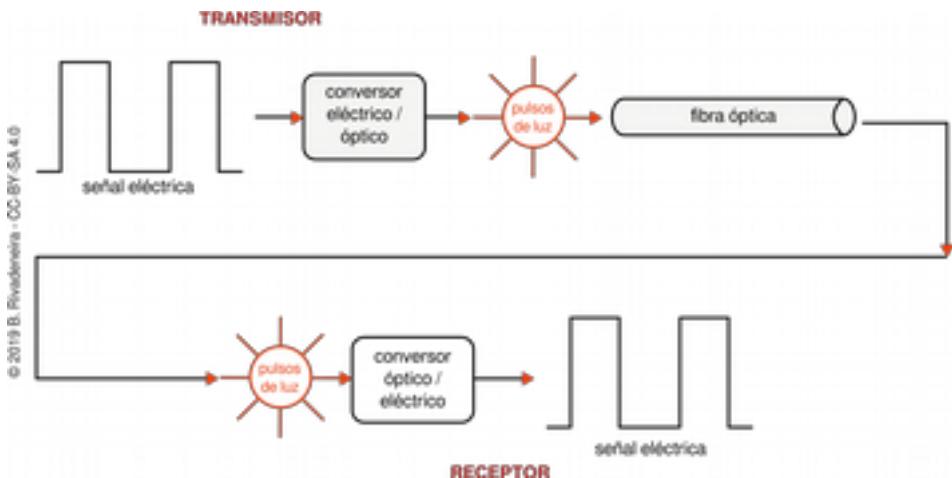
La infraestructura de los sistemas de telefonía y televisión se compone parcialmente de enlaces de fibra óptica en sus redes de distribución (redes híbridas como HFC). En particular la tendencia es llegar con fibra hasta el hogar, es decir hasta el abonado, coexistiendo con otros medios de transmisión en el camino de conexión desde el proveedor hasta el abonado.

Respecto a las redes de datos, o más específicamente a la red de redes, Internet, son los enlaces de fibra óptica, en especial los submarinos los que permiten comunicaciones casi entre dos puntos cualesquiera del planeta.

Fundamentos de las comunicaciones ópticas

Un sistema de comunicaciones ópticas transporta la información a través de un medio, en forma de pulsos de luz, y en el transmisor y receptor se realizan

las adaptaciones correspondientes de la señal, como se muestra en el siguiente diagrama.



La señal se origina en el **transmisor**. Ésta, que es de naturaleza eléctrica, debe ser convertida a pulsos de luz para inyectarse en el canal de comunicación. Para ello se usa un conversor eléctrico-óptico, y la señal eléctrica de información se modula empleando un diodo de emisión de luz (LED) o un diodo de inyección láser.

Las señales de luz viajan por la fibra óptica, el **medio de transmisión** cuyo diseño se realiza conforme a las propiedades físicas de la luz y de los materiales del cable, guiando a la señal a través del canal de comunicación.

En el **receptor** se realiza el proceso inverso. La señal que llega es de naturaleza óptica de manera que se emplea un conversor óptico-eléctrico sensible a los pulsos de luz, que produce una señal de información eléctrica.

Fundamentos físicos

Los fundamentos físicos en los que se sostienen las comunicaciones ópticas, radican principalmente en los fenómenos que intervienen en la propagación de la luz, y que son descritos a continuación.

Propiedades de la luz

La luz se comporta como una onda electromagnética. Las ondas de luz que se propagan en los sistemas de comunicaciones ópticas corresponden a la sección infrarroja IR, por debajo de la luz visible. Su frecuencia de cambio u oscilación ronda los 230 THz (2.3×10^{14} Hz). Su longitud de onda se encuentra en el rango de los 1310 nm a 1550 nm, y queda determinada por la *ecuación de longitud de onda*:

$$\lambda = \frac{c}{f}$$

Donde c es la velocidad de la luz en el vacío y f , la frecuencia de oscilación.

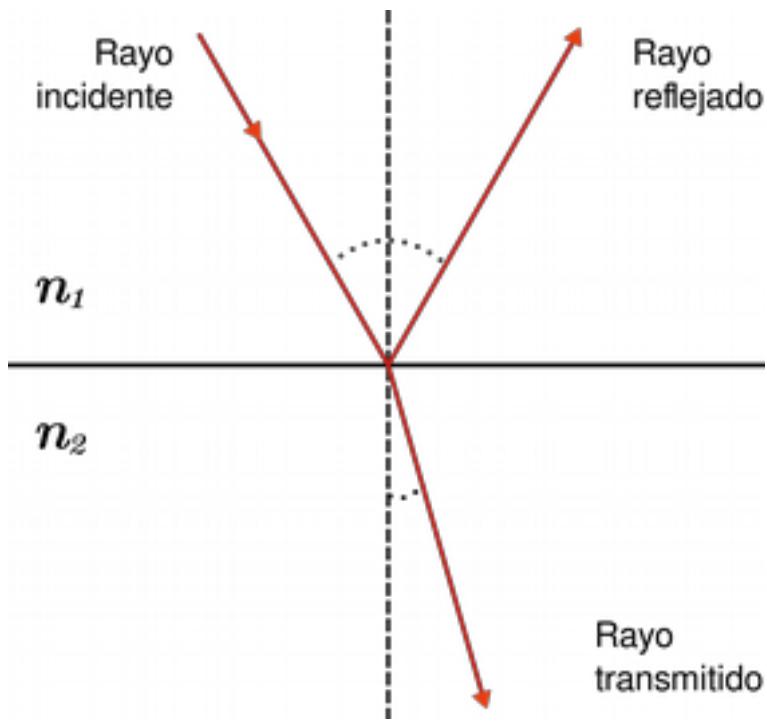
Reflexión y Refracción de la luz

Cuando un haz de luz cambia de medio, también lo hace su velocidad y dirección, según las propiedades materiales de los medios en los que se transmite.

El estudio de los fenómenos físicos que suceden ante la incidencia de un haz de luz en la frontera entre dos medios, permite el diseño de un conductor a modo de guía de onda de la luz como lo es la fibra óptica, según las

propiedades de los materiales que la componen y el medio en el que se encuentra.

El fenómeno de la reflexión es explicado por la *Ley de Snell*. La misma especifica que para un haz de luz propagándose en un medio con índice de refracción n_1 , hacia la frontera de un medio de índice de refracción n_2 , el comportamiento del haz queda determinado por $n_1 \sin(\theta_1) = n_2 \sin(\theta_2)$.



Reflexión

Se da cuando el haz de luz cambia de dirección al incidir en la frontera entre dos medios, de manera que se continúa propagando por el primero de los medios, parcial o totalmente.

En el diagrama precedente, el rayo reflejado se propaga en el medio con índice de refracción n_1 , formando un ángulo θ_r' con la normal a la frontera entre los medios.

Este fenómeno es objeto de interés en el estudio de la fibra óptica ya que el objetivo durante la transmisión de información en forma de pulsos de luz es mantener los rayos en la fibra y más específicamente en el centro del cable.

Refracción

Es la producida por el haz de luz que se continúa propagando a través del medio con índice de refracción n_2 , a una velocidad y en una dirección diferentes al rayo de incidencia.

En el diagrama anterior, el rayo refractado forma un ángulo θ_2 con la normal a la frontera. Para el caso de las comunicaciones ópticas, el diseño de la fibra debe ser tal que las propiedades de refracción de los materiales impida la refracción de los haces de luz.

Ángulo crítico

El ángulo crítico es un parámetro necesario de diseño para las comunicaciones ópticas, dado que de este depende evitar la refracción. Se define como aquel ángulo θ_c de incidencia máximo para el que no se produce el fenómeno de refracción, por lo cual, la reflexión es completa y no hay propagación hacia el otro medio.

En comunicaciones ópticas un ángulo de incidencia mayor al ángulo crítico, implica que parte de la señal de información se propague fuera del medio de transmisión, es decir, fuera de la fibra.

Refracción de los materiales

El índice de refracción de un material es un valor adimensional determinado

por la ecuación: $n = \frac{c}{v}$ donde c es la velocidad de la luz en el vacío y v la velocidad de la luz en el material en cuestión.

Un cable de fibra óptica se fabrica con aquellos materiales cuyas propiedades de refracción permiten hacer de guía de onda para los haces de luz.

Fundamentos técnicos

Composición de la fibra óptica

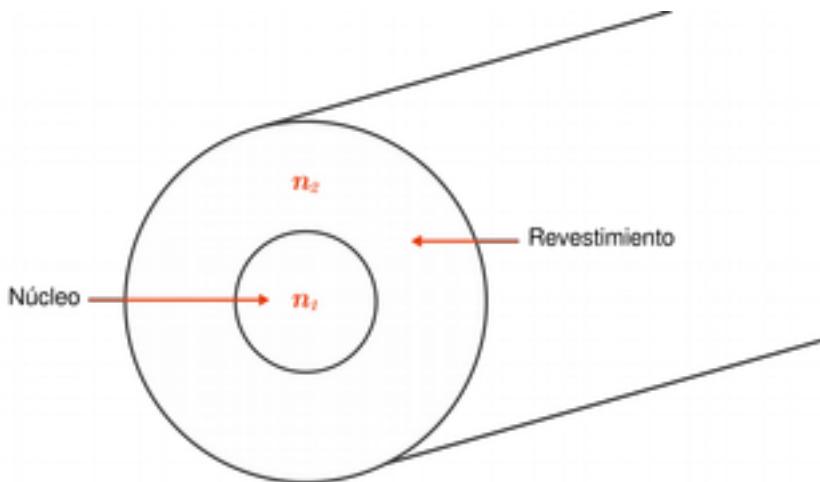
Una fibra óptica consiste en un material transparente y cilíndrico por el que se propagan ondas luminosas. Está compuesta por tres capas diferenciadas:

- **Núcleo:** En esta capa se propaga la luz, generalmente de vidrio de sílice, pero también puede ser plástico. El diámetro del núcleo varía, según el tipo de fibra, entre 8 y 100 μm .
- **Revestimiento:** Cubre el núcleo, y sus características de refracción junto con las del núcleo (n_1 y n_2) aseguran que los haces de luz se reflejen siempre en las paredes de este y no se refracten nunca a los

medios externos. Puede ser de vidrio o de plástico. El diámetro típico del revestimiento es de $125\mu m$.

Típicamente los índices de refracción del núcleo y del revestimiento difieren levemente entre ellos. Los valores se encuentran alrededor de $n_1=1,5$ y $n_2=1,48$ respectivamente, mientras que el índice de refracción del aire es aproximadamente 1 (aunque a los fines prácticos se toma como 1).

- **Recubrimiento:** Es una capa que envuelve al revestimiento. Le provee a la fibra de una protección extra frente a condiciones externas. Suele ser de plástico o acrílico. Según la protección que se requiera, el diámetro del recubrimiento puede ser de 900 o $2000\mu m$.



Clasificación de la fibra óptica

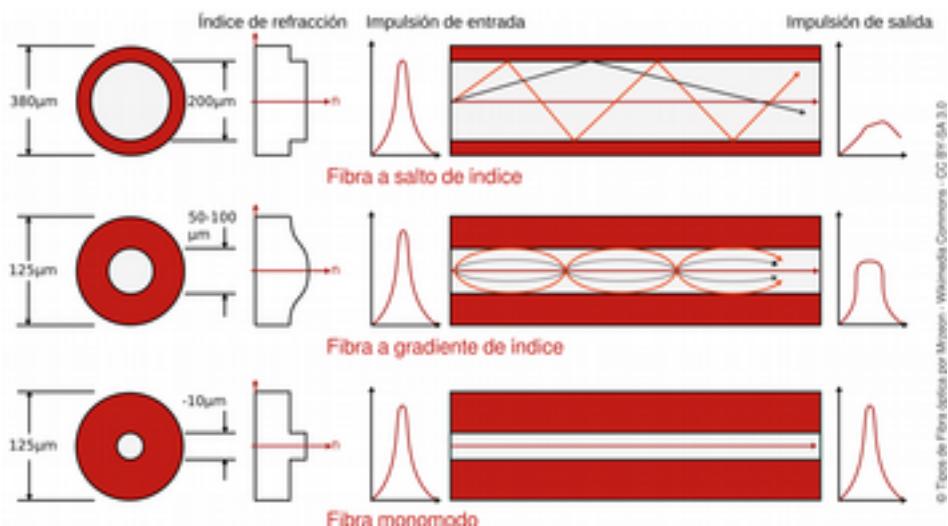
Según la cantidad de modos (camino o trayectoria que describe un haz de luz) que transporte el núcleo, se pueden clasificar los cables de fibra en *monomodo* (modo único o modal) y *multimodo* (múltiple o multimodal).

Algunas de las características distintivas de cada uno son las siguientes:

- **Monomodo**
 - Núcleo pequeño respecto al recubrimiento.
 - Menor dispersión.
 - Adecuado para largas distancias respecto a la fibra multimodo, ya que el haz de luz recorre un camino recto .
 - Emplea láser como fuente de luz.
 - Costo elevado de equipamiento respecto a la fibra multimodo.
 - Se emplea en redes de larga distancia (más de 60 km).
- **Multimodo**
 - Núcleo grande respecto al recubrimiento.
 - Mayor dispersión y por ende mayor pérdida de la señal.

- Menos adecuado para largas distancias respecto a la fibra monomodo, ya que los haces de luz se reflejan constantemente en las paredes del núcleo.
- Emplea LED como fuente de luz.
- Bajo costo de equipamiento respecto a la fibra monomodo.
- Se emplea en redes de corta distancia (hasta 2 km).

Según la variación del índice de refracción desde el núcleo hacia el recubrimiento, se puede clasificar el cable de fibra como sigue:



- **Salto de índice o índice de salto (superior):** El índice de refracción del núcleo hacia el recubrimiento varía abruptamente, lo que describe reflexiones de los haces en las paredes del núcleo como se observa en el gráfico.

- **Gradiente de índice o índice de graduado (intermedio):** El índice de refracción varía gradualmente desde el núcleo hacia afuera, provocando cambios de dirección graduales en los rayos.
- **Monomodo, índice de escalón (inferior):** La función que describe la variación del índice de refracción del núcleo respecto al recubrimiento forma un escalón, lo que guía en una trayectoria sin reflexión al haz de luz.

Aertura numérica

Se trata de un parámetro dado por un ángulo de incidencia máximo a la fibra que define el rango de apertura o cono de aceptación por debajo del cual los rayos se propagan en la fibra, y por ende, por encima del cual los haces de luz cambian de medio y se pierde información.

Pérdidas en fibra óptica

Se denomina pérdida a las atenuaciones de potencia sufridas por la señal luminosa a medida que se propaga por la fibra óptica. Cuantitativamente, la atenuación está determinada por la siguiente ecuación:

$$A_t (dB) = 10 \log \frac{P_{salida}}{P_{entrada}}$$

Una atenuación de $3 dB$ reduce en un 50% la potencia de la señal. Según el tipo de fibra, los valores típicos de atenuación en dB/km son:

- **Unimodal:** 0,4 - 0,5 dB/km

- **Índice gradual:** 4 - 5 dB/km
- **Índice escalonado:** 6 dB/km

Las **causas de atenuación** o pérdida de potencia en los cables de fibra pueden estar dadas por:

- **Absorción** intrínseca al material, que se da cuando las impurezas de la fibra absorben luz y la disipan en forma de calor. La absorción puede clasificarse en absorción ultravioleta, infrarroja, o por resonancia de iones.
- **Pérdidas en material o dispersión de Rayleigh** intrínseca al proceso de fabricación. Se da cuando la fibra presenta irregularidades submicroscópicas y los haces de luz se reflejan y refractan en las mismas.
- **Dispersión cromática o por longitudes de onda.** Dado que la velocidad de un haz de luz en un medio depende de su longitud de onda λ , las variaciones de la fuente de luz producen rayos de diferentes λ que viajan a diferentes velocidades, resultando en una onda distorsionada.
- **Pérdidas por radiación** debidas a discontinuidades y cambios de dirección en el cable de fibra, ya sea producto de los procesos de fabricación, o presión y tensión aplicada a la fibra.

- **Dispersión modal.** La señal compuesta, formada por todos los haces de luz que se propagan a lo largo de la fibra, llega al receptor ensanchada respecto al transmisor. Esto se debe a que los diferentes caminos que recorren los rayos de luz, les imprimen diferentes velocidades.
- **Pérdida por empalme:** Cuando dos cables de fibra se empalman, las imperfecciones en la unión reporta pérdidas de potencia. Las pérdidas por empalme están sujetas a la calidad de la fibra, la habilidad del operario que realiza el empalme, así como la calidad del equipamiento. Los valores típicos de pérdida por empalme en fibra son:
 1. Fusión: 0,02 - 0,2 dB
 2. Mecánico: 0,1 - 0,5 dB

Observaciones finales

La infraestructura de muchos servicios de telecomunicaciones se está reemplazando parcial o totalmente por fibra óptica, tanto en redes de datos, como de telefonía y televisión debido a las grandes velocidades de transmisión que permite. La fabricación de cables de fibra imprime a los enlaces ópticos cada vez mayores velocidades y menores pérdidas de potencia. Las velocidades de transmisión máximas a enero de 2020, rondan las decenas de *Terabits*.

Bibliografía consultada por la autora

- [0] Wayne Tomasi (2003), “Sistemas de comunicaciones electrónicas”. México: PEARSON EDUCATION.
- [1] bob chomycz (1998), “Las instalaciones de fibra óptica: fundamentos, técnicas y aplicaciones”. España: MCGRAW-HILL.

INFORMÁTICA APLICADA

PARTE III: 1. BASES DE DATOS

ENMASCARAMIENTO DE DATOS

Andrea Navarro Moreno

Sobre el enmascaramiento de datos

El **enmascaramiento de datos** o *data masking* puede definirse como el reemplazo de datos sensibles con el objetivo de proteger la información de una revelación no intencional.

Los valores por los que serán reemplazados dependerá del uso que se le dará a los datos. En el caso de un estudio científico o estadístico estos pueden ser reemplazados por valores aleatorios sin significado, sin embargo, si estos datos se utilizarán para en la prueba de aplicaciones y sistemas complejos, será necesario que los mismos resulten verosímiles haciendo más sencillo detectar errores de comportamiento.

Casos prácticos

En el siguiente ejemplo se ve una tabla sencilla sin enmascarar que cuenta con datos personales de los usuarios y su última fecha de ingreso al sistema.

DNI	Nombre	Apellido	Sexo	Último ingreso
765875	María	Rodriguez	F	10/04/2018
858911	Pedro	Sosa	M	11/04/2018
555842	Sonia	Martinez	F	17/03/2018

2. Tabla: Tabla sin enmascarar. El campo DNI actúa como clave primaria.

Si se quiere hacer un estudio estadístico de la frecuencia de ingreso de los usuarios diferenciados por sexo entonces el resto de los datos sensibles no son importantes y pueden ser reemplazados directamente por el carácter X. En este caso la clave primaria es un dato sensible por lo que se reemplazará por un número incremental.

DNI	Nombre	Apellido	Sexo	Último ingreso
1	XXXXXX	XXXXXX	F	10/04/2018
2	XXXXXX	XXXXXX	M	11/04/2018
3	XXXXXX	XXXXXX	F	17/03/2018

3. Tabla: Base de datos con datos sensibles reemplazados por valor incremental y caracteres sin sentido. Útil para un análisis estadístico.

Si en cambio se quiere utilizar dichos datos para un grupo de *testers* que verificarán errores en el sistema, los mismo se modificarán por otros de igual tipo. De esta manera podrán detectarse errores como nombres o DNI

mostrados por pantalla truncados, en minúscula, caracteres especiales incorrectos, problemas con nombres dobles, entre otros.

DNI	Nombre	Apellido	Sexo	Último ingreso
572835	Laura	Torres	F	10/04/2018
749293	Marcos	Córdoba	M	11/04/2018
683954	Lucía	Mendez	F	17/03/2018

4. Tabla: Base de datos con datos sensibles reemplazados por otros datos del mismo tipo y significado. Útil para el análisis de prueba de una aplicación.

Esta modificación, dependiendo de su uso, naturaleza o alcance, debe cumplir con normas, estándares y leyes vigentes de protección de datos y privacidad. Esto es especialmente necesario cuando se trabaja, por ejemplo, con datos médicos y de salud de pacientes.

En el caso de una prueba a un sistema, una alternativa al enmascarado es la **generación artificial de datos falsos** utilizando un *software de generación automática de datos*. Esto eliminaría todo peligro de obtención de datos sensibles y la necesidad de analizar y clasificar los datos sensibles de los que no lo son. Con este método, sin embargo, no puede garantizarse que se dupliquen ciertos comportamientos y sutilezas que se encuentran en los datos generados por el uso real de un sistema durante cierto periodo de tiempo. Por esta razón, el enmascaramiento es una opción más fiable a la hora de realizar pruebas sobre sistemas en producción.

Uno de los desafíos a la hora de enmascarar una base de datos es lograr **mantener el significado** de la información. Si demasiados datos son oscurecidos entonces la muestra puede dejar de ser representativa y no tener utilidad. Es necesario, por lo tanto, no enmascarar más datos de los imprescindibles para garantizar que no puedan recrearse o ser inferidos los datos sensibles.

En este ejemplo se puede ver una base de datos médicos que contiene el diagnóstico de cierta enfermedad en un número de pacientes, cuándo fue diagnosticado y si se encuentra en tratamiento por la misma.

Id	Nombre	Apellido	Sexo	Nacimiento	Dirección	Diagnóstico	Tto.
4	María	Rodríguez	F	08/10/1980	Sarmiento 510, Chaos Malal	10/0/2018	Si
5	Pedro	Sosa	M	15/03/1988	Varvarco 150, Andacollo	11/04/2018	No
6	Sonia	Martínez	F	20/12/1985	Av. Trannack 1200, Zapala	17/03/2018	Si

El grupo de investigadores que utilizará esta base de datos quiere buscar correlaciones entre la enfermedad y la zona en la que viven los pacientes como así también entre la enfermedad y la edad. Si se enmascararan la fecha de nacimiento y la dirección (datos que solos o en combinación podrían permitir identificar pacientes) no podría realizarse el estudio, por lo que en vez de

eliminar directamente estos valores pueden convertirse en datos menos identificativos aunque aún útiles.

Id	Nombre	Apellido	Sexo	Edad	Dirección	Diagnóstico	Tto.
4	XXXX	XXXX	F	37	Q8340	10/0/2018	Si
5	XXXX	XXXX	M	30	Q8354	11/04/2018	No
6	XXXX	XXXX	F	32	Q8354	17/03/2018	Si

La dirección ha sido modificada por el código postal, y la fecha de nacimiento por la edad actual. Ambos procedimientos son fácilmente programables y dan como resultado una base de datos sin valores sensibles.

Al enmascarar una base de datos relacional, es importante tener en cuenta la estructura de la misma para **mantener la integridad referencial**. En caso de que las claves primarias sean datos sensibles como números de documento de identidad u otros de identificación única, será necesario realizar el enmascaramiento de manera uniforme en todas las apariciones de la clave.

Diferencia entre enmascarar y encriptar datos

Al **enmascarar** se modifica u oscurece un dato sensible reemplazándolo por otro dato que permita el funcionamiento normal del sistema que los está utilizando, evitando la identificación de información sensible. Esta información no debe, por definición, ser recuperable.

Al **criptar** (o cifrar) datos lo que se realiza es una modificación de los mismos por otros no legibles, producidos a partir del dato inicial al cual se le ha realizado un número determinado de cálculos matemáticos. El mensaje puede ser descifrado utilizando el algoritmo correspondiente y una clave. Esta información es por tanto recuperable por definición.

Enmascarado estático y enmascarado dinámico

El **enmascarado estático** reemplaza los datos sensibles en reposo. A partir de la base de datos en producción se genera una copia a la que se somete a una serie de transformaciones de datos dando como resultado una base enmascarada.

Los datos quedan permanentemente enmascarados, por lo tanto, un posible atacante que consiga acceso a la base de datos no podrá obtener datos sensibles haciendo innecesaria la programación adicional para proteger los datos originales. Toda la transformación se hace por adelantado por lo que no se necesitan operaciones adicionales cada vez que se inicia una consulta. Esta conversión inicial puede tardar una gran cantidad de tiempo dependiendo del tamaño de la base, y los cambios realizados en la base de datos en producción no se verán reflejados en la base enmascarada dando como resultado datos no actualizados.

En el **enmascarado dinámico**, en cambio, se enmascaran los datos en tránsito. Al realizar la consulta, esta es modificada por un *proxy* para que los datos identificados como sensibles se alteren en los resultados.

Este método es utilizado en modelos de acceso basados en rol donde cada rol es configurado para tener accesos específicos a los datos originales o a los enmascarados.

A diferencia del método estático no requiere un trabajo previo sobre la base de datos en producción y los datos se encuentran actualizados en todo momento. Sin embargo, dependiendo de la cantidad de datos sensibles y roles diferentes, la configuración inicial puede ser muy extensa y el rendimiento dependerá de la cantidad de consultas que se realicen.

Adicionalmente es necesario emplear algoritmos de programación, para evitar que los datos enmascarados resultantes en las consultas no sean rescritos en la base de datos perdiendo en el proceso el dato original. Por esta última razón el enmascarado dinámico se utiliza idealmente en escenarios de solo lectura.

Finalmente la seguridad del *proxy* es fundamental ya que si es accedido o evadido entonces la base de datos original estaría comprometida.

Enmascarado estático	Enmascarado dinámico
Se genera una copia enmascarada de la BD	Se trabaja sobre la BD original
No necesita conexión con la BD original	Se necesita conexión constante con la DB
Requiere procesamiento inicial al enmascarar la DB	No requiere procesamiento inicial

No requiere procesamiento por consulta	Se realizar el proceso de enmascarado por cada consulta
Las actualizaciones de la DB original no se ven reflejados en la copia	Los datos siempre están actualizados
No requiere configuración inicial	Se deben configurar los permisos del <i>proxy</i>
Todos los usuarios acceden a los datos enmascarados	Pueden definirse permisos por rol permitiendo acceder a datos sensibles dependiendo del caso
Seguro para lectura/escritura	Programación adicional para evitar rescritura de datos enmascarados
La DB original no es vulnerable	Seguridad del <i>proxy</i> necesaria para evitar el acceso a la DB

Enmascaramiento de datos y deductibilidad

Puede resultar intuitivo pensar que eliminar datos claramente sensibles como nombres, números de identidad, teléfonos y direcciones es suficiente para considerar que se ha logrado el anonimato. Sin embargo, existen otros puntos a tener en cuenta si se quiere mantener la calidad irreversible del enmascarado.

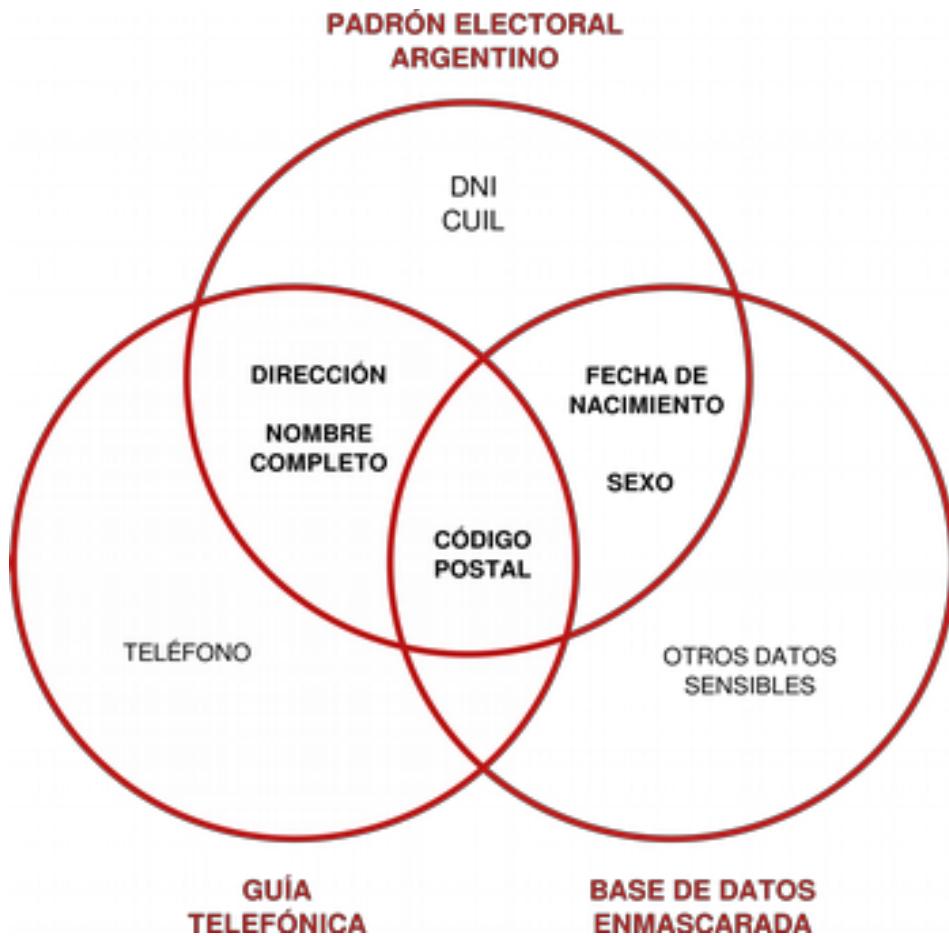
Aunque una base de datos haya sido despojada de los datos sensibles, en ocasiones es posible inferir esos datos a partir de la **combinación** del resto de los datos no enmascarados. Por ejemplo, un estudio realizado sobre la

población de Estados Unidos encontró que el 87% de la misma puede ser inequívocamente identificada utilizando solamente su código postal, sexo y fecha de nacimiento (Sweeney, 2002).

También es importante estar al tanto de otras bases de datos que existan de manera pública o de fácil acceso ya que mapeando entre diferentes fuentes es posible reconstruir la identidad a partir de una serie de datos específicos. El riesgo que implica dejar cada dato visible debe ser calculado con anterioridad teniendo en cuenta el contexto de uso y la existencia de bases de datos tanto gratuitas como pagas.

En el siguiente diagrama de Venn puede verse la relación entre una base de datos hipotética enmascarada, el padrón electoral argentino y la guía telefónica. Aunque no todas estas contienen los mismos datos, existen campos que se comparten entre dos o más de ellas, como la dirección, nombre, fecha de nacimiento, sexo y código postal. Dependiendo de la cantidad de población en un código postal específico y la cantidad de personas con una misma fecha de nacimiento los pocos datos disponibles podrían ser suficientes para realizar una inferencia e identificar correctamente a un gran número de personas en la base de datos enmascarada, perdiendo así la privacidad.

Por último, en los casos de enmascarado dinámico es importante comprobar que no solo las consultas individuales no contengan datos sensibles sino que múltiples consultas de distinta naturaleza no permitan inferir relaciones y por lo tanto generar datos relevantes.



1. Ilustración: Relación entre una base de datos hipotética enmascarada, el padrón electoral argentino y la guía telefónica

Fuentes en línea consultadas por la autora:

- <http://www.iri.com/blog/category/data-protection/>
- <http://mask-me.net/datamaskingwiki/wiki/26/data-masking-definition>
- <https://www.iri.com/solutions/data-masking/static-data-masking/overview>
- <https://www.iri.com/solutions/data-masking/static-data-masking/overview>
- <https://dataprivacylab.org/dataprivacy/projects/kanonymity/kanonymity.pdf>
- <https://dataprivacylab.org/dataprivacy/projects/kanonymity/kanonymity2.pdf>
- <https://dataprivacylab.org/dataprivacy/projects/kanonymity/paper4.pdf>
- <https://www.imperva.com/blog/2017/07/static-versus-dynamic-data-masking/>
- <https://securosis.com/blog/the-five-laws-of-data-masking>

GENERACIÓN DE DATOS DE PRUEBA

Andrea Navarro Moreno

Sobre la generación de datos

A menudo durante el desarrollo o prueba de un software es necesaria la utilización de datos para probar su funcionamiento, tanto en casos normales como anormales, para detectar errores y/o comportamientos anómalos. Al proceso de creación de estos datos de prueba se lo denomina *generación de datos de prueba*.

Aunque en ocasiones sea posible utilizar datos reales de una aplicación en fase de producción, existen varias razones por las cuales se considera una mejor opción generar los datos de manera artificial, siendo la principal de ellas, la **privacidad y seguridad de la información**. Otra de las razones es poder probar el comportamiento de la aplicación para **casos inusuales** que no se encuentren habitualmente durante la ejecución normal del programa; y

finalmente, la realización de **pruebas de estrés**, donde se evalúa la capacidad del sistema para funcionar utilizando una gran cantidad de datos.

Dependiendo de la complejidad y cantidad de los datos de prueba necesitados, esta operación puede ser muy costosa tanto en tiempo como en recursos. Es importante notar que aunque los datos sean generados dentro de los parámetros deseados, estos pueden no ser representativos de los datos que se generarían con la utilización normal de la aplicación, ya sea por una limitación del software utilizado para generarlos como por un error en su diseño.

Aspectos a tener en cuenta en la elección del generador

Variedad de tipos de datos

La mayoría de estos programas ofrecen una selección de tipos de datos listos para generar, con opciones de configuración. Un abanico de opciones amplio, disminuye drásticamente el tiempo de configuración necesario para la generación de los datos. Además de contar con los tipos básicos como texto, número, fecha, hora, UID, entre otros, resultan útiles, tipos más específicos como nombres, direcciones, teléfonos, códigos postales, números de tarjeta de crédito o direcciones IP.

Para que estos datos sean más parecidos a los reales, debe ser posible la parametrización de los tipos (por ejemplo, la generación de fechas debería tener una fecha de inicio y de fin como así también opciones de formato; los

números deberían tener un rango, y los textos, opciones de cantidad de palabras y párrafos).

Creación de tipos personalizados

En el caso que las opciones de tipos brindadas no sean suficientes para lograr el realismo necesario en los datos, debe ser posible la creación de tipos propios, pudiendo obtener los valores probables de una lista, archivo o base de datos o través de una función o patrón de expresiones regulares.

Correlación entre campos

En el caso de que en cada línea de datos generados unos dependan del valor de otros, el generador debe poder recrear esta correlación. Si, por ejemplo, existe un campo que contiene el país y otro que contiene la ciudad, el programa debe permitir, en primer lugar, la selección del país y una selección de ciudad que excluya todas las ciudades que no correspondan.

Distribución de los datos

Normalmente los datos se generan de manera pseudoaleatoria lo que es podría ser útil para generar diversidad en los datos, pero puede no siempre ser realista. Si se está generando, por ejemplo, un conjunto de usuarios con su fecha de nacimiento, una distribución normal sería más apropiada que una aleatoria. Del mismo modo en un software de comercio electrónico (*e-commerce*) sería más realista un incremento de ventas en ciertas fechas claves y una caída en otras.

Tipos de salidas

Aunque la mayoría de los generadores de datos lo hacen en archivos CSV y sentencias SQL, algunos permiten crear adicionalmente, archivos de configuración con XML, y matrices en diferentes lenguajes de programación. En algunos casos puede ser deseable la generación en forma de objetos programables.

Análisis de herramientas de código abierto

Spawner: generador de datos de prueba liberado bajo licencia GPLv2 y escrito con *Lazarus* v1.0.8 sobre *FreePascal* v2.6.2. Su código puede encontrarse en *SourceForge*¹³ y este no ha sido actualizado desde 2016. Puede descargarse también el ejecutable de la aplicación.

Permite la creación y configuración de campos que serán utilizados para la generación de los datos de prueba, pudiendo modificar el nombre y orden de los mismos.

Cuenta con 25 tipo de datos predefinidos que pueden ser configurados, incluyendo una lista de nombres y apellidos para la generación de datos asociados a personas, y tipos asociados a direcciones IP.

Permite también la creación de datos a partir de una lista definida en el programa o a partir de un archivo de texto, sin embargo no permite la

13 <http://spawner.sourceforge.net/>

creación de nuevos tipos de datos, como así tampoco la correlación entre campos por lo que cada dato se generará de manera independiente.

La selección de cada dato se hace de forma pseudoaleatoria, sin posibilidad de crear otro tipo de distribución de los mismos.

Los datos generados pueden ser exportados en formato de texto pudiendo seleccionar los caracteres delimitadores, y en formato SQL, seleccionando el nombre de la tabla y pudiendo personalizar diferentes características de la consulta.

Existe una opción para insertar los datos en una tabla MySQL® de manera directa que al momento de elaborar el presente documento, no se encuentra en funcionamiento, y el proyecto no cuenta con soporte.

GenerateData: herramienta de código abierto, escrita en JavaScript, PHP y MySQL®, que permite la creación de datos de prueba. El código se encuentra disponible bajo licencia GPL v3¹⁴. Una *demo* de la aplicación puede utilizarse en la página oficial del proyecto¹⁵. El mismo está activo y puede encontrarse extensiva documentación en su página oficial¹⁶.

Tanto el uso del sistema, como los procesos de instalación y configuración respectivos, se realizan mediante una interfaz Web.

14 <https://github.com/benkeen/generatedata>

15 <https://www.generatedata.com/>

16 <http://benkeen.github.io/generatedata/developer.html>

Tiene una mayor variedad de datos que otros sistemas similares, contando con más de 30 tipos diferentes, y la característica de poseer un código extensible que permite la creación de tipos de datos propios, y la configuración de opciones de salida.

Cada uno de los tipos de datos definidos funcionan como complementos en sí mismos, y pueden contener opciones de formato, rango y configuración que condicionen la generación del valor. También es posible crear correlación entre campos. En este sentido, el sistema ya cuenta con la opción de correlación entre país, región, código postal, entre otros, pudiendo extenderse a otros datos.

Estos tipos pueden tener un componente JS¹⁷ opcional, que permite validaciones de los datos del lado del cliente, y manipulaciones del DOM, pudiendo controlar la carga y guardado de datos de manera dinámica.

Cada fila de datos creada es independiente de la anterior por lo que no es posible generar comportamientos estadísticos en la distribución de datos, a menos que se configure la función de generación con una rutina que pueda simularlo.

Por defecto, *GenerateData* incluye formatos de salida abiertos como CSV, HTML, XML, JSON y formatos de diferentes lenguajes entre los que se encuentran JavaScript, Perl, PHP, Ruby y SQL, siendo estas opciones

17 JavaScript

configurables, y permitiendo agregar opciones adicionales al adicionar nuevos tipos de exportación, incluyendo validaciones con *JavaScript* para asegurar que el contenido generado cumpla con el formato esperado.

Faker: biblioteca PHP liberada bajo licencia MIT que permite la generación de datos de prueba. Puede instalarse mediante *composer*¹⁸.

Faker funciona bajo un sistema de clases, denominadas *proveedores*, que se encargan de generar tipos de datos específicos. Esto permite el agregado de nuevos tipos a la biblioteca creando nuevas clases que extiendan de la clase base de proveedores y creando nuevos métodos de generación de datos. Por defecto, cuenta con 19 proveedores de diversas variedades, capaces de generar datos de personas, fechas, direcciones IP, imágenes, textos estilo *loren ipsum*, códigos de barras, etc.

A diferencia de otros programas generadores de datos, este permite establecer una opción para que un valor no se repita en el conjunto creado. También permite la presencia de un valor por defecto o nulo a la hora de generar el dato, con la posibilidad de especificar la probabilidad de ocurrencia.

Otra característica de *Faker* es el método *seed* que al ser invocado con un número, generará siempre el mismo conjunto de datos al ejecutar el programa. Esto puede ser de utilidad cuando se utilizan datos de prueba de un sistema que los modifica ya que permite partir siempre desde el mismo estado inicial.

18 <https://getcomposer.org/>

No cuenta con formatos de salidas preestablecidos. El contenido se recupera accediendo a las propiedades del generador. Para generar los datos en un formato específico es necesario combinarlo con un generador de dicho formato en PHP.

Cuenta también con una herramienta de linea de comandos para la generación rápida de datos sencillos sin necesidad de programar.

Generación de datos aleatorios para MySQL®

Si el realismo de los datos no es importante para la prueba a realizar –como es en el caso de una prueba de estrés o una prueba de visualización en una etapa temprana del desarrollo–, entonces puede ser útil la generación sencilla y rápida de los datos.

Generación de números

En el caso de los campos numéricos es posible utilizar la función RAND() de MySQL® que da como resultado un número aleatorio entre 0 y 1 y las funciones CEIL(), FLOOR() y ROUND() para lograr el formato de número deseado.

```
SELECT RAND();          /* Pseudoaleatorio entre 0 y 1 */
SELECT FLOOR(RAND() * 10);    /* Pseudoaleatorio entero entre 0 y 10 */
SELECT CEIL(5 + RAND() * 10);  /* Pseudoaleatorio entero entre 5 y 10 */
SELECT ROUND(RAND() * 10 + 5, 2); /* Pseudoaleatorio entre 5.0 y 10.0 */
```

Es importante notar que la función RAND() permite el paso de un valor numérico como semilla, de manera que puede diseñarse un *script* que de ser necesario, retorne los mismos datos cada vez que se genere.

Generación de fechas u horas

Para los campos de fecha u hora puede seguirse un procedimiento similar sumando o restando una cantidad pseudoaleatoria de días, meses, años, horas, minutos o segundos a la fecha actual.

```
/*Se le suma a la fecha actual un número aleatorio de días no superior a 10*/
SELECT NOW() + INTERVAL FLOOR(RAND() * 10) DAY;

/*Se le resta a la fecha actual un número aleatorio de meses no superior a 5*/
SELECT NOW() - INTERVAL FLOOR(RAND() * 5) MONTH;

/*Se le suma a la fecha actual un número aleatorio de años no superior a 3*/
SELECT NOW() + INTERVAL FLOOR(RAND() * 3) YEAR;

/*Se le resta a la hora actual un número aleatorio de horas no superior a 6*/
SELECT NOW() - INTERVAL FLOOR(RAND() * 6) HOUR;

/*Se le suma a la hora actual un número aleatorio de minutos no superior a 30*/
SELECT NOW() + INTERVAL FLOOR(RAND() * 30) MINUTE;

/*Se le resta a la hora actual un número aleatorio de segundos no superior a 8*/
SELECT NOW() - INTERVAL FLOOR(RAND() * 8) SECOND;
```

Generación de textos

Si es necesario llenar un campo de texto pero no es necesario que este tenga sentido, puede utilizarse la función UUID() que genera una cadena UTF-8 de cinco números hexadecimales, y la función LEFT() para obtener la cantidad de caracteres deseados.

```
SELECT LEFT(UUID(), 10);    /*Cadena de 10 caracteres*/
```

Si se cuenta con una base de datos que contenga información del tipo que se necesita, es posible obtener un valor aleatorio utilizando la función RAND() para generar un ordenamiento pseudoaleatorio, seleccionando solamente el primer valor.

```
/*Selecciona un apellido aleatorio de la tabla personas*/
SELECT      apellido
FROM        personas
ORDER BY    RAND()
LIMIT       1;
```

Automatización

Combinando las opciones anteriores, es posible generar un *script* que rellene una tabla. En el siguiente ejemplo puede verse la aplicación a una tabla de usuarios, con los siguientes campos:

Field	Type
usuarioId	int(11)
nombre	varchar(30)
apellido	varchar(30)
fechaRegistro	date
tipoUsuario	int(1)

← Valores posibles: 0, 1 o 2.

```
INSERT INTO  usuarios
            (nombre, apellido, fechaRegistro, tipoUsuario)
SELECT
    (SELECT nombre FROM personas ORDER BY RAND() LIMIT 1),
    (SELECT apellido FROM personas ORDER BY RAND() LIMIT 1),
    (SELECT NOW() - INTERVAL FLOOR(RAND() * 360) DAY),
    (SELECT CEIL(RAND()*3));
```

Esto generará un registro cada vez que se ejecute.

Para crear varios registros en una sola ejecución es necesario crear un procedimiento. Este procedimiento (al que se denominará “*generar*”) contendrá una variable pasada por parámetro que indicará cuantas veces deberá ejecutarse el INSERT.

```
DELIMITER //
```

```

/* Crear procedimiento */
CREATE PROCEDURE generar (in cant int)
BEGIN
    DECLARE i int DEFAULT 0; /* Declarar variable */

    WHILE i < cant DO
        /* Script de generación */
        INSERT INTO usuarios
        (nombre, apellido, fechaRegistro, tipoUsuario)
        SELECT
            (SELECT nombre FROM personas ORDER BY RAND() LIMIT 1),
            (SELECT apellido FROM personas ORDER BY RAND() LIMIT 1),
            (SELECT NOW() - INTERVAL FLOOR(RAND() * 360) DAY),
            (SELECT CEIL(RAND()*3));

        SET i = i + 1; /* Incrementar variable */
    END WHILE;
END //;

DELIMITER ;

```

Una vez creado el procedimiento puede ser llamado mediante la instrucción CALL:

```

/* Llamar procedimiento */
CALL generar (100);

```

Documentación adicional:

MySQL® Reference Manual <<https://dev.mysql.com/doc/>>

PARTE III: 2. DESARROLLO WEB

GUÍA DE PROCEDIMIENTO PARA LA GENERACIÓN DE HOJAS DE ESTILO ESCALABLES¹⁹

Josefina Monsegur

Para mantener coherencia lógica y escalabilidad en el lenguaje CSS, es necesario tener en cuenta dos factores:

1. El orden convenido
2. El principio de simplicidad (y legibilidad del código)

El cumplimiento de estos factores puede verse reflejado en el rendimiento del código, ya que, entre otras cuestiones, permite la reutilización de las hojas de estilo en aplicaciones de gran tamaño y la colaboración de estas entre programas.

¹⁹ Se entiende por escalabilidad a la capacidad que un sistema informático posee, para incrementar sus capacidades sin detrimento de su rendimiento, comprensión y legibilidad.

El código CSS es por defecto global. Esto implica que el código definido en una clase puede afectar a otra a partir de la herencia.

Dependencia con el marcado HTML

En términos lógicos, un correcto CSS depende también de un correcto marcado de HTML. Para lograr coherencia entre ambos, no se los puede trabajar de forma aislada.

El HTML debe definir solo la estructura del contenido (por ejemplo, debe establecer que un elemento es un párrafo y no un título, o que un archivo multimedia es una imagen y no un vídeo), pero no puede limitar la forma en la que dicho contenido se presenta (por ejemplo, no puede especificar que la fuente de un párrafo deberá verse con mayor o menor tamaño que la fuente de un título). La manera en la que el contenido se presenta, debe definirse mediante CSS, en la hoja de estilos.

En este sentido, el HTML tiene una dependencia directa con la hoja de estilos, para precisar el modo en el que sus elementos son presentados, pero de esto, no puede depender su estructura (por ejemplo, si se requiere que un texto sea más pequeño que otro, no puede decidirse tratarlo como párrafo, si estructuralmente es un título).

Cuando el marcado HTML está correctamente estructurado no debería ser necesaria su edición asidua. Sin embargo, la hoja de estilos sí estaría sometida a modificaciones recurrentes. Esto es porque el HTML depende del CSS para disponer la forma de visualización de los elementos en pantalla, y por lo

tanto, lo que se pretende es que las modificaciones de visualización necesarias, afecten solo a las hojas de estilo, y no, a los archivos de marcado.

Para lograr conservar coherencia lógica en esta relación de dependencia, deben declararse todas las clases que sean necesarias, en el HTML, dejando pocos o casi ningún elemento sin su clase (incluso las etiquetas de párrafo, de título y demás etiquetas globales).

Nombramiento de clases y marcado

- Evitar nombres que denominen colores o determinen dónde está un objeto, puesto que un objeto que hoy se muestra en un lugar determinado y con un color determinado, podría cambiar y por lo tanto, dejar a la clase con un nombre huérfano o inconsistente. Por ejemplo, utilizar `--color-primario` en vez de `--color-rojo`.
- Utilizar siempre minúsculas. El lenguaje CSS distingue mayúsculas y minúsculas.
- Separar las palabras usando guiones, para hacerlas más legibles, `--color-primario` es más legible que `--colorprimario`.
- Evitar la definición de estilos por identificador (atributo *id*), ya que no hace falta utilizarlos en CSS, pues no es un requisito técnico mostrar que se trata de un elemento único. Reservar el uso de identificadores para acceder a los elementos desde *JavaScript*, en donde esta característica es clave.

- El nombramiento de clases debe hacerse utilizando convenciones de nombres globales que permitan la reutilización de clases.
- Utilizar abreviaturas puede ahorrar el procesamiento de varios *bits*, y de repeticiones en la misma declaración de la clase. Esto puede aplicarse a márgenes, rellenos y colores. En el caso de colores se recomienda utilizar la versión hexadecimal corta. Por ejemplo `#fff` en vez de `#ffffff`.

Orden y legibilidad

Convención de ordenamiento

Elegir una convención y adherirse a ella consistentemente, podría ayudar a entender el código de forma rápida y facilitar el escaneo visual. Algunos ejemplos de estas convenciones podría ser escribir el código CSS de manera alfabética u ordenarlo dependiendo de las características de las propiedades.

Selectores múltiples

Otro elemento que podría ayudar a otorgar legibilidad, es la forma en que los selectores múltiples están escritos. Presentarlos en líneas diferentes y dejar un espacio entre la clase y las llaves, es otra característica que aumenta la claridad de la lectura.

```
btn,  
.btn-link {  
}
```

Longitud de las hojas de estilo

Las hojas de estilo de gran longitud podrían dificultar la lectura de las propiedades. Para simplificar el uso, acceso y lectura de las mismas, la hoja de estilo debe tener un tamaño manejable (un punto de referencia podría encontrarse en una longitud no superior a 300 líneas).

Si esta es demasiado larga, se puede considerar dividirla en varias hojas, manteniendo el árbol de archivos legible y coherente. Los nombres deben ser cortos y descriptivos para hacer que el árbol, también sea comprensible.

Se pueden separar las hojas de estilos creando un árbol con diferentes criterios:

- Resets (reset.css)
- Variables (variables.css, si es muy extenso puede dividirse en colores, tipografías, etc)
- Estilos de ayuda (ayuda.css)
- Estructuras de la pagina (estuctura.css)
- Componentes
 - navegacion (navegacion.css)
 - botones (botones.css)

Si se utilizan variables, podría ser práctico tenerlas en una sola hoja de estilos. Esto facilita la edición de las mismas.

No repetición

No repetir reglas (en inglés esto es conocido como «DRY», siglas de *Don't Repeat Yourself*). Evitar la repetición del código (redundancia) mejora el rendimiento de las hojas de estilo. Aunque en aplicaciones o programas pequeños esto es poco notable, esta regla se torna importante cuando el CSS comienza a tener varias hojas.

Uso de **important!**

Evitar los *important!* y los estilos *inline*. Son una clara señal de que el código CSS está fallando, que hay una sobre saturación de código y un abuso de especificación.

Existen excepciones en el uso de este recurso. Por ejemplo: al depurar el código en el inspector aplicando *important!* se puede visualizar un estilo previamente anulado por jerarquía, o para casos específicos de uso, como hojas de estilo de impresión, u hojas de estilo para usuarios con necesidades específicas (lectura izquierda derecha, impedimentos visuales, idiomas no legibles en tipografías pequeñas como el chino, coreano o japones).

Documentación del código

Una documentación larga podría tornarse difícil de mantener. Sin embargo, notas y descripciones cortas para explicar brevemente las dependencias, o el porqué de cuestiones menos habituales, como *hacks*, podrían facilitar las búsquedas y el entendimiento del trabajo.

VARIABLES NATIVAS EN CSS

Josefina Monsegur

Definición

Las *variables nativas* de CSS son valores personalizados que pueden ser usados en una hoja de estilos, se les puede asignar un nombre y pueden ser utilizadas cuantas veces se desee en las clases posteriormente declaradas.

Escritura y uso de variables nativas

En CSS una variable es cualquier propiedad en la cual el nombre comienza con dos guiones. Si se declaran las variables CSS dentro del selector `:root`, las variables tendrán un alcance global, ya que `:root` indica el nivel más alto del documento.

```
:root {  
    --color-primario: #000;  
}
```

Para referirse o utilizar la variable declarada, se utiliza la función `var()`, nativa de CSS. Esta variable declarada puede ser usada en cualquier lugar, y tantas veces como se desee.

```
.nav, .menu {  
    color: var(--color-primario);  
}
```

Las variables de CSS siguen las mismas reglas de la herencia de este lenguaje. En el ejemplo siguiente, la variable `--color-primario` es heredada por todos los elementos `div`:

```
div {  
    --color-primario: #000;  
}  
  
div.p {  
    color: var(--color-primario);  
}  
  
div.em {  
    color: var(--color-primario);  
}
```

Semejanzas y diferencias con otros preprocesadores

No existe impedimento técnico para que las variables de CSS nativas no sean utilizadas en proyectos con múltiples colaboradores o proyectos cuyos colaboradores abarcan múltiples roles.

Con las variables CSS se pueden generar los elementos atómicos de un sistema de diseño, como tipografías, paleta de colores, unidades, entre otros.

La finalidad de las variables nativas es semejante a la de los post-procesadores enlatados, e insumen menor cantidad de pasos en el proceso de desarrollo.

Al utilizar estas variables también, el resultado se ve inmediatamente reflejado, ya que son accesibles en tiempo de ejecución, evitando compilaciones o errores en las rutas.

A diferencia de otros procesadores de CSS, las variables nativas pueden ser utilizadas dentro del atributo de estilos de HTML

```
<html style="--color-primario: red">  
  
body {  
    color: var(--color-primario)  
}
```

Las variables funcionan en cascada, pudiendo definir una propiedad al principio de la hoja de estilos y siendo sobreescrita luego.

Un ejemplo de esto, puede ser el del uso de una grilla. Esta se puede definir y posteriormente, utilizando *media queries*²⁰, se puede sobreescibir para generar una grilla adaptable a los dispositivos móviles, modificando sus margenes y porcentajes.

Como estas variables son nativas, pueden ser utilizarlas con la función *calc()* de css y de ser necesario, ser sometidas a cálculos por medio de *media queries*.

```
.margen {  
    --titulo: calc(20px * 2);  
    font-size: var(--titulo);  
}
```

20 https://developer.mozilla.org/es/docs/CSS/Media_queries

Si bien estas variables pueden ser utilizadas sin necesidad de pre/post procesadores, pueden ser accesibles desde JavaScript, y modificar sus valores para generar, por ejemplo, diferentes temas con una misma hoja de estilos.

Para acceder a las variables de una hoja de estilos asociada en CSS (no es necesario que los estilos estén embebidos en el HTML) se pueden utilizar, en JavaScript, los métodos `getComputedStyle`^{21 22} y `getPropertyValue`²³:

```
var estilo = window.getComputedStyle(document.documentElement);
var valor = estilo.getPropertyValue('--color-primario');
```

Para establecer un nuevo valor a una variable de CSS (tampoco es necesario que los estilos estén embebidos en el HTML), se puede usar el método `setProperty` sobre la propiedad `style` de `documentElement`.

```
document.documentElement.style.setProperty('--my-color', 'pink');
```

Compatibilidad de navegadores

Estas variables están disponibles tanto para navegadores de escritorio como navegadores de dispositivos móviles, pero a la fecha, algunas versiones de estos navegadores, no cuentan con dicho soporte.

Una lista actualizada de compatibilidad con navegadores y versiones puede verse en el sitio Web del *Mozilla Developer Network*:

https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_variables#Browser_compatibility

21 <https://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSSview-getComputedStyle>

22 <https://developer.mozilla.org/es/docs/Web/API/Window/getComputedStyle>

23 <https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleDeclaration/getPropertyValue>

SISTEMAS DE DISEÑO

Josefina Monsegur

Un sistema de diseño es una colección de elementos que pueden ser reutilizables y que funcionan de forma conjunta para desarrollar aplicaciones. No es una colección de elementos aislados y tampoco una biblioteca. Un sistema de diseño debe incluir conceptos, documentación y guías para describir cómo estos funcionan juntos de determinada manera.

Elementos que componen un sistema de diseño

A continuación se da un detalle de los elementos necesarios, para armar un sistema de diseño.

Fundamentos

Son los elementos fundamentales del sistema de diseño, tales como la paleta de colores, tipografías, iconos, y espaciados. Algunos sistemas, también incluyen

la grilla o la estructura entre los átomos. Son elementos más abstractos y no tienen un uso en sí mismos.

Colores

La paleta de colores debe ser definida y delimitada. Para definirla se puede comenzar por establecer los colores que están asociados con la marca y cuáles son aquellos que más se destacan en el producto o en las maquetas de diseño. Una vez que se tiene en claro este listado, deben incorporarse los colores de suceso, alerta y errores, que generen una correcta armonía (siempre teniendo en cuenta, estándares y usabilidad). Se completa definiendo una escala de grises que pueda acompañar y ser el soporte de tipografías y contenedores o fondos.

Como acción final, se puede evaluar el contraste, y hacer las pruebas pertinentes para usuarios con impedimentos visuales, y así, ajustar los tonos necesarios.

Tipografías

En las tipografías se establece una jerarquía organizada y alineada para su uso. Debe incluir una guía clara de tamaños, pesos, color, altura de línea y máximo por linea. Es importante tener en cuenta cómo interactúa la tipografía en pesos y altura de línea con los tamaños de los iconos y los campos de los formularios.

Iconos

Los iconos deben generar un sistema congruente entre sí, que incluye la definición de tamaños de uso y posibles gráficos. La coherencia gráfica y del peso de líneas, puede ayudar a generar una coherencia general. Utilizar la misma geometría, los mismos ángulos (90° , 45° , 30°) y respetar las simetrías, son aspectos que pueden ayudar a que el diseño, se perciba como «un diseño cuidado».

Espaciados

Son las unidades de medida y la cadencia que se aplicará entre estas unidades. Por ejemplo, en unidades basadas en 4, los espaciados serán de 4, 8, 16, 32 y así sucesivamente.

Componentes

Los componentes son bloques de elementos que utilizan los fundamentos. Estos se unen para formar los elementos de una aplicación.

Definición de reglas de estilo

Se refiere a «los sí y los no» del uso de los componentes. Estas reglas deben ser documentadas y accesibles de la misma manera que los elementos. Se trata de guías, estándares de uso y reglas, para la creación o incorporación de nuevos elementos al propio sistema, cuando sea necesario.

Librería

Se refiere a la agrupación de los elementos de un sistema de diseño, con el fin de simplificar su uso.

Dependiendo del tamaño de la aplicación, se pueden emplear librerías con estructuras complejas, o bastará con un archivo, repositorio o carpeta compartida.

Dinamismo

Un sistema de diseño debe ser construido para que pueda ser reutilizable y que pueda ser adaptado y modificado. No es un sistema estático, puesto que irá cambiando y mejorando a medida que la aplicación o producto crezca, y presente nuevas necesidades y desafíos.

Convenciones de nombre, documentación y guías

Estos últimos detalles tienen igual peso que el sistema en sí mismo. De esto podría depender, en gran medida, la adaptación, uso y adopción del sistema.

Para las convenciones de nombre, se puede definir un prefijo de uso global, y luego determinar un nombre que lo suceda. Si el nombre que lo sucede no representa con detalle el elemento o componente pero da una clara idea de lo que contiene, no sufrirá modificaciones (requisito necesario para que un sistema de diseño sea escalable). Por ejemplo: *.sis-color_suceso* se prefiere a *.sis-color_verde*.

No olvidar incluir en las guías, qué es lo que se puede hacer y qué es lo que no se debe hacer, con cada elemento o componente. Documentar además, el proceso y las posibles formas de modificar el sistema de diseño, a fin de generar mayor independencia en el uso.

MAPAS DE VIAJE PARA LA IDENTIFICACIÓN DE PROBLEMAS

Josefina Monsegur

Un **mapa de usuario** es una forma de visualizar información con ciertas características. Según el tipo de información que se necesita visualizar, se pueden elegir diferentes modelos de gráficos que pueden ayudar a comprender mejor los posibles problemas o desafíos del producto.

Hay otros esquemas similares que pueden servir al mismo objetivo, como por ejemplo, mapas de experiencias, *blueprints*, modelos mentales o mapas espaciales.

Una **característica** particular del mapa de usuario, es que cuenta con una organización cronológica que permite relatar una experiencia a través de un proceso o actividad, marcando las acciones, los pensamientos y los sentimientos del usuario. El usuario es siempre el centro del mapa.

El proceso de armado del mapa en si mismo, podría ayudar a alinear a las personas que están trabajando en la misma área o en áreas similares, a comprender y conocer los problemas alrededor del producto.

Elementos

Entre los elementos del mapa de usuario, deben existir uno o varios productos, un usuario que esté en contacto con el producto, y una organización cronológica. Si el problema no cuenta con estos tres elementos, se debería elegir otro tipo de visualización.

El alcance

Para comenzar a armar el mapa de usuario es necesario identificar primero qué es aquello desconocido que se quiere entender. Una vez identificado, analizar los elementos, y corroborar que éstos, junto con el problema, pueden ser visualizados en este tipo de mapa.

Los datos

Analizar y documentar los datos cualitativos y cuantitativos de los que se dispone. Si no se dispone de estos datos, analizar cuáles son las posibles formas de obtenerlos antes de comenzar el proceso.

Los **datos cuantitativos** son todos aquellos datos que pueden contabilizarse. Ejemplo de ello son los datos relacionados a los diferentes tipos de interacciones del usuario a través del producto:

- por ejemplo, en una plataforma de comercio electrónico, la cantidad de usuarios en las etapas, comparadas con las etapas del embudo de conversión²⁴
- cantidad de *clicks* en el caso de que el producto esté en línea
- cantidad de usuarios que empiezan una compra
- cantidad de usuarios que terminan una compra

Los **datos cualitativos** son aquellos que pueden ser calificados pero no contabilizados. Estos datos podrían ser los provenientes de encuestas de satisfacción, preguntas informales, observaciones etnográficas, etc.

La organización

Es la línea de tiempo o narrativa con la que será representado el mapa y responde a preguntas tales como «¿cómo es el viaje del usuario a través del producto?», «¿dónde empieza?» y «¿dónde termina?».

El orden puede ser cronológico o no, y algunas etapas pueden ocurrir reiteradas veces, mientras que otras, solo una. Si se detecta que no hay orden cronológico posible, el mapa de viaje de usuario no sería la herramienta de visualización ideal.

²⁴ NdE: en la jerga del mundo de las ventas en línea, suele denominarse «embudo de conversión», a la sucesión de eventos transcurridos desde que se capta la atención del usuario en la plataforma de ventas, hasta que éste, concreta la compra.

Las personas

Una persona es una representación del tipo de usuario objetivo. A veces puede ser necesario un único tipo de persona, y otras, varios tipos para representar todos los posibles. Se trata de una generalización o prototipo de usuario determinado (o personajes ficticios), y no de un usuario real.

Las personas deben estar basadas, al igual que el resto de elementos, en los datos cuantitativos y cualitativos que hayan sido recolectados.

Fases de la experiencia

Se denomina fase de la experiencia a cada una de las etapas del viaje. Se debe hacer un listado de éstas, y organizarlo de manera cronológica.

Puntos de contacto

Se refiere a los lugares de interacción del usuario, en las distintas fases. Estos puntos pueden ser definidos al esclarecer qué es lo que el usuario está tratando de solucionar a través del producto, cuál es su objetivo final, y si hay un solo objetivo o varios objetivos.

Desencadenantes

Se entiende por desencadenante a los distintos estímulos que provocan la reacción del usuario. Los desencadenantes, por lo general, terminan en acciones. Por ejemplo, las vacaciones de verano podrían ser el desencadenante de tener el deseo de ir de viaje con la familia.

Actitudes

Se trata de tener en cuenta las emociones o estado anímico del usuario, en los puntos de acción específicos y a lo largo de la experiencia (atención, actitud, motivación, humor). Para esto, es necesario evaluar los datos cualitativos, y definir un perfil de persona (usuario) con el que se va a mapear el viaje, basado en los resultados de la investigación.

El armado

Una vez que están definidos y documentados todos los elementos se comienza a armar el mapa de viaje. Una forma de hacerlo es en papel. Para ello, se deberá contar con un papel extenso y con los elementos impresos.

1. Establecimiento de las etapas y los puntos de contacto

El mapa debe representar una línea de tiempo del viaje del usuario a través del producto, y marcar cada etapa trasversalmente. Comenzar por marcar las fases de la experiencia, y sobre éstas, los puntos de contacto.

2. Identificación de las conexiones

Si existe una conexión entre las etapas, establecer dichas conexiones y marcar el tipo de conexión existente.

Los *tipos de conexiones* pueden ser:

- *Conección directa*: una tarea lleva a la siguiente.

- *Conexión bidireccional*: una tarea lleva a la otra, pero el usuario puede volver atrás.
- *Conexión cíclica*: una tarea es finalizada pero vuelve a empezar.

El cuadro posterior indica brevemente la organización de los elementos:

Orden cronológico →			
Desencadenantes	Fase 1 frecuencia	Fase 2 frecuencia	Fase 3 frecuencia
Puntos de contacto			
Actitudes			

5. Tabla: Organización de los elementos en el mapa

3. Elección del formato

Una vez establecidos los elementos principales, se debe elegir un formato. El formato correcto de un mapa va a depender de la cantidad y dispersión del contenido resultante de los diversos elementos (como el número de puntos de interacción, la cantidad de interacciones y etapas resultantes), y en qué aspectos se desea poner énfasis (en los puntos de contacto, las interacciones o en las emociones).

4. Agregado de datos cuantitativos y cualitativos

En este momento las fases o las interacciones pueden ser enriquecidas con breves extractos de datos cualitativos o cuantitativos. Éstos, deben ser cortos

y relevantes, para no perder la visualización, que es lo mas importante del mapa.

5. Predicción

Por ultimo se debe mapear a la persona (o a cada tipo de persona) y a sus actitudes a lo largo del viaje, a fin de identificar cómo reaccionaría a un estímulo concreto, qué haría o cómo se sentiría.

Conclusión

Como se comentó al comienzo, el proceso de desarrollo del mapa, tanto como el mapa en sí mismo, puede aportar información relevante para alinear y estudiar posibles problemas con el producto, que podrían implicar cambios en las funcionalidades del programa. La incorporación de personas que trabajan en el producto, a lo largo de todas las etapas, podría contribuir al aporte de información relevante para la comprensión del problema. Una vez finalizado el mapa, puede someterse a debate entre todos los participantes, a fin de determinar áreas bloqueadas o aquellas que presentan un mayor problema.

PARTE III: 3. DESARROLLO MÓVIL

COUCHDB PARA EL MANEJO DE DATOS FUERA DE LÍNEA EN APLICACIONES BASADAS EN IONIC

Jéssica Sena

Definiciones previas

Aplicación móvil

Software diseñado y desarrollado para ser ejecutado en un dispositivo móvil y/o tableta.

Ionic

Ionic es un *framework* de código abierto para el desarrollo de aplicaciones móviles híbridas, basado en *AngularJS*²⁵ y *Apache Cordova*²⁶.

²⁵ AngularJS: <https://angularjs.org/>

²⁶ Apache Cordova: <https://cordova.apache.org/>

CouchDB

CouchDB es una base de datos de código abierto, de tipo NoSQL. Permite una arquitectura distribuida con sincronización entre las réplicas, es decir, se puede tener la información replicada en diferentes nodos, que pueden modificarla, y posteriormente sincronizarse entre ellos. Por otra parte, cada vez que se actualiza alguna información, se crea una nueva entrada en la base de datos, con su correspondiente identificador de versión, permitiendo la recuperación de datos a estados anteriores en caso de ser necesario.

Introducción

Uno de los factores a tener en cuenta durante el desarrollo de una aplicación móvil, es definir el comportamiento de esta ante la probable pérdida de conexión del dispositivo en el que la misma será ejecutada. Esto implica establecer su **comportamiento en modo offline (fuera de línea)**, con el objetivo de garantizar una mejor experiencia de usuario.

A fin de definir qué funcionalidades será necesario proteger ante pérdidas de conectividad, primero se deberá determinar el tipo de aplicación que será desarrollada.

Tipos de aplicaciones

Se podrán encontrar tres tipos.

- **Aplicación de lectura:** Aplicación que sólo consume y muestra datos al usuario. A su vez, esta información puede ser *dinámica*, en caso que varíe a través del tiempo, o *estática*, en caso contrario.

- **Aplicación de escritura:** Aplicación en la que el usuario genera datos e información.
- **Aplicación de lectoescritura:** Aplicación que implica los dos modos anteriores.

Manejo de los datos

Exceptuando el caso de una aplicación de lectura de datos estáticos (los cuales posiblemente se incorporen directamente en la aplicación), para todas las demás situaciones se deberá realizar un desarrollo específico para la gestión del funcionamiento fuera de línea, que básicamente implicará tener en cuenta dos aspectos:

1. **El almacenamiento de los datos de forma local:** Ante pérdida de conexión, será necesario tener almacenados directamente en el dispositivo, aquellos datos indispensables para el funcionamiento básico de la aplicación.
2. **La sincronización de los datos generados:** Ante pérdida de conexión, será necesario que los datos que se generen localmente se conserven, y sean sincronizados con el servidor, automáticamente al momento de recuperar la conexión.

Solución propuesta

Como solución para el almacenamiento de datos de forma local y la sincronización de los mismos, en aplicaciones basadas en *Ionic*, se propone el uso de *CouchDB*.

Para poder usar *CouchDB* en una aplicación desarrollada con *Ionic*, se necesita una aplicación *puente*. En este caso se emplea *PouchDB*.

PouchDB²⁷ es una implementación en *JavaScript* del *CouchDB sync* protocol²⁸ —tal como se indica en su documentación oficial— que es lo que permitirá mantener la sincronización entre los datos replicados en el dispositivo y la base de datos original.

Por tanto, esta combinación ofrece exactamente lo que se busca: la posibilidad de almacenamiento y uso de datos de forma local, y a la vez, la sincronización automática de los datos locales y remotos, siempre que sea necesario.

Funcionamiento de PouchDB

PouchDB utiliza *IndexedDB* por defecto (disponible en el *webview* en que se ejecuta una aplicación híbrida basada en *Ionic*) para almacenar los datos de forma local.

Los pasos para implementar *PouchDB* en el desarrollo de una aplicación móvil se describen a continuación.

27 PouchDB: <https://pouchdb.com/>

28 PouchDB, «Replication» [online]. Disponible en <https://pouchdb.com/guides/replication.html>

1. Crear una base de datos de tipo *PouchDB* en local:

```
var db = new PouchDB('<database-name>');
```

2. Asociarla a la base de datos remota a replicar en local:

```
db.sync('<remote-uri>', <options>);
```

Donde:

<database-name>	es el nombre de la base de datos local, definida en el paso 1
<remote-uri>	es la URI de la base de datos en remoto
<options>	es un diccionario con opciones de configuración opcionales

Algunas de las opciones disponibles para afinar más la configuración, son la adición de los *flag* “*live*” para añadir sincronización entre las bases de datos, y “*retry*”, para que en caso de fallo (por no haber conexión), se vuelva a intentar la sincronización de los datos hasta realizarla con éxito:

```
var options = {  
    live: true,  
    retry: true  
};
```

A partir de aquí se tendrán todos los datos necesarios en local.

Limitaciones

La cantidad de datos que se pueden almacenar estará limitada por la cantidad de datos que el *webview* en que se ejecute la aplicación, permita almacenar usando *IndexedDB* (normalmente será de unos 50MB).

En principio hoy día prácticamente todos los navegadores permiten el uso de *indexedDB* pero en caso de no ser así, se pueden instalar *adaptadores* para

usar *WebSQL*²⁹ (teniendo presente que actualmente se considera obsoleto), o incluso, en el caso de aplicaciones basadas en Ionic y Apache Cordova, existe la opción de usar *SQLite*³⁰ como adaptador, que no cuenta con la limitación de 50MB. Por motivos de rendimiento, el equipo de *PouchDB* recomienda en su página oficial, no hacerlo a no ser que sea estrictamente necesario³¹.

Aunque esta solución en general suele funcionar, para determinadas casuísticas el rendimiento puede no ser el esperado. Para aquellos casos en los que se trabaje con volúmenes de datos medios (pero aún lejos de los 50MB del máximo), para dispositivos de gama baja y media, con poca RAM y además poco espacio disponible, la sincronización inicial de los datos puede requerir demasiado tiempo, o incluso no finalizarse correctamente.

Conclusión

En general, y para la mayoría de aplicaciones del mercado en que el volumen de datos difícilmente supere los 50MB, *Ionic + PouchDB* puede ser la solución, sobre todo si existe una recogida constante de datos por parte del usuario, y uso de la aplicación, en entornos con poca o nula conectividad.

29 WebSQL: <https://www.w3.org/TR/webdatabase/>

30 SQLite: <https://www.sqlite.org/index.html>

31 PouchDB, «Adapters» [online]. Disponible en <https://pouchdb.com/adapters.html>

PARTE III: 4. GESTIÓN DE PROYECTOS

IMPLEMENTACIÓN DE SCRUM EN UNA EMPRESA DE DESARROLLO DE SOFTWARE

Milagros Mora

Definiciones previas

Scrum

Scrum es una estructura conceptual basada en los principios ágiles³² de desarrollo de Software para administrar procesos de fabricación de este tipo de productos.

Es un marco de trabajo que define al equipo de desarrollo y sus roles, las herramientas que se emplean, y las reglas que los relacionan para que en forma autogestionada y en permanente interacción con el cliente, se llegue a proporcionar Software funcionando en un periodo de tiempo no mayor a cuatro semanas.

32 <https://agilemanifesto.org/principles.html>

Desarrollo ágil

Cuando se habla de desarrollo ágil de Software se hace referencia a un paradigma de gestión de proyectos, que fomenta el permanente intercambio entre el equipo de trabajo y los usuarios del sistema. Tiene en cuenta y valora las necesidades del cliente, incluso si estas cambian a lo largo del proyecto.

Un grupo trabajando bajo esta modalidad, acepta y gestiona los cambios como parte de su rutina, realiza entregas periódicas, evalúa su desempeño, plantea mejoras y vuelve a repetir el proceso hasta que el proyecto haya terminado.

Empresa de desarrollo de Software

A los fines de este documento, por empresa de desarrollo de Software, se entenderá a toda aquella organización que tenga por objetivo principal, la comercialización de servicios de Ingeniería y/o construcción de programas informáticos, destinados a dar soporte a diferentes problemáticas, y cuyos proyectos, sean abarcados por un equipo de trabajo que no supere las diez personas por proyecto.

Introducción

Ken Schwaber y Jeff sutherland, los precursores de Scrum, lo definen como «*un marco de trabajo para entregar y mantener productos complejos*»³³.

33 <https://www.Scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100>

Scrum se emplea, aproximadamente desde finales de los años 90, por la industria del Software para crear, mantener o mejorar funcionalidades, programas embebidos, o servicios relacionados con la tecnología.

Fundamentos de *Scrum*

Scrum está basado en un proceso de control empírico, en el que se hace un análisis del proceso de trabajo, se recogen experiencias de los miembros del equipo en el mismo, o de trabajos similares que se han hecho con esta modalidad, con el objetivo de detectar progresos, desviaciones, viabilidad del producto, mejoras, ajustes, entre otros factores.

Establece que desarrollar un programa de manera incremental, asegura el control de riesgos, y según pruebas empíricas efectuadas por el equipo de investigación³⁴, la adaptación a los cambios, inspección y transparencia, garantizan dicho control.

Marco de trabajo

Equipo de Scrum

El equipo de *Scrum* está compuesto por el dueño del producto (en inglés, *product owner*), el equipo de desarrollo (en inglés, *development team*) y un maestro de *Scrum* (o *Scrum master*). Estos roles trabajan en forma coordinada y autogestionada con el fin de optimizar la productividad.

34 <https://www.scrumguides.org>

Dueño de producto

El dueño de producto es el responsable del programa ante todos los interesados en el desarrollo del mismo. Es una única persona que administra y gestiona la pila del producto y participa de las reuniones de cada sprint.

Esta persona debe conocer lo que se desea del producto para poder trasladar al equipo de desarrollo lo que se debe realizar para cumplir con las expectativas del mismo. Además es la persona encargada de comunicar a todos los interesados en el desarrollo del programa, cómo es el plan de trabajo y cómo serán las entregas.

Equipo de desarrollo

El equipo de desarrollo es un grupo de profesionales cuya cantidad de miembros debe oscilar entre cuatro y nueve personas. Tiene como obligación realizar una entrega al finalizar cada *sprint*. La misma, es el resultado de ir convirtiendo de forma iterativa, la pila del producto en una o varias funcionalidades disponibles para desplegar en el programa.

Para que este equipo sea autoorganizado se presume necesario que cada integrante posea unas cualidades determinadas, que le permitan lograr terminar las historias de usuario pactadas, en una iteración, y entender, que si bien cada persona del equipo se enfoca a una tarea según su especialidad, la responsabilidad del mínimo producto viable es de todos los miembros.

Scrum Master

Es la persona responsable de implementar las reglas de *Scrum* en la empresa encargada de desarrollar el Software. Se ocupa de asesorar y formar tanto al

equipo de desarrollo como al dueño del producto. Ayuda a entender cómo son las interacciones en un equipo de *Scrum*, a las personas involucradas en el proyecto.

El *Scrum Master* es un líder facilitador (Alaimo, 2014). Trabaja con el dueño del producto para encontrar la mejor manera de organizar la pila del producto, facilita la comunicación entre este y el equipo de desarrollo, revisa y valida la pila del producto. Por otra parte, asegura el cumplimiento y modera las distintas reuniones del *sprint*, ayuda a resolver conflictos y reducir los desvíos vinculados con los impedimentos del trabajo.

Eventos de Scrum

Sprint

El *sprint* es el periodo de tiempo cíclico, no mayor a un mes, en el que se trabaja para llegar a un producto que pueda entregarse. Se organiza en:

- una reunión de planificación (*sprint planning*) al comienzo del *sprint*;
- reuniones diarias (*daily Scrums*);
- trabajo de desarrollo diario;
- revisión del sprint (*sprint review*) y retrospectiva del *sprint*, ambas al final de cada ciclo.

Cada *sprint* tiene un objetivo que debe cumplirse dentro del período que dure el ciclo. Para llegar a ese objetivo, se realiza el trabajo de desarrollo en equipo.

Planificación del Sprint

La planificación es una jornada inicial de trabajo con una duración de hasta ocho horas, en la que se reúne todo el equipo de *Scrum* para resolver qué tareas hacer durante el *Sprint*.

Se toma de referencia la pila de producto, las últimas funcionalidades entregadas y el rendimiento obtenido en la interacción anterior para idear la estrategia a seguir.

De esta planificación surge el objetivo del sprint, consensuado entre los desarrolladores, el *Scrum master* y el dueño de producto. En esta instancia los desarrolladores deben ser capaces de informar cómo se va a trabajar para llegar al objetivo.

Scrum diario

Es una reunión interna diaria del equipo de desarrollo que dura quince minutos y que se realiza siempre a la misma hora y en el mismo lugar. En esta reunión, el equipo organiza su jornada laboral.

Cada miembro debe responder a las siguientes preguntas: ¿Qué hice ayer?, ¿Qué haré hoy?, ¿Qué impedimentos tuve para la realización de alguna funcionalidad? Las respuestas se colectivizan y, si así se requiriera, se pueden profundizar en una reunión posterior.

Revisión del Sprint

Es una reunión que se lleva a cabo al final del Sprint. Su duración no supera las cuatro horas. En ella se encuentran el equipo de desarrollo, el dueño del producto, el *Scrum master* y el resto de los actores. En ella, el dueño del producto informa cuáles elementos de la lista se han terminado y cuáles no. Los desarrolladores realizan una demostración del software y sus nuevas funcionalidades y cuentan los inconvenientes a los que se han enfrentado durante ese tiempo.

Se analizan los pasos a seguir para la próxima entrega, y se entrega la *pila de producto* revisada que será la base del próximo sprint.

Retrospectiva del Sprint

Se trata de un encuentro de no más de tres horas con el objetivo de analizar el comportamiento de las personas dentro del sprint. Se evalúa la productividad y las percepciones individuales en el marco del desarrollo. Se intenta establecer las cosas que se hicieron bien y las que requieren corrección. Se intenta obtener un plan de mejoras para ejecutar en el siguiente ciclo.

Artefactos de Scrum

Backlog del producto (pila de producto)

La *pila de producto* es una lista creada por el dueño del producto y enumera todas las funcionalidades, requisitos y elementos deseables del programa. El equipo de desarrollo es el encargado de apuntar detalles a esa lista, tales como características particulares de una funcionalidad, orden de desarrollo,

estimación de esfuerzo y valor que la funcionalidad tiene para el dueño de producto.

El estado de un elemento de esta lista puede variar según el cliente para el cual se esté trabajando, y la experiencia en el desarrollo del mismo, pero se considera que se terminó una tarea cuando cumple con las características consensuadas entre el equipo *Scrum* y dueño de producto.

Backlog del Sprint

La lista del sprint es la lista de funcionalidades que se eligen de la lista del producto para llevar adelante durante un sprint. A esta lista se le agrega tanto detalle como sea necesario para poder armar una estrategia de desarrollo.

Puede pasar que este mayor nivel de detalle haga que se deban generar nuevas funcionalidades. Cuando esto pasa, esa nueva funcionalidad que no había sido contemplada se agrega en la pila del producto.

Bajo esta dinámica se genera una retroalimentación entre ambos artefactos.

*

Aspectos a tener en cuenta para la implementación de Scrum en el ámbito de trabajo. Aproximación práctica.

Descargo de responsabilidad: debe considerarse que *Scrum* es un marco de trabajo y no una teoría científica que pueda someterse a prueba y refutación. El resultado que persigue es el de lograr un Software funcional, acorde a los requerimientos del usuario, optimizando para ello los recursos disponibles, y priorizando el aspecto humano. Como marco de trabajo solo se enfoca en la forma de gestionar el proyecto, no especificando un método de implantación. H. Kniberg sugiere que podría obtenerse dicho método a partir de experiencias previas. Sobre la base de lo antedicho, la autora expone sus conclusiones a partir de lo observado en experiencias profesionales previas, de las cuales se aportan datos para su consideración y discusión.

Según lo observado hasta el momento, el resultado de implementar *Scrum* depende de cada grupo de trabajo, sus prácticas, y de la evolución de las dinámicas laborales a lo largo del ciclo de desarrollo, por lo que no pueden alcanzarse resultados concluyentes.

Las siguientes conclusiones surgen de lo observado a partir de la implantación de *Scrum* en 3 (tres) empresas de desarrollo de *Software*, con un promedio de 7 (siete) integrantes por equipo trabajando *in situ* en 2 de ellas, y de forma remota en la empresa restante.

En las últimas páginas se exponen dos tablas con datos adicionales sobre las mismas, y las características particulares de cada equipo.

Consideraciones a ser tenidas en cuenta al inicio de la implementación

Para la implementación de *Scrum*, a partir de los casos observados, se concluye que al comienzo de la implantación podría considerarse que:

1. La organización entiende, comparte y practica todos o al menos seis de los doce principios del manifiesto ágil³⁵.
2. Se estudia el marco teórico de *Scrum* al inicio de la implementación, y todas las personas involucradas en los desarrollos de *software* conocen su funcionamiento.
3. Se cuenta con un grupo de personas que tengan las destrezas suficientes para llegar a un *software* de calidad (según los términos definidos), durante el tiempo establecido para la duración del *sprint*.
4. Se conocen exactamente cuáles son los requisitos del cliente, sus prioridades, y la complejidad de las tareas a realizar, a fin de documentar la pila de producto.

35 <https://agilemanifesto.org/iso/es/principles.html>

Observaciones adicionales: dado que la **documentación** de tareas puede depender de la práctica narrativa y de los datos que se deseen incluir en la misma, se podría considerar apuntar los siguientes atributos:

- un identificador de tarea
- un nombre de tarea
- una descripción
- y una estimación aproximada.

Si bien pueden presentarse en el formato que mejor se adapte a las necesidades del equipo, en los casos observados se emplearon *panillas de cálculo*, con los atributos descritos anteriormente.

Aspectos a considerarse durante la implementación

Una vez considerados los puntos previos, en el trabajo periódico con *Scrum*, se estima que deben considerarse los siguientes aspectos:

5. **La pila de producto es entendida y conocida por todos los miembros del equipo.** Este documento está a disposición para la consulta y manipulación en cualquier momento, y se encuentra concluido a la hora de comenzar con la planificación. De esta pila depende la lista del sprint.

6. **Se cumplen con los eventos** que propone *Scrum*. Se define una agenda en la cual se establecen dónde y cuándo se realizarán las reuniones.

Observaciones adicionales al punto 6: en la **planificación** se toma la pila del producto, se define un objetivo y una pila de *sprint*. Un objetivo se cumple si se terminan todas las tareas del *sprint*. En esta reunión inicial se deja en claro qué es lo que se considera como un ítem terminado. En el aspecto técnico, un ítem está terminado cuando las tareas de análisis, codificación y prueba han concluido, y se logra la funcionalidad en términos de software. Encontrar un lenguaje que acerque al equipo al cliente, podría ser útil a la hora de comunicar esta información.

Las **reuniones diarias del equipo de desarrollo** ayudan a mantener actualizada la lista de trabajo, y a mantener la comunicación entre las personas que integran el equipo, así como también, conocer sus perspectivas.

Es posible ayudarse de **herramientas visuales para seguir el progreso** de cada ítem, y de quién es la responsabilidad de avanzar en el mismo. Cuando se piensa en *Scrum* es habitual pensar en pósits sobre la pared, y si bien la guía de *Scrum* no dice nada al respecto, en la práctica profesional se observa que usar alguna variante de tablero *Kanban*³⁶ podría dar una perspectiva global de cómo se está haciendo el trabajo.

36 Kanban and Scrum making the most of both. Keniberg and Skarin. INFO Q. ISBN 978-0-557-13832-6, USA.

Cada pósit será una tarea a realizar que estará esperando ser asignada a un miembro. Se puede definir un símbolo gráfico para identificar a cada integrante del equipo (por ejemplo, figuras geométricas). Cuando la tarea sea asignada se dibujará en ella el símbolo gráfico correspondiente a la persona responsable de su ejecución, y el pósit irá avanzando en el tablero según los estados que se hayan definido, hasta alcanzar al estado de *terminado*.

Utilizar la **reunión de revisión** para mostrar *software* funcionando e informar lo que se ha realizado durante la iteración, y entregar la pila del producto revisada. Si en la planificación se estimó el tiempo que llevaría terminar una tarea, cuando se realice la reunión, incluir en la tarjeta de la tarea cuánto tiempo llevó realmente. Esta práctica dará información para calcular los desvíos y poder utilizar estos datos en la reunión de retrospectiva. Revisar el comportamiento general del equipo y plantear un plan de mejoras es el objetivo de realizar las reuniones de **retrospectiva** del *sprint*.

7. **Se utilizan métodos de estimación** de tareas como el método de la estimación por Póquer, su variante de *fibonacci* como se propone en «*Scrum and XP from the Trenches*» de H. Kniberg (InfoQ, 2007), o se estima sobre la base de los conocimientos previos que se conozcan en la organización haciéndolo de manera relativa.
8. **Se implementan prácticas de programación que se puedan combinar con *Scrum* y que ayuden a obtener la calidad deseada** como la estandarización de código, la programación (y

revisión de código) en pares, y probar en un entorno configurado para ello.

9. **Se comienza el siguiente *sprint* inmediatamente después de haber finalizado el anterior** incorporando las lecciones aprendidas.
10. **Se trabaja en equipo** y las personas involucradas son receptivas a los cambios, y a las críticas.

Datos adicionales

A fin de otorgar mayor objetividad a las observaciones precedentes, se aportan los siguientes datos cuantitativos, relativos a promedios generales para el total de empresas.

Dato contabilizado	Valor
(A) Cantidad de empresas en las que se implementó <i>Scrum</i>	3
(B) Cantidad de integrantes por equipo	7
(C) Años de experiencia de cada integrante	6.88
(D) Cantidad de integrantes con experiencia previa en <i>Scrum</i>	<3

Datos discriminados por empresa:

	Empresa 1	Empresa 2	Empresa 3
(B)	7	8	6
(C)	7.14	3	10.5
(D)	0	4	3
(E)	~2.25	~1	<1

(E) Período de tiempo implementando *Scrum* (en años)

PARTE III: 5. HARDWARE

MANTENIMIENTO PREVENTIVO Y CORRECTIVO DE IMPRESORAS LÁSER Y DE INYECCIÓN DE TINTA

Lorena Santamaría Jiménez

Definiciones previas

Mantenimiento preventivo

Mantenimiento que se efectúa para garantizar el correcto funcionamiento de un equipo, y prevenir futuras averías, realizándose una puesta a punto de los componentes más comunes.

Mantenimiento correctivo

Mantenimiento que se realiza cuando se detecta el funcionamiento incorrecto de la impresora, se verifica si la impresora tiene algún tipo de fallo, sea mecánico, estructural o de impresión y se realiza la reparación de la misma.

Impresora

Equipo periférico conectado al ordenador que sirve para realizar una copia de los datos almacenados en formato electrónico a papel u otro tipo de material dependiendo de las características de la misma.

Avería

Se puede entender como cualquier anomalía o desperfecto que afecte al correcto funcionamiento de la impresora.

Tipos de impresoras

Se abarcarán dos tipos de impresoras, de inyección de tinta y láser. Se explicará de forma general su funcionamiento, y cómo realizar un mantenimiento preventivo de ambos tipos de impresoras. Por otro lado, se darán ejemplos de las averías más comunes para su posterior reparación.

Impresora de inyección de tinta

Una impresora de inyección funciona básicamente espolvoreando tinta sobre el papel, tienen un cabezal de impresión que esparce la tinta sobre el este. Sobre dicho cabezal se encuentran los cartuchos que contienen la tinta, uno para la tinta negra y otro con los colores primarios, los cuáles se pueden encontrar en un mismo cartucho o en diferentes, dependiendo del modelo de impresora.

Impresora láser

Una impresora láser tiene un funcionamiento más complejo que la anterior. La impresora lleva un tóner con tinta y dicha tinta está formada por pigmentos convertidos en polvo. El proceso consiste en que un láser graba en un cilindro fotosensible el contenido que debe imprimir mediante una carga electrostática. Este cilindro pasa por el depósito del tóner, los pigmentos se le adhieren, y cuando llega el papel y entra en contacto con el cilindro, el polvo cae para que después un rodillo con calor fije del todo la tinta al papel.

Componentes principales

Se puede decir que las impresoras se averían debido al uso y desgaste de los componentes. El tipo de componente, varía de acuerdo al tipo de impresora. Y por ello, el tipo de avería, dependerá entonces, del tipo de impresora que se utilice.

Componentes de una impresora de inyección de tinta

Entre otros, los principales componentes que pueden averiarse en una impresora de inyección de tinta, son los siguientes:

- 1. Cartuchos de tinta**
- 2. Cabezales:** componente de la impresora que esparce la tinta. De forma más coloquial, es la pieza que imprime sobre el papel. Se puede encontrar dentro de los propios cartuchos o incorporado en la impresora.

3. **Correa:** se utiliza para montar el cabezal de impresión.
4. **Barra estabilizadora:** hace que el cabezal de impresión se mueva con un movimiento estable y controlado.
5. **Bandeja porta papel:** como mínimo se encuentra una bandeja llamada cassette donde se coloca el papel a imprimir. Algunas impresoras pueden tener 2 o más bandejas adicionales.
6. **Fuente de alimentación:** es la encargada de suministrar electricidad a la impresora.
7. **Placa base (tarjeta lógica):** encargada de la comunicación con el ordenador por puerto USB, por red o cualquier puerto de conexión.

Componentes de una impresora láser

Entre los principales componentes de una impresora láser que pueden averiarse, se encuentran:

1. **Rodillos o *pickup roller*:** Son las piezas de la impresora que sirven para coger o empujar el papel. Al ser de goma, sufren desgaste. Se encuentran tanto a la entrada del papel como a la salida.
2. **Fusor:** Es la pieza encargada, con el calor y la presión que ejerce, de fijar la imagen que es transferida del tóner al papel. La posición del mismo depende del modelo de impresora, pero siempre se encuentra después de que el papel pase por el tóner.

3. **Piñones y engranajes:** son los responsables de los movimientos de los rodillos, cilindros y varias partes “*movibles*” de las impresoras.
4. **Electroimán o solenoide:** Es el encargado de girar el *pickup roller* para que coja el papel.
5. **Unidad láser:** se encarga de dar luz al tóner para que se cargue de partículas de polvo que serán adheridas al papel.
6. Bandeja porta papel
7. Fuente de alimentación
8. Placa base (tarjeta lógica)

Averías frecuentes

Como se comentó anteriormente, el tipo de avería depende del tipo de impresora que se use. Se listan a continuación, las averías más frecuentes según el tipo de impresora.

Averías comunes en impresora de inyección de tinta

1. **Sequedad de cartuchos:** es una de las principales averías de las impresoras de inyección ya que la tinta se enfriá y tapa el inyector por lo que la impresora deja de imprimir.
2. **Obstrucción de cabezal:** si la impresora lleva mucho tiempo sin ser utilizada, el cabezal de la misma puede llegar a quedar atascado por lo que la impresión no es correcta. Para evitar la avería se recomienda

realizar una impresión cada cierto tiempo y no dejar la impresora parada más de mes y medio aproximadamente.

Para las averías anteriores, cada modelo de impresora suele tener su propio software de mantenimiento, el cual se recomienda utilizar, aunque dichos sistemas consumen gran cantidad de tinta.

Averías comunes en impresoras láser

1. **Desgaste de rodillos (o *pickup roller*):** dependiendo del tipo de hoja que se utilice y la cantidad de impresiones que lleve la máquina, es una de las averías que más se producen. El rodillo de entrada de papel se desgasta y hace que tenga problemas a la hora de coger el papel.
2. **Desgaste de fusor:** cuando la impresión empieza a tener distintos fallos como líneas horizontales, impresiones repetidas en la misma hoja, marcas o manchas repetidas, es señal de que el fusor empieza a estar defectuoso y se debe sustituir. Siempre antes de sustituir el fusor lo primero es descartar un fallo del consumible bien probando con otro o detener la impresión antes de que la hoja llegue al fusor.
3. **Desgaste de piñones y engranajes:** las impresoras están compuestas por distintos piñones y engranajes que con el paso del tiempo y debido al uso empiezan a desgastarse, y hacen que la impresora empiece a emitir ruido cuando imprime, ya que no realizan correctamente el juego que deberían. Conocer el piñón

correspondiente permitirá sustituirlo, o bien realizar pequeños ajustes para disminuir el ruido.

4. **Fallo del electroimán o solenoide:** cuando la impresora no coge papel y se descarta un fallo de rodillos, suele ser problema del electroimán. Se puede diferenciar dicho fallo ya que la impresora hace un ruido (similar a un “clic”), al intentar enganchar el papel, bastante continuo.
5. **Rotura de bandejas:** la mayoría de las roturas proceden por el mal uso que se da a las impresoras o por golpes. La solución a dichas roturas es el cambio de las mismas o bien hacer una reparación casera ya que son piezas que suelen ser caras para su reposición.
6. **Fallo de conexión con el ordenador:** si se encuentran problemas de conexión de la impresora y no hay comunicación se debe de revisar la placa lógica (*formatter*), ya que los puertos USB o de red pueden estar estropeados.

MANTENIMIENTO PREVENTIVO DE ORDENADORES

Lorena Santamaría Jiménez

Esta guía de mantenimiento está destinada a aquellas personas profesionales del área de sistemas, que deseen realizar las tareas de conservación física rutinarias de sus ordenadores personales (PC).

Serán abarcados los dos tipos de ordenadores habituales a nivel usuario: los ordenador portátiles y los de sobremesa. Se explicará de forma genérica el procedimiento para realizar un mantenimiento rutinario de ambos tipos.

Al hablar de **ordenador de sobremesa o escritorio**, se hará referencia a un computador personal diseñado para su instalación en un puesto fijo, bien sea en su configuración habitual (integrado por un monitor, una torre y dos periféricos tales como teclado y ratón), o bien en modelos que incorporan la torre junto con el monitor, e incluso, un teclado táctil.

Toda vez que un **ordenador portátil** sea mencionado, se estará haciendo referencia a aquel que puede ser transportado fácilmente por el usuario, debido al diseño compacto y a su peso, y que además, durante un periodo de tiempo limitado, pueda funcionar sin necesidad de estar conectado a la red eléctrica.

Componentes

Los componentes que se describirán a continuación, se suelen encontrar en ambos tipos de ordenadores, con la salvedad de un formato y/o tamaño distinto dependiendo del tipo de ordenador que se utilice. Solo se abarcarán aquellos en los cuales sea posible realizar un mantenimiento preventivo rutinario.

- **Placa Base:** tarjeta de circuito impreso a la que se acoplan todos los componentes del ordenador.
- **Microporcesador:** se encuentra integrado en un zócalo de la placa base. Se encarga de ejecutar todos los programas que se necesitan para el funcionamiento del mismo.
- **Disipador de temperatura de la Unidad Central de Procesamiento (CPU):** ayuda a eliminar el exceso de calor del microporcesador. Va unido a una pasta térmica que aumenta su eficacia.
- **Ventiladores:** ayudan a mantener refrigerado el ordenador, extrayendo el exceso de calor hacia el exterior.

- **Memoria RAM:** en ella se guardan instrucciones, programas o datos de forma temporal, para su posterior utilización.
- **Tarjeta Gráfica:** puede estar integrada dentro de la placa base o como tarjeta de expansión. Es el componente encargado de procesar los datos procedentes de la CPU y transformarlos en información que luego será mostrada en el monitor.
- **Disco Duro:** sistema de almacenamiento de datos en el que se almacena cualquier tipo de información digital.
- **Fuente de alimentación:** es el dispositivo que provee energía eléctrica a todos los componentes del ordenador. Convierte la corriente alterna en continua, que es la utilizada por componentes electrónicos.

Limpieza de hardware

La limpieza se efectúa con el equipo desconectado de la corriente eléctrica, siempre que el mismo no se encuentre en garantía, ya que la misma puede perderse al manipular los componentes.

Para realizar la limpieza de todos los componentes que se encuentran en el interior del ordenador y a fin de quitar el polvo acumulado en los mismos, se puede utilizar o bien un aspirador manual, o un bote de gas seco.

Limpieza de equipo de sobremesa

Una vez abierto se procede a desconectar cada uno de los componentes internos para poder realizar una correcta manipulación y limpieza. Si no se está familiarizado con dichos componentes, para poder volver a montarlos sin error una vez realizada la limpieza, una opción es realizar fotografías (o trazar un plano) de la ubicación de cada componente y del cableado.

Se quitará la **fuente de alimentación** desenchufando todo el cableado de la misma y retirándola del ordenador. Con el bote de gas seco se limpiará el ventilador además de utilizar una brocha para retirar el polvo sobrante. Una vez deje de salir polvo se realizará la misma operación por la parte de la fuente que deja ver el interior de la misma.

Todos los **cables conectados a la placa base** se quitarán, además de retirar tanto la **memoria RAM** como la **tarjeta gráfica** (o cualquier tarjeta externa) que esté insertada en las ranuras de la placa base. Se retirará el **ventilador** junto con el **disipador del microprocesador** para realizar una limpieza más profunda.

En ocasiones, la **tarjeta gráfica viene con un ventilador**. En dicho caso, se limpiará con el gas seco y con la ayuda de la brocha, para retirar todo el polvo acumulado. Las unidades de memoria RAM apenas acumulan polvo por lo que pasando la brocha sería suficiente para su limpieza.

Para los **ventiladores**, tanto los de la carcasa del ordenador como el del disipador del microprocesador, se realizará la misma operación que para la limpieza del ventilador de la tarjeta gráfica.

Tanto en el disipador como en el microprocesador se puede ver **pasta térmica**. Esta debe limpiarse y sustituirse por una nueva. Para su retirada, se puede utilizar una gasa y un poco de alcohol isopropílico (*propanol*), o algún *aerosol para limpieza de contactos de componentes electrónicos* que no deje residuos. Antes de volver a colocar pasta térmica, se utilizará gas seco para terminar de limpiar todo el polvo de la placa base. La brocha se utilizará para la limpieza de las ranuras donde se ubican la placa base o la tarjeta gráfica.

Verificado que todo está limpio, sin rastro de polvo y pelusas, se procede al montaje de todos los componentes y el primer paso es volver a echar pasta térmica en el disipador para ayudar al ventilador a disipar el calor acumulado. Se irán insertando todos los componentes y cables en el orden contrario al que se ha desmontado.

Un punto importante es realizar un **chequeo del disco duro**, ya que una avería común y que puede ocasionar la perdida de todos los datos es la rotura física del mismo. Existen herramientas libres³⁷ como *fsck* y *smartmontools*³⁸ para comprobar el estado del disco duro e intentar adelantarse a dicha rotura,

37 Laura Mora, 2012. Hackers & Developers Magazine Nro. 2. pp. 14-15:
<https://openlibra.com/es/book/download/hackers-developers-2>

38 <https://www.smartmontools.org/>

aunque en ocasiones no se puede predecir por lo que siempre se debe mantener actualizada una copia de respaldo de todos los datos.

Limpieza de ordenadores portátiles

La limpieza interior de los ordenadores portátiles es más compleja si no se tienen conocimientos informáticos o electrónicos, ya que tanto la apertura de las carcasa como el desmonte de los componentes, es menos directo.

Limpieza sin desmontaje: es una forma simple de limpieza para realizar un mantenimiento básico. Consiste en usar las ranuras que tiene la carcasa, bien por la parte de abajo o por los laterales y con la ayuda del gas seco realizar la limpieza sin desmontar ni la carcasa ni los componentes. Con la ayuda del tubo de extensión que viene en los botes, se introduce por las aperturas y se descarga el gas presionando la válvula de descargue por unos segundos. Se realizará la misma operación por cada una de las aberturas visibles.

Si se desea **desmontar el portátil**, deberá contarse con el plano del fabricante, dado que cada equipo tiene unas condiciones de montaje específicas. La limpieza se realiza de forma similar a la descrita en los ordenadores de sobremesa, puesto que si bien los componentes cambian en ubicación y tamaño, el procedimiento de limpieza continúa siendo el mismo.

PARTE III: 6. INTELIGENCIA ARTIFICIAL

INTRODUCCIÓN A LAS REDES NEURONALES APLICADAS AL APRENDIZAJE SUPERVISADO

Andrea Navarro Moreno

Definiciones previas

Red neuronal artificial

Las *redes neuronales artificiales* son un conjunto de unidades de procesamiento llamadas *neuronas artificiales* conectadas entre sí a través de conexiones ponderadas que permiten la transmisión de información.

Aprendizaje supervisado

Se refiere al proceso mediante el cuál un programa informático puede realizar predicciones, a partir de patrones que aprende continuamente con base en datos. Dicho en otras palabras, es la automatización del proceso de creación de modelos analíticos que permiten que un programa informático se adapte a situaciones nuevas (Ana Loyo, 2018).

Base biológica: la neurona

La neurona es una célula del sistema nervioso cuyo funcionamiento puede compararse con el de un interruptor que será activado si recibe un estímulo de entrada suficiente y de ser así, enviará un pulso de salida. Anatómicamente, se compone de las siguientes partes:

1. **Dendritas:** ramificaciones protoplasmáticas que sirven de canal de entrada al soma neuronal, para los estímulos provenientes de los axones de las neuronas aledañas.
2. **Soma neuronal:** es el cuerpo celular de la neurona en el que se encuentra el núcleo —sección central de la neurona donde se generan los estímulos—, rodeado por el citoplasma.
3. **Axón:** prolongación fibrosa cubierta por una vaina de mielina que sirve como canal de salida de los estímulos generados en el soma, realizando conexiones sinápticas con otras neuronas.

La transmisión de señales entre neuronas es un proceso que se inicia cuando las dendritas reciben un estímulo sináptico. Esta sinapsis puede ser química o eléctrica. La *sinapsis química* se produce por la liberación de neurotransmisores en mayor o menor medida, aumentando o disminuyendo la intensidad del estímulo en el núcleo, mientras que la *sinapsis eléctrica* se produce por un impulso nervioso. Cuando las señales recibidas se acumulan en el soma superando un cierto umbral, se produce un nuevo estímulo que viaja a través del axón y se transporta a las neuronas vecinas.

Neurona artificial

Las redes neuronales artificiales son una simplificación matemática del modelo neuronal biológico que consiste en un conjunto de unidades de procesamiento (neuronas artificiales) interconectadas entre sí (Kriesel, 2007).

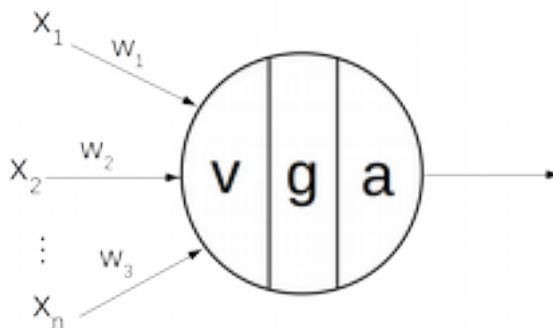
Esta versión simplificada cuenta con los siguientes componentes:

Sinapsis o señales de entrada: cada sinapsis cuenta con un dato de entrada que representa la señal enviada, y además, con un peso asociado. Este peso representa el nivel de estímulo que acompañará a cada señal.

Función de agregación o sumatorio: determina cómo la neurona acumulará las señales de entrada recibidas. Una aplicación común es realizar la suma de las señales de entrada ponderadas utilizando el peso asociado a cada sinapsis.

Función de activación: calcula la señal de salida de la neurona a partir del resultado de la función de propagación. La elección de la función de activación dependerá de los tipos de datos utilizados, la arquitectura de la red y el algoritmo de aprendizaje.

Señal de salida: el resultado de la función de activación que saldrá de la neurona.



2. Ilustración: Neurona artificial

Donde:

x =Señal de entrada

w =Peso

v =Función de agregación

g =Función de activación

a =Señal de salida

Primeras redes neuronales artificiales

McCulloch-Pitts MPN

Warren S. McCulloch y Walter Pitts introdujeron en 1943 el modelo de neurona simplificado MPN con el objetivo de entender la capacidad del cerebro para producir y entender patrones complejos (Kriesel, 2007) (Russell & Norvig, 2009). La función de agregación de esta neurona consiste en una suma de las señales de entrada ponderadas:

$$v = \sum_{i=1}^n x_i w_i$$

Los pesos de las conexiones son constantes por lo que no es posible el aprendizaje para este tipo de red. La función de activación es del tipo binaria, donde la función es activada si se supera un cierto umbral θ .

$$f(v) = \begin{cases} 1, & v \geq \theta \\ 0, & v < \theta \end{cases}$$

Una práctica común consiste en integrar el umbral dentro de la neurona como si se tratara de una entrada más x_0 cuya señal siempre es enviada $x_0=1$ y su peso es equivalente al negativo del valor del umbral $w_0=-\theta$. De esta manera la nueva función de agregación se modificaría para ser:

$$v = \sum_{i=0}^n x_i w_i$$

y la función de activación podría expresarse como:

$$f(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

Perceptrón

En 1949 fue introducida la regla de *Hebb*. Esta técnica teorizada por Donald Hebb permitía que las redes neuronales realizaran un aprendizaje a través del debilitamiento y refuerzo de las sinapsis (Russell & Norvig, 2009).

En 1962, Frank Rosenblatt la aplicó a un modelo MPN generando así lo que se conocería como perceptrón. A diferencia de la neurona MPC, el perceptrón permite un aprendizaje supervisado comparando la salida esperada con la salida real de la neurona y ajustando los pesos de las entradas hasta que el error disminuya o desaparezca. Sus funciones de activación más comunes son la función binaria ya utilizada por MPN, la función de signo y la función sigmoide.

Función de signo

$$f(v) = \begin{cases} 1, & v \geq 0 \\ -1, & v < 0 \end{cases}$$

Función sigmoide

$$f(v) = \frac{1}{1+e^{-x}}$$

Solo debe utilizarse en el caso que se espere siempre una salida negativa ya que devuelve valores entre 0 y 1.

para cada conexión:

establecer valor de peso aleatorio

mientras que no se identifiquen correctamente todos los ejemplos:

para cada ejemplo del conjunto de entrenamiento:

para cada neurona:

calcular señal de salida

calcular error

por cada conexión:

actualizar pesos

El **error** de la red para cada ejemplo presentado se calcula restando la salida esperada b y la señal de salida a : $\delta = b - a$

Luego el peso de cada conexión sináptica es modificado realizando un incremento o decremento resultante de la multiplicación del error de la red δ , la entrada correspondiente a esa conexión y un parámetro llamado **tasa de aprendizaje** η . La tasa de aprendizaje determina qué tanto se ajustan los pesos durante el entrenamiento. Una tasa muy alta generará un aprendizaje más rápido pero aumentará el riesgo de que la red nunca logre clasificar correctamente todos los ejemplos, mientras que una tasa muy pequeña generará un aprendizaje muy lento. La modificación de pesos se calcula entonces mediante la ecuación:

$$w_i = w_i + \delta \eta x_i$$

Una vez ajustados los pesos se presenta a la red un nuevo ejemplo del grupo de entrenamiento y se repite el proceso. Cuando un perceptrón ha sido capaz de clasificar correctamente todos los ejemplos brindados se dice que esta ha convergido.

No todo problema, sin embargo, puede ser representado por un perceptrón. El teorema de convergencia del perceptrón asegura el aprendizaje en un número infinito de pasos para problemas linealmente separables. Se dice que una función es linealmente separable si su salida puede ser separada o discriminada por una función que se componga de una combinación lineal de características.

Perceptrones multicapa

Los perceptrones multicapa (en inglés, *Feed Forward* —FF—), se caracterizan por tener sus neuronas divididas en capas, comenzando por la capa de entrada

por la que se recibirá la señal entrante de la red, y terminando con la capa de salida por donde la red enviará su señal de salida, pudiendo existir entre ellas, capas intermedias denominadas *capas ocultas*. Cuando una red de estas características tiene más de una capa oculta suele denominarse *Deep Feed forward (DFF)*. Las señales solo se mueven en una sola dirección desde la entrada hacia la salida, no permitiendo bucles o conexiones a neuronas de la misma capa o capas anteriores (Gurney) (Kriesel, 2007).

Dependiendo de las características del problema y el algoritmo de aprendizaje, las neuronas de estas redes pueden tener diferentes funciones de activación. Algunas de las más comunes son la función sigmoide, la tangente hiperbólica, ReLU y Leaky ReLU.

Retropropagación

Retropropagación (en inglés, *backpropagation*) es un algoritmo que permite el entrenamiento de redes *FF* que contengan una o más capas ocultas. A diferencia de los perceptrones simples es necesario determinar el nivel de injerencia o “responsabilidad” de cada neurona en el error total de la red, al calcular la modificación de pesos que sufrirá cada conexión de manera que la modificación de dicha sinapsis disminuya el error total de la red.

Al igual que en el algoritmo de aprendizaje anterior, se presentarán uno a uno los ejemplos de entrenamiento de la red, se calculará el error y se ajustarán los pesos. Este proceso se repetirá hasta que el error esté dentro del rango aceptado. Para cada ejemplo presentado, el algoritmo de retropropagación

ejecuta dos etapas: la **propagación hacia adelante** y la **propagación hacia atrás**.

La **propagación hacia adelante** consiste en presentar el ejemplo a la capa de entrada y calcular la salida de cada una de las neuronas; luego se calcula la salida de cada una de las neuronas de la capa oculta, y así sucesivamente, hasta obtener la salida de las neuronas de la capa de salida. La función de activación de las neuronas en esta red debe ser no lineal y tener una derivada calculable por lo que una función muy utilizada es la función sigmoide.

En la **propagación hacia atrás** se calcula el error comenzando desde la capa de salida. Se convierte este error en una derivada del mismo y se propaga este dato a la capa anterior, para calcular la derivada de error para cada una de las neuronas. Este proceso se continúa retrocediendo a través de las capas ocultas hasta llegar a la capa de entrada.

El error de una neurona de la capa de salida se calcula como la diferencia de la salida esperada y la salida de dicha neurona multiplicada por la derivada de la función de activación:

$$\delta = f'(v)(b-a)$$

Para las neuronas de capas ocultas, el error se calcula como la suma de la diferencia entre los errores de las neuronas de la capa siguiente y el peso de las conexiones con estas neuronas:

$$\delta = f'(v) \sum_k (\delta_k - w_k)$$

Luego de calcular los errores, se ajustan los pesos de las sinapsis utilizando la tasa de aprendizaje y la derivada de error de cada una de las neuronas:

$$W_{ji} = W_{ji} + \eta \delta_{ji} x_{ji}$$

Para evitar que el gradiente de error resulte en una convergencia de la red a un mínimo local es posible ingresar un nuevo parámetro a la red llamado inercia o *momentum* λ . Este valor tiene en cuenta la dirección de la modificación del peso, en una iteración de entrenamiento anterior t , lo que acelerará el aprendizaje:

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_{ji} x_{ji}(t+1) + \eta \delta_{ji} x_{ji}(t) \lambda$$

para cada conexión:

establecer valor de peso aleatorio

mientras que el valor del error no sea aceptable:

para cada ejemplo del conjunto de entrenamiento:

por cada capa partiendo de capa de entrada:

para cada neurona:

propagar señal de salida a siguiente capa

por cada capa partiendo de capa de salida:

para cada neurona:

propagar error capa anterior

por cada conexión:

actualizar pesos

Elección de arquitectura

Aunque las capas de entrada y salida tienen la cantidad de neuronas determinada por la naturaleza del problema, se necesita establecer la cantidad óptima de capas ocultas y neuronas en cada una de estas, para asegurar el correcto funcionamiento y rapidez de la red.

Si se está trabajando con problemas linealmente separables, entonces no será necesaria la utilización de capas ocultas. Para representar problemas que trabajen con conjuntos finitos de entrada a salida, una capa oculta será suficiente. Con dos o más capas ocultas es posible representar un problema con límites arbitrarios y funciones de activación racionales.

Para definir el número correcto de neuronas en capas ocultas pueden utilizarse reglas básicas que luego se irán ajustando en un proceso de prueba y error, verificando la tasa de error resultante en cada caso.

Siendo h la cantidad de neuronas en la capa oculta; i , la cantidad de neuronas en la capa de entrada; y o , la cantidad de neuronas en la capa de salida, algunas reglas de elección de neuronas ocultas comunes son las siguientes:

- La cantidad de neuronas en la capa oculta debe ser un número entre el número de neuronas en la capa de entrada y el número de neuronas en la capa de salida: $i \leq h \geq o \vee o \leq h \geq i$
- La cantidad de neuronas en la capa oculta debe ser equivalente a las dos terceras partes de la cantidad de neuronas en la capa de entrada más la cantidad de neuronas en la capa de salida:

$$h = \frac{2}{3}i + o$$

- La cantidad de neuronas en la capa oculta debe ser menor que el doble de neuronas en la capa de entrada: $h < 2 * i$

Una técnica más costosa pero más efectiva es la del **podado**. Esta técnica permite obtener la arquitectura de capa oculta más eficiente que resuelva el problema planteado, eliminando o no incluyendo neuronas innecesarias en la capa oculta. Existen dos técnicas de podado: **selectivo** e **incremental**.

En el **podado selectivo** se trabaja con redes ya entrenadas para resolver el planteo. Se evalúan los pesos sinápticos de las neuronas de capas ocultas y se procede a eliminar la que tenga el peso más pequeño, entendiendo que esta neurona es la que tiene un efecto menor en el resultado total de la red.

La red neuronal es entonces evaluada con esta nueva configuración y si la tasa de error sigue encontrándose dentro de un valor aceptable, el proceso se repite dando como resultado la red más pequeña, y por lo tanto más eficiente, capaz de resolver el problema dentro de la tolerancia de error aceptada.

```
hacer
    n=1
    si error <= error_aceptado:
        eliminar neurona n
    sino:
        n++
        si n > numero_de_neuronas:
            evaluar red sin neurona n
mientras que se hayan eliminado neuronas en este ciclo
```

El **podado incremental** sin embargo, se realiza sobre una red sin entrenar y que por lo tanto no tiene todavía pesos asociados. En esta técnica se inicia con la red más pequeña y de manera iterativa se van agregando neuronas a la capa oculta y evaluando la tasa de error de la red hasta que este sea aceptable o se llegue a un límite de neuronas predefinido. Al no contar con un red ya entrenada es necesario entrenar cada una de las nuevas configuraciones para

poder obtener la tasa de error. Esto aumenta el tiempo de procesamiento con respecto al podado selectivo.

```
n=1
hacer
    si cantidad_neuronas >= error_aceptado:
        salir con error
    crear red con n neuronas
    entrenar red
    n++
mientras error > error_aceptado
```

Redes Neuronales convolucionales (CNN)

Una CNN es una red FF cuyas capas ocultas se componen de una combinación de tres tipos diferentes: capas convolucionales, capas de agrupamiento y capas completamente conectadas. Estas redes fueron presentadas por LeCun en 1988, y han permitido la identificación automática de información espacial contenida dentro de imágenes, sin requerir una gran cantidad de experiencia por parte de la red (Nielsen, n.d.).

Capas convolucionales: son las primeras en colocarse. Su propósito es el de convolucionar³⁹ datos de entrada a través del uso de unidades de datos llamadas *kernel*⁴⁰ o filtros. Cada filtro se utiliza para que la red aprenda una característica diferente de los datos de entrada. Los filtros de las primeras capas convolucionales pueden utilizarse para detectar bordes, formas, etc. A medida que la arquitectura de la red se hace más compleja, las capas

39 Convolución de Funciones (Universidad de Almería):

<https://w3.ual.es/~vruiz/Docencia/Apuntes/Signals/Theory/index.html#x1-2100011>

40 Convolución de matrices (Universitat Politècnica de València):

<https://riunet.upv.es/bitstream/handle/10251/69639/4524-15767-1-PB.pdf>

convolucionales más profundas pueden detectar elementos más específicos como texturas y objetos. La operación de convolución consiste en utilizar una ventana para tomar una sección del valor que sea del mismo tamaño que el filtro a aplicar. Se realiza el producto entre cada valor de los datos dentro de la ventana con el del filtro, y finalmente se suman estos valores que darán como resultado los de la señal convolucionada. Luego la ventana se desplaza una cierta distancia, según un parámetro llamado zancada (*stride*), y se repite el proceso.

Es posible aplicar varios filtros a una misma entrada obteniendo de esta manera diferente información sobre la misma imagen o dato. Esto generará una matriz n-dimensional llamada *mapa de características*. Esta operación es combinada con la función de activación que generalmente es ReLU o similar.

Función de activación

Cuando dentro de la arquitectura de la red se utilizan dos capas convolucionales secuenciales es necesario utilizar una función de activación no lineal. Comúnmente se utiliza ReLU o alguna de sus variantes.

ReLU (Unidad lineal rectificada)

$$f(v) = \frac{1}{1+e^{-x}} \text{ o } f(v) = \begin{cases} v, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

ReLU es lineal para todos los valores positivos y/o para los valores negativos. Esta función tiene una característica desfavorable para la red, llamada *ReLU muerto*, que surge cuando una neurona mantiene valores negativos generando

que nunca pueda producirse una salida con un valor diferente de 0 . Cuando esto sucede, las neuronas afectadas no cumplen ninguna función en la red.

Leaky ReLU

Es una alternativa a ReLU que soluciona el problema de *ReLU muerto* modificando el valor fijo de 0 para valores negativos por una pequeña pendiente, determinado por la siguiente función:

$$f(v) = \begin{cases} v, & v \geq 0 \\ px, & v < 0 \end{cases}$$

donde p es una pendiente pequeña del orden de 0,01 .

Capas de agrupación

Estas capas permiten simplificar la dimensión de los datos eliminando parámetros y permitiendo un aprendizaje más rápido de la red. Para calcular la salida de cada una de estas neuronas es necesario recorrer nuevamente los valores de la señal de entrada con una ventana móvil y calcular los datos resultantes. Dos de los métodos más comunes, son el uso funciones de cálculo del valor mayor dentro de la ventana, y la función de cálculo del promedio.

Capa completamente conectada

Una vez colocadas las capas de convolución y agrupación, se colocan capas que convierten los valores de salida de estas en un vector. Estas capas se utilizan para permitir a la red aprender una función no lineal que represente una combinación de las características obtenidas de las capas anteriores.

Para esto es necesario convertir el mapa 3D creado por las capas convolucionales y de agrupación, en un vector unidimensional.

Arquitectura

La elección de combinaciones de estas capas para formar la arquitectura final de la red depende de la naturaleza del problema y de los resultados obtenidos en otras arquitecturas ya probadas para problemas similares, puesto que no existe un método para determinar la arquitectura exacta para la resolución correcta de un problema. Una aplicación común es encadenar repetidamente capas convolucionales seguidas por capas de agrupamiento y finalmente colocar una capa completamente conectada.

El diseño original de LeCun utilizado para el reconocimiento de dígitos, llamado LeNet, está compuesto por un encadenamiento de dos capas convolucionales y de agrupamiento con dos capas completamente conectadas. Debido a la aparición del GPU, arquitecturas más complejas se han vuelto computacionalmente practicables.

Redes Neuronales recurrentes (RNN)

Las redes neuronales recurrentes, a diferencia de las redes FF, permiten a las neuronas enviar señales hacia capas anteriores, neuronas de la misma capa o incluso a ellas mismas. Esto genera en la red un estado interno que causa un comportamiento temporal dinámico que las convierten en herramientas para inferir datos secuenciales como es el caso de las series temporales.

Hopfield Network

En 1982 Hopfield propuso un modelo de red neuronal completamente conectada donde las neuronas cumplen simultáneamente la función de capa de entrada, capa oculta y capa de salida. Las conexiones entre neuronas son bidireccionales, lo que significa que los pesos son simétricos.

Para lograr el aprendizaje se presentan a la red los patrones de tipo binario. Cada neurona, entonces, ejecuta su función de activación y envía la señal al resto de las neuronas, y a su vez recibe señales de todas ellas. El resultado obtenido es la salida de la red que puede compararse con el patrón de entrada para detectar el error y hacer los ajustes de pesos necesarios. Este proceso se repite sucesivamente provocando que la salida obtenida reduzca su variación con respecto a los patrones de entrada hasta llegar a un punto de estabilidad.

Una de las ventajas de este tipo de redes es que una vez entrenadas son muy tolerantes al ruido, lo que significa que son capaces de reconocer un patrón, incluso si se le presenta de manera parcial (Szandała, 2015).

Regla de aprendizaje de Hebb

En este algoritmo de aprendizaje, los pesos entre las conexiones de las neuronas son incrementados cuanto más señales positivas sean enviadas entre ellas. Utilizando la función de activación de signo, se calculan los valores de salida de las neuronas y se suma su multiplicación para obtener el nuevo valor de peso (Szandała, 2015), mediante la siguiente ecuación:

$$w_{ij} = \frac{1}{n} \sum_{k=1}^m x_i^k x_j^k$$

Donde:

$n = \text{número de neuronas}$

$m = \text{número de patrones}$

Regla de aprendizaje de Oja

Este método tiene como objetivo mantener normalizados los pesos de la red a fin de evitar problemas de estabilidad. Para hacerlo utiliza un valor llamado *factor V* (Szandała, 2015):

$$V_{ij} = \sum_{k=1}^m w_{ij} x_j^k$$

Este factor es utilizado junto con la tasa de entrenamiento para calcular la modificación de peso de cada sinapsis:

$$w_{ij}^{k+1} = w_{ij}^k + \mu V (x_j^{k+1} - V w_{ij}^k)$$

Referencias

- [0] Gurney, K. Introduction to Neural Networks. Taylor & Francis.
- [1] Kriesel, D. (2007). A Brief Introduction to Neural Networks. Recuperado de <http://www.dkriesel.com>
- [2] Nielsen, M. (n.d.). Neural Networks and Deep Learning. 224.

- [3] Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.
- [4] Szandała, T. (2015). Comparison of Different Learning Algorithms for Pattern Recognition with Hopfield's Neural Network. Procedia Computer Science, 71, 68–75. <https://doi.org/10.1016/j.procs.2015.12.205>

PARTE III: 7. MINERÍA DE TEXTO

ANÁLISIS DE TEXTO SOBRE REDES SOCIALES

Indira Luz Burga Menacho

Introducción

La **minería de texto** es un proceso de análisis de texto no estructurado que busca encontrar patrones, tendencias y/o relaciones en su contenido.

Su finalidad principal es la de extraer conocimiento e información útil para la toma de decisiones en diferentes ámbitos, como pueden ser la publicidad, la política, o el sector empresarial, entre otros.

Sin embargo, dado que lo que se está analizando son lenguajes naturales, resulta difícil definir reglas explícitas para su análisis, ya que por un lado, los lenguajes evolucionan naturalmente con el paso de cada generación, y por el otro, cada ámbito específico cuenta con su propia jerga. Por ello, este trabajo busca abordar los procedimientos necesarios para conseguir el análisis de textos no estructurados, con el fin de entender la interacción de los comentarios de los usuarios, utilizando para ello, procesos de lenguaje natural (NLP) orientados a textos en español. A los fines del mismo, se ha

implementado un análisis sobre un total de **11.060** comentarios realizados en redes sociales sobre un video de publicidad para el fomento del turismo, realizado por el gobierno Peruano.

Elementos del análisis

Una red social es un tipo de red compleja que modela relaciones e interacciones humanas y sociales. Dependiendo de qué datos se analicen, estas interacciones pueden producirse en redes de amistad, de empresas, o en grupos de colaboración. Mediante el análisis de una red se busca entender un sistema complejo.

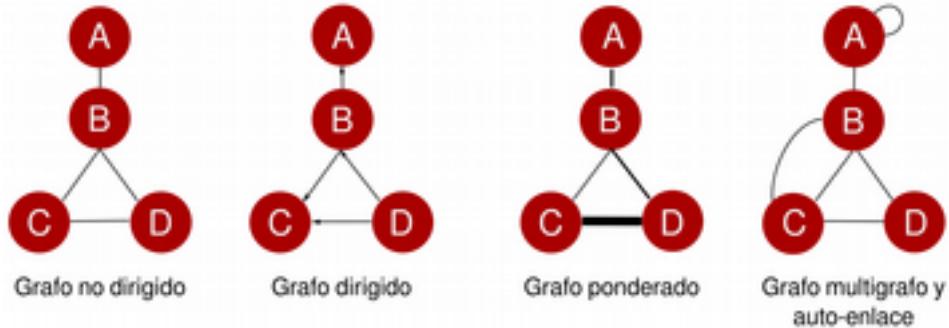
Grafos

Dependiendo del tipo de red analizada, las mismas pueden ser representadas por grafos dirigidos o no dirigidos. Para los propósitos de este trabajo se emplearán los grafos ponderados.

Un **grafo** es la representación matemática de una red, y está compuesto por dos componentes: nodos, denotados por N , y enlaces, denotados por E .

- *Nodos (N):* Los usuarios de la red a construir. En una red social serían los autores de comentarios.
- *Enlaces (E):* Cada una de las conexiones entre dos nodos. Cada enlace representa una relación entre dos nodos de la red, como la relación entre usuarios que responden un comentario.

Los grafos pueden ser representados de diferentes maneras dependiendo del tipo de estructura que se analice, como se muestra en la Ilustración 1.



3. Ilustración – Tipos de grafo

Cada grafo, puede tener un objetivo específico. Por ejemplo:

- Grafos no dirigidos → redes de proteínas.
- Grafos dirigidos → planificación de tareas.
- Grafos ponderados, multi-enlace o autoenlace → redes sociales.

Para entender el comportamiento de una red es necesario conocer los valores de su medida. En este sentido, serán necesarios los siguientes:

- **Distribución de grados:** probabilidad de encontrar nodos con n enlaces. La teoría dice que hay mayor probabilidad de encontrar nodos con pocos enlaces y menor probabilidad de encontrar nodos con muchos enlaces o *hubs*.

- **Grado medio:** determina el promedio de nodos que pueden estar conectados a cualquier otro nodo de una cadena, no mayor a k intermediarios.

$$k = 2E / N$$

- **Coeficiente de agrupamiento (*clustering*):** está dado por la proporción de nodos vecinos conectados a cada nodo. Esto muestra la densidad de la red analizada.
- **Distancia media:** cantidad de saltos promedio de un nodo a otro.

$$d = \log(N) / \log(k)$$

Minería

El **análisis de texto** es un procedimiento complejo, dado que cambia según el contexto, por lo que se debe contar con el conocimiento específico del sector a analizar, a fin de poder identificar con mayor precisión los patrones y/o tendencias.

Los clasificadores comunes y los algoritmos de aprendizaje no pueden procesar directamente los documentos de texto en su forma original. Por lo tanto, se realiza un preprocesamiento donde los documentos se convierten en una representación más manejable, a la cual se denomina *bolsa de palabras (bag-of-words)*.

El preprocesamiento requiere una limpieza que se realiza mediante los siguientes pasos:

- Convertir a minúscula.
- Retirar acentos.
- Eliminar las puntuaciones.
- Eliminar caracteres especiales.
- Eliminar números.
- Eliminar espacios en blanco.
- Eliminar palabras de búsqueda.
- Eliminar palabras vacías (*stop words*)⁴¹.

Los pasos anteriores dependen en gran medida del texto que se analiza. Por ejemplo, una limpieza de comentarios de redes sociales requiere la eliminación de enlaces, emoticones, *hashtags*, o nombres de usuarios, entre otros. Hay que tener en cuenta que no siempre es necesario aplicar cada paso; esto va a depender de lo que se quiera analizar.

Finalizados estos pasos, se procede a aplicar la **lematización**, que consiste en unir palabras que comparten una raíz o lema común y, por tanto, hacen referencia al mismo concepto básico. La importancia de este procedimiento

41 Las palabras vacías, o en inglés, *stop words*, son palabras sin significado de peso propio, tales como artículos, pronombres, preposiciones, entre otras. Estas suelen ser retiradas como parte de la limpieza de datos y análisis de términos frecuentes.

radica en reducir el número de palabras que expresen lo mismo. Por ejemplo. «juego», «juegos», y «juega» son palabras con el mismo lema: *juego*.

Sin embargo, la aplicación de la lematización en español suele ser compleja, pues no existe mucha investigación. Para ello hay dos maneras diferentes de proceder: contar con una base de datos de lematización o utilizar un modelo estadístico entrenado con bases de datos confiables. Un modelo estadístico puede contener un porcentaje de error, debido a que muchas de las raíces de ciertas palabras pueden estar mal asignadas. Sin embargo, un modelo bien entrenado genera un resultado aceptable. Estos modelos suelen entrenarse con grandes y variadas bases de datos. Por otro lado, trabajar con una base de datos de lemas según el idioma elegido puede ser difícil de conseguir, aunque los resultados son buenos pero limitados, dado que una base de datos no suele contener todo los lemas.

El modelo de bolsa de palabras usa los vocablos de cada documento como características. Los métodos de **ponderación** (dar peso) a las características pueden variar. El más simple es el binario en el que el peso de la característica es 1 (uno), si la palabra correspondiente está presente en el documento, o 0 (cero) en caso contrario. Son posibles esquemas de ponderación más complejos, que tengan en cuenta las frecuencias de la palabra en el documento, en la categoría y en la colección completa.

El TF-IDF, es una técnica de recuperación de documentos relevantes de una colección. Esta técnica muestra la importancia de una palabra en un

documento de una colección. Este valor permite determinar qué palabras son más comunes en un documento.

Esta representación es necesaria para entender el contexto, los patrones o las relaciones entre las palabras.

Metodología

Se recopilaron 11.060 comentarios sobre un video publicado en una plataforma en línea, entre las fechas 13 de Setiembre y 19 de Noviembre de 2018. Estos datos fueron extraídos para la demostración de este trabajo para aplicar técnicas de NLP.

La información recuperada fue agrupada y se almacenó con la siguiente estructura de datos por cada comentario:

comment_id	Identificador único de cada comentario
user_comment	Texto del comentario
user_name	Usuario que realizó el comentario
comments_like	Cantidad de <i>likes</i> a un comentario
comment_replycount	Cantidad de réplicas a un comentario
comment_parent	Identificador del comentario padre
comment_date	Fecha en la que se realizó el comentario

Como análisis inicial de la información recuperada, se utilizó la cantidad de comentarios diarios empleando las bibliotecas *matplotlib*⁴², *numpy*⁴³, y el módulo *datetime*⁴⁴ de Python, a fin de representar la distribución de los comentarios en el tiempo:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import datetime
import matplotlib.dates as m_dates
import matplotlib.ticker

def make_month_axis(dates, y, ax, fig):
    newax = fig.add_axes(ax.get_position())
    newax.spines['bottom'].set_position(('outward', 25))
    newax.patch.set_visible(False)
    newax.yaxis.set_visible(False)
    newax.plot_date(dates, y, visible=False)
    newax.xaxis.set_major_locator(m_dates.MonthLocator())
    newax.xaxis.set_minor_locator(m_dates.MonthLocator(bymonthday=15))
    newax.xaxis.set_major_formatter(ticker.NullFormatter())
    newax.xaxis.set_minor_formatter(m_dates.DateFormatter('%b'))
    for tick in newax.xaxis.get_minor_ticks():
        tick.tick1line.set_markersize(0)
        tick.tick2line.set_markersize(0)
        tick.label1.set_horizontalalignment('center')

dates = []
for r in distribucion['comment_date_start']:
    d = datetime.datetime.strptime(f'{r}', '%Y-%m-%d')
    dates.append(d)

y = distribucion['comment_id'].values
fig = plt.figure()
fig.subplots_adjust(bottom=0.2, right=1, wspace=1, hspace=1)
ax = fig.add_subplot(1,1,1)
ax.plot(dates, y)
ax.xaxis.set_major_formatter(matplotlib.dates.DateFormatter('%d'))
make_month_axis(dates = dates, y = y, ax = ax, fig = fig)
plt.show()
```

42 <https://matplotlib.org/>

43 <http://www.numpy.org/>

44 <https://docs.python.org/3/library/datetime.html>

A continuación, se obtuvo información relevante de los usuarios y sus comentarios:

- Los 5 usuarios que han realizado más comentarios.
- Los 5 comentarios con más *likes* por usuario.
- Los 5 comentarios con más respuestas por usuario.

Para ello se utilizó la función *groupby()* de la biblioteca *Pandas*⁴⁵, buscando los 5 usuarios más representativos en cada caso mencionado anteriormente.

```
distribucion_autor = comment_analysis.groupby(
    ['user_name'])['comment_id'].count()
distribucion_autor = distribucion_autor.add_suffix('').reset_index()
distribucion_autor = distribucion_autor.rename(
    columns={'comment_id': 'count'})
distribucion_autor.sort_values(by=['count'], ascending=[False]).head(5)
```

Diseño de los grafos

Para este ejemplo, los nodos serán los usuarios que realizaron los comentarios y estarán relacionados con los usuarios que respondieron a los mismos. Con esta información se armó un conjunto de nodos y enlaces que se usaron para generar un grafo no dirigido (Ilustración 1), que representase la interacción entre ellos.

```
replies = comment_analysis[['comment_parent', 'user_name']]
[comment_analysis['comment_parent'] != '']
data_graph = pd.DataFrame(columns=["node", "edge"])
```

45 <https://pandas.pydata.org>

```

for index, row in replies.iterrows():
    r = (comment_analysis['comment_id'] == row['comment_parent'])
    user = comment_analysis['user_name'][r]
    if user.iloc[0] != row['user_name']:
        data = {
            "node": user.iloc[0],
            "edge": row['user_name']
        }
        data_grah = data_grah.append(data, ignore_index=True)

nodes = data_grah.groupby(['node'])['edge'].count()
nodes = nodes.add_suffix('').reset_index()
nodes = nodes.rename(columns={'edge': 'count'})
node_analizar = nodes['node'][nodes['count'] > 4]
node_analizar
data_grah = data_grah[data_grah['node'].isin(node_analizar)]

```

Se realizó el gráfico con la clase *Graph()* de la biblioteca *networkx*⁴⁶. Los nodos agregados han sido solo aquellos con más de cuatro interacciones, para no saturar el gráfico.

```

import networkx as nx
Graph = nx.Graph()

for index, row in data_grah.iterrows():
    Graph.add_edge(row['node'], row['edge'])

plt.figure(6, figsize=(20,20))
nx.draw_networkx(Graph, node_size=40, font_size=9)

```

Para poder entender mejor el grafo, se realizó el cálculo de las medidas más significativas, tales como el número de nodos, el de enlaces y el grado medio. Para estos cálculos se emplearon las funciones *number_of_nodes()*,

46 <https://networkx.github.io/>

`number_of_edges()`, y la ecuación de k , empleando `Graph.degree()` de la biblioteca `networkx`, respectivamente:

```
num_nodos = Graph.number_of_nodes()
num_edge = Graph.number_of_edges()

values = dict(Graph.degree()).values()
grado_medio = float(sum(values)) / float(num_nodos)
```

Conociendo el grado medio, se implementó la distribución de grados con la función `Graph.degree()` de las biblioteca `networkx`, y `collections.Counter()` de la biblioteca `collections`⁴⁷.

```
import collections
import matplotlib.pyplot as plt
import networkx as nx

data = [d for n, d in Graph.degree()]
degree_sequence = sorted(data, reverse=True) # secuencia de grados
degree_count = collections.Counter(degree_sequence)
deg, cnt = zip(*degree_count.items())
fig, ax = plt.subplots()
plt.bar(deg, cnt, width=0.60, color='g')
plt.title("Distribución de Grados")
plt.ylabel("Cantidad")
plt.xlabel("Grados")
ax.set_xticks([d + 0.4 for d in deg])
ax.set_xticklabels(deg)

# Grafico interno de la red
plt.axes([0.4, 0.4, 0.5, 0.5])
Gcc = sorted(nx.connected_component_subgraphs(Graph), key=len,
            reverse=True)[0]
pos = nx.spring_layout(Graph)
plt.axis('off')
nx.draw_networkx_nodes(Graph, pos, node_size=5)
nx.draw_networkx_edges(Graph, pos, alpha=0.4)
plt.show()
```

47 <https://pymotw.com/2/collections/index.html>

Para obtener el promedio de coeficiente de agrupación (*clustering*) se utilizó la función *average_clustering()*. Por otro lado, para poder realizar el cálculo de la distancia media, se hace necesario conocer si se está trabajando con una red conectada usando la función *is_connected()*. Si la respuesta es *False*, no es posible realizar el cálculo. Para poder interpretar el resultado se necesita verificar si la red cuenta con componentes dispersos, y esto se implementa con la función *number_connected_components()*.

```
promedio_cluster = nx.average_clustering(Graph)
es_conectado = nx.is_connected(Graph)
num_componentes = nx.number_connected_components(Graph)
```

Análisis del texto

La implementación del preprocesamiento de texto inicia con la limpieza. Para el mismo se emplearon las funciones *lower()* y *re.sub()* de la biblioteca *Pandas*⁴⁸, así como las funciones *remove_punctuation()*, *remove_accents()* y *remove_stopwords()* que se encuentran alojadas en un repositorio externo⁴⁹.

```
ca = comment_analysis # alias

# Convertir a minúscula
ca['user_comment'] = ca['user_comenta_original'].str.lower()

# Eliminar palabra de búsqueda
ca['user_comment'] =
    ca['user_comment'].apply(
        lambda x: re.sub('carlos vives', '', x))

# Eliminar las puntuaciones
```

48 <https://pandas.pydata.org/>

49 https://bitbucket.org/indirabm/mineria_texto/src/master/funciones.py

```

ca['user_comment'] = ca['user_comment'].apply(
    lambda x: remove_punctuation(x))

# Eliminar acentos
ca['user_comment'] = ca['user_comment'].apply(
    lambda x: remove_accents(x))

# Eliminar los numeros
ca['user_comment'] = ca['user_comment'].apply(
    lambda x: re.sub(r'\d+', ' ', x))

# Eliminar espacios dobles
ca['user_comment'] = ca['user_comment'].apply(
    lambda x: re.sub(r'\s+', ' ', x))

# Eliminar stopwords
ca['user_comment'] = ca['user_comment'].apply(
    lambda x: remove_stopwords(x))

```

Para finalizar el preprocesamiento se aplicó la función *spacy.blank('es')*, que implementa el modelo *es_core_news_md*⁵⁰. Este modelo es un CNN multitarea en español, una clase de redes neuronales artificiales profundas y avanzadas (donde las conexiones entre nodos no forman un ciclo) y utilizan una variación de perceptrones multicapa diseñados para requerir un preprocesamiento mínimo. Estos están inspirados en la corteza visual animal. El *es_core_news_md* fue entrenado en el corpus de *AnCora* y *WikiNER*. La función asigna vectores de *token* específicos del contexto, etiquetas POS⁵¹, análisis de dependencia y entidades con nombre. Admite la identificación de entidades PER, LOC, ORG y MISC⁵².

50 https://spacy.io/models/es#es_core_news_md

51 POS = Especifica la sintaxis de cada palabra. Ejemplo, NOUN, ADJ, DET, ADP, PROPN, VERB, CÓNJ, PUNCT.

52 PER indica el nombre de la persona; LOC, ORG, MISC representa las locaciones, organizaciones y diversos lugares.

```

import spacy
nlp = spacy.blank('es')
comment = comment_analysis['user_comment']
comment_lema= []

for sentence in comment:
    lema = []

    for token in nlp(sentence):
        lema.append(token.lemma_)

    lema_comment = " ".join(lema)
    comment_lema.append(lema_comment)

comment_analysis['user_comment'] = comment_lema
comment_analysis['user_comment'] =
comment_analysis['user_comment'].apply(
    lambda x: re.sub('limar', 'lima', x))

```

Al revisar los datos se identificó un error de la aplicación del modelo: al encontrar la palabra “lima” el modelo la cambió por “limar”, siendo *Lima* el nombre propio de una ciudad. Este se rectificó al final del código.

Con esta corrección se obtuvieron los 10 *bigramas*. Los **bigramas** son una combinación de 2 palabras consecutivas que se repiten muchas veces a lo largo del texto. Se buscan las más frecuentes, pues estas brindan un contexto de lo que se habla en los comentarios. Para ello se *tokeniza* el conjunto de datos con la función *word_tokenize()* y luego se le aplica la función *bigrams()*. Finalmente se determina la frecuencia de los mismos para encontrar los más usados, y luego mostrarlos en una gráfica acumulada. Para todo este proceso se usó la librería *nltk*⁵³.

```
words_rows = ca['user_comment'].apply(
```

53 <https://www.nltk.org/>

```
lambda x: nltk.word_tokenize(x))

words = []
for w in words_rows:
    words.extend(w)

# Calcular los bigramas del texto
works_freq = nltk.bigrams(words)

# calcular la frecuencia de los bigramas
word_dist = nltk.FreqDist(works_freq)
word_dist.most_common(10)
word_dist.plot(10,cumulative=False)
```

Para obtener una representación gráfica de la relevancia de las palabras en un texto se utilizó *wordcloud*.

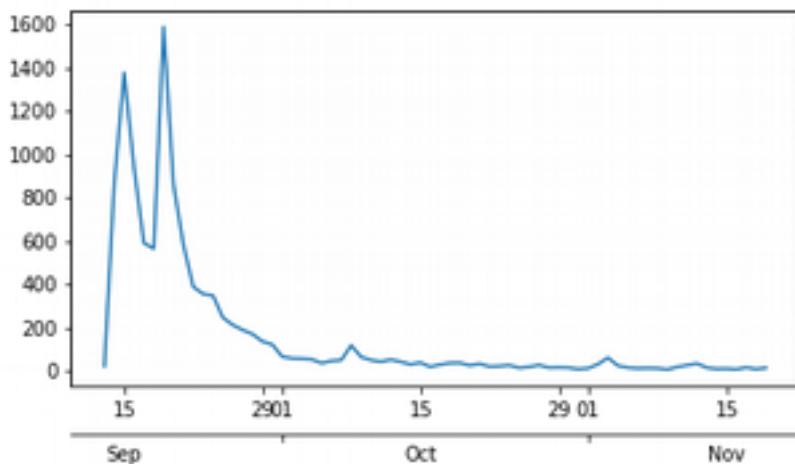
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

stopwords = set(STOPWORDS)
show_wordcloud(words_rows)
```

Resultados y discusión

En este trabajo se recopilaron 11.060 comentarios de un video de Internet durante algo más de dos meses. Como se observa en la Ilustración 4, se alcanzaron dos picos de comentarios, uno 15 y otro el 19 de Setiembre. Después de esta última fecha los comentarios son mucho menores.

Este resultado muestra que el impacto fue inicial y tuvo un crecimiento alto en los 2 primeros días, y sin embargo no se mantuvo en el tiempo.



4. Ilustración: Distribución de los comentarios

Las Ilustraciones 5, 6 y 7, cuyos nombres de usuarios han sido sustituidos por código autogenerado para proteger la privacidad de los mismos, muestran un *ranking* de la relación de los comentarios y los usuarios.

Como se aprecia en las ilustraciones 5 y 6, no necesariamente los comentarios con más *likes* son realizados por los usuarios que más comentan, dado que uno solo de ellos podría ser viral rápidamente. Esto se apoya en el *Modelado de epidemias en redes sociales on-line*⁵⁴ mencionado también en el artículo *Las redes sociales y la propagación de epidemias*.

54 http://www.aepia.org/aepia/files/EVIA/Transparencias/Cordn_EVIA2014.pdf

user_name	count
226rvb922w	64
396tps40y	51
740ke5172g	31
276h9l252z	28
721l2l883b	26

5. Ilustración

user_name	count
620xvzb606r	6474
780glg729l	2669
807qzv781j	1727
429koe274k	1611
792aqz855r	1337

6. Ilustración

user_name	count
620xvzb606r	273
807qzv781j	192
780glg729l	142
602bfq964t	127
792aqz855r	113

7. Ilustración

Referencias:

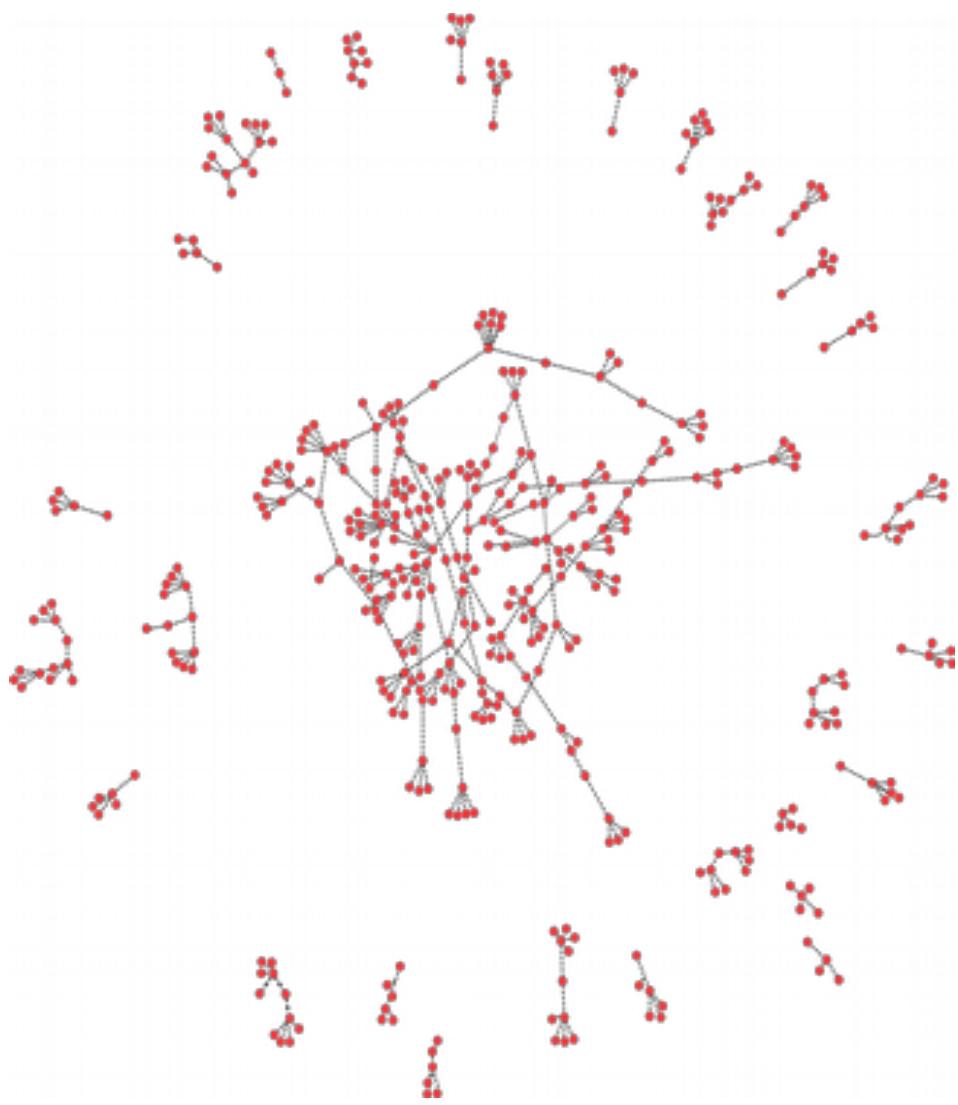
Ilustración 5: Los cinco autores con mayor cantidad de comentarios.

Ilustración 6: Los cinco comentarios con mayor cantidad de “likes”.

Ilustración 7: Los cinco comentarios con mayor cantidad de respuestas por autor.

El análisis de redes sociales muestra un grafo no dirigido muy disperso, según se aprecia en la Ilustración 8 (*ver página 302*).

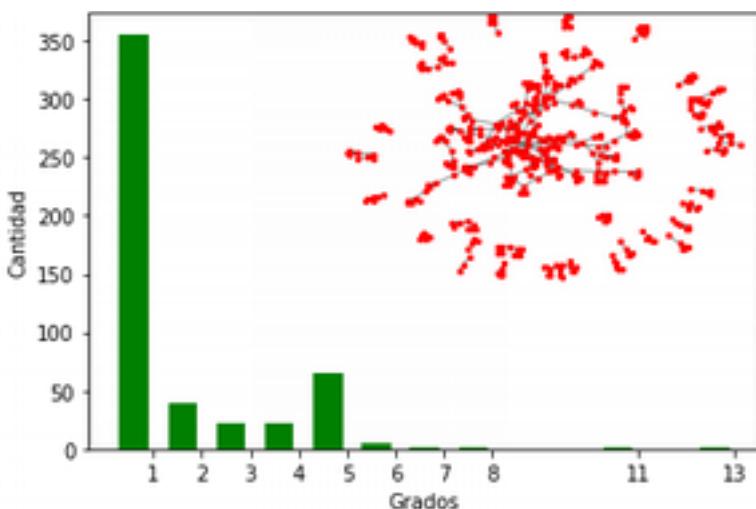
La distribución de los grados en dicha ilustración, muestra que la mayoría de nodos solo cuentan con un grado y luego disminuyen radicalmente.



8. Ilustración - Grafo no dirigido de comentarios y sus replicas

© 2019 CC BY - Vectorizado por Luisa Aurora Rojas Goicochea

Ilustración 9: Distribución de Grados del grafo.



Para poder entender con más detalle el grafo, se revisan los resultados obtenidos. Como se aprecia, el grafo tiene un promedio de dos conexiones por nodo.

Número de nodos: 514

Número de enlaces: 492

Grado medio: 1.9143968871595332

Densidad del grafo: 0.00373176781122716

¿Grafo conectado?: False

Promedio del coeficiente de clustering: 0.0

Número de componentes del grafo: 34

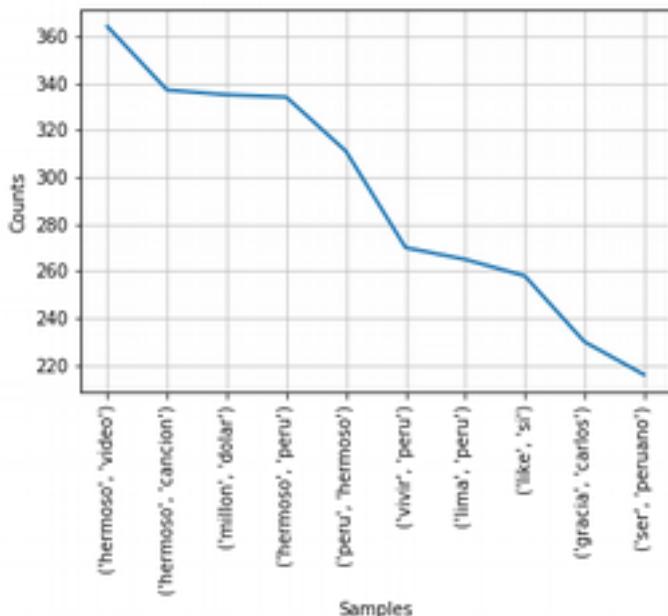
La densidad del grafo es 0.004, siendo el rango de valores de 0 a 1. Esto significa que el grafo analizado es de muy poca densidad, y por lo tanto es disperso.

El siguiente dato confirma que el grafo no está conectado. Por esta razón cuando se quiere obtener el coeficiente de *clustering* se muestra en cero.

Con este resultado se verifica el número de componentes del grafo, el cual es 34. Hay muchas islas no conectadas lo que hace que sea un grafo muy poco denso.

Los datos no muestran una red sólida para analizar, por lo que no es válido realizar más análisis. Es importante analizar estos datos pues otras redes pueden brindar información relevante del comportamiento de sus nodos.

Con la bolsa de palabras se obtienen los 10 bigramas más relevantes. Para este caso se usan bigramas que brindan un contexto de lo que se quiere decir en los comentarios, dado que una sola palabra no necesariamente brinda tanta información.



10. Ilustración - Los 10 bigramas más relevantes de los comentarios.

Los dos bigramas iniciales son “*hermoso video*” y “*hermosa canción*”. Por tanto, muestran una opinión positiva sobre del video. El tercer bigrama habla de un millón de dólares, esto es realmente un comentario negativo dado que el millón de dólares se refiere al gasto que se hizo para producir el video.

El resultado final de este análisis es una nube de palabras (*wordcloud*) que representa los vocablos relevantes del texto según su tamaño. Como se puede visualizar, las palabras más usadas son *lima*, *hermosa* y *gracias*. De tamaño medio se puede ver la palabra *millón*.



11. Ilustración

Si bien es cierto que la nube de palabras es una representación visualmente interesante por su facilidad de lectura a primera vista, esta no necesariamente genera tanto valor para entender el contexto. Una manera de mejorar este análisis sería aplicar *análisis de sentimientos*, lo que permitiría dar un peso emocionalmente negativo o positivo a cada palabra.

En este trabajo se han podido aplicar diferentes técnicas de análisis de texto las cuales han permitido poder entender el contexto de los datos analizados. Cabe mencionar que para poder analizar texto es necesario conocer del tema sobre el cual se está generando dicho análisis, a fin de reconocer con mayor facilidad las tendencias, los patrones o errores mostrados en los resultados, como por ejemplo, saber que el bigrama “*millón dolares*” proviene en su mayoría de comentarios críticos al gasto realizado por el gobierno peruano.

Como anteriormente se menciona, estas no son las únicas técnicas de minería de texto a aplicar, y no existe una regla o flujo exacto de cómo trabajar. Por ello, todo depende de aquello que se quiera analizar.

Bibliografía

1. Bing Liu, May (2012). Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers.
2. Matthew A. Russell (2014), Mining the Social Web, 2nd Edition.
3. J. M. Alonso, D. P. Pancho, O. Cordón, A. Quirin, L. Magdalena, Social network analysis of co-fired fuzzy rules. Soft Computing: State of the Art Theory and Novel Applications, R.R. Yager, A.M. Abbasov, M.Z. Reformat, S.N. Shahbazova (Eds.), Studies in Fuzziness and Soft Computing, Springer, Volume 291, 2013, 113-128
4. Oscar Cordón García, Escuela de Verano de Inteligencia Artificial (EVIA 2014), Redes Sociales y Modelado Basado en Agentes.

5. Joel Grus (2015), Data Science from Scratch.
6. Piccardi, C. Lett Mat Int (2013), Social networks and the spread of epidemics, 1: 119. <https://doi.org/10.1007/s40329-013-0022-0>
7. Steven Bird, Ewan Klein, Edward Loper , Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit 1st Edition
8. Yoon Kim, Convolutional Neural Networks for Sentence Classification
9. Ronen Feldman, James Sanger, The Text Mining Handbook

PARTE III: 8. PROGRAMACIÓN

GUÍA DE REFERENCIAS DEL LENGUAJE C

Eugenia Bahit

Acerca de C

C es un lenguaje de programación de propósitos generales que abarcan desde la programación de sistemas operativos y lenguajes informáticos, hasta el desarrollo de programas de escritorio, aplicaciones Web, y herramientas a más alto nivel. Si bien es un lenguaje de alto nivel, permite un manejo avanzado de operaciones a bajo nivel, las cuales actúan directamente sobre el *hardware*. Es fuertemente tipado y perteneciente a la categoría de lenguajes compilados.

Estructura de un programa C

Un programa C, contiene un orden específico de secciones, que son descritas a continuación.

Sección 1: preprocesadores

Los preprocesadores son directivas que se ejecutan modificando el código fuente, antes de que este sea compilado. Uno de los preprocesadores más

comunes, es la directiva `#include`, que trae al código fuente, todos los elementos definidos en un archivo de encabezado, antes de que los fuentes sean compilados.

```
#include <archivo.h>
```

Existen otros preprocesadores que no necesariamente se incluyen al comienzo, los cuáles se describen más adelante, en la sección «*Preprocesadores comunes y definición de macros*».

Sección 2: declaraciones globales

Identificadores de ámbito global tales como variables o constantes.

```
int number = 5;
```

Sección 3: función main()

Es la primera función ejecutada por todo programa C.

Puede definirse con o sin valor de retorno, y con o sin parámetros:

```
int main()
{
    ...
}

void main()
{
    ...
}
```

```
int main(int argc, char* argv[])
{
    ...
}
```

Internamente, la estructura de la función *main()*, se dividirá en tres secciones:

1. Declaraciones: toda variable o constante necesaria en el ámbito de la función.
2. Instrucciones ejecutables: instrucciones directas de código ejecutable.
3. Subrutinas: llamados a funciones definidas por el usuario.

Sección 4: funciones definidas por el usuario

Funciones definidas por el programador.

Requerimientos sintácticos

Toda función se inicia con una llave de apertura y finaliza con una llave de cierre.

```
void function()
{
}
```

Todas las estructuras de control se inician con una llave de apertura y finalizan con una de cierre. Esto no es obligatorio si la estructura de control, ejecuta una única instrucción.

```
if(condicion) {
    instrucción 1
    instrucción 2
}

if(condicion) instrucción única
```

```
if(condicion)
    instrucción única
```

Toda instrucción finaliza con un punto y coma.

```
int variable = 5;
```

Los comentarios se inician con una barra diagonal y un asterisco y finalizan con un asterisco y una barra diagonal. Pueden escribirse en una sola línea o en bloque. También pueden escribirse comentarios de una sola línea, iniciando los mismos por una doble barra diagonal.

```
/* Comentario de una línea */
// Comentario de una sola línea
/*
    Comentario en bloque
*/
```

Como norma general, se prefieren las instrucciones escritas en letras minúsculas, reservando las mayúsculas para las constantes.

Compilación básica y ejecución

Se puede utilizar GCC para compilar en GNU/Linux, aunque otros compiladores como *Clang* están disponibles.

Para instalar GCC, correr uno de los siguientes comandos según corresponda:

```
apt install gcc # Debian y derivados
dnf install gcc # Fedora y derivados
```

Para una compilación básica, ejecutar:

```
gcc -o <salida> <origen>
```

Algunos ejemplos:

```
gcc -o salida archivo.c  
gcc -o salida directorio/
```

Las instrucciones previas, en ambos casos, generarán un archivo ejecutable denominado *salida*, el cual podrá ser lanzado con la instrucción:

```
./salida
```

Archivos de encabezado más comunes de la biblioteca estándar de C

Archivo	Descripción
<i>ctype.h</i>	Funciones de evaluación de tipo de caracteres.
<i>stdio.h</i>	Funciones de entrada y salida. Operaciones de archivos.
<i>string.h</i>	Funciones para cadenas de texto.
<i>math.h</i>	Funciones matemáticas.
<i>stdlib.h</i>	Funciones de uso general.
<i>assert.h</i>	Funciones de diagnóstico. Utilizadas principalmente con propósitos de depuración.
<i>stdarg.h</i>	Funciones para manipular argumentos de funciones (cuando la cantidad y tipo de argumentos es conocida).
<i>signal.h</i>	Identificadores que permiten efectuar un manejo de las señales

	arrojadas por un programa.
<i>time.h</i>	Funciones relacionadas con fecha y hora.
<i>limits.h</i>	Define las longitudes mínimas para cada tipo de datos.
<i>float.h</i>	Constantes relacionadas con la aritmética de coma flotante.
<i>Incorporado en el estándar C99</i>	
<i>stdbool.h</i>	Proporciona el tipo <i>bool</i> como alternativa a <i>_Bool</i> , y los tipos <i>true</i> y <i>false</i> , como alternativas a los valores booleanos 1 y 0 respectivamente.

6. Tabla: Lista de archivos de encabezado de la biblioteca estándar

Tipos de datos

Enteros, números de coma flotante y subtipos correspondientes

ENTEROS					
Tipo	S	U	Tamaño	Valor mínimo	Valor máximo
<i>char</i>		X	1 byte	0	255
	X			-128	127
<i>int</i>	X		2 o 4 bytes	-32,768 o -2,147,483,648	32,767 o 2,147,483,647

	X			0	65,535 o 4,294,967,295
<i>short</i>	X		2 bytes	-32,768	32,767
		X		0	65,535
<i>long</i>	X		4 bytes	-2,147,483,648	2,147,483,647
		X		0	4,294,967,295

Incorporado en el estándar C99

	X		8 bytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
		X		0	18,446,744,073,709,551,615

NÚMEROS DE COMA FLOTANTE

Tipo	Tamaño	Precisión	Valor mínimo	Valor máximo
<i>float</i>	4 bytes	6 dígitos	±3.4E+38	1.2E-38
<i>double</i>	8 bytes	15 dígitos	±1.7E+308	2.3E-308
<i>long double</i>	10 bytes	19 dígitos	±1.1E+4932	3.4E-4932

7. Tabla: Tipos de datos y sus características

S: ***signed*** (acepta positivos o negativos); U: ***unsigned*** (no acepta negativos).

El tipo `char` y las cadenas de texto

Una variable de tipo `char`, admite como valor un carácter con entrecomillado simple, o su representación entera en el juego de caracteres, determinada por un valor entre 0 y 255:

```
char letra = 65; // decimal correspondiente al carácter 'A'  
char letra = 'A'; // carácter correspondiente al decimal 65
```

Un símbolo formado por más de un carácter (como el caso de una vocal acentuada), excede la cantidad de bytes que pueden ser almacenados por una variable de tipo `char`. Por lo tanto, se debe crear una *colección de caracteres* con entrecomillado doble:

```
char letra[] = "á";
```

El mismo sistema debe emplearse para generar cadenas de texto:

```
char nombre[] = "Eugenia Bahit";
```

Matrices (tipo: `Array`)

Son colecciones de un tipo determinado, y pueden ser de una sola dimensión (como la anterior colección de caracteres utilizada para generar cadenas de texto) o de múltiples dimensiones.

Mientras que la forma preferida para crear colecciones de caracteres es el entrecomillado, se prefieren las llaves para otras colecciones.

Una colección que se rellena con 10 ceros:

```
int array[10];
```

La colección inicializada anteriormente, donde los dos últimos elementos se rellenan con ceros:

```
int array[10] = {1, 1, 2, 3, 5, 8, 13, 21};
```

Matriz de 3 filas y 5 columnas cuyos elementos son ceros:

```
int matriz[3][5];
```

Matriz de 3 filas con 5 elementos cada una:

```
int matriz[3][5] = {  
    {1, 2, 3, 4, 5},  
    {10, 20, 30, 40, 50},  
    {100, 200, 300, 400, 500}  
};
```

Notar que la alineación estética de la matriz, solo se ha utilizado a fin de lograr un mejor entendimiento. La alineación de código estética no debe utilizarse en el código fuente.

Colección de números de coma flotante

```
float pesos[] = {74.5, 56.12, 84.3}
```

Acceso a elementos de un array

El acceso se efectúa mediante posición del elemento deseado:

```
variable[posicion];
```

Valores booleanos

En C, un booleano se representa por el tipo `_Bool`, cuyo valor *false* se representa por un cero, mientras que el verdadero, *true*, por un 1.

```
_Bool activo = 1; // activo es verdadero
```

Se puede utilizar el tipo `bool` con valores *true* y *false*, incluyendo el archivo de encabezados `stdbool.h` incorporado en el estándar **C99**.

```
#include <stdbool.h>
...
bool activo = true;
```

Valores nulos (`void`)

Utilizado para definir un valor de retorno nulo en una función.

```
void main()
{
    ...
}
```

Conversión de tipos o encasillamiento (*typecast*)

La conversión de tipos se lleva a cabo anteponiendo el nuevo tipo entre paréntesis, al valor que se desea convertir:

```
float numero_real = 10.25;
int numero_entero = (int) numero;
```

Lo anterior, convertirá a 10.25 en 10 y lo asignará como valor a la variable `numero_entero`.

Variables y Constantes

Ámbito y alcance de las variables

El alcance de una variable se determina por el ámbito en el cuál la variable es definida.

Variable global. Cuando una variable se define al comienzo del archivo, tiene como alcance cualquier parte del fichero (esto incluye, dentro de cualquier estructura de control y/o función).

Variable local. Cuando una variable se define al comienzo de una función, tiene como alcance toda la función.

Variable temporal. Cuando dentro de una función, se define una variable dentro de una subestructura de control encapsulada entre llaves (como por ejemplo, un condicional), la variable solo existe en el momento que se la declara y deja de existir cuando la estructura de control finaliza.

Definición y asignación de variables.

Una variable se define con la siguiente sintaxis:

```
[<tipo de almacenamiento> ]<tipo de datos> <nombre de variable>;
```

Ejemplo:

```
static int numero;
```

Una variable puede ser asignada al momento de ser definida o posteriormente:

```
int numero = 5;      // asignada en tiempo de definición  
int numero;  
numero = 5;          // asignada luego de haber sido definida
```

Especificador de tipo de almacenamiento

El tipo de almacenamiento es opcional. Los especificadores pueden ser cualquiera de los siguientes:

typedef

No define una variable propiamente dicha sino, un alias para un tipo determinado.

```
typedef int entero; // ahora 'entero' es sinónimo del tipo 'int'  
entero numero;     // se declara una variable de tipo entero
```

extern

Genera que una variable definida en un archivo externo pase al ámbito actual.

```
extern int numero;  
// ahora puede utilizarse la variable numero definida en otro archivo
```

static

Genera que una variable temporal (definida dentro de una función), conserve su valor en las sucesivas llamadas a dicha función.

```
void funcion()  
{  
    static int numero = 0;
```

```
    numero++;
}

funcion(); // numero = 0
funcion(); // numero = 1
funcion(); // numero = 2
funcion(); // numero = 3
```

auto

Es el valor por defecto de una variable. Genera que una variable tenga un tiempo de vida local.

register

Mantiene el valor de una variable en un registro de la CPU para que su acceso pueda desarrollarse tan rápido como sea posible.

Constantes

Una constante se define con la siguiente sintaxis:

```
const <tipo de datos> <NOMBRE DE CONSTANTE> = <valor>;
```

Por lo general, se prefiere el empleo de letras mayúsculas para la definición de nombres de constantes. Por otra parte, dado que la constante mantiene su valor inmutable, debe ser asignada al momento de ser definida:

```
const int NUMERO = 5;
```

Enumeración

La enumeración permite definir múltiples constantes de tipo entero, donde cada elemento definido, asume como valor el correspondiente a su posición en la lista (comenzando en cero), a menos que se le declare uno diferente de forma explícita:

```
enum {CERO, UNO, DOS, TRES};
```

Cada constante tiene los valores 0, 1, 2, y 3 respectivamente.

```
enum {CERO, UNO, DIEZ=10, TRES};
```

Aquí, las constantes *CERO*, *UNO* y *TRES*, asumen los valores 0, 1 y 3 de forma implícita, mientras que *DIEZ*, asume el valor 10 de forma explícita.

Si a una lista enumerada se le asigna un *nombre de identificador*, luego puede ser utilizado como identificador de tipo de los parámetros de una función:

```
enum puntaje {CERO, UNO, DOS, TRES, CUATRO, CINCO, MIN=0, MAX=5};  
void funcion(enum puntaje p);
```

Punteros

Los punteros son enlaces a un registro concreto de la memoria.

```
int *puntero; // puntero  
int variable; // variable convencional
```

Mientras que una variable maneja los datos de un registro de memoria, un puntero, maneja directamente el registro concreto. De esta forma, el uso de punteros suele economizar el uso de memoria. Por ejemplo, al pasar una

variable como parámetro de una función, el contenido del registro de la variable se copia literalmente al ser pasado a dicha función, y por lo tanto, se duplican los datos. Sin embargo, si lo que se pasa a la función es un puntero, en realidad, se le está pasando un enlace hacia el registro de memoria que contiene esos datos, por lo tanto, los datos son leídos desde un único registro, por lo que no se hace necesario duplicarlos.

Cuando una función recibe un puntero como parámetro, dicho parámetro queda enlazado a la variable original (puntero). Por consiguiente, cualquier modificación sobre los datos de dicho parámetro, repercute sobre el registro en el cuál los datos originales son almacenados. Es decir, que cuando una variable se pasa a una función como puntero, y la función modifica los datos del puntero, los datos de la variable se alteran (esto es justamente porque el parámetro, en realidad, es un puntero hacia dicha variable y no el valor en sí mismo).

Si se desea crear un puntero hacia una variable preexistente, se debe asignar a dicho puntero, la dirección de memoria de la variable a la cuál se desea enlazar. Esto se logra anteponiendo el signo *ampersand* delante del nombre de la variable a la cuál se desea enlazar el puntero:

```
int variable;  
int *puntero = &variable;
```

Un puntero se puede mover *n* bytes hacia adelante o hacia atrás mediante:

```
puntero + n  
puntero - n
```

Por ejemplo:

```
char cadena[7] = "abcdef", *puntero = cadena;
// puntero se mueve al byte 0 de cadena

char *puntero2 = puntero + 4;
// puntero se mueve 4 bytes y desde ahí se asigna a puntero2

printf(puntero2); // Salida: ef
```

Estructuras

Una estructura es un conjunto de variables de diversos tipos pertenecientes a un mismo grupo. Una estructura puede ser vista como una forma de crear modelos de datos reutilizables.

Definición de estructuras

La definición de estructuras puede efectuarse de tres maneras posibles.

Forma 1. Una estructura puede ser creada con un **nombre de estructura**:

```
struct nombre {
    // variables separadas por punto y coma
};
```

Forma 2. Se define sin nombre de estructura pero **asignada a las variables** cuyo valor estará basado en dicha estructura:

```
struct {
    // variables separadas por punto y coma
} variable1, variable2, variable3;
```

Forma 3. Una estructura también puede ser declarada como un **identificador de tipo** y en este caso, el nombre de la estructura es opcional:

```
typedef struct nombre_opcional {
    // variables separadas por punto y coma
```

```
    } tipo;
```

Definición de cadenas de texto dentro de una estructura

Las colecciones de tipo *char* (cadenas de texto) definidas dentro de una estructura, pueden ser declaradas de dos formas:

1) *Como punteros*

```
char *variable;
```

Para luego ser modificadas mediante:

```
estructura.variable = "nuevo valor";
```

2) *Como colecciones de longitud fija reservada*

```
char variable[longitud];
```

Para luego ser modificadas replicando una colección, mediante el uso de la función *strcpy()* del archivo de encabezado *string.h*

```
strcpy(estructura.variable, "nuevo valor");
```

Dependiendo de la longitud de los valores, puede ser preferible crear dichas variables como punteros, dado que será menos costoso modificar la variable de forma directa, que crear una copia exacta de la cadena en memoria.

En la instrucción:

```
strcpy(estructura.variable, "nuevo valor");
```

El valor:

```
"nuevo valor"
```

Es almacenado en memoria y luego copiado de forma exacta en *estructura.variable*. Mientras que si *estructura.variable* es un puntero, el valor se almacenaría directamente en dicho puntero.

Creación de variables basadas en estructuras

Para estructuras definidas con la *forma 1*, la sintaxis es:

```
struct <nombre-estructura> <nombre-variable>;
```

Para estructuras definidas con la *forma 2*, las variables quedan inicializadas en tiempo de definición, por lo que no es necesario crearlas.

Para estructuras definidas con la *forma 3*, se utiliza el identificador de tipo de la estructura como identificador de tipo nativo:

```
<identificador-estructura> <nombre-variable>;
```

Acceso a las variables de una estructura

La forma habitual de acceder a las variables de una estructura, es mediante la siguiente sintaxis:

```
variable.variable_interna
```

Ejemplo:

```
struct estructura variable;
variable.variable_interna = 1;
```

Ejemplos de uso de estructuras

Estructuras definidas con la *forma 1*:

```
struct Persona {  
    int identificador;  
    char *apellido;  
    char *nombre;  
    unsigned short edad;  
};  
  
struct Persona persona;  
  
persona.identificador = 1005;  
persona.apellido = "Bahit";  
persona.nombre = "Eugenio";  
persona.edad = 40;
```

Misma estructura pero definida con la *forma 2* (aquí el nombre de estructura es opcional):

```
struct Persona {  
    int identificador;  
    char *apellido;  
    char *nombre;  
    unsigned short edad;  
} persona;  
  
persona.identificador = 1005;  
persona.apellido = "Bahit";  
persona.nombre = "Eugenio";  
persona.edad = 40;
```

Misma estructura pero definida con la *forma 3* (aquí el nombre de estructura también es opcional):

```
typedef struct Persona {  
    int identificador;  
    char *apellido;  
    char *nombre;  
    unsigned short edad;  
} Persona;
```

```
Persona persona;  
  
persona.identificador = 1005;  
persona.apellido = "Bahit";  
persona.nombre = "Eugenio";  
persona.edad = 40;
```

Para un ejemplo de estructuras como punteros, ver el apartado «*Observaciones finales sobre las estructuras*».

Observaciones finales sobre las estructuras

Una estructura puede resultar semejante al manejo de objetos si se tiene por costumbre utilizar un lenguaje orientado a objetos. Sin embargo, **una estructura no es similar a un objeto**, dado que cuando una variable es creada, no queda enlazada a la memoria a no ser que se cree explícitamente un puntero enlazado a dicha variable:

```
struct estructura variable, *apuntador = &variable;
```

En el código anterior, *apuntador* es el puntero a la variable creada bajo la estructura llamada “*estructura*”. En este caso, el acceso a las variables internas se hará mediante el operador **->** de la siguiente forma:

```
apuntador->variable_interna = valor;
```

Se puede apreciar mejor en el siguiente ejemplo:

```
struct Persona {  
    int identificador;  
    char *apellido;  
    char *nombre;  
    unsigned short edad;
```

```
};

struct Persona persona, *p = &persona;

p->identificador = 1005;
// persona.identificador es 1005
```

Dado que *p* solo es un puntero hacia la variable, es posible seguir manipulando los datos directamente desde la variable:

```
persona.apellido = "Bahit";
persona.nombre = "Eugenio";
persona.edad = 40; // p->edad es 40
```

Uniones

En una estructura, cada variable interna se aloja en un registro de memoria independiente. Esto permite que todos los elementos de una misma estructura, puedan ser utilizados de forma simultánea.

En una unión, todas las variables internas comparten el mismo registro de memoria. Esto permite utilizar una única variable interna de una misma unión, a la vez (es decir, cada vez que se invoca a una variable interna de una misma unión, dicha variable sobrescribe el contenido existente del registro de memoria reservado por la unión).

La unión se define de la misma forma que una estructura, sustituyendo la palabra *struct* por *union*.

```
union Data {
    int entero;
    float real;
};
```

Operadores básicos

Operadores aritméticos	Operadores lógicos	
+ SUMA - RESTA * MULTIPLICACIÓN / DIVISIÓN % MÓDULO	&& AND OR ^ XOR !	NOT
Prefijos y sufijos		Unarios
++ INCREMENTO EN 1 -- DECREMENTO EN 1	~	COMPLEMENTO
Operadores de comparación		
> MAYOR QUE < MENOR QUE => MAYOR O IGUAL QUE	<= MENOR O IGUAL QUE == IGUAL QUE != NO IGUAL QUE	

8. Tabla: Tipos de operadores soportados por C

Operador unario sizeof

El operador unario sizeof permite conocer la longitud de un objeto (bien sea una variable, array o estructura) o de un tipo concreto como int, float, el nombre de una estructura o puntero. Su sintaxis es:

`sizeof objeto;`

o para nombres:

`sizeof (nombre del tipo);`

Estructuras de control de flujo

Estructuras de control condicionales

if/else

Las llaves son requeridas cuando engloban más de una instrucción. Sino, son opcionales.

```
if(condición) instrucción;  
else if(condición) instrucción;  
else instrucción;
```

```
if(condición) {  
    instrucción 1;  
    instrucción 2;  
} else if(condición) {  
    instrucción 1;  
    instrucción 2;  
} else {  
    instrucción 1;  
    instrucción 2;  
}
```

switch

```
switch(variable) {  
    case VALOR1:  
        instrucción;  
        break;  
    case VALOR2:  
        instrucción;  
        break;  
    case VALOR3:  
        instrucción;  
        break;  
    default:  
        instrucción;  
}
```

Estructuras de control iterativas

Al igual que en el *if/else*, las llaves solo son necesarias si la estructura engloba más de una instrucción.

while

```
while(condición) {
    instrucción;
}
```

for

```
for(expresión_inicial; condición; expresión_final) {
    instrucción;
}
```

Lo anterior es equivalente a:

```
expresión_inicial;
while(condición) {
    instrucción;
    expresión_final;
}
```

do

A diferencia de *while*, aquí la instrucción se ejecuta al menos una vez al inicio, y luego, hasta tanto se cumpla la condición.

```
do {
    instrucción;
} while(condición);
```

break y continue

La instrucción *break*, permite finalizar el bucle antes de tiempo, cuando una condición se cumple.

```
for(i=0; string[i]; i++) {
    if(i > 10) break;
```

```
}
```

La instrucción *continue*, permite saltar a la siguiente iteración cuando una condición se cumple.

```
for(i=0; string[i]; i++) {  
    if(i < 10) continue;  
}
```

Declaración de retorno

goto

En palabras de los propios creadores del lenguaje C, Denis Ritchie y Brian Kernighan:

«Formalmente, *goto* nunca es necesario y en la práctica, casi siempre es fácil escribir código sin él» —Ritchie & Kernighan, 1988

Por recomendación de los autores del lenguaje, la expresión *goto* solo debería ser utilizada para el manejo excepcional de errores, mediante la siguiente sintaxis:

```
if(desastre) {  
    goto etiqueta;  
}  
  
etiqueta:  
    // instrucciones para limpiar el desastre
```

Dado que existen formas más simples, fáciles de entender, mantener y que no ofuscan tanto el código como la declaración *goto*, **se recomienda no utilizar esta instrucción**.

return

La instrucción *return* finaliza la ejecución de una función, retornando un valor determinado y permitiendo regresar al lugar mismo desde el cuál la función fuera invocada.

```
return valor;
```

Preprocesadores comunes y definición de macros

Los preprocesadores (introducidos al hablar sobre la «*Estructura de un programa C*»), son directivas que se ejecutan y modifican el código antes de que sea compilado.

Un preprocesador puede encargarse de diversas acciones, entre las más comunes:

Efectuar la sustitución de macros

```
#define NOMBRE_MACRO valor-de-reemplazo  
#define NOMBRE_MACRO(parámetros) (valor-de-reemplazo)
```

Las macros sustituyen cada aparición en el código fuente, por el valor correspondiente, de forma tal que al compilarse el programa, el código fuente estará ya sustituido.

```
#define VERSION 1.1
```

Sustituirá todas las apariciones de “VERSION” por el valor “1.1”.

```
#define BYE(persona) printf("Hasta luego, %s!", persona);
```

Sustituirá todas las apariciones de BYE(variable) por la instrucción:

```
printf("Hasta luego, %s!", variable);
```

Efectuar acciones de compilación condicional

```
#if condición
    instrucción
#elif condición
    instrucción
#else
    instrucción
#endif
```

Y la ya mencionada, **efectuar la inclusión de archivos**:

```
#include <archivo-de-encabezado.h>
```

Los preprocessadores son interpretados línea por línea y pueden aparecer en cualquier parte del código. Sin embargo, es esperable que los archivos de encabezado se incluyan al comienzo del fichero.

Como regla general, se sugiere **evitar el uso innecesario de preprocessadores**.

Usos comunes de la biblioteca stdio.h

Manejo básico de entrada y salida

Manejo de salida con printf()

```
printf("formato", variables...);
```

El *formato* se genera a partir del modificador % seguido de un *carácter de conversión de tipo* (entre los más comunes: *i*, *d* para enteros; *o* para octal; *x*, *X* para hexadecimales en minúsculas y mayúsculas respectivamente; *u* para decimales, *f* para números de coma flotante; *c* para un carácter simple; *s* para colecciones de caracteres; entre otros).

Opcionalmente, entre el modificador y el carácter de conversión de tipo, pueden emplearse *banderas de formato* (+, -, espacio en blanco, 0 y #).

Se espera una variable por cada modificador de formato especificado.

Algunos formatos de ejemplo:

%X	Dirección de memoria de una variable
%.2f	Número de coma flotante con 2 decimales
%s	Cadena de texto (colección de caracteres)

Ejemplo de uso:

```
int numero = 15;
char cadena[] = "peso";

printf("El %s es de %.2f", cadena, numero);
```

Manejo de la entrada con scanf()

```
scanf("formato", variable_destino);
```

El formato se genera aplicando los modificadores y banderas explicados en «Manejo de salida».

Algunos formatos de ejemplo:

%[^n]s	Lee una cadena de texto hasta que se pulse <ENTER>
%50[^n]s	Lee los primeros 50 caracteres hasta que se pulse <ENTER>
%12s	Lee los primeros 12 caracteres

Ejemplo de uso:

```
char nombre[32];
scanf("%32[^n]s", nombre);

printf("Ha escrito: %s\n", nombre);
```

Manipulación de archivos

Apertura y cierre de archivos con fopen() y fclose()

fopen – apertura de archivos

```
FILE *archivo = fopen("<file>", "<modo>");
```

La función *fopen*, abre el archivo especificado en un modo concreto, y retorna un puntero del archivo (o puntero nulo si falla). Los modos de apertura posible se describen en la siguiente tabla:

Modo	Significado	Descripción
<i>r</i>	<i>read</i>	Abre un archivo existente para ser leído.
<i>r+</i>	<i>read / write</i>	Abre un archivo existente para ser leído y luego escrito.
<i>w</i>	<i>write</i>	Abre un archivo para ser escrito desde el byte 0. Esto significa que si el archivo existe, reemplazará su contenido. Si el archivo no existe, lo crea.
<i>w+</i>	<i>write / read</i>	Abre un archivo para ser escrito desde el byte 0 y luego leído. Al igual que en modo <i>write</i> , si el archivo existe, reemplazará su contenido, y si el archivo no existe, lo creará.
<i>a</i>	<i>append</i>	Abre un archivo para ser escrito desde el último byte. Esto significa que si el archivo existe, agrega el nuevo contenido al final del contenido actual. Si el archivo no existe, lo crea.
<i>a+</i>	<i>append / read</i>	Abre un archivo para ser escrito desde el último byte y luego leído. Al igual que en el modo <i>append</i> , si el archivo existe, agrega el nuevo contenido al final del contenido actual, y si no existe, lo crea.

9. Tabla: Modos de apertura de un archivo

fclose – cierre de archivos

```
int fclose(*<puntero>);
```

La función *fclose* cierra el canal abierto para un archivo con *fopen*, liberando la memoria utilizada en las operaciones previas. Si el cierre se produce sin fallos, retorna 0.

Ejemplo de uso:

```
FILE *archivo = fopen("/tmp/archivo_temporal.tmp", "w");
...
int resultado = fclose(archivo);
```

En el ejemplo anterior, el archivo “/tmp/archivo_temporal.tmp” se crea si no existe, generando un puntero hacia dicho fichero, denominado “archivo”. Al haber sido abierto en modo escritura, podrá escribirse en dicho fichero pero no podrá leerse. Finalmente, cierra el puntero “archivo”.

Lectura y escritura de archivos con *fread()* y *fwrite()*

fread – lectura de archivos

```
size_t fread(*<buffer>, size_t <N>, size_t <n>, *<puntero>);
```

La función *fread* lee por lo menos *n* objetos de los primeros *N* bytes del archivo, y los almacena en el bloque de memoria apuntado. El búfer de datos empleado, debe reservar la cantidad de memoria precisa para que *fread* realice su trabajo.

Si se quiere leer todo el archivo y se desconoce la cantidad de bytes, se puede mover el punto de acceso al final del archivo con *fseek* y obtener dicha posición con *ftell*.

Otras funciones de lectura: *fgets()*, *fgetc()*, *fgetwc()*.

fwrite – escritura de archivos

```
size_t fwrite(*<buffer>, size_t <N>, size_t <n>, *<puntero>);
```

A la inversa de *fread*, la función *fwrite* escribe por lo menos *n* objetos de los primeros *N* bytes del búfer de datos, en el archivo apuntado. Para obtener la longitud total del búfer de datos, puede utilizarse el *operador unario sizeof*.

Manipulación del punto de acceso interno con fseek() y ftell()

fseek – cambio de la posición del cursor

```
int fseek(*<puntero>, long <offset>, int <procedencia>);
```

La función *fseek* moverá el punto de acceso al archivo, la cantidad de bytes indicados en el *offset* a ser contabilizados desde la procedencia. En un archivo de texto plano, *offset* debe establecerse en 0. La *procedencia*, puede ser uno de tres valores:

```
SEEK_SET (comienzo del archivo, byte 0)  
SEEK_CUR (posición actual)  
SEEK_END (final del archivo)
```

ftell – obtención el punto de acceso actual

```
long ftell(*<puntero>);
```

La función *ftell* retorna el actual punto de acceso al archivo.

Ejemplos de lectura y escritura de archivos

Ejemplo de lectura de un archivo completo:

```
FILE *archivo = fopen("/tmp/archivo.tmp", "r");
```

```
fseek(archivo, 0, SEEK_END);
long longitud = ftell(archivo);
fseek(archivo, 0, SEEK_SET);

char buffer[longitud];
fread(buffer, longitud, 1, archivo);

fclose(archivo);
```

Ejemplo de escritura de un archivo:

```
char buffer[] = "Hola Mundo!\nAdiós Mundo!";

FILE *archivo = fopen("/tmp/archivo.tmp", "w");
long longitud = sizeof buffer;
fwrite(buffer, longitud, 1, archivo);
fclose(archivo);
```

Manipulación de argumentos de línea de comandos

Cuando un programa C es ejecutado por línea de comandos, la función *main* recibe de forma automática, 2 parámetros:

1. Un entero con la cantidad de argumentos recibidos por el programa.
2. Una matriz con el valor de cada argumento.

Los espacios en blanco separan argumentos. Si el valor de un argumento debe contener espacios en blanco, entonces, irá entre comillas.

```
usuario@host:~$ ./programa uno dos "tres y cuatro"
```

Estos dos argumentos, deben ser definidos en la función *main*, con cualquier nombre. Comúnmente se les denomina *argc* y *argv*, a pesar de que podrían llamarse, por ejemplo, *cantidad* y *argumentos*.

Los siguientes dos ejemplos son igualmente válidos:

```
void main(int argc, char *argv[])
void main (int cantidad, char *argumentos[])
```

Asignación dinámica de memoria con las funciones malloc() y free() de stdlib.h

Las funciones *malloc()* y *free()* de *stdlib.h* son aquellas que permiten asignar y liberar, respectivamente, memoria del montón, en tiempo de ejecución.

Observación léxica: en adelante, se emplea el término “**montón**” (y no “**montículo**”) para referirse al castellano del término “**heap**”, dado que la primera acepción del término “**montón**” es más precisa que la de “**montículo**”, representando así de forma concreta lo que se pretende denotar: «Conjunto de cosas puestas sin orden unas encima de otras.» (Real Academia Española, 2017, 23º ed.).

Entendiendo el montón

El **montón (heap)** es la memoria no utilizada del ordenador. La **memoria asignada**, es aquella memoria del montón, que se encuentra reservada por un programa, mientras que la **memoria libre**, es aquella que aún no ha sido asignada. Tanto la asignación de memoria como su correspondiente liberación, se efectúan de forma dinámica, en tiempo de ejecución.

Funcionamiento del montón

El montón es un apilamiento de memoria único que se comparte de forma simultánea, por el sistema operativo y por los programas que en él se estén ejecutando por todos los usuarios.

Para entender su funcionamiento, puede imaginarse al montón, como a un mazo de cartas boca abajo, y al sistemas operativo y a los programas, como a los jugadores. A medida que cada jugador necesite cartas, las irá tomando del mazo y cuando no las necesite, las irá descartando, devolviéndolas así al mismo mazo. Por la tanto, no puede saberse con exactitud qué cartas le tocarán a qué jugador, ni qué cartas estarán en qué lugar preciso del mazo.

Esto mismo sucede con el montón. De la misma forma que cuando un jugador necesita tres cartas no las elige sino que toma las tres primeras del maso, cuando se reservan 100 *bytes* del montón, se asignarán 100 *bytes* de forma consecutiva. Sin embargo, al solicitar los siguientes 100 *bytes*, no necesariamente serán consecutivos a los primeros, ya que entre cada una de las solicitudes, otro programa pudo haber pedido lo suyo.

De esta forma, cada una de las reservas del montón es otra **pila** de memoria a la cuál se recorrerá desde el primer elemento para llegar al último, al igual que se haría con un mazo de cartas más pequeño, como el que cada jugador tendrá consigo.

Utilizar el montón

En C es preciso definir de antemano todas las variables que serán necesarias. Cuando lo que se necesita es una matriz, solo puede definirse el número exacto de elementos si se conoce dicha cantidad. De lo contrario, se reservará un número aproximado o estimativo.

Si se sabe que una matriz tendrá 1000 enteros, se pide a C reservar el espacio de memoria equivalente a 1000 enteros:

```
short int enteros[1000];
```

El tipo *short int* reserva 2 *bytes* por cada elemento. Es decir: $1000 \times 2 = 2000$ *bytes*. Sin embargo, si la reserva anterior se hubiese llevado de forma estimativa y no exacta, y finalmente, se utilizasen menos *bytes* de los reservados, se estaría creando un programa que consumiría mayor cantidad de recursos de los necesarios. También podría suceder que el tamaño reservado resulte ser insuficiente. En ese caso, sería necesario modificar dicho valor en el código fuente y volver a compilar el programa.

Por otra parte, la memoria reservada para la matriz, no puede ser liberada a lo largo de la ejecución del programa, por lo que si solo fuese necesario hacer uso de dicha matriz al comienzo del programa, esa memoria no podría ser aprovechada por otro programa.

Recurrir al montón, permite asignar memoria de forma dinámica e ir liberándola a medida que ya no sea necesaria.

En C, la memoria del montón se reserva en punteros:

```
tipo *puntero = malloc(bytes);
```

Liberar memoria

Todo puntero cuya memoria haya sido reservada mediante *malloc()* requiere que la memoria sea liberada cuando ya no sea necesaria:

```
free(puntero);
```

malloc() y typecast

Dado que *malloc()* siempre retorna un puntero de tipo *char*, si lo que se desea utilizar no es de tipo *char*, tendrá que hacerse una **conversión de tipo (typecast)**:

```
int bytes = 100 * sizeof (int); // tamaño equivalente a 100 enteros
int *enteros = (int *) malloc(bytes);
```

El *** dentro del *typecast* es necesario dado que la variable *enteros* es un puntero, no un valor.

Comprobación del puntero

Es importante notar que *malloc()* no podrá asignar memoria si **no hay memoria libre** en el montón, por lo que se hará necesario constatar dicha reserva antes de confiar en el puntero:

```
char *cadena = malloc(1024);

if(cadena == NULL) {
    printf("No hay memoria disponible\n");
    exit(1); // sale del programa
}
```

Referencias

- [1] D. M. Ritchie, B. W. Kernighan. The C Programming Language, 2nd. ed. New Jersey York: Prentice Hall, 1988.
- [3] E. Huss. The C Library Reference Guide. Illinois: Eric Huss, 1997.
- [2] T. Crawford, P. Prinz. C in a Nutshell. Cambridge: O'Reilly Media, 2005.
- [3] Real Academia Española. (2017). Montón. En Diccionario de la lengua española (23º ed.). Recuperado de <http://dle.rae.es/?id=PkZlcXO>

PARTE III: 9. SEGURIDAD DE LA INFORMACIÓN

PREVENCIÓN DE CIBERATAQUES EN EL SECTOR DE SALUD

Florencia Paula Vilardel Tignanelli

Resumen

Es de observar que las nuevas tecnologías digitales permiten optimizar los procesos, las herramientas y experiencias de los usuarios del sector sanitario.

La historia clínica de los pacientes en formato electrónico conlleva la necesidad de reforzar la seguridad de los datos de salud a fin de lograr mantener la privacidad y confidencialidad de la información.

A pesar de la gran cantidad de medidas de seguridad, técnicas y recomendaciones existentes para el ámbito de la sanidad, parece existir un aumento considerable de los ciberataques no fallidos a los centros de salud, como consecuencia de errores o descuidos humanos relacionados a la ciberseguridad. Este trabajo pretende abordar las distintas herramientas, modelos y recomendaciones en las que las entidades sanitarias podrían

apoyarse para prevenir el acceso, uso, divulgación, alteración, modificación o destrucción no autorizada de la información, como resultado de ciberataques.

Introducción

El aprovechamiento de avances tecnológicos ha sido objeto de entidades sanitarias, con el fin de mejorar la calidad y eficacia de la atención, la investigación y la resolución de temas de salud.

Existen equipos médicos conectados de forma continua a internet, registrando la salud del paciente y emitiendo alertas en caso de riesgo. Por otra parte, la digitalización de las historias clínicas permite al personal de sanidad tener acceso a los datos e información de los pacientes en tiempo real, o conocer los resultados de los estudios médicos sin estar físicamente presentes, algo que podría facilitar la detección temprana de enfermedades.

Los ciberataques a la sanidad ponen en riesgo los datos de los pacientes y del propio personal. Considerar la aplicación de controles de seguridad, utilizando metodologías, normativas y mejores prácticas avaladas internacionalmente, podría permitir proteger la información de salud, a fin de garantizar tanto la confidencialidad e integridad de los datos, como la disponibilidad de los sistemas de información de salud críticos. Como resultado, se podría esperar una reducción del número y de la gravedad de los incidentes de seguridad.

Confidencialidad, integridad y disponibilidad de los datos

En ciberseguridad, se definen tres pilares fundamentales relacionados a los sistemas de información en el sector de la salud, los cuales son descritos a continuación.

Confidencialidad: se refiere a que la información pueda ser accedida sólo por personal autorizado y de manera autorizada. Busca que la información no sea revelada sin autorización. Asimismo, se transforma en *privacidad* al buscar proteger la información de los individuos, por ejemplo, cuando se habla de datos personales (Ley Argentina 25.326 – *Habeas Data*)⁵⁵.

Para el entorno sanitario, es el principio básico de la política de seguridad porque obliga a todo el personal a no revelar información suministrada por el paciente.

Integridad: se refiere a que la modificación de la información sólo pueda ser realizada por personas autorizadas a través de métodos autorizados. Garantiza exactitud y fiabilidad de la información y de los sistemas. También, previene accesos no autorizados que modifiquen la información (en inglés, *Alteration*) y protege los datos mediante mecanismos de encriptado, a través de funciones *hash*.

⁵⁵ Ley 25.326 – Habeas Data – Datos Personales, sancionada el 04 de octubre de 2000. Disponible en: https://www.oas.org/juridico/PDFs/arg_ley25326.pdf

Por razones de seguridad, la modificación no autorizada de la información médica es un factor de alerta que puede repercutir en problemas médicos.

Disponibilidad: se refiere a que la información y los datos se encuentran accesibles a todo el personal autorizado cuando se necesite, y se previene de pérdidas de la información y/o recursos.

El acceso a la información en un instante preciso es esencial en sanidad, dado que, en caso de una urgencia médica, no poder disponer de ella puede conllevar a resultados dramáticos para la salud del paciente.

Para el aseguramiento de estos tres pilares, es necesario considerar el factor humano, dado que con frecuencia, algunos problemas e incidentes de seguridad se presentan por errores o descuidos por parte del personal que trabaja en la industria de la salud.

Amenazas a la seguridad en ámbitos de la salud

Según un estudio realizado por investigadores españoles⁵⁶ (Sánchez-Henarejosa et. al, 2014) y el “Anexo A” de la norma internacional ISO/IEC 27799⁵⁷, las amenazas a las organizaciones de salud pueden agruparse en seis grupos:

1. *Divulgación de información confidencial a un receptor no autorizado* de forma no intencional o accidental. Se trata de un empleado de la

⁵⁶ Ana Sánchez-Henarejosa, José Luis Fernández-Alemána, Ambrosio Tovala, Isabel Hernández-Hernándezb, Ana Belén Sánchez-Garcíab y Juan Manuel Carrillo de Geaa - Guía de buenas prácticas de seguridad informática en el tratamiento de datos de salud para el personal sanitario en atención primaria. Artículo Especial- 2014;46(4):214--222

⁵⁷ Norma ISO/IEC 27799. Disponible en <https://www.iso.org/standard/62777.html>

sanidad que revela información de un paciente a otro o a una persona externa. Este tipo de amenaza es una falla en los controles de los empleados y personal de seguridad, especialmente en relación al nivel de conciencia sobre temas en seguridad y ciberseguridad.

2. *Acceso legítimo a los datos de un paciente con fines ilegítimos.* Puede configurar un robo o divulgación de información confidencial por parte de personal interno que tiene privilegios de acceso, ya sea que lo realizó por curiosidad o para sus propios fines con ánimo de lucro.
3. *Violación de la privacidad de los datos de los pacientes por un agente externo* que ingresa en la institución sanitaria de forma física y, de manera forzada, accede al sistema.
4. *Intrusión por fallas de seguridad en los sistemas informáticos.* Bien sea mediante la propagación de virus, o bien, por la inserción de código malicioso en los dispositivos a través de la proliferación de gusanos en correos electrónicos, que facilitarán la explotación de vulnerabilidades por parte de ciberdelincuentes, debido a fallas en los sistemas de detección y/o prevención de amenazas y/o intrusiones, o en el control de cambios de los programas.
5. *Sustitución de la identidad de personal autorizado.* Cuando una persona no autorizada se hace pasar por una que sí lo está, a fin de obtener acceso a los datos o recursos del sistema, de forma fraudulenta.

6. *Intrusión no autorizada en la red del sistema.* Cuando una persona externa (exemplar, paciente o ciberdelincuente) se introduce en la red del sistema de la organización desde el exterior y accede a la información del paciente o hace que el sistema deje de funcionar (como puede ser un ataque a la disponibilidad).

Motivos y Consecuencias

Possiblemente, los ciberdelincuentes sientan atracción a atacar a las instituciones en salud por toda la información sensible y confidencial que estas manejan. Algunas empresas fabricantes de Software destinado a prevenir ataques informáticos sobre sistemas operativos no libres, dicen creer que existen indicios⁵⁸ de que uno de los motivos para atacar los sistemas informáticos de un hospital, es la alta demanda y el valor que la información de los pacientes posee en los mercados clandestinos y, que además, puede emplearse con fines extorsivos.

Otro motivo por el cual un ciberdelincuente puede tener como objetivo atacar un centro de salud, es la posibilidad de realizar espionaje corporativo a través de la información que la institución posee. Esto, podría provocar un impacto negativo en la imagen de la organización, generando la pérdida de confianza de los pacientes. La información clínica de un paciente es un dato sensible⁵⁹ y

58 Kaspersky. “Boletín de Seguridad Kaspersky. Predicciones sobre amenazas para el 2018” (15 de noviembre de 2017). Disponible en https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2017/11/12102109/KSB_Predictions_2018_sp.pdf

59 Ley 25.326 – Habeas Data – Datos Personales, sancionada el 04 de octubre de 2000. Disponible en: https://www.oas.org/juridico/PDFs/arg_ley25326.pdf

el robo o su incorrecta utilización puede tener consecuencias sobre la seguridad del paciente, puesto que la manipulación de los datos de los mismos podría derivar incluso, en diagnósticos y tratamientos erróneos.

Los ciberataques podrían dirigirse a equipos conectados a internet como implantes, marcapasos o bombas de insulinas, entre otros, con el fin de sustraer información médica y datos de pacientes, secuestrar información y/o inhabilitar el acceso a los dispositivos (*ransomware*), o inhabilitar servicios (denegación de servicios), por citar algunos.

Recomendaciones

El objetivo es que los incidentes en ciberseguridad que puedan ocurrir en una institución del sector de la salud, no sean el elemento que frene el proceso de informatización de un Hospital. Controlar y proteger la información implica dar prioridad a la ciberseguridad desde el sector más alto de la institución, dado que afecta a cualquier ámbito y es clave para generar confianza en los pacientes, más aún cuando estos depositan sus datos más confidenciales, como el de la información clínica. La evaluación de los potenciales riesgos de ciberseguridad a los que la institución se enfrenta, se hace necesaria para el descubrimiento de amenazas y vulnerabilidades.

Los centros de salud pueden implementar, dentro de la organización, las siguientes recomendaciones para evitar y/o hacer frente al acceso no

autorizado, uso, divulgación, alteración, modificación no autorizada de la información como consecuencias de ciberataques:

- Educación y concienciación periódica y con material estandarizado, sobre los riesgos en ciberseguridad tanto del personal médico, como del de toda persona que trabaje dentro de la organización de salud.
- A fin de evitar la suplantación de identidad, las contraseñas no deben ser compartidas por ningún medio.
- Comprender las amenazas, los diferentes ataques potenciales, e identificar sus consecuencias a fin de tener herramientas para poder afrontarlas.
- Realizar auditorías de seguridad de forma periódica.
- Realizar copias de seguridad periódicas para minimizar los efectos del secuestro extorsivo de la información.
- Disponer de herramientas de seguridad (cortafuegos, antivirus, *antispam*, *antimalware*) actualizadas.
- Utilizar cuentas de correo corporativas para el intercambio de datos de salud de los pacientes.

- Cumplir con la *Ley 25.326 de Protección de Datos Personales* para Argentina⁶⁰ o sus equivalentes para otros países.
- Evitar la conexión de dispositivos extraíbles en los equipos del centro de salud dado que en algunos sistemas operativos, puede suponer un alto riesgo de entrada de virus.
- Establecer un plan completo de respuesta en términos de seguridad informática.
- Descargar siempre el software de los sitios/repositorios oficiales.
- Implementar acuerdos de confidencialidad en los que se especifique el carácter confidencial y sensible de la información, aplicable a todo el personal con acceso a la información de salud y que haga referencia a sanciones en su violación.
- Utilizar perímetros de seguridad física para proteger las áreas que contienen instalaciones de procesamiento de información que soporten las aplicaciones de salud, garantizando que sólo el personal autorizado pueda ingresar.

Debe tenerse en cuenta que mantener la confidencialidad, integridad y disponibilidad de la información sobre la salud de los pacientes, previene

60 Ley 25.326 – Habeas Data – Datos Personales, sancionada el 04 de octubre de 2000. Disponible en: https://www.oas.org/juridico/PDFs/arg_ley25326.pdf

errores en la práctica médica, garantiza la continuidad de los servicios médicos y protege la información que puede ser objeto de auditoría e imputabilidad.

Marcos o modelos de trabajo en Salud

Además de las recomendaciones que una organización dedicada a la salud puede implementar, existen guías de buenas prácticas, normas internacionales y organismos dedicados a la mejora en la calidad de un sistema de gestión sanitaria como así también, en la calidad de los productos médicos que se desarrollan y salen a la venta. La siguiente lista, es un compendio de algunas de ellas:

- a. **ISO 13485: Gestión de la calidad en dispositivos médicos o sanitarios.** Especifica los requisitos de un sistema de gestión de la calidad cuando una organización precisa demostrar su capacidad de proporcionar productos sanitarios o médicos y servicios relacionados que cumplen, de forma coherente, los requerimientos del cliente y reglamentarios, aplicables a los productos de sanidad y a los servicios relacionados. Todos los requisitos de la ISO 13485 son específicos para las organizaciones que suministran productos sanitarios, sea cual fuere el tipo o tamaño de la organización.

La norma incluye algunos requerimientos particulares para productos sanitarios y excluye alguno de los requisitos de la Norma ISO 9001 (que no son apropiados como requisitos obligatorios). Debido a estas exclusiones, las organizaciones cuyos sistemas de gestión de la calidad cumplen esta norma internacional no pueden declarar la conformidad

con la Norma ISO 9001 a menos que sus sistemas de gestión de la calidad cumplan todos los requisitos de la Norma ISO 9001⁶¹.

- b. **ISO 14971: Gestión de Riesgos de productos de la salud o médicos.** En esta norma se establecen los requerimientos de la gestión de riesgos para determinar la seguridad de un producto médico por parte del fabricante durante todo el ciclo de vida del producto⁶².

La norma incluye el análisis del riesgo, una evaluación, control, aceptación de los riesgos globales, informe de la gestión de riesgos del producto e informe de la producción y post producción.

- c. **ISO/IEC 27799: Seguridad de la Información en el sector de salud.** Orienta a las organizaciones en salud y aquellos custodios de información personal de salud, sobre la mejor manera de proteger la confidencialidad, integridad y disponibilidad de esa información, mediante la aplicación de la norma ISO/IEC 27002. La norma, garantiza que se mantiene la confidencialidad y la integridad de los datos a su cuidado, la disponibilidad de los sistemas de información de salud y la imputabilidad de ella. La aplicación de esta norma tiene por objetivo la reducción del número y de la gravedad de los incidentes de seguridad⁶³.

61 Concepto de la Norma ISO 13485. Disponible en https://es.wikipedia.org/wiki/ISO_13485

62 Concepto de la Norma ISO 14971. Disponible en https://es.wikipedia.org/wiki/ISO_14971

63 Norma ISO/IEC 27799. Disponible en <https://www.iso.org/standard/62777.html>

d. **HIPPA - Ley de Responsabilidad y Transferibilidad de Seguros Médicos (*Health Insurance Portability and Accountability Act*).** Se trata de una Ley Federal de los Estados Unidos de América, para cumplir con las garantías administrativas, técnicas y físicas necesarias para proteger la privacidad de la información de los clientes y mantener la integridad de los datos de los empleados, los clientes y los accionistas.

La ley exige que los proveedores de atención médica y los planes de seguro médico también protejan la privacidad de la información de salud del paciente. Los expedientes médicos se deben guardar bajo llave y estar disponibles únicamente cuando sea necesario⁶⁴.

e. **FDA (*Food and Drug Administration - Administración de Medicamentos y Alimentos*).** Es la Agencia del Gobierno de los Estados Unidos, que protege la salud pública al garantizar la eficacia y seguridad de los medicamentos humanos y veterinarios, productos biológicos y dispositivos médicos; y garantizando la seguridad del suministro de alimentos, cosméticos y productos que emiten radiación, dentro el país. El objetivo principal de la FDA es regular los productos médicos de una manera tal que se garantice la seguridad a

64 Ley Federal de los EEUU, HIPPA. Disponible en <https://www.hhs.gov/hipaa/index.html>

los consumidores estadounidenses y la efectividad de los medicamentos comercializados⁶⁵.

Además, la FDA tiene la responsabilidad de regular la fabricación, comercialización y distribución de productos de tabaco para proteger la salud pública y reducir su consumo por parte de los menores de edad.

- f. **ANMAT (Administración Nacional de Medicamentos, Alimentos y Tecnología Médica).** Es un organismo descentralizado de la Administración Pública Nacional en Argentina, creado en agosto de 1992, mediante el decreto 1490/92. Colabora en la protección de la salud humana, garantizando que los medicamentos, alimentos y dispositivos médicos a disposición de los ciudadanos posean eficacia (que cumplan su objetivo terapéutico, nutricional o diagnóstico) seguridad (alto coeficiente beneficio/riesgo) y calidad (que respondan a las necesidades y expectativas de la población). Para ello, se encarga de llevar adelante los procesos de autorización, registro, normalización, vigilancia y fiscalización de los productos de su competencia en todo el territorio nacional.

65 FDA (Food and Drug Administration). Disponible en https://es.wikipedia.org/wiki/Administraci%C3%B3n_de_Alimentos_y_Medicamentos y <https://www.fda.gov/>

La ANMAT depende técnica y científicamente de las normas y directivas que le imparte la Secretaría de Políticas, Regulación e Institutos de la Secretaría de Salud, con un régimen de autarquía económica y financiera⁶⁶.

Conclusión

A pesar de los esfuerzos para evitar ciberataques, las industrias sanitarias no logran, en algunos casos, frenarlos ni evitarlos. El contar con presupuesto para la compra y/o contratación de *software* y *hardware* que permita detectar estos ciberataques, la elaboración de una Política de Seguridad con procedimientos, normativas y estándares disponibles para la protección de los datos y los activos tecnológicos resulta de referencia para los trabajadores. Se hace necesaria la formación de todo el personal que trabaja en las instituciones de salud, como también, el construir hábitos y buenas conductas en materia de ciberseguridad, comprometerse a evaluar y auditar la seguridad informática y orientar a las personas, tanto externas como internas, en la mejora continua.

66 ANMAT (Administración Nacional de Medicamentos, Alimentos y Tecnología Médica). Disponible en <https://www.argentina.gob.ar/anmat>

PARTE III: 10. SEGURIDAD INFORMÁTICA

APROXIMACIÓN A UN SISTEMA DE CODIFICACIÓN PARA EL TRATAMIENTO DECIMAL DE CARACTERES UNICODE

Eugenia Bahit

Objetivo. El presente trabajo pretende servir como aproximación teórica a un sistema de codificación de caracteres en base 10 para la validación aritmética de datos.

Definiciones previas: Unicode y UTF-8

Unicode es un estándar que asigna un número único a cada carácter existente, independientemente del alfabeto o la plataforma sobre la que se opere.

UTF-8 es un sistema de codificación de caracteres que permite el uso de *Unicode* mediante representaciones de una a cuatro secuencias de 8 bits, por lo que un carácter *Unicode*, codificado en UTF-8, puede tener entre 8 y 32 bits (es decir, entre 1 y 4 *bytes*).

En *Unicode*, a cada número único se lo denomina *punto de código*. Un rango continuo de puntos de código en la tabla *Unicode*, se denomina *bloque*, y posee una denominación descriptiva tal como pueden ser «*Basic Latin*» o «*Latin-1 supplemnt*», bloques que serán utilizados en lo sucesivo.

Los primeros 128 puntos de código *Unicode* (desde U+0000 a U+007F) corresponden al bloque «*Basic Latin*», los cuáles codificados en UTF-8, emplean 8 bits.

Los siguientes 128 puntos de código (desde U+0080 a U+07FF), se corresponden al segundo bloque *Unicode*, «*Latin-1 supplemnt*», que en su codificación UTF-8 emplean 16 bits.

En adelante, toda referencia a *Unicode* se hará sobre los caracteres imprimibles de los primeros 256 caracteres *Unicode*, codificados con UTF-8. Es decir, que sólo se hará referencia y se trabajará sobre caracteres de 1 y 2 bytes.

Necesidad de codificar datos de entrada y manipularlos aritméticamente

En ciencias en general, el origen de toda investigación surge de una pregunta sin respuesta, que busca hallar una, tan exacta como sea posible. La necesidad de codificar datos y manipularlos de forma aritmética nace de la pregunta que diera origen al presente estudio: ¿es posible que un sistema informático admita el ingreso indiscriminado de caracteres sin dejar de ser seguro?

Se ha buscado responder a esta pregunta a través de respuestas lógicas, teniendo en cuenta el principio de simplicidad.

Necesidad de transmitir datos codificados de forma alfanumérica

En los sistemas de autenticación, permitir al usuario utilizar todo tipo de caracteres *Unicode*, tanto en el nombre de usuario como en la contraseña, refuerza la seguridad externa de la aplicación haciendo más difícil la efectividad de ataques por fuerza bruta o actos de ingeniería social, tendientes a descubrir (o inducir), la contraseña de un usuario a partir de un perfil victimológico.

Un nombre de usuario seguro y fácil de recordar para alguien que programa en PHP y gusta del fútbol, podría ser:

```
<?php echo "P3k3rm4n" ?>
```

Sin embargo, trabajar en bruto con caracteres *Unicode* de forma indiscriminada, representa un riesgo de seguridad, puesto que se podría inyectar código dejando al descubierto, una vulnerabilidad en el sistema que podría ser explotada.

Una forma simple de permitir al usuario operar con todo tipo de caracteres sin restricción, podría consistir en reducir al mínimo la cantidad de símbolos y signos de puntuación a manipular en bruto en una cadena, codificando los datos antes de que estos sean transmitidos.

Esto podría lograrse, en principio, mediante una codificación de los datos con Base64, ya que Base64 solo emplea tres caracteres no alfanuméricos: el signo de adición “+”, la barra diagonal “/” y como relleno, el signo de igualdad “=”.

Necesidad de efectuar validaciones aritméticas

Mientras que la codificación de datos se hace necesaria para reforzar la seguridad del sistema (y como efecto colateral, aumentar la facilidad de uso de la aplicación, puesto que el usuario no necesitaría aprender reglas o someterse al cumplimiento de normas que restrinjan el uso de los datos que se deseen manipular), la validación aritmética de esos datos, representa un medio matemáticamente exacto de diseñar reglas de validación, en especial, aquellas cuyo fin sea la verificación del cumplimiento de políticas de nombres de usuario y contraseña.

Necesidad de un tratamiento decimal de los datos

Si bien la codificación de los datos facilita el almacenamiento y manipulación de estos, no facilita la disposición de los mismos de forma humanamente legible, puesto que codificados no son humanamente viables, y decodificados, no son informáticamente seguros. Manipular los datos no seguros (entendiendo por dato no seguro a todo carácter que no sea alfanumérico⁶⁷) de forma decimal, facilita las validaciones aritméticas al mismo tiempo que su representación visual por medio de entidades HTML decimales.

⁶⁷ https://www.researchgate.net/publication/333967818_Alnum_alphanumeric_character_encoding_system_CIDSI_2018

Evaluación de Base64 como medio para el tratamiento aritmético de datos

Si bien Base64 no ofrece una codificación 100% alfanumérica, se estudia como posibilidad, dado su bajo índice de simbología no segura.

Sobre Base64

La codificación Base64 consiste en una tabla de 64 caracteres, formada por las letras mayúsculas y minúsculas de la “A” a la “Z”, los dígitos del “0” al “9”, el signo “+” y la barra diagonal “/”. Un carácter adicional se utiliza además, como carácter de relleno, este es el signo “=”.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/

10. Tabla: Tabla de codificación Base64

Mecanismo de codificación

Para codificar en Base64, los datos de entrada se dividen, de izquierda a derecha, en 3 grupos de 8 bits cada uno.

Cada 3 grupos de 8 bits, se dividen los 24 bits en 4 grupos de 6 bits cada uno.

A cada grupo de 6 bits, se lo convierte a decimal y se reemplaza el decimal por el valor de equivalencia de la tabla de codificación Base64.

Cuando no pueden completarse 24 bits, se llenan los bits faltantes con ceros, y cada grupo de 8 bits faltantes (es decir, de ceros) se lo reemplaza con un signo “=”.

Cadena de entrada:	A
Valor equivalente de 8 bits:	01000001
Relleno de bits faltantes:	01000001 00000000 00000000
Reagrupación de 6 bits:	010000 010000 000000 000000
Equivalencia decimal:	16 16
Equivalencia Base64:	Q Q
Equivalencia Base64 y relleno:	Q Q = =

Codificación de caracteres Unicode en Base64 del lado del cliente

Cuando del lado del cliente se trabaje con JavaScript, la codificación en Base64 de caracteres *Unicode*, podría resultar en un fallo de “carácter fuera de rango” si el carácter excediese los 8 bits (*Mozilla Developer Network, Base64 encoding and decoding, “The Unicode Problem”*. Consultado el 19/05/2018 en https://developer.mozilla.org/en-US/docs/Web/API/WindowBase64/Base64_encoding_and_decoding#The_Unicode_Problem).

Si dicho error⁶⁸ se produjese, podría ser subsanado aplicando cualquiera de las dos soluciones propuestas por Mozilla:

1. Escapar la cadena antes de codificarla.
2. Sobrescribir los métodos *atob()* y *boat()* del DOM, lo que implicaría un mayor costo sobre los recursos del equipo.

Se copia a continuación, la solución propuesta por Mozilla para la codificación *Unicode* en Base64, escapando los caracteres antes de codificarlos:

```
function b64EncodeUnicode(str) {  
    return btoa(  
        encodeURIComponent(str).replace(  
            /[0-9A-F]{2}/g,  
            function toSolidBytes(match, p1) {  
                return String.fromCharCode('0x' + p1)  
            }));  
}
```

Salida utilizando la solución de Mozilla:

```
b64EncodeUnicode('Ñandú!') // "w5FhbmTDuiE="
```

Codificación de caracteres Unicode en Base64 del lado del servidor

A nivel servidor, la codificación de caracteres *Unicode* con Base64 no parece presentar problemas que requieran un tratamiento especial.

68 Se intentó reproducir el fallo reportado por Mozilla, en un navegador Firefox Quantum versión 60.0.1 (64-bits) operando sobre un sistema Debian GNU/Linux 9.11 de 64 bits, sin obtener resultados que permitiesen constatar el error reportado.

A continuación, se muestra la salida de dos ejemplos escritos en Python y PHP, respectivamente.

```
eugenia@hdmag:~$ python
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
...
>>> from base64 import b64encode
>>> cadena = "Ñandú!"
>>> print(b64encode(cadena)) # Difiere en Python 3+
w5FhbmtDuiE=
```

```
eugenia@hdmag:~$ php -a
Interactive mode enabled

php > $cadena = "Ñandú!";
php > print base64_encode($cadena);
w5FhbmtDuiE=
```

Validación de datos a partir de cadenas codificadas con Base64

Al recibir datos codificados con Base64, y buscar hallar un método de reversión que permita una validación adecuada de los datos, se plantearon tres dificultades:

1. Dificultad para revertir la cadena a una representación visual humanamente legible distinta a la cadena original
2. Dificultad para determinar de forma fehaciente si una cadena se encuentra codificada con Base64
3. Dificultad para revertir la cadena a su equivalente decimal

Se explica cada una de ellas en el siguiente apartado.

Dificultades halladas en el uso de Base64

La cadena se puede almacenar codificada, pero debe proveer un mecanismo de decodificación seguro que la haga humanamente legible para el usuario. Base64 puede ser revertido a su original (método inseguro) o podría convertirse a otro sistema. Para convertirse a otro sistema, se hace necesario ir hacia atrás en el proceso de codificación previo. Al hacerlo, llegamos al decimal (el cuál no corresponde a la entrada original), a los grupos de 6 bits (que tampoco se corresponden) y a la reagrupación de 8 bits (primer sistema que sí se corresponde con la cadena original). A partir de allí, cada grupo de 8 bits, se podría convertir a su decimal correspondiente, y trasladado a su entidad HTML correspondiente, anteponiendo el prefijo “&#” al decimal, y “;” como sufijo.

Codificar los datos de entrada antes de la transmisión, solo es efectivo si al recibir los datos, se toma la precaución de sanearlos.

Esto se puede hacer, bien sea limpiando la cadena de caracteres no deseados (en el caso de Base64, cualquier carácter que no sea una letra, un número, el signo de relleno “=”, el signo “+” o la barra diagonal “/”) o volviendo a codificar los datos. La limpieza carece de sentido en este contexto, ya que lo que se busca es conservar la totalidad de los caracteres originales. Por lo tanto, codificar los datos del lado del servidor, sería lo más efectivo. Sin embargo, solo sería efectivo si la cadena llegase sin estar codificada. La dificultad se presenta al momento de verificar si la cadena se encuentra o no, realmente, codificada en Base64.

Se necesita una cadena codificada para a partir de ella, validar la cadena original. Una cadena codificada con Base64, no posee en apariencia, un mecanismo que permita la validación de la cadena original, dado que no siempre el mismo carácter Base64 (o la misma combinación de ellos), equivale al mismo carácter Unicode. Esto es debido a la reagrupación de bits.

```
php > print base64_encode('A');  
QQ==  
php > print base64_encode('AA');  
QUE=  
php > print base64_encode('LA');  
TEE=
```

La cadena codificada con Base64 debería ser revertida hasta un punto que permitiese la validación aritmética, como por ejemplo, el proceso de reversión antes mencionado, previo a la conversión a entidad HTML decimal (por ejemplo, si la matriz de decimales contiene un número entre 32 y 47, significa que la cadena original contiene un carácter no alfanumérico).

Reversión de una cadena codificada con Base64 hacia sus valores decimales correspondientes (Base64 a decimal)

Se parte de una cadena inicial simple, **YWJj** (Base64 correspondiente a los caracteres “abc”). Se evitan cadenas con caracteres de relleno para no emplear validaciones que compliquen el entendimiento del código fuente.

Base64 a binario

El **primer paso**, consiste en recuperar el decimal correspondiente a cada carácter, en la tabla de equivalencias de la codificación Base64.

Materiales:

1. Tabla de equivalencias de codificación Base64
2. Cadena de entrada codificada en Base64

Método (se emplea PHP como lenguaje de alto nivel):

```
# TABLA BASE64
$letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
$b64_table = array_merge(
    str_split($letters),                      # Letras mayúsculas
    str_split(strtolower($letters)),           # Letras minsúculas
    range(0, 9),                            # Números
    ['+', '/']                                # Signos
);

# CADENA DE ENTRADA CODIFICADA
$b64 = "YWJj";

# OBTENCIÓN DE EQUIVALENCIAS DECIMALES
$b64_arr_dec = array_map(
    function($e) use ($b64_table) {
        return array_search($e, $b64_table);
    },
    str_split($b64)
);
```

Resultado:

```
Array
(
    [0] => 24
    [1] => 22
    [2] => 9
    [3] => 35
)
```

Una vez obtenidos los decimales de Base64, el **segundo paso** consiste en obtener los grupos de 6 bits de cada decimal.

Materiales:

3. Matriz de decimales obtenida en el paso anterior

Método:

```
# OBTENCIÓN DE BINARIOS
$b64_arr_bin = array_map('decbin', $b64_arr_dec);

# RELLENO A GRUPOS DE 6 BITS (agrega ceros a la izquierda)
$b64_arr_bin6 = array_map(
    function($e) {
        return str_pad($e, 6, "0", STR_PAD_LEFT);
    },
    $b64_arr_bin
);
```

Resultado:

```
Array
(
    [0] => 011000
    [1] => 010110
    [2] => 001001
    [3] => 100011
)
```

El **tercer paso** consiste en reagrupar los bits en grupos de 8.

Materiales:

4. Matriz de grupos de 6 bits obtenida en el paso previo

Método:

```
$b64_bin6_str = implode('', $b64_arr_bin6);
$bin_arr = explode(' ', chunk_split($b64_bin6_str, 8, ' '));
```

Resultado:

El resultado de lo anterior es una matriz con los binarios correspondientes a cada carácter de la cadena original (el último elemento sobrante, se agrega con el *explode*, y podría haberse eliminado al finalizar el paso 3).

```
Array
(
    [0] => 01100001
    [1] => 01100010
    [2] => 01100011
    [3] =>
)
```

Binario a decimal

Se reproducen los pasos 1 a 3 del paso de Base64 a binario, y **finalmente**, cada grupo de 8 bits se convierte a su valor correspondiente en sistema decimal.

Materiales:

5. Matriz de binarios de 8 bits

Método:

```
$dec_arr = array_map('bindec', $bin_arr);
```

Resultado:

El resultado de lo anterior es una matriz con los decimales (el último elemento sobrante, se agrega con el *explode*, del paso 3 de la conversión Base64 a binario, y podría haberse eliminado al finalizar el paso 3).

```
Array
```

```
(  
    [0] => 97  
    [1] => 98  
    [2] => 99  
    [3] => 0  
)
```

Conclusión:

Como puede observarse, se trata de un procedimiento de alto coste, dada la cantidad de instrucciones que deben ser ejecutadas a alto nivel (siendo multiplicadas por el intérprete, en capas más cercanas al hardware).

Sin perjuicio de lo anterior, dado que no parece resultar posible la verificación del sistema de codificación de una cadena con Base64, no se puede inferir que la codificación con Base64 y su consecuente reversión para la obtención de decimales, resulte ser el método adecuado.

No obstante, nada indica que la obtención de una matriz de decimales, no resulte ser la adecuada para una manipulación aritmética de los datos, que facilite la validación de cadenas de entrada.

Lo antedicho, podría sugerir la necesidad de crear un nuevo sistema de codificación de caracteres que permita, por un lado, o bien reconocer si una cadena se encuentra codificada, o bien, que al aplicar una función de codificación, no modifique la cadena si esta se encontraba previamente codificada; por otro lado, revertir la cadena a un punto decimal que permita el tratamiento aritmético de los datos y la presentación de los mismos de modo humanamente legible, sin que ello represente un riesgo de seguridad (por ejemplo, convirtiendo los caracteres sensibles para el sistema, en entidades HTML decimales).

Aproximación a un nuevo sistema de codificación de caracteres que permita un tratamiento aritmético de los datos en lenguajes de alto nivel

De las «*Dificultades halladas en el uso de Base64*» surgen tres necesidades que deben ser satisfechas por un nuevo sistema de codificación de caracteres, las cuales se definen a continuación.

- Necesidad #1: la necesidad de una función de *conversión no acumulativa*, que permita codificar los datos para manipularlos de forma segura pero arrojando el mismo valor de salida, tanto si el valor de entrada estuviese previamente codificado como si no, de forma tal que se cumpla $f(x) \rightarrow y = f(y) \rightarrow y$, siendo x un valor de entrada no codificado, e y , un valor $f(x)$.
- Necesidad #2: la necesidad de una función que permita que los datos codificados puedan ser mostrados de forma humanamente legible pero sin volver al estado original de los mismos, de forma tal que se cumpla $f(y) \rightarrow z \wedge z \neq x$, siendo x un valor sin codificar; y un valor $f(x)$ codificado, y z un valor codificado humanamente legible distinto que x .
- Necesidad #3: la necesidad de una función que permita pasar a valores decimales, un valor previamente codificado $f(y) \rightarrow y_{10}$ (donde y es un valor codificado e y_{10} , su base decimal correspondiente).

Por lo tanto, siendo X la cadena de todos los caracteres x de entrada (donde $X = \{x_{i \dots n}\}$), en $f(x_i) \rightarrow y_i$ debe evitarse $y_i = x (x \in X)$, de modo tal que un valor codificado no pueda ser igual a un carácter sin codificar, debiéndose cumplir $\forall x \in X (X = \{x_{i \dots n}\}) \quad f(x) \rightarrow y \wedge y \neq x$.

Codificación alfanumérica de caracteres Unicode

Para evitar una conversión acumulativa que asegure el cumplimiento de $f(x) \rightarrow y \wedge f(y) \rightarrow y$, se propone la conversión de cualquier carácter no alfanumérico mediante una combinación de caracteres alfanuméricos de *probabilidad de frecuencia nula* en X .

Dicha *combinación de probabilidad de frecuencia nula Zf* se define como: $Zf \equiv H \circ I \circ d \circ E \circ H$, siendo I y E dos literales, H un valor obtenido mediante una función $f(X^{-1}) \rightarrow H$ (en la cual X^{-1} es el conjunto de todos los caracteres alfanuméricos no codificables de X), y d el número decimal del carácter no alfanumérico a codificarse, obtenido mediante una función $f(x) \rightarrow x_{10}$ (donde x es un carácter no alfanumérico a ser codificado, y x_{10} , su base decimal correspondiente).

La función $f(X^{-1}) \rightarrow H$ será una función *hash* cualquiera y para H se deberá cumplir $(H \circ I) \not\subseteq X \wedge (E \circ H) \not\subseteq X$.

Por lo tanto, se requiere una *función de conversión no acumulativa* $f(X) \rightarrow X_{Zf}$, donde X_{Zf} es la cadena codificada, que sirviendo de valor de

entrada a la misma función, arroje siempre X_{zf} , de forma tal que $f(X) \rightarrow X_{zf} = f(X_{zf}) \rightarrow X_{zf}$.

Cadena original	X	I'm	
Valor hash	H	0435377b	(digesto crc32 de 'Im')
Concatenación HI	$H \circ I$	0435377bI	
Concatenación EH	$E \circ H$	E0435377b	
Decimal de '	d	39	
Valor Zf	Zf	0435377bI39E0435377b	
Cadena codificada	X_{zf}	I0435377bI39E0435377bm	

En todo caso, la codificación tanto del lado del cliente como del lado del servidor, se debe llevar a cabo mediante una función de reemplazo, a partir de una tabla de equivalencias de caracteres no alfanuméricos y sus decimales correspondientes.

Caracteres de reemplazo	10	32 - 126	160 – 255
Caracteres de eliminación	<10	>10 y < 32	>255

11. Tabla: Valores decimales de los caracteres de reemplazo y eliminación sugeridos (incluidos el espacio en blanco y el salto de línea).

Resumen de funciones necesarias

Se resumen a continuación, las funciones de codificación necesarias y su aproximación en pseudocódigo.

Función	Descripción
$f(x) \rightarrow x_{10}$	Obtener valor decimal (base 10) de un carácter.

	<p><i>Entrada:</i> $\text{char } X$ carácter Unicode codificado en UTF-8</p> <p><i>Salida:</i> $\text{integer } x_{10}$ valor decimal correspondiente a X, si X no está en la tabla de caracteres alfanuméricos permitidos.</p>
$f(X) \rightarrow X^{-1}$	<p>Obtener caracteres alfanuméricos de una cadena en crudo.</p> <p><i>Entrada:</i> $\text{string } X$ cadena en crudo, codificada en UTF-8</p> <p><i>Salida:</i> $\text{string } X^{-1}$ caracteres alfanuméricos del valor de entrada.</p>
$f(X^{-1}) \rightarrow H$	<p>Obtener valor hash Zf.</p> <p><i>Entrada:</i> $\text{string } X^{-1}$ cadena alfanumérica obtenida mediante $f(X) \rightarrow X^{-1}$</p> <p><i>Salida:</i> $\text{string } H$ valor hash de la cadena de entrada.</p>
$f(X) \rightarrow X_{zf}$	<p>Obtener una cadena codificada de forma alfanumérica.</p> <p><i>Entrada:</i> $\text{string } X$ cadena en crudo, codificada en UTF-8</p> <p><i>Salida:</i> $\text{string } X_{zf}$ cadena codificada de forma alfanumérica.</p>

12. Tabla: Resumen de funciones de codificación alfanumérica.

Función	Implementación en pseudocódigo
$f(x) \rightarrow x_{10}$	<pre>char get_dec(char x): return (x in table) ? table[x] : 0</pre>
$f(X) \rightarrow X^{-1}$	<pre>string get_Xsup1(string X): string Xsup1 char x for x in X: if alnum(x): Xsup1 += x return Xsup1</pre>
$f(X^{-1}) \rightarrow H$	<pre>string get_H(string Xsup1): return hash(Xsup1).hex_digest()</pre>
$f(X) \rightarrow X_{zf}$	<pre>string zf_encode(string X): string Xsup1 = get_Xsup1(X) string H = get_H(Xsup1) string XsubZf char x for x in X: if alnum(x): continue XsubZf += get_dec(x) ? {H}I{get_dec(x)}E{H} : '' return XsubZf</pre>

13. Tabla: Pseudocódigo para funciones de codificación alfanuméricas.

Codificación HTML decimal

El paso de una cadena codificada X_{zf} a una cadena final Z (humanamente legible de forma segura), requiere una función $f(X_{zf}) \rightarrow Z$, encargada de pasar cada valor $Zf \in X_{zf}$ a su correspondiente entidad HTML decimal z , mediante una subrutina $f(y) \rightarrow z$ donde y es cada valor $Zf \in X_{zf}$.

Valor Zf	Zf	0435377bI39E0435377b
Entidad HTML decimal	z	'

Una función $f(y) \rightarrow z$ que recibe un valor Zf de entrada, sustituirá $H \circ I$ por los símbolos *ampersand* y numeral “ $\'$ mediante una función $f(i) \rightarrow i_p$ (donde $i = H \circ I$ e i_p es el prefijo formado por los caracteres “ $\'$ ”), y $E \circ H$ por el punto y coma “ $\,$ ”, a través de una función $f(e) \rightarrow e_s$ (donde $e = E \circ H$ y e_s es el sufijo “ $\,$ ”), de forma tal de poder formar una entidad HTML decimal válida.

Valor Zf	Zf	0435377bI39E0435377b
Reemplazo de prefijo	i_p	'0435377b
Reemplazo de sufijo	e_s	'

Resumen de funciones necesarias

Las siguientes especificaciones y su implementación en pseudocódigo, son modificadas en la sección «*Obtención del valor H a partir de una cadena X_{zf}* ».

Función	Descripción
$f(X_{zf}) \rightarrow Z$	<p>Obtener una cadena codificada en HTML a partir de una cadena codificada de forma alfanumérica.</p> <p><i>Entrada:</i> string X_{zf} cadena codificada de forma alfanumérica.</p> <p><i>Salida:</i> string Z cadena codificada con entidades HTML decimales.</p>

14. Tabla: Resumen de funciones de codificación HTML.

Función	Implementación en pseudocódigo
$f(X_{zf}) \rightarrow Z$	<pre>// require string H string z_encode(string XsubZf): string prefix = "{H}I" string sufix = "E{H}" string array from = [refix, sufix] string array to = ['&', '''] return replace(from, to, XsubZf)</pre>

15. Tabla: Pseudocódigo para funciones de codificación HTML.

Obtención del valor H a partir de una cadena X_{zf}

Un valor H utilizado en las concatenaciones Zf de una cadena X_{zf} se obtiene de la base alfanumérica de X , es decir, de X^{-1} .

El proceso de conversión inversa $f(X_{zf}) \rightarrow X$ que permita hallar X^{-1} mediante $f(X) \rightarrow X^{-1}$, requiere conocer H . Por lo tanto, no puede utilizarse un proceso de conversión inversa directa de X_{zf} para obtener H .

Se propone un método pragmático para la obtención de H , el cual supone el paso del valor *hash* de X^{-1} como ***token de validación de integridad t***. Dicho token podría pasarse como parte de la cadena, a partir del byte 0 con reserva de los bytes necesarios según la función hash empleada (por ejemplo, en el caso de crc32, serán necesarios 8 bytes).

Resumen de funciones necesarias

Las siguientes especificaciones y su implementación en pseudocódigo, modifican lo expuesto en la sección «*Codificación HTML decimal*».

Función	Descripción
$f(X_{zf}, t) \rightarrow Z$	<p>Obtener una cadena codificada en HTML a partir de una cadena codificada de forma alfanumérica.</p> <p><i>Entrada:</i> string X_{zf} cadena codificada de forma alfanumérica. string t <i>token</i> (valor <i>hash</i> Zf).</p> <p><i>Salida:</i> string Z cadena codificada con entidades HTML decimales.</p>

16. Tabla: Resumen de funciones de codificación HTML.

Función	Implementación en pseudo código
$f(X_{zf}, t) \rightarrow Z$	<pre style="font-family: monospace; color: black;"> string z_encode(string XsubZf, string t): string prefix = "{t}I" string sufix = "E{t}" string array from = [refix, sufix] string array to = ['&', ''] return replace(from, to, XsubZf) </pre>

17. Tabla: Pseudocódigo para funciones de codificación HTML.

Integridad de los datos

Integridad de la cadena

En un escenario ideal, se espera que los datos sean transmitidos de forma codificada. Es decir, que se espera recibir una cadena codificada X_{zf} que al ser utilizada como valor de entrada de una función $f(X) \rightarrow X_{zf}$ arroje un valor X_{zf} idéntico al recibido.

Por lo tanto, la integridad de una cadena X_{zf} se mantiene si se cumple $X_{zf} = f(X_{zf})$.

Integridad del token

Mediante una función $f(X_{zf}, t) \rightarrow X^{-1}$, se deberán hallar y excluir todas las apariciones de $t \circ I \circ d \circ E \circ t$ (donde d será la expresión regular “ $|d^+$ ”) en X_{zf} , y aplicar la función *hash* sobre el resultado $f(X^{-1}) \rightarrow H$. Se espera $t = H$ para afirmar la integridad de t . Si se cumple $t = H$, luego X^{-1} también es válida, de modo tal que $(f(X^{-1}) \rightarrow H) = t \Rightarrow X^{-1}$.

Resumen de rutinas necesarias

Las siguientes rutinas en pseudocódigo son necesarias para validar la integridad de la cadena y del *token* recibidos.

Rutina	Descripción
$X_{Zf} = f(X_{Zf})$	<p>Rutina. Comparar una cadena recibida con la codificación alfanumérica de la misma.</p> <p><i>Función necesaria:</i> <code>zfencode(X)</code> función de codificación alfanumérica <code>zf_encode()</code></p> <p><i>Variable necesaria:</i> <code>string X_{Zf}</code> cadena codificada de forma alfanumérica.</p> <p><i>Evaluación necesaria:</i> comparar igualdad entre la cadena recibida y el resultado de la función con la cadena recibida como valor de entrada.</p>
$f(X_{Zf}, t) \rightarrow X^{-1}$	<p>Obtener una cadena base a partir de la cadena recibida.</p> <p><i>Entrada:</i> <code>string X_{Zf}</code> cadena codificada de forma alfanumérica. <code>string t token</code> (valor <i>hash Zf</i>).</p> <p><i>Salida:</i> <code>string X⁻¹</code> cadena base de la cadena recibida.</p>

Rutina	Descripción
$f(X_{Zf}, t) \rightarrow X^{-1}$	<p>Obtener una cadena base a partir de la cadena recibida.</p> <p><i>Entrada:</i> string X_{Zf} cadena codificada de forma alfanumérica. string t token (valor hash Zf).</p> <p><i>Salida:</i> string X^{-1} cadena base de la cadena recibida.</p>
$(f(X^{-1}) \rightarrow H) = t \Rightarrow X^{-1}$	<p>Rutina. Comparar el <i>token</i> recibido con el <i>hash Zf</i> de la cadena base recibida.</p> <p><i>Función necesaria:</i> getH(X^{-1}) función de retorno de valor <i>hash Zf</i>, get_H()</p> <p><i>Variable necesaria:</i> string t token recibido.</p> <p><i>Evaluación necesaria:</i> comparar igualdad entre el <i>token</i> recibido y el resultado de la función con la cadena obtenida mediante $f(X_{Zf}, t) \rightarrow X^{-1}$ como valor de entrada.</p>

18. Tabla: Resumen de funciones de validación de integridad.

Función/Rutina	Implementación en pseudo código
$X_{Zf} = f(X_{Zf})$	<code>bool string_integrity = (zf_encode(XsubZf) == XsubZf)</code>
$f(X_{Zf}, t) \rightarrow X^{-1}$	<code>string get_base_Xsup1(string XsubZf, string t): string regex = "{t}I\dE{t}" return regex_replace(regex, '', XsubZf)</code>
$(f(X^{-1}) \rightarrow H) = t \Rightarrow X^{-1}$	<code>string Xsup1 = get_base_Xsup1(zf_encode(XsubZf), t) bool token_integrity = (get_H(Xsup1) == t)</code>

19. Tabla: Pseudocódigo para funciones y rutinas de validación de integridad.

Anexo A: Funciones PHP para la conversión de Base64 a binario y decimal

Desde Base64 a código binario:

```
function base64_to_binary($b64str) {
    $letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    $b64_table = array_merge(
        str_split($letters),
        str_split(strtolower($letters)),
        range(0, 9),
        ['+', '/']
    );
    $b64_arr_dec = array_map(
        function($e) use ($b64_table) {
            return array_search($e, $b64_table);
        },
        str_split($b64str)
    );
    $b64_arr_bin = array_map('decbin', $b64_arr_dec);
    $b64_arr_bin6 = array_map(
```

```
function($e) {
    return str_pad($e, 6, "0", STR_PAD_LEFT);
},
$b64_arr_bin
);

$b64_bin6_str = implode('', $b64_arr_bin6);
$bin_arr = explode(' ', chunk_split($b64_bin6_str, 8, '
'));

return $bin_arr;
}
```

Desde Base64 a decimal:

```
function base64_to_decimal($b64str) {
    $bin_arr = base64_to_binary($b64str);
    $dec_arr = array_map('bindec', $bin_arr);
    return $dec_arr;
}
```

Anexo B: caracteres Unicode de reemplazo y su equivalencia decimal en HTML

UNI	CHR	DEC	UNI	CHR	DEC	UNI	CHR	DEC
U+0020		32	U+002D	-	45	U+003A	:	58
U+0021	!	33	U+002E	.	46	U+003B	;	59
U+0022	"	34	U+002F	/	47	U+003C	<	60
U+0023	#	35	U+0030	0	48	U+003D	=	61
U+0024	\$	36	U+0031	1	49	U+003E	>	62
U+0025	%	37	U+0032	2	50	U+003F	?	63
U+0026	&	38	U+0033	3	51	U+0040	@	64
U+0027	'	39	U+0034	4	52	U+0041	A	65
U+0028	(40	U+0035	5	53	U+0042	B	66
U+0029)	41	U+0036	6	54	U+0043	C	67
U+002A	*	42	U+0037	7	55	U+0044	D	68
U+002B	+	43	U+0038	8	56	U+0045	E	69
U+002C	,	44	U+0039	9	57	U+0046	F	70

UNI	CHR	DEC	UNI	CHR	DEC	UNI	CHR	DEC
U+0047	G	71	U+0056	V	86	U+0065	e	101
U+0048	H	72	U+0057	W	87	U+0066	f	102
U+0049	I	73	U+0058	X	88	U+0067	g	103
U+004A	J	74	U+0059	Y	89	U+0068	h	104
U+004B	K	75	U+005A	Z	90	U+0069	i	105
U+004C	L	76	U+005B	[91	U+006A	j	106
U+004D	M	77	U+005C	\	92	U+006B	k	107
U+004E	N	78	U+005D]	93	U+006C	l	108
U+004F	O	79	U+005E	^	94	U+006D	m	109
U+0050	P	80	U+005F	_	95	U+006E	n	110
U+0051	Q	81	U+0060	`	96	U+006F	o	111
U+0052	R	82	U+0061	a	97	U+0070	p	112
U+0053	S	83	U+0062	b	98	U+0071	q	113
U+0054	T	84	U+0063	c	99	U+0072	r	114
U+0055	U	85	U+0064	d	100	U+0073	s	115

UNI	CHR	DEC	UNI	CHR	DEC	UNI	CHR	DEC
U+0074	₧	116	U+00A4	₩	164	U+00B3	₳	179
U+0075	₨	117	U+00A5	₪	165	U+00B4	₵	180
U+0076	₩	118	U+00A6	₫	166	U+00B5	₶	181
U+0077	₪	119	U+00A7	₷	167	U+00B6	₸	182
U+0078	₫	120	U+00A8	₯	168	U+00B7	₹	183
U+0079	€	121	U+00A9	₵	169	U+00B8	₺	184
U+007A	₷	122	U+00AA	₸	170	U+00B9	₻	185
U+007B	{	123	U+00AB	«	171	U+00BA	₼	186
U+007C		124	U+00AC	¬	172	U+00BB	»	187
U+007D	}	125	U+00AD		173	U+00BC	¼	188
U+007E	~	126	U+00AE	®	174	U+00BD	½	189
U+00A0		160	U+00AF	-	175	U+00BE	¾	190
U+00A1	ᵫ	161	U+00B0	°	176	U+00BF	ᵦ	191
U+00A2	₵	162	U+00B1	±	177	U+00C0	ᵾ	192
U+00A3	₭	163	U+00B2	²	178	U+00C1	ᵿ	193

UNI	CHR	DEC	UNI	CHR	DEC	UNI	CHR	DEC
U+00C2	Â	194	U+00D1	Ñ	209	U+00E0	à	224
U+00C3	Ã	195	U+00D2	Ò	210	U+00E1	á	225
U+00C4	Ä	196	U+00D3	Ó	211	U+00E2	â	226
U+00C5	À	197	U+00D4	Ô	212	U+00E3	ã	227
U+00C6	Æ	198	U+00D5	Õ	213	U+00E4	ä	228
U+00C7	Ҫ	199	U+00D6	Ӯ	214	U+00E5	å	229
U+00C8	È	200	U+00D7	×	215	U+00E6	æ	230
U+00C9	É	201	U+00D8	Ø	216	U+00E7	ç	231
U+00CA	Ê	202	U+00D9	Ù	217	U+00E8	è	232
U+00CB	Ë	203	U+00DA	Ú	218	U+00E9	é	233
U+00CC	Ì	204	U+00DB	Û	219	U+00EA	ê	234
U+00CD	Í	205	U+00DC	Ü	220	U+00EB	ë	235
U+00CE	Î	206	U+00DD	Ý	221	U+00EC	ì	236
U+00CF	Ї	207	U+00DE	Ҧ	222	U+00ED	í	237
U+00D0	Ҕ	208	U+00DF	ҧ	223	U+00EE	î	238

UNI	CHAR	DEC	UNI	CHR	DEC	UNI	CHR	DEC
U+00EF	í	239	U+00F5	ó	245	U+00FB	û	251
U+00F0	ð	240	U+00F6	ö	246	U+00FC	ü	252
U+00F1	ñ	241	U+00F7	÷	247	U+00FD	ý	253
U+00F2	ø	242	U+00F8	ø	248	U+00FE	þ	254
U+00F3	ó	243	U+00F9	ù	249	U+00FF	ÿ	255
U+00F4	ô	244	U+00FA	ú	250			

20. Tabla: Puntos de código Unicode y sus equivalencias decimales en UTF-8

Referencias

- [0] R. Gillam, "Unicode Storage and Serialization Formats", in *Unicode Demystified*. Ed. Addison-Wesley Professional: USA, 2002, pp. 138-140.
- [1] Z. Kissel, J. Wang, "Appendix D: Base64 Encoding", in *Introduction to Network Security 2nd Edition*. Ed. John Wiley & Sons: USA, 2015, pp. 383.
- [2] P. Prakhar, "Common Security Protocols", in *Mastering Modern Web Penetration Testing*. Ed. Packt Publishing: USA, 2016, pp. 14-16.

SISTEMA DE AUTENTICACIÓN POR CREDENCIALES CRIPTOGRÁFICAS DISOCIADAS (SACRED)

Eugenia Bahit

Resumen

Con el Sistema de Autenticación por Credenciales Disociadas (**SACRED**) se pretende generar un mecanismo de autenticación de usuarios, que aísle a estos de sus respectivas credenciales de acceso.

SACRED propone el uso concomitante de tres recursos informáticos, como artílugo para la generación y manejo seguro de credenciales criptográficas de alta entropía:

1. **Algoritmos hash** preexistentes, como medio para la ofuscación e irreversibilidad de datos sensibles.

2. **Sistemas de persistencia de objetos y Objetos en Estado Puro**, como medio para la lectura (recuperación) y escritura (almacenamiento) de usuarios.
3. **Generación de datos en tiempo de ejecución**, como único medio para relacionar usuarios con sus respectivas credenciales.

La conjunción de estos tres mecanismos genera como resultado, que incluso ganando acceso a los datos, se dificulte el reconocimiento y manipulación, de nombres de usuario y contraseñas, o poder deducir qué credenciales pertenecen a cuáles usuarios.

usuario_id	denominacion	nivel
bdb11458	Jane Doe	25
c334c51a	Administradora general	1
e15f9a89	Ficus Master	1
f936f26c	Juan Pérez	15
5de97892	John Doe	99
83993ec4	Ramona García	25

21. Tabla: Ejemplo de tabla de usuarios empleada en un sistema SACRED

Directorio: `/path/to/.credentials`

```
.1b3c3dd18ec6dd33669727b63078e518d753de3d  
.5d0a3b698fecf59027a9c2547acd84d4760e6a61  
.a011033e465aff4d44ca96ab65a3750e93af38eb  
.a9d30746053b546d41617f61e17c0d05acb4a46e  
.ad9f4c989d118103db43da25e188b37e4fa06e92  
.ee1d6014143dd077facb2e7298c020ac441babeb
```

22. Tabla: Ejemplo de credenciales basadas en SACRED

Como se puede observar en el ejemplo anterior, no puede establecerse una relación fehaciente entre las credenciales almacenadas en un directorio oculto del sistema y sus respectivos propietarios, almacenados en la tabla de usuarios. Esto es debido a que los nombres de usuario y contraseñas empleados para acceder al sistema, no son almacenados ni siquiera de forma cifrada. En SACRED, los nombres de usuario y contraseña, se emplearían para generar tanto las ID de los usuarios (UID) como las credenciales, como partes complementarias de estos, pero no como piezas únicas.

De esta forma, **SACRED** podría estar garantizando que solo el propietario de los datos pueda manipularlos, y sin que esto suponga un cambio visible en la forma en la que el usuario final del Software, se autentica en el sistema.

No obstante, debe tenerse en cuenta el orden cronológico de creación de los archivos y registros de la base de datos (y de modificación y/o acceso en el caso de archivos). En los archivos de bases de datos, los registros quedarán almacenados en orden sucesivo de creación, mientras que los archivos, conservarán la fecha y hora de creación del *i-nodo*, acceso y modificación, que podrían evitarse, alterando dichos valores de forma pseudoaleatoria. La

modificación pseudoaleatorias de *i-nodos* será tratada en futuras investigaciones.

Definiciones previas

Sistema de autenticación

Mecanismo que permite validar la identidad de un usuario.

Credencial

Componente que interviene en la autenticación, el cual permite constatar que un usuario es quién afirma ser.

Credencial criptográfica

Credencial cuya información se encuentra cifrada mediante un conjunto de funciones *hash* combinadas, las cuales no poseen un algoritmo derivado que permita que su valor *hash* sea revertido.

Funcionamiento del sistema

Proceso de autenticación

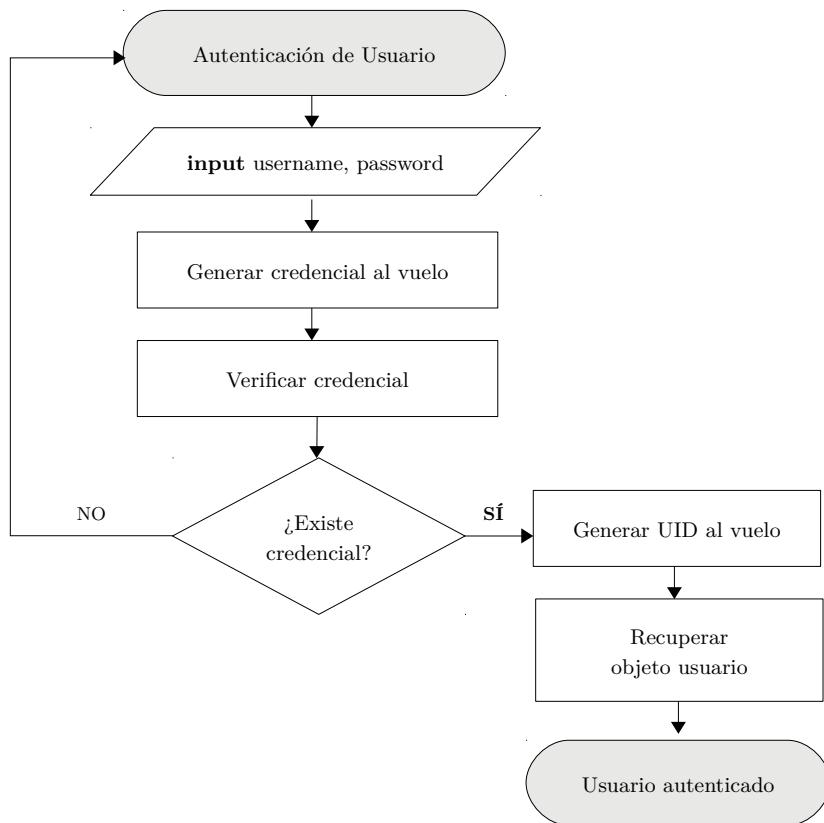
En el proceso de autenticación, el usuario aporta una combinación tradicional de *nombre de usuario* y *contraseña* de forma habitual.

A partir de dicha entrada, el sistema genera en tiempo de ejecución, la credencial correspondiente, mediante el empleo combinado de funciones *hash*.

A continuación, busca dicha credencial en el sistema de archivos (el nombre del archivo es el valor hash de la credencial).

Si la credencial existe, genera el UID (ID del usuario) correspondiente en tiempo de ejecución y utiliza dicho valor, para recuperar el objeto Usuario.

Si la credencial no existe, se vuelve al punto de inicio y el usuario deberá aportar nuevamente sus datos de acceso.



12. Ilustración: Proceso de autenticación

Generación de la credencial

Funciones necesarias y obtención de valores *hash*

Función *hash* para encriptado del nombre de usuario, $f(u) \rightarrow U$, donde u es el nombre de usuario en texto plano (valor de entrada) y U , el nombre de usuario encriptado (valor de salida).

Función *hash* para encriptado de la contraseña, $f(p) \rightarrow P$, donde p es la contraseña en texto plano (valor de entrada) y P , la contraseña encriptada (valor de salida).

Valor aleatorio reproducible de longitud fija (sal) de alta entropía. Dicho valor, se encontrará dado por una función *hash*, para cualquier orden de combinación de U , P y $x: f(U) \rightarrow x \vee f(P) \rightarrow x$. Dicha función se define como $f(U, x, P) \rightarrow s$, donde f es la función *hash* para la sal, x un valor aleatorio reproducible, obtenido a partir de U o P , y s , la sal resultante.

Valor hash para la generación de la credencial, determinado por una función *hash* $f(P, U, s) \rightarrow c$ para cualquier orden de combinación de U , P y s , donde c es la credencial (o valor *hash* resultante).

A continuación, se propone una implementación sencilla en lenguaje PHP, a modo de ejemplo.

Implementación a alto nivel (ejemplo en PHP)

```
function set_credencial() {
    $user_hash = hash('sha512', $_POST['usuario']);   f(u) → U
    $pass_hash = hash('md4', $_POST['clave']);         f(p) → P
    $x = substr($user_hash, 5, 12);                   f(U) → x
```

```

$salt_hash = hash('crc32', "{$user_hash}{$x}{$pass_hash}");
return hash('md5', "{$pass_hash}{$user_hash}{$salt_hash}");
}

```

Generación del UID

Funciones necesarias y obtención de valores *hash*

Función *hash* para encriptado del nombre de usuario, $f(u) \rightarrow U$.

Donde u es el nombre de usuario en texto plano (valor de entrada) y U , el nombre de usuario encriptado (valor de salida). Esta función puede (y debería) ser una función distinta a la función *hash* utilizada para encriptar el nombre de usuario en la generación de credenciales.

Valor aleatorio reproducible de longitud fija (sal) de alta entropía. Dicho valor, se encontrará dado por una función *hash* $f(x, U) \rightarrow s$, para cualquier orden de combinación de U y $x: f(U) \rightarrow x$, donde f es la función *hash* para la sal, x un valor aleatorio reproducible, obtenido a partir de U , y s es la sal resultante.

Valor hash para la generación del UID, determinado por una función *hash* $f(U, s) \rightarrow y$, para cualquier orden de combinación de U y s , donde y es el UID (o valor *hash* resultante).

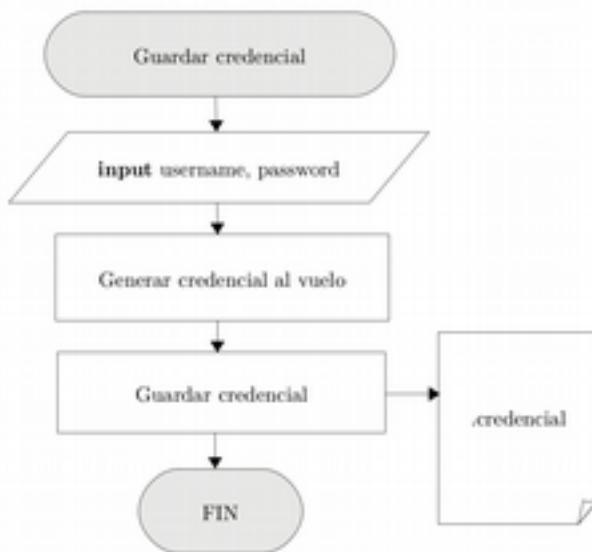
A continuación, se propone una implementación sencilla en lenguaje PHP, a modo de ejemplo.

Implementación a alto nivel (ejemplo en PHP)

```
function set_uid() {  
    $user_hash = hash('crc32', $_POST['usuario']);      f(u)→U  
    $x = substr($user_hash, -5);                      f(U)→x  
    $salt_hash = hash('md5', "{$x}{$user_hash}");      f(x,U)→s  
    return hash('crc32', "{$user_hash}{$salt_hash}");  f(U,s)→y  
}
```

Persistencia de la credencial

Las credenciales podrán persistir como parte del sistema de archivos, utilizando el valor *hash* de la misma como nombre del fichero.



13. Ilustración: Persistencia de credenciales en el sistema de archivos

Cuestiones relativas a la seguridad del sistema de archivos

Se deberá destinar un directorio privado oculto, para el almacenaje de credenciales, el cual no pueda ser accesible por el usuario. En aplicaciones

Web, esto será un directorio no servido , con permisos 777, bajo la propiedad del usuario *root*.

Validación de credenciales

Siendo C el directorio destinado al almacenamiento de todas las credenciales c del sistema tal que $C=\{c\}$, y x el valor *hash* de una credencial obtenida en tiempo de ejecución mediante una función $f(P,U,s) \rightarrow x$, una credencial será válida sí y solo sí, se cumple:

$$f(P,U,s) \rightarrow x , x \in C \Leftrightarrow x = c$$

A continuación, se propone una implementación sencilla en lenguaje PHP, a modo de ejemplo.

Implementación a alto nivel (ejemplo en PHP)

```
$credencial = set_credencial();      f(P,U,s) → x
$credencial_file = "/path/to/.credentials/.{$credencial}";
if(file_exists($credencial_file)) { x ∈ C
    // credencial válida           x = c
}
```

El objeto Usuario

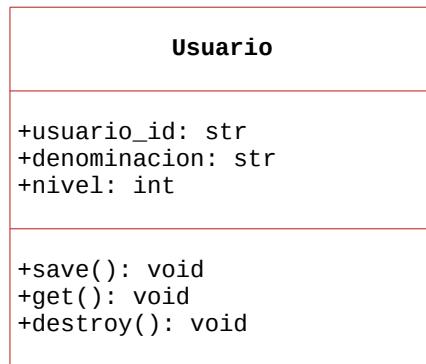
Para que pueda implementarse **SACRED** como método de autenticación en un sistema informático, un objeto *Usuario* debe tener, necesariamente, una propiedad ID (*UID*) *y* de tipo *string*, cuyo valor se encuentre determinado por $f(U,s) \rightarrow y$ (definido anteriormente), y no puede tener, bajo ningún concepto, propiedades destinadas a almacenar el nombre de usuario, la contraseña ni la credencial. De esta forma, cuando una denominación pública

(o identificación humanamente legible) sea requerida, una propiedad *denominacion* formará parte de los atributos del objeto.

Dado que **SACRED** como sistema de autenticación no aporta una forma concreta de prevenir nombres de usuarios duplicados, deberían tomarse los recaudos necesarios para respetar el *principio de unicidad* del valor de *UID*, que plantea que siendo $K^{(O)}$ el universo de todos los objetos O de una misma clase, y p una propiedad de identidad única (*UID*), debe cumplirse que:

$$\forall O_i^n \in K^{(O)}, (p \in O_i) \Rightarrow p \notin O_n$$

Un objeto *Usuario* típico podría verse como el siguiente:



Donde *usuario_id* es el *UID*, *denominación* la denominación pública del usuario (distinta al nombre de usuario de la credencial), y *nivel* el nivel de acceso (permiso) del usuario con respecto a su rol dentro del sistema.

Sobre la persistencia del objeto Usuario en bases de datos relacionales

Cuando se trabaje con persistencia en bases de datos relacionales, dado que el UID será generado a partir del nombre de usuario de la credencial, y que la modificación periódica de credenciales (tanto de la contraseña como del nombre de usuario) se debe considerar una buena práctica de seguridad, cuando el valor del UID sea clave primaria en una tabla, toda clave foránea que haga referencia a la misma, deberá establecerse por defecto con la opción de actualización en cascada.

Ejemplo en MariaDB:

```
usuario VARCHAR(12),
FOREIGN KEY(usuario)
    REFERENCES usuario(usuario_id)
    ON UPDATE CASCADE
```

Modificación de credenciales

La modificación de una credencial, supone una actualización tanto del UID como de la propia credencial.

Cuando un usuario previamente autenticado en el sistema, manteniendo una sesión válida abierta, pretende modificar sus datos de acceso, se hace necesario conocer el UID, para actualizar la propiedad del objeto usuario, y la credencial actual, bien sea para moverla (renombrarla), o bien para eliminarla y que no quede una credencial huérfana.

El UID puede mantenerse persistente en una variable de sesión, puesto que será necesario para identificar al usuario a lo largo de las diversas funcionalidades del sistema. Sin embargo, almacenar el valor *hash* de la credencial en una variable de sesión, supondría un riesgo de seguridad, puesto que sería contrario a la disociación, que es el principio en el que se basa **SACRED** para garantizar la seguridad. Dado que para obtener el valor *hash* de la credencial, el nombre de usuario es igual de necesario que la contraseña, hacer que persista en una variable de sesión, sería un riesgo aún superior que hacer persistir el valor *hash* de la credencial, puesto que dejaría expuesto un dato sensible.

Por lo tanto, la única forma de modificar una credencial, es pedir al usuario sus datos de acceso actuales en el mismo momento en el que intenta modificarlos. De esta forma, un descuido habitual del usuario que dejase una sesión abierta, no implicaría un riesgo para la manipulación de sus datos de acceso, puesto que su credencial continuaría estando a resguardo. En este sentido y mediante este artilugio, **SACRED** serviría para **evitar** uno de los **principales riesgos de la Ingeniería Social**: si una sesión queda abierta, con el usuario ausente, nadie que pretenda asumir su rol, podrá modificar sus datos de acceso.

Tras la solicitud de datos de acceso actuales y nuevos, y la generación de los valores *hash* correspondientes, el proceso de modificación de una credencial, consistirá en una sucesión de tres pasos:

1. Generar una nueva credencial

2. Actualizar el UID del usuario
3. Eliminar la credencial actual

Sobre la interfaz de entrada de los datos

Si bien la disociación es el mecanismo de seguridad primario de **SACRED**, se debe tener en cuenta que la ofuscación y el ocultamiento (o volatilidad) de los datos, también son parte esencial para la seguridad del sistema. Por consiguiente, tanto las interfaces gráficas (GUI) como las de texto (TUI), deben emplear (toda vez que sea posible) campos de contraseña tanto para la propia clave como para el nombre de usuario.

Mantenimiento del Sistema

Un sistema de autenticación por credenciales disociadas, requiere del establecimiento de ciertos parámetros constantes y mecanismos internos, que permitan ejercer un control de seguridad complementario para el mantenimiento del sistema. Estos parámetros y mecanismos se definen a continuación.

Parámetros constantes

Se definen dos valores constantes para establecer el *tiempo de vida* y el *período de inactividad* específicos, que afectarán a la validez de una credencial.

Tiempo de vida

Por tiempo de vida L , se entiende al período de validez máximo de una credencial desde el momento de su creación, expresado en segundos.

$$L=365 \cdot 24 \cdot 3600 ; \text{ tiempo de vida equivalente a un año}$$

Siendo la constante L el tiempo de vida establecido para cualquier credencial; t_c los segundos transcurridos desde 1970-01-01 a las 00:00:00 HS UTC, hasta el momento de creación de la credencial; y t_a los segundos transcurridos desde 1970-01-01 a las 00:00:00 HS UTC, hasta la actualidad, una credencial será válida mientras se cumpla $t_a - t_c \leq L$ o $t_c + L \geq t_a$.

En sistemas GNU/Linux, se establecen tres tipos de fecha para los archivos: fecha de acceso, de modificación del archivo, y fecha de modificación de los atributos del archivo. Estas fechas pueden verse con el comando *stat* (o mediante *struct stat* en la biblioteca GNU C).

```
eugenia@bella:~$ echo "hola" > archivo
eugenia@bella:~$ stat archivo
  Fichero: archivo
    Tamaño: 5           Bloques: 8          Bloque E/S: 4096
  fichero regular
Dispositivo: fe03h/65027d Nodo-i: 1449828   Enlaces: 1
Acceso: (0644/-rw-r--r--)Uid: ( 1000/ eugenia) Gid: ( 1000/ eugenia)
          Acceso: 2018-06-20 20:17:04.772791949 -0300
Modificación: 2018-06-20 20:17:04.772791949 -0300
  Cambio: 2018-06-20 20:17:04.772791949 -0300
  Creación: -
eugenia@bella:~$ chmod +x archivo
eugenia@bella:~$ stat archivo
  Fichero: archivo
    Tamaño: 5           Bloques: 8          Bloque E/S: 4096
  fichero regular
Dispositivo: fe03h/65027d Nodo-i: 1449828   Enlaces: 1
Acceso: (0755/-rwxr-xr-x)Uid: ( 1000/ eugenia) Gid: ( 1000/ eugenia)
          Acceso: 2018-06-20 20:17:04.772791949 -0300
Modificación: 2018-06-20 20:17:04.772791949 -0300
  Cambio: 2018-06-20 20:17:35.505052227 -0300
  Creación: -
```

Sin embargo, la fecha de creación no es soportada directamente sobre el archivo. La obtención de la misma se hace a partir del *inode* del fichero:

```
inode=`stat -c %i archivo`
```

Conociendo el *inode* del archivo y el dispositivo en el que se encuentra:

```
device=`df -P archivo | awk 'END{print $1}'`
```

Es posible obtener la fecha de creación del fichero mediante *debugfs* (aunque no directamente del propio archivo, sino, a partir del *inode* en el dispositivo):

```
str_time=`debugfs -R "stat <$inode>" $device 2>/dev/null | grep -oP 'crttime.*--\s*\K.*'`
```

Dado que la fecha se obtiene como una cadena, se hace necesario convertirla a fin de obtener la cantidad de segundos transcurridos desde 1970-01-01 a las 00:00:00 HS UTC:

```
crttime=`date -d "$str_time" +%s`
```

Como **alternativa**, la fecha de creación podría grabarse dentro de la credencial en texto plano (siempre en segundos). En ese caso, siendo *C* la credencial, esta será válida si se cumple $t_a - (t_c \in C) \leq L$ y también $(t_c \in C) + L \geq t_a$.

Es importante aclarar que una credencial caducada, permanece en el sistema hasta su renovación, pero no debe ser eliminada del sistema antes de ser renovada, debido a este período. La excepción será una credencial inactiva, es decir, una credencial que estando caducada, supere el período de inactividad (el cuál es inferior al tiempo de vida). Este período se explica a continuación.

Período de inactividad

Se entiende por período de inactividad de una credencial, al tiempo máximo que puede trascurrir después de la fecha de último acceso del archivo al tiempo actual. Dicho período es una constante similar al tiempo de vida,

denotada por P , y quedará determinada por: $P = \frac{L}{2}$.

Siendo la constante P el período de inactividad establecido para cualquier credencial; t_l los segundos transcurridos desde 1970-01-01 a las 00:00:00 HS UTC, hasta la fecha de último acceso de la credencial (arrojada por *stat*); y t_a los segundos transcurridos desde 1970-01-01 a las 00:00:00 HS UTC, hasta la actualidad, una credencial será válida mientras se cumpla $t_a - t_l \leq P$ o $t_l + P \geq t_a$.

Las credenciales inactivas podrán ser eliminadas siempre que se cumpla la negación de lo anterior. Es decir que para toda credencial c en el conjunto C de credenciales, se elimina una credencial si se cumple $\neg(t_l + P \geq t_a)$ o su equivalente $t_l + P \leq t_a$.

Siendo $f(c) \rightarrow t_l$ la función de retorno de fecha de último acceso de una credencial, la lógica sistemática de eliminación de credenciales inactivas quedaría determinada por el siguiente enunciado:

$$\forall c \in C : \neg(f(c) \rightarrow t_l + P \geq t_a) \Rightarrow \neg c$$

Mecanismos internos de control complementario

A fin de ejercer un control de seguridad complementario sobre un sistema de autenticación por credenciales disociadas, se deberán establecer los siguientes mecanismos:

- *Mecanismo de renovación de credenciales*: determinado por el tiempo de vida de una credencial, obliga al usuario a modificar sus datos de acceso al sistema cada cierto período de tiempo.
- *Mecanismo de recolección de basura*: determinado por el período de inactividad de una credencial, permite mantener el sistema libre de credenciales huérfanas o en desuso, que podrían ser explotadas con fines maliciosos.

Renovación de credenciales

La renovación de una credencial, es la acción de generar una nueva credencial para un usuario preexistente. La renovación de una credencial puede llevarse a cabo cuando una credencial alcanza el tiempo de vida establecido L , o bien, cuando la persona propietaria de la misma lo solicite como consecuencia del extravío de su credencial (olvido de su nombre de usuario y/o contraseña). En

este último caso, quedará una credencial huérfana que deberá ser eliminada a través de un mecanismo de limpieza o recolección de basura.

Renovación por caducidad

Se llevará a cabo en tiempo de autenticación.

En el momento de comprobar los datos de una credencial (ver «Validación de credenciales») y tras constatar que una credencial existe, el sistema deberá verificar el tiempo de vida de la credencial. Cuando no se cumpla $t_a - t_c \leq L$, deberá solicitarse al usuario que aporte nuevos datos de acceso.

El proceso de renovación por caducidad, debería llevarse a cabo de la misma forma que el proceso de modificación convencional descrito anteriormente:

1. Generación de nueva credencial
2. Actualización del UID del usuario
3. Eliminación de la credencial actual

Renovación por extravío

Por **extravío** de una credencial, se entiende a la dificultad práctica y cognitiva del propietario de una credencial, para recuperar sus datos de acceso al sistema.

En caso de que un usuario extravíe sus datos de acceso, el sistema le deberá ofrecer la posibilidad de establecer unos nuevos. Esta acción dejará una credencial huérfana (extraviada) en el sistema, la cual deberá ser purgada mediante un mecanismo de recolección de residuos (descrito más adelante).

Por lo tanto, la revocación por extravío se llevará a cabo mediante solicitud expresa del usuario.

Dicha renovación, deberá hacerse mediante la *generación de un token criptográficamente* seguro, enviado al usuario a través de un dato de contacto aportado previamente y almacenado en el sistema (antes del extravío). Dicho token deberá almacenarse de forma temporal, y tener una vida media no superior a las 24 horas.

Diferentes formas de crear un token criptográficamente seguro con PHP:

```
# Empleando random_bytes (desde PHP 7.0)
$token = bin2hex(random_bytes(128));

# Mediante la biblioteca OpenSSL
$token = bin2hex(openssl_random_pseudo_bytes(128));

# Utilizando OAuth (requiere el paquete php-oauth)
$token = OAuthProvider::generateToken(128);
```

Definir un método de «recuperación de credenciales» va más allá de los objetivos de SACRED. No obstante, un mecanismo viable podría incluir una serie de pasos como los siguientes:

- 1 USUARIO solicita la generación de una nueva credencial por extravío
- 2 SISTEMA solicita al usuario aportar un dato de contacto (se espera que este dato exista en el sistema)
- 3 USUARIO aporta un dato de contacto

4 SISTEMA realiza una búsqueda inversa de la ID del usuario, a partir del dato de contacto (esto se hace en realidad, para constatar que el dato de contacto pertenezca a un usuario preexistente en el sistema).

4.1 Si encuentra coincidencia:

- 4.1.1 Genera un token criptográficamente seguro
- 4.1.2 Almacena el token temporalmente junto a la ID del usuario
- 4.1.3 Envía al usuario a través del dato de contacto indicado, el token junto a una URL en la cual intercambiar dicho token por una nueva credencial

5 USUARIO ingresa a la URL indicada y aporta el token recibido

6 SISTEMA verifica el token

6.1 Si encuentra coincidencia:

- 6.1.1 Elimina el token
- 6.1.2 Solicita al usuario un nuevo nombre de usuario y contraseña
- 6.1.3 Modifica la ID del usuario en la base de datos
- 6.1.4 Genera nueva credencial

Purga de credenciales y recolección de basura

La purga de credenciales es la acción sistemática y automatizada de «*recolección de basura*», entendiéndose por tal —en el presente contexto—, a la eliminación de credenciales huérfanas (credenciales extraviadas) o en desuso (credenciales inactivas).

Una credencial solo podrá considerarse huérfana cuando no pueda ser asociada a un usuario existente del sistema. Dado que SACRED disocia las credenciales de los usuarios, solo podrá saberse que existe al menos una credencial huérfana en el sistema (pero no cuál), en el caso de que un usuario reporte su credencial como extraviada. Y esto solo sucederá, en el momento que el usuario solicite una renovación por extravío. Cuando esto suceda antes de cumplirse el plazo establecido para P , la credencial será considerada huérfana, aunque en un sentido práctico no puede determinarse cuál es esa credencial. Por ese motivo, una credencial huérfana solo podrá ser purgada cuando iguale o supere el período de inactividad P establecido en el sistema.

Este procedimiento de eliminación de credenciales huérfanas o inactivas, deberá llevarse a cabo de forma periódica (al menos una vez por día), eliminando toda credencial cuando la diferencia entre la fecha actual t_a y la fecha de último acceso t_l sea mayor o igual al período de inactividad P establecido en el sistema, tal que:

$$\forall c \in C : \neg(f(c) \rightarrow t_l + P \geq t_a) \Rightarrow \neg c$$

PARTE III: 11. SISTEMAS DE INFORMACIÓN GEOGRÁFICA

CONFIGURACIÓN DE UN ENTORNO VIRTUAL PARA LA CENTRALIZACIÓN DE DATOS EN SISTEMAS DE INFORMACIÓN GEOGRÁFICA (SIG)

Montserrat Ortega Gallart

Resumen

Los *Sistemas de Información Geográfica (SIG)* son herramientas de información y gestión que permiten trabajar con datos geográficamente referenciados, tales como coordenadas geográficas. El objetivo de un SIG, es el tratamiento de información espacial que normalmente se almacena en una base de datos con extensiones que soporten información geográfica.

Uno de los sistemas de gestión de base de datos relacional empleado en los SIG, es *PostgreSQL*, distribuido bajo licencia *PostgreSQL License* (software libre). Junto con la extensión *PostGIS*, permite trabajar con objetos

geográficos, índices espaciales y funciones geográficas y geoespaciales específicas, que operan sobre dichos datos.

Como base de trabajo para el Sistema de Información Geográfica, se empleará una máquina virtual, es decir, un programa que permita emular un ordenador con un sistema operativo determinado, ejecutando los programas necesarios en un entorno virtual, y centralizando la estructura de datos en un único equipo, sin necesidad de alterar su configuración actual.

Materiales necesarios

La siguiente, es una lista de las herramientas que serán empleadas para la centralización de datos geográficos:

- **Oracle VM VirtualBox®:** Software de virtualización multiplataforma que permite crear máquinas virtuales simulando distintos sistemas operativos.
- **Ubuntu:** Sistema Operativo de código abierto. Distribución de GNU/Linux basada en Debian.
- **PostgreSQL:** Sistema de gestión de base de datos relacional, distribuido bajo licencia PostgreSQL License (software libre).
- **PostGIS:** Extensión para PostgreSQL que permite trabajar con objetos geográficos y funciones espaciales, utilizada en los Sistemas de Información Geográfica (SIG).

Procedimiento

Requerimientos

1. Descargar e instalar VirtualBox® desde el sitio oficial:
<http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html>

2. Descargar la ISO de Ubuntu desde la siguiente página:
<http://releases.ubuntu.com/> seleccionando la versión 18.04 LTS del S.O., y teniendo en cuenta la arquitectura del ordenador (32 o 64 bits⁶⁹).

AVISO: La presente guía de procedimiento ha sido probada por la autora con las versiones **16.04 LTS** de Ubuntu y PostgreSQL **9.5**, y verificada posteriormente con las versiones **18.04 LTS de Ubuntu** y **PostgreSQL 10**.

Creación de la máquina virtual

Ejecutar el programa y desde la pantalla principal, seleccionar la opción “*Crear máquina virtual Ubuntu*” con la siguiente información:

Tipo: Linux

69 Nota de la editora: ante la duda, se recomienda descargar la versión para 32 bits (soporta hasta 4 GiB de memoria).

Versión: Ubuntu (64 bits) o Ubuntu (32 bits) según la arquitectura que se tenga

Tamaño de memoria: ≥ 2 GB (el mínimo de memoria recomendado es de 2 GB. A partir de aquí se puede escoger la memoria que se desee en función de la memoria RAM de la que disponga la máquina)

Disco duro: Crear un disco duro virtual. **Tipo:** VDI. **Tamaño de almacenaje fijo:** ≥ 8 GB (el tamaño mínimo recomendado del disco es de 8 GB. A partir de aquí hay que seleccionar el tamaño que sea más apropiado según las necesidades del proyecto).

Una vez creada la máquina virtual, pedirá el fichero ISO de Ubuntu. Hay que seleccionar el fichero descargado en el punto 2 del apartado de “*Requerimientos*”.

Configuración del Sistema Operativo

A continuación se detallan los pasos de instalación de Ubuntu:

1. Seleccionar el idioma (en este caso, español) y luego, “*Instalar Ubuntu*”.
2. Escoger la opción “*Descargar actualizaciones e instalar software de terceros*”. La instalación de Software de terceros permitirá trabajar con formatos multimedia no libres.
3. El siguiente paso es el tipo de instalación que se quiere hacer. Se escoge la de “*Borrar disco e instalar Ubuntu*” y se pulsa “*Continuar*”.

4. Se escoge la franja horaria y la disposición del teclado.
5. Y la última pantalla es la de configuración de usuario y contraseña. El usuario que se crea tiene permisos de administrador (*root*) en Ubuntu.
6. Finalmente se reinicia el Sistema y se tiene Ubuntu funcionando en la máquina virtual.

Instalación de la base de datos

Una vez dentro de la máquina virtual de Ubuntu se instalarán *postgreSQL* y *postGIS* para poder trabajar con datos espaciales. A continuación se detallan los pasos a seguir.

1. Se instala *postgreSQL* y *postGIS* desde los repositorios:

```
apt update && apt upgrade -y  
apt install postgresql postgis
```

2. Se ingresa a la shell interactiva de PostgreSQL:

```
sudo -u postgres psql
```

Observaciones: si se ejecuta este comando desde un directorio en el que el usuario *postgres* no tenga permisos, se obtendrá un mensaje de alerta del tipo «*could not change directory*» como se muestra en el siguiente ejemplo:

```
root@localhost:~# pwd  
/root  
root@localhost:~# sudo -u postgres psql  
could not change directory to "/root": Permission denied  
psql (9.5.19)  
Type "help" for help.
```

El mensaje precedente es solo una advertencia que no afectará al correcto avance del procedimiento y si se desea evitar dicha salida, se puede acceder a la *shell* de *PostgreSQL* desde la *home* de cualquier usuario.

3. Se activa la extensión *adminpack*, a fin de ampliar las funcionalidades adicionales de administración⁷⁰:

```
CREATE EXTENSION adminpack;
```

4. Se crea el esquema de base de datos con el que trabajar:

```
CREATE USER <usuario> WITH PASSWORD '<contraseña>';
```

```
CREATE DATABASE "<basededatos>"  
    WITH OWNER = <usuario>  
        ENCODING = 'UTF8'  
        TABLESPACE = pg_default  
        CONNECTION LIMIT = -1;
```

```
CREATE SCHEMA <esquema>  
AUTHORIZATION <usuario>;
```

```
CREATE EXTENSION postgis;
```

Donde los datos encerrados entre *< y >* serán reemplazados por los correspondientes. A modo de ejemplo:

```
CREATE USER montse WITH PASSWORD 'secret';
```

```
CREATE DATABASE "DBMONTSE"  
    WITH OWNER = montse  
    ENCODING = 'UTF8'  
    TABLESPACE = pg_default  
    CONNECTION LIMIT = -1;
```

```
CREATE SCHEMA nuevoesquema
```

70 <https://www.postgresql.org/docs/10/adminpack.html>

```
AUTHORIZATION montse;  
CREATE EXTENSION postgis;
```

5. Una vez instalado todo hay que editar el fichero */etc/postgresql/<VERSION>/main/pg_hba.conf* para que permita conexiones de otros *hosts*. La sintaxis empleada es la siguiente:

```
<TYPE> <DATABASE> <USER> <ADDRESS> <METHOD>
```

Notar que *<ADDRESS>* se omitirá en el primer caso:

```
local    all    postgres      peer  
host     all    all          0.0.0.0/0      md5
```

6. En el fichero */etc/postgresql/<VERSION>/main/postgresql.conf* se habilita la escucha desde todas las interfaces TCP/IP disponibles (el valor por defecto, *localhost*, solo permite conexiones TCP/IP locales):

```
listen_addresses = '*'
```

7. Se reinicia *postgreSQL* para que coja los cambios:

```
service postgresql restart
```

Observaciones: si se desea obtener una salida de estado del servicio, reiniciar mediante *init.d*:

```
/etc/init.d/postgresql restart
```

8. Finalmente, se configura la máquina virtual para que tenga una IP externa y pueda recibir peticiones desde fuera. Para ello, se debe acceder a la opción *Settings* de la máquina virtual y establecer los siguientes valores:

Modo de conexión: Adaptador puente

Interfaz: (la interfaz de red que se desee utilizar)

Modo promiscuo: Permitir todo (permite ver por ese adaptador de red todo el tráfico que circula por esta)

Cable conectado: indica que se conectará a la red mediante cable. Si no se indica esta opción se tendría que conectar vía *Wi-Fi*.

9. Una vez hecho el último paso, si se accede a la máquina virtual, se puede abrir la línea de comandos y ejecutar:

```
ifconfig # o en su defecto, ejecutar: ip addr
```

Deberá aparecer la IP de la máquina que permitirá las conexiones entrantes.

A partir de aquí, se puede abrir cualquier programa que permita conexiones con *postGIS* (por ejemplo *pgAdmin*) y probar la conexión con la IP de la máquina virtual. Esto permite tener programas operando en cualquier máquina sobre diversas plataformas, pero utilizando la base de datos de la máquina virtual.

DISEÑO DE MAPAS GEOGRÁFICOS APLICADOS AL DESARROLLO WEB Y MÓVIL

Jéssica Sena

Definiciones previas

Geocodificación y geocodificación reversa

Se denomina geocodificación, al proceso para encontrar coordenadas espaciales, a partir de alguna otra información geográfica georeferenciada, como una dirección o un código postal. Al proceso inverso, buscar una información geográfica georeferenciada a partir de unas coordenadas, se lo denomina geocodificación reversa.

Geolocalización

Consiste en determinar la posición geográfica actual de un dispositivo conectado a una red de comunicaciones, como Internet.

Diseño e implementación de funciones de geocodificación y geolocalización

Antes de iniciar el diseño e implementación de las mismas, será necesario tomar dos decisiones relativas a cómo se mostrarán en las aplicaciones, los datos que de ellas se obtengan, tales como las posiciones, los diferentes datos georeferenciados o las coordenadas. Las decisiones a tomar son:

1. Qué base cartográfica utilizar (es decir, la fuente desde la cual se obtendrán los mapas).
2. Qué servicios, *frameworks*, y/o herramientas se emplearán para mostrar la cartografía elegida y para que el usuario pueda interactuar con ella.

Elección de una base cartográfica

Existen diferentes opciones, desde bases cartográficas públicas y oficiales ofrecidas como servicio gratuito por las administraciones públicas, hasta bases no oficiales, abiertas y colaborativas, como OpenStreetMap⁷¹.

71 <https://www.openstreetmap.org>

*OpenStreetMap*⁷² (OSM) es un proyecto de mapa colaborativo en el que cualquier persona puede aportar datos y utilizarlos, siempre y cuando cumpla su política de uso⁷³. Como se especifica en la documentación, aunque los datos sean gratuitos y abiertos a todo el mundo, el mantenimiento de sus servidores tiene un coste, pagado con donaciones, y por ello, se requiere permiso explícito para hacer un uso intensivo de sus servicios.

La información aportada se almacena en una base de datos, que según se informa en la página oficial, en 2017 superaba los 800 GB de información, y en 2018 alcanzó el millón de contribuidores⁷⁴.

Algunos países cuentan con instituciones públicas oficiales dedicadas a la cartografía, que ofrecen servicios abiertos para el consumo de sus mapas por parte de la ciudadanía. Algunos ejemplos de ello son el *IGN Francés*⁷⁵ o el *Institut Cartogràfic i Geològic de Catalunya*⁷⁶.

Carga e interacción con la cartografía

Si el objetivo del proyecto es solo mostrar datos en un mapa para que el usuario pueda consultarlos, se puede utilizar algún proveedor de servicios como *Umaps*⁷⁷ o *Instamaps*⁷⁸, que permita la carga en línea de esos datos.

72 OpenStreetMap: <https://wiki.openstreetmap.org/wiki/>

73 OSM Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>

74 OpenStreetMap Wiki, «In Summary» [online]. Mediawiki.org: 2014. Disponible en <https://wiki.openstreetmap.org/wiki/Stats>

75 IGN: <http://www.ign.fr/>

76 ICGC: <http://icgc.cat/>

77 Umap: <http://umap.openstreetmap.fr/es/>

78 InstaMaps: <https://instamaps.cat>

Si en cambio se necesita algo más que la consulta de datos, o si se requiere de cierto nivel de personalización en la visualización, que implique el desarrollo de código propio, se puede utilizar un *framework* especializado o biblioteca. Algunos de estos *frameworks* se describen a continuación.

OpenLayers⁷⁹

Se trata de un proyecto de código abierto, basado en JavaScript, que permite la carga de mapas dinámicos en aplicaciones Web.

Leaflet⁸⁰

Es otro proyecto de código abierto, también basado en Javascript, que como se indica en su documentación oficial, nace con el objetivo de ser una biblioteca de mapas dinámicos *mobile-friendly*, con un peso de apenas 138KB.

Mapbox GL JS⁸¹

Es una biblioteca de código abierto para la carga de mapas en web, combinando el uso de la tecnología *Vector Tiles⁸²* y de la API WebGL (*Web Graphics Library*) para el *renderizado*.

79 OpenLayers: <https://openlayers.org/>

80 Leaflet: <https://leafletjs.com/>

81 Mapbox GL JS: <https://www.mapbox.com/mapbox-gl-js/api/>

82 Vector Tile: <https://github.com/mapbox/vector-tile-spec>

Creación básica de un mapa

Teniendo estas tres bibliotecas, a continuación se compara el uso de las mismas, mediante un ejemplo de inicialización mínima, de un mapa para un entorno web, cada una de ellas a partir del mismo código base en HTML:

HTML: <div id="map" class="map"></div>

OpenLayers:

```
var map = new ol.Map({
  target: 'map',
  layers: [new ol.layer.Tile({source: new ol.source.OSM()})],
  view: new ol.View({
    center: ol.proj.fromLonLat([37.41, 8.82]),
    zoom: 4
  })
});
```

Leaflet (JS):

```
var map = L.map('map').setView([51.505, -0.09], 13);
L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png',
{
  attribution: '© OpenStreetMap' // ver nota83 al pie
}).addTo(map);
```

Mapbox GL JS:

```
var map = new mapboxgl.Map({
  container: 'map',
  style: 'mapbox://styles/mapbox/streets-v9'
});
```

83 Valor requerido por las políticas de uso de OpenStreetMap

Se observa que las tres bibliotecas se basan en el mismo concepto para la generación de un mapa:

- La declaración y creación de un objeto de tipo mapa.
- El paso de parámetros de configuración a dicho objeto, tales como la URL de la cartografía base a mostrar (paso 1), y un punto o zona donde centrar la visualización del mapa.

Referencias

- [0] M. Zurbaran, «*Geocoding and reverse geocoding using the GeoNames datasets*» en PostGIS Cookbook. Packt Publishing: UK, 2014.
- [1] A. Serafini, «*Some idea about geolocalization*» en Apache Solr Beginner's Guide. Packt Publishing: UK, 2013.
- [2] J. Nunes, «*Geolocalització*» [online]. Institut Cartogràfic de Catalunya: España, 2013. Disponible en <http://www.icgc.cat/ca/Ciutada/Informa-t/Diccionaris/Geolocalitzacio>

INTRODUCCIÓN AL TRAZADO DE MAPAS WEB MEDIANTE MOSAICOS VECTORIALES

Jéssica Sena

Definiciones previas

Mosaicos vectoriales

Un mosaico es una composición de pequeños fragmentos, denominados teselas. La tecnología de «*mosaicos vectoriales*» (*vector tiles*) es aquella que propone la generación de imágenes como un único mosaico, compuesto por diferentes piezas (teselas), las cuales se generan a partir de gráficos vectoriales.

Trazado de mapas Web

La técnica de «*trazado de mapas y planos geográficos*» se conoce en inglés como «*mapping*» (cartografía). Algunos autores definen el trazado de mapas web o «*webmapping*» como «*un proceso de diseño, implementación,*

generación y distribución de mapas, datos geospatiales, y provisión de Sistemas de Información Geográfica (SIG) como servicio o funcionalidad de una aplicación Web» (Li, 2011).

Antecedentes

Formatos gráficos

De la misma forma que en los SIG⁸⁴ de escritorio, para el trazado de mapas en aplicaciones Web, existen dos opciones de formatos de representación de la información geográfica:

1. Gráficos **de trama**, comúnmente conocidos como «ráster», basados en píxeles de tres colores de 8 bits (de allí, conocidos también como «mapa de bits»), que producen imágenes de píxeles fijos.
2. Gráficos **vectoriales**, basados en curvas logradas a partir de fórmulas matemáticas, que otorgan al gráfico mayor precisión, puesto que, entre otras cosas, no depende de la resolución de pantalla como en el caso de los gráficos de trama basados en píxeles.

Tipos de trazado

En orden evolutivo, es posible observar cuatro tipos de técnicas de trazado de mapas para aplicaciones Web:

84 Sistemas de Información Geográfica, SIG

1. **Trazado basado íntegramente en tramas o mapas de bit:** mapas basados en imágenes estáticas e interacción prácticamente nula.
2. **Trazado basado en mosaicos de mapas de bit:** se trata de un sistema de trazado en teselas para el consumo de imágenes entramadas, que a diferencia del anterior, no trabaja con una única imagen correspondiente a una zona, sino con pequeños trozos o *teselas*, que componen una zona específica (mosaico). Este tipo de trazado, introdujo técnicas de optimización como el almacenamiento temporal en memoria de recursos visitados o la anticipación de los mismos (conocidos en inglés como *caching* y *prefetching*, respectivamente), los cuáles reducen los tiempos de espera del usuario al visualizar un mapa.
3. **Trazado basado en mosaicos de mapas de bits con capas de información vectorial superpuestas:** Estos trazados aprovechan la estandarización y evolución en la web, de formatos como SVG o de la API Canvas, para el añadido de capas de información vectorial a los mapas, por encima de las capas de entramado, a modo de información adicional, ampliando el nivel de interacción del usuario con el mapa. El mapa de bits se usa principalmente para representar mapas de base (como por ejemplo, el plano de una ciudad con sus calles, parcelas de edificios y parques, etc.) con gran cantidad de información, mientras que los gráficos vectoriales, como capa de información complementaria, superpuesta al mapa base (como por

ejemplo, una capa de puntos correspondientes a los cines de la ciudad).

4. **Trazado netamente basado en mosaicos vectoriales.** Este tipo de trazado, ofrece en los mapas un mayor nivel de interacción que sus antecesores basados en mapas de bits, y se describen en lo sucesivo.

Trazado de mosaicos vectoriales

Los mosaicos vectoriales son un formato para la entrega de datos geográficos vectoriales, estructurados a su vez en un sistema de teselas, en el cual la información se divide en pequeños trozos que son enviados al cliente bajo demanda para su consumo.

Cada tesela contiene únicamente una parte de un todo y es el cliente quien une dichas teselas para obtener un conjunto inicial.

La información geográfica de las teselas vectoriales no sólo consiste en geometrías vectoriales como líneas o polígonos de caminos o edificios, sino que además pueden incorporar datos asociados a cada uno de estos elementos, como su identificador, su nombre, su longitud, su propietario, entre otros.

Por ello, la forma en que se almacenan los datos dentro de cada tesela es un factor clave, puesto que el objetivo es guardar la máxima cantidad de información en el menor espacio posible, evitando que la interpretación de los datos para la generación de los mapas al vuelo, afecte dramáticamente al rendimiento del cliente.

Tipos de formato para los datos de gráficos vectoriales

Formatos de fichero único sin compresión

Existen formatos basados en XML, como KML, SVG, GML o SLD. También basados en JSON, como el GeoJSON. Todos ellos ya han sido usados en los SIG de escritorio con anterioridad. Su uso resulta en la obtención de un único fichero que contiene toda la información vectorial requerida, e incluso, sus estilos de visualización.

Son formatos desarrollados, en algunos casos, con el objetivo implícito de ser legibles, autoexplicativos y autodescriptivos, y para seguir el formato de pares clave-valor para la definición de sus propiedades. Así, para grandes cantidades de información como la que puede haber en un mapa de base, puede implicar un peso de fichero demasiado grande para su uso y transmisión mediante la Web.

El consumo de estos datos demanda la descarga del fichero completo en el cliente, sin tener en cuenta qué zona de datos se desea visualizar, lo que incrementa el consumo de ancho de banda y los tiempos de respuesta.

Formato TopoJSON

A fin de reducir el consumo de ancho de banda y tiempo de respuesta, mediante la aplicación de técnicas de compresión, pueden encontrarse formatos reducidos como *TopoJSON*, un tipo de formato *GeoJSON* simplificado, que entre otras cosas, elimina redundancias.

Formato PBF (*Protocolbuffer Binary Format*)

PBF es un formato binario compacto, que según estadísticas de OSM⁸⁵, para el conjunto de datos correspondientes al *planet.osm* (todos los datos mundiales en OSM), es cinco veces más rápido de escribir que el mismo archivo en *gzipped* y seis veces más rápido de leer⁸⁶.

La integración de soporte a la tecnología WebGL en los navegadores Web, utilizada para el trazado final de los datos vectoriales contenidos en los PBF, junto a la aplicación de técnicas de teselado para su distribución, hacen factible —en términos de rendimiento, estabilidad, y estándares—, el uso de gráficos vectoriales como mapa base, y no sólo como complemento informativo.

Características de los mosaicos vectoriales

Interacción

Una vez recibidas las teselas en el cliente, se tiene acceso completo a la información asociada a los objetos contenidos en ellas, sin que sea necesario hacer peticiones adicionales al servidor, como sucede con los gráficos de trama. Esto permite:

- Mostrar los datos de forma inmediata, bien sea a través de un *tooltip*, o bien, a través de una ventana de información.

85 OpenStreetMap, <https://www.openstreetmap.org>

86 https://wiki.openstreetmap.org/wiki/PBF_Format

- Realizar operaciones propias de los SIG entre las geometrías, directamente en el cliente, como por ejemplo, calcular el área de una parcela. Existen bibliotecas JS de código abierto para realizar este tipo de operaciones en el cliente, como por ejemplo *Turf.js*⁸⁷.

Estilización de geometrías al vuelo

El conjunto de datos puede ser personalizado tanto a nivel de capa como de geometría, mostrándose para cada cliente según sus necesidades y/o especificaciones. Esto significa que puede cambiarse al momento, el grosor de la linea de un río, su color, o incluso su transparencia. Por otro lado, también puede ocultarse por completo, toda la capa de ríos o mostrarse únicamente una sola capa con la información de carreteras. Todo ello, **sin necesidad de recargar el mapa**. Además, al poder hacerse todas estas modificaciones de forma dinámica, también ofrece la posibilidad de **crear animaciones**.

Dado que la representación de lo que se desea visualizar, se realiza directamente en el cliente, se hace necesario que este, cumpla unos requerimientos específicos de potencia de procesamiento y memoria para que los mosaicos vectoriales puedan tratarse de la forma esperada.

Se puede verificar esta característica de personalizado de estilos, probando algunas herramientas públicas como *PINTAMAPS*⁸⁸ o *Cartogram*⁸⁹ de Mapbox.

⁸⁷ <https://www.mapbox.com/help/how-analysis-works/>

⁸⁸ <http://betaserver.icgc.cat/pintamaps>

⁸⁹ <https://www.mapbox.com/cartogram>

Rotación de etiquetas

Al no tratarse de una imagen estática con las etiquetas incrustadas en ella, se puede rotar el mapa sobre sí mismo y darle inclinación, sin que las etiquetas (nombres de ciudades, calles, etc.) queden invertidas.

Escalabilidad y resolución

A medida que el usuario se mueve sobre el mapa o enfoca una zona específica, la imagen se vuelve a pintar. Por tanto, permite hacer *overzoom*⁹⁰ sin pérdida de resolución, como sucede con el formato de tramas, en que el *overzoom* implica la distorsión de los píxeles de la imagen debido a las diversas resoluciones.

Peso de los mapas

El uso de formatos binarios compactos para el almacenamiento de los datos vectoriales, sumado a la posibilidad de aplicar procesos de simplificación a las geometrías, hacen que la visualización de un mapa sea más ligera en este formato que en un trazado de tramas, corriendo bajo las mismas condiciones (mismo software y mismo hardware).

Referencias

- [0] V. Antoniou, J. Morley, M. Haklay, «Tiled Vectors: A Method for Vector Transmission over the Web» [online]. Semantic Scholar: London, 2009.
Disponible en

⁹⁰ Se denomina *overzoom* a la acción de acercar un conjunto de teselas fuera del rango de acercamiento establecido.

<https://pdfs.semanticscholar.org/e055/3260c24a489e31e044ba67a5eccf0b43ba8.pdf>

- [1] I. de Beukelaar, «Cartographic implications of Vector Tile technology» [online]. Australian National University: Australia, 2018. Disponible en <https://openresearch-repository.anu.edu.au/handle/1885/11794>
- [2] J. Gaffuri, «Toward Web Mapping with Vector Data» en Lecture Notes in Computer Science. Springer: Alemania, 2012. pp. 87-101.
- [3] R. Garcia, J. P., de Castro, E. Verdú, M. J. Verdú, L. M. Regueras, «Web Map Tile Services for Spatial Data Infrastructures: Management and Optimization» [online]. Semantic Scholar: Valladolid, 2012. Disponible en <https://pdfs.semanticscholar.org/8192/efb4f2178b324d5558e0cf69d4642426cf79.pdf>
- [4] Y.-K. Kang, K.-C. Kim, Y.-S. Kim, «Probability-Based Tile Pre-fetching and Cache Replacement Algorithms for Web Geographical Information Systems» [online]. Semantic Scholar: Korea, 2001. Disponible en <https://pdfs.semanticscholar.org/e7f5/636a075f622c761c61384ee994b79d9cccb4.pdf>
- [5] S. Li, S. Dragicevic, B. Veenendaal, «Advances in Web-based GIS, Mapping Services and Applications» [online]. Disponible en https://www.researchgate.net/publication/215684076_Advances_in_Web-based_GIS_Mapping_Services_and_Applications

- [6] E. López, R. Béjar, J. Barrera, F.J. López-Pellicer, A.F. Rodríguez, P. Abad, «Servicios de visualización INSPIRE basados en teselas vectoriales» [online]. Direção-Geral do Território (DGT): Portugal, 2017. Disponible en http://www.dgterritorio.pt/jiide2017/docs/Artigos/JIIDE2017_C-E4_Servicios_visualizacion_INSPIRE_basados_teselas_vectoriales_Ruben_Bejar.pdf
- [7] E. Stefanakis, «Web Mercator and Raster Tile Maps: Two Cornerstones of Online Map Service Providers» [online]. 2017. Disponible en https://www.researchgate.net/publication/321064657_Web_mercator_and_raster_tile_maps_two_cornerstones_of_online_map_service_providers
- [8] M. Taraldsvik, «The future of web-based maps: can vector tiles and HTML5 solve the need for high-performance delivery of maps on the web?» [online]. Norwegian University of Science and Technology: Noruega, 2012. Disponible en <http://hdl.handle.net/11250/232144>
- [9] B. Veenendaal, M. A. Brovelli, S. Li, I. Ivánova, «What is web mapping anyway?» [online]. Politecnico di Milano: Italia, 2017. Disponible en <https://re.public.polimi.it/retrieve/handle/11311/1046148/264002/isprs-archives-XLII-2-W7-155-2017.pdf>
- [10] B. Veenendaal, M. A. Brovelli, S. Li, «Review of Web Mapping: Eras, Trends and Directions» [online]. MDPI: Suiza, 2017. Disponible en <https://www.mdpi.com/2220-9964/6/10/317>

MATEMÁTICA TEÓRICA

EL TEOREMA DE LOS CUATRO COLORES: UN PROBLEMA TOPOLOGICO DEMOSTRADO CON AYUDA DE UN ORDENADOR

Marta Macho Stadler⁹¹

El *teorema de los cuatro colores* se enuncia de la siguiente manera:

Para colorear cualquier mapa geopolítico plano de tal manera que dos países con frontera común sean de distinto color, basta con cuatro colores.

Enunciado en el año 1852 en forma de conjetura, se tardó más de un siglo en dar una prueba de este enunciado. Además, la demostración del teorema no estuvo exenta de polémica. Pero empecemos por el principio.

91 Universidad del País Vasco

Planteamiento del problema y primeras simplificaciones

Para colorear cualquier mapa geopolítico plano, de tal manera que dos países con frontera común sean de distinto color, basta con cuatro colores.

Este es uno de esos problemas matemáticos tan fácil de entender que ‘pone en marcha’ a muchas personas que piensan que van a ser capaces de encontrar una prueba -o un contraejemplo- dedicándole un momento de su tiempo. Pero, como suele ocurrir, los problemas que tienen un planteamiento sencillo suelen ser los más complicados de demostrar o refutar.

Aclaremos algunos puntos del enunciado. Los mapas considerados son de una pieza (mapas conexos⁹²) y cada una de sus regiones también es conexa. Además, para que dos regiones se consideren vecinas, su frontera compartida debe consistir en una línea (curva) de longitud no nula.

Observemos que no importan ni el tamaño ni la forma de las regiones del mapa, sino la manera en la que están colocadas las unas respecto a las otras. Es decir, se trata de un problema topológico: la demostración del enunciado se

92 https://es.wikipedia.org/wiki/Conjunto_conexo

enmarca dentro del área de la *topología*⁹³, la rama de las matemáticas que estudia las propiedades cualitativas de los espacios.

La primera referencia a esta conjetura se debe al abogado y botánico *Francis Guthrie* quien fue capaz de colorear el complejo mapa de los cantones de Inglaterra con solo cuatro colores. Pensó que si un mapa tan intrincado solo necesitaba cuatro tonalidades, debía suceder lo mismo con cualquier otro mapa geopolítico. En 1852, se lo comentó a su hermano *Frederic* y este a su vez al matemático y lógico *Augustus de Morgan*, su profesor.

Por supuesto, algunos mapas no necesitan cuatro colores, por ejemplo, uno que posea solo dos regiones. Pero existen mapas que sí.

Frederick Guthrie fue el primero en observar que el problema de los cuatro colores no se podía generalizar a dimensión 3. El matemático *Heinrich Tietze* proporcionó un ejemplo de mapa tridimensional que precisaba tantos colores como se deseara. Su propuesta consistía en disponer n barras (numeradas de 1 hasta n), una junto a otra, y colocaba otras n barras (también numeradas de 1 hasta n) transversalmente sobre las primeras. Cada número representaba a una región (compuesta entonces por dos barras atravesadas) y el mapa tridimensional de n regiones así construido precisaba exactamente de n colores para no contradecir las reglas de la conjetura.

93 Marta Macho Stadler, ¿Qué es la topología?, Sigma 20 (2002) 63-77,
<http://www.ehu.eus/~mtwmastm/sigma20.pdf>

Augustus de Morgan se interesó por la conjetura de *Guthrie* y la difundió entre sus colegas. El 23 de octubre de 1852 escribió al científico *William Rowan Hamilton* en estos términos:

«A student of mine [Frederick Guthrie] asked me today to give him a reason for a fact which I did not know was a fact - and do not yet. He says that if a figure be anyhow divided and the compartments differently coloured so that figures with any portion of common boundary line are differently coloured —four colours may be wanted, but not more— the following is the case in which four colours are wanted. Query cannot a necessity for five or more be invented (...)»

Hamilton trabajaba en ese momento en *teoría de cuaterniones*, y respondió cuatro días después a *De Morgan*, con sarcasmo:

«I am not going to attempt your quaternion of colour very soon».

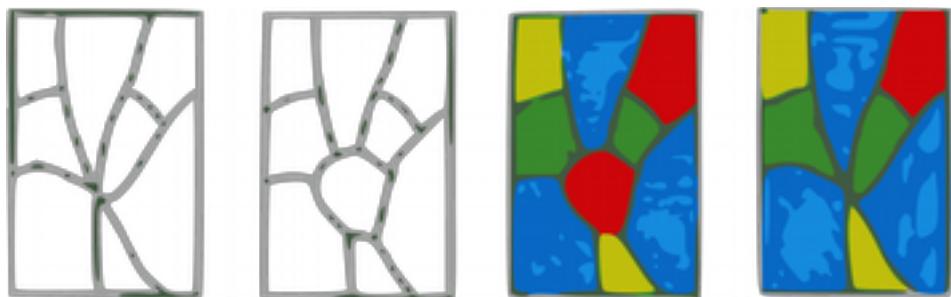
Decepcionado, *De Morgan* se puso en contacto con otros matemáticos que no mostraron excesivo interés por el tema. Tras su fallecimiento, ocurrido en 1871, el problema de los cuatro colores pareció olvidarse. Pero, siete años más tarde, en junio de 1878, el matemático *Arthur Cayley* realizó la siguiente pregunta en un encuentro de la *London Mathematical Society*:

«Has a solution been given of the statement that in colouring a map of a country, divided into counties, only four colours are required, so that no two adjacent counties should be painted in the same colour?».

Cayley estaba realmente interesado por el problema y, en 1879, publicó una nota sobre el tema en los *Proceedings of the Royal Geographical Society* admitiendo la dificultad del tema:

«*I have not succeeded in obtaining a general proof: and it is worth while to explain wherein the difficulty consists»*

Cayley observó que, a la hora de abordar la prueba, se podían imponer condiciones más restrictivas a los mapas a colorear. En particular, bastaba con limitarse a *mapas cúbicos*, es decir, aquellos en los que hay exactamente tres regiones en cada punto de encuentro. En efecto, supongamos un mapa en el que hay más de tres regiones en alguno de los puntos de encuentro. Sobre este punto se puede pegar un pequeño parche que produce un mapa cúbico. Si se puede colorear este mapa con cuatro colores, se puede obtener también un 4-coloreado del mapa original, simplemente colapsando el parche en un punto como se puede observar en la imagen inferior.



14. Ilustración: Transformación a vectores del mapa original, parcheado, coloreado y coloreado final.

Así, a partir de ahora, solo nos preocuparemos de mapas cúbicos.

Por otro lado, la conjetura de 4-coloreado sobre mapas planos equivale a la conjetura de 4-coloreado sobre mapas esféricos. En efecto, basta con realizar la proyección estereográfica de la esfera de dimensión 2 sobre el plano⁹⁴; el hecho de que dos regiones vecinas deban tocarse en una línea de longitud no nula garantiza que el coloreado de mapas sobre la esfera de dimensión 2 equivale al coloreado de mapas planos.

Las ideas de Alfred Kempe

El abogado *Alfred Kempe* se interesó por el *problema de los cuatro colores* tras la pregunta de *Cayley* en la *London Mathematical Society*. En 1879 publicó una solución al problema en la revista *Nature*. Al año siguiente publicó una versión simplificada en los *Proceedings of the London Mathematical Society*, corrigiendo algunas erratas de su primera demostración.

Su prueba se realizaba por inducción sobre el número de regiones colindantes con una fijada previamente y utilizaba como herramienta clave una generalización de la fórmula de *Euler* para poliedros (convexos)⁹⁵ a mapas.

¿Por qué usó esta herramienta? Porque es posible pasar de poliedros a mapas del siguiente modo: se infla el poliedro sobre una esfera, se proyecta estereográficamente y se obtiene de este modo una proyección del poliedro sobre el plano. Ahora, si C_i representa la cantidad de regiones que tienen i regiones vecinas en el mapa, *Kempe* demostró que:

94 https://es.wikipedia.org/wiki/Proyecci%C3%B3n_estereogr%C3%A1fica

95 La fórmula de Euler para poliedros (convexos) afirma que: n^o de caras – n^o de aristas + n^o de vértices = 2. https://es.wikipedia.org/wiki/Teorema_de_Euler_para_poliedros

$$4C_2 + 3C_3 + 2C_4 + C_5 - C_7 - 2C_8 - 3C_9 - \dots = 12$$

por lo que debe de existir necesariamente algún $C_{i=1,2,3,4,5}$ no nulo. Es decir, todo mapa cúbico debe contener al menos una región con a lo sumo cinco regiones vecinas. Es decir, cada mapa contiene al menos un digón, un triángulo, un cuadrado o un pentágono.

Además, si $C_i = 0$ para $i=2,3,4$, debe ser $C_5 \geq 12$, es decir un mapa cúbico que no contiene digones, triángulos o cuadrados debe contener al menos 12 pentágonos.

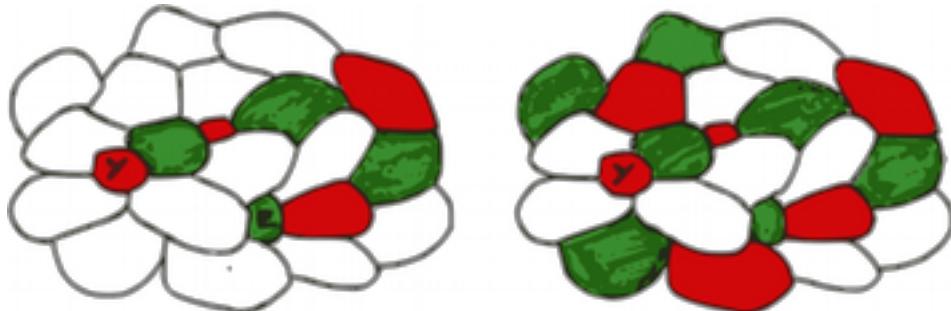
Como hemos comentado, *Kempe* realizó su prueba de la conjectura por inducción. Dado un mapa cúbico M , si X es un país en él, denotamos por $v(X)$ el número de sus regiones vecinas. La prueba se hace por inducción sobre el número de regiones. Como M es un mapa cúbico, sabemos que existe una región X con $v(X) \leq 5$. La hipótesis de inducción es que $M - \{X\}$ es 4-coloreable y debe probarse que entonces M también lo es.

Hay cuatro casos posibles, que $v(X) = 2, 3, 4$ ó 5 .

Los dos primeros casos son inmediatos (quedan colores suficientes para utilizar). Para hacer la prueba en los casos de $v(X) = 4$ ó 5 , *Kempe* introdujo el concepto de *componente* que introducimos a continuación.

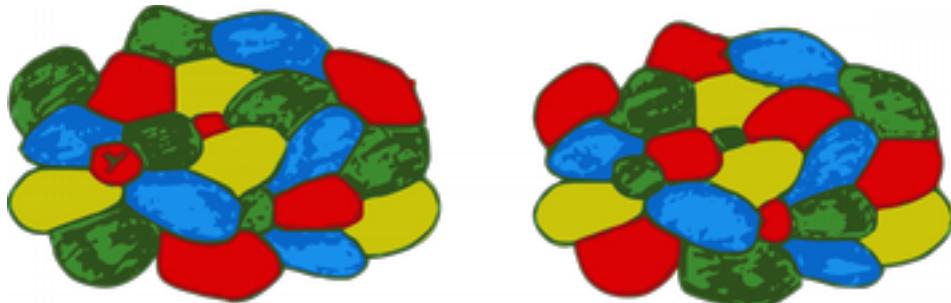
Si Z e Y son dos regiones, Y de color rojo y Z de color verde en un mapa 4-coloreado, se llama *cadena de Kempe* rojo-verde de Y a Z un camino que va de Y a Z , alternando los colores rojo y verde de región en

región. Una *componente* rojo-verde de Y es el conjunto de todas las regiones Z del mapa, tales que existe una cadena de *Kempe* rojo-verde de Y a Z .



15. Ilustración: Cadena de Kempe y Componente de Kempe

La importancia de la noción de componente radica en que en una componente rojo-verde cualquiera de un mapa 4-coloreado se pueden invertir los colores rojo y verde para obtener un nuevo 4-coloreado respetando la regla de los 4 colores.



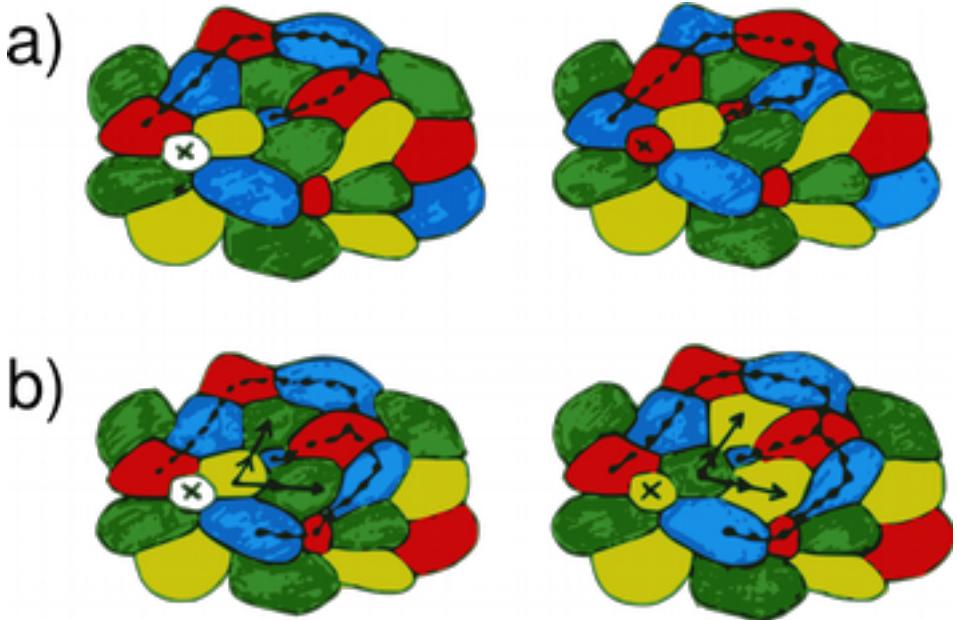
16. Ilustración: Mapa original y mapa obtenido por inversión de la componente rojo-verde.

Continuando con la prueba de *Kempe*, en el caso de $v(X)=4$ se distinguen dos posibilidades:

a) X_3 no está en la componente rojo-azul de X_1 .

b) X_3 está en la componente rojo-azul de X_1 .

En el caso a) se invierten el rojo y el azul en esta componente y se libera un color (el rojo) para X . En el caso b), como X_2 no está en la componente amarillo-verde de X_4 , se invierten los colores en esta componente, liberando un color (amarillo) para X .



En el último paso de la prueba de *Kempe*, se estudia el caso en el que $v(X)=5$. De nuevo hay dos posibilidades:

a) que X_2 no pertenezca a la componente amarilla-verde de X_5 o X_2 no pertenezca a la componente azul-verde de X_4 , o

b) que X_2 pertenezca a la componente amarilla-verde de X_5 y X_2 pertenezca a la componente azul-verde de X_4 .

En el caso a), suponemos que X_2 no pertenece a la componente amarilla-verde de X_5 (se procede de igual modo en el otro caso). Se invierten el amarillo y el verde en esta componente y se libera un color (amarillo) para X .

En el caso b) se invierten las componentes roja-azul de X_1 y roja-amarilla de X_3 , para liberar un color (rojo) para X .

Y de este modo, *Alfred Kempe* pensó que había finalizado con la prueba de la conjectura. Pero, en 1890, el matemático *Percy John Heawood* publicó el artículo *Map Colour Theorem* en el *Journal of Pure and Applied Maths*. Había encontrado (como el mismo comentaba *muy a su pesar*) un mapa en el que la prueba de Kempe no funcionaba. El problema radicaba en la demostración del caso b) de $v(X)=5$. Kempe no se dio cuenta de que las componentes amarilla-verde de X_5 y azul-verde de X_4 podían cruzarse. Y en tal caso las componentes roja-azul de X_1 y roja-amarillo de X_3 no se pueden invertir simultáneamente.

Kempe admitió su error en los *Proceedings of the London Mathematical Society*, y el 9 de abril de 1891, en un encuentro de esta sociedad matemática afirmó:

«*My proof consisted of a method by which any map can be coloured with four colours. Mr. Heawood gives a case in which the method fails,*

and thus shows the proof to be erroneous. I have not succeeded in remedying the defect, though it can be shown that the map which Mr. Heawood gives can be coloured with four colours, and thus his criticism applied to my proof only and not to the theorem itself.»

Es decir, debía seguir intentándose probar la conjetura. A pesar de este error, *Heawood* usó parte de los argumentos de *Kempe* para probar el teorema de los cinco colores (es decir, que cualquier mapa geopolítico plano se puede colorear con a lo sumo cinco colores) y otros resultados sobre el coloreado de mapas sobre superficies. Y, de hecho, el argumento de *Kempe* fue crucial en la prueba definitiva del teorema, como veremos a continuación.

El paso de mapas a grafos. Un acercamiento alternativo

Para progresar en la prueba del teorema se cambió de enfoque, abordando el enunciado de manera dual al sustituir mapas por grafos⁹⁶. En efecto, si se marca la capital de cada región en el mapa a colorear y se unen las capitales de países contiguos, se obtiene el llamado *grafo dual* del mapa. Colorear el mapa equivale a pintar las capitales (es decir, los *vértices* del grafo) asignando distintos tonos a dos capitales unidas por una trayectoria (una *arista* del grafo).

96 <https://es.wikipedia.org/wiki/Grafo>

Los grafos duales de mapas son siempre *planares*⁹⁷, es decir, se puede dibujar en el plano una representación concreta del grafo, en la cual las aristas no se cortan excepto en un eventual vértice común.

El número de regiones vecinas de una región dada de un mapa se corresponde en el grafo dual con el grado⁹⁸ de cada vértice (número de aristas incidentes).

Además, se comprueba que un mapa es cúbico si y sólo si su grafo dual es *triangulado* (grafo planar en el que cada cara tiene exactamente 3 aristas).

Además de pasar de mapas a grafos, se realiza un cambio de estrategia al abordar la prueba. Supongamos por un momento que el teorema de los cuatro colores fuera falso, es decir, que existen mapas (grafos) que no pueden 4-colorearse. Entre estos mapas (grafos) que necesitan cinco colores o más, deber de haber alguno con el menor número posible de regiones: es lo que se denomina un *minimal criminal*. Así, un *minimal criminal* no puede 4-colorearse, pero un mapa (grafo) con menos regiones (vértices) *sí*. De este modo, probar el teorema de los cuatro colores equivale a demostrar que no existen *minimales criminales*.

Por cierto, con su argumentación *Kempe* demostró que un *minimal criminal* no puede contener digones, triángulos o cuadrados, y falló al intentar probar que tampoco puede contener pentágonos. Si lo hubiese conseguido, habría quedado establecida la conjetura al no existir *minimales criminales*

97 https://es.wikipedia.org/wiki/Grafo_plano

98 [https://es.wikipedia.org/wiki/Grado_\(teor%C3%ADA_de_grafos\)](https://es.wikipedia.org/wiki/Grado_(teor%C3%ADA_de_grafos))

(recordemos que cualquiera de ellos debe contener obligatoriamente una de las anteriores cuatro configuraciones).

La demostración correcta del teorema de los 4 colores toma la de Kempe, pero para la inducción, en vez de eliminar un único vértice, se elimina un determinado trozo del grafo (una *configuración*).

El plan de la prueba consiste entonces en encontrar *conjuntos inevitables* K (una familia finita de configuraciones⁹⁹ tal que todo grafo contiene una copia de una de K). Si K estuviese formado solo por configuraciones *reducibles*¹⁰⁰, la demostración del teorema de los cuatro colores estaría terminada ya que en tal caso no podría existir un *minimal criminal*.

La estrategia general de la demostración del teorema de los cuatro colores no difiere excesivamente de lo realizado por *Kempe* en su trabajo de 1879. Ahora la prueba se hace por inducción sobre el número de vértices. El primer paso de la inducción consiste en comprobar que cualquier grafo que tiene a lo sumo cuatro vértices es 4-coloreable.

El paso inductivo general consiste en quitar uno o varios vértices al grafo, reduciendo el problema a un caso con un menor número de vértices, que por hipótesis inductiva se asume que es 4-coloreable. La cuestión es saber extender el coloreado para incluir el vértice (o vértices) extra, o en

99 Una configuración es un ciclo con vértices internos triangulados.

100 \mathbf{k} es una *configuración reducible*, si se puede deducir el coloreado de cualquier grafo que contenga a \mathbf{k} , a partir de un grafo menor.

generalmente, encontrar una manera de cambiar el coloreado sobre los vértices dados para poder extenderlo al vértice (o vértices) extra.

Por otro lado, gracias a la prueba de Kempe, sabemos que los vértices de grado menor o igual a 4 son reducibles, con lo que se puede asumir que nuestro grafo no contiene vértices de grado menor que 5. Y en tal caso, recordemos que la fórmula de *Euler* asegura que debe de haber al menos doce vértices de grado 5.

Durante años, diferentes investigadores desarrollaron nuevos métodos para demostrar la *reducibilidad*. Pero con cada conjunto inevitable producido aparecían algunas de sus configuraciones resistentes a todos los esfuerzos por demostrar su reducibilidad. Así, era necesario encontrar un método que produjera conjuntos inevitables sin “malas” configuraciones; todos ellos se basan, de hecho, en la fórmula de *Euler*.

Los ordenadores entran en juego

En 1969, el matemático alemán *Heinrich Heesch* sistematizó la prueba de la *reducibilidad* desarrollando un algoritmo que intentó implementar en un ordenador. Realizó diversas pruebas en una máquina *CDC1604A*¹⁰¹ y entró en contacto con su antiguo alumno *Wolfgang Haken* que trabajaba en Estados Unidos. Le comentó que la demostración de la *reducibilidad* necesitaba estudiar *solamente* 8900 configuraciones.

101 https://en.wikipedia.org/wiki/CDC_1604

A través de su denominado *algoritmo de descarga*, *Heesch* proponía un método de construcción de *conjuntos inevitables*: consideraba el grafo como una *red eléctrica*. Asociaba a cada vértice una *carga* inicial de $6-d(v)$ (donde $d(v)$ es el grado del vértice v) y, usando la fórmula de *Euler* (adaptada a grafos a través de los mapas planos), demostraba que la suma de las cargas en un grafo triangulado es 12. Al desplazar las cargas eléctricas sobre la red (con su *algoritmo de descarga*) la suma total no variaba. Con este sistema, su objetivo era eliminar de esta *red eléctrica* los vértices de carga negativa, obteniendo un conjunto de configuraciones con vértices de cargas positivas o nulas. Como todo grafo triangulado es de carga total 12, debía contener al menos una de las 8900 configuraciones citadas anteriormente, que formaba entonces un *conjunto inevitable*. Una vez obtenida la extensa lista de *configuraciones inevitables*, probando su *reducibilidad*, se tendría una prueba inductiva del teorema.

En 1976, *Ken Appel* y *Wolfgang Haken* dieron una prueba del teorema de los cuatro colores cuyos principales ingredientes eran precisamente los conceptos de *reducibilidad* y *descarga*.

La primera prueba de Appel, Haken y John A. Koch usaba un algoritmo de descarga muy sofisticado, que producía una lista de 1936 *configuraciones inevitables*, cada una de las cuales se demostraba reducible con la ayuda de un ordenador. Modificando consecutivamente el algoritmo de descarga, encontraron otro (con 300 reglas de descarga) produciendo 1482 *configuraciones inevitables* y comprobando su *reducibilidad* mediante un ordenador programado por Koch. Necesitó 1200 horas de cálculo. Appel y

Haken completaron la demostración en 1976, tras seis años de duro trabajo: más de cien páginas impresas, cuatrocientos microfilms con dibujos y la verificación de varios miles de casos particulares. En ese año, todo este trabajo precisaba más de cincuenta días de trabajo en los tres ordenadores de la Universidad de Illinois, con un programa de varios miles de líneas... es decir, nadie podía comprobar la veracidad o falsedad del teorema *a mano*.

Appel y Haken reconocieron el papel fundamental del acercamiento de Kempe en la demostración que ellos mismos realizaron:

«Kempe's argument was extremely clever, and although his “proof” turned out not to be complete, it contained most of the basic ideas that eventually led to the correct proof one century later.»

La mayor parte de la comunidad matemática aceptó esta prueba como irrefutable, pero algunas personas estaban en contra, argumentando que no se trataba de una demostración matemática: la máquina había verificado que una enorme cantidad de mapas podían colorearse usando como mucho cuatro colores pero, ¿y si existía un mapa que el ordenador no hubiese contemplado?

En 1996, un grupo de investigadores del *Instituto de Tecnología de Georgia* publicó un artículo en el que reducían las complicaciones (probando que la *inevitabilidad* de un conjunto reducible podía demostrarse con menos ejemplos que en la prueba de Appel y Haken), su conjunto inevitable era *solo* de tamaño 633. Usaron únicamente 32 reglas de descarga y reemplazaron la comprobación *a mano* de la *inevitabilidad* por una prueba formal verificable

con un ordenador. El proceso final de comprobación necesitaba únicamente tres horas en cualquier ordenador doméstico.

Posteriores investigaciones han ido confirmando cada una de las etapas de esta última prueba, utilizando diferentes programas y métodos, pero siempre con la ayuda de un ordenador para comprobar cálculos complejos.

El teorema de los cuatro colores ha estimulado muchas nuevas líneas de investigación. Existen numerosas conjeturas matemáticas de las cuales el problema planteado por Francis Guthrie en 1852 no es más que un caso especial.

Observaciones finales

La demostración de Appel y Haken fue la primera prueba de un gran teorema matemático que utilizó herramientas informáticas. ¿Pero, es realmente una demostración? ¿Se puede garantizar la corrección de la compilación realizada por un ordenador? ¿Las computadoras son infalibles? ¿Se puede aceptar como válida una afirmación que solo una máquina, y no una mente humana, puede comprobar?

Tras la demostración del teorema de los cuatro colores, otras pruebas matemáticas se han apoyado en los ordenadores, como la solución de la conjetura de Kepler¹⁰² sobre el empaquetamiento óptimo de esferas¹⁰³ en 1998

102 https://es.wikipedia.org/wiki/Conjetura_de_Kepler

103 https://es.wikipedia.org/wiki/Empaquetamiento_de_esferas

o la clasificación de los grupos simples¹⁰⁴ finitos en 2004 (que depende de cálculos imposibles de ejecutar con detalle a mano). Y cada vez más pruebas de teoremas matemáticos requieren la comprobación de cálculos complejos imposibles de ejecutar a mano.

De cualquier manera, en el *Congreso Internacional de Matemáticos* celebrado en 2018 en Río de Janeiro, *Peter Kronheimer* y *Tomasz Mrowka* propusieron una nueva idea para abordar *una* demostración del teorema de los cuatro colores¹⁰⁵. Sugirieron emplear sus nuevos resultados de topología algebraica para abordar el problema de coloreado de las aristas de un grafo con tres colores (equivalente al coloreado de vértices con cuatro colores). En ese momento lo enunciaron como *A motivating project* que proporcionaría una prueba del teorema de los cuatro colores sin la necesidad de utilizar un ordenador. Habrá que esperar para ver si su aproximación es fructífera.

The Four Colour Theorem is the tip of the iceberg, the thin end of the wedge and the first cuckoo of Spring. —W. T. Tutte (1978)

Referencias

- [1] K. Appel, W. Haken, *Every planar map is four colourable, Part I: discharging*, Illinois Journal of Mathematics 21 (1977) 429-490.
- [2] K. Appel, W. Haken, J. Koch, *Every planar map is four colourable, Part II: reducibility*, Illinois Journal of Mathematics 21 (1977) 491-567.

104 https://es.wikipedia.org/wiki/Teorema_de_clasificaci%C3%B3n_de_grupos_simples

105 Puede verse la conferencia completa en este enlace: <https://youtu.be/UW8KmtY6LlI>

[3] M. Macho Stadler, *Mapas, colores y números*, Descubrir las matemáticas hoy: Sociedad, Ciencia, Tecnología y Matemáticas 2006 (2008) 41-68.

<https://sctmates.webs.ull.es/modulo1lp/4/mmacho.pdf>

[4] N. Robertson, D.P. Sanders, P. Seymour, R. Thomas, A new proof of the four-colour theorem, *Electronic Announcements of the AMS* 2 (1996) 17-25.

[5] R. Wilson, *Four colors suffice: how the map problem was solved*, Princeton University Press, 2002

GLOSARIO LEXICOGRÁFICO

GLOSARIO LEXICOGRÁFICO DE LA OBRA

Compilado por la Asistente de Edición **Maiara Beatriz Cercasi**

Aplicación móvil

Software diseñado y desarrollado para ser ejecutado en un dispositivo móvil y/o tableta.

Aprendizaje supervisado

Se refiere al proceso mediante el cuál un programa informático puede realizar predicciones, a partir de patrones que aprende continuamente con base en datos. Dicho en otras palabras, es la automatización del proceso de creación de modelos analíticos que permiten que un programa informático se adapte a situaciones nuevas (Ana Loyo, 2018).

Argumento

Razonamiento deductivo que ofrece en sus premisas las pruebas de su conclusión.

Avería

Cualquier anomalía o desperfecto que afecte al funcionamiento de un equipo.

Axioma

Principio indemostrable que se acepta como punto de partida para el desarrollo de una teoría formal.

C

Lenguaje de programación de propósitos generales que abarcan desde la programación de sistemas operativos y de lenguajes informáticos, hasta el desarrollo de programas de escritorio, aplicaciones Web, y herramientas a más alto nivel.

Ciberataque

Accionar malintencionado, producido por medios electrónicos, y efectuado con el fin de acceder a información sin autorización, modificarla y/o impedir el buen funcionamiento de los servicios asociados a ella, comprometiendo la disponibilidad, integridad y confidencialidad de dicha información.

Ciencia

Disciplina que por medio de un método estudiado, aceptado y reproducible, emplea el razonamiento y la observación, para producir conocimiento demostrable mediante la experimentación, y a partir del cual pueden inferirse principios y leyes generales capaces de predecir sucesos futuros, y teorías falsables.

Ciencia aplicada

Aquella ciencia que enfoca el estudio en razón de su utilidad.

Ciencia fáctica

Aquella ciencia que requiere de demostración empírica, es decir, que demanda probar sus teorías mediante la experimentación.

Ciencia formal

Aquella ciencia cuya validación se realiza por medio de la forma del razonamiento.

Ciencia teórica

También llamada ciencia básica o ciencia pura. Es aquella que se enfoca en la obtención del conocimiento en sí mismo, con independencia de cualquier aplicación.

Computación

Proceso donde dados valores de entrada (*input*) se obtiene un valor de salida esperado (*output*). Esto se logra a partir de un proceso que puede ser un algoritmo o una función, ya que toma los valores de entrada y a través de una serie de pasos o evaluaciones obtiene los valores de salida. Las máquinas que realizan este proceso son las computadoras.

Comunicación de datos

Forma en la cual la información digital (datos), se transmite a un receptor, por medio de una señal codificada, desde una fuente de

origen (transmisor), a través de un canal determinado (medio).

Comunicación óptica

Medio de transmisión que transporta la información en forma de pulsos de luz.

Convicción

Acto de convencerse sobre una verdad que a partir de las pruebas presentadas, no puede ser negada razonablemente.

CouchDB

Base de datos de código abierto, de tipo NoSQL.

Credencial

Componente que interviene en la autenticación, el cual permite constatar que un usuario es quién afirma ser.

Credencial criptográfica

Credencial cuya información se encuentra cifrada mediante un conjunto de funciones hash combinadas, las cuáles no poseen un

algoritmo derivado que permita que su valor hash sea revertido.

Creencia

Acto voluntario en el que se da crédito absoluto y se asume como cierta, una afirmación o hecho determinado, sin mediar un análisis racional ni constatación (formal o empírica) de aquello que se afirma.

Dato anecdótico

Cualquier información obtenida a partir de cualquier hecho circunstancial pasado.

Deducción

Conclusión que se desprende necesariamente de las proposiciones de un razonamiento válido.

Demostación

Acción de explicar y comprobar, mediante métodos específicos, reproducibles y falsables, la forma en la que se produce un determinado hecho o suceso.

Dogma

Proposición acabada (es decir, no flexible a cambios) y que se acepta como cierta e innegable, sin posibilidad de ser sometida a ensayo.

Esto implica que como tal, no puede ser refutada.

Enmascaramiento de datos

Reemplazo de datos sensibles con el objetivo de proteger la información de una revelación no intencional.

Enunciado categórico

Afirmación compuesta por dos términos, sujeto y predicado, donde la categoría del primer término, se incluye o excluye, total o parcialmente, de la categoría del segundo.

Enunciado categórico compuesto

Enunciados categóricos simples concatenados, que pueden surgir a partir de una traducción del lenguaje cotidiano.

Enunciado empírico básico

Enunciados de observación directa sobre ciertas características específicas de las entidades observables.

Enunciado hipotético

Enunciado condicional en el que se afirma una consecuencia que se cumple toda vez que la condición sea verdadera.

Enunciado teórico

Enunciado que, pudiendo ser particular o universales, emplea al menos un término teórico.

Epistemología

Rama de la filosofía que tiene por objeto el estudio del conocimiento científico, y define los criterios a los que debe someterse toda investigación.

Escepticismo científico

Duda o desconfianza que se tiene sobre una afirmación, desde una perspectiva científica.

Evidencia científica

Demostración formal o empírica, clara y manifiesta, obtenida a partir de un método válido desde una perspectiva epistemológica.

Explicación

Conjunto de proposiciones que solo establecen relaciones causales, temporales (u otras no lógicas) entre el enunciado y su conclusión.

Explicación científica

Hipótesis provisoria sujeta a falsabilidad. Esto implica, que aquello que se afirma puede ser refutado si se lo somete a prueba empírica, y nunca es final y definitivo.

Falacia

Argumento psicológicamente persuasivo que sin ser válido lo aparenta.

Falacia formal

Aquella falacia que viola al menos una de las reglas de cualquiera de los tipos de silogismos.

Falacia no formal

Aquella falacia que conserva una forma válida pero con conclusiones falsas, bien sea por falta de atinencia lógica, o bien, por ambigüedad.

Falsar

Acción de desmentir una teoría a partir de pruebas empíricas.

Física

Estudio de las propiedades y dinámica de la materia, representadas a través de variables en ecuaciones matemáticas.

Geocodificación

Proceso para encontrar coordenadas espaciales a partir de alguna otra información geográfica georeferenciada, como una dirección o un código postal.

Geocodificación reversa

Proceso inverso a la geocodificación, mediante el cual se busca una información geográfica

georeferenciada a partir de unas coordenadas.

Geolocalización

Acción de determinar la posición geográfica actual de un dispositivo conectado a una red de comunicaciones, como Internet.

Grafo

Diagrama empleado para resolver problemas lógicos, topológicos y de cálculo combinatorio mediante el cual, empleando nodos y aristas que los conectan, se representan pares de relaciones.

Hipótesis

Suposición que se establece de forma provisoria como base para una investigación científica y tiene por objetivo la confirmación o desestimación de algo que se supone posible.

Impresora

Equipo periférico conectado al ordenador que sirve para realizar

una copia de los datos almacenados en formato electrónico a papel u otro tipo de material dependiendo de las características de la misma.

Inducción

Es aquella que implica un grado de probabilidad aparentemente suficiente sobre su conclusión, pero donde ésta, no se deduce necesariamente de sus premisas.

Inferencia

ver conclusión.

Ionic

Marco de trabajo de código abierto para el desarrollo de aplicaciones móviles híbridas.

Lenguaje C

ver C.

Ley empírica

Enunciado de observación directa sobre características específicas de entidades observables, que hace uso solo de términos empíricos.

Lógica

Disciplina que tiene a su cargo el estudio de los métodos que permiten establecer la validez de un razonamiento, entendiendo como tal al proceso mental que, partiendo de ciertas premisas, deriva en una conclusión inferida sobre la base de éstas.

Lógica matemática

Conocida también como lógica simbólica, es aquella que emplea el uso de un lenguaje de símbolos abstracto —denominado lenguaje formal—, como base de un sistema.

Mantenimiento

Conjunto de operaciones realizadas sobre un equipo o programa informático con el fin de que los mismos sean capaces de continuar con su correcto funcionamiento.

Mantenimiento correctivo

Mantenimiento que se realiza cuando se detecta el mal

funcionamiento de un equipo o programa informático.

Mantenimiento preventivo (hardware)

Mantenimiento que se efectúa para garantizar el buen funcionamiento de un equipo informático, con el fin de prevenir futuras averías, realizándose una puesta a punto de los componentes más comunes.

Mapa de usuario

Forma de visualizar información con unas características determinadas con el fin de comprender los posibles problemas o desafíos de un programa informático.

Método axiomático

Método científico de las ciencias formales, basado en axiomas para la formulación de teoremas.

Método científico

Método utilizado por las ciencias para la producción de conocimiento válido.

Método hipotético-deductivo

Método científico de las ciencias fácticas, que se basa en hipótesis para formular teorías científicas sujetas a refutación.

Minería de texto

Proceso de análisis de texto no estructurado que busca encontrar patrones, tendencias y/o relaciones en su contenido, con el fin de extraer información que sirva para la toma de decisiones en diferentes ámbitos.

Mosaico vectorial

Composición de pequeños fragmentos, denominados teselas, que se emplea en el trazado de mapas y planos geográficos sobre plataformas Web, generados a partir de gráficos vectoriales.

Neurona

Célula del sistema nervioso cuyo funcionamiento puede compararse con el de un interruptor que será activado si recibe un estímulo de

entrada suficiente y de ser así enviará un pulso de salida.

Neurona artificial

Cada una de las unidades de procesamiento, interconectadas entre sí, y que en su conjunto, se establecen como una simplificación matemática del modelo neuronal biológico, denominada *red neuronal artificial*.

Observación

Acto de examinar un hecho o suceso, y describirlo de forma cuantitativa o cualitativa.

Pensamiento

Conjunto de todos los procesos mentales de cualquier índole y naturaleza, pudiendo abarcar desde un proceso imaginativo, e incluso un recuerdo hasta una asociación libre y al propio razonamiento.

Pensamiento crítico

Cualidad cognitiva que hace referencia a la capacidad de analizar

ideas, hechos, situaciones hasta incluso afirmaciones y teorías, mediante el uso simultáneo de diferentes formas de pensamiento, que abarcan desde el análisis racional hasta el pensamiento creativo, abstracto y lateral.

Postulado

Proposición que se acepta como verdadera sin necesidad de demostración.

Pseudociencia

Disciplina que puede parecer o proclamarse como científica, pero que no presenta una o más de las características presentes en las disciplinas científicas (ver definición de ciencia).

Razonamiento

Proceso mental que partiendo de ciertas premisas deriva en una conclusión inferida sobre la base de éstas.

Red neuronal artificial

Conjunto de unidades de procesamiento conectadas entre sí, a través de conexiones ponderadas que permiten la transmisión de información.

SACRED

Acrónimo de «*Sistema de Autenticación por Credenciales Disociadas*». Se trata de un mecanismo de autenticación de usuarios, que aísla a los usuarios de sus respectivas credenciales de acceso, valiéndose del uso concomitante de funciones hash, sistemas de persistencia y generación de datos en tiempo de ejecución, como artilugio para la generación y manejo seguro de credenciales criptográficas de alta entropía.

Scrum

Estructura conceptual basada en los principios ágiles de desarrollo de Software para administrar procesos

de fabricación de este tipo de productos.

Semiotica

También conocida como «*Teoría General de los Signos*» de C. S. Peirce, es la metadisciplina que se encarga del estudio de los signos, y de entender la relación de estos entre sí, su significado y uso.

Sesgo / sesgo cognitivo

En psicología cognitiva, se refiere a un error sistemático de pensamiento, que conduce a afirmaciones falsas, formuladas a partir de procesos heurísticos, basados en juicios subjetivos realizados a partir del conocimiento parcial de uno o más eventos auto experimentados, de la intuición, suposiciones y creencias individuales.

SIG

ver «Sistema de Información Geográfica».

Silogismo

Razonamiento deductivo en el que a partir de dos premisas, se infiere una conclusión.

Silogismo categórico

Silogismo que emplea tres proposiciones categóricas, dos de ellas como premisa y la tercera, como conclusión.

Silogismo disyuntivo

Silogismo que como premisa, tiene un enunciado disyuntivo.

Silogismo hipotético

Silogismo en el cual su premisa es un enunciado condicional.

Sistema de autenticación

Mecanismo que permite validar la identidad de un usuario.

Sistema de diseño

Colección de elementos que pueden ser reutilizables y que funcionan de forma conjunta para desarrollar aplicaciones.

Sistemas de Información Geográfica

Conjunto de herramientas de información y gestión que permiten trabajar con datos geográficamente referenciados, tales como coordenadas geográficas, cuya finalidad es el tratamiento de información espacial que normalmente se almacena en una base de datos con extensiones que soporten información geográfica.

Teorema

Proposición solo demostrable a través de la lógica, que no necesita ser falsada.

Teorema Matemático

ver «teorema».

Teoría

Hipótesis provisoria de algo que aún no ha sido falsado.

Tesis

Conclusión confirmada obtenida a partir del razonamiento, la cual

supone la confirmación de una hipótesis.

Trazado de mapas

Técnica que consiste en el proceso de diseño, implementación, generación y distribución de mapas, datos geospaciales, y provisión de Sistemas de Información Geográfica (SIG) como servicio o funcionalidad de una aplicación Web (Li, 2011).

Unicode

Estándar que asigna un número único a cada carácter existente, independientemente del alfabeto o la plataforma sobre la que se opere.

UTF-8

Sistema de codificación de caracteres que permite el uso de Unicode mediante representaciones de una a cuatro secuencias de 8 bits, por lo que un carácter Unicode, codificado en UTF-8, puede tener entre 8 y 32 bits (es decir, entre 1 y 4 bytes).

Variable nativa (CSS)

Variable definida por el usuario que puede ser usada en una hoja de estilos en cascada (CSS), a la que se le puede asignar un nombre y un valor determinado, pudiendo ser reutilizadas en clases posteriormente declaradas.

Verdad fáctica

Toda afirmación confirmada por experimentación a través de un número limitado de casos.

Verdad formal

Consecuencia lógica de proposiciones previamente establecidas como verdaderas. Verdad lógica.



MAIARA BEATRIZ CERCASI

Asistente de Edición

(Argentina, 1986)

Desarrolladora Web Full Stack y Analista de Pruebas. Tras cursar la carrera de Economía en la Universidad Nacional de Cuyo, Argentina, y habiéndose capacitado como Desarrolladora Web *Full Stack*, incursiona actualmente en el sector de Aseguramiento de la Calidad del Software (*Software Quality Assurance*), desempeñándose como Analista de Pruebas de Control de Calidad (*QA Analyst*), con el objetivo de obtener una certificación *ISTQB Foundation Level (CTFL)* de la *International Software Testing Qualifications Board (ISTQB)*.

ÍNDICE GENERAL

Resumen de Contenidos.....	9
Índice de autoras.....	17
Prólogo.....	43
Glosario lexicográfico.....	461

PARTE I. LÓGICA Y EPISTEMOLOGÍA

1. Hallazgo y determinación del Conocimiento Científico.....	71
Validez del conocimiento.....	72
Metadisciplinas que definen los criterios de validación del conocimiento.....	73
Epistemología.....	73
Lógica.....	73
Semiótica.....	74
Contraste de conceptos.....	74
Diferencia entre divulgación científica y comunicación científica.....	74
Diferencia entre contrario y contradictorio.....	75
Diferencia entre verdad y validez.....	76
Diferencia entre verdad formal y verdad fáctica.....	77
Diferencia entre razonamiento y pensamiento.....	77
Diferencia entre deducción e inducción.....	78
Diferencia entre argumento y explicación.....	78
Diferencia entre objetivo y subjetivo.....	79
Diferencia entre «bueno y malo», «correcto e incorrecto».....	81
Diferencia entre falacia y sesgo cognitivo.....	83
Diferencia entre dato anecdótico y evidencia científica.....	84
Diferencia entre dogma y explicación científica.....	84
Diferencia entre creencia y convicción.....	85
Diferencia entre ciencia formal y ciencia fáctica.....	86
Diferencia entre axioma y postulado.....	86
Diferencia entre observación y demostración.....	87
Diferencia entre ciencia y pseudociencia.....	88
Diferencia entre escepticismo científico y pensamiento crítico.....	88
Diferencia entre ciencia teórica y ciencia aplicada.....	89
Diferencia entre teoría, teorema, tesis e hipótesis.....	89

2. Nociones de Lógica y las Formas del Razonamiento.....	91
Introducción.....	91
Enunciados categóricos.....	92
Forma típica de un enunciado categórico.....	92
Distribución.....	95
Enunciados categóricos compuestos.....	98
Enunciados hipotéticos.....	99
Diferencia entre enunciado hipotético y enunciado bicondicional.....	100
Función de verdad.....	100
Silogismos.....	102
Silogismo categórico.....	102
Términos de un silogismo.....	102
Forma de los silogismos.....	103
La figura de un silogismo.....	104
Reglas de validación de un silogismo categórico.....	105
Silogismo disyuntivo.....	108
Silogismo hipotético.....	109
Falacias.....	111
Falacias formales.....	112
Falacias no formales.....	113
Lógica matemática.....	116
Elementos básicos de la lógica simbólica.....	116
Validación de enunciados lógicos y argumentos.....	118
3. Introducción al método científico.....	121
Fundamentos epistemológicos.....	121
Diferentes tipos de discurso.....	121
La neutralidad emotiva en el lenguaje científico.....	122
Principios de la definición.....	124
El método científico.....	126
El método axiomático de las ciencias formales.....	127
Propiedades.....	127
El método hipotético-deductivo de las ciencias fácticas.....	128
Tipos de enunciados científicos.....	128
Enunciados empíricos básicos.....	129
Generalizaciones y leyes empíricas.....	129
Enunciados teóricos.....	130
Validez de una teoría científica y predictibilidad.....	131
4. Las Ciencias Informáticas y su relación con la física y la matemática.	132
Bibliografía de la primera parte.....	133

PARTE II. FUNDAMENTOS TEÓRICOS

5. Fundamentos Físicos de las Ciencias Computacionales.....	137
Definiciones previas.....	137
Física.....	137
Computación.....	137
Objetivo.....	138
Introducción.....	138
Teorías y procesos físicos sobre los que se concibe la computación.....	139
Electrónica.....	140
Circuitos integrados (CI).....	140
Sensores y señales.....	141
Teoría de semiconductores.....	142
Transistores y diodos.....	144
Electromagnetismo.....	145
Tecnología inalámbrica y ondas electromagnéticas.....	146
Restricciones a la transmisión de señales.....	148
Referencias.....	148
Bibliografía consultada por la autora.....	149
Fuentes de consulta sugeridas.....	150
6. Introducción a la Comunicación de Datos.....	151
Introducción.....	151
Codificación de los datos.....	152
Codificación de la información en sistema binario.....	152
Codificación de la señal.....	153
Sistemas NRZ de no retorno a cero (NRC).....	154
Sistema RZ de retorno a cero (RC).....	155
Sistemas bifásicos.....	155
Bipolar.....	156
Transmisión de los datos.....	156
Transmisión de datos en serie.....	156
Tipos de transmisión en serie.....	157
Sincronismo.....	157
Estándar RS-232.....	158
Aplicaciones.....	159
USB (Universal Serial Bus).....	159
Redes USB.....	162
Transmisión de datos en paralelo.....	163
Referencias.....	163
7. Introducción a las Comunicaciones Ópticas.....	165
La fibra óptica frente a otros medios de transmisión.....	166

Aplicaciones.....	168
Fundamentos de las comunicaciones ópticas.....	168
Fundamentos físicos.....	170
Propiedades de la luz.....	170
Reflexión y Refracción de la luz.....	170
Fundamentos técnicos.....	173
Composición de la fibra óptica.....	173
Clasificación de la fibra óptica.....	175
Apertura numérica.....	177
Pérdidas en fibra óptica.....	177
Observaciones finales.....	179
Bibliografía consultada por la autora.....	180

PARTE III. INFORMÁTICA APLICADA E INVESTIGACIÓN

B BASES DE DATOS

8. Enmascaramiento de Datos.....	183
Sobre el enmascaramiento de datos.....	183
Casos prácticos.....	183
Diferencia entre enmascarar y encriptar datos.....	187
Enmascarado estático y enmascarado dinámico.....	188
Enmascaramiento de datos y deductibilidad.....	190
9. Generación de datos de prueba.....	193
Sobre la generación de datos.....	193
Aspectos a tener en cuenta en la elección del generador.....	194
Variedad de tipos de datos.....	194
Creación de tipos personalizados.....	195
Correlación entre campos.....	195
Distribución de los datos.....	195
Tipos de salidas.....	196
Análisis de herramientas de código abierto.....	196
Generación de datos aleatorios para MySQL®.....	200
Generación de números.....	200
Generación de fechas u horas.....	201
Generación de textos.....	201
Automatización.....	202
10. Generación de Hojas de Estilo Escalables.....	205

D DESARROLLO WEB

Dependencia con el marcado HTML.....	206
Nombramiento de clases y marcado.....	207
Orden y legibilidad.....	208
Documentación del código.....	210
11. Variables nativas en CSS.....	211
Definición.....	211
Escritura y uso de variables nativas.....	211
Semejanzas y diferencias con otros preprocesadores.....	212
Compatibilidad de navegadores.....	214
12. Sistemas de Diseño.....	215
Elementos que componen un sistema de diseño.....	215
Fundamentos.....	215
Definición de reglas de estilo.....	217
Librería.....	218
Dinamismo.....	218
Convenciones de nombre, documentación y guías.....	218
13. Mapas de Viaje para la Identificación de Problemas.....	221
Elementos.....	222
El alcance.....	222
Los datos.....	222
La organización.....	223
Las personas.....	224
Fases de la experiencia.....	224
Puntos de contacto.....	224
Desencadenantes.....	224
Actitudes.....	225
El armado.....	225
Conclusión.....	227
D DESARROLLO MÓVIL	
14. CouchDB para el manejo de datos fuera de línea en aplicaciones basadas en Ionic.....	229
Definiciones previas.....	229
Aplicación móvil.....	229
Ionic.....	229
CouchDB.....	230
Introducción.....	230
Tipos de aplicaciones.....	230
Manejo de los datos.....	231

Solución propuesta.....	232
Funcionamiento de PouchDB.....	232
Limitaciones.....	233
Conclusión.....	234

D DESARROLLO ÁGIL Y GESTIÓN DE PROYECTOS

15. Implementación de Scrum en empresas de Desarrollo de Software.....	235
Definiciones previas.....	235
Scrum.....	235
Desarrollo ágil.....	236
Empresa de desarrollo de Software.....	236
Introducción.....	236
Fundamentos de <i>Scrum</i>	237
Marco de trabajo.....	237
Equipo de Scrum.....	237
Dueño de producto.....	238
Equipo de desarrollo.....	238
Scrum Master.....	238
Eventos de Scrum.....	239
Sprint.....	239
Planificación del Sprint.....	240
Scrum diario.....	240
Revisión del Sprint.....	241
Retrospectiva del Sprint.....	241
Artefactos de Scrum.....	241
Backlog del producto (pila de producto).....	241
Backlog del Sprint.....	242
Aspectos a tener en cuenta para la implementación de Scrum en el ámbito de trabajo. Aproximación práctica.....	243
Consideraciones a ser tenidas en cuenta al inicio de la implementación.....	244
Aspectos a considerarse durante la implementación.....	245
Datos adicionales.....	248

H HARDWARE

16. Mantenimiento Preventivo y Correctivo de Impresoras Láser y de Inyección de Tinta.....	251
Definiciones previas.....	251
Mantenimiento preventivo.....	251
Mantenimiento correctivo.....	251
Impresora.....	252

Avería.....	252
Tipos de impresoras.....	252
Impresora de inyección de tinta.....	252
Impresora láser.....	253
Componentes principales.....	253
Componentes de una impresora de inyección de tinta.....	253
Componentes de una impresora láser.....	254
Averías frecuentes.....	255
Averías comunes en impresora de inyección de tinta.....	255
Averías comunes en impresoras láser.....	256
17. Mantenimiento Preventivo de Ordenadores.....	259
Componentes.....	260
Limpieza de hardware.....	261
Limpieza de equipo de sobremesa.....	262
Limpieza de ordenadores portátiles.....	264
I INTELIGENCIA ARTIFICIAL Y MINERÍA DE TEXTO	
18. Introducción a las Redes Neuronales aplicadas al Aprendizaje Supervisado.....	265
Definiciones previas.....	265
Red neuronal artificial.....	265
Aprendizaje supervisado.....	265
Base biológica: la neurona.....	266
Neurona artificial.....	267
Primeras redes neuronales artificiales.....	268
McCulloch-Pitts MPN.....	268
Perceptrón.....	269
Perceptrones multicapa.....	271
Retropropagación.....	272
Elección de arquitectura.....	274
Redes Neuronales convolucionales (CNN).....	277
Redes Neuronales recurrentes (RNN).....	280
Hopfield Network.....	281
Referencias.....	282
19. Análisis de Texto sobre Redes Sociales.....	285
Introducción.....	285
Elementos del análisis.....	286
Grafos.....	286
Minería.....	288

Metodología.....	291
Diseño de los grafos.....	293
Análisis del texto.....	296
Resultados y discusión.....	299
Bibliografía.....	306

P PROGRAMACIÓN

20. Guía de Referencias del Lenguaje C.....	309
Acerca de C.....	309
Estructura de un programa C.....	309
Requerimientos sintácticos.....	311
Compilación básica y ejecución.....	312
Archivos de encabezado más comunes de la biblioteca estándar de C.....	313
Tipos de datos.....	314
Enteros, números de coma flotante y subtipos correspondientes.....	314
El tipo char y las cadenas de texto.....	316
Matrices (tipo: <i>Array</i>).....	316
Acceso a elementos de un array.....	317
Valores booleanos.....	318
Valores nulos (void).....	318
Conversión de tipos o encasillamiento (<i>typecast</i>).....	318
Variables y Constantes.....	319
Ámbito y alcance de las variables.....	319
Definición y asignación de variables.....	319
Especificador de tipo de almacenamiento.....	320
Constantes.....	321
Enumeración.....	322
Punteros.....	322
Estructuras.....	324
Definición de estructuras.....	324
Definición de cadenas de texto dentro de una estructura.....	325
Creación de variables basadas en estructuras.....	326
Acceso a las variables de una estructura.....	326
Ejemplos de uso de estructuras.....	326
Observaciones finales sobre las estructuras.....	328
Uniones.....	329
Operadores básicos.....	330
Operador unario <i>sizeof</i>	330
Estructuras de control de flujo.....	331
Estructuras de control condicionales.....	331
if/else.....	331

switch.....	331
Estructuras de control iterativas.....	332
while.....	332
for.....	332
do.....	332
break y continue.....	332
Declaración de retorno.....	333
goto.....	333
return.....	334
Preprocesadores comunes y definición de macros.....	334
Usos comunes de la biblioteca stdio.h.....	335
Manejo básico de entrada y salida.....	335
Manejo de salida con printf().....	335
Manejo de la entrada con scanf().....	336
Manipulación de archivos.....	337
Apertura y cierre de archivos con fopen() y fclose().....	337
Lectura y escritura de archivos con fread() y fwrite().....	339
Manipulación del punto de acceso interno con fseek() y ftell().....	340
Ejemplos de lectura y escritura de archivos.....	340
Manipulación de argumentos de línea de comandos.....	341
Asignación dinámica de memoria con las funciones malloc() y free() de stdlib.h.....	342
Entendiendo el montón.....	342
Funcionamiento del montón.....	342
Utilizar el montón.....	343
Liberar memoria.....	344
malloc() y typecast.....	345
Comprobación del puntero.....	345
Referencias.....	345

S SEGURIDAD DE LA INFORMACIÓN

21. Prevención de Ciberataques en el Sector de Salud.....	347
Resumen.....	347
Introducción.....	348
Confidencialidad, integridad y disponibilidad de los datos.....	349
Amenazas a la seguridad en ámbitos de la salud.....	350
Motivos y Consecuencias.....	352
Recomendaciones.....	353
Marcos o modelos de trabajo en Salud.....	356
Conclusión.....	360

22. Aproximación a un sistema de codificación para el tratamiento decimal de caracteres Unicode con codificación UTF-8.....	361
Definiciones previas: Unicode y UTF-8.....	361
Necesidad de codificar datos de entrada y manipularlos aritméticamente.....	362
Necesidad de transmitir datos codificados de forma alfanumérica.....	363
Necesidad de efectuar validaciones aritméticas.....	364
Necesidad de un tratamiento decimal de los datos.....	364
Evaluación de Base64 como medio para el tratamiento aritmético de datos.....	365
Sobre Base64.....	365
Mecanismo de codificación.....	366
Codificación de caracteres Unicode en Base64 del lado del cliente.....	366
Codificación de caracteres Unicode en Base64 del lado del servidor.....	367
Validación de datos a partir de cadenas codificadas con Base64.....	368
Dificultades halladas en el uso de Base64.....	369
Reversión de una cadena codificada con Base64 hacia sus valores decimales correspondientes (Base64 a decimal).....	370
Base64 a binario.....	370
Binario a decimal.....	373
Aproximación a un nuevo sistema de codificación de caracteres que permita un tratamiento aritmético de los datos en lenguajes de alto nivel.....	375
Codificación alfanumérica de caracteres Unicode.....	376
Resumen de funciones necesarias.....	377
Codificación HTML decimal.....	379
Resumen de funciones necesarias.....	380
Obtención del valor H a partir de una cadena XZf.....	381
Resumen de funciones necesarias.....	382
Integridad de los datos.....	383
Integridad de la cadena.....	383
Integridad del token.....	383
Resumen de rutinas necesarias.....	384
Anexo A: Funciones PHP para la conversión de Base64 a binario y decimal.....	386
Anexo B: caracteres Unicode de reemplazo y su equivalencia decimal en HTML.....	387
Referencias.....	391
23. Sistema de Autenticación por Credenciales Criptográficas Disociadas (SACRED).....	393
Resumen.....	393
Definiciones previas.....	396
Sistema de autenticación.....	396
Credencial.....	396

Credencial criptográfica.....	396
Funcionamiento del sistema.....	396
Proceso de autenticación.....	396
Generación de la credencial.....	398
Funciones necesarias y obtención de valores <i>hash</i>	398
Implementación a alto nivel (ejemplo en PHP).....	398
Generación del UID.....	399
Funciones necesarias y obtención de valores <i>hash</i>	399
Implementación a alto nivel (ejemplo en PHP).....	400
Persistencia de la credencial.....	400
Cuestiones relativas a la seguridad del sistema de archivos.....	400
Validación de credenciales.....	401
Implementación a alto nivel (ejemplo en PHP).....	401
El objeto Usuario.....	401
Sobre la persistencia del objeto Usuario en bases de datos relacionales.....	403
Modificación de credenciales.....	403
Sobre la interfaz de entrada de los datos.....	405
Mantenimiento del Sistema.....	405
Parámetros constantes.....	405
Tiempo de vida.....	405
Período de inactividad.....	408
Mecanismos internos de control complementario.....	409
Renovación de credenciales.....	409
Renovación por caducidad.....	410
Renovación por extravío.....	410
Purga de credenciales y recolección de basura.....	412

S SISTEMAS DE INFORMACIÓN GEOGRÁFICA

24. Configuración de un Entorno Virtual para la Centralización de datos en Sistemas de Información Geográfica (SIG).....	415
Resumen.....	415
Materiales necesarios.....	416
Procedimiento.....	417
Requerimientos.....	417
Creación de la máquina virtual.....	417
Configuración del Sistema Operativo.....	418
Instalación de la base de datos.....	419

25. Diseño de mapas geográficos aplicados al desarrollo web y móvil.....	423
Definiciones previas.....	423
Geocodificación y geocodificación reversa.....	423
Geolocalización.....	424
Diseño e implementación de funciones de geocodificación y geolocalización.....	424
Elección de una base cartográfica.....	424
Carga e interacción con la cartografía.....	425
Creación básica de un mapa.....	427
Referencias.....	428
26. Introducción al trazado de mapas mediante mosaicos vectoriales.....	429
Definiciones previas.....	429
Mosaicos vectoriales.....	429
Trazado de mapas Web.....	429
Antecedentes.....	430
Formatos gráficos.....	430
Tipos de trazado.....	430
Trazado de mosaicos vectoriales.....	432
Tipos de formato para los datos de gráficos vectoriales.....	433
Formatos de fichero único sin compresión.....	433
Formato TopoJSON.....	433
Formato PBF (<i>Protocolbuffer Binary Format</i>).....	434
Características de los mosaicos vectoriales.....	434
Interacción.....	434
Estilización de geometrías al vuelo.....	435
Rotación de etiquetas.....	436
Escalabilidad y resolución.....	436
Peso de los mapas.....	436
Referencias.....	436
 PARTE IV. MATEMÁTICA TEÓRICA	
27. El teorema de los cuatro colores: un problema topológico demostrado con ayuda de un ordenador.....	441
Planteamiento del problema y primeras simplificaciones.....	442
Las ideas de Alfred Kempe.....	446
El paso de mapas a grafos. Un acercamiento alternativo.....	451
Los ordenadores entran en juego.....	454
Observaciones finales.....	457
Referencias.....	458

Derechos sobre la propiedad intelectual de la obra «*Onna Bugeisha — Guerreras de la Ciencia*» realizados a través de inscripción privada en **Safe Creative S.L.**, bajo los siguientes códigos de registro, titulares de los derechos y bajo las siguientes licencias:

Código	Concepto	Titular	Licencia
1912252750253	Portada	Eugenia Bahit	CC BY NC ND
1912272754163	Diseño gráfico de la obra	Eugenia Bahit	CC BY NC ND
---	Texto de contraportada	---	Dominio Público
1912272754156	Textos de la obra (excepto texto de contraportada)	Eugenia Bahit	CC BY
1912222729630	Sistema de Autenticación por Credenciales Criptográficas Disociadas (SACRED)	Eugenia Bahit	CC BY
1912222729647	Aproximación a un sistema de codificación para el tratamiento decimal de caracteres Unicode	Eugenia Bahit	CC BY
1912222729654	Guía de Referencias del Lenguaje C	Eugenia Bahit	CC BY
1912222729661	Introducción al método científico	Eugenia Bahit	CC BY
1912222729678	Nociones de Lógica y las Formas del Razonamiento	Eugenia Bahit	CC BY
1912222729685	Hallazgo y determinación del Conocimiento Científico	Eugenia Bahit	CC BY
1912222729708	Fundamentos Físicos de las Ciencias Computacionales	Fernanda Hernández González	CC BY
1912222729715	Introducción a las Comunicaciones Ópticas	Bibiana Mollinedo Rivadeneira	CC BY
1912222729722	Introducción a la Comunicación de Datos	Bibiana Mollinedo Rivadeneira	CC BY
1912222729746	Introducción a las Redes Neuronales aplicadas al Aprendizaje Supervisado	Andrea C. Navarro Moreno	CC BY
1912222729753	Generación de datos de prueba	Andrea C. Navarro Moreno	CC BY
1912222729760	Enmascaramiento de Datos	Andrea C. Navarro Moreno	CC BY
1912222729777	Parte 3.2. Desarrollo Web	Josefina Monsegur	CC BY
1912222729807	Introducción al trazado de mapas Web mediante mosaicos vectoriales	Jéssica Sena	CC BY
1912222729814	Diseño de mapas geográficos aplicados al desarrollo web y móvil	Jéssica Sena	CC BY
1912222729821	CouchDB para el manejo de datos fuera de línea en aplicaciones basadas en Ionic	Jéssica Sena	CC BY
1912222729838	Implementación de Scrum en una empresa de Desarrollo de Software	Milagros Mora	CC BY
1912222729845	Parte 3.5. Hardware	Lorena Santamaría Jiménez	CC BY
1912222729869	Análisis de Texto sobre Redes Sociales	Indira Luz Burga Menacho	CC BY
1912222729876	Prevención de Ciberataques en el Sector de Salud	Florencia Vilardel Tignanelli	CC BY
1912222729890	Configuración de un Entorno Virtual para la Centralización de datos en Sistemas de Información Geográfica (SIG)	Montserrat Ortega Gallart	CC BY
1912222729906	El teorema de los cuatro colores: un problema topológico demostrado con ayuda de un ordenador	Marta Macho Stadler	CC BY

Prensa y Distribución: Eugenia Bahit <eugenia@hdmag.press>. **Asuntos legales** deberán dirigirse a la editora y a las respectivas titulares de los derechos previamente informados. **Hackers & Developers™** es marca comercial propiedad de Eugenia Bahit.

Visitar la Web del Libro

El contenido de este libro se encuentra disponible de forma abierta y gratuita, en el sitio Web:

<https://onnabugeisha.hdmag.press>



El sitio Web de Hackers & Developers™ Press, así como los micrositios dentro del dominio hdmag.press, son posibles gracias a la colaboración de un **equipo de personas voluntarias**.

Salvador Macías
Líder de Proyecto Web

Hackers & Developers™ Press agradece a **Salvador Macías** y equipo, por su invaluable labor profesional, y hacer posible el sitio Web de *Onna Bugeisha — Guerreras de la Ciencia*. ¡Nuestro enorme agradecimiento a Salva y a todo el equipo!

* FIN *

Fuertes y decididas como las Onna Bugeisha, las guerreras samurái del antiguo Japón feudal, este grupo de **doce mujeres profesionales de las Ciencias Informáticas, Físicas y Matemáticas**, se ha abierto paso en un mundo que ha tenido por costumbre, intentar socavar el intelecto de toda aquella persona que no encajase en determinados estereotipos de género fundados en creencias irracionales. Doce Mujeres. Doce profesionales especializadas en diferentes campos y disciplinas, se han abierto su propio camino con la frente en alto y sin pedir permiso, y hoy nos ofrecen el fruto de su trabajo profesional con esta magnífica **obra científico-académica**, que denota un gran dominio intelectual y se ofrece a la comunidad de la mano de **Hackers & Developers™ Press**, en un **formato libre y abierto** para servir tanto de guía e inspiración a estudiantes, como de consulta e intercambio a profesionales de las diversas áreas y disciplinas científicas.

