TABASCO

COBATAB
COLEGIO DE BACHILLERES DE TABASCO

TIC 5to semestre Desarrollo de Sistemas

Módulo III

Guía del Estudiante

SUBMÓDULO I

SISTEMAS DE INFORMACIÓN

SUBMÓDULO II

PROGRAMACIÓN

DATOS DEL ESTUDIANTE

Nombre_____

Plantel _____ Grupo _____ Turno _____

COLEGIO DE BACHILLERES DE TABASCO

M.C. ERASMO MARTÍNEZ RODRÍGUEZ
Director General

L.C.P. SONIA LÓPEZ IZQUIERDO
Directora Académica

DRA. GISELLE OLIVARES MORALES
Subdirectora de Planeación Académica

DR. JOSÉ LUIS MADRIGAL ELISEO
Subdirector de Servicios Educativos

MTRO. GERARDO LÓPEZ GARCÍA
Subdirector de Educación Media Superior Abierta a Distancia

MTRO. ALLAN LÓPEZ GALLEGOS
Jefe del Departamento de Capacitación para el Trabajo

CAPACITACIÓN: TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN
Modulo III. Desarrollo de Sistemas.
Submódulo I. "Sistemas de Información"
Submódulo II. "Programación"

En la realización del presente material participaron:

Jorge Alberto Almeida Alejandro
Inés Filiberta Díaz Ramírez
Carlos Alberto Segovia Zapata
Adriana Hernández Almeida
Andrea Esteban Alor
Belda Liz García Córdova
Dioselina de la Cruz Ruiz
Elsy Ruth Domínguez Izquerdo
Gabriel Ramón Velueta
Griselda Elizabeth Hernández Magaña.
Juan Manuel Zentella Lázaro
Juana Antonia Moo Salvador
Karina Del Carmen Gómez López
Libi García Córdova
Marta Elba Helston Arias
Miguel Bolio Álvarez
Milton Hernández García
Miriam Córdoba Hernández
Patricia del Carmen Rabanales Cervantes
Roberto Alonso Pérez López

Edición: 2023-2024A

REVISADO POR: Mtro. Allan López Gallegos

Este material fue elaborado bajo la coordinación y supervisión de la Dirección académica del Colegio de Bachilleres del Estado de Tabasco. www.cobatab.edu.mx

CONTENIDO

Fundamentación V
Enfoque de la disciplina VIII
Ubicación del Módulo X
Mapa de la Capacitación TIC XI
Relación de los contenidos con los aprendizajes claves XII
Competencias Genéricas XIII
Competencias Profesionales básicas XVI
Evaluación por competencias XVII
Instrumentos de Evaluación XIX
Simbología XX
Temario XXI

Submódulo 1. Desarrollo de Sistemas 1
Competencias 2
DOSIFICACIÓN PROGRAMÁTICA SUBMÓDULO 1 3
Encuadre Submódulo 1 Sistemas de Información 5
Situación Didáctica 6
¿Cómo resuelvo la situación didáctica? 7
Evaluación diagnóstica Submódulo 1 9
Lectura 1. Fundamentos de sistemas: "Metodología para el desarrollo de software" 11
Actividad 1. Crucigrama "Mi Método" 21
Lectura 2. Identificación de necesidades 25
Actividad 2 Solicitud de Servicio de Sistemas 29
Actividad 3: Esquema "Identificación de necesidades" 32
Lección Construye – T ¿Qué voy a ver en este curso 34
Lectura 3. Análisis del Sistema "Modelando mis Datos" 36
Actividad 4. Análisis del Sistema "Modelando mis Datos" 48
Lectura 4. Diseño del Sistema "Diseñando mis pantallas" 51
Actividad 5. Opción A: "Diseñando mi interfaz" 60
Lectura 5. Codificación "Soy un programador" 66
Actividad 6. Rally de Codificación 71
Actividad 6.1 ¡Ahora te toca a ti! 77
Lectura 6. Validación y pruebas de un Si 79
Actividad 7. Cuadro sinóptico "Validación y pruebas de un SI" 83

Lectura 7. Base de Datos: Tablas y Relaciones en Access 86
Actividad 8. "Mapa conceptual, "todo cabe en una tabla" 98
Lectura 8. Consultas en Access "¿Qué quieres buscar?" 100
Actividad 9. "Herramientas de consulta en Access" 109
Práctica 1. Consultas en Access "¿qué quieres buscar?" 110
Lectura 9. Formularios e Informes en Access "Otra forma de ver los datos" 113
Actividad 10. Tabla "Encontrando sus diferencias" 125

Submódulo 2. Programación 131
Propósito del Submódulo 132
Aprendizajes Esperados 132
Competencias 132
DOSIFICACIÓN PROGRAMÁTICA SUBMÓDULO 2 133
Encuadre Submódulo 2 135
Situación Didáctica 136
¿Cómo soluciono la situación didáctica? 137
Evaluación diagnostica submódulo 2 138
Lectura 1: Lógica de programación 141
Actividad 1. "Mi tabla de algoritmos" 146
Lectura 2: Decisiones y ciclos 149
Actividad 2. Algoritmo y Diagrama de Decisiones y Ciclos 157
Lectura 3: Lenguajes de programación. 161
Actividad 3. Crucigramas "Lenguajes de Programación" 167
Lectura 4. Metodología de la Programación estructurada 170
Actividad 4. Mapa conceptual de "Metodologías de Programación" 178
Lección Construye-T Lección 4 "Decisiones y emociones creativas" 181
Lectura 5. Programación utilizando un lenguaje de alto nivel. 183
Práctica 1. Mi primer programa 194
Lectura 7. Estructuras repetitivas 209
Practica 3: Calculadora de comisiones de ventas 217
Lectura 8. Aprendamos sobre Listas en Python 221
Práctica 4 Manipulación de listas 231

Fundamentación

Teniendo como referencia el actual desarrollo económico, político, social, tecnológico y cultural de México, la Dirección General del Bachillerato inició la Actualización de Programas de Estudio integrando elementos tales como los aprendizajes claves, contenidos específicos y aprendizajes esperados, que atienden al Nuevo Modelo Educativo para la Educación Obligatoria. Además de conservar el enfoque basado en competencias, hacen énfasis en el desarrollo de habilidades socioemocionales y abordan temas transversales tomando en cuenta lo estipulado en las políticas educativas vigentes.

Considerando lo anterior, dicha actualización tiene como fundamento el Programa Sectorial de Educación 2013-2018, el cual señala que la Educación Media Superior debe ser fortalecida para contribuir al desarrollo de México a través de la formación de hombres y mujeres en las competencias que se requieren para el progreso democrático, social y económico del país, mismos que son esenciales para construir una nación próspera y socialmente incluyente basada en el conocimiento. Esto se retoma específicamente del objetivo 2, estrategia 2.1., en la línea de acción 2.1.4., que a la letra indica: "Revisar el modelo educativo, apoyar la revisión y renovación curricular, las prácticas pedagógicas y los materiales educativos para mejorar el aprendizaje".

Asimismo, este proceso de actualización pretende dar cumplimiento a la finalidad esencial del Bachillerato que es: "generar en el estudiantado el desarrollo de una primera síntesis personal y social que le permita su acceso a la educación superior, a la vez que le dé una comprensión de su sociedad y de su tiempo y lo prepare para su posible incorporación al trabajo productivo", así como los objetivos del Bachillerato General que expresan las siguientes intenciones formativas: ofrecer una cultura general básica; que comprenda aspectos de la ciencia; de las humanidades y de la técnica; a partir de la cual se adquieran los elementos fundamentales para la construcción de nuevos conocimientos; proporcionar los conocimientos, los métodos, las técnicas y los lenguajes necesarios para ingresar a estudios superiores y desempeñarse de manera eficiente, a la vez que se desarrollan las habilidades y actitudes esenciales sin que ello implique una formación técnica especializada, para la realización de una actividad productiva socialmente útil.

El **Componente de Formación Profesional** aporta al estudiantado elementos que le permiten iniciarse en diversos aspectos del sector productivo, fomentando una actitud positiva hacia el trabajo y en su caso, su integración al mismo. Los módulos que conforman este programa son el resultado del trabajo colegiado con personal Docente que imparte esta capacitación en los diferentes subsistemas coordinados por esta Dirección General, quienes brindan su experiencia y conocimientos buscando responder a los diferentes contextos existentes en el país, así como a la formación de una ciudadanía socialmente útil, para que el estudiantado cuente con la opción de iniciar una ruta laboral que le promueva una proyección hacia las diferentes modalidades laborales.

Aunado a ello, en virtud de que la Educación Media Superior debe favorecer la convivencia, el respeto a los derechos humanos y la responsabilidad social, el cuidado de las personas, el entendimiento del entorno, la protección del medio ambiente, la puesta en práctica de habilidades productivas para el desarrollo integral de los seres humanos, la actualización del presente programa de estudios, incluye **temas transversales** que según Figueroa de Katra (2005)[1], enriquecen la labor formativa de manera tal que conectan y articulan los saberes de los distintos sectores de aprendizaje que dotan de sentido a los conocimientos disciplinares, con los temas y contextos sociales, culturales y éticos presentes en su entorno; buscan mirar toda la experiencia escolar como una oportunidad para que los aprendizajes integren sus dimensiones cognitivas y formativas, favoreciendo de esta forma una educación incluyente y con equidad.

De igual forma, con base en el fortalecimiento de la educación para la vida, se abordan dentro de este programa de estudios los **temas transversales**, mismos que se clasifican a través de ejes temáticos de los campos Social. Ambiental, Salud y Habilidad Lectora como en el componente básico, con la particularidad de que se complementan con características propias de la formación para el trabajo. Dichos temas no son únicos ni pretenden limitar el quehacer educativo en el aula, ya que es necesario tomar en consideración temas propios de cada comunidad, por lo que el personal Docente podrá considerar ya sea uno o varios, en función del contexto escolar y de su pertinencia en cada submódulo:

- **Eje transversal Emprendedurismo**: se sugiere retomar temas referentes a la detección de oportunidades y puesta en práctica de acciones que contribuyen a la demostración de actitudes tales como iniciativa, liderazgo, trabajo colaborativo. visión, innovación y creatividad promoviendo la responsabilidad social.
- **Eje transversal Vinculación Laboral**: se recomienda abordar temas referentes a la realización de acciones que permiten al estudiantado identificar los sitios de inserción laboral o autoempleo.
- **Eje transversal Iniciar, Continuar y Concluir sus estudios de nivel superior**: se recomienda abordar temas referentes a los mecanismos que permiten al estudiantado reflexionar sobre la importancia de darle continuidad a sus estudios superiores.

Asimismo, otro aspecto importante que promueve el programa de estudios es la **Interdisciplinariedad** entre asignaturas del mismo semestre, en donde diferentes disciplinas se conjuntan para trabajar de forma colaborativa para la obtención de resultados en los aprendizajes esperados de manera integral, permitiendo al estudiantado confrontarse a situaciones cotidianas aplicando dichos saberes de forma vinculada.

Por otro lado, en cada submódulo se observa la relación de la competencias genéricas y profesionales

[1] Figueroa de Katra, L. (2005). Desarrollo curricular y transversalidad. Revista Internacional Educación Global, Vol. 9. Guadalajara, Jalisco, México. Asociación Mexicana para la Educación Internacional. http://paideia.synaptium.net/pub/pesegpatt2/tetra_ir/tt_ponencia.pdf

básicas, los conocimientos, las habilidades y actitudes que darán como resultado los aprendizajes esperados, permitiendo llevar de la mano al personal Docente con el objetivo de generar un desarrollo progresivo no sólo de los conocimientos, sino también de aspectos ACTITUDINALES.

En ese sentido, el rol Docente dentro del proceso de enseñanza – aprendizaje, tiene un papel fundamental, como lo establece el acuerdo Secretarial 447, ya que el profesorado que imparte el componente de formación profesional, es quien facilita el proceso educativo al diseñar actividades significativas que promueven el desarrollo de las competencias (conocimientos, habilidades y actitudes); propicia un ambiente de aprendizaje que favorece el conocimiento social, la colaboración, la toma responsable de decisiones y la perseverancia a través del desarrollo de habilidades socioemocionales del estudiantado, tañes como la confianza, seguridad, autoestima, entre otras, propone estrategias disciplinares y transversales en donde el objetivo no es la formación de técnicas en diferentes actividades productivas, sino la promoción de las diferentes competencias profesionales básicas que permitan a la población estudiantil del Bachillerato General tener alternativas para iniciar una ruta a su integración laboral, favoreciendo el uso de herramientas tecnológicas de la información y la comunicación; así como el diseño de instrumentos de Evaluación que atiendan al enfoque por competencias.

Enfoque de la Disciplina

La capacitación de Tecnologías de la Información y Comunicación pertenece al campo disciplinar de Comunicación, tiene la finalidad de desarrollar en el estudiantado las habilidades comunicativas, verbales y no verbales para expresarse a través de diversos códigos y herramientas del lenguaje a través de las diferentes tecnologías de la información. Por otra parte, las Tecnologías de la Información y Comunicación se vinculan de manera interdisciplinar tanto con el campo de Matemáticas como con el de Comunicación, ya que aportan los elementos para la resolución de problemas mediante los algoritmos y la programación.

El propósito general de la capacitación de Tecnologías de la Información y Comunicación es: Desarrollar la capacidad para proponer soluciones a problemas del contexto laboral y escolar, mediante la aplicación de las Tecnologías de la Información y de la Comunicación, de forma creativa e innovadora, con una postura ética y responsable como ciudadano digital.

El uso de las Tecnologías de la Información y Comunicación, desde esta capacitación, destaca el manejo avanzado de las aplicaciones de software y hardware para la resolución de problemas de los diferentes ámbitos de la vida cotidiana, desarrollando los aspectos metodológicos, creativos y comunicativos, sin olvidar un comportamiento propositivo en beneficio personal y dentro de la sociedad.

La capacitación de Tecnologías de la Información y Comunicación busca desarrollar en el alumnado las competencias profesionales en las áreas de aplicaciones de oficina, los elementos del hardware. las comunicaciones mediante las redes informáticas, el desarrollo de sistemas y el software de diseño, sin olvidar la promoción de las competencias genéricas, la interdisciplinariedad y los ejes transversales de vinculación laboral, emprendedurismo así como la continuación de sus estudios a nivel superior.

En el contexto curricular de la capacitación de Tecnologías de la Información y Comunicación, el contenido se divide en cuatro módulos que se imparten a partir del tercer semestre con una carga de 7 horas semanales, cada módulo se integra por dos Submódulos en los que se busca desarrollar el manejo de aplicaciones de oficina que permiten elaborar documentos electrónicos con características avanzadas utilizando el procesador de textos y la hoja de cálculo, crear y participar en comunidades virtuales para el intercambio de información incluyendo el ámbito educativo, aplicar mantenimiento al equipo de cómputo; para el desarrollo de sistemas con fundamento en las bases de datos y la programación, mediante la creación de páginas web y el software de diseño lograr comunicar ideas e información, en el entorno laboral y escolar.

Todas estas competencias posibilitan al egresado en su incorporación al mundo laboral o bien para desarrollar procesos productivos independientes de acuerdo con sus intereses profesionales o las necesidades de su entorno social como asistente en las siguientes áreas: administrativas, soporte técnico, área de sistemas, publicidad, y otras, en diferentes instituciones tanto públicas como privadas.

La enseñanza de la Capacitación de Tecnologías de la Información y Comunicación en la formación para el trabajo de los jóvenes, basada en las Normas Técnicas de Competencia Laboral (NTCL) del Consejo Nacional de Normalización y Certificación de Competencias Laborales (CONOCER) se convierte en una necesidad de primer orden para cumplir con las exigencias de los sectores productivos, de contar con personal calificado que permita desarrollar las potencialidades de sus organizaciones al promover productos y servicios tanto en el mercado nacional como en el internacional, además de proporcionar las herramientas técnicas básicas al estudiantado egresado del nivel medio superior, que les permitirán vencer las fronteras e internarse en el mundo global a través de las Tecnologías de la información y de la comunicación (TIC'S), además de la utilización de las Tecnologías del Aprendizaje y del conocimiento (TAC'S).

CINF0376.01 Elaboración de documentos y comunicación mediante el empleo de las características avanzadas aplicaciones de cómputo
UINF0650.01 Preservar el equipo de cómputo, insumos, información y el lugar de trabajo UINF0947.01 Operar las Herramientas de Cómputo en ambiente de red
UINF0948.01 Elaborar documentos de texto mediante el empleo de las características avanzadas de la aplicación cómputo
UINF0949.01 Elaborar hojas de cálculo mediante el empleo de las características avanzadas la aplicación de cómputo

Ubicación del Módulo

<table>
<tr><th>1er. Semestre</th><th>2do. Semestre</th><th>3er. Semestre</th><th>4to.Semestre</th><th>5to. Semestre</th><th>6to. Semestre</th></tr>
<tr><td>Informática I</td><td>Informática II</td><td>Matemáticas III</td><td>Matemáticas IV</td><td rowspan="3">Asignaturas de 5to. Semestre del componente de formación propedéutico</td><td rowspan="3">Asignaturas de 6to. Semestre del componente de formación propedéutico</td></tr>
<tr><td>Inglés I</td><td>Inglés II</td><td>Física I</td><td>Física II</td></tr>
<tr><td rowspan="2">Taller de Lectura y Redacción I</td><td rowspan="2">Taller de Lectura y Redacción II</td><td>Inglés III</td><td>Inglés IV</td></tr>
<tr><td>Asignaturas de 3er. semestre</td><td>Asignaturas de 4to. semestre</td></tr>
<tr><td>Asignaturas de 1er. semestre</td><td>Asignaturas de 2do. semestre</td><td colspan="4">CAPACITACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN</td></tr>
<tr><td colspan="6">TUTORÍAS</td></tr>
</table>

Mapa de la Capacitación TIC

Módulo I
Software de aplicación.

- Submódulo 1. Gestión de archivos de texto. 48 horas. 6 créditos.
- Submódulo 2. Hoja de cálculo aplicado. 64 horas. 6 créditos.

Módulo II
Hardware y comunicaciones.

- Submódulo 1. Comunidades virtuales. 48 horas. 6 créditos.
- Submódulo 2. Mantenimiento y redes de cómputo. 64 horas. 8 créditos.

Módulo III
Desarrollo de sistemas.

- Submódulo 1. Sistemas de Información. 48 horas. 6 créditos.
- Submódulo 2 Programación. 64 horas. 8 créditos.

Módulo IV
Software de diseño.

- Submódulo 1 Páginas web. 48 horas. 6 créditos.
- Submódulo 2. Diseño digital. 64 horas. 8 créditos.

Relación de los contenidos con los aprendizajes claves

Campo Disciplinar: **Comunicación**

EJE	TEMÁTICA GENERAL	COMPONENTE	CONTENIDO CENTRAL	MÓDULO I
EMPRENDIMIENTO	Desarrollo de habilidades personales	Coordinación de Proyectos	Fomentar la habilidad para concretar los medios y esfuerzos de un equipo de trabajo para una acción común y así lograr los objetivos de un proyecto.	SUBMÓDULO I SUBMÓDULO II
		Responsabilidad Social	Generar el compromiso, obligación y deber de contribuir voluntariamente para generar una sociedad más justa y proteger el ambiente por medio de una conducta ética para consigo mismo/a y su entorno.	SUBMÓDULO I SUBMÓDULO II
		Fomento a la innovación y la creatividad	Promover la capacidad de crear e idear algo nuevo y original, además de producir, asimilar y explotar con éxito una novedad, de manera que aporte soluciones inéditas a los problemas y permita responder a las necesidades de la sociedad en general.	SUBMÓDULO I SUBMÓDULO II
	Gestión del recurso humano	Administración del capital humano	Fomentar un mejor desempeño, aprovechamiento, acrecentamiento y mejora de las capacidades, habilidades, experiencias, conocimientos y competencias de las personas como miembros de un equipo de trabajo.	SUBMÓDULO I SUBMÓDULO II
		Comunicación Organizacional	Establecer estrategias de desarrollo, productividad y una mejora de las relaciones tanto internas como externas a fin de obtener un mejor desempeño por parte de los miembros de un equipo de trabajo, por lo que su finalidad se relaciona con los logros, éxitos o fracasos de estos en las organizaciones.	SUBMÓDULO I SUBMÓDULO II
		Manejo de relaciones sociales	Fomentar un desarrollo sólido y sano de sus relaciones personales, sociales, escolares y profesionales utilizando su inteligencia social y emocional	SUBMÓDULO I SUBMÓDULO II
VINCULACIÓN LABORAL	Identificación de las oportunidades laborales o de emprendimiento en su contexto.	Contacto empresarial y/o institucional con la localidad	Análisis de los servicios de trabajo, los distintos tipos de remuneración y promoción de habilidades laborales que sirvan para identificar las características de cada perfil con las áreas fuertes y débiles referentes al ámbito laboral.	SUBMÓDULO I SUBMÓDULO II
		Búsqueda de Empleo	Brindar herramientas para encontrar un empleo acorde a sus actitudes, aptitudes y que responda a las necesidades del mercado laboral.	SUBMÓDULO I SUBMÓDULO II
		Principios básicos de inversión	Conocer y utilizar el capital en cierta actividad económica con el objetivo de incrementarlo. Fomentar la cultura del ahorro y la ubicación del capital durante el proceso de inversión.	N/A
		Creación de hábitos de estudio efectivo	Creación de hábitos de estudio que permitan facilitar el acceso a la universidad y favorezcan sus hábitos de estudio	SUBMÓDULO I SUBMÓDULO II
INICIAR, CONTINUAR, CONCLUIR SUS ESTUDIOS DE NIVEL SUPERIOR	Estrategias y planeación de la vida profesional	Formación de un plan de vida profesional	Planteamiento de objetivos que una persona desea lograr a lo largo de su vida, acompañado por un plan de acción para su consecución.	SUBMÓDULO I SUBMÓDULO II
		Creación de hábitos de estudio efectivo	Creación de hábitos de estudio que permitan facilitar el acceso a la universidad y favorezcan sus hábitos de estudio.	SUBMÓDULO I SUBMÓDULO II

Competencias Genéricas

Competencias Genéricas	Clave
Se autodetermina y cuida de sí	
1. Se conoce y valora a sí mismo y aborda problemas y retos teniendo en cuenta los objetivos que persigue.	
1.1. Enfrenta las dificultades que se le presentan y es consciente de sus valores, fortalezas y debilidades.	CG1.1.
1.2. Identifica sus emociones, las maneja de manera constructiva y reconoce la necesidad de solicitar apoyo ante una situación que lo rebase.	CG1.2.
1.3. Elige alternativas y cursos de acción con base en criterios sustentados y en el marco de un proyecto de vida.	CG1.3.
1.4. Analiza críticamente los factores que influyen en su toma de decisiones.	CG1.4
1.5. Asume las consecuencias de sus comportamientos y decisiones.	CG1.5.
1.6. Administra los recursos disponibles teniendo en cuenta las restricciones para el logro de sus metas.	CG1.6.
2. Es sensible al arte y participa en la apreciación e interpretación de sus expresiones en distintos géneros	
2.1. Valora el arte como manifestación de la belleza y expresión de ideas, sensaciones y emociones.	CG2.1.
2.2. Experimenta el arte como un hecho histórico compartido que permite la comunicación entre individuos y culturas en el tiempo y el espacio, a la vez que desarrolla un sentido de identidad.	CG2.2.
2.3. Participa en prácticas relacionadas con el arte.	CG2.3.
3. Elige y practica estilos de vida saludables	
3.1. Reconoce la actividad física como un medio para su desarrollo físico, mental y social.	CG3.1.
3.2. Toma decisiones a partir de la valoración de las consecuencias de distintos hábitos de consumo y conductas de riesgo.	CG3.2.
3.3. Cultiva relaciones interpersonales que contribuyen a su desarrollo humano y el de quienes lo rodean.	CG3.3.
Se expresa y comunica	
4. Escucha, interpreta y emite mensajes pertinentes en distintos contextos mediante la utilización de medios, códigos y herramientas apropiados	
4.1. Expresa ideas y conceptos mediante representaciones lingüísticas, matemáticas o gráficas.	CG4.1.
4.2. Aplica distintas estrategias comunicativas según quienes sean sus interlocutores, el contexto en el que se encuentra y los objetivos que persigue.	CG4.2.

Competencias Genéricas	Clave
4.3. Identifica las ideas clave en un texto o discurso oral e interfiere conclusiones a partir de ellas.	CG4.3.
4.4. Se comunica en una segunda lengua en situaciones cotidianas	CG4.4.
4.5. Maneja las tecnologías de la información y la comunicación para obtener información y expresar ideas	CG4.5.
Piensa crítica y reflexivamente	
5. Desarrolla innovaciones y propone soluciones a problemas a partir de métodos establecidos	
5.1. Sigue instrucciones y procedimientos de manera reflexiva, comprendiendo como cada uno de sus pasos contribuye al alcance de un objetivo	CG5.1.
5.2. Ordena información de acuerdo a categorías, jerarquías y relaciones	CG5.2.
5.3. Identifica los sistemas y reglas o principios medulares que subyacen a una serie de fenómenos	CG5.3.
5.4. Construye hipótesis y diseña y aplica modelos para probar su validez	CG5.4.
5.5. Sintetiza evidencias obtenidas mediante la experimentación para producir conclusiones y formular nuevas preguntas	CG5.5.
5.6. Utiliza las tecnologías de la información y comunicación para procesar e interpretar información	CG5.6.
6. Sustenta una postura personal sobre temas de interés y relevancia general, considerando otros puntos de vista de manera crítica y reflexiva	
6.1. Elige las fuentes de información más relevantes para un propósito específico y discrimina entre ellas de acuerdo a su relevancia y confiabilidad	CG6.1.
6.2. Evalúa argumentos y opiniones e identifica prejuicios y falacias	CG6.2.
6.3. Reconoce los propios prejuicios, modifica sus puntos de vista al conocer nuevas evidencias, e integra nuevos conocimientos y perspectivas al acervo con el que cuenta	CG6.3.
6.4. Estructura ideas y argumentos de manera clara, coherente y sintética	CG6.4.
Aprende de forma autónoma	
7. Aprende por iniciativa e interés propio a lo largo de la vida	
7.1. Define metas y da seguimiento a sus procesos de construcción de conocimiento	CG7.1.
7.2. Identifica las actividades que le resultan de menor y mayor interés y dificultad, reconociendo y controlando sus reacciones frente a retos y obstáculos	CG7.2.
7.3. Articula saberes de diversos campos y establece relaciones entre ellos y su vida cotidiana	CG7.3.
Trabaja en forma colaborativa	
8. Participa y colabora de manera efectiva en equipos diversos	

Competencias Genéricas	Clave
8.1. Propone maneras de solucionar un problema o desarrollar un proyecto en equipo, definiendo un curso de acción con pasos específicos	CG8.1.
8.2. Aporta puntos de vista con apertura y considera los de otras personas de manera reflexiva	CG8.2
8.3. Asume una actitud constructiva, congruente con los conocimientos y habilidades con los que cuenta dentro de distintos equipos de trabajo	CG8.3.
Participa con responsabilidad en la sociedad	
9. Participa con una conciencia cívica y ética en la vida de su comunidad, región, México y el mundo	
9.1. Privilegia el diálogo como mecanismo para la solución de conflictos	CG9.1.
9.2. Toma decisiones a fin de contribuir a la equidad, bienestar y desarrollo democrático de la sociedad	CG9.2.
9.3. Conoce sus derechos y obligaciones como mexicano y miembro de distintas comunidades e instituciones, y reconoce el valor de la participación como herramienta para ejercerlos	CG9.3.
9.4. Contribuye a alcanzar un equilibrio entre el interés y bienestar individual y el interés general de la sociedad	CG9.4.
9.5. Actúa de manera propositiva frente a fenómenos de la sociedad y se mantiene informado	CG9.5.
9.6. Advierte que los fenómenos que se desarrollan en los ámbitos local, nacional e internacional ocurren dentro de un contexto global interdependiente	CG9.6.
10.Mantiene una actitud respetuosa hacia la interculturalidad y la diversidad de creencias, valores, ideas y prácticas sociales	
10.1. Reconoce que ala diversidad tiene lugar en un espacio democrático de igualdad de dignidad y derechos de todas las personas, y rechaza toda forma de discriminación	CG10.1.
10.2. Dialoga y aprende de personas con distintos puntos de vista y tradiciones culturales mediante la ubicación de sus propias circunstancias en un contexto más amplio	CG10.2.
10.3. Asume que el respeto de las diferencias es el principio de integración y convivencia en los contextos local, nacional e internacional	CG10.3.
11.Contribuye al desarrollo sustentable de manera crítica, con acciones responsables	
11.1. Asume una actitud que favorece la solución de problemas ambientales en los ámbitos local, nacional e internacional	CG11.1.
11.2. Reconoce y comprende las implicaciones biológicas, económicas, políticas y sociales del daño ambiental en un contexto global interdependiente	CG11.2.
11.3. Contribuye al alcance de un equilibrio entre los intereses de corto y largo plazo con relación al ambiente	CG11.3.

Competencias Profesionales Básicas

CPBTIC1

•1. Integra información digital mediante la creación de documentos electrónicos, empleando software de aplicación, como procesadores de textos y editor de imágenes de manera responsable y creativa en ámbitos laborales, escolares y de la vida cotidiana

CPBTIC2

•2. Prepara información a través de la manipulación de datos y fórmulas elaborando gráficos en una aplicación de hoja de cálculo, resolviendo de manera creativa e innovadora, situaciones en diversos ambientes y contextos

CPBTIC3

•3. Plantea el uso, creación ya dminsitración de paaltaformas electrónicas de consulta, comunicación y distribución de contenidos multimedia, proponiendo comunidades virtuales que le permita comunicarse, favoreciendo su autoparendizaje en un ambiente innovador en sus diferentes contextos

CPBTIC4

•4. Desarrolla acciones correctivas para los problemas de operación del equipo de cómputo mediante la aplicación de mantenimiento preventivo y correctivo de acuerdo a las especificaciones del fabricante, prolongando la vida útil del equipo, mostrando responsabilidad e iniciativa en diversos ámbitos

CPBTIC5

•5. Propone el diseño de sistemas de información, a partir del análisis de las necesidades de los usuarios, permitiendo la solución de problemas de manera responsable e innovadora en diferentes contextos

CPBTIC6

•6. Construye sistemas de información organizacionales mediante la codificación y compilación de instrucciones algorítmicas pertinentes utilziando lenguajes de programación y bases de datos para cumplir con los requerimientos de funcionalidad y rendimiento establecidos en el diseño de sistemas de información asumiendo la frustración como parte del proceso en ambientes laborales,e ducativos y de la vida cotidiana

CPBTIC7

•7. Construye sitios web creativos mediante software de diseño web, para transmitir información electrónica diversa a gran escala de manera responsable y empática en constextos laborales, educativos y de la vida cotidiana

CPBTIC8

•8. Elabora diversos recursos gráficos publicitarios utilizando software de diseño, permitiendo su publicación en medios digitales e impresos para comunicar ideas o emociones aplicables a contextos laborales, escolares y de la vida cotidiana, en un ambiente ético e innovador, mostrando flexibilidad y apertura a diferentes puntos de vista

Evaluación por Competencias

gesvinromero.com
(https://gesvin.files.wordpress.com/2018/05/evaluacic3b3ncompetenciasaspectosimportantesdisec3b1o-ebook-bloggesvin.jpg)

La Evaluación por competencias funge como un elemento que de manera complementaria al programa de estudios se convierte en un dispositivo que fortalece la formación del estudiante, al encauzar acciones, reflexiones y proporcionar situaciones en las que se desarrollan las competencias de manera pertinente; representan un espacio para enriquecer el propósito de la capacitación y la estructura misma de los elementos que constituyen el programa, a fin de atender a las funciones fundamentales del planteamiento curricular para el nivel medio superior, como son: la formación de jóvenes como ciudadanos competentes y personas capaces de construir sus proyectos de vida y la preparación para ingresar al mundo de trabajo.

La demostración del desarrollo de la competencia profesional básica, así como de los aprendizajes esperados, se realiza a partir del diseño de un producto integrador y se lleva a cabo al término del desarrollo del submódulo, momento para el que es necesario definir los criterios de Evaluación mediante una lista de cotejo, rúbrica etc., que permitan recuperar conocimientos, habilidades, desempeños y actitudes que evidencien que el alumno arribó a la competencia profesional y los aprendizajes esperados.

El producto integrador que se realiza al término de cada submódulo de aprendizaje es la base para la concreción de las competencias profesionales y genéricas, éste se determina y se registra en el momento del cierre de la secuencia didáctica. Para la Evaluación se propone recuperar evidencias de conocimiento, producto y desempeño, esto con la finalidad de que sean abordados con base en criterios e indicadores de Evaluación que sean la directriz para el diseño de instrumentos pertinentes para evaluar de manera objetiva, válida y confiable el desarrollo de las competencias y la apropiación del conocimiento.

El Modelo Educativo para la Educación Obligatoria (SEP 2017) señala que la Evaluación es un proceso que tiene como objetivo mejorar el desempeño del alumnado e identificar sus áreas de oportunidad. Además, es un factor que impulsa la transformación de la práctica pedagógica y el seguimiento de los aprendizajes.

Para que la Evaluación sea un proceso transparente y participativo donde se involucre al personal Docente y al estudiantado, debe favorecerse:

- **La autoEvaluación**: En ésta el bachiller valora sus capacidades con base a criterios y aspectos definidos con claridad por el personal Docente, el cual debe motivarle a buscar que tome conciencia de sus propios logros, errores y aspectos a mejorar durante su aprendizaje.
- **La coEvaluación**: A través de la cual las personas pertenecientes al grupo valoran, evalúan y retroalimentan a un integrante en particular respecto a la presentación y evidencias de aprendizaje, con base en criterios consensuados e indicadores previamente establecidos
- • **La heteroEvaluación**: A cuál consiste en un juicio emitido por el personal Docente sobre las

características del aprendizaje del estudiantado, señalando las fortalezas y aspectos a mejorar, teniendo como base los aprendizajes logrados y evidencias específicas.

Para evaluar por competencias, se debe favorecer el proceso de formación a través de:
La Evaluación Diagnóstica: Se realiza antes de algún proceso educativo (curso, secuencia o segmento de enseñanza) para estimar los conocimientos previos del estudiantado, identificar sus capacidades cognitivas con relación al programa de estudios y apoya al personal Docente en la toma de decisiones para el trabajo en el aula.

- La Evaluación Formativa: Se lleva a cabo durante el proceso educativo y permite precisar los avances logrados en el desarrollo de competencias por cada estudiante y advierte las dificultades que encuentra durante el aprendizaje. Tiene por objeto mejorar, corregir o reajustar su avance y se fundamenta, en parte, en la autoEvaluación.
- La Evaluación Sumativa: Se realiza al final de un proceso o ciclo educativo considerando el conjunto de diversas evidencias que surgen de los aprendizajes logrados.

Instrumentos de Evaluación

Con el fin de que estudiantado muestre el saber hacer que subyace en una competencia, los aprendizajes esperados permiten establecer una estrategia de Evaluación, por lo tanto, contienen elementos observables que deben ser considerados en la Evaluación tales como:

- ✓ La participación (discurso y comunicación, compromiso, empeño e iniciativa, cooperación).
- ✓ Las actividades generativas (trabajo de campo, proyectos, solución de casos y problemas, composición de textos, arte y dramatizaciones).
- ✓ Las actividades de análisis (comprensión e integración de conceptos como interpretación, síntesis y clasificación, toma de decisiones, juicio y Evaluación, creación e invención y pensamiento crítico e indagación).

Para ello se consideran instrumentos que pueden agruparse principalmente en (Díaz Barriga, 2014):

- ➢ **Rúbricas:** Son guías que describen las características específicas de lo que se pretende evaluar (productos, tareas, proyectos, exposiciones, entre otras) precisando los niveles de rendimiento que permiten evidenciar los aprendizajes logrados de cada estudiante, valorar su ejecución y facilitar la realimentación.
- ➢ **Portafolios:** Permiten mostrar el crecimiento gradual y los aprendizajes logrados con relación al programa de estudios, centrándose en la calidad o nivel de competencia alcanzado y no en una mera colección al azar de trabajos sin relación. Estos establecen criterios y estándares para elaborar diversos instrumentos para la Evaluación del aprendizaje ponderando aspectos cualitativos de lo cuantitativo.
- ➢ **Lista de cotejo.** Es considerada un instrumento de observación y verificación porque permite la revisión de ciertos indicadores durante el proceso de aprendizaje, su nivel de logro o la ausencia de este.

Los trabajos que se pueden integrar en un portafolio y que pueden ser evaluados a través de rúbricas son: ensayos, videos, series de problemas resueltos, trabajos artísticos, trabajos colectivos, comentarios a lecturas realizadas, autorreflexiones, reportes de laboratorio, hojas de trabajo, guiones, entre otros, los cuales deben responder a una lógica de planeación o proyecto. Con base en lo anterior, los programas de estudio de la Dirección General del bachillerato al incluir elementos que enriquecen la labor formativa tales como la transversalidad, las habilidades socioemocionales y la interdisciplinariedad trabajadas de manera colegiada y permanente en el aula, consideran a la Evaluación formativa como eje central al promover una reflexión sobre el progreso del desarrollo de competencias del estudiantado. Para ello, es necesario que el personal Docente brinde un acompañamiento continuo con el propósito de mejorar, corregir o reajustar el logro del desempeño del bachiller, sin esperar la conclusión del semestre para presentar una Evaluación final.

Simbología

Icono	Descripción
	**Lectura**: en esta sección se encuentra una lectura acorde con el plan de estudio que facilita los conocimientos necesarios para el cumplimiento de las actividades programadas
	**Material audio visual**: integrado por enlaces tanto QR como URL para facilitar el acceso a información relevante que complemente una actividad
	**Práctica plan A**: prácticas guiadas que coadyuven a la aplicación de los saberes obtenidos, se acompaña con el instrumento de Evaluación correspondiente.
	**Actividad plan B**: Actividades opcionales para planteles que no les es posible llevar a cabo una práctica, retomando en ella los elementos necesarios de cada lectura.
	**Docente explica**: se sugiere que esta sección el Docente profundice los conocimientos para adecuarlos a su contexto.
	**Actividad para SIGA:** actividad requerida como evidencia para subir a la plataforma SIGA.

Temario

SUBMÓDULO 1 SISTEMAS DE INFORMACIÓN

- Fundamentos de sistemas:
  - Metodología para el desarrollo de software
  - Definición de necesidades
  - Análisis
  - Diseño
  - Codificación
  - Pruebas
  - Validación
  - Mantenimiento y Evaluación
  - Bases de datos
  - Tablas y relaciones
  - Consultas

- Formularios e informes

SUBMÓDULO 2 Programación

- Lógica de programación
  - Algoritmos
  - Diagramas de flujo
  - Pseudocódigo
  - Decisiones
  - Ciclos

- Lenguajes de programación
  - Tipos de lenguajes
  - Metodología de programación
    - Estructurado
    - Orientado a objetos.

- Programación utilizando un lenguaje de alto nivel
  - Entorno de desarrollo
  - Variables
  - Operadores

- Constantes
- Palabras reservadas
- Sentencias de decisión
- Estructuras
- Condición
- Repetición
- Listas

SUBMÓDULO I

SISTEMAS DE INFORMACIÓN

Propósito del Submódulo

Plantea soluciones críticas y responsables mediante la metodología de desarrollo de software para demostrar eficiencia en el manejo de base de datos y software de programación de alto nivel, que sean aplicables a necesidades de una empresa o institución para el tratamiento de información.

Aprendizajes Esperados

- Utiliza la metodología para el desarrollo de software, favoreciendo el trabajo colaborativo y creativo en la resolución de problemas de su contexto.
- Emplea los diferentes modelos de bases de datos, mostrando disposición al trabajo metódico y organizado, para resolver problemas de su contexto.

Competencias

Genéricas	Profesionales
• CG5.2 Ordena información de acuerdo con categorías, jerarquías y relaciones. • CG5.6 Utiliza las tecnologías de la información y comunicación para procesar e interpretar información • CG8.1 Propone maneras de solucionar un problema o desarrollar un proyecto en equipo, definiendo un curso de acción con pasos específicos.	CPBTIC5 Propone el diseño de sistemas de información, a partir del análisis de las necesidades de los usuarios, permitiendo la solución de problemas de manera responsable e innovadora en diferentes contextos.

DOSIFICACIÓN PROGRAMÁTICA
Capacitación: Tecnologías de la Información y la Comunicación
Modulo III: Desarrollo de sistemas
Submódulo I: Sistemas de Información **Clave:** B5SI
Semestre: 5to. Periodo: 2023 – 2024A Turno: Matutino/ Vespertino

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
Submódulo 1. Sistemas de Información	Inicio	350	Encuadre del módulo Evaluación diagnóstica Lectura 1 Fundamentos de sistemas: • Metodología para el desarrollo de software • Definición de necesidades • Análisis.	1	21 al 25 de agosto	
	Desarrollo	350	Lectura 2. identificación de necesidades • Definición de necesidades Lectura 3. Análisis del Sistema “Modelando mis Datos” • Análisis **Construye t: Lección 01. ¿Qué voy a ver en este curso?**	2	28 de agosto al 1 de septiembre	
	Desarrollo	450	Lectura 4. Diseño del Sistema “Diseñando pantallas” • Diseño Lectura 5. Codificación “Programando Ando” • Codificación	3	4 al 8 de septiembre	
		250	Lectura 6. Validación y pruebas de SI “Validar para no Errar”	4	11 al 15 de septiembre	

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
			• Pruebas, • Validación Mantenimiento y Evaluación.			
		350	Lectura 7. Base de datos: Tablas y relaciones en Access Bases de datos • Tablas y relaciones	5	18 al 22 de septiembre	
	Cierre	400	Lectura 8. Consulta en Access “Consulta la Info” • Consultas Lectura 9. Formularios e Informes en Access “Otra forma de ver los datos” • Formularios e informes	6	25 al 29 de septiembre	
		350	Situación didáctica	7	2 al 6 de octubre	

Encuadre Submódulo 1. Sistemas de Información

Sistemas de Información		
Elemento para evaluar		**Puntaje**
Actividades	Actividad 1. Crucigrama “Mi Método”	3.64
	Actividad 2 Solicitud de Servicio de Sistemas	3.64
	Actividad 3 Identificación de necesidades	3.64
	Actividad 4. Análisis del Sistema “Modelando mis Datos”	3.64
	Actividad 5. “Diseñando mi interfaz”	3.64
	Actividad 6. Rally de Codificación	3.64
	Actividad 6.1. “Ahora te toca a ti”	3.64
	Actividad 7. “Cuadro sinóptico “Validación y pruebas de un SI””	3.64
	Actividad 8. Mapa conceptual, “todo cabe en una tabla”	3.64
	Actividad 9. “Herramientas de consulta en Access”	3.64
	Actividad 10. Tabla “Encontrando sus diferencias”	3.64
	Total	**40**
Prácticas o Evaluación	Práctica 1. Consultas en Access “qué quieres buscar”	15
	Práctica 2. Formularios e Informes en Access “Otra forma de ver los datos”	15
	Total	**30**
SD1	**Exposición “Mi tiendita”**	**20**
	CONSTRUYE-T Lección 01. ¿Qué voy a ver en este curso?	**10**
	Total	**100**

Situación Didáctica	
Título:	"Organizando y Operando mi Información"
Contexto:	Actualmente, utilizando la tecnología y el desarrollo de sistemas de información las personas y empresas tienen la facilidad de realizar búsquedas y consultas de información que les sea conveniente para satisfacer sus necesidades. En la comunidad se encuentran diversos negocios que aun manejan su información de manera manual, teniendo como resultado pérdida de tiempo, pero sobre todo perdida en cuestión económica es por esto por lo que uno de los pequeños empresarios se acercó a los alumnos de quinto semestre para que los apoyen a llevar su control de inventario de su mercancía, con esto facilite el desplazamiento de esta, la mejor organización de su negocio y con esto mejorar la toma de decisiones operativas. Teniendo en cuenta las necesidades de las microempresas locales, los estudiantes del programa de TIC en el submódulo de sistemas de información desarrollarán un sistema utilizando el método de desarrollo de Software abordando una situación real.
Propósito	Elaborar en equipos de 5 integrantes a través de la técnica de Aprendizaje Basado en Problemas (ABP) un ***Sistema de gestión informático (Base de datos)*** tomando en cuenta las necesidades de las microempresas de su localidad, crear una base de datos que apoye a una empresa local, para la toma de decisiones, usando informes y formulario. .
Conflicto cognitivo	¿En qué ayuda un sistema de información? ¿Por qué usar una metodología para el diseño de sistemas de información? ¿Cuál es el impacto de tener el conocimiento de la información de manera clara y en tiempo?
Producto esperado	Sistema de gestión informático "Mi Tiendita"

¿Cómo resuelvo la Situación Didáctica?

"Mi tiendita"

Propósito: Elaborar en equipos de 5 integrantes a través de la técnica de Aprendizaje Basado en Problemas (ABP) un ***Sistema de gestión informático (Base de datos)*** tomando en cuenta las necesidades de las microempresas de su localidad aplicando la metodología del desarrollo de software para la creación de una base de datos en Microsoft Access, considerando el instrumento de Evaluación propuesto y socializarlo ante el grupo.

Instrumento de Evaluación
LISTA DE COTEJO
Situación Didáctica "Organizando y operando mi Información"

DATOS GENERALES

Nombre(s) del alumno(s)	**Matrícula(s):**
Producto: Sistema de gestión Informático "Mi tiendita"	**Fecha:**
Materia:	**Periodo:**
Nombre del Docente:	**Firma del Docente:**

VALOR DEL REACTIVO	CARACTERISTICAS A CUMPLIR	VALOR OBTENIDO		CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SI	NO		
1	Realiza la solicitud de servicio de acuerdo con el problema a resolver del Sistema de gestión Informático.				
1	Presenta el modelo de contexto y el modelo Entidad-Relación del Sistema de gestión Informático.				
1	Presenta el modelo Entidad-Relación del Sistema de gestión Informático				
1	Establece las relaciones y cardinalidad entre las tablas de la base de datos.				
1	Crea la base de datos y diseña de manera correcta las tablas, en la aplicación de Access.				
2	Crea al menos 2 consultas de cada una de las tablas de la base de datos.				
1	Crea 1 formulario y 1 informe de cada una de las tablas de la base de datos.				
1	Entrega en tiempo y forma.				
1	Participan todos los integrantes del equipo				
10	**CALIFICACIÓN**				

Evaluación Diagnóstica

Submódulo I Sistemas de Información.

NOMBRE: _____ **GRUPO:** _____

1. Conjunto de elementos que interactúan entre si con el fin de apoyar las actividades de una institución (pública-privada) empresa o negocio. ( )
   A) Sistema informático
   B) Sistema
   C) Sistema de aplicación
   D) Sistema de información

2. Actividades básicas de un sistema de información. ( )
   A) Entrada, proceso y salida de información de datos.
   B) Entrada, almacenamiento, procesamiento y salida de información.
   C) Entrada y salida de información de datos.
   D) Entrada y proceso de información de datos.

3. Son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado. ( )
   A) Metodología para el desarrollo de Software
   B) Metodología de sistemas
   C) Desarrollo de proyectos
   D) Metodología de aplicaciones

4. Son los pasos de Metodología de desarrollo de software. ( )
   A) Requisitos, diseño, implementación, verificación y mantenimiento.
   B) Planeación, diseño, codificación prueba
   C) Definición de necesidades, análisis, diseño, codificación, validación y Evaluación.
   D) Plan rápido, Modelado, Construcción, Desarrollo y comunicación

5. Son herramientas estructuradas de análisis y diseño, las cuales permiten al analista comprender el sistema y los subsistemas en forma visual, como un conjunto de flujos de datos interrelacionados. ( )
   A) Algoritmos
   B) Diagrama de Flujo de Datos.
   C) Pseudocódigos.
   D) Programación

6. En distintas metodologías esta fase podría ser llamada fase de definición ofase de diseño lógico. ( )

A) Definición de problemas y objetivos de mejora del sistema.
C) La fase de análisis de requerimientos.
B) Identificación de problemas.
D) Identificación de necesidades

7. Es un diagrama que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. ( )

A) El modelo entidad relación
C) Diseño físico.
B) Diagrama de casos de uso
D) Diagrama de contexto.

8. Es el proceso de definición de la arquitectura, módulos, interfaces y datos de un sistema para la especificación de una solución detallada basada en la computadora. ( )

A) Desarrollo
C)Diseño
B) Codificación
D)Sistema de diseño

9. Es un conjunto de datos interrelacionados, almacenados en su conjunto, sin redundancias perjudiciales o innecesarios, que sirven a una aplicación o más para llevar obtener información. ( )

A) Archivos
C) Base de datos
B) Tablas
D) Consultas

10. Es el software que nos sirve para crear y manipular la base de datos. ( )

A) Administrador de archivos
C) Programa de aplicación
B) Sistema manejador de base de datos
D) Panel de control

11. Es la persona que tiene el control central sobre el sistema de base de datos. ( )

A) Administrador de la Base de datos
C) Usuario final
D) Navegante
B) Programador de aplicaciones

12. Es un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla de base de datos. ( )

A) Identificador
C) Clave primaria o clave principal
B) Clave secundaria
D) Encabezado

Lectura 1. Fundamentos de sistemas: "Metodología para el desarrollo de software"

Instrucciones: Realiza el análisis de la siguiente lectura posteriormente realiza la Actividad 1. Crucigrama "Mi método"

Fundamentos de sistemas

En la vida diaria estamos rodeados de sistemas como el sistema solar, digestivo, el sistema de transporte público, el sistema escolar entre otros; cada uno de estos sin importar su naturaleza (naturales o creados por el hombre) cumplen con una función específica. La principal característica de estos elementos es que interactúan y su objetivo es que dicha interacción hace que este complejo de elementos se comporte como un todo. Arnold, M. & Osorio, F., (1998) concibieron la definición de sistemas como "Un conjunto de elementos que guardan estrechas relaciones entre sí, que mantienen al sistema directo o indirectamente unido de modo más o menos estable y cuyo comportamiento global persigue, normalmente, algún tipo de objetivo".

Un sistema de información es un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización, para elevar el nivel de conocimientos que permitan un mejor apoyo a la toma de decisiones y desarrollo de acciones. (Peña, 2006).

Otros autores como Peralta (2008), de una manera más acertada define sistema de información como: conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Teniendo muy en cuenta el equipo computacional necesario para que el sistema de información pueda operar y el recurso humano que interactúa con el Sistema de Información, el cual está formado por las personas que utilizan el sistema.

Por lo tanto, los Sistemas de Información (SI) ayudan a ***recopilar, distribuir, procesar, datos de una***

organización para la mejora de procesos y toma de decisiones. Mientras que ***las Tecnologías de la***

Información (TI) soporta las funciones de un SI, es de decir el conjunto de procedimientos, almacenamiento y transmisión de la información derivados de las herramientas (Hardware, software, periféricos y redes).

Figura 1. Moo Salvador. (2023, 24 abril). Funciones del sistema de información (SI).

Funciones básicas de un SI

- **Entrada de datos (Datos e información):** En este proceso el usuario ingresa información al sistema mediante la interfaz de entrada.

- **Proceso:** Al ingresar los datos, el sistema los transforma, convierten y analizan mediante procesos adecuados de almacenamiento, análisis, y transferencia a un dispositivo de salida.

- **Almacenamiento:** Los datos ingresados se almacenan en la memoria central o en dispositivos de almacenamiento internos o externos de manera que pueden ser usados en

diversas operaciones desde cualquier lugar, en diferentes momentos, por lo que la capacidad de almacenamiento es vital.

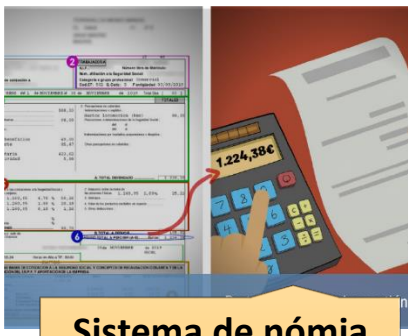
- **Salida de datos (Reportes o informes):** Es la capacidad de un SI para mostrar información procesada o datos de entrada al exterior, los SI cuentan con diversas interfaces de salida.
- **Retroalimentación:** Se produce cuando las salidas o la influencia de las salidas de los sistemas en el contexto, vuelven a ingresar al sistema como recursos o información.

Tipos de SI

Para que un sistema de información funcione adecuadamente se debe tomar en cuenta su diseño, facilidad de uso, flexibilidad, mantenimiento automático de los registros, apoyo en toma de decisiones críticas y mantener el anonimato en informaciones no relevantes. Los SI principalmente pueden ser:

- **Manuales:** Donde todos los procesos, actividades, reportes se realizan a mano.
- **Automatizados**: Se transfieren las tareas realizadas por el individuo a un conjunto de elementos tecnológicos.

Anteriormente los sistemas eran manuales, por ejemplo, el sistema de pago de nómina de una empresa dónde se utilizaban largas hojas tabulares de registro donde tenían todos los datos de los empleados y en cada columna se registraban los movimientos de la semana o quincena, haciendo dobleces para tener todas las cifras y calcular todas las percepciones, impuesto, deducciones y el sueldo neto. Actualmente, un sistema computarizado de nómina hace uso de diferentes archivos relacionados en una base de datos. Algunos de estos archivos se van formando con los movimientos del periodo;



Sistema de nómia manual

SI automático

otros permanecen sin cambio, como las tablas del impuesto, sueldos, etcétera, y el proceso se hace muy dinámico y sin posibilidad de error.

De manera cotidiana todos interactuamos con SI, para fines tanto personales como profesionales; utilizamos cajeros automáticos, también para controlar la información de los pacientes de un hospital, en las tiendas comerciales registrando nuestras compras por medio de códigos de barras, en los bancos para tener el control y seguimiento de las operaciones registradas por los clientes, en los aeropuertos, con un sinfín de herramientas que hacen más fácil los diferentes procesos.

Objetivos principales de los SI

- Automatizar procesos.
- Recolectar y distribuir la información exacta y confiable.
- Proporcionar información que sirva de apoyo al proceso de toma de decisiones.
- Almacenamiento adecuado de tal forma que esté disponible cuando se necesite.
- Lograr ventajas competitivas a través de su implantación.

Elementos de un SI

Los SI están constituidos por diversos elementos, los cuales tienen una función específica que, sin alguno de estos, el sistema no podría ser funcional (García, B. 2000, citado en Lapiedra, R, Devece, C., 2019).

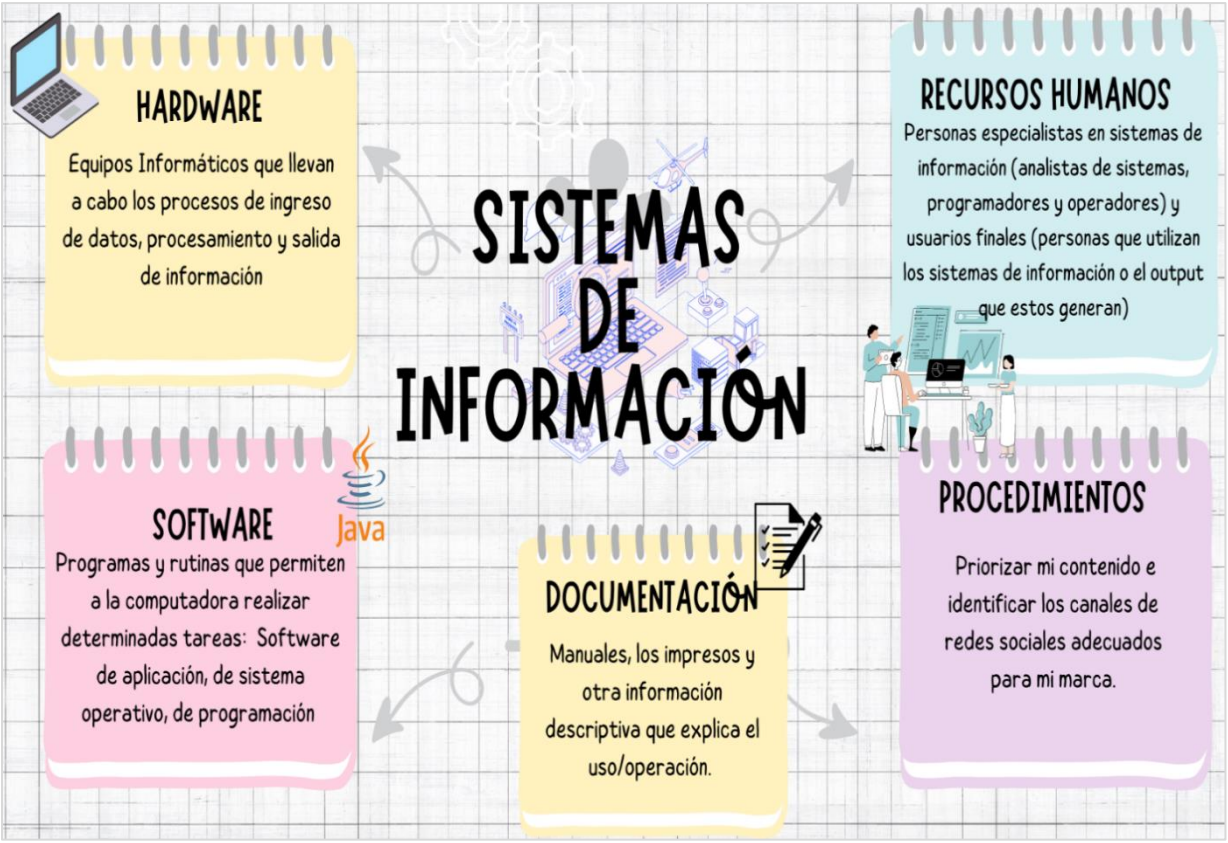


Figura 2. Moo Salvador. (2023, 24 abril). Elementos de un sistema de información (SI).

Metodología para el desarrollo de software

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo (Maida, E., & Pacienzia, J., 2015, p.12). Cada metodología de desarrollo de software tiene su propio enfoque, ahora bien; para explicar la metodología de desarrollo de software utilizaremos el método del ciclo de vida para el desarrollo de sistemas.

Por su parte, el método del ciclo de vida para desarrollo de sistemas (SDLC) es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información.

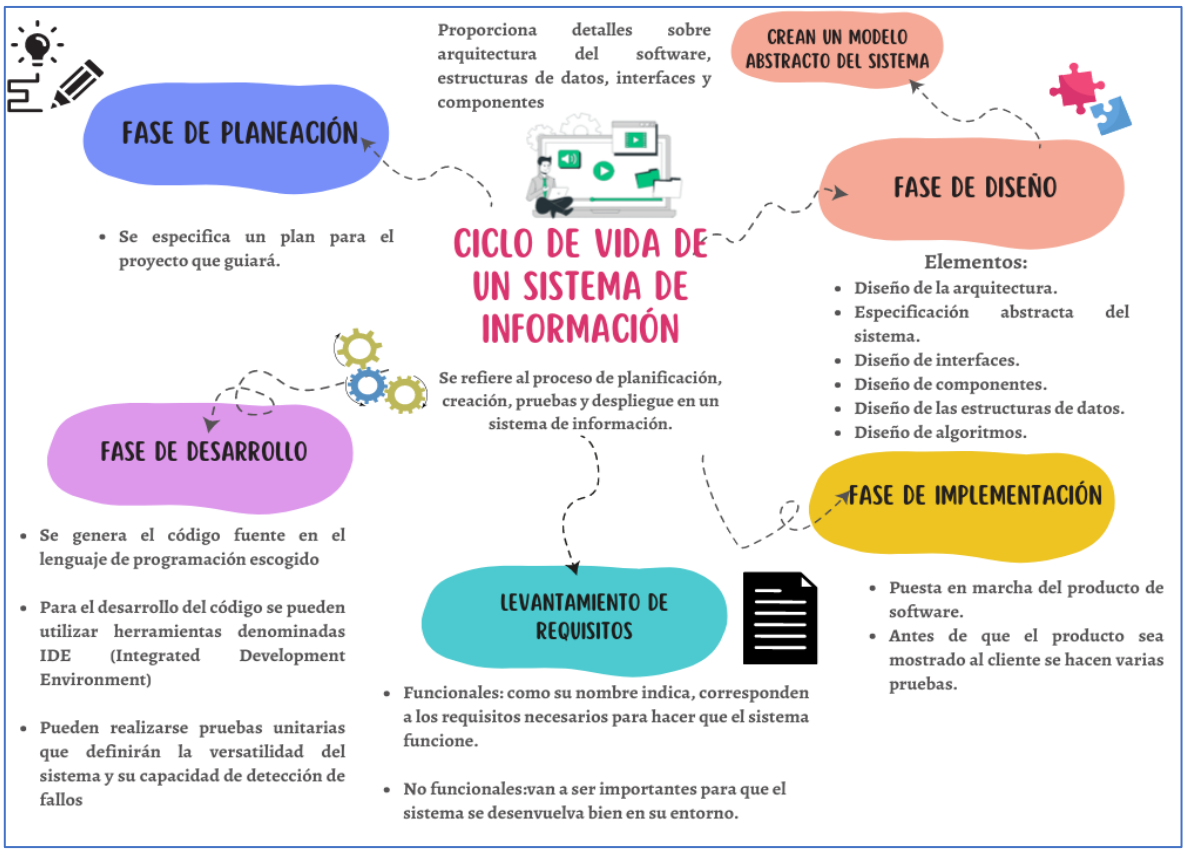


Figura 3. Moo Salvador. (2023, 24 abril). Pasos del método del ciclo de vida para el desarrollo de sistemas

1.-Investigacion preliminar (Definición de necesidades)

También llamado fase de investigación preliminar, de estudio inicial o de planeación, es la primera etapa del proceso de desarrollo de sistemas clásico que inicia cuando se realiza una solicitud para recibir ayuda de un sistema de información. Responde la pregunta, "¿vale la pena atender este problema?", en esta fase intervienen los propietarios del sistema, administradores del proyecto y analistas.

En la investigación preliminar se evalúan las solicitudes de los proyectos, para emitir un juicio sobre su factibilidad. Por lo cual se debe: determinar el tamaño del proyecto, evaluar costos y beneficios, determinar la factibilidad técnica, operacional y financiera a través de revisión de documentos o entrevistas al personal de la organización.

2.- Determinación de los requerimientos del sistema (Análisis). Consiste en comprender todos los procesos de la empresa, los analistas al trabajar con los empleados y administradores deben estudiar los procesos de una empresa y definir los requerimientos de negocio para un sistema nuevo, a la vez deben contestar algunas de las siguientes preguntas:

- ¿Qué necesitan y desean los usuarios de un sistema nuevo?
- ¿Cuáles son los límites impuestos por el tiempo y carga de trabajo?
- ¿Qué controles de desempeño utiliza?

3.-Diseño del sistema

El diseño de un SI produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. El diseño también indica los datos de entrada, aquellos que serán calculados ylos que deben ser almacenados; por su parte, los diseñadores son los responsables de dar a los programadores las especificaciones de software completa.

4.-Desarrollo del software (codificación)en esta etapa se elige el lenguaje de programación adecuado al sistema y se desarrollan programas ejecutables funcionales, y listos para implementarse.

Los encargados de desarrollar software pueden instalar, (o modificar y después instalar) software comprado a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para el software y la disponibilidad de los programadores. Principalmente se deben diseñar procesos para la captura de datos, una interfaz adecuada, diseño de archivos o bases de datos, procedimiento de control y respaldo.

5.-Prueba de los sistemas

Durante la fase de prueba de sistemas, se emplea de manera experimental para asegurarse de que el software no tenga fallas, en algunas organizaciones las pruebas son conducidas por personas ajenas al grupo que escribió los programas originales, con esto se busca asegurar que el software sea confiable.

6.-Implantación (validación). La implantación es el proceso de verificar e instalar nuevo equipo,

entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla.

7.-Evaluación. La Evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes. La Evaluación ocurre a lo largo de las siguientes dimensiones:

Figura 4. Metodologías para el desarrollo de software actuales

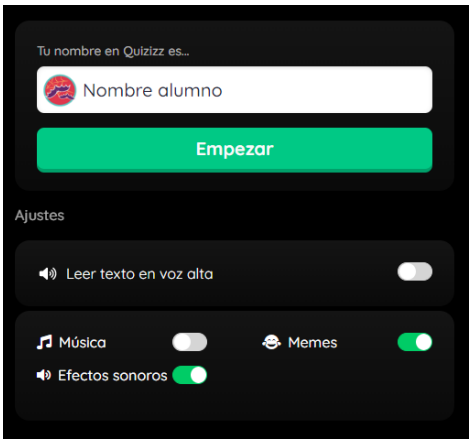
Recurso didáctico sugerido

https://www.youtube.com/watch?v=HhC75IonpOU

Actividad 1. Crucigrama "Mi Método"

Instrucciones: Tomando en cuenta los conceptos aprendidos en la lectura 1. Fundamentos de sistemas: "Metodología para el desarrollo de software", elige una forma para realizar la actividad de acuerdo con las opciones del inciso A (Contestando el Quizz, tomando foto de la actividad como evidencia de su solución.) o B (Respondiendo el crucigrama de la guía impresa).

Opción A. Ingresa a la siguiente página y responde el cuestionario:



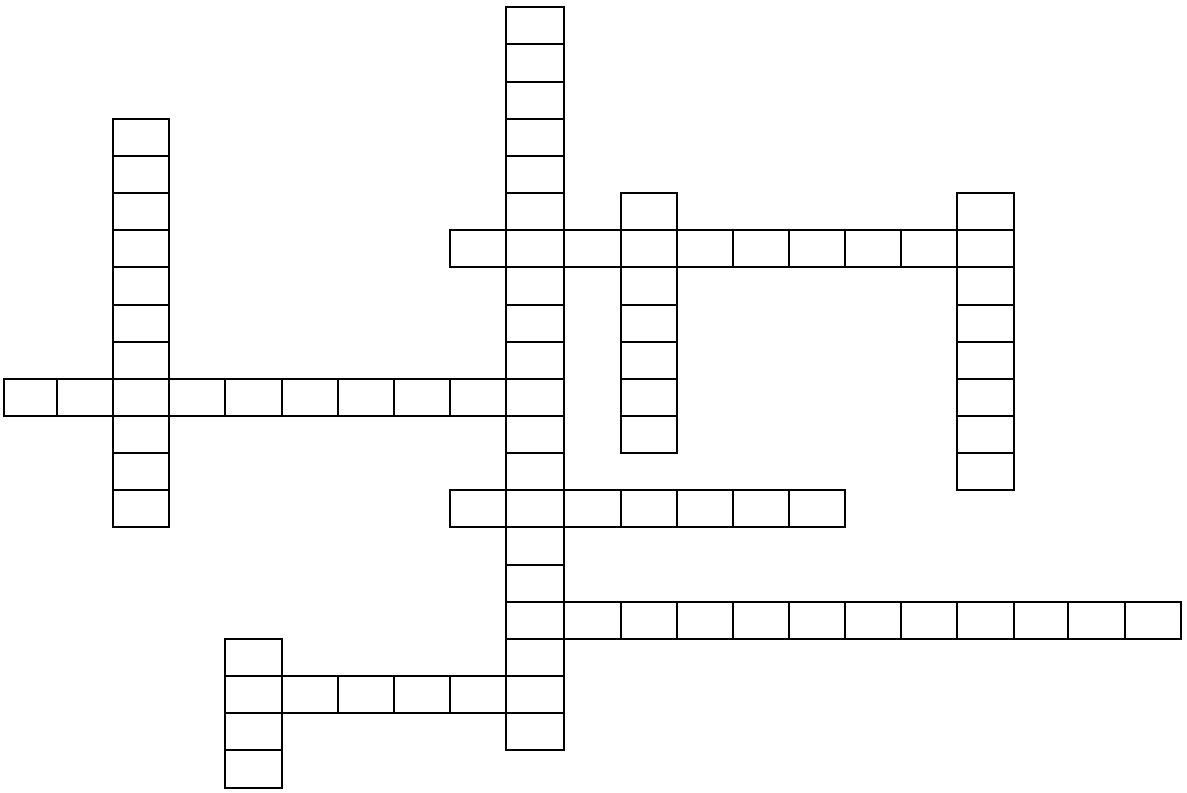
Opción B. Responde las siguientes preguntas para posteriormente dar respuesta al crucigrama

Columnas

1. Partes y objetos que interactúan y que forman un todo que se encuentra bajo influencia de fuerza en alguna relación definida _____.
2. Ayudan a recopilar, distribuir, procesar, datos de una organización para la mejora de procesos y toma de decisiones _____.
3. Colección de datos interrelacionados y organizada para que se pueda acceder a ellos por sus atributos _____.
4. Siglas que describen el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información _____.
5. Consiste en comprender todos los procesos de la empresa, respondiendo que "¿Qué necesitan y desean los usuarios de un sistema nuevo? _____.

Filas

6. En esta etapa se elige el lenguaje de programación adecuado al sistema y se desarrollan programas ejecutables funcionales _____.
7. Indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados _____.
8. El sistema se emplea de manera experimental para asegurarse de que el software no tenga fallas _____.
9. Permite verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación _____.
10. Permite identificar puntos débiles y fuertes de un sistema _____.



Referencias

Arnold, M., & Osorio, F. (1998). Introducción a los Conceptos Básicos de la Teoría General de Sistemas. Cinta de Moebio.

Andreu, R., J. Ricart & J. Valor (1996). Estrategia y sistemas de información. Bertoglio, J. (1993). Introducción a la Teoría General de Sistemas.

Escuela, de N. F. (s/f). Escuela de Negocios FEDA. Escueladenegociosfeda.com. Recuperado el 25 de abril de 2023, de https://www.escueladenegociosfeda.com/blog/50-la-huella-de-nuestros-Docentes/471-gestion-agil-vs-gestion-tradicional-de-proyectos-como-elegir

Gonzalez-Longatt, F. (2007). Introducción a los sistemas de información: fundamentos. Aragua, Venezuela.

Kuz, A., Falco, M., & Giandini, R. S. (2018). Comprendiendo la Aplicabilidad de Scrum en el Aula: Herramientas y Ejemplos. Revista iberoamericana de tecnología en educación y educación en tecnología, 21, e07. https://doi.org/10.24215/18509959.21.e07

Marín, M. (s.f). Sistemas de Información. Unidad de Apoyo para el Aprendizaje. http://132.248.48.64/repositorio/moodle/pluginfile.php/1415/mod_resource/content/1/contenid o/index.html.

Maida, E., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Senn James A. (2001) Análisis y diseño de sistemas de información. Segunda edición. ISBN:978968422914.

Sistemas de Información. (s/f). Com.ar. Recuperado el 25 de abril de 2023, de

https://www.econlink.com.ar/sistemas-informacion/definicion

Whitten Jefrey L. (2008). Análisis de sistemas, diseños y métodos. ISBN:9789701066140.

Santander Becas, (21/122020) | Santander Universidades. https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html

(S/f). Researchgate.net. Recuperado el 15 de abril de 2023, de https://www.researchgate.net/figure/Traditional-methodologies-vs-Agile-methodologies_tbl1_273302003

(S/f-b). Recuperado el 15 de abril de 2023, de http://chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://tesis.uson.mx/digital/tesis/docs/2034/Capitulo3.pdf

Lectura 2. Identificación de necesidades

Instrucciones: *Realiza el análisis de la siguiente lectura, posteriormente realiza la Actividad 2 "Solicitud de servicios de sistemas" y la Actividad 3 "Esquema: Identificación de necesidades".*

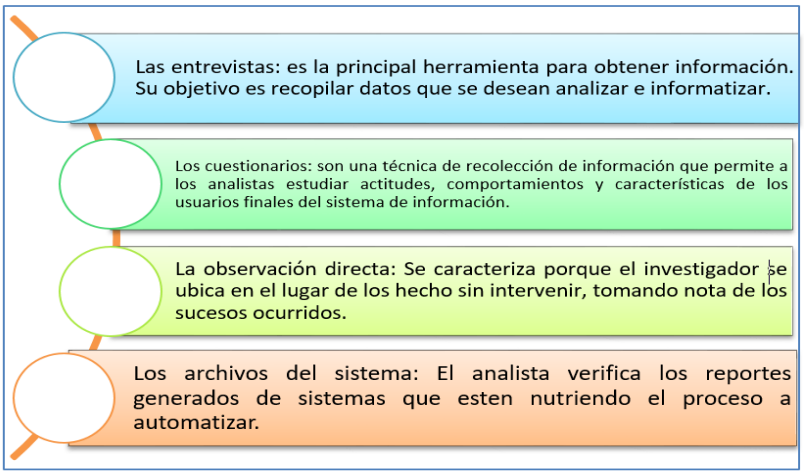
En esta etapa se hace un examen exhaustivo de las necesidades de la organización que va a emplear el sistema, ¿Qué necesita?, ¿Cómo lo necesita?, son preguntas que regularmente se realiza en esta etapa, ya que en este apartado es en donde se definen el ámbito y alcance del sistema. Para contestar estas interrogantes intervienen los propietarios del sistema, administradores del proyecto, usuarios y analistas del sistema con el propósito de identificar los problemas que se pretende resolver con el sistema de información (Kendall y Kendall, 2011).

Es muy importante la etapa de identificación de necesidades, ya que de ella depende realizar un sistema de información efectivo para los usuarios, en caso de que este análisis no se haga cubriendo las necesidades, es muy probable que se tengan que realizar ajustes al mismo con el fin de corregir fallos o agregar requerimientos. Las necesidades del sistema a estudio se obtendrán principalmente de los usuarios de este (Narváez, 2021).

La figura muestra las diversas técnicas de recopilación de la información:

Figura

Técnicas de recopilación de información



Fuente: Paredes, E. y Velasco, M.E. (s.f.). Análisis y Diseño Sistema información.

Según Tantani (2020), esta fase incluye las siguientes tareas:

1. Identificar problemas

En esta fase el analista se encarga de identificar correctamente los problemas, las oportunidades y los objetivos, en este caso, se realiza una solicitud de proyecto donde se solicita el desarrollo de sistema.

Por este motivo, el analista se reúne con el cliente e identifican las metas globales, las necesidades y requerimientos del sistema, así como otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

2. Solicitud de servicios de sistemas

La solicitud de servicios de sistemas es un documento que sirve a los analistas para recopilar información que permita identificar:

- Empresa o institución a la que va dirigido el sistema de información.
- El tipo de servicios solicitado por el cliente.
- Identificación de problemas y soluciones que se proponen implementar en el sistema.
- Fecha tentativa de entrega del sistema.

3. Definición de objetivos de mejoras del sistema

En esta fase el analista debe identificar qué trata de hacer la empresa; después debe ser capaz de establecer si alguna de las características del sistema de información, puede ayudar a la empresa a lograr sus objetivos al revolverse problemas u oportunidades específicos.

Ejemplo del Formato de Solicitud de Desarrollo de Sistemas

Fecha: 06 de abril del 2023	
Solicitante:	
Nombre del solicitante:	José Francisco González Morales
Numero de empleado:	39045
Unidad Responsable:	Administración
Cargo que desempeña:	Jefe de Oficina
Correo electrónico:	mitiendita@gmail.com
Teléfono o extensión:	9361198717
*Nombre del responsable ante el departamento de Desarrollo	Ing. Erick Eduardo Antonio Díaz
Especificaciones	
Nombre del proceso a automatizar: Facturación	
Tipo de servicio solicitado:	
Mejora del sistema existente ◯ Desarrollo del nuevo sistema ◯ Otro (especifique): _____	

Explicación breve del proceso a automatizar:

La abarrotera "Mi tiendita" es un lugar en el que se encuentran a la venta artículos comerciales, especialmente comidas, bebidas y conservas.

Para llevar a cabo el proceso de ventas, el administrador de la abarrotera registra manualmente en una libreta los datos de los productos **(Numero, Nombre del producto, Precio del producto, cantidad del producto, fecha compra)**, *clientes* **(Numero, Nombre, Edad)** y facturas **(Numero de la factura, fecha de la venta, nombre del cliente, producto, precio, cantidad, subtotal)** de cada venta.

Al realizar sus procesos de manera manual ocasiona que el registro de las facturas y clientes sea tardado, lo cual genera atrasos en las ventas de los productos, además de que los clientes se quejan del tiempo que esperan para ser atendidos.

Otro inconveniente se presenta al momento de realizar los reportes del día, semana o mes, ya que se tiene que revisar toda la bitácora e ir escribiendo en otra hoja los datos requeridos para el reporte.

Lo anterior origina que no se pueda analizar la información, tanto de ingreso y salida de los productos de la tienda, así como las ventas totales y control de las facturas. Como consecuencia existen perdidas de dinero y productos, causando un mal manejo de la información que no ayuda a la toma de decisiones y buen control de la empresa.

Breve declaración de la solución esperada:	
Elaborar el análisis y diseño de un sistema deinformación para mejorar los procesos de facturación de ventas de la abarrotera "Mi tiendita", el cual permitirá llevar el control de los productos, clientes y facturas de cada venta realizada.	
Documento para anexar a este formato	
Requisitos del sistema /modulo	
fecha tentativa para la liberación del desarrollo del módulo:	

Firmas autorizadas

_____ _____

Analistas del sistema *Encargado del departamento*

Definición de problemas y objetivos de mejoras del sistema

Seguidamente **del formato solicitud de servicios** de sistemas, se realiza la **definición de problemas y objetivos** de mejoras del sistema, este documento permite realizar una comprensión del dominio del problema, a la vez identificar las posibles mejoras del sistema que se expresan a través de definiciones precisas que definen las expectativas del nuevo sistema.

Elementos que integran el documento ***"Definición de problemas y objetivos de mejoras delsistema"***

Definición breve del problema	Urgencia	Visibilidad	Prioridad o Lugar	Objetivo delsistema
¿Cuáles son los problemas?	¿En qué tiempo (días, meses) debe ser resuelto el problema?	¿A qué grado una solución o un sistema nuevo deben ser visible para los clientes o la administración ejecutiva?	¿Cuáles son las prioridades acordadas entre todos para cada problema?	¿Qué mejoras pueden definir las expectativas del nuevo sistema?

En la siguiente sección, se muestra un ejemplo del documento definición de problemas y objetivos de mejoras del sistema para la abarrotera “Mi Tiendita”:

No	*Definición breve del problema*	*Urgencia*	*Visibilidad*	*Prioridad lugar*	*Objetivo del sistema*
1	Desconocimiento de los precios de los productos al momento de atender a los clientes, lo cual ocasiona mal cálculo de la venta y mala atención al cliente.	*10min*	*Alta*	*1*	El sistema a través de una clave del producto permitirá obtener precios para agilizar el cálculo de la venta.
2	Mucho tiempo invertido en la facturación de las ventas.	10 minutos	Alta	1	El sistema permitirá generar la factura de las ventas efectuadas por los clientes.
3	Alto tiempo en la elaboración de los reportes de las ventas diarias, semanales y mensuales, debido a que la información está escrita, y es necesario revisar hoja por hoja, e ir anotando los datos en otro documento.	Diario, semanal y mensual	Alta	1	El sistema permitirá generar consultas y reportes de las ventas diarios, semanales y mensuales.
4	Alto tiempo para la elaboración del reporte de las facturas solicitadas.	Diario, semanal y mensual	Mediana	2	Contar con informes actualizados de las facturas.
5	Alta demanda de tiempo para generar los reportes de las ganancias obtenidas de la venta de los productos, ya que se requiere ir sumando los montos recibidos por cada cliente al momento de pagar el monto de la venta.	Diario, semanal y mensual.	Alta	1	Evitar pérdida de tiempo recopilando información que está almacenada en base de datos que se puede compartir.

Actividad 2 Solicitud de Servicio de Sistemas

Etapa 1: Solicitud de servicios de sistemas

Instrucciones: Con tu equipo de trabajo colaborativo, realiza la solicitud de servicios de sistemas, de acuerdo con el ejemplo mencionado en la lectura 2 Identificación de necesidades.

	Fecha: .
Solicitante:	
Nombre del solicitante:	
Numero de empleado:	
Unidad Responsable:	
Cargo que desempeña:	
Correo electrónico:	
Teléfono o extensión:	
*Nombre del responsable ante el departamento de Desarrollo	
Especificaciones	
Nombre del proceso a automatizar:	
Tipo de servicio solicitado:	
Mejora del sistema existente ◯ Desarrollo del nuevo sistema ◯ Otro (especifique): _____	
Explicación breve del proceso a automatizar:	
Breve declaración de la solución esperada:	
Documento para anexar a este formato	
Requisitos del sistema /modulo	
Fecha tentativa para la liberación del desarrollo del módulo:	

Firmas autorizadas

_____	_____
Analistas del sistema	*Encargado del departamento*

Sistema:
Analistas del sistema:
Fecha de creación:

Etapa 2: Tabla de definición de problemas y objetivos

Instrucciones: Con la lectura y el ejemplo anterior, de acuerdo con el sistema de Información a realizar de tu comunidad Rellena la tabla "**Definición de problemas y objetivos de mejoras del sistema**".

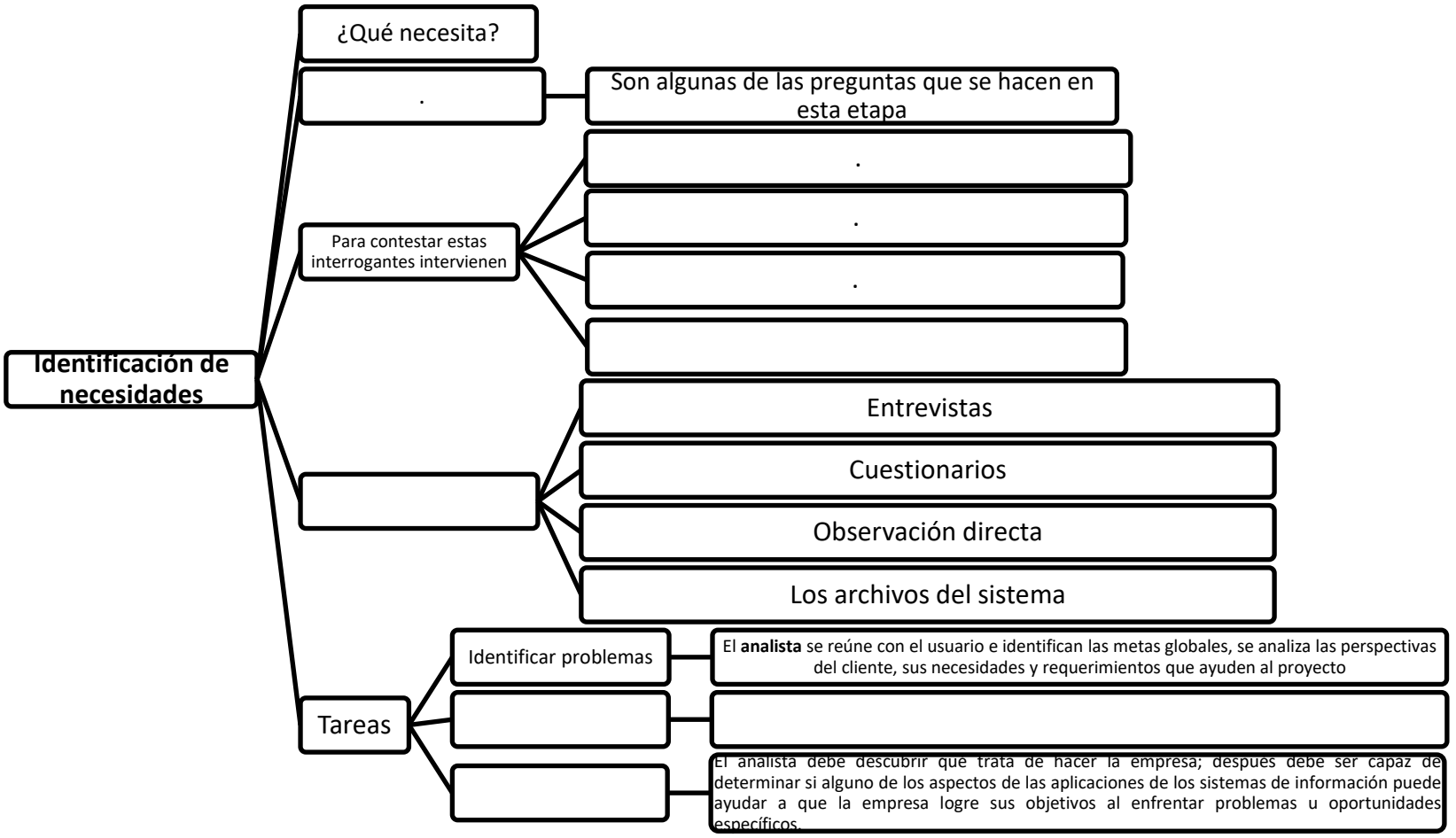
Definición breve del problema	*Urgencia*	*Visibilidad*	*Prioridad / lugar*	*Objetivo del sistema*

NOTA: Estimado Docente no olvides evaluar a tus estudiantes con la lista de cotejo de la página siguiente.

INSTRUMENTO DE EVALUACIÓN LISTA DE COTEJO Práctica 2. Solicitud de servicio de sistemas					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s):**		
Producto: Solicitud de Servicios			**Fecha:**		
Materia:			**Periodo:**		
Nombre del Docente:			**Firma del Docente:**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Identifica correctamente el problema a resolver.				
2	Relaciona la información de forma precisa.				
1	Describe la importancia de clasificar los elementos para resolver el problema				
2	Realiza correctamente la solicitud de servicio de acuerdo con el problema a resolver				
1	Involucra las áreas pertinentes para la complementación del sistema de Información.				
2	Utiliza las tecnologías para procesar e interpretar información y dar solución a la problemática planteada.				
1	Participa y colabora de forma colaborativa, mostrando interés en su realización.				
	CALIFICACIÓN				

Actividad 3: Esquema “Identificación de necesidades”

Instrucciones: Completa el siguiente esquema didáctico relativo a la lectura 2 “Identificación de necesidades”.



Referencias

Kendall, K. E. y Kendall, J. E. (2011). Análisis y diseño de sistemas. ISBN: 978-607-32-0577-1. 8va edición. PEARSON EDUCACIÓN, México

Navaez, B. M. (2021). Definición de necesidades. YouTube. https://www.youtube.com/watch?v=jEGlB4ecM6c&feature=youtu.be

Paredes, E. y Velasco, M.E. (s.f.). Análisis y Diseño Sistema información. Universidad de Pamplona. Facultad de estudios a distancia. https://www.unipamplona.edu.co/unipamplona/portalIG/home_109/recursos/octubre2014/administraciondeempresas/semestre7/11092015/analisisydisenosistinformacion.pdf

Tantani, J. F. (2020). Sistema de solicitud de servicio de la empresa “SERVICE WEB”. YouTube. https://www.youtube.com/watch?v=MUFQwdM43zI&feature=youtu.be

Genérica

1.1 ¿Qué voy a ver en este curso?

"Son nuestras decisiones las que muestran lo que podemos llegar a ser. Mucho más que nuestras propias habilidades".
J. K. Rowling

Soy Lorenzo. En cuatro días es mi examen para entrar a la Universidad. Pero... me invitaron a ir a una fiesta mañana. No sé qué hacer, pues ambas cosas son importantes para mí. No sé si ir a la fiesta o repasar los últimos temas del examen.

¿Te ha pasado algo similar? En este curso te presentaremos un método que te servirá de guía cuando tengas que tomar una decisión importante para que ésta promueva tu bienestar y el de los demás.

El reto es identificar elementos del curso que te ayudarán a tomar decisiones de manera consciente, autónoma responsable y ética ante diversas situaciones de la vida, con el fin de promover el bienestar individual y colectivo.

Actividad 1.

Trabaja de manera individual.
Lee la siguiente historia:

José Luis vive en Huetamo, Michoacán. Estudia la preparatoria y una vez que termine quiere irse a Morelia a estudiar Derecho.

Sin embargo, empezando el tercer año, su tío que vive en Los Ángeles le propone llevárselo para allá a trabajar como mesero en el restaurante de unos conocidos. El sueldo que le ofrece es muy bueno y podría viajar con su tío, quien además está dispuesto a pagar el pasaje. José Luis considera que es una oportunidad única que quizás no se vuelva a presentar. Tiene dos semanas para decidir: ¿se queda o se va?

Si José Luis fuera tu amigo y te pidiera un consejo, ¿qué le dirías? Escríbelo en un minuto.

Actividad 2

Reúnete en un equipo de cuatro o cinco compañeros.

a. Compartan sus respuestas de la actividad anterior.
b. Imaginen que están en la situación de José Luis. Hagan una lista con máximo cinco elementos que tendrían en cuenta para tomar la decisión con base en su experiencia.
c. Consensúen y escriban tres consejos que darían a José Luis para tomar su decisión.

Reafirmo y ordeno

¿Qué decisión debe tomar José Luis? Probablemente, la información que tenemos al respecto no es suficiente para dar una respuesta definitiva. ¿Cómo podríamos ayudarle? ¿Qué aspectos sería relevante colocar en cada platillo de la balanza para lograr **tomar su decisión**? Si te interesa conocer las respuestas a estas cuestiones, ¡No faltes a clase! Lo vas a descubrir en las siguientes lecciones.

¿Quieres saber si lo que anotaron en la Actividad 2 es correcto? No te preocupes, lo revisaremos en la variación 12.1. Es muy probable que los puntos que tocaron en sus respuestas sean abordados a lo largo del curso.

Para tu vida diaria

Revisa el video de la siguiente sección y, a partir de ello, anota alguna decisión que te gustaría tomar. Escríbela aquí o en tu cuaderno.

¿Quieres saber más?

Te recomendamos el siguiente video sobre la importancia de tomar decisiones y las desventajas de no hacerlo:
https://www.youtube.com/watch?v=Vdx0OxbiBL8&t=55s

Concepto clave

Toma responsable de decisiones: habilidad de elegir de forma autónoma, consciente, responsable y ética ante diversas situaciones, considerando las metas asociadas a un proyecto de vida, las alternativas disponibles y las posibles consecuencias de su comportamiento, con el fin de promover el bienestar individual y colectivo.

Lectura 3. Análisis del Sistema "Modelando mis Datos"

Instrucciones: *Realiza el análisis de la siguiente lectura, posteriormente realiza la Actividad 4 y 5 Análisis del Sistema "Modelando mis Datos".*

¿Qué es el análisis de sistemas de información?

El Análisis de Sistemas según James Senn (2003). Es el propósito de examinar la situación de una empresa con el propósito de mejorarla, con métodos y procedimientos más adecuados. Es comprender en su totalidad el viejo sistema y determinar la mejor forma en que se puede (si es posible), utilizar la informática para hacer la operación más eficiente.

El análisis es el proceso de clasificación y de interpretación de los hechos, diagnósticos de problemas y empleo de la información para recomendar mejoras al sistema.

El Análisis de Sistemas según **Victor Weimberg.**
La define como "la encargada de examinar el problema, los objetivos, los requerimientos, prioridades y límites del entorno, más la identificación de costos beneficios estimados y el tiempo requerido para una solución tentativa." (Structured Analysis)

Propósito del análisis de sistemas de información
Es conseguir la especificación detallada del sistema de información, a través de una serie de modelos que cubran las necesidades de información de los usuarios.

El rol del analista de un sistema de información es el que se encarga de identificar las necesidades de sistemas de información en Tecnología de la Información y Comunicación (TIC) en las empresas para elaborar soluciones integrales a través de proyectos que proponga.

Análisis
Define los requerimientos de negocios para un **sistema nuevo** y responde a la pregunta, **"¿Qué necesitan y desean los usuarios de un sistema nuevo?**

En distintas metodologías la fase de **análisis de requerimientos** podría ser llamada fase de definición o fase de diseño lógico. La base para esta tarea se estableció en el primer paso de la metodología de desarrollo de software con la identificación de necesidades cuando realizamos la definición de problemas y los objetivos de mejoras del sistema.

Para identificar y expresar los requerimientos del sistema, los analistas expresan requerimientos funcionales, estos son frecuentemente identificados en términos de entradas, salidas, procesos y datos almacenados que son necesarios para satisfacer los objetivos de mejora del sistema.

Existen métodos interactivos clave que puede usar para obtener los requerimientos humanos de información de los miembros de la organización:

- **Entrevistas:** Se pueden estructurar en tres formas: pirámide, embudo o diamante. Las estructuras de pirámide empiezan con preguntas cerradas detalladas y se amplían con preguntas más generalizadas. Las estructuras de embudo empiezan con preguntas abiertas generales y después se restringen a preguntas cerradas más específicas. Las estructuras en forma de diamante combinan las ventajas de las otras dos estructuras, pero se requiere más tiempo para llevarlas a cabo.
- **Diseño de aplicaciones conjuntas (JAD)**: Mediante el uso de JAD los analistas pueden analizar los requerimientos humanos de información y diseñar una interfaz de usuario con los usuarios en un ambiente de grupo.
- **Encuestas aplicadas a las personas mediante cuestionarios**: Los analistas de sistemas pueden recopilar datos sobre cuestiones de HCI (Interacción, Humano, Computadora), posturas, creencias, comportamiento y características de las personas clave en la organización.

Los analistas utilizan la entrevista para desarrollar su relación con un cliente, observar el entorno de trabajo y recolectar datos.

En forma general se obtiene datos que provienen de la revisión de documentos y un resumen de cada entrevistado debe contener:

• Resumen de las funciones que realiza
• Clasificación de los problemas identificados
• Análisis de las mejoras potenciales
• Cambios propuestos y su impacto Análisis de la relación entre los cambios propuestos y los planes existentes para la organización y el departamento.

Se considera la opinión de los usuarios del sistema para la obtención de los requerimientos del sistema.

Análisis de Requerimientos

Un requerimiento es una característica necesaria que deberá poseer el nuevo sistema. Por otra parte, la determinación de requerimientos es el estudio de un sistema para comprender cómo trabaja y dónde es necesario efectuar mejoras.

Existen tres formas (= actividades) de determinar de requerimientos, a saber:

Anticipación de requerimientos.	Prever las características del nuevo sistema con base en experiencia previa.
Investigación de requerimientos.	Actividad más importante del análisis de sistemas. Es el estudio y documentación del sistema actual usando para ellos técnicas para hallar hechos, análisis de flujo de datos y análisis de decisión. Es aquí donde aplicamos entrevistas, cuestionarios, observación y revisión de documentación entre otros.
Especificación de requerimientos.	Los datos obtenidos durante la recopilación de hechos se analizan para determinar las especificaciones de los requerimientos, es decir, la descripción de las características del nuevo sistema.

Las siguientes preguntas sugeridas presentarán un conjunto de hechos que al dárseles respuestas se obtendrá una especificación de requerimientos lo más apegada posible a las necesidades de cualquier organización.

Requerimientos básicos: los analistas estructuran su investigación al buscar respuestas a algunas de las siguientes preguntas y de las cuales deben de tener respuestas concretas al tener terminada la fase de investigación de requerimientos:

- ¿Cuál es el proceso básico de la empresa?
- ¿Qué datos utiliza o produce este proceso?

Siempre se debe comenzar con lo básico. Los analistas hacen preguntas que cuando reciben respuesta, proporcionan antecedentes sobre detalles fundamentales relacionados con el sistema y que sirven para describirlo. Las siguientes preguntas son de utilidad para adquirir la comprensión necesaria:

- ¿Cuál es la finalidad de la actividad dentro de la empresa?
- ¿Qué pasos se siguen para realizarla?
- ¿Dónde se realizan estos pasos?
- ¿Quiénes los realizan?
- ¿Cuánto tiempo tardan en efectuarlos?
- ¿Con cuánta frecuencia lo hacen?
- ¿Quiénes emplean la información resultante?

Respuestas concisas a estas preguntas proporcionan un conocimiento amplio de una actividad en particular y muestra también su objetivo. Pero el analista no se detiene ahí, todavía no existe información

para comprender en su totalidad la actividad; más bien lo que se tiene son los antecedentes que permiten a los analistas formular preguntas más detalladas.
Durante esta, debemos identificar muy claramente los siguientes elementos:

- procesos
- flujos de datos entre procesos
- datos de cada flujo de datos
- almacenes de datos
- datos de los almacenes de datos

Algunas preguntas clásicas para la determinación de requerimientos:

Preguntas generales.	¿Cuáles son las personas claves en el sistema? ¿Por qué son importantes? ¿Qué áreas necesitan un control específico?
Determinación de procesos.	¿Cuáles son las principales actividades que se realizan en la organización y que tienen relación con el proceso que se está modelando?
Descripción de cada proceso identificado.	¿Qué es lo que da inicio a la actividad? ¿Cuál es el objetivo de la misma?
Determinación de datos (flujos y contenido de los flujos). Se debe hacer la pregunta por cada proceso o actividad identificada.	¿De dónde proviene la información que se utiliza en esta actividad? (fuentes) ¿Cuáles son específicamente los datos que recibe esta actividad? ¿De qué manera ingresan a este proceso? (flujos) ¿Qué tablas de referencia y diagramas u otros datos intervienen en la actividad? (documentación involucrada) ¿Qué información se genera en esta actividad? (producto de la actividad) El resultado identificado anteriormente producto de los datos que se procesan ¿Hacia qué o quién van dirigidos? (destinos: persona o entidad) ¿Con qué finalidad la utilizan? ¿Cuáles datos se conservan o almacenan en este proceso? Y ¿en qué forma quedan almacenados? ¿Existe información que se genera pero que no es utilizada nunca por nadie?
Para cada dato identificado.	¿Qué formato posee cada dato que interviene en esta actividad? ¿Para qué es usado? ¿Qué tan importante es dicho dato?

Una vez que se tenga recopilado el conjunto de hechos que se generan con relación al sistema que estamos modelando, es posible dar una especificación de requerimientos, mediante un análisis de los datos obtenidos durante la recopilación de hechos. Es después de esto, que se obtiene un conjunto de requerimientos que nos servirán para modelar el sistema mediante un Diagrama de flujo de datos (DFD) y del que surge el diagrama Entidad Relación (E−R).

LA METODOLOGÍA DEL FLUJO DE DATOS PARA DETERMINAR LOS REQUERIMIENTOS HUMANOS

Para que los analistas de sistemas puedan comprender los requerimientos de información de los usuarios, deben ser capaces de conceptualizar la forma en que los datos se mueven a través de la organización, los procesos o la transformación por la que pasan los datos y las salidas de estos.
Por medio de una técnica de análisis estructurado conocida como diagramas de flujo de datos (DFD), el analista de sistemas puede ensamblar una representación gráfica de los procesos de datos a través de la organización. Al usar combinaciones de sólo cuatro símbolos (los cuales abordaremos más adelante), el analista puede crear una descripción ilustrada de los procesos con el fin de elaborar una documentación sólida para el sistema.

Análisis del Flujo de Datos
La estrategia del flujo de datos muestra el empleo de éstos en forma gráfica. Las herramientas usadas para seguir esta estrategia muestran todas las características esenciales del sistema y la forma en que se ajustan entre sí. Puede ser difícil comprender en su totalidad un proceso de la empresa si se emplea para ello solo una descripción verbal; las herramientas para el flujo de datos ayudan a ilustrar los componentes esenciales de un sistema junto con sus interacciones.

Diagramas de flujo de datos
Esta es una herramienta gráfica importante que se emplea para describir y analizar el movimiento de los datos a través de un sistema (manual o automatizado), incluyendo procesos, lugares para almacenar datos y retrasos en el sistema.
Según Kendall & Kendall (2011), para comprender mejor el movimiento lógico de los datos a través de una empresa, el analista de sistemas dibuja diagramas de flujo de datos (DFD).

La transformación de datos de entrada en salida por medio de procesos puede describirse en forma lógica e independiente de los componentes físicos (computadoras, gabinetes de archivos, y procesadores de texto) asociados con el sistema.

Ventajas de la metodología del flujo de datos.

La metodología del flujo de datos tiene cuatro ventajas importantes en comparación con las explicaciones narrativas sobre la forma en que se mueven los datos a través del sistema:
1. No hay que comprometerse demasiado pronto con la implementación técnica del sistema.
2. Permite comprender con más detalle la capacidad de interrelación de los sistemas y subsistemas.
3. Se puede comunicar el conocimiento del sistema actual a los usuarios por medio de diagramas de flujo de datos.
4. Se puede analizar un sistema propuesto para determinar si se han definido los datos y procesos necesarios.

Los DFD se enfocan en el procesamiento de los datos o en la transformación de los mismos a medida que avanzan a través de varios procesos.

Los DFD se concentran en el movimiento de los datos a través del sistema, no en los dispositivos o el equipo. Los analistas identifican y describen, desde el inicio hasta el final del proceso, para comprender un área de aplicación o los datos que fluyen por todo el sistema y entonces explican por qué los datos entran o salen y cuál es el procesamiento que se realiza con ellos. Es muy importante determinar cuándo entran los datos al área de aplicación y cuándo salen de ésta.

Estos diagramas son herramientas estructuradas de análisis y diseño, las cuales permiten al analista comprender el sistema y los subsistemas en forma visual, como un conjunto de flujos de datos interrelacionados.

Durante el análisis de flujo de datos se evalúan todos los detalles en términos de los componentes lógicos de flujo de datos, procesos, almacenes de datos, orígenes y destinos. Por lo que los DFD se pueden dibujar con solo cuatro **notaciones sencillas**:

•**Flujo de datos:** Movimiento de datos en determinada dirección, desde un origen hasta un destino en forma de documentos, cartas, llamadas telefónicas o virtualmente cualquier otro medio. El flujo de datos es un paquete de datos.

• **Procesos:** Personas, procedimientos o dispositivos que usan o producen (transforman) datos.

•**Fuente o destino de datos:** Fuentes o destinos externos de datos, que pueden ser personas, programas, organizaciones u otras entidades que interactúan con el sistema pero que se encuentran fuera de sus fronteras. La diferencia fundamental con los procesos es que las fuentes o destinos no transforman información, al menos no dentro de las fronteras del sistema que se está modelando.

• **Almacenamiento de datos:** Es el lugar donde se guardan los datos o al que referencian los procesos en el sistema. El almacenamiento de datos puede representar dispositivos tanto computarizados como no computarizados.

Se utilizan cuatro símbolos básicos para graficar el movimiento de los datos en los diagramas, en la tabla 1 se muestra las convenciones usadas en los diagramas de flujo de datos.

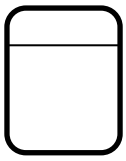
SÍMBOLO/NOMBRE	USO
cuadrado doble 	Se utiliza para describir una entidad externa (otro departamento, una empresa, una persona o una máquina) que pueda enviar/recibir datos hacia/desde el sistema. Se puede utilizar la misma entidad más de una vez en un diagrama de flujo de datos para evitar cruzar las líneas de flujo de datos. Se debe denominar a las entidades con un sustantivo.
flecha 	Muestra el movimiento de los datos de un punto a otro; Los flujos de datos que ocurren al mismo tiempo se pueden describir mediante el uso de flechas paralelas. Como una flecha representa datos sobre una persona, lugar o cosa, también se debe describir con un sustantivo.
rectángulo con esquinas redondas 	Se utiliza un rectángulo con esquinas redondas para mostrar la ocurrencia de un proceso de transformación. Los procesos siempre expresan un cambio o transformación en los datos; por ende, el flujo de datos que sale de un proceso siempre se identifica de manera distinta al flujo que entra al proceso
rectángulo con un extremo abierto (cerrado del lado izquierdo y abierto del lado derecho) 	El almacén de datos puede representar un almacén manual como un archivero, o un archivo o una base de datos computarizada. Como los almacenes de datos representan a una persona, lugar o cosa, se denominan con un sustantivo. Los almacenes de datos temporales, como el papel de borrador o un archivo temporal de computadora, no se incluyen en el diagrama de flujo de datos. Hay que dar a cada almacén de datos un número de referencia único, como D1, D2, D3, por ejemplo.

Tabla 1. Convenciones usadas en los diagramas de flujo de datos.

¿Qué es un diagrama de contexto?

Un diagrama de contexto es una vista de alto nivel del sistema. Es la forma principal de definir entidades en términos de sus restricciones de alcance y sus relaciones con componentes externos, como partes relacionadas.

Este diagrama permite tener un panorama general de los procesos del sistema, centrandose en su interacción con todos los elementos que intervienen en el, así como los subprocesos que trabajan internamente.

DIAGRAMA DE CONTEXTO

CÓMO DESARROLLAR DIAGRAMAS DE FLUJOS DE DATOS

Para empezar un diagrama de flujo de datos, contraiga la narrativa (o historia) del sistema de la organización en una lista con las cuatro categorías de entidad externa, flujo de datos, proceso y almacén de datos. A su vez, esta lista ayuda a determinar los límites del sistema que va a describir. Una vez que haya compilado una lista básica de elementos de datos, empiece a dibujar un diagrama de contexto.
He aquí unas cuantas reglas básicas a seguir:

1. El diagrama de flujo de datos debe tener por lo menos un proceso y no debe haber objetos independientes o conectados a sí mismos.
2. Un proceso debe recibir por lo menos un flujo de datos entrante y debe crear por lo menos un flujo de datos saliente.
3. Un almacén de datos debe estar conectado con por lo menos un proceso.
4. Las entidades externas no se deben conectar entre sí. Aunque se comunican en forma independiente, esa comunicación no forma parte del sistema que diseñamos mediante el uso de DFD

En la Figura 5 se describe los cuatro símbolos básicos que se utilizan en los diagramas de flujo de datos, sus significados y ejemplos.

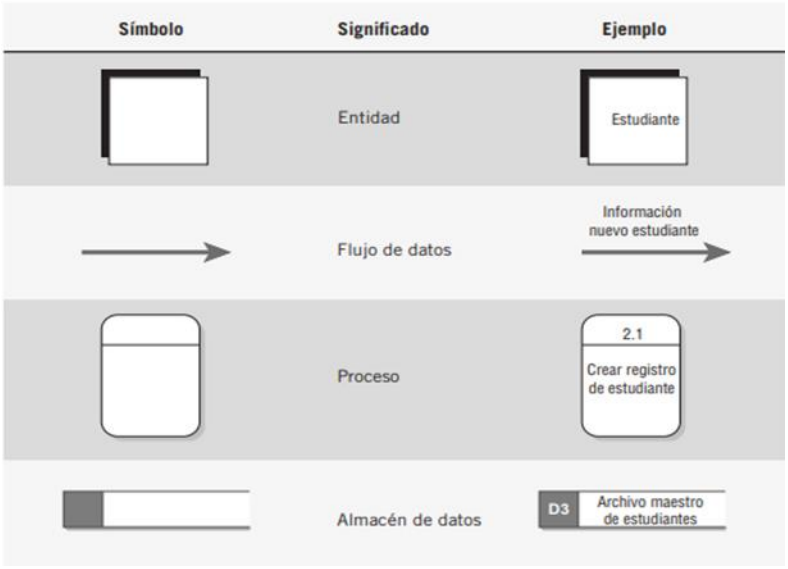
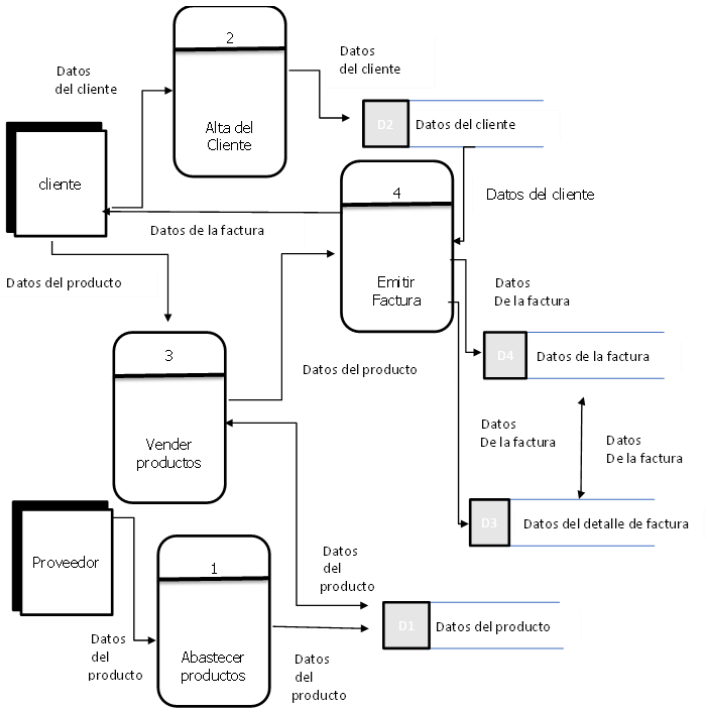


Figura 5. Kendall, Kenneth E. Y Kendall, Julie E (2011). Los cuatro símbolos básicos que se utilizan en los diagramas de flujo de datos, sus significados y ejemplos.

Diagrama de flujo ejemplo de acuerdo con el sistema “Mi tiendita”



Particionar el diagrama de flujo para facilita su programación e implementación. (En caso de ser necesario).

El Diagrama de flujo de datos (DFD) fue utilizado para modelar el sistema Mi Tiendita, del cual surge el diagrama Entidad Relación (E-R).

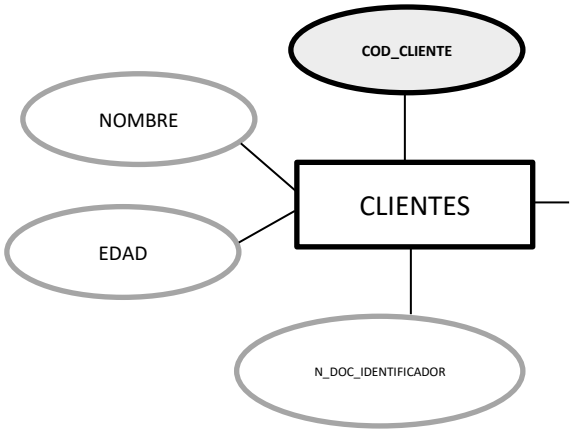
Modelo Entidad-Relación

El modelo entidad relación es una herramienta que permite representar de manera simplificada los componentes que participan en un proceso de negocio y el modo en el que estos se relacionan entre sí.

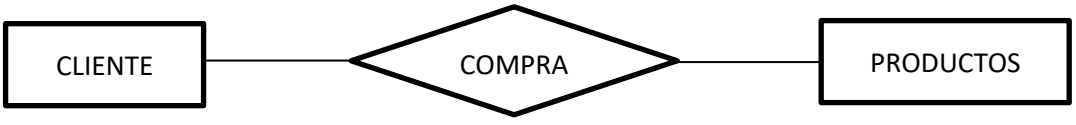
El modelo entidad relación tiene tres elementos principales:

- **Entidades:** El modelo contará con una entidad por cada uno de los componentes del proceso de negocio. Ejemplo: En el sistema Mi tiendita en la que se venden productos, podemos tener entidades "Productos", "Cliente", "Factura", "Detalle_Factura", entre otras.

- **Atributos:** Los atributos, componente fundamental de cada modelo entidad-relación, nos permiten describir las propiedades que tiene cada entidad. "Cod_cliente", "Nombre", "Edad", "N_Doc_Identificador", etc. serán atributos de la entidad "Cliente". Ejemplo:



- **Relaciones:** Con las relaciones se establecen vínculos entre parejas de entidades. Cada "Cliente" podrá adquirir uno o varios "Productos", el proyecto generará una factura en cada compra realizada por un cliente.

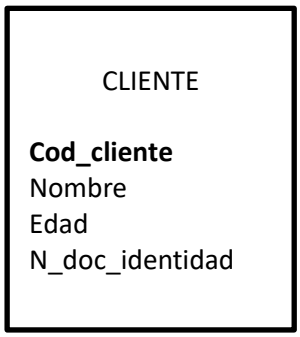


Identificación

Una entidad tiene muchas instancias, allí existe una necesidad de identificar en forma única cada instancia basada en el valor de datos de uno o más atributos, por tanto, cada entidad debe tener una clave, a veces llamado **identificador**.

¿Qué es una clave primaria?

Una **clave primaria** es la clave candidata que más comúnmente se usará para **identificar** en forma única una instancia de entidad individual. Por ejemplo, en la escuela es tu Matrícula, en otros ladosque necesitas hacer algún trámite es tu CURP, el RFC, el código del producto y el código del cliente en el caso del sistema Mi tiendita, etc.



¿Qué es una clave foránea?

Es una referencia a una llave en otra tabla y determina una relación existente entre 2 tablas.

El diagrama entidad relación es la expresión gráfica del modelo entidad relación. En él las ***entidades se representan utilizando rectángulos, los atributos por medio de círculos o elipses y las relaciones como líneas que conectan las entidades***.

Cardinalidad de las relaciones.

El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: **"1:1", "1: N" y "N:M**", aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado.

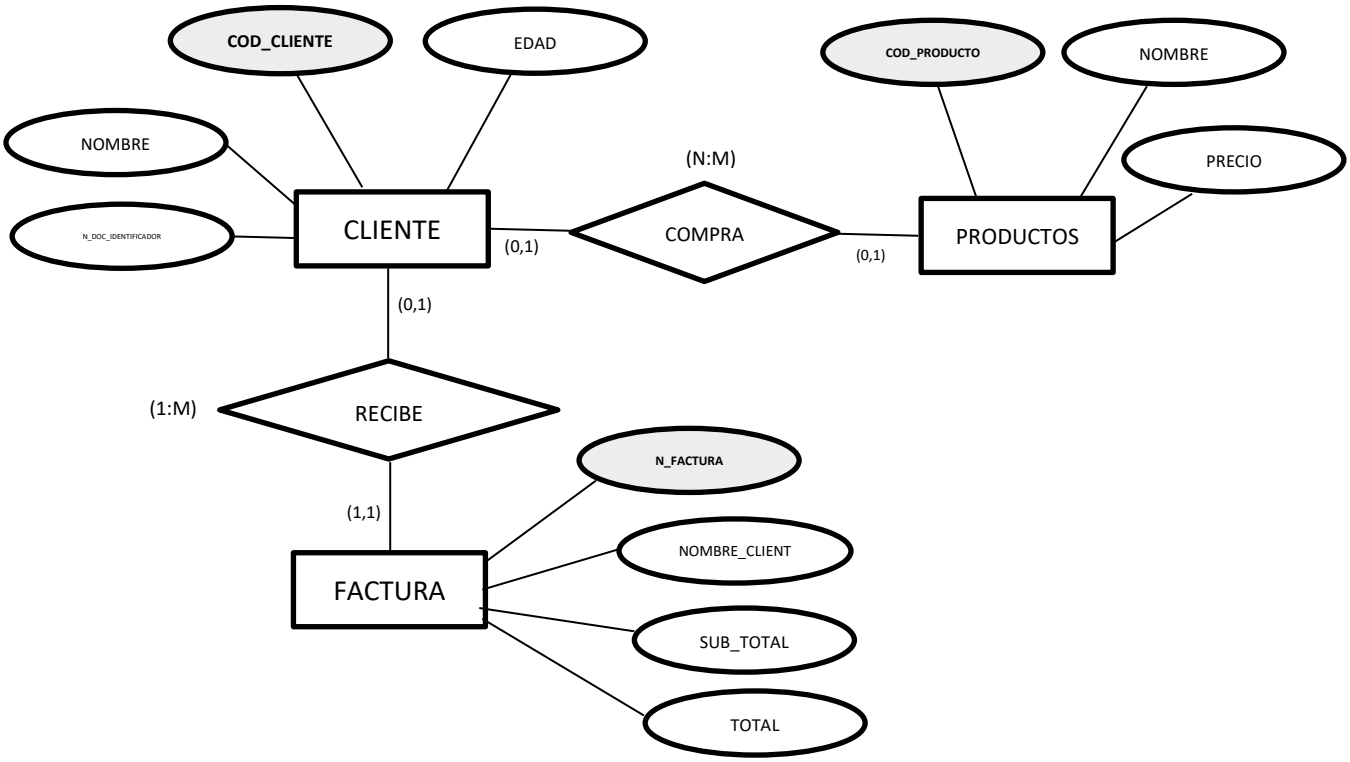
Otra forma de expresar la cardinalidad es situando un símbolo cerca de la línea que conecta una entidad con una relación:

- ✓ "0" si cada instancia de la entidad no está obligada a participar en la relación.
- ✓ "1" si toda instancia de la entidad está obligada a participar en la relación y, además, solamente participa una vez.
- ✓ "N", "M", ó "*" si cada instancia de la entidad no está obligada a participar en la relación y puede hacerlo cualquier número de veces.

Ejemplos de relaciones que expresan cardinalidad:

1. Cada esposo (entidad) está casado (relación) con una única esposa (entidad) y viceversa. Es una relación 1:1.
2. Una factura (entidad) se emite (relación) a una persona (entidad) y sólo una, pero una persona puede tener varias facturas emitidas a su nombre. Todas las facturas se emiten a nombre de alguien. Es una relación 1: N.
3. Un cliente (entidad) puede comprar (relación) varios artículos (entidad) y un artículo puede ser comprado por varios clientes distintos. Es una relación N: M.

Ejemplo del Modelo Entidad-Relación para el sistema de Mi tiendita.



Actividad 4. Análisis del Sistema “Modelando mis Datos”

Instrucciones: Con apoyo de tu Docente, los datos de la lectura y el ejemplo, en equipo de trabajo colaborativo, elabora en alguna aplicación el mismo diagrama de contexto, diagrama de flujo y modelo entidad relación, que se le da como ejemplo, en caso de no contar con computadora o celular elabore el diagrama en su libreta de apuntes.

Diagrama de contexto, Diagrama de flujo de datos del sistema Mi tiendita, modelo entidad relación.

INSTRUMENTO DE EVALUACIÓN LISTA DE COTEJO Actividad 4 y 5. Análisis del Sistema "Modelando mis Datos"					
DATOS GENERALES					
Nombre(s) del estudiante(s)				**Matrícula(s):**	
Producto: Modelo de contexto y Diagrama de flujo.				**Fecha:**	
Materia:				**Periodo:**	
Nombre del Docente:				**Firma del Docente:**	
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Entrega en tiempo y forma la elaboración de los Modelos de datos.				
2	Identifica los requerimientos funcionales para el sistema de gestión informático en el Modelo de contexto.				
2	Identifica actividades y enlista: Entidades externas Flujos de datos Procesos Almacenes de datos				
2	Utiliza adecuadamente los cuatro símbolos para la elaboración de diagramadas de flujo.				
2	El diseño diagrama de flujo, no presenta errores ortográficos y de estética en su elaboración.				
1	Participa y colabora de forma colaborativa, mostrando interés en su realización.				
	CALIFICACIÓN				

Referencias

Caso de uso. Julio 2021, de Educared Sitio web: https://www.ecured.cu/Caso_de_uso

Co, W. U. E. (s/f). Análisis y Diseño de Sistemas de Información. Edu.co. Recuperado el 3 de junio de 2023, de https://www.unipamplona.edu.co/unipamplona/portalIG/home_109/recursos/octubre2014/administraciondeempresas/semestre7/11092015/analisisydisenosistinformacion.pdf

Dzul, F. E. (2020). Tecnología de la Información y la Comunicación III. México: Klik soluciones educativas.

Kendall, Kenneth e. Y Kendall, Julie e. (2011). Análisis y diseño de Sistemas. México: Pearson educación.

Ministerio, Públicas, A., & Versión, M. M. (s/f). Gob.es. Recuperado el 3 de junio de 2023, de https://administracionelectronica.gob.es/pae_Home/dam/jcr:ad870327-56b7-4dfa-af15-e62222f45b90/METRICA_V3_Analisis_del_Sistema_de_Informacion.pdf

Senn James A. (2001). Análisis y diseño de Sistemas de Información. México: Mc. Graw Hill.
Whitten Jefrey L. (2008). Análisis de sistemas, diseños y métodos. México: Mc Graw Hill.

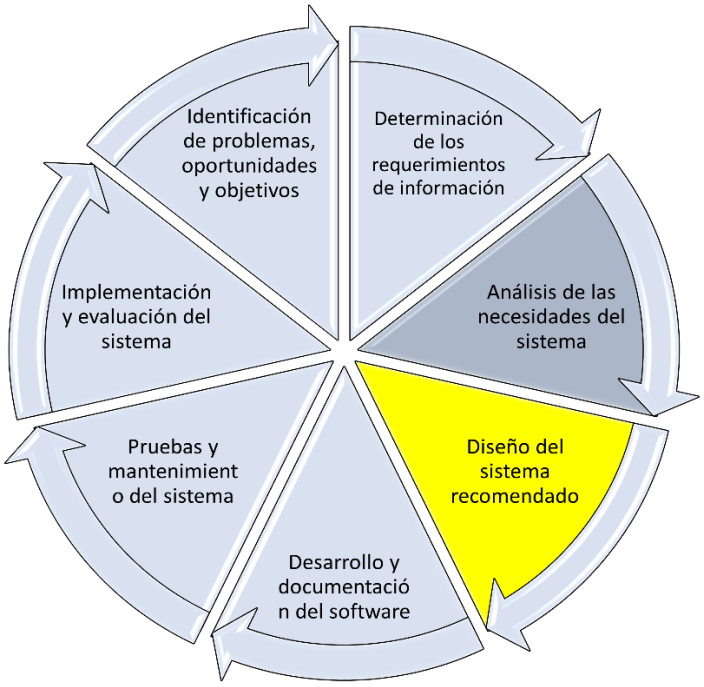
Lectura 4. Diseño del Sistema "Diseñando mis pantallas"

Instrucciones: *Realiza el análisis de la siguiente información, posteriormente realiza la Actividad 7.*

Diseño del sistema recomendado

En la fase de diseño del ciclo de vida del desarrollo de sistemas, el analista utiliza la información recopilada en las primeras fases para realizar el diseño lógico del sistema de información.

El analista diseña procedimientos precisos, para la captura de datos que aseguran que los datos que ingresen al sistema de información sean correctos. Además, el analista facilita la entrada eficiente de datos al sistema de información, mediante técnicas adecuadas de diseño de formularios y pantallas. La concepción de la interfaz de usuario forma parte del diseño lógico del sistema de información. La interfaz conecta al usuario con el sistema y por tanto es sumamente importante. Entre los ejemplos de interfaces de usuario se encuentran, el teclado (para teclear preguntas y respuestas), los menús en pantalla (para obtener los comandos de usuario) y diversas interfaces gráficas de usuario (GUIs, Graphical User Interfaces] que se manejan a través de un ratón o una pantalla sensible al tacto de acuerdo con (Kendall, 2011).



La fase de diseño también incluye el diseño de archivos o bases de datos que almacenarán gran parte de los datos indispensables para los encargados de tomar las decisiones en la organización. Una base de datos bien organizada es el cimiento de cualquier sistema de información. En esta fase, el analista también interactúa con los usuarios para diseñar la salida (en pantalla o impresa) que satisfaga las necesidades de información de estos últimos. Finalmente, el analista debe diseñar controles y procedimientos de respaldo que protejan al sistema y a los datos, y producir paquetes de especificaciones de programa para los programadores.

Cada paquete debe contener esquemas para la entrada y la salida, especificaciones de archivos y detalles del procesamiento; también podría incluir árboles o tablas de decisión, diagramas de flujo de datos, un diagrama de flujo de sistema, y los nombres y funciones de cualquier rutina de código previamente escrita. En resumen, en esta etapa se determina la estructura del sistema, se crean tanto formas para obtener los datos, así como archivos o bases de datos que lo almacenen, los procesos que son necesarios de realizar y las salidas que se obtendrán de cada uno de los procesos. También se crean los procesos de todos los datos, para que tengan una interrelación lógica.

Etapas del diseño del sistema:

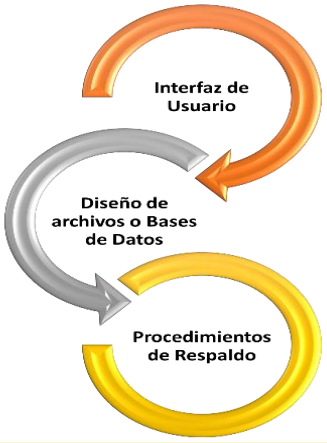
1.- Interfaz de usuario

La interfaz de usuario con el prototipo (y posteriormente con el sistema) es muy importante. Puesto que en realidad su principal objetivo con el prototipo es conseguir que los usuarios expresen mucho mejor sus requerimientos de información, éstos deben interactuar fácilmente con el prototipo del sistema. Para muchos usuarios la interfaz es el sistema. Esto no debe representar un obstáculo.

Aunque no se desarrollarán muchos aspectos del sistema en el prototipo, la interfaz de usuario se debe desarrollar lo mejor posible para permitir a los usuarios una rápida comprensión del sistema y no sentirse desorientados.

En el mismo sentido como parte de la interfaz de usuario, existe la ***interfaz de lenguaje natural*** que es quizá el sueño e ideal de usuarios inexpertos, debido a que permiten a usuarios interactuar con la computadora en su lenguaje cotidiano o natural.

No se requieren habilidades especiales de usuarios, quienes interactúan con la computadora mediante lenguaje natural. Como se observa en la siguiente imagen



Como parte de esta etapa también tenemos las ***interfaces de***

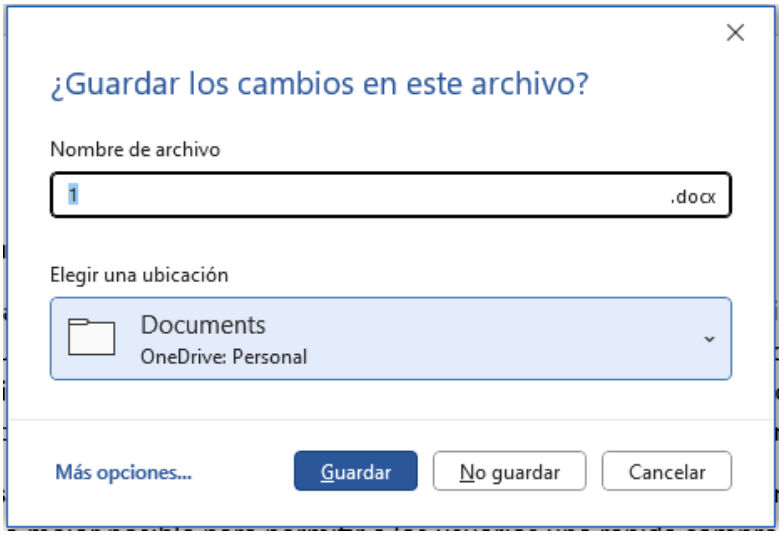
En esta etapa se podrá determinar si algunas funciones de las aplicaciones de los sistemas de información pueden contribuir a que el negocio alcance sus objetivos aplicándolas a problemas u oportunidades específicos. *

- Definición de necesidades
- Análisis
- Diseño

preguntas y respuestas, donde la computadora despliega en pantalla una pregunta para el usuario. Para interactuar el usuario introduce una respuesta, mediante el teclado o ratón y la computadora después actúa en esa información de entrada de acuerdo con su programa normalmente pasando a la siguiente pregunta de acuerdo con (Senn, 2010). Observa la imagen de la derecha

Debido a que una salida es esencial para asegurar el uso y aceptación del sistema de información son varios los objetivos que el analista de sistemas debe tener en mente al diseñarla, estos son (Kendall, 2011):

- Diseñar la salida para satisfacer un propósito específico
- Hacer significativa la salida para el usuario
- Integrar la cantidad de información adecuada para su salida
- Proporcionar una distribución adecuada de salida
- Proporcionar la salida a tiempo
- Elegir el método de salida más adecuado

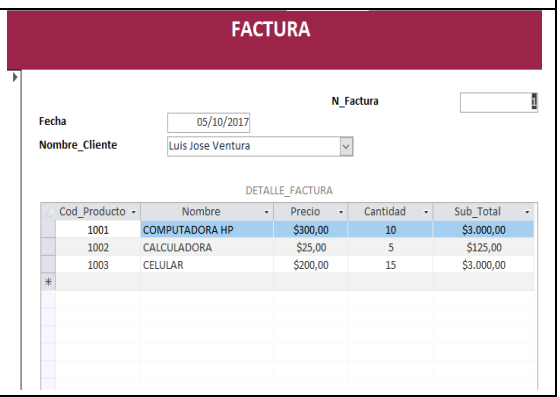


En el diseño de la interfaz de usuario debe ser pensado en que dicho usuario, debe ser capaz de interactuar con la base de datos contenida en el sistema gestor de base de datos de una forma amigable, además de que al usuario común le presenta únicamente la información que requiere, generando vistas de la información. En una **pantalla** el usuario puede incluir imágenes, al igual que en los **reportes**, **encabezados de la pantalla**, **etiquetas de los campos** a utilizar y los campos. Comprende elementos como los **menús**, ventanas, teclado, ratón, sonidos de alerta, es decir, todos aquellos canales por medio de los cuales se establece una comunicación efectiva entre el ser humano y la máquina.

Ejemplo: Formularios, informes y consultas

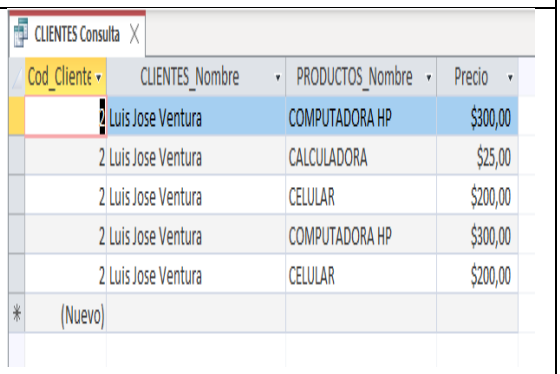
Formulario

Formulario



Informes

Consultas



Toda la salida debe tener un propósito. No es suficiente poner a disposición de los usuarios un informe, una pantalla o una página Web sólo porque la tecnología permite hacerlo. Durante la fase de determinación de los requerimientos de información, el analista de sistemas averigua qué propósitos se deben satisfacer. A continuación, diseña la salida con base en esos propósitos.

Usted verá que tiene numerosas oportunidades de proporcionar salida simplemente porque la aplicación le permite hacerlo. Sin embargo, recuerde la regla del propósito. Si la salida no es funcional, no debe crearse, porque toda salida del sistema representa costos de tiempo y recursos.

Diseño de salida para satisfacer al usuario

En un sistema de información grande que atiende a muchos usuarios con muchos propósitos diferentes, a menudo es difícil personalizar la salida. Con base en las entrevistas, las observaciones, los costos y tal vez los prototipos, será posible diseñar una salida que satisfaga lo que muchos usuarios, si no es que todos, necesitan y prefieren. En términos generales, es más práctico crear salida específica para el usuario,

o que él pueda personalizar, cuando ésta se diseña para un sistema de apoyo a la toma de decisiones u otras aplicaciones sumamente interactivas, como las que se desarrollan para la Web. Sin embargo, aun así, es posible diseñar salidas que satisfagan una función del usuario en la organización, lo cual nos lleva al siguiente objetivo. Hay que recordar que se procura una interfaz amigable y fácil de interactuar con el usuario final.

Menús

Una interfaz de menús adquiere apropiadamente su nombre de la lista de platillos que se pueden seleccionar en un restaurante. De forma similar, una interfaz de menú proporciona al usuario una lista en pantalla de las selecciones disponibles. En respuesta al menú, un usuario está limitado a las opciones desplegadas. El usuario no necesita conocer el sistema, pero tiene que saber qué tarea se debe realizar. Por ejemplo, con un menú típico de procesamiento de texto, los usuarios pueden escoger opciones para editar, copiar o imprimir. Sin embargo, para utilizar el mejor menú los usuarios deben saber qué tarea desean desempeñar. Los menús no dependen del hardware. Las variaciones abundan. Los menús se establecen para usar el teclado, lápiz óptico o el ratón. Las selecciones se pueden identificar con un número, carta o palabra clave. La consistencia es importante en el diseño de una interfaz de menú. Los menús también se pueden ocultar hasta que el usuario quiera usarlos.

Los menús se pueden anidar dentro de otro para llevar a un usuario a las opciones de un programa, también permiten a la pantalla aparecer menos desordenada, lo cual es consistente con el adecuado diseño. En este sentido, permiten a usuarios evitar ver opciones de menú en las que no están interesados y además pueden mover rápidamente a los usuarios a través del programa.

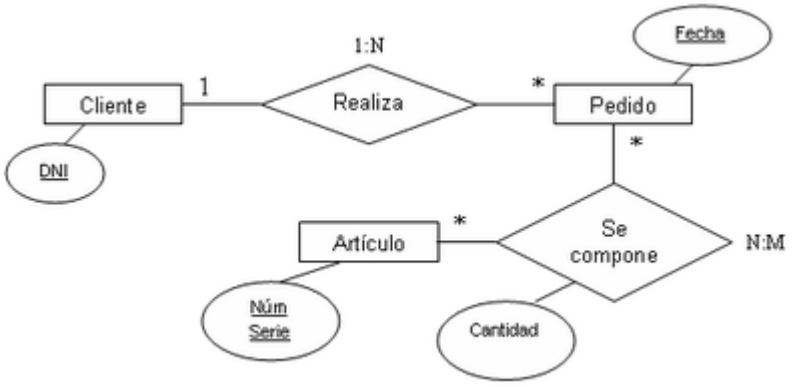
2.- El diseño de archivos o bases de datos del sistema

Los diseñadores de sistemas utilizan con frecuencia los diagramas entidad-relación (E-R) para modelar archivos o la base de datos. Sin embargo, es aún más importante que el analista de sistemas entienda en las fases iniciales las entidades y las relaciones del sistema

organizacional. El modelo E-R, es utilizado en diseño de bases de datos relacionales. Lo importante de este modelo es representar a los datos como entidades que se pueden relacionar con otras, cuya característica es que tienen atributos propios.
(Piattini, 2015)

Para bosquejar algunos diagramas E-R básicos, el analista necesita:

1. Enumerar las entidades de la organización para comprenderla mejor.
2. Seleccionar entidades clave para reducir el alcance del problema a una dimensión manejable y que tenga sentido.
3. Identificar cuál debe ser la entidad principal.
4. Confirmar los resultados de los pasos 1 a 3 a través de otros métodos de recopilación de Datos



Es muy importante que el analista de sistema comience a dibujar diagramas E-R tan pronto se incorpore a la organización en vez de esperar a que se diseñe la base de datos, porque estos diagramas son útiles para que el analista entienda cabalmente el negocio en el que se desenvuelve la organización, determine el tamaño del problema y distinga si se está abordando el problema correcto.

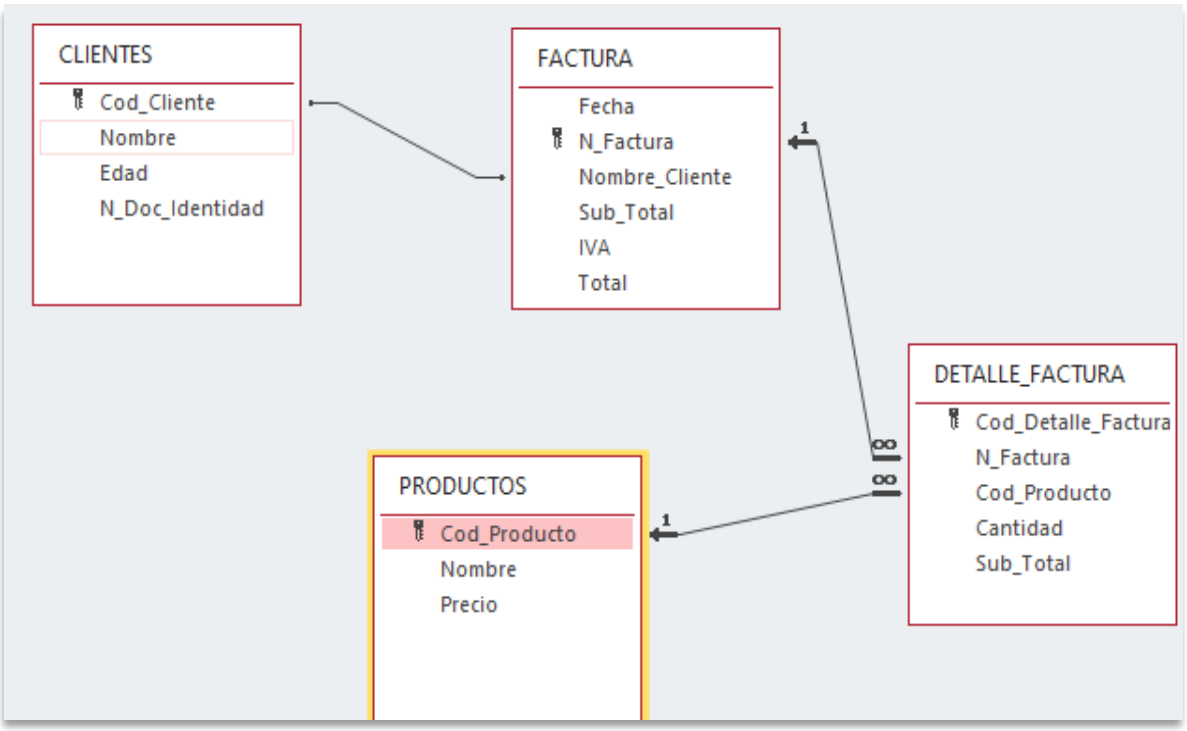
Es necesario confirmar o revisar los diagramas E-R conforme se realiza el proceso de recopilación de datos.

Una ***Base de Datos*** según (Kroenke, 2003) es un conjunto de datos organizados y relacionados entre sí, de forma lógica que en su totalidad dan información sobre aspectos reales. Entendemos como ***dato*** un hecho real conocido que podemos registrar, el cual por sí solo no da información. Definiremos ***información*** como un conjunto de datos que le dan significado a diversos aspectos del mundo real.

Características de una base de datos electrónica o digital:

- Almacena información.
- La información se encuentra indexada o registrada.
- Se mantienen los datos almacenados en un orden tal que permite su búsqueda rápida.
- Incluye un sistema de recuperación rápida llamada consulta.

Ejemplo de relaciones:



Ahora bien, para realizar el diseño de la base de datos de un sistema de información, se requiere contar con el análisis de un **diccionario de datos.**

¿Qué es un diccionario de datos?

El diccionario de datos es una aplicación especializada de los tipos de diccionarios usados como referencia en la vida cotidiana. El diccionario de datos es una obra de consulta con información acerca de los datos (es decir, metadatos), compilada por los analistas de sistemas para guiarse en el análisis y diseño. Como un documento, el diccionario de datos recopila y coordina términos de datos específicos, y confirma lo que cada término significa para las diferentes personas en la organización. (Kendall, 2011). Una razón importante para mantener un diccionario de datos es guardar datos ordenados. Esto significa que los datos deben, ser consistentes. Si usted guarda datos acerca del sexo de un hombre como "M" en un registro, "Masculino" en un segundo registro y como el número "1" en un tercer registro, los datos no son consistentes. Un diccionario de datos ayudará en este aspecto.

Necesidad de entender el diccionario de datos:

Muchos sistemas de administración de base de datos están equipados con un diccionario de datos automatizado, estos pueden ser complejos o sencillos. Algunos son computarizados catalogan automáticamente los elementos de datos cuando se hace la programación; otros simplemente proporcionan una plantilla para motivar a la persona que llene el diccionario a que lo haga de una manera

uniforme para cada entrada.

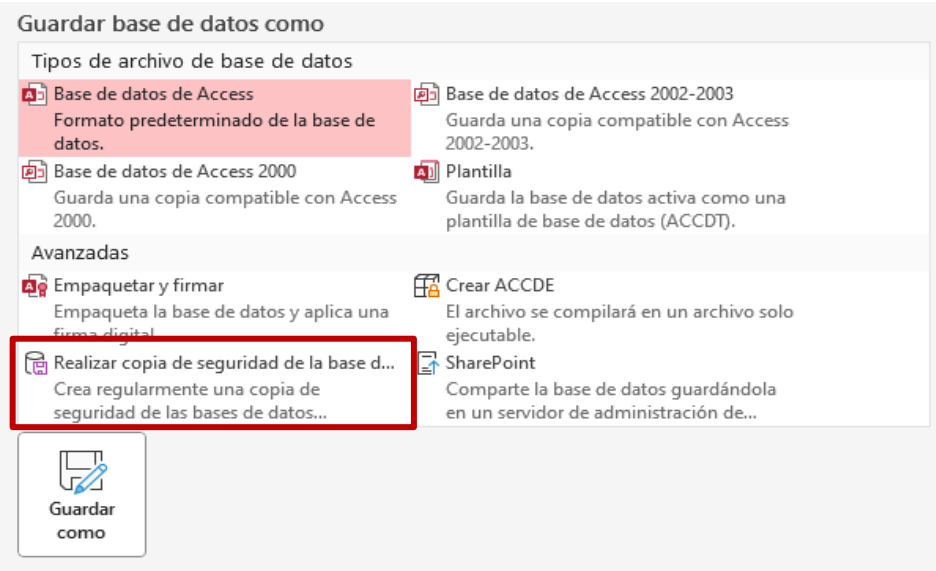
Nombre del campo	Tipo de datos	Tamaño del campo	Descripción

El diccionario de datos estará formado por la descripción de cada una de las entidades de su base de datos: ***Clientes, Detalle factura, factura y Productos.***

Ejemplo de un diccionario de datos de la Tabla Clientes:

Nombre del campo	Tipo de datos	Tamaño del campo	Descripción
Cod_cliente	Autonumeración	Entero	Código de identificación del cliente
Nombre	Texto corto	50	Nombre del cliente
Edad	Número	Entero	Edad del cliente
N_Doc_Identidad	Número	Entero	Número de documento de identidad

3.-Procedimientos de respaldo



Finalmente, el analista debe diseñar controles y procedimientos de respaldo que protejan al sistema y a los datos, y producir paquetes de especificaciones de programa para los programadores. Cada paquete debe contener esquemas para la entrada y la salida, especificaciones de archivos y detalles del procesamiento; también podría incluir árboles o tablas de decisión, diagramas de flujo de datos, un diagrama de flujo de sistema, y rutina de código previamente escrita.

En el mismo sentido, la seguridad física se refiere a proteger el sitio donde se encuentra la computadora, su equipo y software a través de medios físicos. Puede incluir acceso controlado a las salas de cómputo por medio de signos legibles por la máquina o un registro de entrada y salida del sistema por un humano, usando cámaras de televisión de circuito cerrado para supervisar las

áreas de la computadora y frecuentemente apoyando los datos y almacenando los respaldos en un área a prueba de fuego y agua.
Además, el equipo de cómputo pequeño se debe asegurar para que un usuario típico no pueda moverlo y se debe garantizar el suministro ininterrumpido de energía eléctrica. Las alarmas que notifican a las personas apropiadas en caso de fuego, inundación o intrusión no autorizada de una persona deben estar en todo momento en funcionamiento activo. (Kendall, 2011)

El analista debe tomar las decisiones acerca de la seguridad física cuando esté planeando las instalaciones de cómputo y la compra de equipo. Obviamente, la seguridad física puede ser mucho mejor si se planea con antelación a la instalación real y si las salas de cómputo se dotan de equipo de seguridad especial cuando se construyen en lugar de equiparse después de que están construidas. En Access se hace mediante la siguiente opción, de acuerdo con la imagen anterior.

Actividad 5. Opción A: "Diseñando mi interfaz"

Nombre del Sistema y Logotipo del SI

Instrucciones: En equipos colaborativos realiza el diseño de tu primer Sistema de Información piensen en un nombre y un logotipo creativo que lo haga único. Ejemplo: **¿Recuerdas las siglas SGII?** Significan "**S**istema de **G**estión **I**ntegral **I**nformático" y es el sistema donde los Docentes registran calificaciones y donde los estudiantes hacen las consultas respectivas de su historial académico.

Algunos de los aspectos a considerar en el nombre y logo de tu sistema deben ser: que sea distintivo, atractivo, fácil de pronunciar, escribir y de recordar, además debe estar relacionado con el contexto o sistema que están diseñando, así que es momento de poner manos a la obra y empezar con el diseño.
Nota: En caso de no contar con equipo de cómputo, deberás realizar la actividad en tu libreta de apuntes, para posteriormente ser revisada por tu Docente

Etapa 1: Diseñando nuestro logotipo

En primer lugar, iniciaremos diseñando nuestro logotipo que será el distintivo de nuestro sistema. Podemos hacer uso de cualquier programa que hayas utilizado en algunas de tus actividades de otras asignaturas o algún programa en línea que nos permita generar nuestro logotipo. Por ejemplo, Canva o Turbologo, si te decides por alguno de ellos, te invito a revisar los materiales de apoyo

Nombre del Sistema	Logotipo del Sistema

Etapa 2: Diccionario de Datos

Instrucciones: Elaboren el diccionario de datos de su sistema de gestión de información, para cada una de las ***tablas o entidades*** de su base de datos de manera digital, de acuerdo con el ejemplo mencionado en la lectura. Recuerda que ya debes estar empleando Access como nuestro sistema manejador de Base de Datos. Tu diccionario debe ir quedando de la siguiente manera, algo similar a lo que haces con las tablas que ya generaste.

Nombre del campo	Tipo de datos	Tamaño del campo	Descripción

Etapa 3: Diseño de pantallas

Instrucciones: Con la información del diccionario de datos, diseña la interfaz del sistema (**pantallas**) de manera digital para realizar las capturas de los registros, consultas y los reportes generados en su sistema de gestión informático que se está desarrollando y utiliza como referencia los ejemplos de la lectura anterior.

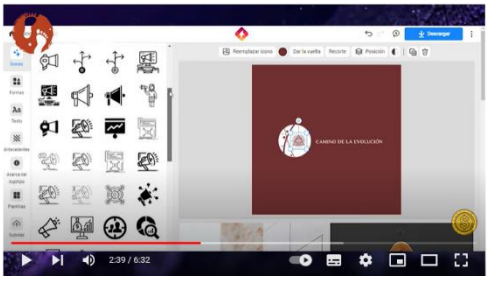
PANTALLAS O FORMULARIOS

Instrumento de Evaluación
LISTA DE COTEJO
Actividad 4. "Diseñando mis pantallas"

DATOS GENERALES	
Nombre(s) del alumno(s)	**Matrícula(s)**
Producto: Diseño de pantallas	**Fecha**
Materia:	**Periodo**
Nombre del Docente	**Firma del Docente**

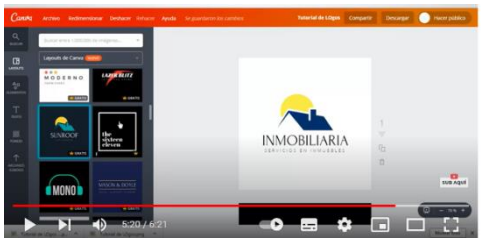
VALOR DEL REACTIVO	CARACTERISTICAS A CUMPLIR	VALOR OBTENIDO		CALIF	OBSERVACIONES Y/O SUGERENCIAS DE MEJORA
		SI	NO		
1	Identifica correctamente las entidades o tablas dentro de su base de datos.				
1	Coloca un nombre a su sistema de gestión informático				
1	Diseña un logotipo para su sistema de gestión informático				
1	Realiza el diccionario de datos de todas sus entidades o tablas.				
2	Diseña de manera correcta y creativa las pantallas de ingreso de datos				
1	Realiza los informes o reportes generados por su sistema de gestión informático				
1	Participa y colabora de forma colaborativa, mostrando interés en su realización.				
1	Identifica correctamente las entidades o tablas dentro de su base de datos.				
1	Coloca un nombre a su sistema de gestión informático				
10	**CALIFICACIÓN**				

Recurso Didáctico Sugerido (Turbologo)



https://youtu.be/BKkGWHe_SAA

Recurso Didáctico Sugerido (Canva)



https://youtu.be/d14ET_FVpYA

Recurso Didáctico Sugerido

https://www.efectodigital.online/single-post/2018/04/18/dise%C3%B1o-de-interfaz-de-usuario-ui

Recurso Didáctico Sugerido (Diccionario)

Tabla: Profesor			Fecha: 31/03/2020
Descripción:	Tabla que contendrá los datos necesarios del profesor para la base de datos.		
Campo	**Tamaño**	**Tipo de dato**	**Descripción**
Id__MatriculaP	7	Numérico	Matrícula asignada al profesor que contiene el año de ingreso.
Apellidos_P	20	Texto	Contiene el Apellido Paterno y Materno del profesor: Apellidos_P = Apellido Paterno + Apellido Materno.
Nombre	20	Texto	Nombre completo del profesor, puede ser 1 o más nombres.
Edad	2	Numérico	Edad del profesor
Grado_A	15	Texto	Nivel de estudio del profesor: Técnico, Licenciatura, Maestría o Doctorado.

https://youtu.be/73RfbzNr-LM

Referencias

Currículos exploratorios en TIC (2020). Diseño de interfaz. http://contenidos.sucerman.com/nivel4/desarrollo/unidad2/leccion4.html

Educativa ITCA (s. f.). Esquema de menús y diseño de pantallas. https://virtual.itca.edu.sv/Mediadores/ads/124_esquema_de_mens_y_diseo_de_pantallas.html

Microsoft. (14 de junio de 2021). *Introducción a formularios*. Obtenido de https://support.microsoft.com/es-es/office/introducci%C3%B3n-a-formularios-e8d47343-c937-44e8-a80f-b6a83a1fa3ae

Kendall, K. E. (2016). *Análisis y Diseño de Sistemas de Información.* México: Pearson.

Kroenke, D. M. (2003). *Procesamiento de Bases de Datos.* México: Pearson.

Piattini, M. y. (2015). *Fundamentos y modelos de base de datos.* México: Alfaomega.

Senn, J. (2010). *Análisis y diseño de sistemas de información.* México: McGraw-Hill.

Whitten Jefrey L. (2008). Análisis de sistemas, diseños y métodos. Séptima edición. México.Editorial Mc Grw Hill. ISBN:9789701066140

Lectura 5. Codificación “Soy un programador”

Instrucciones: *Realiza el análisis de la siguiente información posteriormente realiza la Actividad 5.*

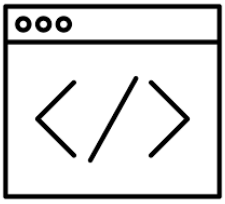
La codificación es el proceso de usar lenguajes de programación para dar instrucciones a una computadora. Estas instrucciones impulsan los sitios web, el software y las aplicaciones que la gente usa todos los días.

Es la conversión de un algoritmo en programa, utilizando un lenguaje de programación.

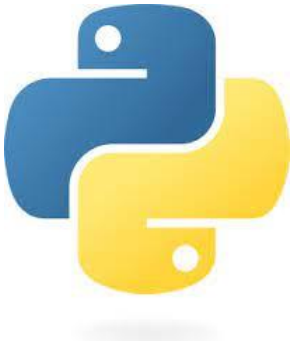
Las instrucciones expresadas en lenguaje natural deben ser expresadas en lenguaje de programación correspondiente. (Codificación - Programación Estructurada, 2023)

Los lenguajes de programación más conocidos son :

- Python.
- Java.
- Java script.
- C#.
- PHP.



Python



Python es uno de los lenguajes de programación más usados actualmente y su uso sigue creciendo. Posee unas características muy potentes: es de código abierto, tiene una sintaxis sencilla y es fácil de entender, por lo que ahorra tiempo y recursos. (Welcome to Python.org, 2023).

Python es un lenguaje versátil que puede tener múltiples aplicaciones. Una de ellas, es la Inteligencia Artificial, gracias a bibliotecas como Keras o TensorFlow.

Java

Java es un lenguaje de propósito general, orientado a objetos y diseñado para tener las dependencias de implementación mínimas posibles. Con Java se pueden crear aplicaciones y procesos en múltiples dispositivos.

Su ámbito de aplicación es muy amplio, por lo que permite crear software para dispositivos móviles, terminales de venta, IoT, además de páginas web. (Software Java, 2020)

JavaScript

JavaScript es un lenguaje de programación interpretado, orientado a objetos y que se utiliza principalmente en la forma del lado del cliente.

Este lenguaje de programación sirve para todo: aplicaciones web, servidores, aplicaciones móviles… Su peculiaridad es que no necesita compilación ninguna, ya que es el propio navegador quién lee el código y realiza las acciones que le indica. Por este motivo, es uno de los lenguajes para crear páginas web cuando se quieren plantear elementos interactivos o más visuales. (Learn JavaScript Online - Courses for Beginners - Javascript.com, 2020)

Sin embargo, gracias a las prestaciones de HTML5 y las librerías de gráficos 2D y 3D, Javascript tiene también un papel relevante para el desarrollo de videojuegos, especialmente, si éste se ejecuta desde el navegador

C#

C# es un lenguaje que sigue apareciendo en los listados de lenguajes de programación más usados. Creado por Microsoft, está presente en entornos empresariales, como instituciones gubernamentales, entidades bancarias o médicas.

Aunque más allá de este tipo de aplicaciones, tiene también usos muy versátiles: internet de las cosas, desarrollo de videojuegos, web o aplicaciones móviles. Para el desarrollo web, puedes utilizar el framework ASP.NET o la herramienta Xamarin para desarrollar aplicaciones nativas para Android e iOS. (BillWagner, 2023).

PHP

PHP es un lenguaje de programación de propósito general de código del lado del servidor. Este lenguaje garantiza una buena comunicación entre web y servidor, por lo que las páginas web desarrolladas con este lenguaje son estables y con buen rendimiento. (PHP: ¿Qué Es PHP? - Manual, 2023)

Actualmente, muchas páginas web están diseñadas con WordPress, que trabaja con PHP. Por ello, este lenguaje te permitirá desarrollar tanto proyectos como plugins para esta plataforma.

FASES

Una vez que se cuenta con los documentos de las fases de análisis y diseño, se inicia la fase de codificación, es decir la programación. La codificación es el proceso de escribir cientos de líneas de código en un lenguaje de programación y para ello es evidente que se elija un determinado lenguaje de programación, conocer la sintaxis del mismo y requerimientos para su empleo.

En esta parte, el desarrollador deberá seguir los lineamientos impuestos en el diseño y tomando en consideración siempre los requisitos funcionales y no funcionales. Las actividades que comprende esta etapa son:

a) Escribir el código: en esta parte, el desarrollador deberá asegurarse que durante la escritura del código está siguiendo las normas y convenciones de codificación. Durante esta fase, el código pasa por diferentes estados:

- ***Código Fuente:*** es el escrito por los programadores en algún editor de texto. Se escribe usando algún lenguaje de programación de alto nivel y contiene el conjunto de instrucciones necesarias.

Para obtener el código fuente de una aplicación informática:

- Se debe partir de las etapas anteriores de análisis y diseño.
- Se diseñará un algoritmo que simbolice los pasos a seguir para la resolución del problema.
- Se elegirá un Lenguaje de Programación de alto nivel apropiado para las características del software que se quiere codificar.
- Se procederá a la codificación del algoritmo antes diseñado.

- ***Código Objeto:*** es el código binario resultado de compilar el código fuente.
  - La *compilación* es la traducción de una sola vez del programa, y se realiza utilizando un *compilador*. La *interpretación* es la *traducción* y *ejecución* simultánea del programa línea a línea.
  - El código objeto no es directamente inteligible por el ser humano, pero tampoco por la computadora. Es un código intermedio entre el código fuente y el ejecutable y sólo

existe si el programa se compila, ya que si se interpreta (traducción línea a línea del código) se traduce y se ejecuta en un solo paso.

- ***Código Ejecutable:*** Es el código binario resultante de enlazar los archivos de código objeto con ciertas *rutinas* y *bibliotecas* necesarias. El sistema operativo será el encargado de cargar el código ejecutable en memoria RAM y proceder a ejecutarlo. También es conocido como *código máquina* y así es directamente inteligible por la computadora.

b) **Realizar pruebas unitarias:** consiste en probar la funcionalidad de una parte del código fuente, tales como rutinas, funciones, etc. con el objetivo de asegurar que los resultados devueltos sean los correctos.

c) **Realizar pruebas de funcionalidad:** una vez que se tenga una versión terminada del producto, el desarrollador debe realizar pruebas para asegurarse que las entradas definidas producen los resultados esperados y que todos los componentes del producto funcionan correctamente.

d) **Realizar el formato de pruebas de interfaces y contenido:** este formato se utiliza para validar el cumplimiento de algunos criterios de estandarización relacionados con: Diseño, Usabilidad, Seguridad, Base de Datos y Código fuente. Debe aplicarse luego de que el desarrollador ha culminado con el entregable y antes de pasar a la instancia de producción.

Programadores y/o desarrolladores de Software

Los programadores o desarrolladores son los encargados de crear software y pueden instalar (o modificar y después instalar) software comprado a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para escribir el software y de la disponibilidad de los programadores. Por regla general, los programadores (o analistas programadores) que trabajan en las grandes organizaciones pertenecen a un grupo permanente de profesionales. En empresas pequeñas, donde no hay programadores, se pueden contratar servicios externos de programación.

Los programadores también son responsables de la documentación de los programas y de proporcionar una explicación de cómo y por qué ciertos procedimientos se codifican en determinada forma. La documentación es esencial para probar el programa y llevar a cabo el mantenimiento una vez que la aplicación se encuentra instalada.

¿Dónde codificar o programar?

Un ***lenguaje de programación*** es como un idioma creado de forma artificial, formado por un conjunto de símbolos y normas que se aplican sobre un alfabeto para obtener un código, que el hardware de la computadora pueda entender y ejecutar. Los lenguajes de programación son los que nos permiten comunicarnos con el hardware del ordenador. Los lenguajes de programación han sufrido su propia evolución, como se puede apreciar en la figura siguiente:

LENGUAJES DE PROGRAMACION

TIC´S

Lenguaje máquina:
- Fue el primer lenguaje utilizado.
- Es único para cada procesador (no es portable de un equipo a otro).
- Hoy día nadie programa en este lenguaje.
- instrucciones son combinaciones de unos y ceros.

Lenguaje ensamblador:
- Sustituyó al lenguaje máquina para facilitar la labor de programación.
- En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas).
- Necesita traducción al lenguaje máquina para poder ejecutarse.

Lenguaje de alto nivel:
- Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación.
- Se utilizan sentencias y órdenes derivadas del idioma inglés. (Necesita traducción al lenguaje máquina).
- Son más cercanos al razonamiento humano.
- Son utilizados hoy día, aunque la tendencia es que cada vez menos.

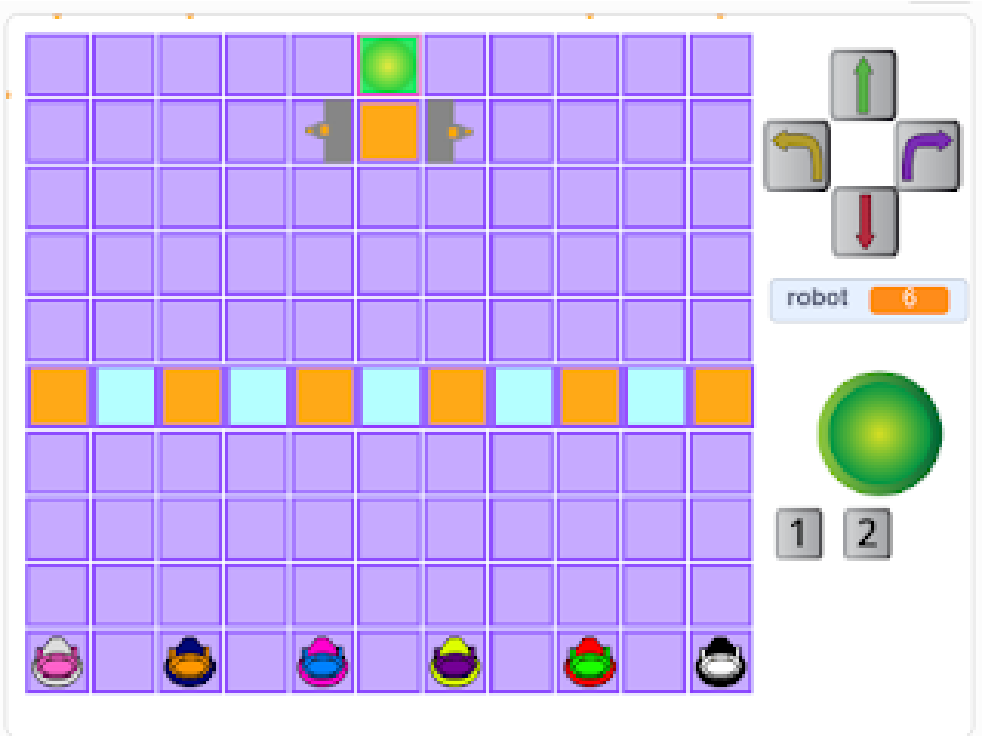
Lenguaje visual:
- Están sustituyendo a los lenguajes de alto nivel basados en código.
- En lugar de sentencias escritas, se programa gráficamente usando el ratón y diseñando directamente la apariencia del software.
- Su correspondiente código se genera automáticamente.

Actividad 6. Rally de Codificación

Instrucciones:

Este juego de robots tipo tablero (de 2 a 6 participantes) para aprender programación, y poner a prueba el razonamiento lógico. Sobre el tablero cada jugador inicia una carrera peligrosa para llegar al final del circuito, cada uno desplaza su propio robot realizando movimientos que previamente debe programar con la ayuda de unas tarjetas que le indican al robot, por ejemplo, que gire, se mueva o recoja energía. Sin embargo, no lo tendrán tan fácil porque existen elementos externos que pueden modificar sus movimientos y cómo se comportan.

El juego está inspirado en la carrera de robots, para poder jugar en pequeños grupos en clase, para utilizar la programación y la lógica.



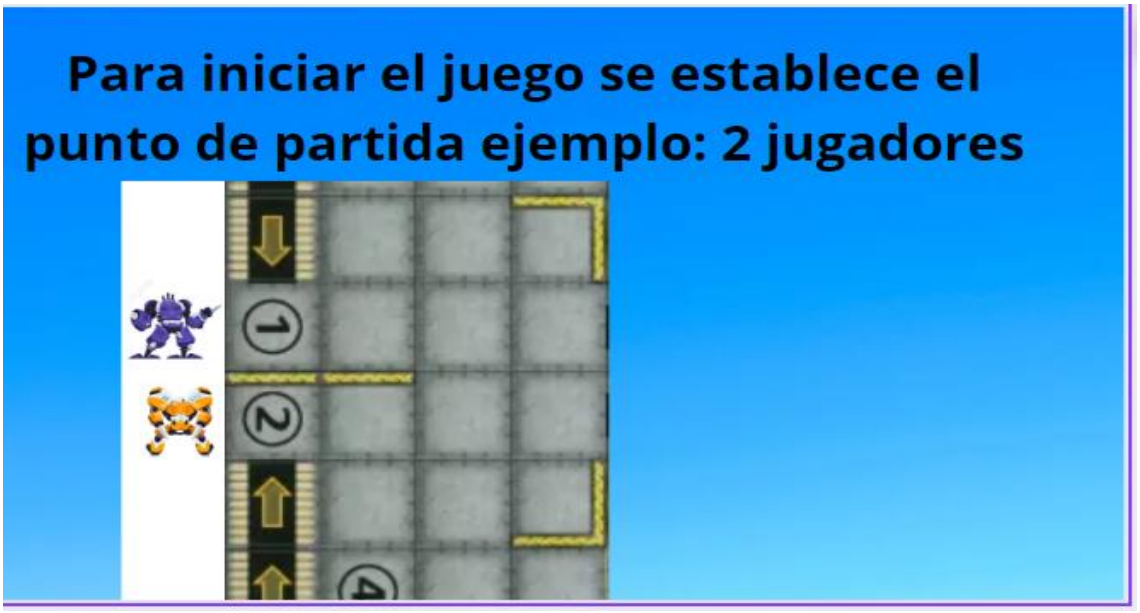
Reglas de juego

- ·Forman equipos, dúos (dependerá del tipo de juego) o individual.
- ·Se decidirá el orden en el que se moverán los robots al azar, orden alfabético, o con un dado.
- ·Se repartirán 9 tarjetas de programación aleatorias a cada integrante.
- ·Cada equipo elegirá 5 de esas cartas para hacer su programación y las colocará en orden sobre el control de la mesa.
- ·Los robots, en el orden establecido, realizarán el primer movimiento.
- ·Si tratan de avanzar y topan con otro robot o una pared, no se moverán. (excepto todo contra vs todos).
- ·Si un robot cae en un pozo de lava, agua, etc., o se sale de los límites aparecerá en su punto de inicio orientado en la última dirección a la que apuntaba.
- ·Cuando todos los robots hayan terminado de hacer su primer movimiento, se realizarán los segundos movimientos en el mismo orden, y se seguirá de igual manera hasta completar los 5 movimientos programados.

Modos de juego 2 -8

- Captura la bandera
    - Consiste en buscar la mayor cantidad de banderas que se encuentran por todo el circuito, gana el jugador que cuente con más banderas.
- Todos contra todos
    - Es una eliminación tipo toca - toca, gana quien quede al final
- Laberinto
    - Consiste en que de los mapas ya existente o creados, busquen la salida de el mismo.

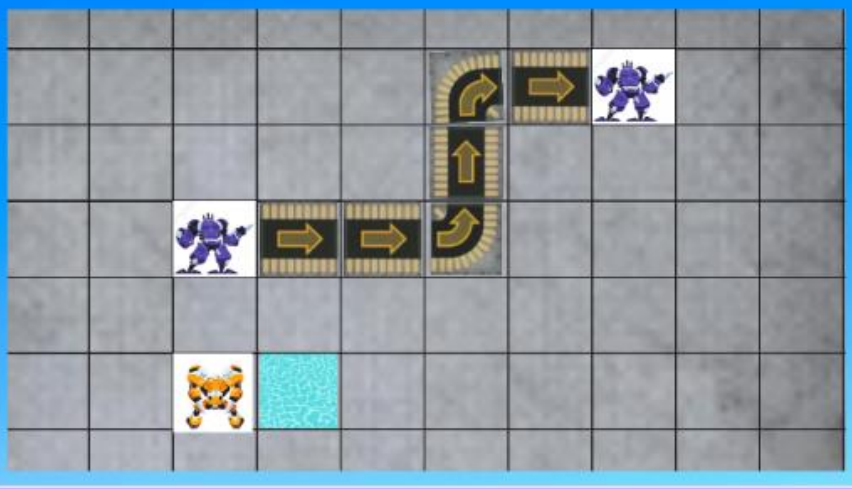

Vamos a jugar !!
Robots
AVANCE 1
AVANCE 2
AVANCE 3
ROTAR DERECHA
RETROCEDE
ROTAR IZQUIERDA
Cartas de codificación
Tablero


Para iniciar el juego se establece el punto de partida ejemplo: 2 jugadores

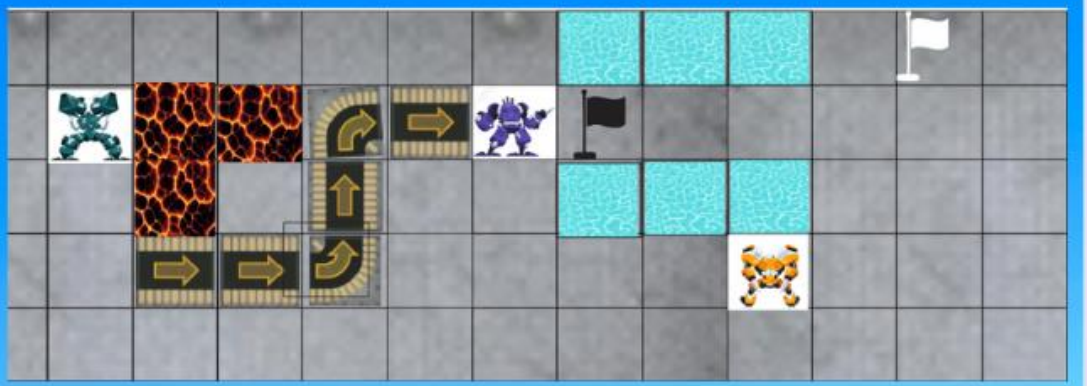

El orden se decide el principio, para movernos usaremos las tarjetas previamente revueltas y se dan 9 a cada participante los movimientos son los siguientes:
AVANCE 1
AVANCE 2
Estas cartas permiten el movimiento del robot hacia adelante 1 o 2 casillas


El orden se decide el principio, para movernos usaremos las tarjetas previamente revueltas y se dan 9 a cada participante los movimientos son los siguientes:
AVANCE 1
AVANCE 2
AVANCE 1
AVANCE 1
De las 9 tarjetas seleccionadas solo se usaran 5 por turno, para acomodarlas de acuerdo al tableto

Tenemos bandas transportadoras que nos ayudan a ir más rápido son como camino, pero tenemos obstaculos como son :



Aqui en captura la bandera ganá el jugador con mas banderas, y que sobrepaso todos los obstaculos.

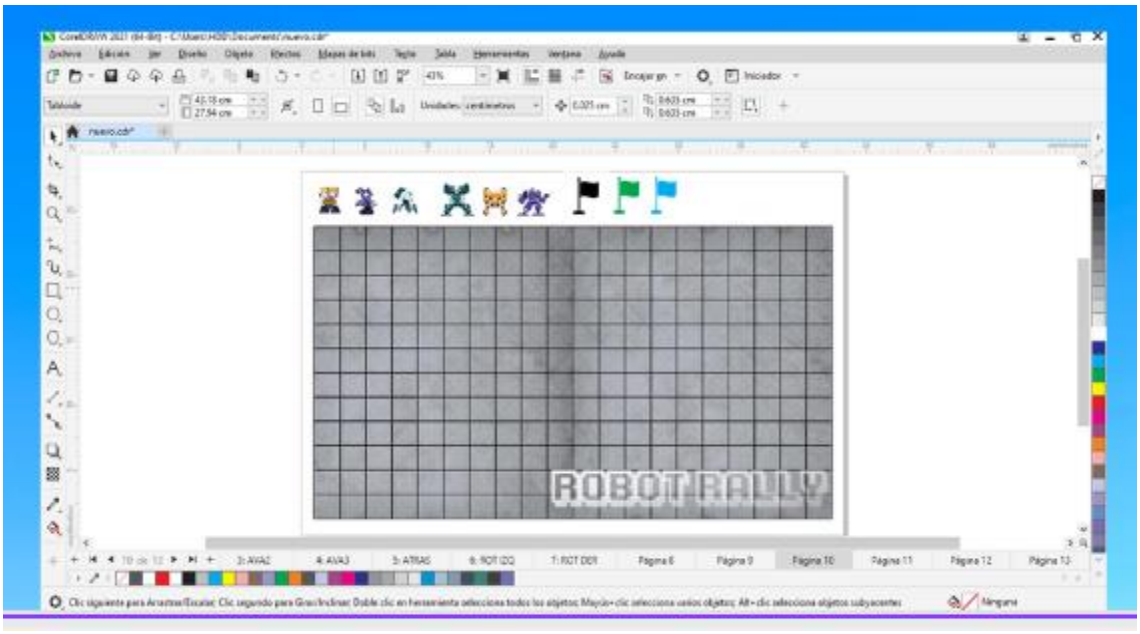


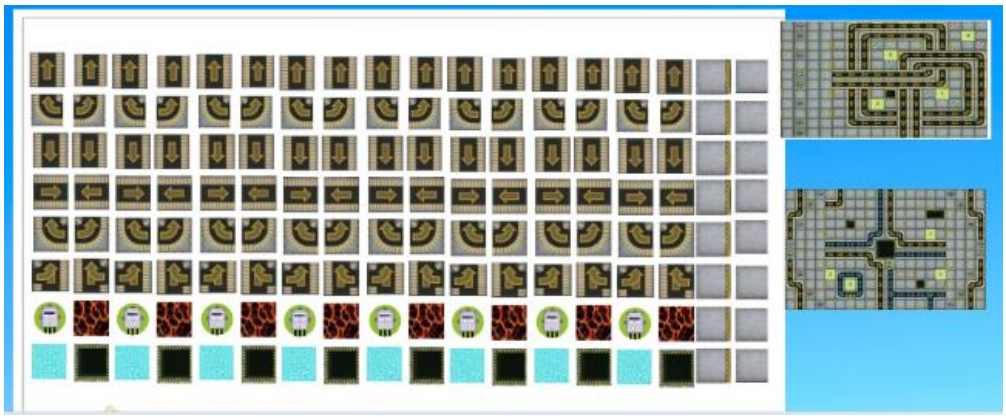
Diviértete y usa tus mejores estrategias para adaptarte a todas las situaciones.

Actividad 6.1 ¡Ahora te toca a ti!

Instrucciones:

En este juego se trata de elaborar un tablero nuevo, con otras rutas y más retos, es colocar caminos y desafíos para compartir con sus compañeros, usando las plantillas ya establecidas con ayuda de su tutor en turno, se pueden recortar y pegar los iconos de bandas transportadoras, lava, agua y agujeros, sobre el tablero original, diviértete con tus compañeros, retándolos a que compitan en tu nueva pista.





Referencias

BillWagner. (s/f). Documentos de C#: inicio, tutoriales y referencias. Microsoft.com. Recuperado el 3 de junio de 2023, de https://learn.microsoft.com/es-es/dotnet/csharp/

Cómo borrar la caché y las cookies. (s/f). Google.com. Recuperado el 3 de junio de 2023, de https://sites.google.com/site/progstr5i/codificacin

Eran. (2020, mayo 17). Actividades de programación sin conexión a internet para niños. TekkieUni - Coding for Kids. https://tekkieuni.com/es/blog/unplugged-coding-activities/

Java. (s/f). Oracle.com. Recuperado el 3 de junio de 2023, de https://www.oracle.com/mx/java/

Learn JavaScript Online - Courses for Beginners - javascript.com. (s/f). Javascript.com. Recuperado el 3 de junio de 2023, de https://www.javascript.com/

¿Qué es PHP? (s/f). Php.net. Recuperado el 3 de junio de 2023, de https://www.php.net/manual/es/intro-whatis.php

Ríos, G. (2022, febrero 16). La codificación de la comunicación es clave para comunicar - Comunicare. Comunicare - Agencia de Marketing Online. https://www.comunicare.es/la-codificacion-un-elemento-vital-para-la-comunicacion/

Welcome to. (s/f). Python.org. Recuperado el 3 de junio de 2023, de https://www.python.org/

Lectura 6. Validación y pruebas de un Si

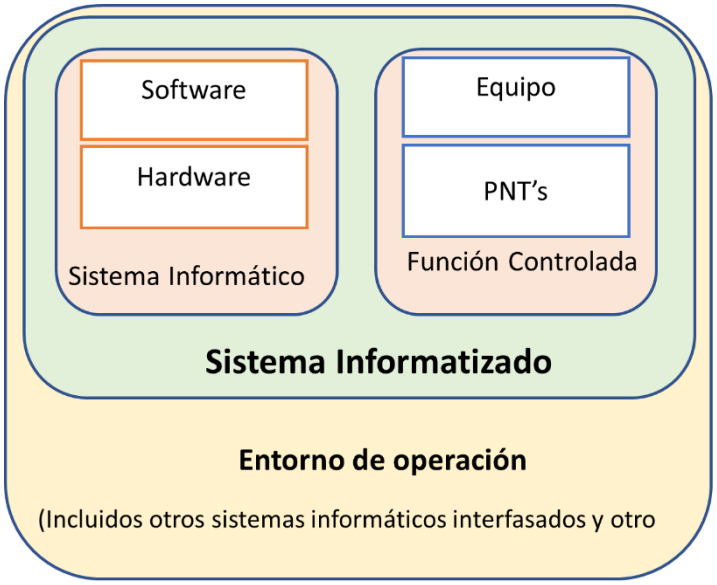
Instrucciones: *Realiza lectura y análisis de la siguiente lectura posteriormente realiza la Actividad 6.*

La validación es el proceso de revisión al que se somete el sistema informático para comprobar que cumple con todas sus especificaciones y es capaz de realizar su cometido en cumplimiento con la normativa aplicable y el uso esperado por el usuario regulado. (Quality, 2020)

Pero la validación de sistemas informatizados no se limita únicamente a las aplicaciones o programas, sino que también se aplica al funcionamiento del hardware y a la integración en tiempo real de dispositivos móviles.

La validación de un sistema informatizado se compone de dos elementos (ver figura siguiente):

- ✓ Sistema informático, compuesto por un hardware y software que actúan al unísono.
- ✓ Función controlada o proceso; que los equipos utilizados, así como los procedimientos normalizados de trabajo (PNT) y la formación de los usuarios que intervienen en el sistema y el proceso.



Pruebas

Llevar a cabo pruebas consiste en ejecutar el sistema con datos de entrada específicamente formulados para la prueba que se realiza. La prueba de insuficiencias o defectos del programa se obtienen analizando la respuesta que proporciona y buscando anomalías respecto de lo esperado.
Las pruebas se pueden llevar a cabo durante la fase de implementación para verificar que el software se comporta tal como los pretendió el diseñador, y después de que la implementación está completa.

Tipos de pruebas
Existen dos tipos diferentes de prueba, que se utilizan en las diferentes etapas de desarrollo del software: (Drake, 2009).

Pruebas de defectos	Pruebas estadísticas
•Buscan la incosistencia entre un programa y su especificación. •Las puebas se diseñan para buscar los erores en el código. •Demuestran la presencia, y no la ausencias de defectos.	•Buscan demostrar que satisface la especificación operacional y su fiabilidad. •Se diseñan para reflejar la carga de trabajo habitual. •Sus resultados se procesan estadisticamente para estimar fiabilidad (contando el número de caídas del sistema) y sus tiempos de respuesta.

Las pruebas de defectos: Pretenden encontrar las inconsistencias entre un programa y su especificación. Estas inconsistencias se deben habitualmente a los fallos o defectos en el código del programa. Las pruebas se diseñan para revelar la presencia de defectos en el sistema, más que para evaluar su capacidad operacional.
Las pruebas estadísticas: se utilizan para probar el desempeño y la fiabilidad del programa y comprobar cómo trabaja bajo condiciones operacionales. Las pruebas se diseñan para reflejar las entradas de los usuarios y su frecuencia. Después de llevar a cabo las pruebas, se puede hacer una estimación de la fiabilidad operacional del sistema contando el número de caídas observadas en el sistema. La capacidad del programa se valora midiendo el tiempo de ejecución y el tiempo de respuesta del sistema cuando procesa los datos estadísticos de la prueba.
No existe una separación clara entre estos dos tipos de pruebas. Durante las pruebas de defectos los desarrolladores obtienen una visión intuitiva de la fiabilidad, y durante las pruebas estadísticas, se descubren obviamente muchos fallos.

Mantenimiento

El objetivo de este proceso es la obtención de una nueva versión de un sistema de información desarrollado, a partir de las peticiones de mantenimiento que los usuarios realizan con motivo de un problema detectado en el sistema, o por la necesidad de una mejora de este.

En este proceso se realiza el registro de las peticiones de mantenimiento recibidas, con el fin de llevar el control de estas y de proporcionar, si fuera necesario, datos estadísticos de peticiones recibidas o atendidas en un determinado periodo, sistemas que se han visto afectados por los cambios, en qué medida y el tiempo empleado en la resolución de dichos cambios. Es recomendable, por lo tanto, llevar un catálogo de peticiones de mantenimiento sobre los sistemas de información, en el que se registren una serie de datos que nos permitan disponer de la información antes mencionada.
En el momento en el que se registra la petición, se procede a diagnosticar de qué tipo de mantenimiento se trata. Atendiendo a los fines, podemos establecer los siguientes tipos de mantenimiento:

- **Correctivo**: son aquellos cambios precisos para corregir errores del producto software.
- **Evolutivo**: son las incorporaciones, modificaciones y eliminaciones necesarias en un producto software para cubrir la expansión o cambio en las necesidades del usuario.
- **Adaptativo:** son las modificaciones que afectan a los entornos en los que el sistema opera, por ejemplo, cambios de configuración del hardware, software de base, gestores de base de datos, comunicaciones, etc.
- **Perfectivo:** son las acciones llevadas a cabo para mejorar la calidad interna de los sistemas en cualquiera de sus aspectos: reestructuración del código, definición más clara del sistema y optimización del rendimiento y eficiencia.

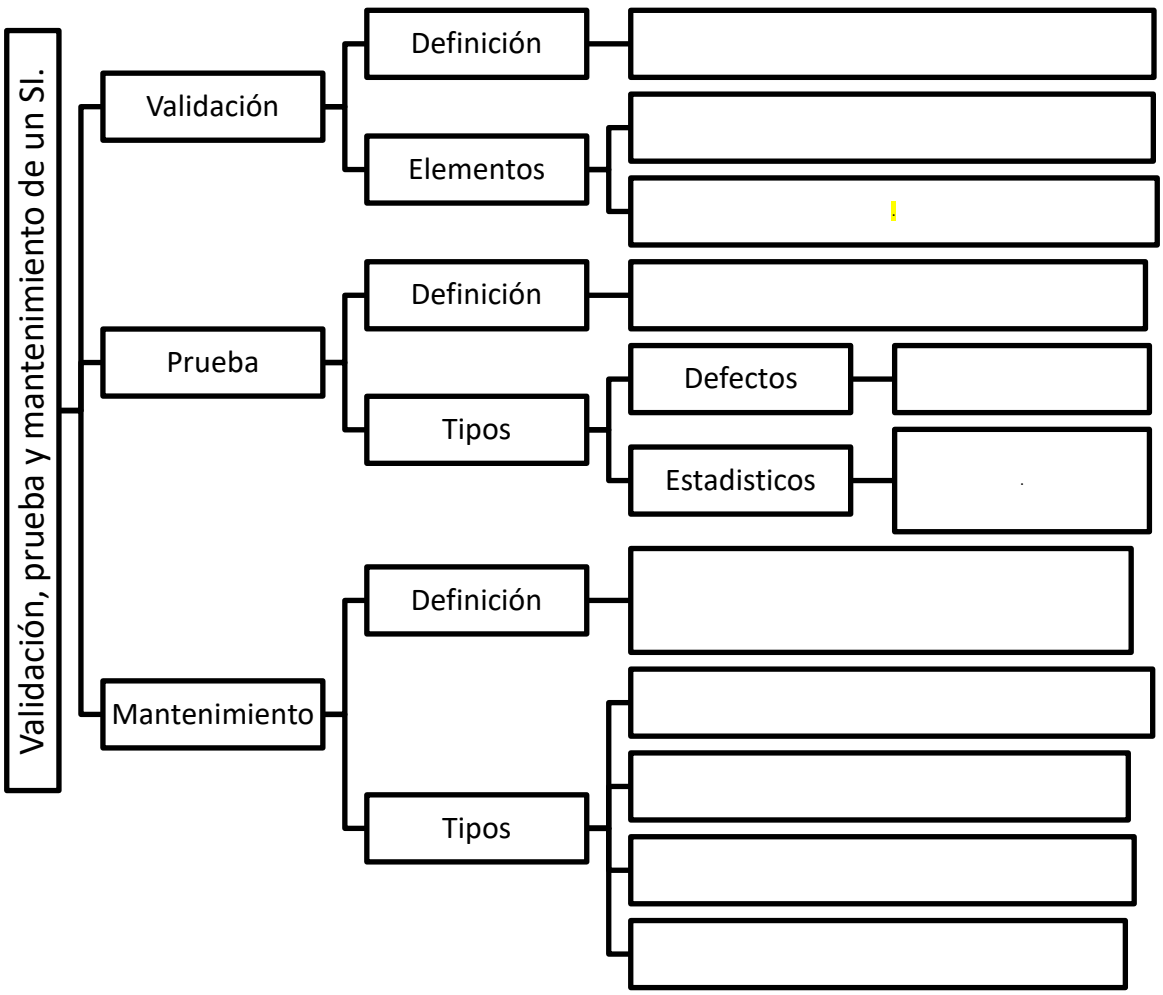
Posteriormente, según se trate de un mantenimiento correctivo o evolutivo, se verifica y reproduce el problema, o se estudia la viabilidad del cambio propuesto por el usuario. En ambos casos se estudia el alcance de la modificación. Hay que analizar las alternativas de solución identificando, según el tipo de mantenimiento de que se trate, cuál es la más adecuada. El plazo y urgencia de la solución a la petición se establece de acuerdo con el estudio anterior.

Recurso Didáctico Sugerido

https://www.youtube.com/watch?v=U3RWZi2CwQo

Actividad 7. Cuadro sinóptico "Validación y pruebas de un SI"

Instrucciones: Formar equipos de 5 integrantes para realizar un análisis reflexivo de los conceptos: Validación, Prueba y Mantenimiento que están en la lectura anterior, extrae ideas principales, y termina de rellenar el siguiente cuadro sinóptico incompleto, utilizando una aplicación de diseño (Canva, Geneally, PowerPoint) o en tu libreta.



Instrumento de Evaluación LISTA DE COTEJO Actividad 9 “Cuadro sinóptico,” Validación y prueba de un SI”					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s):**		
Producto: Cuadro sinóptico			**Fecha:**		
Materia:			**Periodo:**		
Nombre del Docente:			**Firma del Docente:**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	El cuadro sinóptico contiene el nombre del tema.				
2	Incluye los tres conceptos que se le solicitan.				
1	Utiliza las llaves o corchetes en su diseño.				
2	Sintetiza adecuadamente el tema propuesto.				
2	Presentación sobresaliente y creativa. Uso correcto de los colores en la fuente y estilo de fuentes. El texto es legible.				
1	No presenta errores ortográficos				
1	Entrega en tiempo y forma.				
	CALIFICACIÓN				

Referencias

Drake J.M, López P. (2009) Ingeniería de Software. Verificación y Validación. https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf

Ecosistema de Recursos Educativos Digitales SENA. Pruebas de validación de bases de datos: introducción. [Video]. YouTube. https://www.youtube.com/watch?v=U3RWZi2CwQo

Mi circunstancia digital.(2020). Mantenimiento de Sistemas de Información (MSI). https://manuel.cillero.es/doc/metodologia/metrica-3/procesos-principales/msi/

Quality (2020). Guía definitive de la validación de sistemas informáticos. https://www.ambit-bst.com/blog/gu%C3%ADa-definitiva-de-la-validaci%C3%B3n-de-sistemas-inform%C3%A1ticos

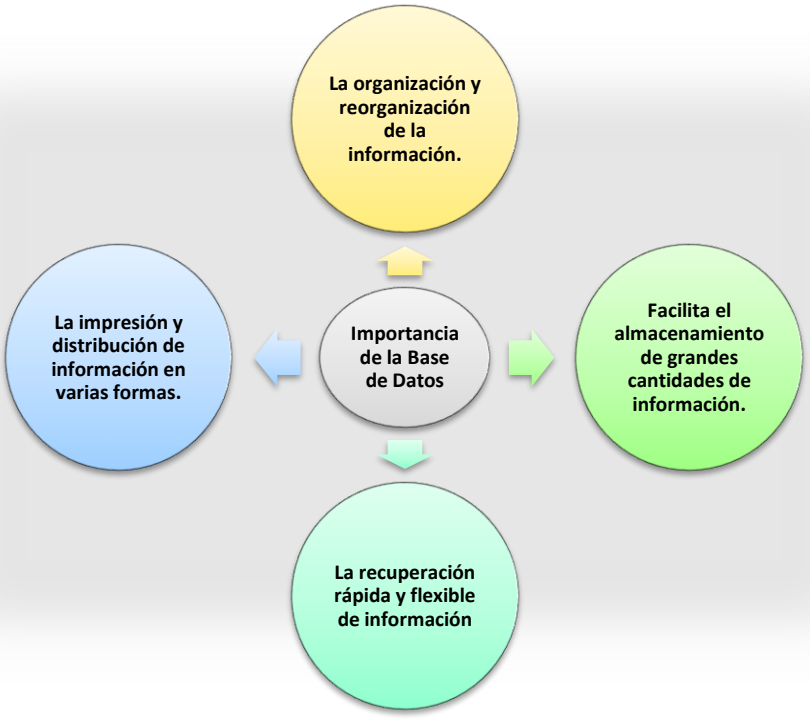
Lectura 7. Base de Datos: Tablas y Relaciones en Access

Instrucciones: *Realiza el análisis de la siguiente lectura y posteriormente realiza la practica guiado número 1.*

¿QUÉ ES UNA BASE DE DATOS?

Es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema gestor de base de datos (DBMS).

Las Bases de Datos permiten almacenar grandes volúmenes de datos, los cuales ayudan a la toma de decisiones en las empresas.



Sistemas de administración de Base de Datos

Un sistema de administración de bases de datos, DBMS por sus siglas en inglés (*Database Management System*) es un software que se usa para definir, manipular, recuperar, almacenar y gestionar datos en bases de datos.

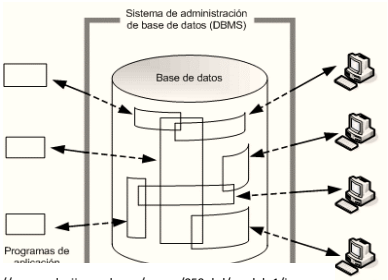
En resumen, los sistemas de bases de datos se encargan de:

- Definir reglas para validar y manipular datos.
- Interactuar con bases de datos, aplicaciones y usuarios finales.
- Recuperar, almacenar y analizar datos.
- Actualizar datos.

Los usuarios pueden ser:

Normales o finales, son las personas que pueden acceder a ciertos datos desde:

- ***Programadores de aplicación,*** son profesionales informáticos que escriben programas de aplicación para satisfacer las necesidades de la empresa
- ***Administrador de la base de datos***, es la persona que tiene el control central sobre el sistema de base de datos.



https://cursos.clavijero.edu.mx/cursos/059_bd/modulo1/imagenes/tema1.1_clip_image001.gif

COMPARACIÓN DE SISTEMAS DE ADMINISTRACIÓN DE BASES DE DATOS

DBMS	Tipo	Sistemas operativos
MySQL	RDBMS	Canonical, FreeBSD, Linux, MacOS, Solaris y Windows
MariaDB	RDBMS	Linux, MacOS y Windows
Microsoft SQL Server	RDBMS	Linux y Windows
Oracle DBMS	Sistema de administración de bases de datos multi-modelo	AIX, BS2000, HP-UX, Linux, MacOS y Windows
PostgreSQL	RDBMS	FreeBSD, Linux, MacOS, OpenBSD y Windows
MongoDB	Base de datos documentales	FreeBSD, Linux, MacOS y Windows
Redis	Base de datos clave-valor	Unix-like
IBM DB2	RDBMS	Linux, Unix-like y Windows
Elasticsearch	Búsqueda e index	Linux, MacOS y Windows
SQLite	RDBMS	Android, BSD, iOS, Linux, MacOS, Solaris, VxWorks y Windows

Según el ranking de DB-Engines, a fecha de abril de 2023*, estos son los top 10 sistemas de administración de bases de datos:

1. Oracle
2. MySQL
3. Microsoft SQL Server
4. PostgreSQL
5. MongoDB
6. Redis
7. IBM DB2
8. Elasticsearch
9. SQLite
10. Microsoft Access

Tablas de Base de Datos.

Las tablas son uno de los componentes fundamentales de una base de datos y permiten almacenar grandes cantidades de información de manera organizada y eficiente, están compuestas por campos y registros, en donde:

Campo: Se refiere al nombre de la columna.

Registro: Se refiere a cada fila que conforma la tabla, dicho de otra manera, son los datos y registros que almacenamos. Cabe aclarar que en ocasiones pueden quedar datos nulos. (agregar nombre de la tabla y marcar de forma gráfica cual es el campo y el registro)

Ejemplo

Nombre de la tabla

Registro

Cliente			
Nombre	**Apellido**	**Edad**	**Teléfono**
Juan	Domínguez	55	923 157 8022
Daniel	Ramírez	22	917 456 3287

Campo

Diccionario de Datos

Enlista de manera organizada los nombres, definiciones y características de cada uno de los campos o atributos de una base de datos y/o conjunto de datos.

Tiene por objetivo proveer un lenguaje común entre el autor de dichos datos y sus posibles usuarios. Nos permiten entender e interpretar un conjunto de datos o base de datos al proporcionar información básica sobre los campos o variables que contiene. Brindan la siguiente información:

- Qué significa cada campo o variable.
- Qué tipo de datos contiene.

- Qué valores puede tomar, o si usa algún catálogo.
- Si contiene información pública, confidencial, o reservada.

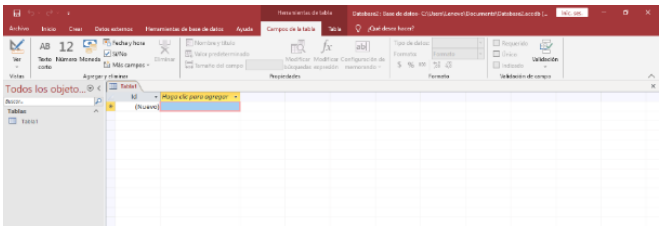
Ejemplo de Diccionario de Datos

Cliente				
Nombre	Campo	Tipo	Tamaño	Descripción
Código de empleado	cdcl	Numérico	4	Almacena código de cliente
Nombre	nbcl	Texto	30	Almacena nombre del cliente
Apellidos	apcl	Texto	30	Almacena apellido del cliente
Teléfono	tlcl	Numérico	10	Almacena el teléfono del cliente
Correo	cocl	Hipervínculo	30	Almacena el correo del cliente
Dirección	dicl	Memo	50	Almacena la dirección del cliente

Conociendo Access.

1. Dar clic en el ícono de Microsoft Access.
2. Seleccionar la opción “Base de datos del escritorio en blanco”. Aparece esta pantalla, en la que escribiremos el nombre de nuestra base de datos y presionar el botón “Crear”.

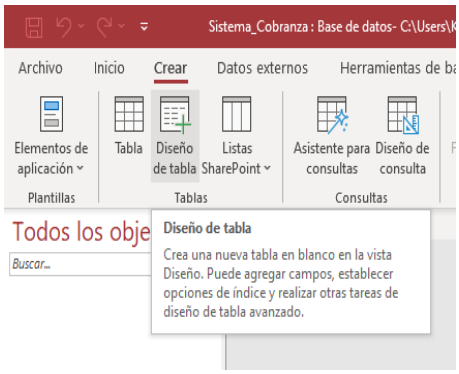
Automáticamente se creará la nueva base de datos a la cual Access asignará la extensión .ACCDB



Para crear una tabla en una base de datos existente.

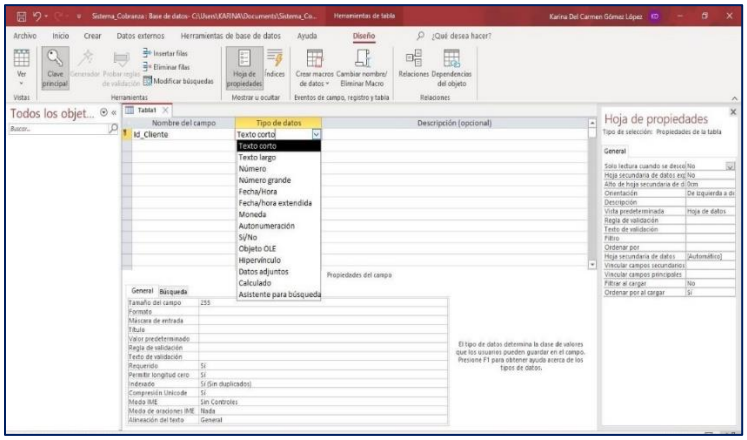
1. Haga clic en la pestaña Crear en el grupo de Tablas y haga clic en Tabla.
2. Se inserta una tabla nueva en la base de datos y se abre la tabla en la vista Hoja de datos.

- **El botón Tabla** abre la Vista Hoja de datos, consiste en introducir directamente los datos en tabla.
- **Vista diseño:** En él puedes agregar campos, establecer opciones de índice y realizar otras tareas de diseño de tabla avanzado.
- **Listas SharePoint** Esta opción se escoge cuando queremos configurar una tabla de Access predeterminada que encontramos dentro de las diferentes plantillas de diseño



Inicia la creación de las tablas con la opción **Diseño de Tablas**, para ir colocando el nombre del campo y el tipo de datos.

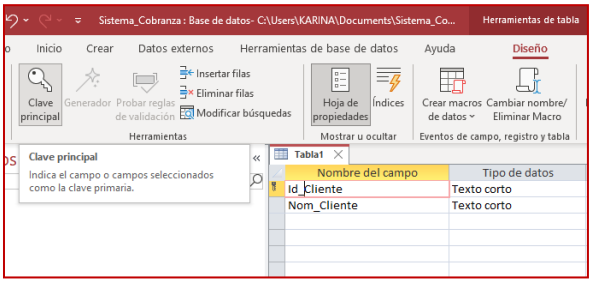
Al presionar en la pestaña **Inicio>Vistas>Ver >vista diseño** de tabla nos aparece la siguiente pantalla:



Tipos de datos. El tipo de datos de un campo indica el tipo de datos que almacena el campo, como una gran cantidad de texto o archivos adjuntos. (Microsoft, 2021).

Los campos que forman parte de una relación de tabla se denominan claves. Existen dos tipos de claves:

Para asignar una clave principal a un campo, seguir los siguientes pasos:

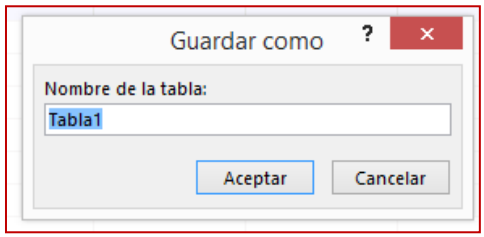


1. Hacer clic sobre el nombre del campo que será clave principal.
2. En la pestaña Diseño de Herramientas de tabla, hacer clic sobre el botón Clave principal del grupo Herramientas.
3. A la izquierda del nombre del campo aparecerá una llave indicándonos que dicho campo es la clave principal de la tabla.
4. Si quieres que el sistema se encargue automáticamente de generar los valores del campo que es clave principal, puedes definirlo con el tipo de datos Auto numeración.

5. Si quieres definir una clave principal compuesta (basada en varios campos), seleccionar loscampos pulsando simultáneamente la tecla CTRL y el campo a seleccionar y una vez seleccionados todos los campos hacer clic en el botón Clave principal, que acabamos de ver.

Para guardar una tabla, podemos:

1. Pulsar en el botón Archivo y elegir la opción Guardar o bien hacer clic sobre el botón Guardar de la barra de Acceso Rápido.
2. Como nuestra tabla aún no tiene nombre asignado, aparecerá el siguiente cuadro de diálogo, escribir el nombre de la tabla y hacer clic sobre el botón Aceptar.



Práctica guiada No. 1 “Tablas en Access, todo cabe en una tabla sabiéndolo acomodar”

Sistema de Facturación “Mi tiendita”.

Creación de la Base de datos.

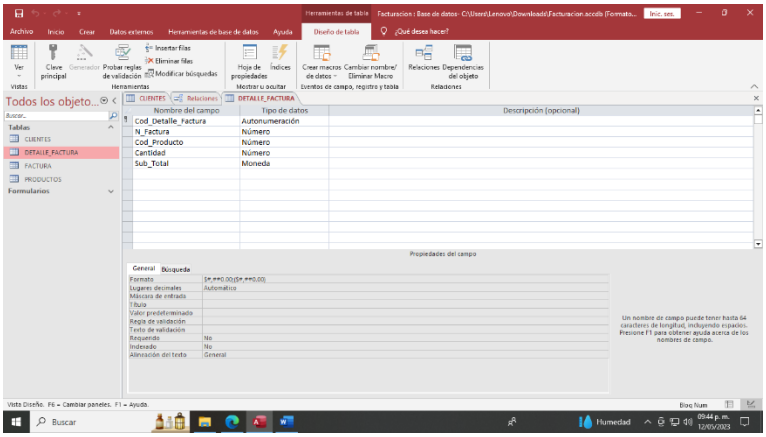
1. Ingresa a Access y crea una base de datos en blanco con el nombre **FACTURACIÓN.**

2. A continuación, se insertará automáticamente una nueva tabla vacía y guárdala con el nombre de **CLIENTE.**

3. Ingresa cada uno de los campos de la **Tabla CLIENTE** con sus propiedades.

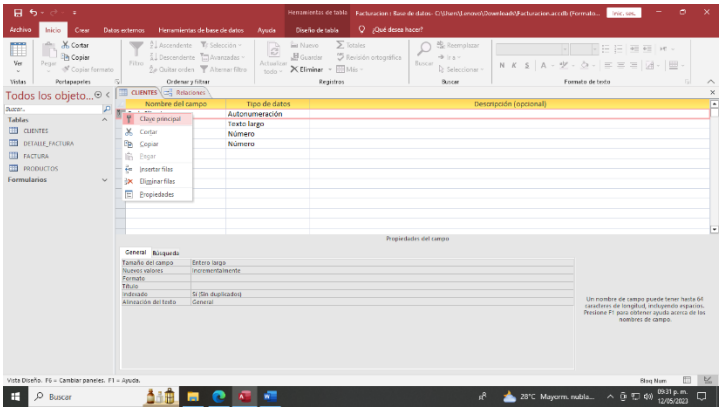
Campo	Tipo	Tamaño	Descripción
Cod_Cliente	Autonumeración	Entero largo	
Nombre	Texto largo	30	NOMBRE DEL CLIENTE
Edad	Numero	2	EDAD DEL CLIENTE
N_Doc_Identidad	Numero	Entero largo	DOCUMENTO DE IDENTIDAD

Así se vera la pantalla al ingresar todos los campos de la tabla CLIENTE.

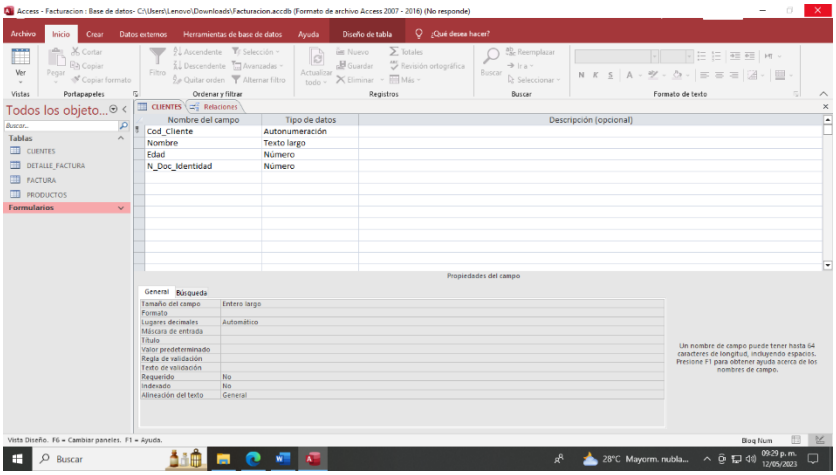


- Antes de guardar la tabla se debe asignar una clave principal
- Al crear la tabla el primercampo se considera como clave principal, para asignar otro campo como clave principal se debe seleccionar el campo que se desea que sea, darle clic derecho y seleccionar clave principal.

- Al finalizar se debe dar clic en la **X** (cerrar) de la ventana de la pestaña de la tabla y guardar los cambios.



De la misma forma se tienen que crear las tablas de **DETALLE_FACTURA, FACTURA Y PRODUCTOS**



repitiendo los pasos anteriores.

Tabla DETALLE_FACTURA

Campo	Tipo	Tamaño	Título
Cod_Detalle_Factura	Autonumeración	Entero largo	Código
N_Factura	Numero	Entero largo	Numero de factura
Cod_Producto	Numero	Entero Largo	Código del producto
Cantidad	Numero	Entero largo	Cantidad de producto
Sub_total	Moneda		Sub_total

Tabla FACTURA

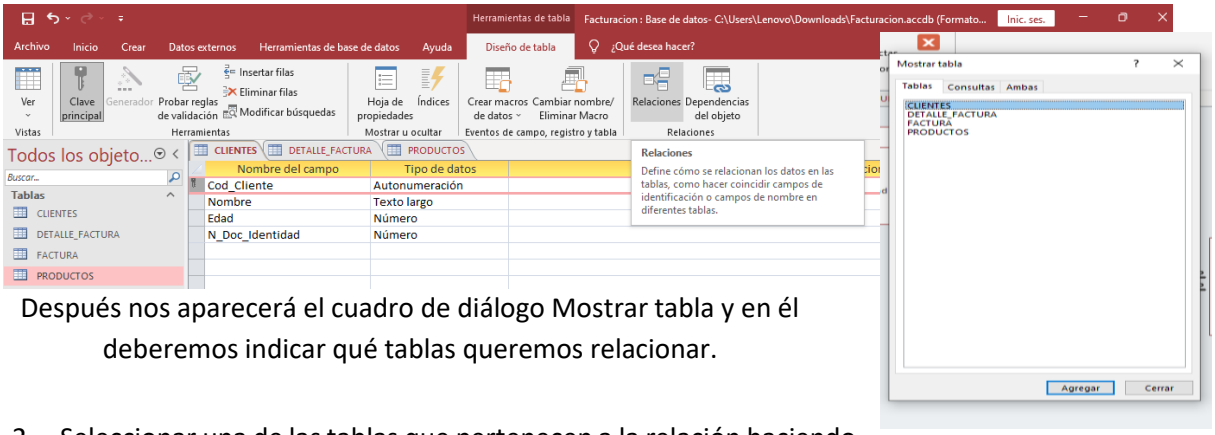
Campo	Tipo	Tamaño	Descripcion
Cod_Detall_Factura	Autonumeración	Entero largo	Codigo
N_Factura	Número	Número	Numero de Factura
Cod_Producto	Número	Número	Codigo de producto
Cantidad	Número	Número	Cantidad de producto
Sub_total	Moneda	Moneda	Subtotal

Tabla PRODUCTOS

Campo	Tipo	Tamaño	Descripcion
Cod_Producto	Número	Entero largo	Codigo de producto
Nombre	Texto corto	255	Nombre de producto
Precio	Moneda	---	Precio de producto

Establecer las relaciones entre las tablas.

1. Haz clic en la pestaña de Herramientas de base de datos> Relaciones> opción Relaciones.



Después nos aparecerá el cuadro de diálogo Mostrar tabla y en él deberemos indicar qué tablas queremos relacionar.

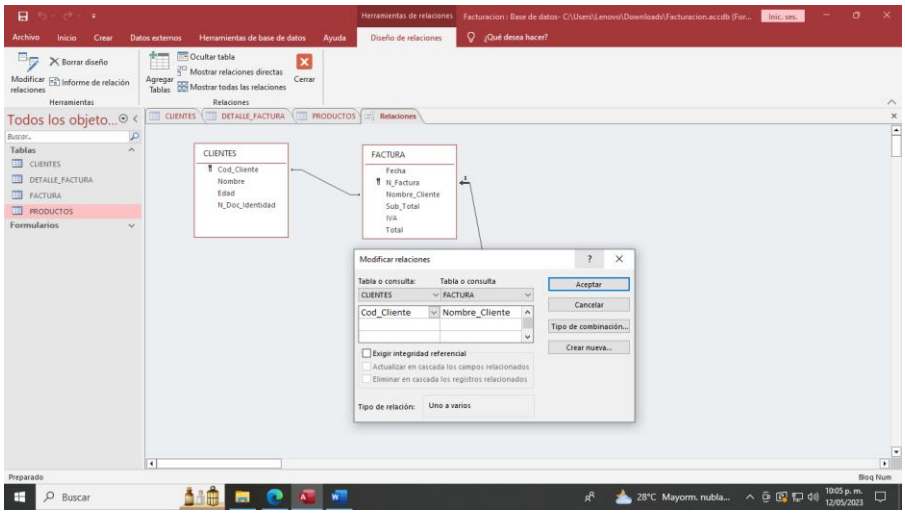
2. Seleccionar una de las tablas que pertenecen a la relación haciendo clic sobre ella, aparecerá dicha tabla remarcada. También puedes seleccionar varias a la vez pulsando CTRL.
3. Hacer clic sobre el botón Agregar.
4. Repetir los dos pasos anteriores hasta añadir todas las tablas sobre las cuales queramos efectuar relaciones.
5. Hacer clic sobre el botón Cerrar.

Para crear la relación entre la tabla **CLIENTE** y **FACTURA.**

1. Seleccionar la clave principal **Cod_Cliente** de la tabla **CLIENTE,** pulsar el botón izquierdo del mouse y manteniéndolo pulsado arrastrar hasta la llave foránea **N_Factura** de la tabla **FACTURA.**

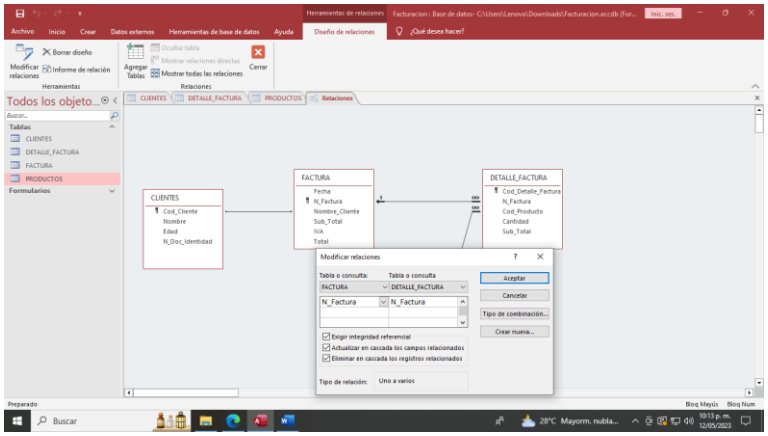
2. Soltar el botón del ratón. Aparece la siguiente pantalla y activar Exigir integridad referencialy listo.

3.

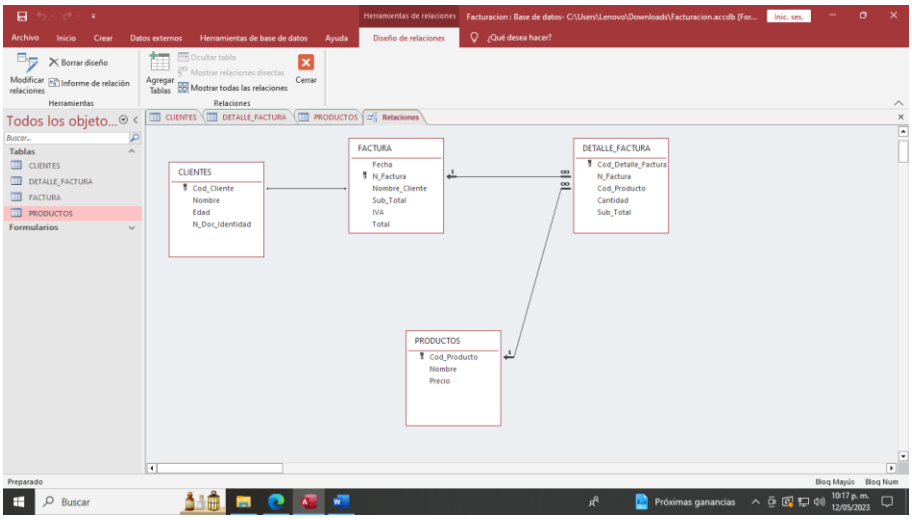


Ahora para crear la relación entre la tabla **FACTURA** y **DETALLE_FACTURA.**

1. Seleccionar la clave principal N_Factura de la tabla FACTURA, pulsar el botón izquierdo del ratón y manteniéndolo pulsado arrastrar hasta la llave foránea N_Factura de la tabla DETALLE_FACTURA.
2. Soltar el botón del ratón. Aparece la siguiente pantalla y activar Exigir integridad referencial.



Al final deben quedar así las tablas relacionadas:



Consulta el Material adicional para crear Relaciones en Access

https://youtu.be/qqjjKjTPIVA

Actividad 8. “Mapa conceptual, “todo cabe en una tabla”

Instrucciones: *De manera individual, después de realizar la lectura y con apoyo de diversas fuentes bibliográficas de tu selección,* ***elabora un mapa conceptual****, donde utilices los siguientes conceptos:*

Base de Datos	**Nivel físico**	**Registros**
Sistema Manejador de BD	**Nivel de vistas**	**ACCDB**
Administrador de la BD	**Nivel lógico**	**Microsoft Access**
Usuario Final	**Clave principal**	**Relación**
Programador de aplicaciones	**Clave foránea**	

Referencias

(s.f.). www.oracle.com/mx/database/what-is-database/

Dzul, F. E. (2020). Tecnologia de la Información y la Comunicación III. México: Klik soluciones educativas.

Accesstutoriales. (2020). Las tablas y los datos en Access. https://accesstutoriales.com/las-tablas- y-los-datos-en-access/

Aulaclic. (18 de 07 de 2020). Access 2016. https://www.aulaclic.es/access-2016/index.htm

Date, C. J. (2001). Introducción a los Sistemas de Base de Datos. PEARSON, Prentice Hall.

Ecured. (Julio de 2021). https://www.ecured.cu/Caso_de_uso

Microsoft. (14 de junio de 2021). Introducción a formularios. https://support.microsoft.com/es-es/office/introducci%C3%B3n-a-formularios-e8d47343-c937-44e8-a80f-b6a83a1fa3ae

Microsoft (2020). "Introducción a las tablas". Obtenido de: https://support.microsoft.com/es-es/office/introducci%C3%B3n-a-las-tablas-78ff21ea-2f76-4fb0-8af6-c318d1ee0ea7

Oracle. (2023). OCI. OCI: www.oracle.com/mx/database/what-is-database/

Stack Overflow contributors. (2018). Aprendizaje Python Languaje. En Aprendizaje Python Languaje (pág. 1003).

Silberschatz, A. (2002). Fundamentos de Base de Datos. España: Mc Graw Hill.

Stackscale. (18 de Abril de 2023). Retrieved 2023, from Stackscale: https://www.stackscale.com/es/blog/sistemas-administracion-bases-datos-populares/#%20%C2%BFQu%C3%A9%20Es%20Un%20Sistema%20de%20Administraci%C3%B3n%20de%20Bases%20de%20datos?

Lectura 8. Consultas en Access "¿Qué quieres buscar?"

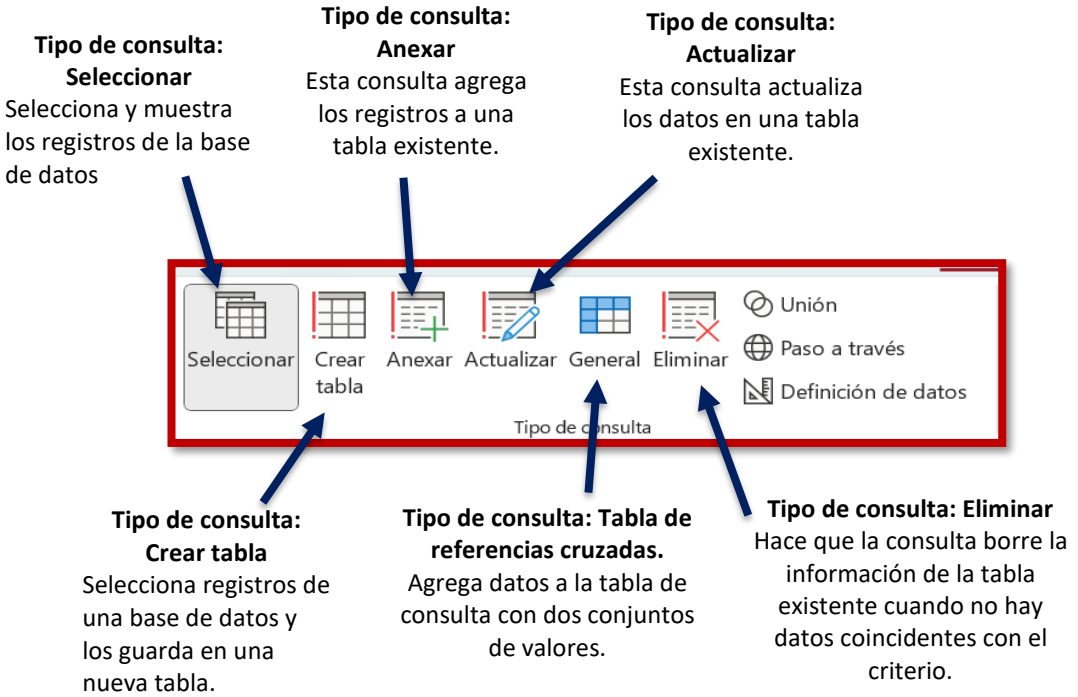
Instrucciones: Realiza la siguiente lectura e identifica los conceptos más importantes, posteriormente completa la actividad 11 y la Práctica 2.

Una consulta es una herramienta de análisis de la información, contenida en una base de datos, que te permitirá obtener los registros específicos de datos que atienden a características determinadas en la consulta. Por ejemplo, si se tiene una base de datos de alumnos con sus Matrículas, nombre, edad, semestre, promedio y dirección, puede ser de interés el promedio de los estudiantes por semestre o si se tiene una tabla de clientes con los campos nombre, dirección, teléfono, se puede generar una consulta para saber cuántos clientes son de un lugar geográfico específico; en este sentido, (Microsoft, s.f.) menciona que "las *consultas* son como las preguntas que se realizan para buscar información relacionada, incluso muy específica, en la base de datos".

En cuanto a los tipos de consulta, son mucho los existentes, dependiendo de la información buscada en la base de datos. De acuerdo con (Microsoft, s.f.) los principales tipos de consultas son:

Selección	Este tipo de consulta recupera y muestra los datos de una o más tablas, permitiendo realizar cálculos en los campos numéricos. Esta consulta no modifica el contenido de las tablas que conforman la base de datos.
Acción	Esta consulta se caracteriza por realizar modificaciones, como agregar, cambiar o eliminar datos, a las tablas que conforman la base de datos.

En la cinta de opciones de Microsoft Access para Microsoft 365 MSO se pueden encontrar ambos tipos de consulta en la cinta de opciones ***Crear/Diseño de consulta***:



CÓMO CREAR UNA CONSULTA DE SELECCIÓN EN ACCESS

En la pestaña Crear de Microsoft Access existen 2 opciones para realizar la consulta.

- ✓ ***Asistente para consulta:*** En Microsoft Access se pueden realizar consultas de selección de una manera sencilla e intuitiva, siguiendo los pasos indicados por el asistente de consulta, así como agregando los datos solicitados por este.
  Se puede crear mediante el asistente: una consulta simple, una tabla de referencias cruzadas, búsqueda de coincidencias o búsqueda de duplicados.
- ✓ ***Diseño de consulta:*** En esta opción se crea una consulta en blanco en la vista de diseño, mostrando la opción agregar tablas y seleccionar los campos que vayan a formar parte de esta.

Pasos para crear una consulta utilizando el asistente para consulta

1. En la pestaña ***crear***, haga clic en *Asistente para consultas*.
2. En el cuadro de diálogo ***Nueva consulta***, selecciona ***Asistente para consultas*** **sencillas** y **Aceptar.**
3. Selecciona en **Tabla/consultas** la tabla con la que vas a realizar la consulta.

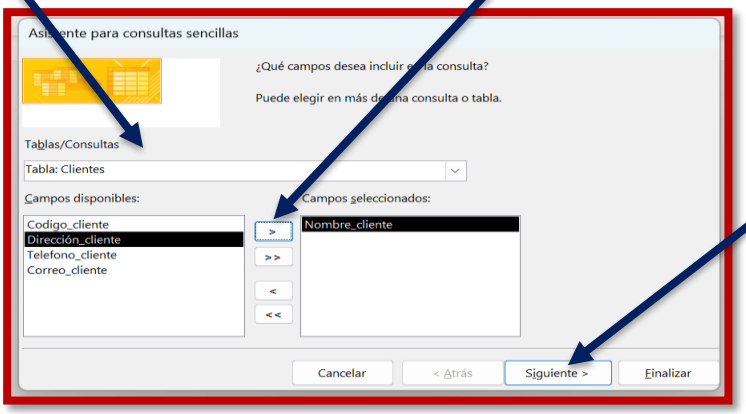
Agregue los campos a consultar. Para cada campo, realiza los pasos mostrados en la siguiente imagen: ***Para realizar la consulta de diferentes tablas, repite este proceso.***

1) Selecciona la tabla o tablas a consultar

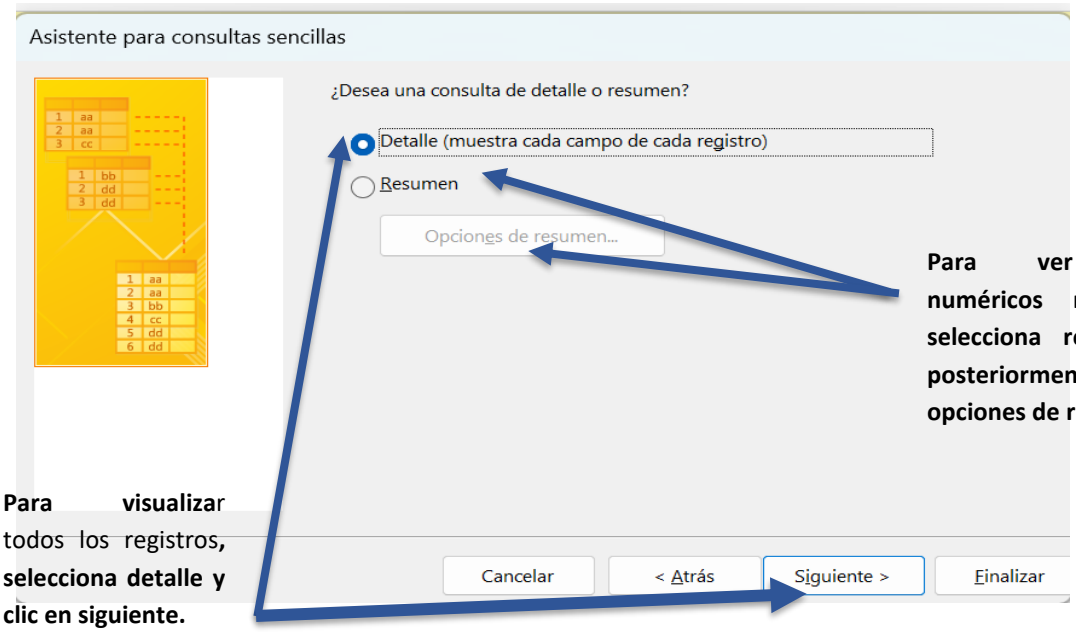
3) Selecciona el botón > para añadir los campos a la consulta.

2) Elige los campos. Aquí se muestran todos los campos de la tabla.

4) Selecciona siguiente



Nota: Si alguno de los campos seleccionados es de tipo numérico, el asistente le preguntara si desea que la consulta devuelva detalles o datos de resumen.



Para ver datos numéricos resumidos, selecciona resumen y posteriormente clic en opciones de resumen.

Para visualizar todos los registros, **selecciona detalle y clic en siguiente.**

Opciones de Resumen

El cuadro de diálogo Opciones de resumen, especifica los campos a resumir y una vez especificados selecciona Aceptar.

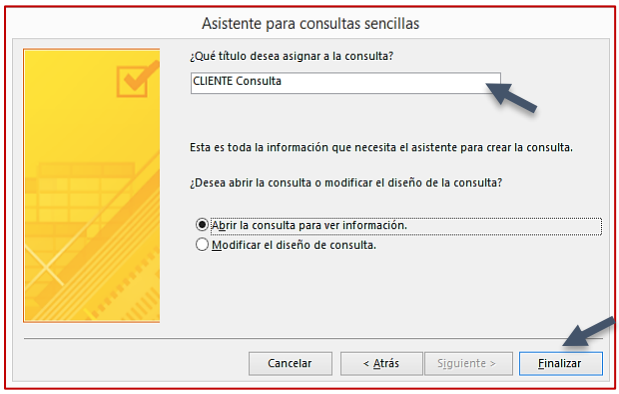
***Suma*:** Muestra la suma de todos los valores del campo.

Prom: *Da el* promedio de todos los valores del campo.

***Mín*:** Da el valor más bajo del campo.

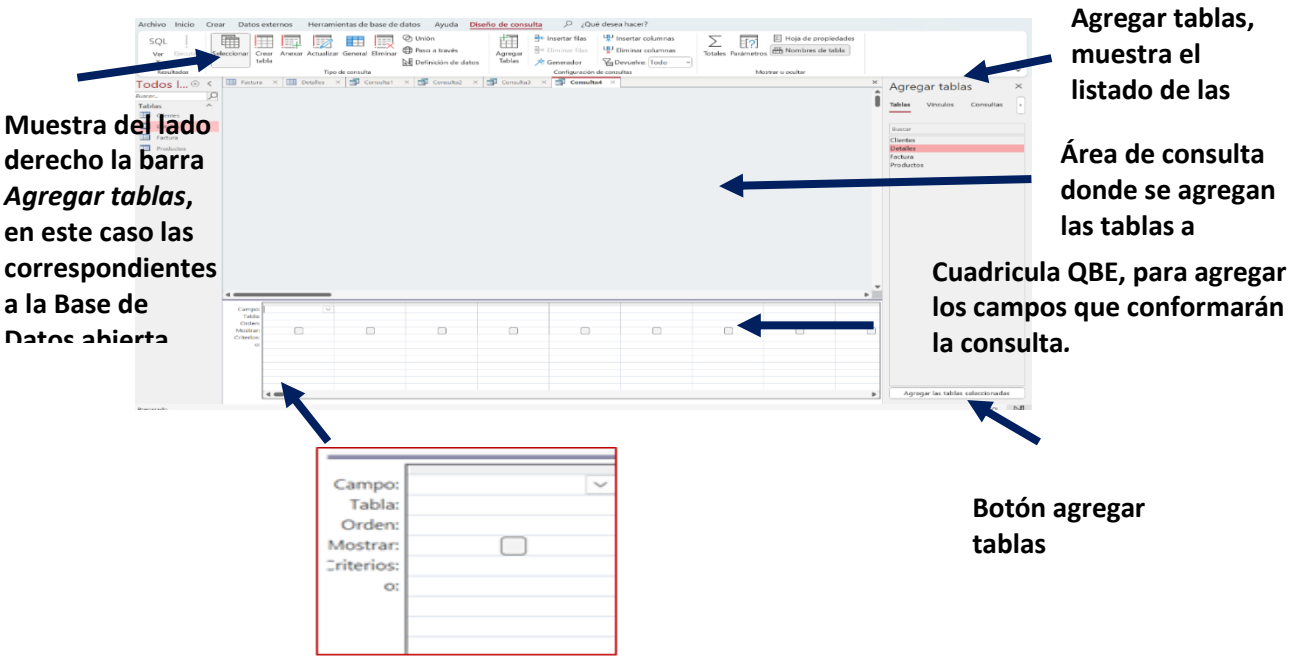
***Máx*:** Muestra el valor más alto del campo

4. Para terminar la consulta, escribe el nombre y selecciona finalizar.



Crear una consulta utilizando Diseño de consulta.

La opción Diseño de consulta crea, tanto consultas de selección, como consultas de acción. A continuación, se presenta la **interfaz de la opción Seleccionar**, para realizar una consulta de selección.



- ***Campo***: Nombre del campo a visualizar en la consulta.
- ***Tabla***: Nombre de la tabla que se consultará. Se puede elegir una de las tablas agregadas a la consulta.
- ***Orden***: Ordena los registros, de la consulta, de manera ascendente o descendente.
- ***Mostrar***: Si la casilla de verificación aparece desactivada el campo y sus registros no se mostrarán

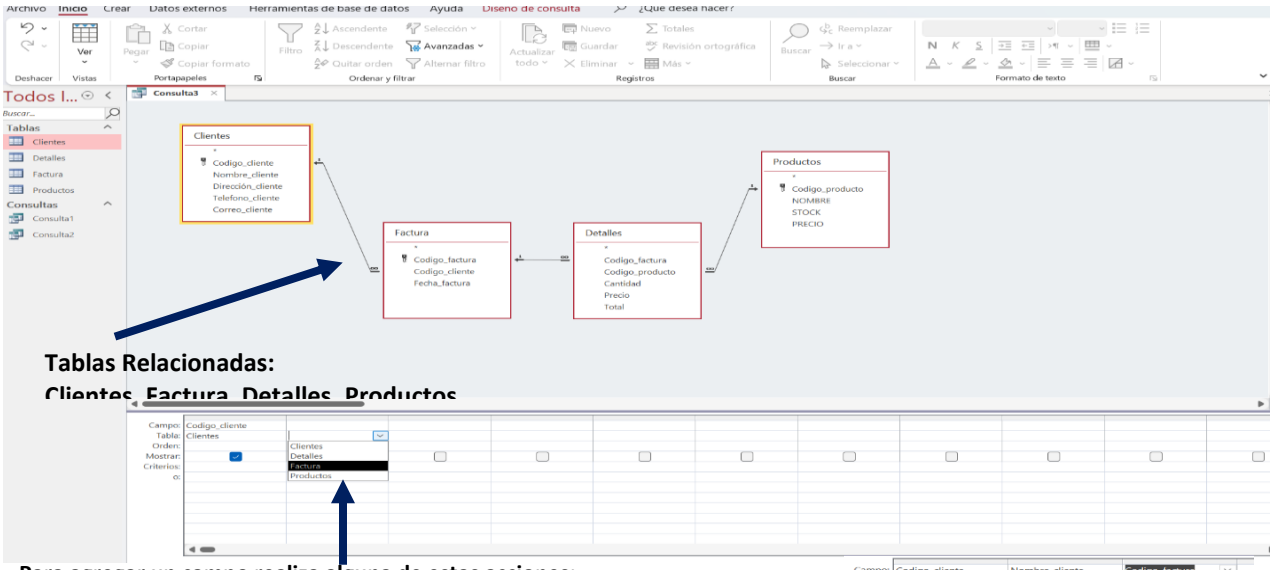
en la consulta, aunque formen parte de esta.

- ***Criterios***: Sirve para especificar un criterio de búsqueda. Un criterio de búsqueda es una condición que deben cumplir los registros que se mostrarán en la consulta. Por lo tanto, está formado por una condición o varias condiciones unidas por los operadores **Y(AND)** y **O(OR).**
- ***o***: Esta opción se utiliza para combinar condiciones.

Nota: Para que se puedan habilitar las consultas debe existir una base datos abierta.

Pasos para crear una consulta utilizando Diseño de consulta.

1. En la **cinta de opciones** selecciona la **pestaña *crear y*** *elige* ***Diseño de consulta***.
2. A continuación, elige la opción Seleccionar. Aparecerá la interfaz de la opción seleccionar mostrada previamente.
3. En el panel derecho, en agregar tablas, arrastra las tablas hacia el lado izquierdo o Selecciona la tabla y haz clic sobre el botón **Agrega**r. Repetir este proceso para agregar las tablas restantes.
4. Una vez agregadas las tablas, estas se muestran en el área de consulta, mostrado sus relaciones, en caso estar relacionadas. El panel derecho se puede cerrar en caso de haber agregado todas las tablas a la consulta.
5. Selecciona en la matriz QBE los campos que conformarán la consulta.



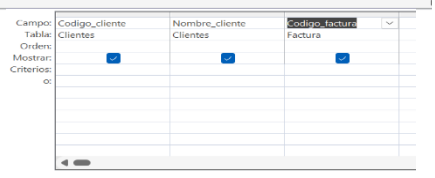
Tablas Relacionadas:
Clientes, Factura, Detalles, Productos

Para agregar un campo realiza alguna de estas acciones:

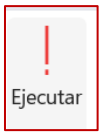
A. Elige directamente en la lista desplegable de tablas y campo en la cuadricula QBE.

B. Arrastra el campo deseado hacia la columna vacía en QBE.

C. Haciendo doble clic sobre el campo en la tabla.

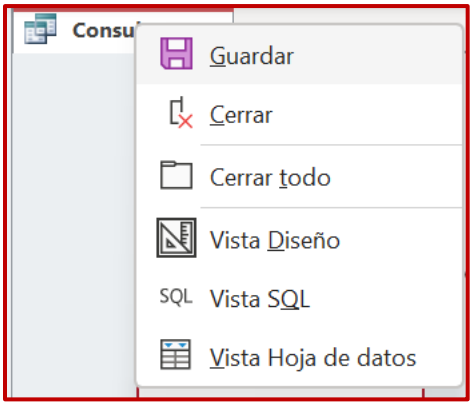


6. Para ver la consulta haz clic en el botón ejecutar o haciendo doble clic en el nombre de la consulta.



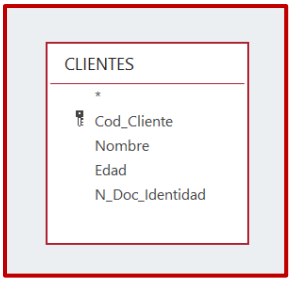
7. Para guardar la consulta haz clic derecho sobre la consulta, escribe el nombre de la consulta y selecciona guardar o en el menú Archivo/Guardar.

   Nota: No debe darse el mismo nombre de una tabla a una consulta.

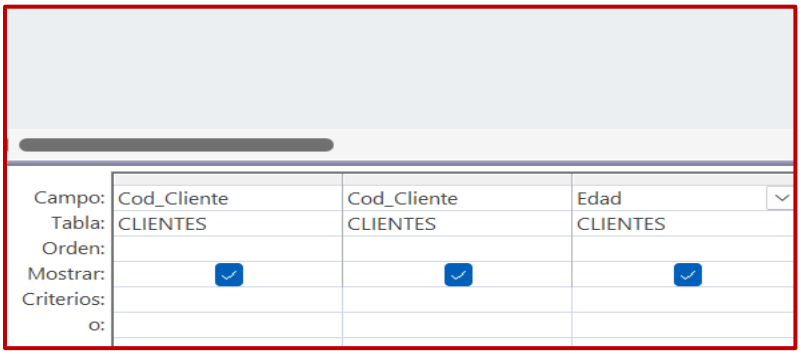


Ejemplo guiado consultas de Selección "Mi tiendita"

1. Abre la base de datos que creaste en la lectura No.7 "**Mi tiendita".**
2. Selecciona pestaña **Crear/Consulta/Diseño de consulta**.
3. En el panel derecho agregar tablas, se visualizan las tablas **CLIENTES, DETALLE DE FACTURA, FACTURA Y PRODUCTOS** arrastra la tabla CLIENTES hacia el área de consulta.



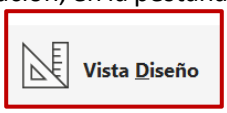
4. En la cuadricula QBE selecciona la tabla cliente y campos Cod_Cliente, Nombre y Edad.



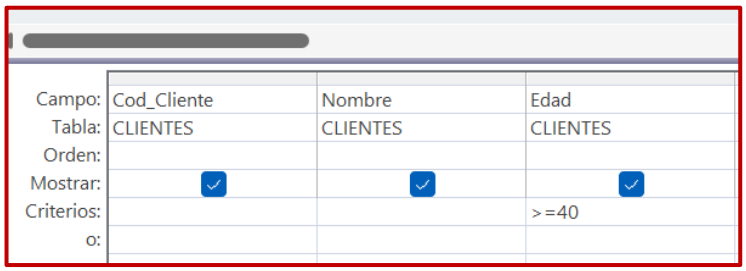
5. En la pestaña Diseño de consulta selecciona el botón ejecutar consulta. Se deben mostrar **Cod_cliente**, **Nombre** y **Edad** de todos los clientes registrados en la tabla.

Cod_Cliente	Nombre	Edad
1	Maria Laura Mejia	25
2	Luis Jose Ventura	26
3	Jorge Alejandro	12
4	Carlos Alberto Perez Sanchez	40
5	Maria Hernandez de la Cruz	30
6	Maria Hernandez de la Cruz	30
7	Luis Alberto Rojas Frias	28
8	Juan Hernandez Alvarez	45
9	Leticia Ramirez Camacho	53
10	Rosa Maria Castellanos Juarez	44
(Nuevo)		

6. A continuación, en la pestaña Inicio selecciona el botón ver vista de diseño para modificar la consulta.



7. En la cuadricula QBE ubica el campo edad y en criterio escribe **>=40** para que consulta muestre los registros de clientes con edad de 40 años o mayores.

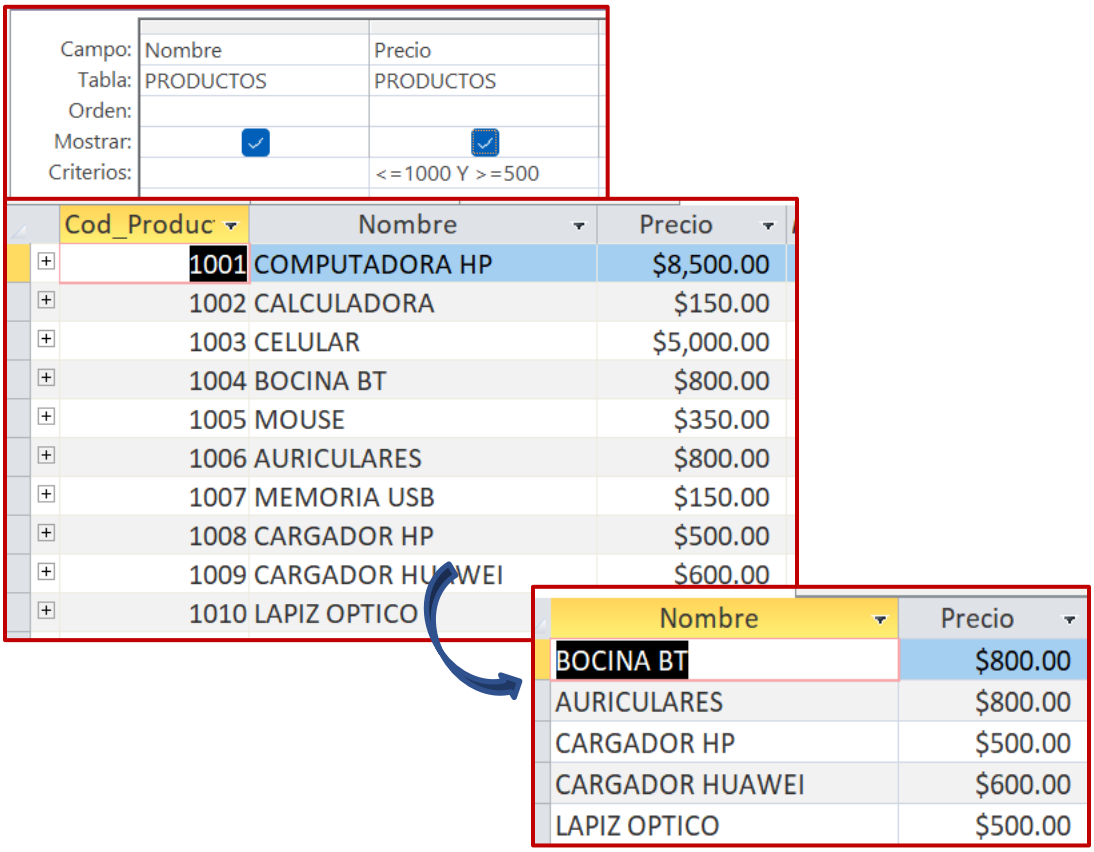


Campo:	Cod_Cliente	Nombre	Edad
Tabla:	CLIENTES	CLIENTES	CLIENTES
Orden:			
Mostrar:	☑	☑	☑
Criterios:			>=40
o:			

Cod_Cliente	Nombre	Edad
4	Carlos Alberto Perez Sanchez	40
8	Juan Hernandez Alvarez	45
9	Leticia Ramirez Camacho	53
10	Rosa Maria Castellanos Juarez	44
(Nuevo)		

8. Guarda la consulta con el nombre Consulta_Cliente.

9. Crea una nueva consulta de la tabla productos.
10. Selecciona pestaña **Crear/Consulta/Diseño de consulta**.
11. En el panel derecho agregar tablas, se visualizan las tablas **CLIENTES, DETALLE DE FACTURA, FACTURA Y PRODUCTOS** arrastra la tabla PRODUCTOS hacia el área de consulta.
12. En la cuadricula QBE selecciona la tabla productos y los campos Nombre y Precio.
13. En la columna del campo Precio, escribe el criterio >=500 y <=1000, para que se visualicen únicamente los productos con precios entre 500 y 1000 pesos.
14. Ejecuta la consulta.
15. Guarda la consulta con el nombre Consulta_Cliente.



Campo:	Nombre	Precio
Tabla:	PRODUCTOS	PRODUCTOS
Orden:		
Mostrar:	☑	☑
Criterios:		<=1000 Y >=500

Cod_Produc	Nombre	Precio
1001	COMPUTADORA HP	$8,500.00
1002	CALCULADORA	$150.00
1003	CELULAR	$5,000.00
1004	BOCINA BT	$800.00
1005	MOUSE	$350.00
1006	AURICULARES	$800.00
1007	MEMORIA USB	$150.00
1008	CARGADOR HP	$500.00
1009	CARGADOR HUAWEI	$600.00
1010	LAPIZ OPTICO	$500.00

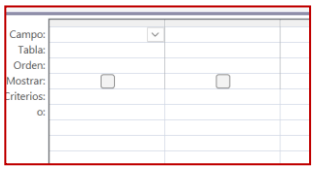
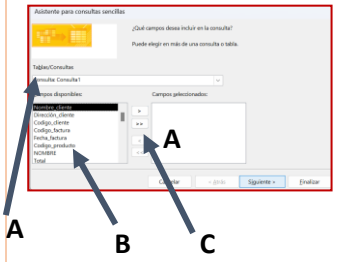
Nombre	Precio
BOCINA BT	$800.00
AURICULARES	$800.00
CARGADOR HP	$500.00
CARGADOR HUAWEI	$600.00
LAPIZ OPTICO	$500.00

Recurso Didáctico Sugerido

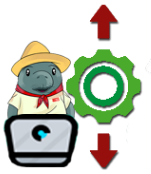
https://www.youtube.com/watch?v=SV54406GWQU&t=279s

Actividad 9. "Herramientas de consulta en Access"

Instrucciones: Después de realizar la lectura, identifica las herramientas de consulta, escribe su nombre y describe su función.

No.	Herramienta	Función
1.		
2.		
3.		
4.		
5.		A.- B.- C.-

Práctica 1. Consultas en Access “qué quieres buscar”



Propósito: Realizar las consultas, pertinentes, para visualizar la información más relevante de la base de datos del proyecto “Mi tiendita”, usando Microsoft Access.

Conocimientos:

- ✓ Consultas
- ✓ Consultas de selección.

Instrucciones: Siguiendo lo visto en la lectura y el ejemplo guiado, realiza una consulta de selección en Microsoft Access para crear:

1. Un listado de todos los clientes registrados mostrando Código de cliente, nombre y edad.
2. La consulta de productos, que muestre los campos nombre del producto y precio.
3. La consulta de facturas teniendo en cuenta la fecha y el total.
4. Para cada consulta utiliza los criterios de consulta que permitan obtener un análisis de los datos; por ejemplo, mostrar los clientes que se encuentren en un intervalo de edad especifico, los productos de mayor o menor precio y las facturas de acuerdo con un periodo de tiempo.

Instrumento de Evaluación LISTA DE COTEJO Práctica 1. Consulta en Microsoft Access "¿qué quieres buscar?"					
DATOS GENERALES					
Nombre de los estudiantes:					
Producto: Diseño de consultas "mi tiendita".			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente:			**Firma del Docente**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
2	Diseñan una consulta de clientes.				
2	Diseñan una consulta de productos.				
2	Diseñan una consulta de facturas.				
1	Utilizan los criterios de búsqueda en cada una de las consultas.				
2	Realizan las tres consultas correspondientes a la base de datos del proyecto "mi tiendita".				
1	Entregan su trabajo en tiempo y forma.				
	CALIFICACIÓN				

Referencias

[Silvherinformatica] (31 de octubre de 2020). Las consultas en Access[video]. https://www.youtube.com/watch?v=V-i5mKjeW_A&t=777s

Carlos Alberto Sotelo. (17 de septiembre de 2017). Base de datos supermercado en Microsoft Access 2010 [Video]. YouTube. https://www.youtube.com/watch?v=y36QmzsSjx4

Excel y Mas. (22 de noviembre de 2014). Consultas de selección-Access desde cero [video]. https://www.youtube.com/watch?v=SV54406GWQU&t=279s

Mario Alexander Puerres Montalván (19 de mayo de 2015). Asistente para consultas en Access [video]. https://www.youtube.com/watch?v=npd6fFouArs

Microsoft. (s.f.). Introducción a las consultas - Soporte técnico de Microsoft 365. https://support.microsoft.com/es-es/office/introducción-a-las-consultas-a9739a09-d3ff-4f36-8ac3-5760249fb65c#__toc355883436

Microsoft.(s.f.). Access (Version 2303 de 64 bits). Office 365 MSO. Microsoft.

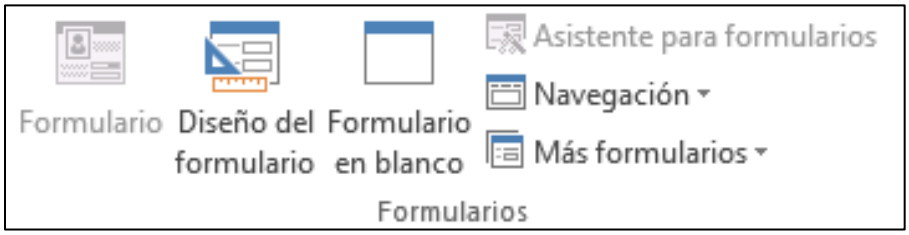
Lectura 9. Formularios e Informes en Access "Otra forma de ver los datos"

Instrucciones: Analiza la siguiente lectura y resuelve la Actividad 12.

¿Qué son los formularios?

Los formularios son herramientas que se utilizan para crear o diseñar la interfaz de un sistema, permitiendo ingresar información a la base de datos. De acuerdo con su función, permiten su edición y modificación de los datos, logrando trabajar de una forma más fácil y sencilla para el usuario.

A continuación, se desglosa el Grupo **formularios** de la pestaña **Crear** de Microsoft Access**.**



Opciones disponibles para crear un formulario:

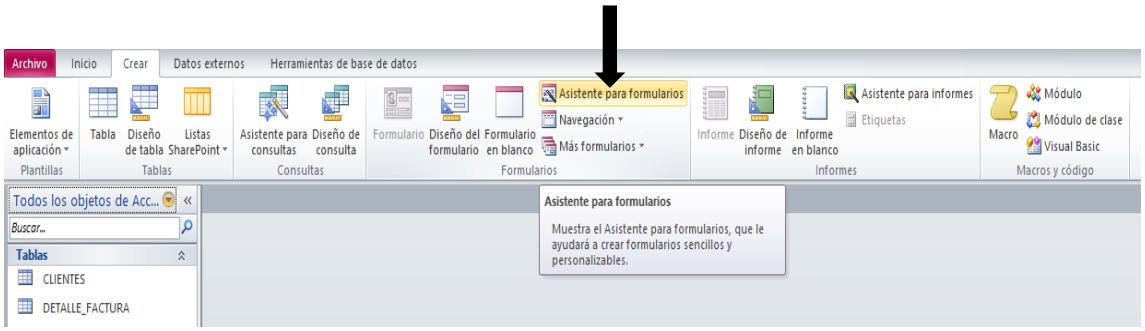
Formulario	**Crea un formulario nuevo que contiene todos los datos de la tabla o consulta seleccionada, si la tabla tiene alguna relación esta opción intentará adaptar una hoja de datos relacionados.**
Diseño del formulario	De la opción de ir insertando los objetos que quieras que aparezcan en el formulario.
Formulario en blanco	Abre la Vista Presentación, y permite insertar los datos con arrastrar la consulta o tabla.
Asistente para formularios	Te guía paso a paso durante la creación del formulario.
Navegación	Crea un formulario dedicado a la navegación web.
Más formularios	Tiene otros tipos de formularios disponibles como cuadro de diálogo modal.

Ahora que sabes qué es un formulario y las opciones para crearlo, se te guiará en la ***elaboración de tu***

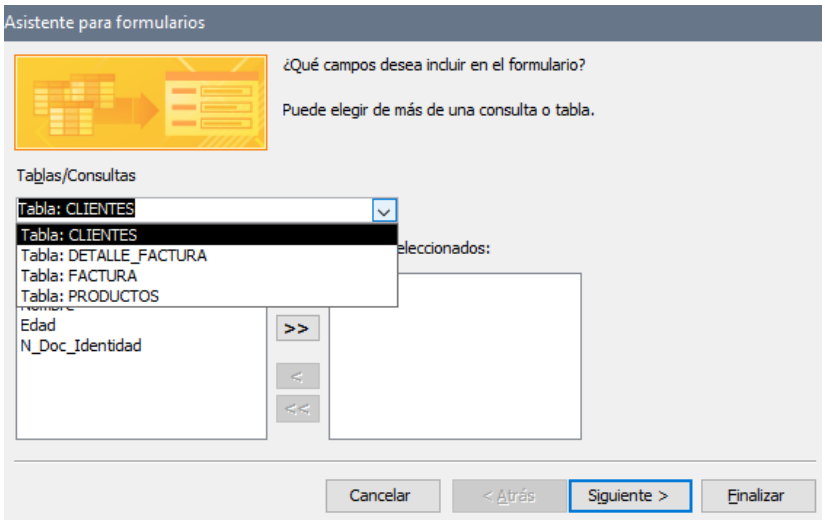
formulario a través del Asistente para formularios.

Asistente para formularios

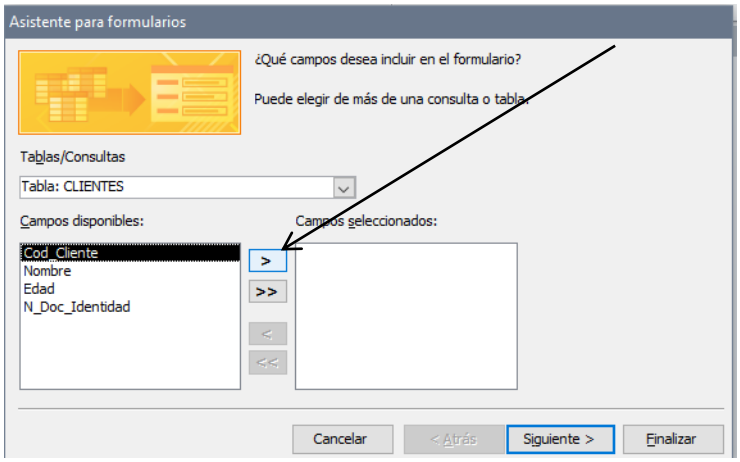
Esta es la modalidad más sencilla y dirigida de creación de formularios. El asistente se inicia desde la ficha **Crear/Grupo Formularios/Asistente para formularios**

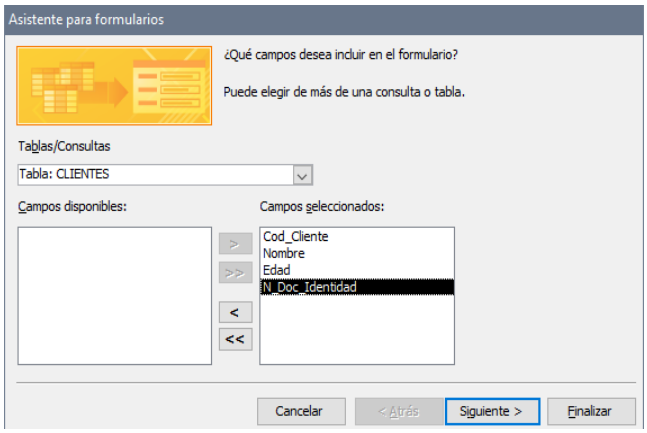


Inmediatamente aparecerá el cuadro de diálogo de abajo. Haz clic en la "**Tabla CLIENTES"** para diseñar el formulario e incluir los campos con los que se va a trabajar:

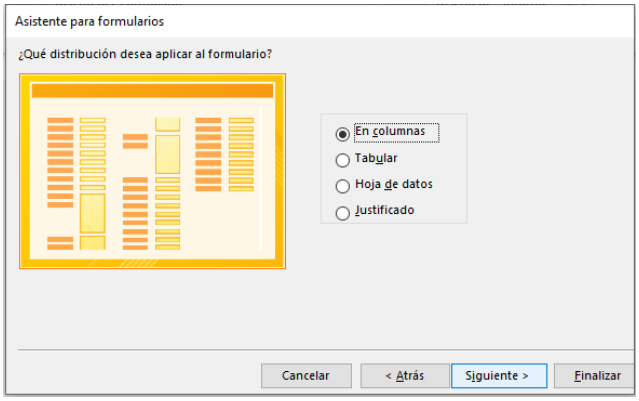


Al elegir la "**Tabla CLIENTES"** irás seleccionando los campos con los que se van a trabajar, haciendo clic en cada uno de ellos en la lista desplegable de "Campos seleccionados", de tal manera que podrás ver los campos agregados en Campos disponibles y posteriormente harás clic en "Siguiente".

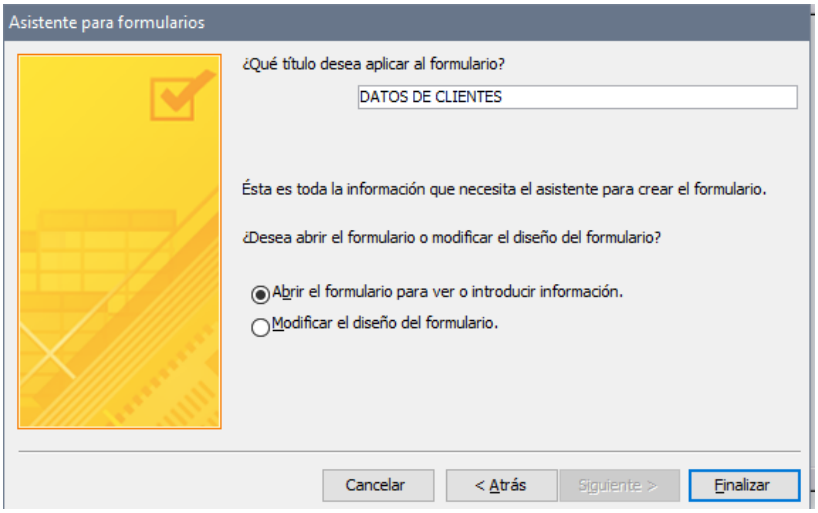




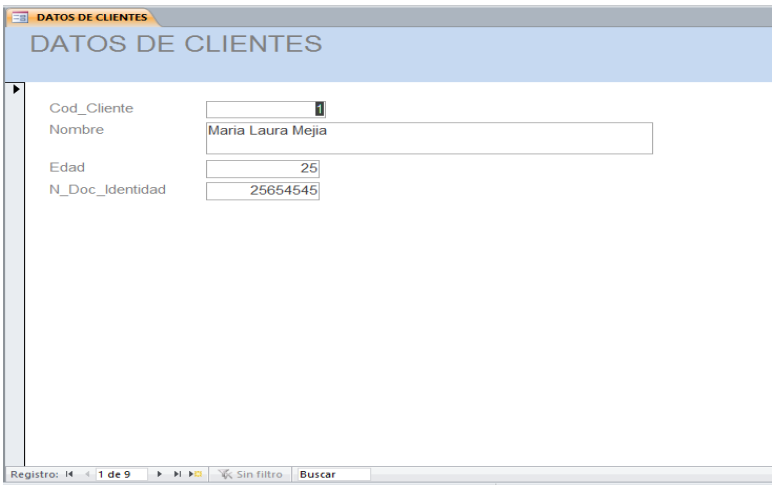
Al hacer clic en **"Siguiente"** aparecerá el cuadro de diálogo que te permitirá elegir la organización de los datos; por lo que en este caso harás clic en la opción **"En columnas"** y **"Siguiente"**. Así tus datos se verán en forma de columna, como una tabla tradicional.



El siguiente paso será colocar un título al formulario, en este caso escribirás **"FORMULARIO CLIENTES"**, debajo de la pregunta **¿Qué título deseas aplicar al formato?**, para guardar el nombre del formulario darás clic en **"Finalizar"**.



De esta manera aparecerá la ventana del formulario de clientes que diseñaste:



Podemos a continuación buscar datos, reemplazar valores, modificarlos como si estuviéramos en la vista Hoja de datos de una tabla, desplazarnos a lo largo de la tabla utilizando la barra de desplazamiento por los registros. 	La vista diseño es la que nos permite definir el formulario, en ella le indicamos a Access cómo debe presentar los datos del origen del formulario, para ello nos servimos de ciertos controles.

Trabajando con la vista diseño de formulario

El área de diseño consta de tres secciones:

- ***La sección Encabezado de formulario***, Se coloca lo que queremos que aparezca al principio del

formulario.

- ***La sección Detalle,*** en ella aparecerán los registros del origen del formulario, o varios registros o uno sólo por pantalla según el tipo de formulario. Aunque se visualicen varios registros en una pantalla, debemos indicar en la sección Detalle el diseño correspondiente a un sólo registro.
- ***La sección Pie de formulario,*** en ella colocamos lo que queremos que aparezca al final del formulario.

Podemos mostrar u ocultar el encabezado o el pie desde la opción Encabezado o Pie del formulario del menú contextual de los mismos.

A continuación, vamos a modificar el formulario de Clientes desde la pestaña o ficha "Diseño" que se presenta a continuación:

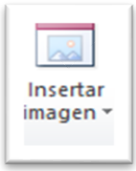
	El primer botón que vemos está localizado en el grupo Vistas, y nos permite pasar de una vista a otra: *vista formulario, vista diseño y vista presentación.*
	En el grupo Temas encontrarás herramientas para dar un estilo homogéneo al formulario.
	En la parte central puedes ver el grupo Controles en el que aparecen todos los tipos decontroles para que sea más cómodo añadirlos en el área de diseño. También encontramos algunos elementos que podemos incluir en el encabezado y pie de página.
	En el grupo Herramientas podrás encontrar el botón Agregar campos existentes entre otros, que hace aparecer y desaparecer el cuadro Lista de campos en el que aparecen todos los campos del origen de datos para que sea más cómodo añadirlos en el área de diseño.

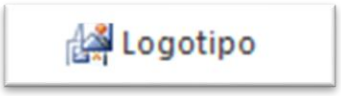
Con el botón  hacemos aparecer y desaparecer el cuadro Propiedades del control seleccionado.

Podemos realizar un diseño creativo y llamativo a nuestros formularios con la **vista diseño**. Logrando obtener una ventana como esta.

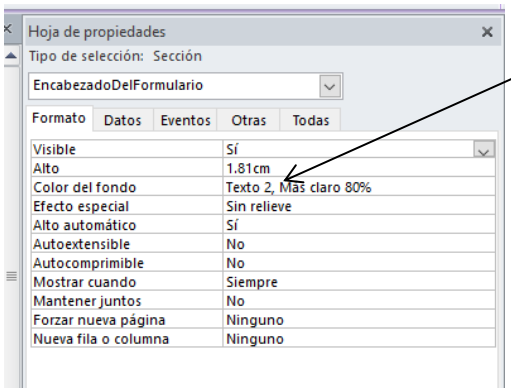
Utilicemos algunos controles:

Para insertar nuestro logotipo utilizamos Insertar imagen\l grupo de herramientas o la herramienta logotipo\Encabezado y pie de página





Para cambiar los colores de nuestro formulario, dar clic en cada sección (Encabezado, pie de página, detalle) y dar clic en la herramienta hoja de propiedades\grupo herramienta.

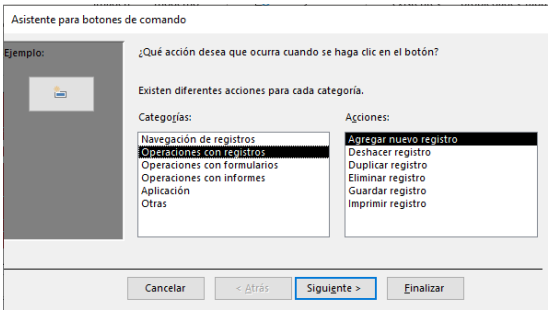
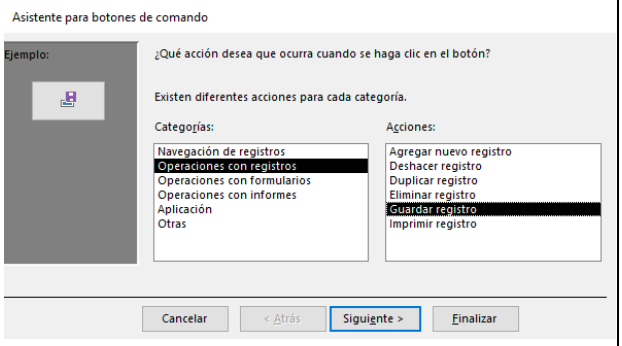
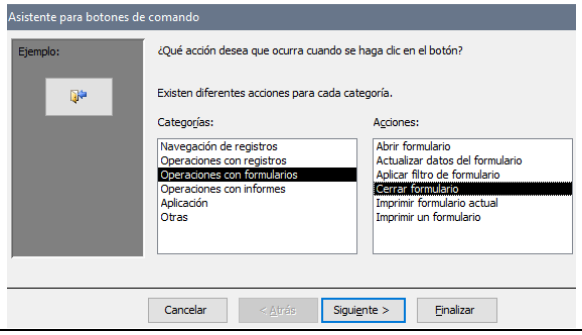


Elige el color que desees para esa sección.

Para agregar los botones de acción utilizaremos el Asistente de botones para dar de Alta, Baja, Modificar o Buscar registro y el Botón de Cerrar formulario.

Se realiza de la siguiente manera:

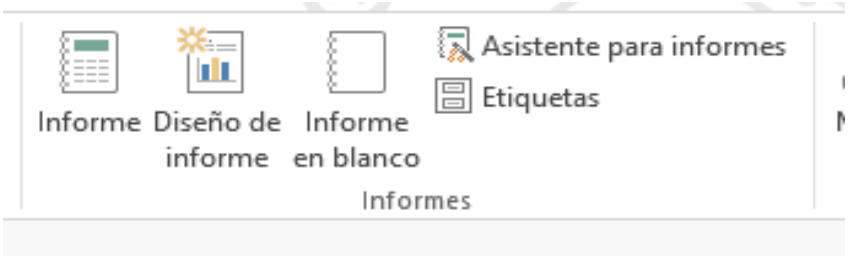
1. Colocar el formulario en la vista diseño **Vistas/vista diseño.** Deberá ampliar el formulario de tal manera que haya espacio para colocar los botones de acción.
2. Ir a la pestaña **Diseño/controles** y selecciona **Botón,** y arrastra el botón hasta el formulario, se genera el Asistente para botones de comando y prosigue como se indica:

Botón	Pasos	Imagen
Nuevo	se genera el Asistente para botones de comando, selecciona la categoría **Operaciones con Registro** y la Acción **Agregar nuevo Registro,** clic en siguiente y selecciona un texto o una imagen para el botón de acción y clic en siguiente, coloca un nombre al botón o puedes dejar el sugerido por el asistente y clic finalizar	
Guardar	selecciona la categoría **Operaciones con Registro** y la Acción **Guardar Registro**, clic en siguiente y selecciona un texto o una imagen para el botón de acción y clic en siguiente, coloca un nombre al botón o puedes dejar el sugerido por el asistente y da clic finalizar.	
Cerrar	selecciona la categoría **Operaciones con Formulario** y la Acción **Cerrar Formulario**, clic en siguiente y selecciona un texto o una imagen para el botón de acción y clic en siguiente, coloca un nombre al botón o puedes dejar el sugerido por el asistente y clic finalizar.	

¿Qué son los informes?

Los informes muestran información de una tabla o consulta contenida en una base de datos, los datos presentados en los informes de acuerdo con su función solo permiten visualizar o imprimir los datos. Una de sus ventajas es que se puede agrupar más fácilmente la información.

Para crear un informe podemos utilizar las opciones del grupo **Informes**, en la pestaña **Crear**:



- ***Informe*** consiste en crear automáticamente un nuevo informe que contiene todos los datos de la tabla o consulta seleccionada en el Panel de Navegación. Si no tenemos ninguna tabla o consulta seleccionada la opción aparece deshabilitada.

- ***Diseño* de informe** abre un informe en blanco en la vista diseño y tenemos que ir incorporando los distintos objetos que queremos aparezcan en él. Este método no se suele utilizar ya que en la mayoría de los casos es más cómodo y rápido crear un auto informe o utilizar el asistente y después sobre el resultado modificar el diseño para ajustar el informe a nuestras necesidades.

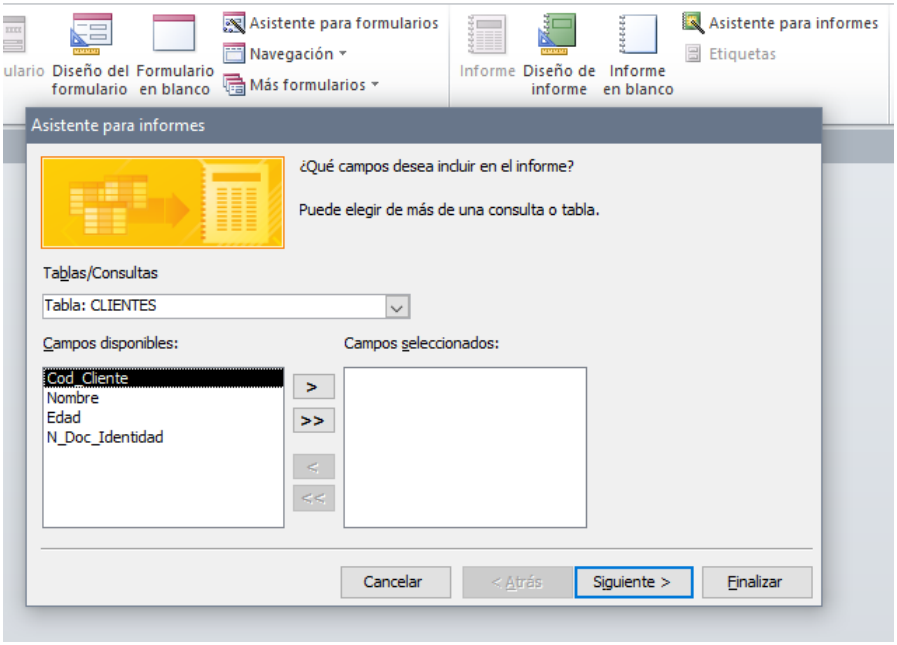
- ***Informe en blanco*** abre un informe en blanco en vista Presentación.

- ***Asistente para informes*** utiliza un asistente que nos va guiando paso por paso en la creación del informe.

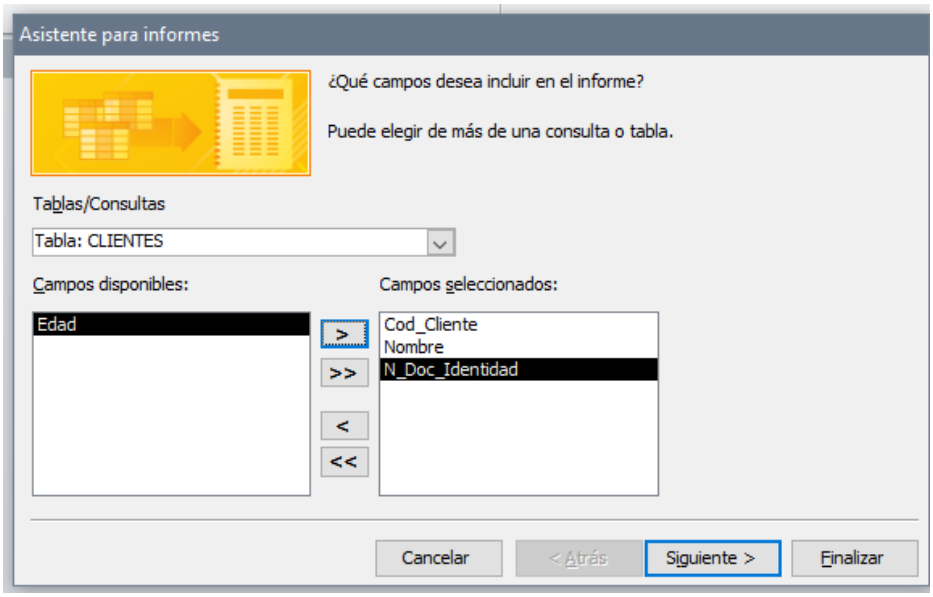
- ***Etiquetas*** utiliza un asistente que nos va guiando paso por paso en la creación del informe para imprimir etiquetas a partir de los datos de una tabla o consulta. Si no tenemos ninguna tabla o consulta seleccionada la opción aparece deshabilitada.

A continuación, vamos a ***crear un informe a partir de la tabla Clientes***, para conocer todos los clientes que fueron dados de alta al sistema. Realiza lo siguiente:

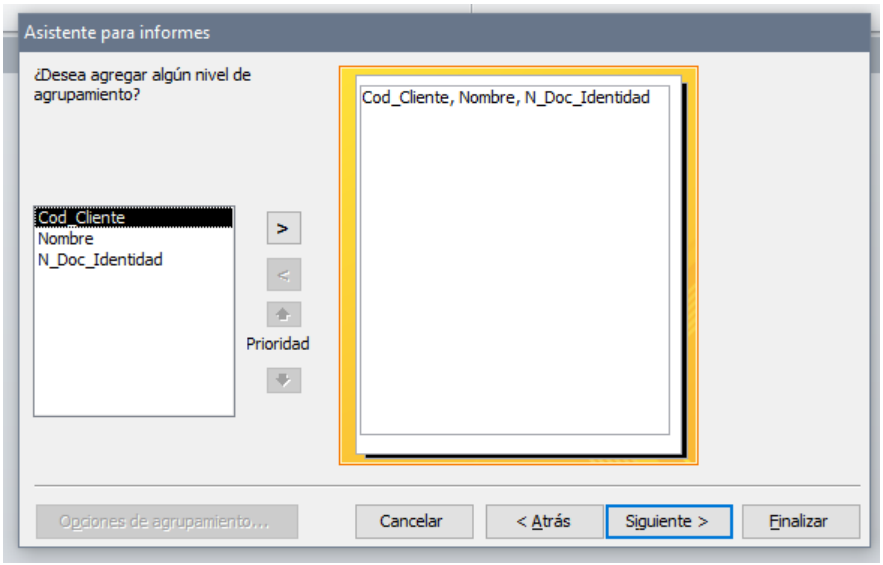
1. Ve a la ficha **Crear** y seleccionar à **asistente para informe**. Aparece el siguiente cuadro de diálogo:



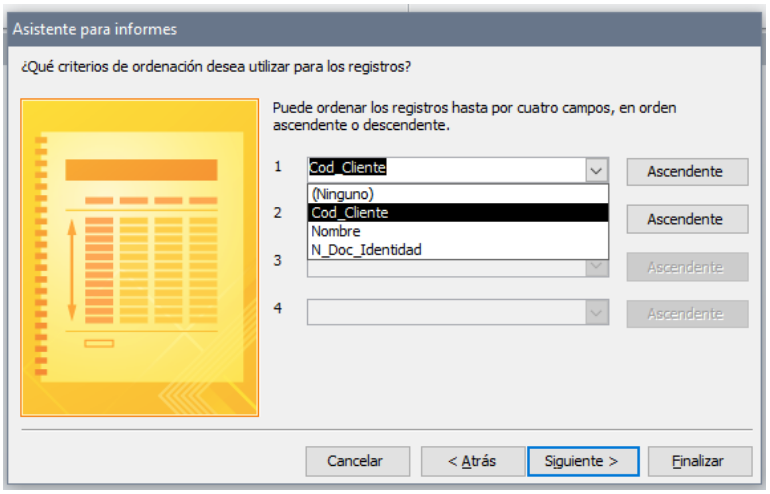
2. Selecciona la tabla/consulta a la cual le diseñará el reporte o informe. Selecciona los campos de la consulta que van a aparecer en tu informe, estos pueden ser todos o algunos campos. Posteriormente presiona el botón siguiente.



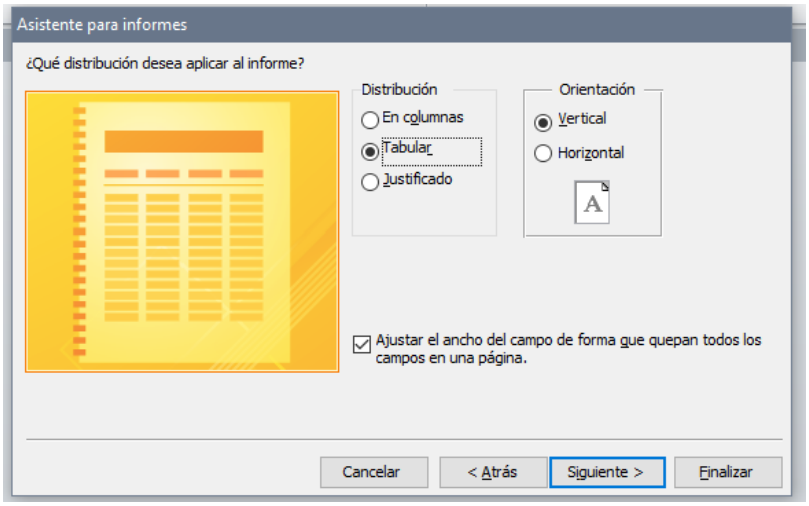
3. Definimos el nivel de agrupamiento de nuestros datos para el informe.



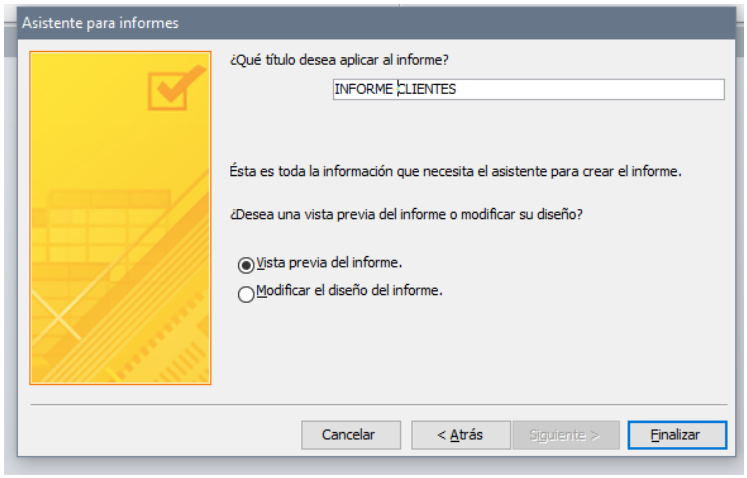
4. Este cuadro de diálogo sirve, para establecer el criterio de ordenación de los datos. Una vez establecido el orden, presiona el botón siguiente.



5. Continuamos con la distribución del informe y la orientación de la hoja

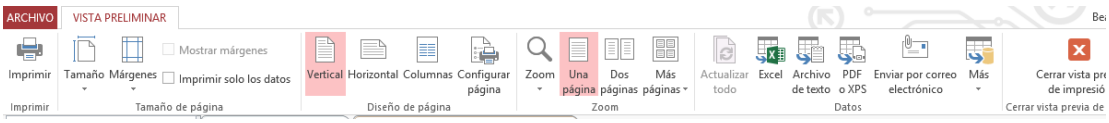


6. Aplica el título del informe, que este caso será “Informe de clientes”; activa la casilla vista previa del informe. Presiona Finalizar



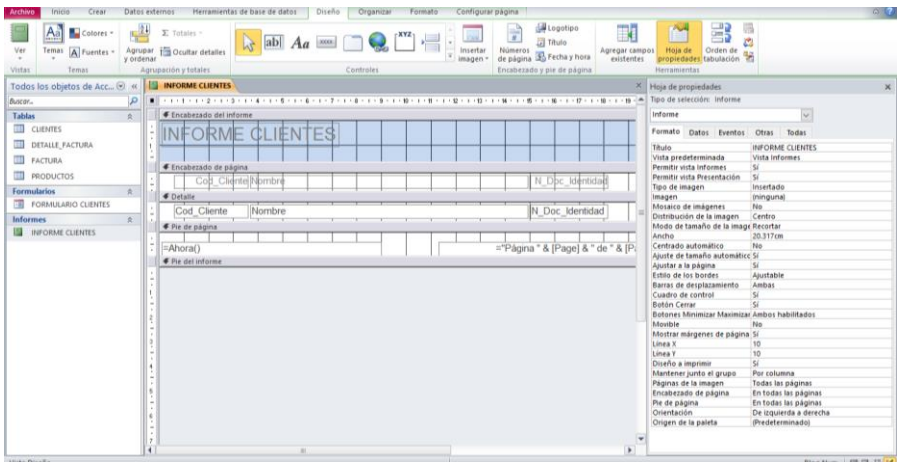
Aparece la Vista previa del Informe

En la ficha Vista preliminar, también llamada vista diseño de impresión, puedes seleccionarlas opciones para dar formato a tu informe, tal como si fuese un documento del procesador de textos. Una vez que termines, presiona el botón cerrar vista diseño de impresión.



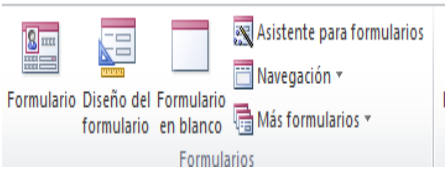
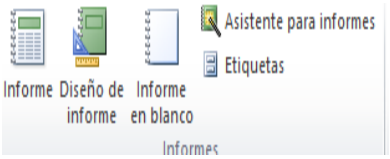
Desde la vista diseño de informe, puedes hacer las modificaciones pertinentes, así como hiciste las modificaciones a los formularios.

Tu imaginación y creatividad son el límite para que tu sistema quede genial y el diseño de losinformes sea agradable. ¡BUENA SUERTE!



Actividad 10. Tabla "Encontrando sus diferencias"

Instrucciones: En binas o de forma individual, elabora un cuadro comparativo sobre las características principales que tiene un formulario y un informe, tomando en cuenta la lectura 9.

CARACTERÍSTICAS	FORMULARIOS	INFORMES
Definición		
Función		
Botones que se utilizan para su creación		
Ejemplo (Imagen)		

Práctica 2. Formularios e Informes en Access "Otra forma de ver los datos"

Propósito: Facilitar el ingreso y la manipulación de los datos o información por medio de pantallas y reportes creativos (formularios e informes) para nuestro usuario final o responsable del sistema de gestión informático de un negocio local. Al finalizar, entrega el Archivo a tu Docente para su Evaluación y retroalimentación.

Conocimientos:

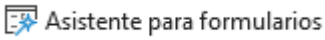
- ✓ Formularios
- ✓ Informes
- ✓ Asistente para formularios y/o informes
- ✓ Vista diseño

Formularios e Informes

Instrucciones: En equipos colaborativos y con el apoyo de la lectura 9. Realiza un formulario y un informe asignado por tu Docente, acorde a tus tablas creadas en Microsoft Access.

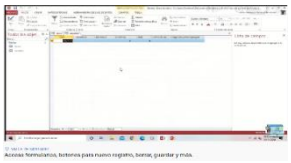
***Recordatorio**: La **vista Formulario** permite introducir los datos una vez finalizada la tabla.*

Desarrollo del formulario:

1. Crea los formularios con ayuda del 

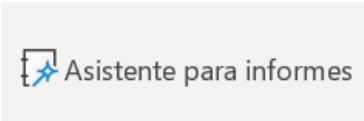
.
2. Cambia a ***Vista Diseño*** teniendo seleccionado uno de los formularios creado, para Modificarlos y/o personalizarlos con las herramientas de control de manera que se vea atractivo.
    a. Agregar un logo de tu sistema.
    b. Agregar botones de acción (Consulta, alta, modificar, eliminar registros, entre otros que consideres necesarios.).
3. Guarda los cambios generados en la ***vista diseño*** de tu **formulario**.

Recurso Didáctico Sugerido



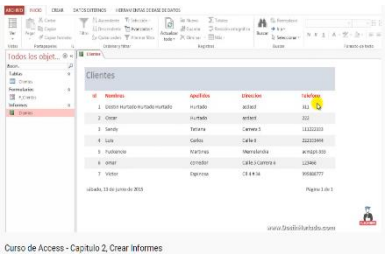
https://www.youtube.com/watch?v=elZzkzYluZE

Desarrollo del informe:

1. Crea los formularios con ayuda del 

.
2. Cambia a ***Vista Diseño*** teniendo seleccionado uno de los informes creados, para Modificarlos y/o personalizarlos con las herramientas de control de manera que se vea atractivo.
3. Guarda los cambios generados en la ***vista diseño*** de tu **informe**.

Recurso Didáctico Sugerido



https://www.youtube.com/watch?v=bnF0Rl0OUwc

Instrumento de Evaluación LISTA DE COTEJO Práctica 2. Formularios e informes en Access "Otra forma de ver los Datos"					
DATOS GENERALES					
Nombre(s) del estudiante(s)				**Matrícula(s)**	
Producto: FORMULARIO				**Fecha**	
Materia:				**Periodo**	
Nombre del Docente				**Firma del Docente**	
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Trabaja con sus compañeros deforma colaborativa.				
2	Realiza un formulario del sistema de gestión de Información				
2	Incluye alguna imagen dentro del formulario				
2	Coloca los botones de acción de manera correcta en el formulario.				
2	Muestra creatividad en la elaboración del diseño del formulario.				
1	Entrega en tiempo y forma				
	CALIFICACIÓN				

Instrumento de Evaluación LISTA DE COTEJO Práctica 2. Formularios e informes en Access "Otra forma de ver los Datos"					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s)**		
Producto: INFORME			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente			**Firma del Docente**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
2	Trabaja con sus compañeros deforma colaborativa.				
2	Realiza un informe del sistema de gestión de Información				
2	Muestra creatividad en la elaboración del diseño del informe.				
2	Coloca los botones de acción de manera correcta en el informe.				
2	Entrega en tiempo y forma				
	CALIFICACIÓN				

Referencias

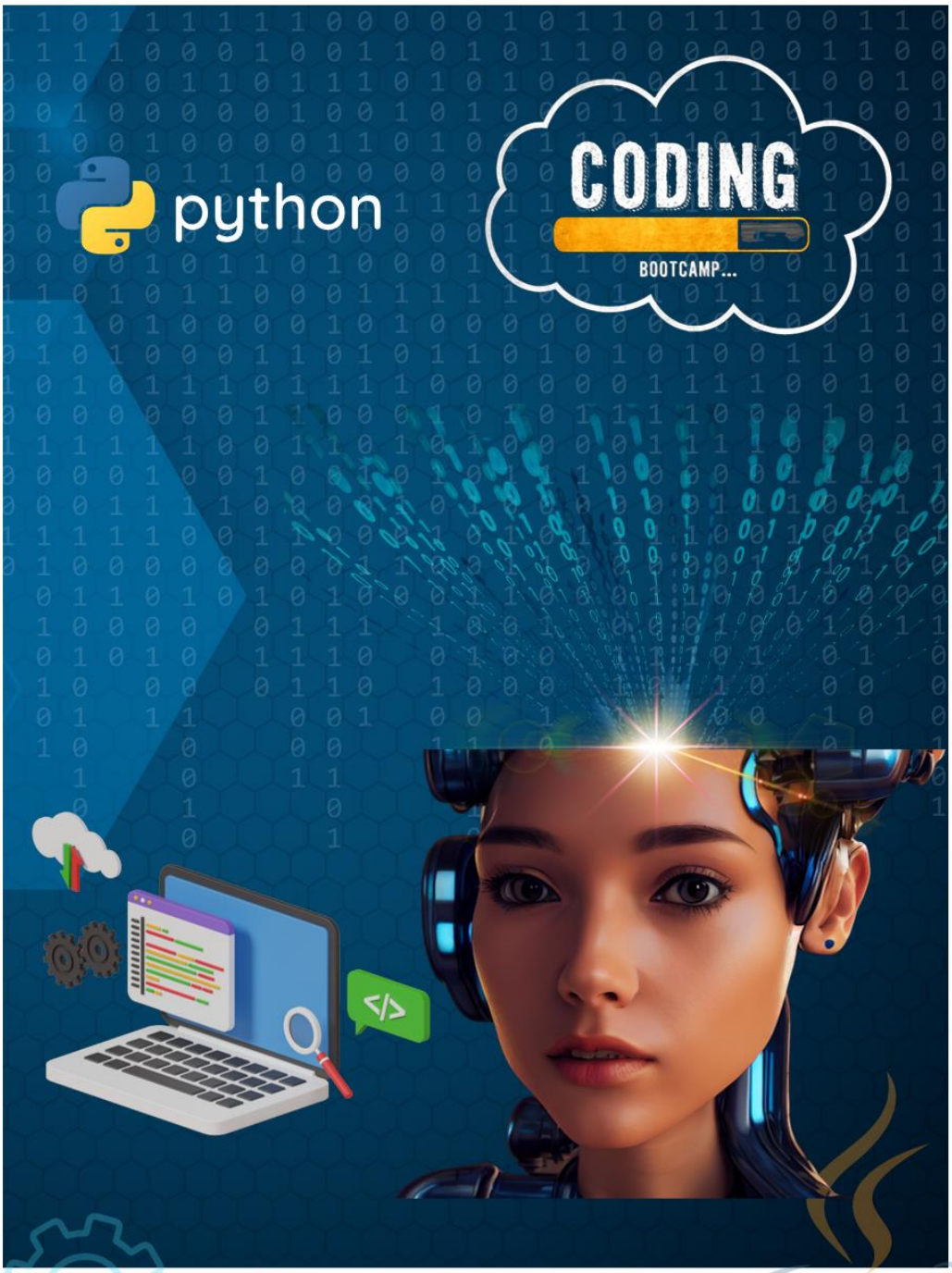
Dzul, F. E. (2020). Tecnología de la Información y la Comunicación III. México: Klik soluciones educativas.

Hurtado D. (2017) Dostin Hurtado. “Curso de Access - Capitulo 2, Crear Informes”. YouTube. https://www.youtube.com/watch?v=bnF0Rl0OUwc

Microsoft (2021). “Introducción a formularios”. Obtenido de: https://support.microsoft.com/es-es/office/introducci%C3%B3n-a-formularios-e8d47343-c937-44e8-a80f-b6a83a1fa3ae

Microsoft (2021). “Crear informes sencillos”. Obtenido de: https://support.microsoft.com/es-es/office/crear-informes-sencillos-408e92a8-11a4-418d-a378-7f1d99c25304

Todo en sistemas Computacionales (2020). “Access formularios, botones para nuevo registro, borrar, guardar y más”. YouTube. https://www.youtube.com/watch?v=elZzkzYluZE



SUBMÓDULO II

PROGRAMACIÓN

PROGRAMACIÓN

Propósito del Submódulo

Plantea soluciones críticas y responsables mediante la metodología de desarrollo de software para demostrar eficiencia en el manejo de base de datos y software de programación de alto nivel, que sean aplicables a necesidades de una empresa o institución para el tratamiento de información.

Aprendizajes Esperados

- Plantea el uso de diagramas de flujo y algoritmos. fomentando su desarrollo creativo. Para solucionar problemas cotidianos de su contexto.
- Explica los lenguajes de programación y sus metodologías de forma consciente. asertiva y empática, en la resolución de problemas del ámbito académico y laboral.
- Propone la creación de códigos con instrucciones secuenciales. Condicionales y/o repetitivas, asumiendo la frustración como parte del proceso de aprendizaje, en la solución de problemas de su entorno.

Competencias

Genéricas	Profesionales
• CG5.2 Ordena información de acuerdo a categorías, jerarquías y relaciones. • CG5.6 Utiliza las tecnologías de la información y comunicación para procesar e interpretar información • CG8.1 Propone maneras de solucionar un problema o desarrollar un proyecto en equipo, definiendo un curso de acción con pasos específicos	CPBTIC5 Propone el diseño de sistemas de información, a partir del análisis de las necesidades de los usuarios, permitiendo la solución de problemas de manera responsable e innovadora en diferentes contextos.

DOSIFICACIÓN PROGRAMÁTICA

Capacitación: Tecnologías de la Información y la Comunicación
Módulo III: Desarrollo de sistemas
Submódulo II: Programación **Clave:** B5PO
Semestre: 5to. **Turno:** Matutino/ Vespertino **Periodo:** 2023 – 2024A

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
Submódulo II: Programación B5PO	Apertura	350	Encuadre del módulo Evaluación diagnóstica **Lógica de programación:** • Algoritmos • Diagrama de flujo • Pseudocódigo	8	9 al 13 de octubre	
	Desarrollo	350	• Decisiones • Ciclos **Lenguajes de programación** • Tipos de lenguajes	9	16 al 20 de octubre	
		350	• Metodología de programación ✓ Estructurado ✓ Orientado a objetos **Actividad construye-T** Lección 4 “Decisiones y emociones”	10	23 al 27 de octubre	 CONSTRUYET
		350	**Programación utilizando un lenguaje de alto nivel.** • Entorno de desarrollo • Variables • Operadores	11	30 de octubre al 3 de noviembre	1 y 2 de noviembre festivo

Submódulo	Momento	Tiempo (minutos)	Conocimientos	Semana	Fecha inicio	Observaciones
			• Constantes • Palabras reservadas			
		350	• Sentencia de decisión • Estructuras • Condición	12	6 al 10 de noviembre	
		350	• Repetición	13	13 al 17 de noviembre	
		350	• Listas	14	20 al 24 de noviembre	20 de noviembre festivo
		350	• Listas	15	27 de noviembre al 1 de diciembre	
	Cierre	350	Situación didáctica	16	4 al 8 de diciembre	
		350	Semana de reforzamiento académico	17	11 al 15 de diciembre	Semana de reforzamiento académico

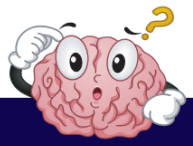
Encuadre Submódulo 2

Submódulo 2 Programación		
Elemento para evaluar		Puntaje
Actividades	L1- Actividad 1. Mi tabla de algoritmos.	7.5
	L2- Actividad 2. Mis primeros algoritmos	7.5
	L3 - Actividad 3. Crucigramas "Lenguajes de Programación"	7.5
	L4- Actividad 4: Mapa conceptual de "Metodologías de Programación"	7.5
	Total	**30**
Prácticas	L5- Práctica 1. Mi primer programa.	10
	L6 - Práctica 2. En busca del código.	10
	L7 -Práctica 3 Calculadora de comisiones.	10
	L8 - Práctica 4 Manipulación de listas.	10
	Total	**40**
SD2	**"un genio para el ingenio"**	**20**
	L3 -CONSTRUYE-T	**10**
	Total	**100**

Situación Didáctica	
Título:	**“Un genio para el ingenio”**
Contexto:	En la actualidad el uso e interacción con las aplicaciones de los dispositivos móviles ha marcado una era en la cual las empresas y los individuos buscan automatizar sus tareas o procesos con la mayor eficiencia y rapidez posible para facilitar la vida de quienes utilizan estas tecnologías. Por tal motivo la programación juega un papel sumamente importante en el desarrollo de estas soluciones innovadoras. Diana es una alumna que estudia el 5to semestre de Bachillerato y cursa la capacitación de TIC, ha visto la importancia de automatizar procesos para agilizar el trabajo y quiere apoyar a su abuelo, quien tiene un local donde vende sus productos de manera física y tiene contratados a varios promotores de venta, pero el día de paga le resulta difícil el calcular sus comisiones, bonos y salario final, por lo que la nieta le explicó que ella en el submódulo 2 Programación, el maestro de la capacitación les había dicho que verían el lenguaje de programación Phyton para el desarrollo de aplicaciones, por lo que platicó la problemática de su abuelo al Docente de capacitación y se obtuvo la idea de generar un proyecto con los alumnos de la capacitación para elaborar un programa que ayude a pagar de manera fácil y exacta los salarios a los empleados por la labor realizada en un periodo de tiempo dentro de la empresa.
Propósito	Elaborar en equipo de 5 integrantes un sistema para calcular el salario a pagar de un trabajador en el lenguaje Python aplicando condiciones, estructuras repetitivas, listas e instrucciones propios del lenguaje de programación para automatizar el proceso de pago de la microempresa y socializarlo en el aula.
Conflicto cognitivo	¿Sabes cómo se crean los programas que se utilizan en tu computadora o tu celular? ¿Conoces a alguien que trabaje desarrollando aplicaciones y que tan remunerada es esta actividad? ¿Qué tan importante es aplicar una metodología para realizar una actividad? ¿Qué herramientas de software debemos conocer para poder desarrollar otras aplicaciones?
Producto esperado	Programa de cálculo de salario en el lenguaje Python.

¿Cómo soluciono la Situación Didáctica?
"Diseñando en mi comunidad"

¡Manos a la obra!



Analiza para resolver:
(Práctica 4 Manipulación de listas)

Calcular las comisiones de ventas de "n" números de trabajadores, en donde los vendedores reciben $500 por semana, más el 12% de sus ventas brutas durante esa semana.

Por ejemplo, un vendedor vende $3,500 de mercancía en una semana, recibe $500 (salario base) + $420 (12% de la venta) **cobrando un total de $920.**

1

Elabora tu algoritmo.

Dentro del algoritmo elabora un menù (usando **switch**) que realice lo siguiente:

1. **Agregar vendedor y su pago total.**
2. **Mostrar pagos semanales de los vendedores.**
3. **Eliminar vendedor y su pago semanal.**

Datos de entrada:
- numero de vendedores.
- venta semanal.

Calculos:
- calcular comisión del 12% de su venta semanal.
- Sumar al salario base ($500) la comisión del 12%

Datos de salida:
- numero de vendedor y pago semanal.

2

Elabora tu diagrama de flujo.

Teniendo tu algoritmo, elabora el correspondiente diagrama de flujo.
puedes utilizar un programa de diseño como:
- **Flowdia**
- **Raptor**
- **PSeInt, entre otros.**

3

Codifica en Python

Teniendo tu algoritmo, codifica las instrucciones siguiendo las sintàxis correspondiente empleado las siguientes estructuras de control:
- **If.**
- **While.**
- **For, etc.**

4

Elabora un documento que contenga los siguientes elementos:
- **Portada.**
- **Propósito de la situación didáctica.**
- **Algoritmo.**
- **Diagrama de flujo.**
- **Código fuente con sus correspondientes comentarios.**
- **Capturas de pantalla de la ejecución del programa.**
- **Conclusiones de lo aprendido.**

5

Entrega de manera digital o impresa.

Evaluación Diagnóstica Submódulo 2

Submódulo II. Programación.

NOMBRE: _____ **GRUPO:** _____

INSTRUCCIONES. Subraya la respuesta que consideres correcta a cada interrogante.

1. ¿Cuál es el proceso por medio del cual se diseña, codifica, limpia y protege el código fuente de programas computacionales?
   a) Base de datos
   b) Programación
   c) Software
   d) Diagrama de bloques

2. ¿Qué es un algoritmo?
   a) Es una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas.
   b) Es la representación gráfica.
   c) Está escrito en lenguaje normal
   d) Es la representación de tablas

3. ¿Tipo de algoritmo que en sus pasos o instrucciones no se realizan cálculosnuméricos?
   a) Algoritmo literario
   b) Algoritmo cualitativo
   c) Algoritmo cuantitativo
   d) Algoritmo estructural

4. Es la representación gráfica de un algoritmo
   a) Pseudocódigo
   b) Lenguaje natural
   c) Diagrama de flujo
   d) Diagrama Nassi-Sheneiderman

5. Es el entorno usado para estructurar, planear y controlar el procesode desarrollo en sistemas:
   a) Metodología para el desarrollo de software
   b) Definición de necesidades
   c) Análisis
   d) Diseño

6. Elige. ¿Cuál es la etapa en donde se obtienen los requisitos para el desarrollodel software?
   a) Metodología para el desarrollo de software
   b) Definición de necesidades
   c) Análisis
   d) Diseño

7. ¿Cuál es la etapa en donde es fundamental que, a través de una colecciónde requerimientos funcionales y no funcionales, el desarrollador o desarrolladores comprendan completamente la naturaleza de los programas quedeben construirse para desarrollar la aplicación, la función requerida, comportamiento, rendimiento e interconexión?
   a) Análisis
   b) Diseño
   c) Definición de necesidades
   d) Metodología para el desarrollo de software

8. Es el conjunto de herramientas que permiten al programador desarrollar programas de informática, usando diferentes alternativas de una manera práctica.
   a) Lenguaje de programación
   b) Algoritmo
   c) Software de programación
   d) Lógica de programación

9. Es un sistema estructurado de comunicación que nos permite comunicarnos, ya sea, a través de palabras, signos, sonidos o gestos con una computadora.
   a) Lógica de programación
   b) Lenguaje de programación
   c) Software de programación
   d) Pseudocódigo

10. Proceso por el cual se escribe, se prueba, se depura, se compila(de ser necesario) y se mantiene el código fuente.
   a) Programación
   b) Pseudocódigo
   c) Programación estructurada
   d) Programación orientada a objetos

11. Es un paradigma de programación basado en utilizar funcioneso subrutinas, y únicamente tres estructuras de control.
   a) Programación
   b) Pseudocódigo
   c) Programación estructurada
   d) Programación orientada a objetos

12. Es un paradigma de programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.
    a) Programación orientada a objetos.
    b) Programación
    c) Programación estructurada
    d) Pseudocódigo

13. En programación, permiten modificar el flujo de ejecución de las instrucciones de un programa.
    a) Estructura secuencial
    b) Estructuras de control
    c) Estructura selectiva
    d) Estructura cíclica

14. Serie de acciones a ser ejecutadas en forma consecutiva.
    a) Estructuras de control
    b) Estructura selectiva
    c) Estructura cíclica
    d) Estructura secuencial

15. Permiten repetir la ejecución de una acción o un grupo de acciones un número determinado de veces o indeterminado.
    a) Estructura selectiva
    b) Estructura cíclica
    c) Estructura secuencial
    d) Estructura de control

Lectura 1: Lógica de programación

Instrucciones: Realiza el análisis de la siguiente lectura, subrayando los conceptos principales.

¿Qué es la lógica de programación?

Si queremos entender que es la Lógica de programación, debemos de iniciar analizado dos conceptos fundamentales que la componen: lógica y programación. Iniciaremos definiendo a la **lógica** como el razonamiento de una secuencia de ideas o hechos llevados a cabo de forma coherente y sin contradicción, en cuanto a la **programación** la podemos definir como el desarrollo o creación de programas, software, sistemas o aplicaciones que tienen un determinado fin y que para ello hace el uso de uno o varios lenguajes de programación.

Tomando en cuenta lo anterior podemos decir que la ***"Lógica de Programación"*** es la organización coherente de las instrucciones de un programa para que su fin sea logrado. Es la base fundamental de un programa por lo que la información proporcionada deber ser clara y organizada ya que serán las instrucciones interpretadas por la computadora o el ordenador.
Así, cualquier persona que desee construir una programación que dé solución a cierta problemática, debe tener en cuenta lo siguiente:

- Construir **El QUÉ**; que son las acciones a realizar para resolver el problema. Esta tarea es previa a toda actividad de programación.
- Definir **El CÓMO**; que son las instrucciones escritas en código para que se realicen las acciones determinadas en el QUÉ.

No es lo mismo **Programación** y **Lógica de Programación** ya que la primera hace referencia a la técnica instruccional en un determinado lenguaje para que un ordenador logre efectuar una tarea. Y la segunda, hace referencia a las técnicas, conceptos y manera en la que se organizan para lograr la solución a una problemática a través de la implementación en un ordenador. (EcuRed, 2022).
Si se desea ser un buen programador, se debe poner en práctica la ***Lógica de Programación*** y en ésta se desarrollan **Algoritmos** los cuales usan para su construcción una serie de elementos que ayudan a su desarrollo y que debes de comprender su función para su uso adecuados, estos son:

- Instrucciones
- Datos.
- Variables.
- Constantes.
- Operadores.

1.1. Algoritmo.
Es un conjunto ordenado y finito de pasos o instrucciones que se encuentran perfectamente definidas de forma simple y por medio de la cuales podemos encontrar o dar solución a un problema. Tienen en su estado inicial una entrada (datos) que son transformados siguiendo los sucesivos pasos indicados (proceso) que nos darán un resultado en su estado final obteniendo así una solución (salida).

Clasificación de los algoritmos:

- **Algoritmo Cualitativo:** Es aquel algoritmo en cuya resolución no intervienen cálculos numéricos, sino secuencias lógicas y/o formales.
- **Algoritmo Cuantitativo:** Es aquel algoritmo que para la implementación de su solución utiliza cálculos numéricos en la definición de sus pasos.

Características de los algoritmos:

Si queremos desarrollar un buen algoritmo debemos tomar en cuenta que debe cumplir con las siguientes características:

- **Secuenciales.** Operan en una secuencia de pasos para ser procesados uno a la vez.
- **Precisos**. No pueden ser ambiguos o subjetivos deben ser claros en sus instrucciones.
- **Ordenados.** Deben de respetar una secuencia precisa y exacta para que al seguirlos tengan sentido y se resuelvan un problema.
- **Finitos**. Deben tener una cantidad finita de pasos; iniciar y terminar en algún momento.
- **Concretos.** Debe de dar un resultado en base a las funciones que fue diseñado.
- **Definidos.** Es decir, si se repite tantas veces como se requiera, este debe arrojar el mismo resultado.

Existen diversas formas de representar un Algoritmo, a continuación, te describimos cada una de ellas:

a) **Texto Narrativo:** Permite observar un procedimiento desde las siguientes etapas: Inicio, proceso y salida de la información de manera simple.

b) **Diagramas de Flujo:** Es la representación gráfica de un algoritmo en la cual se emplean diferentes símbolos para los cuales se describen las acciones a ejecutar; todos ellos están conectados entre sí mediante líneas siguiendo un orden. La forma de representarlos ayuda a solucionar y estructurar un problema.

c) **Pseudocódigo:** Se trata de la descripción de un algoritmo informático de programación de alto nivel compacto e informal. Este utiliza las convenciones estructurales de un lenguaje de programación verdadero pero que a su vez es independiente de cualquiera de ellos.

1.2. Diagrama de flujo.

Hacen uso de una serie símbolos que permiten decir lo mismo que en el lenguaje natural, de una forma gráfica y más entendible. Estos símbolos se conectan entre sí a través de flechas y líneas que marcan la dirección del flujo y establecen el recorrido del proceso.

Sugerencias para la creación de Diagramas de Flujo

1. Los Diagramas de flujo deben escribirse de arriba hacia abajo, y/o de izquierda a derecha.
2. Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección en la que fluye la información en los procesos, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
3. Se debe evitar el cruce de líneas, para lo cual se quisiera separar el flujo del diagrama a un sitio distinto, se pudiera realizar utilizando los conectores. Se debe tener en cuenta que solo se van a utilizar conectores cuando sea estrictamente necesario.

4. No deben quedar líneas de flujo sin conectar
5. Todo texto escrito dentro de un símbolo debe ser legible, preciso, evitando el uso de muchas palabras.
6. Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
7. Solo los símbolos de decisión pueden y deben tener más de una línea de flujo de salida

Simbología empleada en la elaboración de los Diagramas de flujo:

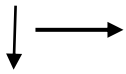
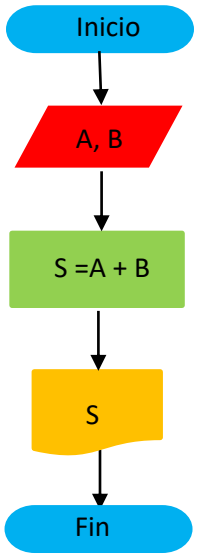
Nombre	Símbolo	Función
Inicio/Final		Se utiliza para representar el inicio o fin de un proceso o programa.
Entrada/Salida		Se utiliza para representar la introducción de datos por medio de periféricos.
Proceso		Representar cualquier tipo de operación que pueda originar cambios de valor, formato o posición de la información almacenada en memoria, aritméticas, de transformaciones,
Documento		Se usa para representar la salida de datos por impresora, pero en ocasiones es usado para mostrar datos o resultados.
Decisión		Se utiliza para indicar operaciones lógicas o de comparación entre datos.
Conector en página		Se utiliza para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama.
Conector fuera de página		Se utiliza para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere en distinta página del diagrama.
Línea de flujo		Indica el sentido de la ejecución de las operaciones.

Tabla 1 Símbolos utilizados en la elaboración de diagramas de flujo.

1.3. Pseudocódigo.

Es una técnica de representación de algoritmos que sirve para escribir programas de computadora en lenguaje natural de tal manera que se facilite la comprensión, prueba y posterior codificación en un lenguaje de programación especifico haciendo uso de estructuras básicas de control: las secuenciales, las selectivas y las iterativas. (Gómez, 2021).

Ejemplo: Representación de las tres formas de un algoritmo que lee dos números y da el resultado de la suma de ellos.

Pasos	Texto Narrativo	Diagrama de flujo	Pseudocódigo
1	Inicio		Inicio
2	Introducir los datos de A y B		Leer A, B
3	Calcular la suma de A con B		Calcular S = A + B
4	Escribir el resultado de la suma de A con B		Escribir S
5	Fin		Fin

Recurso Didáctico Sugerido

SCAN ME

https://youtu.be/as1opL254NA

Actividad 1. “Mi tabla de algoritmos”

Instrucciones: Después de haber analizado la lectura sobre la **lógica de programación**, elabora una tabla donde representes las tres formas de un algoritmo que lea tres calificaciones de un alumno y calcule el promedio.

Pasos	Texto Narrativo	Diagrama de flujo	Pseudocódigo

Referencias

Lógica de programación. (s/f). Ecured.cu. https://www.ecured.cu/Lógica_de_programación

Lógica de programación: el primer paso para aprender a programar. (s/f). Hostgator.mx. https://www.hostgator.mx/blog/logica-de-programacion-primer-paso/

Marker, G. (2020, marzo 22). Pseudocódigo: ¿Qué es? Ejemplos. Tecnología + Informática; Tecnología+Informatica. https://www.tecnologia-informatica.com/pseudocodigo/

Universidad Autónoma del Estado de Hidalgo. (s/f). Algoritmos. Edu.Mx. https://www.uaeh.edu.mx/scige/boletin/prepa4/n10/e1.html

Instrumento de Evaluación LISTA DE COTEJO Actividad 1 “Mi tabla de Algoritmo”					
DATOS					
Nombre(s) del estudiante(s)			**Matrícula(s):**		
Producto: Tabla de Algoritmos			**Fecha:**		
Materia:			**Periodo:**		
Nombre del Docente:			**Firma del Docente:**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
2	Resuelve correctamente el algoritmo en texto narrativo				
2	Resuelve correctamente el algoritmo en Pseudocódigo				
2	Emplea en forma correcta los símbolos en el Diagrama de flujo				
2	Representa las tres formas solicitadas en la actividad.				
1	Expresa claramente las instrucciones en cada uno de los algoritmos				
1	Entrega en tiempo y forma.				
	CALIFICACIÓN				

Lectura 2: Decisiones y ciclos

Instrucciones: Realiza la siguiente lectura, marca las ideas principales, que te ayudarán a resolver la actividad No. 2 Decisiones y ciclos.

La programación estructurada surge a finales de los años 70 para hacer eficiente y fácil la realización y comprensión de los programas. Esta fue propuesta por Böhm-Jacopini y demuestra que todos los programas independientemente del lenguaje que usen para su realización pueden escribirse usando solo tres estructuras de control:

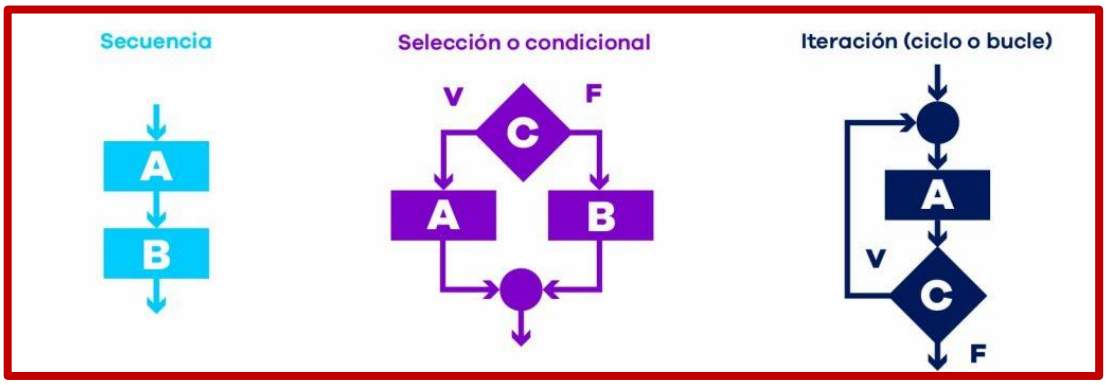


Imagen 1 Estructuras de control. Fuente: https://www.edix.com/es/wp-conte 1

- **Secuencia**. Indica que las instrucciones del código se leerán de principio a fin una tras otra desde la primera línea de código hasta la última, sin excepción.
- **Selección o condicional**. Realizan una pregunta evaluando una condición que retorna en verdadero o falso para seleccionar la siguiente instrucción a realizar.
- **Iteración o repetición**. Indican la repetición de una o varias instrucciones según cierta condición.

En programación estructurada el problema se divide en un número de subproblemas, que deben ser más sencillos que el original. En un programa podemos hacer uso de varios elementos que permiten el control del flujo en los programas los cuales mencionamos a continuación:

a) **Identificador:** Es el nombre o etiqueta que se les asigna a las variables, constantes, funciones, procedimientos y al algoritmo; esto con el fin de identificar claramente cada uno de estos elementos. Ejemplo: volumen, masa, kilogramo, peso.

b) **Variable:** Es un espacio reservado en la memoria en donde se puede almacenar un valor que puede cambiar a lo largo de la ejecución de un programa. Si se desea dar un valor inicial a la variable, éste se coloca a continuación con el símbolo del "=". Ejemplo: x=1.

c) **Constante:** Es un número o carácter, establecido o asignado como valor que no se modifica durante la ejecución de un programa. Por ejemplo: el valor de Pi.

d) **Operadores**: Son elementos que indican las operaciones aritméticas a realizar sobre variables o a otros objetos en una expresión. Ejemplo: (), *, /, +, -.

e) **Operadores de igualdad**: Se utilizan para probar la validez de la relación especificada: ==(es igual que), ! = (no es igual que).

f) **Operadores de relación**: Se utiliza para probar la validez de la relación: >(mayor que) , < (menor que), ≥ (mayor o igual que), ≤ (menor o igual que).

g) **Contador:** En un algoritmo una variable de este tipo se encarga de contabilizar el número de repeticiones, eventos, accesos, etc., se puede declarar de las siguientes formas:

Contador =Contador + constante cuando se desea incrementar o aumentar, o como
Contador = Contador - constante cuando se desea decrementar o disminuir, lo cual siempre hará de forma constante.

h) **Acumulador:** Esta variable se encarga de acumular distintas cantidades e ir guardando o almacenando el total. Generalmente se declara de la siguiente forma:

Acumulador = Acumulador + variable si queremos incrementarlo o
Acumulador = Acumulador - variable si se desea decrementarlo.

Una recomendación muy importante para el uso del contador y el acumulador es que antes de usarlos se le asignen valores iniciales en el algoritmo según el propósito con los que serán utilizados.

i) **Operadores de Asignación:** se utiliza para escribir programas más rápidos, y una forma abreviada de realizarlos es de la siguiente forma:

Suponga que c=3, d=5, e=4, f=6, g=12			
Operador de Asignación	**Expresión**	**Explicación**	**Valor de la Expresión**
+=	c +=7	c= c+3	c=10
-=	d -=4	d=d-4	d=1
*=	e *=5	e= e*5	e=20
/=	f /= 3	f = f/3	f=2

Derivado de lo anterior en este apartado estudiaremos dos de estas estructuras: la selección o condicional (decisiones) y la iteración o repetición (ciclos).

Sentencias de decisión o estructura de control

"Hoy decidí viajar a la escuela en taxi". Esta decisión implica que tendrás que gastar un poco más a diferencia si tomas el bus u otro medio de transporte, pero también obtienes un beneficio, llegar más rápido a la escuela, pero tal vez lo que te orilló a tomar el taxi, es que ya se te hacía tarde para llegar a tu clase o porque te sientes más seguro o cómodo viajar en taxi.

Como te habrás dado cuenta una decisión es influida por diversos factores internos y externos. Una **estructura de decisión** se define como aquel algoritmo que establece la aplicación de ciertas instrucciones si se cumplen o no ciertas condiciones. Las condiciones devuelven un resultado, verdadero o falso, determinando así la secuencia a seguir. Existen 3 tipos de decisiones lógicas:

a. Estructura de decisión simple: Este es el tipo más sencillo de estructura condicional. Ejecuta una instrucción o un bloque de instrucciones cuando la proposición (condición) es verdadera (SI); si es falsa, no hace nada. Se puede representar de la siguiente manera:

```
Si <expresión lógica> Entonces
        Acción
FinSi
```

Imagen 3 Estructura de decisión simple

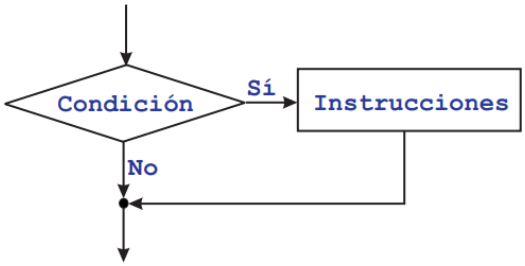


Imagen 3 Diagrama de flujo de la estructura de decisión simple.

Ejemplo:

Realizar algoritmo y diagrama de flujo que determine si un estudiante exenta el examen final, considerando que el promedio de los 2 parciales sea mayor o igual a 8.5.

```
1.  Inicio
2.  Definir Prom, Cal1, Cal2 Como Real;
3.  Escribir 'Calificación del primer parcial: ';
4.  Leer Cal1;
5.  Escribir 'Calificación del segundo parcial: ';
6.  Leer Cal2;
7.  Prom <- (Cal1+Cal2)/2;
8.  Si (Prom>=8.5) Entonces
        Escribir 'Exentas';
9.  Finsi;
10. Fin
```

Imagen 5 Algoritmo con estructura de Control de decisión simple.

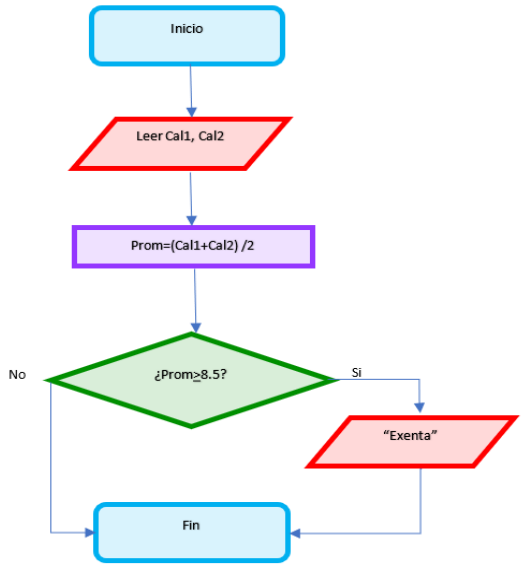


Imagen 5 Diagrama de flujo de la estructura de Control de decisión simple.

B. Estructura de decisión doble: Este tipo de estructura permite implementar condicionales en los que hay dos acciones alternativas:

- Si la proposición (condición) es verdadera se ejecutar una serie de instrucciones (bloque 1).
- Si es falsa (Sino), se ejecuta otra serie de instrucciones (bloque 2).

Su representación se muestras a continuación:

```
Si <condición> Entonces
        Acción(es)
Sino
        Acción(es)
FinSi
```

Imagen 7 Estructura de decisión doble.

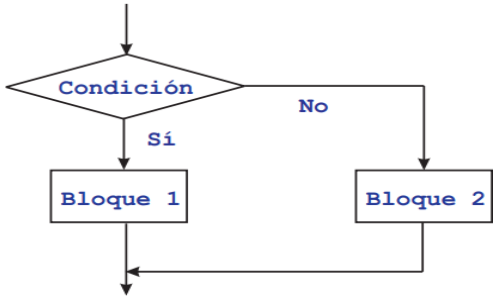


Imagen 7 Diagrama de flujo de la estructura de decisión doble.

Ejemplo:

Realizar algoritmo y diagrama de flujo que determine si el estudiante debe presentar examen final o exentar la materia.

Para resolver este problema se debe considerar que los alumnos deben ser evaluados en 2 parciales y qué deben tener un promedio mayor o igual a 8.5 para exentar el examen final, en caso de tener una calificación menor deben presentar examen final.

```
1  Algoritmo Exenta
2      Definir Prom, Cal1, Cal2 Como Real;
3      Escribir 'Calificación del primer parcial: ';
4      Leer Cal1;
5      Escribir 'Calificación del segundo parcial: ';
6      Leer Cal2;
7      Prom ← (Cal1+Cal2)/2;
8      Escribir 'Tu promedio es: ', Prom;
9      Si (Prom≥8.5) Entonces
10         Escribir 'Exentas';
11     SiNo
12         Escribir 'Presentas examen Final';
13     Finsi;
14 FinAlgoritmo
```

Imagen 8 Algoritmo aplicando estructura de condición doble elaborado en PSeInt

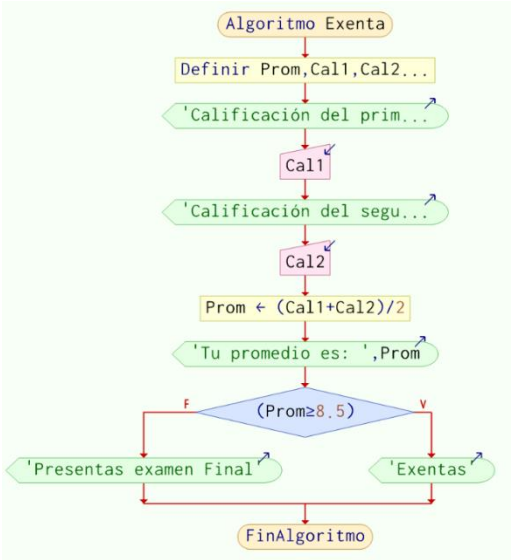


Imagen 9 Diagrama de flujo del programa con estructura de control doble elaborado en PSeInt

```
PSeInt - Ejecutando proceso EXENTA
*** Ejecución Iniciada. ***
Calificación del primer parcial:
> 8
Calificación del segundo parcial:
> 6
Tu promedio es: 7
Presentas examen Final
*** Ejecución Finalizada. ***
```

Imagen 10 Pantalla de resultados del algoritmo con estructura de control doble elaborado en PSeInt.

C. Estructura de decisión múltiple: Permite implementar condicionales más complicados, en los que se "encadenan" condiciones en la forma siguiente:

- Si se verifica la condición 1, ejecutar las instrucciones del bloque 1.
- Si no se verifica la condición 1, pero SI se verifica la condición 2, ejecutar las instrucciones del bloque 2.
- Si no, esto es, si no se ha verificado ninguna de las condiciones anteriores, ejecutar las instrucciones del bloque 3.

```
Si <condición> Entonces
        Acción(es)
Sino
        Si <condición> Entonces
                Acción(es)
           Sino
              .....
              .....        Varias condiciones
```

Imagen 12 Estructura de decisión multiple

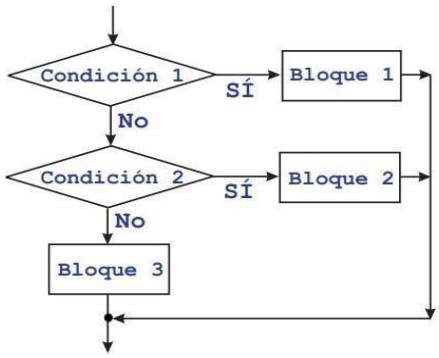


Imagen 12 Diagrama de flujo de la estructura de decisión simple

Ejemplo:

Realizar algoritmo y diagrama de flujo que permita calcular el IMC (índice de masa corporal) de un estudiante y determinar si está en sobrepeso o no, teniendo las siguientes consideraciones:

Si el IMC es inferior a 18.5, está dentro de los valores correspondientes a "bajo peso".
Si el IMC es entre 18.5 y 24.9, está dentro de los valores "normales"
Si el IMC es entre 25.0 y 29.9, está dentro de los valores correspondientes a "sobrepeso".
Si el IMC es 30.0 o superior, está dentro de los valores de "obesidad"

```
1  Algoritmo Masa
2      Definir peso, estatura, imc Como Real;
3      Escribir 'Ingrese su peso (Kg): ';
4      Leer peso;
5      Escribir 'Ingrese su estatura (En metro): ';
6      Leer estatura;
7      imc ← peso/ (estatura*estatura);
8      Escribir 'Su imc es de: ', imc;
9      Si (imc_<18.5) Entonces
10         Escribir 'Bajo Peso';
11     SiNo
12         Si (imc ≥ 18.5 Y imc ≤ 24.9) Entonces
13             Escribir 'Normal';
14         SiNo
15             Si (imc ≥ 25.0 Y imc ≤ 30) Entonces
16                 Escribir 'Sobrepeso';
17             SiNo
18                 Si (imc_ ≤ 30) Entonces
19                     Escribir 'Obesidad';
20                 FinSi
21             FinSi
22         FinSi
23         Finsi
24 FinAlgoritmo
```

Imagen 14 Código aplicando estructura de control múltiple elaborado en PSeInt

```
PSeInt - Ejecutando proceso MASA
*** Ejecución Iniciada. ***
Ingrese su peso (Kg):
> 66
Ingrese su estatura (En metro):
> 1.52
Su imc es de: 28.5664819945Status: Sobrepeso
*** Ejecución Finalizada. ***
```

Imagen 14 Pantalla de ejecución del código con estructura de control de decisión múltiple en PSeInt

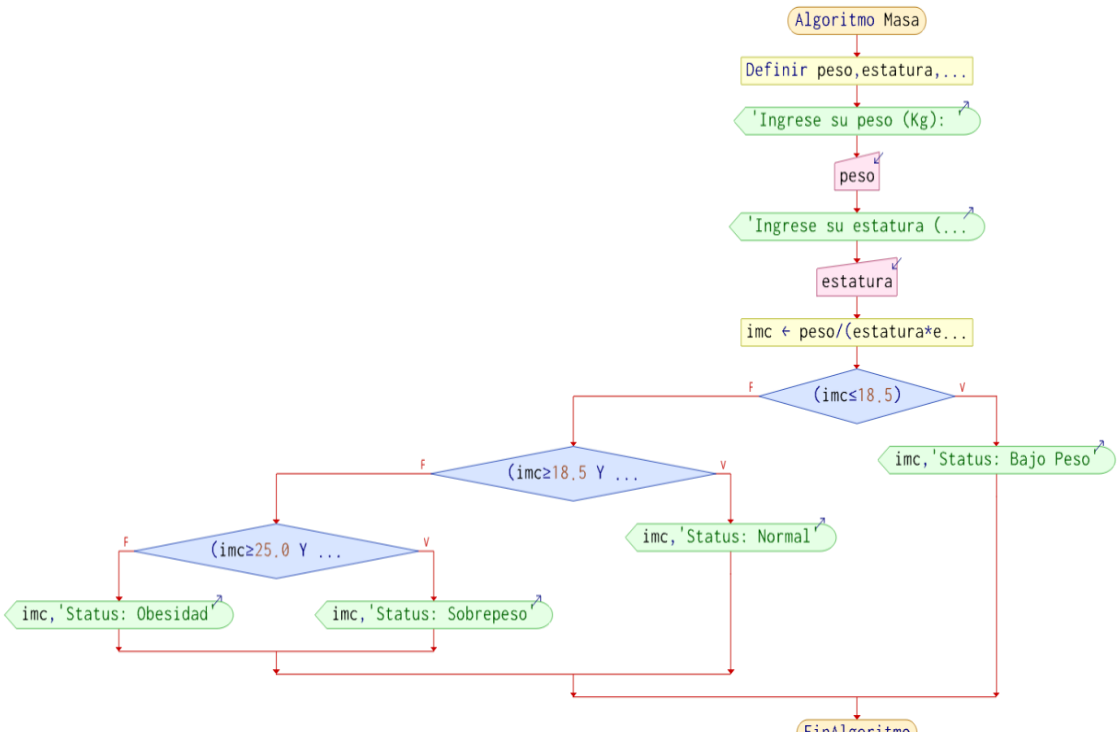
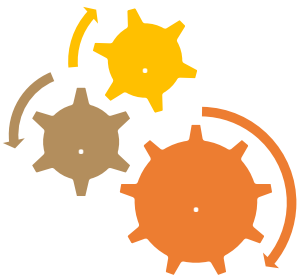


Imagen 15 Diagrama de flujo del algoritmo con estructura de control de decisión múltiple elaborado en PSeInt

Estructura de repetición o ciclos:



Un bucle o ciclo en programación, es una secuencia que ejecuta cero o repetidas veces un conjunto de instrucciones hasta que la condición asignada a dicho ciclo deja de cumplirse. Las operaciones o instrucciones son las mismas pero los datos que se procesan pueden cambiar en la ejecución del ciclo. (UACJ, 2021)

Los tres ciclos o bucles más utilizados en programación son el **while**, el **for** y el **do-while**.

En las siguientes tablas encontrarás los tipos de ciclos que se usan en programación y la forma de representarlos con el ejemplo de un programa que escribe los números del 1 al 10.

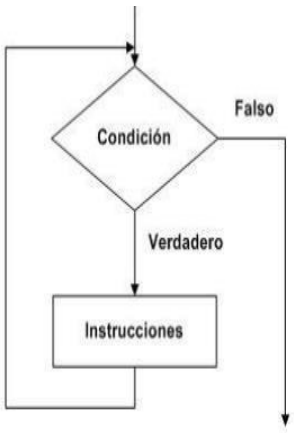
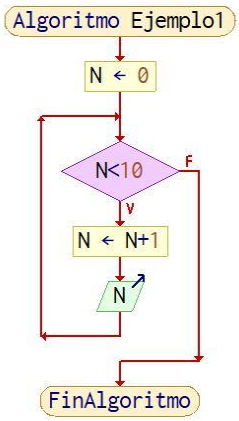
ESTRUCTURA REPETITIVA O BUCLE: MIENTRAS (WHILE)		
Las instrucciones se repiten de forma continua mientras se cumpla la condición. Podrá ocurrir que el ciclo no se ejecute nunca si la condición no llega a cumplirse		
REPRESENTACIÓN	**ALGORITMO**	**DIAGRAMA DE FLUJO RESULTADO**
	1. Inicio 2. Definir N Como entero; 3. N←0; 4. Mientras N<10 Hacer 5. N ← N+1; 6. Escribir N; 7. Fin Mientras	

Tabla 2 Estructura repetitiva while.

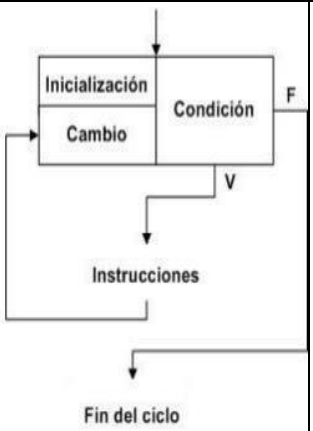
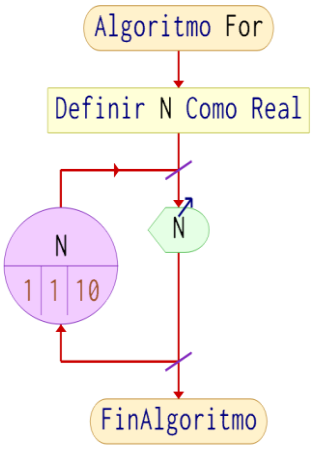
ESTRUCTURA REPETITIVA O BUCLE: PARA (FOR)		
Consiste en una sentencia que engloba un grupo de instrucciones y tiene una variable de tipo entero cuyo valor se va modificando en cada iteración.		
REPRESENTACIÓN	**ALGORITMO**	**DIAGRAMA DE FLUJO RESULTADO**
	1. Inicio 2. Definir N Como entero; 3. Para N←1 Hasta 10 Con Paso 1 hacer 4. Escribir N; 5. Fin para 6. Fin	

Tabla 3 Estructura repetitiva For.

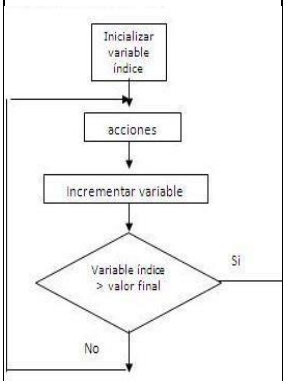
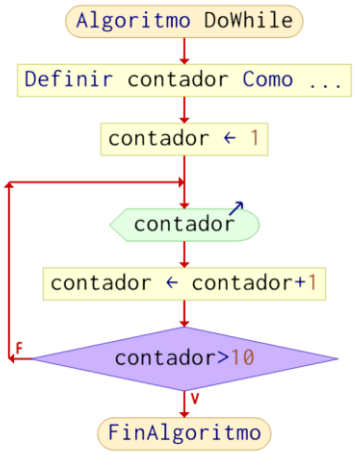
ESTRUCTURA REPETITIVA O BUCLE: REPETIR (DO/WHILE)		
Las instrucciones se repiten de forma continua mientras se cumpla la condición. Podrá ocurrir que el ciclo no se ejecute nunca si la condición no llega a cumplirse		
REPRESENTACIÓN	**ALGORITMO**	**DIAGRAMA DE FLUJO RESULTADO**
	1. Inicio 2. Definir N Como entero; 3. contador<-1; 4. Repetir 5. Escribir contador; 6. contador<- contador+1; 7. Hasta Que contador>10 8. Fin	

Imagen 16 Estructura repetitiva Do-While

Actividad 2. Algoritmo y Diagrama de Decisiones y Ciclos

Instrucciones: En binas resuelve los siguientes planteamientos mediante un software o programa (PSeint, Word, Pseudocode, étc.) sugerido por el Docente.

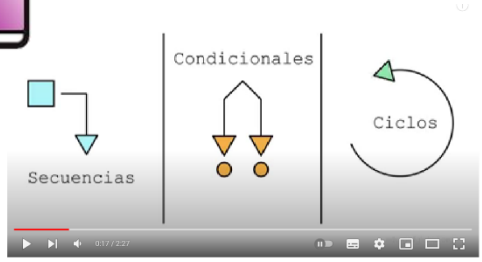
1. Elaborar un algoritmo y diagrama de flujo que lea el salario actual de un empleado y que calcule e imprima el nuevo salario de acuerdo con la siguiente condición: si el salario es menor a $4,000 aumentar el 10%; de lo contrario, no hacer aumento.

Algoritmo	Diagrama de flujo

2. Elaborar el algoritmo y diagrama de flujo que sume los primeros 10 números naturales. Usando uno de los ciclos ya vistos.

Algoritmo	Diagrama de flujo

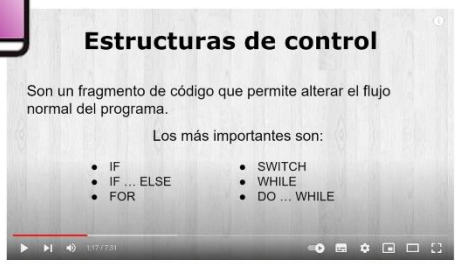
Recurso Didáctico Sugerido



Estructuras de programación (qué es secuencia, condicional, ciclo) | Computación y programación

https://www.youtube.com/watch?v=rNY5eWogl18

Recurso Didáctico Sugerido



EA #7 IF, IF...ELSE - Estructuras de control parte 1 - Curso de programación desde cero en español

https://youtu.be/3hL4wjlcwg8

Referencias

Departamento de Ecuaciones Diferenciales y Análisis Numérico. Algoritmos y estructuras de programación. (2 de marzo de 2009). Universidad de Sevilla. http://departamento.us.es/edan/php/asig/LICFIS/LFIPC/Tema5FISPC0809.pdf

Ejercicios bucles
https://www.edu.xunta.gal/espazoAbalar/sites/espazoAbalar/files/datos/1554983496/contido/RoboticaEnElAula/ejercicios_bucles.html

Ejercicios de algoritmos
http://centros.edu.xunta.es/ieseduardopondal/tecnoweb/temas_informatica/2bac/ejerc_algorit.pdf

EL PROCESO DE LA programación.
http://www3.uacj.mx/CGTI/CDTE/JPM/Documents/IIT/repetitivas/iterativos.html

Esquivel K. (2015). Algoritmos. Unidad III – Estructuras Decisión Lógica. Departamento de Computación. UNAN-León.
https://kesquivel.files.wordpress.com/2015/07/estructuradecisionlogicaiii_2015.pdf

Oviedo E. (2015). Lógica de programación. Capítulo 4. Estructura Decisión Lógica. Sitio web: http://uneweb.com/tutoriales/Diplomado%20Programacion%20Web/LOGICA/Logica- de-Programacion-Efrain-Oviedo.pdf

Instrumento de Evaluación LISTA DE COTEJO Actividad 2. Mis primeros algoritmos					
DATOS GENERALES					
Nombre(s) del estudiante(s):			**Matrícula(s):**		
Producto: Algoritmos de estructuras de control			**Fecha:**		
Materia: Programación			**Periodo: 2023-B**		
Nombre del Docente			**Firma del Docente**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
Algoritmo					
1	Contiene inicio y Fin				
1	Declara correctamente las variables a utilizar				
2	Enumera los pasos y se encuentran ordenados				
2	Usa correctamente la estructura de decisión doble				
2	Usa correctamente la estructura de repetición				
2	Resuelve el problema planteado				
Diagrama de Flujo					
2	Utiliza correctamente los símbolos de inicio y fin				
2	Hace uso del identificador o variable que utilizó en el algoritmo				
2	Utiliza correctamente las flechas de dirección				
2	Utiliza comillas para identificar el mensaje				
2	Usa correctamente la estructura de control: Selección o Repetitiva				
	CALIFICACIÓN				

Lectura 3: Lenguajes de programación.

Instrucciones: Realiza la siguiente lectura, marca las ideas principales y realiza la Actividad 3. Crucigramas "Lenguajes de Programación".

Imagen 17 Fuente: https://piel-l.org/blog/wp-content/uploads/2019/03/singularidad-439.jpg

Los lenguajes de programación han evolucionado enormemente a través de los años, desde sus inicios con el uso de los lenguajes máquinas los cuales eran demasiados complejos y requerían mucho tiempo para su elaboración, hasta nuestros días con el uso de los lenguajes de alto nivel en el que se utilizan lenguajes de programación parecidos al de los seres humanos, el cual hace más fácil la programación.

Los lenguajes de programación son una parte importante para el desarrollo de sistemas o software, por tal motivo, como programador es importante conocer los conceptos básicos de programación, los tipos de lenguajes que se utilizan para su desarrollo y su funcionamiento, para interpretar los algoritmos y para dar solución a los problemas mediante la elaboración de programas.

Un lenguaje de programación es una herramienta de software que nos permite desarrollar **software o programas** para computadora, encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora.

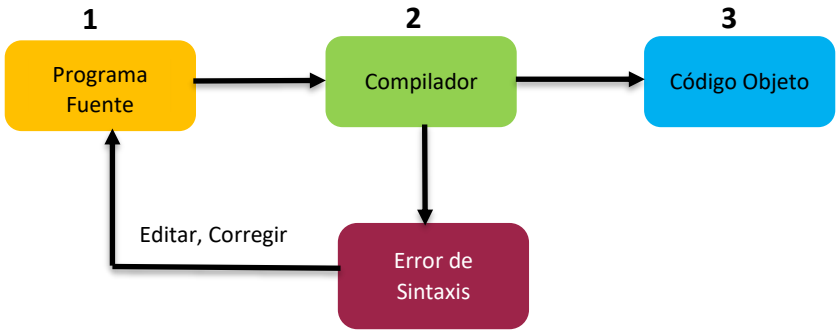
A grandes rasgos, un **lenguaje de programación** se conforma de una serie de símbolos y reglas de sintaxis y semántica que definen la estructura principal del lenguaje y le dan un significado a sus elementos y expresiones.

Programación es el proceso de análisis, diseño, implementación, prueba y depuración de un algoritmo, a partir de un lenguaje que compila y genera un código fuente ejecutado en la computadora. La función principal de los lenguajes de programación es escribir programas que permiten la comunicación usuario-máquina. Unos programas especiales (compiladores o intérpretes) convierten las instrucciones escritas en código fuente, en instrucciones escritas en lenguaje máquina (0 y 1).

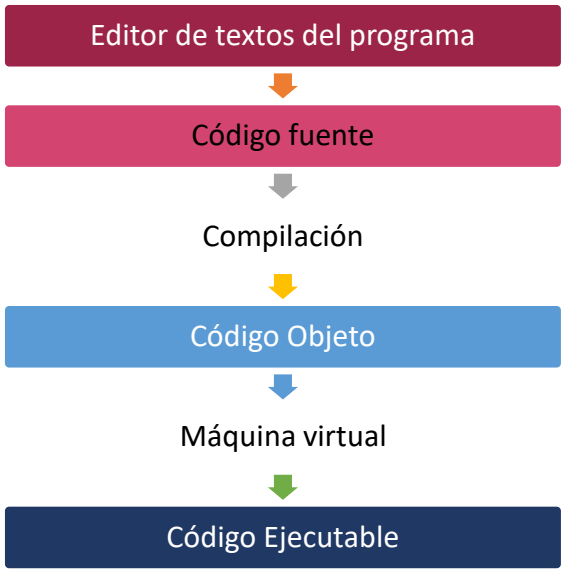
Los **intérpretes** leen la instrucción línea por línea y obtienen el código máquina correspondiente. En cuanto a los compiladores, traducen los símbolos de un lenguaje de programación a su equivalencia escrito en lenguaje máquina (proceso conocido como compilar). Por último, se obtiene un programa ejecutable.

Fases de la compilación

La compilación permite crear un programa de computadora que puede ser ejecutado por ésta y comprende tres pasos:



Algunos procesos de compilación podrían presentar variaciones, pero en general se presenta así:



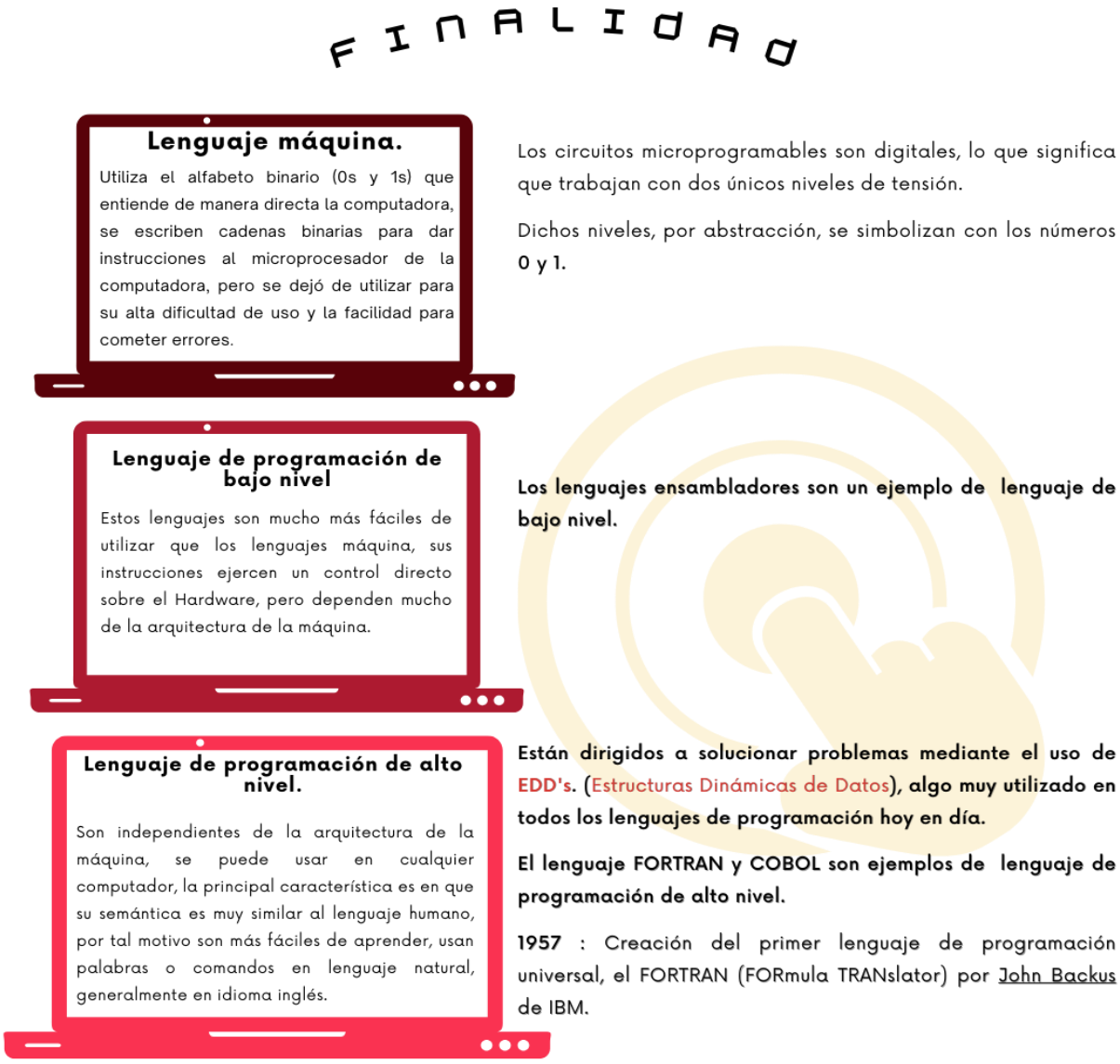
Pero ***¿Cuál es la diferencia entre un compilador y un intérprete?*** Los compiladores realizan la traducción en tiempo de desarrollo; es decir, el programa aún no se está ejecutando. El compilador recibe todo el código fuente, lo analiza, optimiza y traduce a lenguaje máquina dejando un programa completo listo para su ejecución. Por ejemplo, el C o el Pascal son lenguajes compilados.

En cambio, los intérpretes realizan la traducción en tiempo de ejecución, o sea, a medida que el programa se va ejecutando, el intérprete traduce instrucciones al lenguaje máquina. Basic es un lenguaje interpretado.

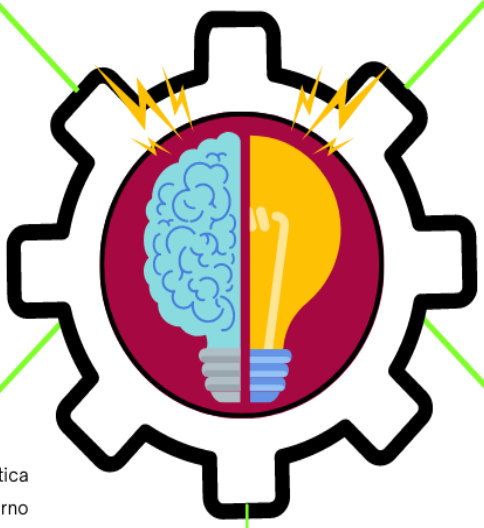
La diferencia está en que el **intérprete realiza la compilación** (interpretación) **y la ejecución de una vez**, mientras que **el compilador genera un formato ejecutable** (código objeto) **que se ejecuta en otra fase posterior**.

Clasificación de los tipos de lenguajes

Hoy en día existen muchos lenguajes de programación por lo que se dificulta un poco su clasificación, ya que puede ser considerada el punto de vista de trabajar, la filosofía, la generación o periodo de su creación, entre otros.



FILOSOFÍA DE SU CREACIÓN



Declarativos

Se enfoca en describe qué/cuál es la solución si entrar en los detalles su control de flujo, es decir, se especifica el resultado deseado, no cómo lograrlo

Prolog, Lisp, Haskell, Miranda

Imperativos

Es este tipo de lenguajes, las instrucciones se ejecutan unas tras otras, de manera secuencial, salvo cuando se encuentran estructuras de control condicionales o bucles.

Cobol, Pascal, C, Ada.

Orientado a Objetos

Es un modelo de programación informática que organiza el diseño de software en torno a datos u objetos, en lugar de funciones y lógica. Un objeto se puede definir como un campo de datos que tiene atributos y comportamiento únicos.

Java, JavaScript, Python, C++, Ruby

Orientado a Eventos

Es un paradigma de programaciòn en el que la estructura y la ejecuciòn de los programas van determinados por los sucesos o acciones que ocurren en el sistema, definidas por el usuario o por el propio sistema.

Java, JavaScript, C#

Naturales

Son los nuevos lenguajes que pretender aproximar el diseño y la construcción de programas al lenguaje de las personas. El **Procesamiento del Lenguaje Natural (PNL)** es el campo de conocimiento de la **Inteligencia Artificial** que se ocupa de la investigar la manera de comunicar las máquinas con las personas mediante el uso de lenguas naturales.

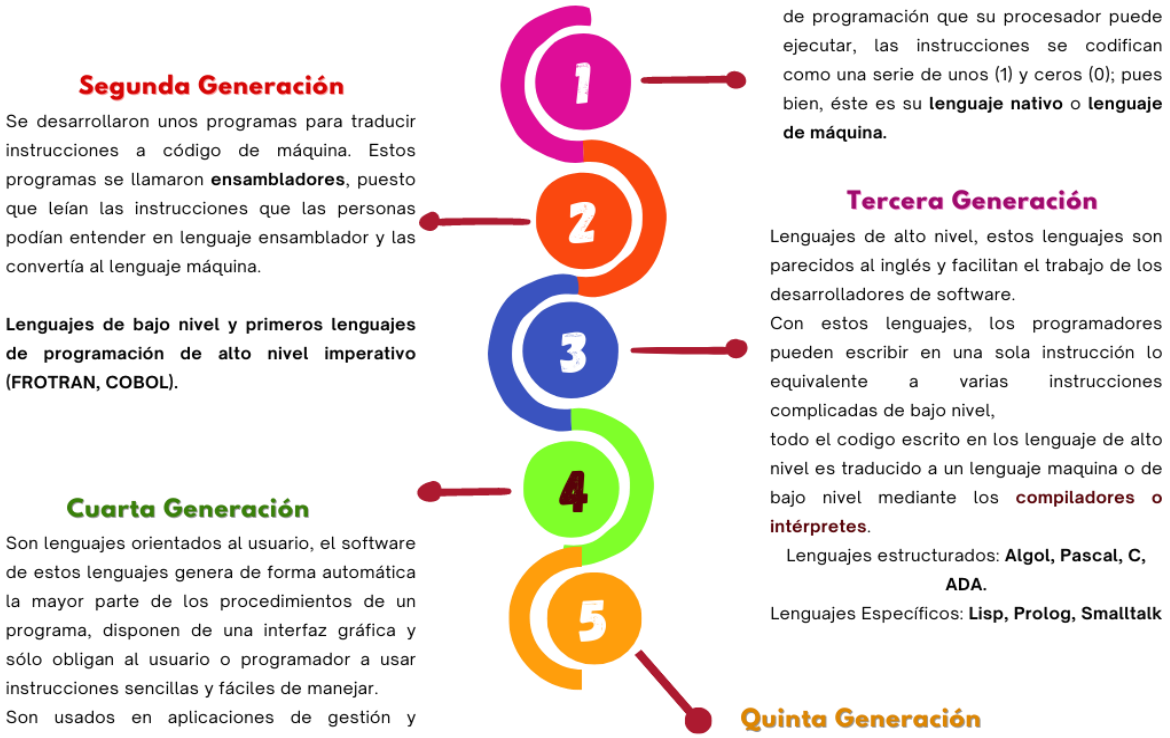
Python, Lenguaje en R, Java, PHP

Orientado al Problema

Es un lenguaje de alto nivel cuyo conjunto de instrucciones está diseñado para hacer frente a un problema especifico

Prolog

SEGUN LAS GENERACIONES



ejemplo de algunos lenguajes de programación.

- **Fortran**. Fue desarrollado en la década de 1950 y es empleado activamente desde entonces, se usa principalmente en aplicaciones científicas y matemáticas. (acrónimo de Formula Translation)
- **C++**. (Lenguaje de alto nivel). Fue diseñado a mediados de la década de 1980 por Bjarne Stroustrup, abarca dos paradigmas de la programación la estructurada y la orientada a objetos.
- **Java**. Lenguaje de programación orientado a objetos desarrollado por Sun Mycrosystems a principios de 1990. Se utiliza mayormente para desarrollar aplicaciones Cliente-Servidor.
- **PHP**. Es empleado frecuentemente para la creación de contenido para sitios web con los cuales se pueden programar las páginas HTML y los códigos fuente. PHP es un acrónimo que significa hypertext preprocessor.
- **Python**. Es un lenguaje de programación moderno, está bajo una licencia de software libre; es orientado a objetos, pero soporta también los estilos de programación procedural y funcional.

Componentes de los lenguajes de programación

Como ya hemos visto, existe una gran variedad de lenguajes de programación los cuales podemos clasificar de muchas maneras, estos programas usan una serie de sintaxis, reglas, etc., las cuales están determinadas por el lenguaje de programación.

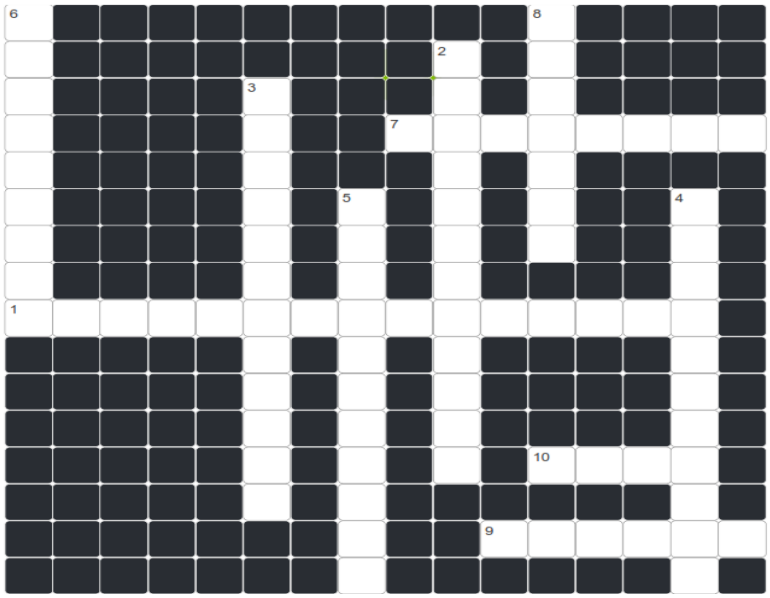
La mayoría de los lenguajes de programación tienen una serie de componentes comunes, pero (sobre todo los más recientes) pueden aportar nuevos elementos que hacen que resulten más atractivos, fáciles de aprender, robustos, flexibles, etc.

Dentro de los componentes que comparten los lenguajes de programación están los siguientes:

Estructura básica	•Instrucciones declarativas, Declaracion de variables, funcion principal (inicio y fin del programa), cuerpo del programa.
Palabras reservadas	•Las palabras reservadas son propias del lenguaje de programación que estamos utilizando, estas palabras ya no podemos ocuparlas al escribir nuestros programas, ya que tienen una función en específico. por ejemplo: **main, cin, cout, if, else, while, for,** entre otras que forman parte de las reglas sintacticas del lenguaje.
Tipos de datos	•Aquellos datos que maneja o soporta cada uno de los lenguajes de programacion, pueden **datos simples** como los enteros, números de coma flotante (reales), caracteres, etc. o bien **datos complejos** como cadenas, listas (vecores), tuplas, diccionarios, entre otros.
Identificadores	•Un **identificador** es una secuencia de caracteres, letras, dígitos y subrayados (_). que utilizamos para declarar **variables, constantes** o **funciones**.
Expresiones y operadores	•Las expresiones **son combinaciones de constantes, variables, símbolos de operación, y paréntesis.** Por ejemplo: **a + (b + 3) / c** . Cada expresión resulta en un valor que se determina al evaluar las operaciones indicadas usando los valores de las variables y constantes implicadas. Un **operador** es un símbolo que se utiliza para manipular datos, se clasifican en: **Asignación, aritméticos, relacionales, lógicos, desplazamiento**.
Instrucciones	•Son estructuras gramaticales predefinidas, para generar secuencias de acciones que conformen un programa. Van desde los operadores aritméticos y lógicos básicos (sumas, restas, *and, or*) hasta instrucciones más especializadas para realizar diversas acciones dentro del programa, como guardado de archivos, volcado de pantalla de un texto, etcétera.
Comentarios	•Nos permiten documentar los programas o códigos, mediante comentarios significativos (documentación interna) y sirven para documentar, explicar o aclarar cómo está hecho el programa. Estos comentarios se representan con diferentes notaciones, según el tipo de lenguaje de programación.

Actividad 3. Crucigramas "Lenguajes de Programación"

Instrucciones. Tomando en cuenta los conceptos aprendidos en la lectura 3, **Resuelve el crucigrama "Lenguajes de Programación".** Toma foto de la actividad como evidencia de su solución.



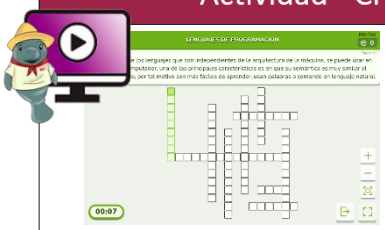
HORIZONTALES

1. Se le llama así al lenguaje que es entendido directamente po la computadora y son cadenas de 0s y 1s.
7. Los lenguajes de programación son una parte para el desarrollo de:
9. Es un lenguaje de programación moderno, está bajo una licencia de software libre; es orientado a objetos, pero soport también los estilos de programación procedural y funcional.
10. Lenguaje de programación orientado a objetos desarrollado por Sun Mycrosystems a principios de 1990. Se utiliza mayormente para desarrollar aplicaciones Cliente-Servidor.

VERTICALES

2. es el proceso de análisis, diseño, implementación, prueba y depuración de un algoritmo, a partir de un lenguaje que compila y genera un código fuente ejecutado en la computadora.
3. (Plural)Programa especial que recibe todo el código fuente, lo analiza, optimiza y traduce a lenguaje máquina dejando un programa completo listo para su ejecución, es decir, crea un archivo .exe
4. Primer Lenguaje de programación en la clasificación de bajo nivel.
5. (Plural)Programa especial que lee la instrucción línea por línea y obtiene el código máquina correspondiente, ejecutando el programa de una vez.
6. Clasificación de los lenguajes que son independientes de la arquitectura de la máquina, se puede usar en cualquier computador, una de las principales características es en que su semántica es muy similar al lenguaje humano, por tal motivo son más fáciles de aprender, usan palabras o comando en lenguaje natural.
8. Fue desarrollado en la década de 1950 y es empleado activamente desde entonces, se usa principalmente en aplicaciones científicas y matemáticas.

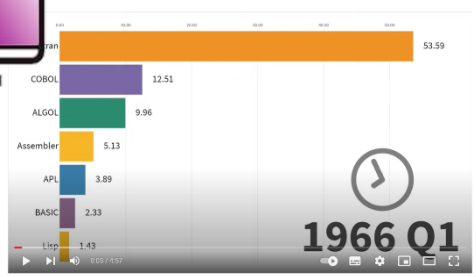
Actividad - Crucigrama



SCAN ME

https://es.educaplay.com/recursos-educativos/12020974-lenguajes_de_programacion.html

Recurso Didáctico Sugerido

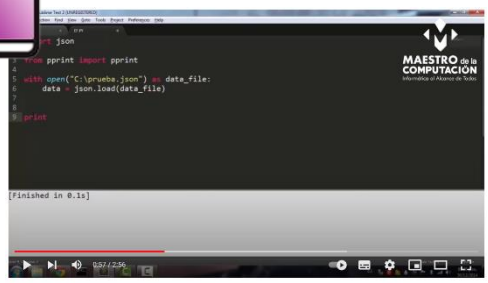


Most Popular Programming Languages 1965 - 2019

SCAN ME

https://youtu.be/Og847HVwRSI

Recurso Didáctico Sugerido



Historia de los Lenguajes de Programación

SCAN ME

https://youtu.be/MtuqC0L2_S0

Referencias

Castelán, J. (2022, marzo 22). ¿Qué lenguaje de programación se usa para inteligencia artificial? Talently Blog. https://talently.tech/blog/que-lenguaje-de-programacion-se-usa-para-inteligencia-artificial/

Clasificación de los lenguajes de programación. (s/f). Larevistainformatica.com. http://www.larevistainformatica.com/clasificacion-de-los-lenguajes-de-programacion.html

González, J. (2017, noviembre 30). Los mejores lenguajes de programación para principiantes. Culturación. https://culturacion.com/los-mejores-lenguajes-programacion-principiantes/

Lenguaje de Programación. (2018, febrero 6). Conogasi. https://conogasi.org/articulos/lenguaje-de-programacion/

Miguel. (2020, diciembre 18). Los lenguajes de programación más usados en Big Data e Inteligencia Artificial. Araba 4.0. https://araba40.eus/en/lenguajes-de-programacion-big-data-inteligencia-artificial/

Monterde, U. M. (s/f). Lenguajes de Programación. Unam.mx. https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

Mota, E. (2019, abril 23). Top 20 Lenguajes de programación a través del tiempo. Azul Web. https://www.azulweb.net/top-20-lenguajes-de-programacion-a-traves-del-tiempo/

Procesamiento del Lenguaje Natural. (2020, octubre 27). Instituto de Ingeniería del Conocimiento. https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/

Software. (s/f). Uoc.edu. https://cv.uoc.edu/moduls/XW02_79049_00373/web/main/m4/v2_2.html

Tipos de datos básicos de Python. (2020, marzo 6). J2LOGO. https://j2logo.com/python/tutorial/tipos-de-datos-basicos-de-python/

Lectura 4. Metodología de la Programación estructurada

Instrucciones: Realiza la siguiente lectura, identificando los conceptos principales y con ello elabora un mapa conceptual.

4.1. Metodología de Programación Estructurada

El modelo de programación estructurada representa un paradigma de programación que utiliza la mayoría de los recursos del lenguaje de programación, basado en utilizar **funciones** o subrutinas, y limitado a un conjunto de estructuras de control (**secuencia, selección o condicional** e **iteración (ciclo o bucle).** Esta metodología plantea la idea de dividir el problema en partes más pequeñas y simples, que sigue un diseño modular descendente.

Metodología: Son los procedimientos, técnicas, herramientas, y soporte documental que facilita al desarrollador implementar programas (software)	**Estructura**: Conjunto de elementos interrelacionados que forman un todo, siguiendo una secuencia de pasos ordenados (secuencial).	**Programación Estructurada**: Es el Diseño, escritura y prueba de un programa, construido con estructura

4.1.1.- Ventajas de la programación estructurada

- Los programas son más fáciles de entender.
- Reducción del esfuerzo en las pruebas.
- Reducción de los costos de mantenimiento.
- Programas más sencillos y rápidos.
- Aumento de la productividad del programador.
- Los programas quedan mejor documentados internamente.

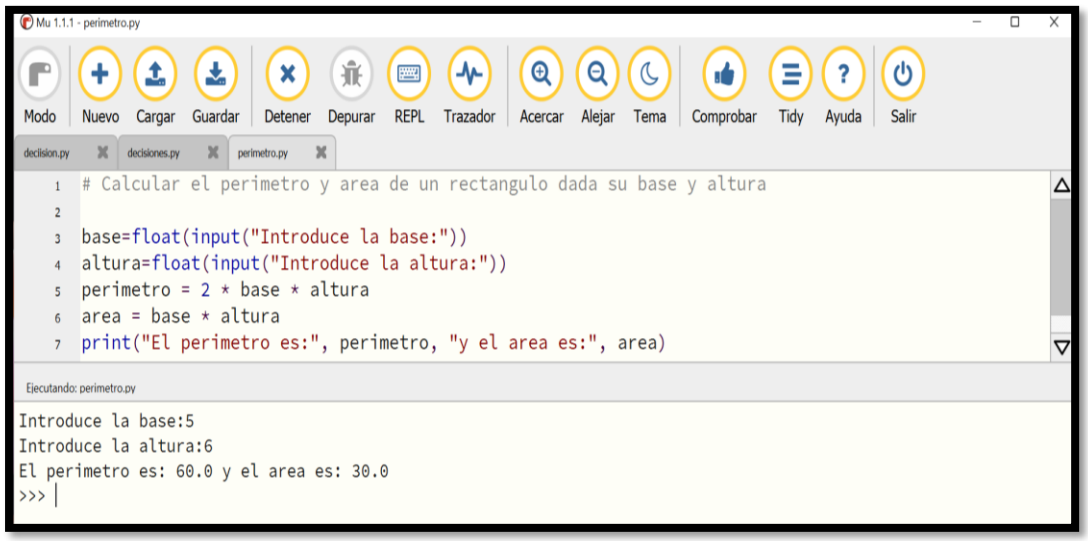
4.1.2.- Desventajas de la programación estructurada

- Se hace complicado el Manejo de programas grandes.
- No permite la reutilización del código.

4.1.3. Estructuras de control:

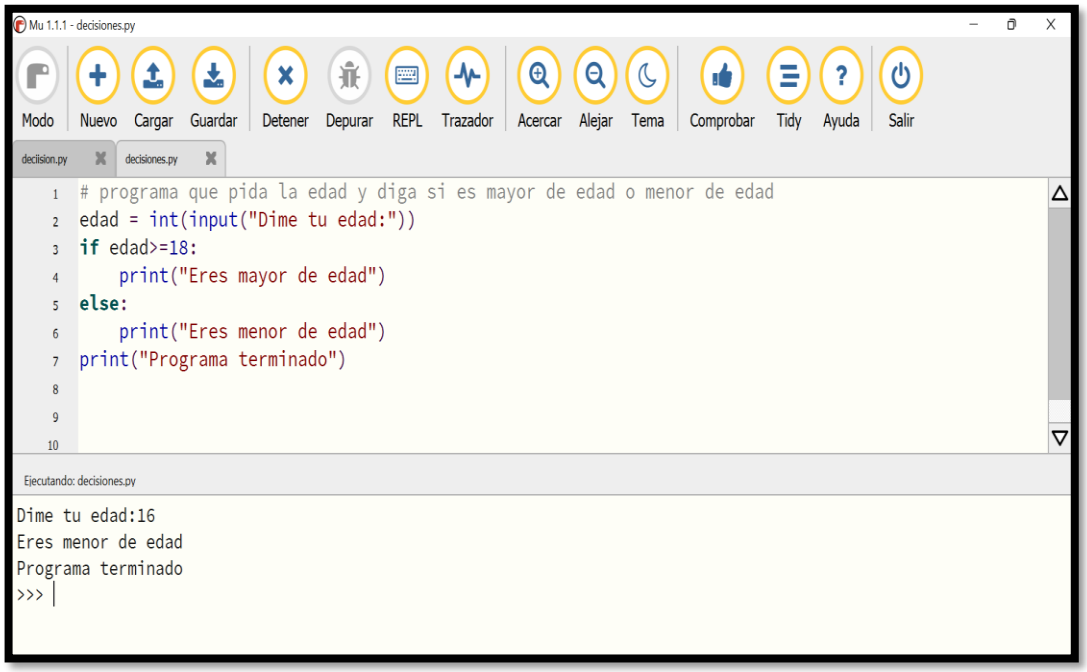
1. La **estructura secuencial** es la que se da naturalmente en el lenguaje, ya que por defecto las sentencias son ejecutadas en el orden en que aparecen escritas en el programa.

Ejemplo en Python



2. Para las **estructuras condicionales o de selección**, Python dispone de la sentencia if, que puede combinarse con sentencias -**else if y/o else.** Existe en sus líneas una decisión de SI o NO.

Ejemplo en Python



3. Para los **bucles o iteraciones** existen las estructuras **while y for**. Las líneas de código serán repetitivas hasta que cumpla la condición.

Ejemplo en Python

4.1.4 Funciones:

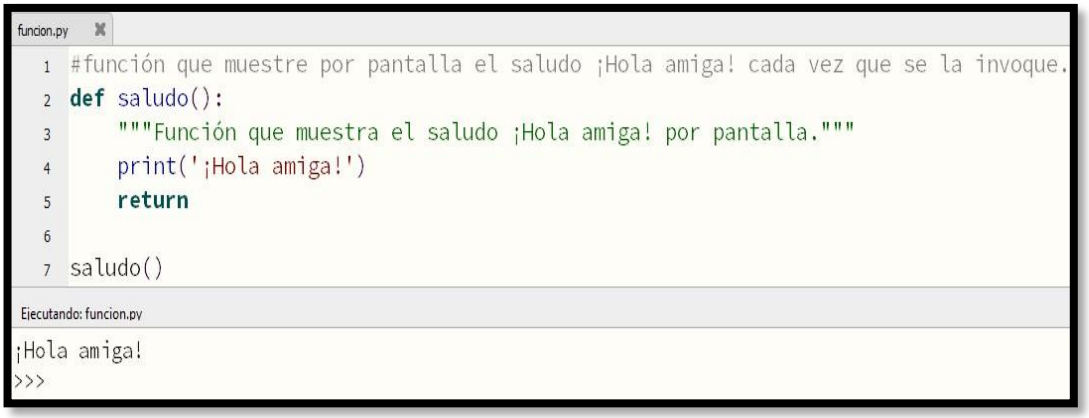
La programación estructurada busca dividir o descomponer un problema complejo en pequeños problemas. La solución de cada uno de esos pequeños problemas nos trae la solución del problema complejo. En Python el planteo de esas pequeñas soluciones al problema complejo se hace dividiendo el programa en funciones.

Una función es un conjunto de instrucciones en Python que resuelven un problema específico. El uso de funciones es un componente muy importante del paradigma de la programación llamada estructurada, y tiene varias ventajas:

- **Modularización:** permite segmentar un programa complejo en una serie de partes o módulos más simples, facilitando así la programación y el depurado.
- **Reutilización:** permite reutilizar una misma función en distintos programas.

Ejemplo de una función en Python.

4.2.- Metodología de Programación Orientada a Objetos



La **Programación Orientada a Objetos** (POO), es un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el **concepto de clases y objetos**. La programación orientada a objetos está enfocada en los atributos de los objetos y en su valor, en cuanto a la clase es la estructura y los procedimientos que se deben de definir en la operación de los datos.

A través de los años, han ido apareciendo diferentes paradigmas o modelos de programación. Lenguajes secuenciales como COBOL o procedimentales como Basic o C, que se centraban más en la lógica que en los datos. Otros más modernos como Java, C# y Python, utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos la más popular.

Con el paradigma de **Programación Orientado a Objetos** lo que se busca es centrarnos en la lógica pura de los programas, para empezar a pensar en **objetos**, lo que constituye la base de este paradigma. Esto es vital para el desarrollo de sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

Un programador diseña un programa de **software** organizando piezas de información y comportamientos relacionados en una plantilla llamada **clase**. Luego, se crean **objetos** individuales a partir de la plantilla de clase.

4.2.1.- Clases, Objetos e instancia

Una **clase** es una entidad, que determina como se comportara y van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse ‘animal’ y tener una serie de **atributos**, como ‘nombre’ o ‘edad’ (que normalmente son propiedades), y una serie con los **comportamientos** que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase (funciones).

Un **ejemplo sencillo de un objeto,** como decíamos antes, podría ser un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de ‘edad’ y, además, puede envejecer, por lo que definimos un nuevo método. Datos y lógica. Esto es lo que se define en muchos programas como la definición de una clase, que es la definición global y genérica de muchos objetos. **Con la clase se pueden crear instancias de un objeto**, cada uno de ellos con sus atributos definidos de forma independiente. Con esto podríamos crear un gato llamado *Luna*, con 1 año de edad, y otro animal, este tipo perro y llamado *Lucky*, con una de edad de 2 años. Los dos están **definidos por la clase animal**, pero son dos **instancias** distintas. Por lo tanto, llamar a sus métodos puede tener resultados diferentes. Los dos comparten la lógica, pero cada uno tiene su estado de forma independiente.

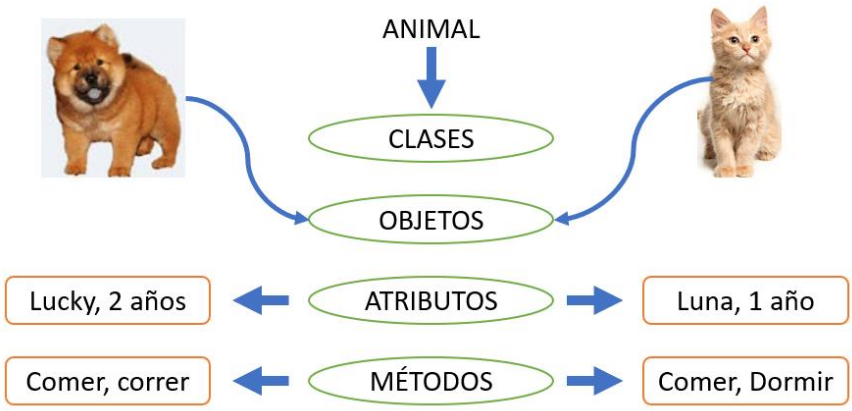


Figura 1.

El siguiente código de ejemplo nos define la clase Perro, con su atributo de clase raza, tamaño, atributo de instancia nombre y edad, acompañado del método características.

Mu 1.2.0 - prueeee.py

Modo | Nuevo | Cargar | Guardar | Detener | Depurar | REPL | Trazador | Acercar | Alejar | Tema | Comprobar | Tidy | Ayuda | Salir

prueeee.py | preuba.py

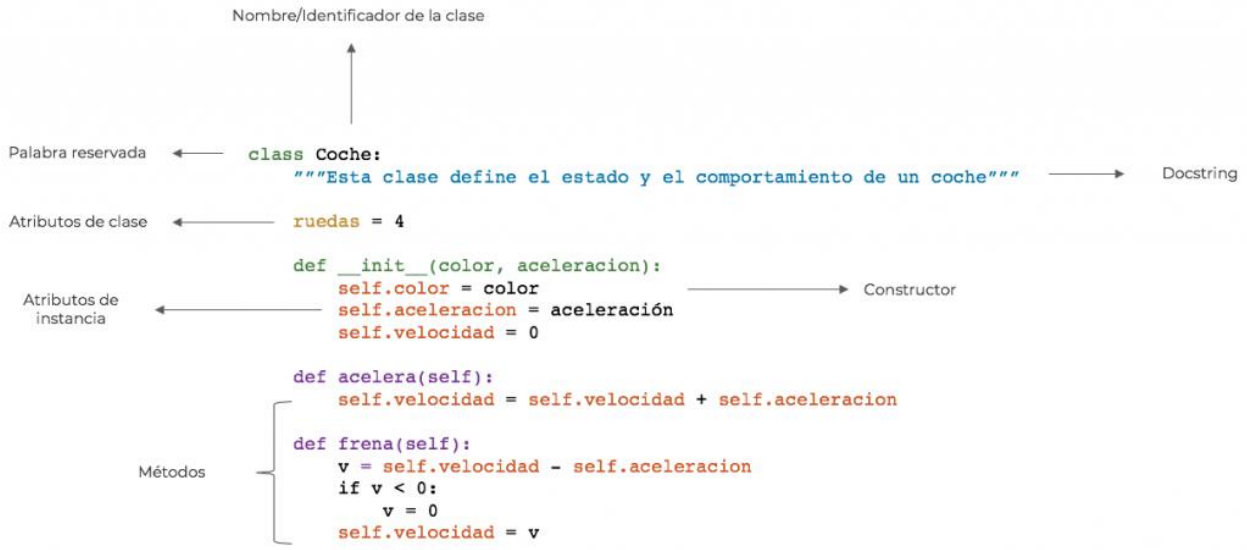
```
class animal():

    def __init__(self, raza,nombre, edad):
        self.raza = raza
        self.nombre = nombre
        self.edad = edad

    def caracteristicas(self):
        print("nombre: ", self.nombre)
        print("edad: ", self.edad, " años")

    def comer(self):
        if( self.raza== "perro"):
            print(self.nombre,"es un perro que come 3 veces al día.")

        if(self.raza=="gato"):
            print(self.nombre,"es un gato que come 2 veces al día.")

miperrito = animal("perro","Lucky", 2)
miperrito.caracteristicas()
miperrito.comer()
```

Ejecutando: prueeee.py

```
nombre:  Lucky
edad:  2  años
Lucky es un perro que come 3 veces al día.
>>>
```

El esquema siguiente define la clase Coche donde podemos identificar la palabra reservada class,

atributos de la clase, las instancias, el constructor y métodos.



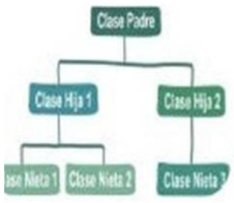
4.2.2.- Principios de la Programación Orientada a Objetos

1.- La encapsulación

La encapsulación contiene **toda la información importante de un objeto dentro del mismo** y solo expone la información seleccionada al mundo exterior, con esto se evita que se pueda acceder a los datos por otros medios no fijados.

2.- La abstracción

La abstracción es cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo.



3.- La herencia

La herencia define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. La principal ventaja de la herencia es la capacidad para definir atributos y métodos nuevos para la herencia donde luego se aplicarán a los atributos y métodos heredados.

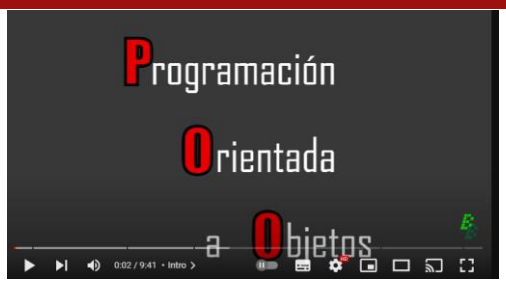
4.- El polimorfismo

El polimorfismo consiste en diseñar objetos para compartir comportamientos, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos.

5.- Modularidad

Consiste en dividir un programa en módulos que puedan compilarse por separado, sin embargo, tendrá conexiones con otros módulos. La modularidad, también tiene principios tales como la capacidad de descomponer un sistema complejo.

Recurso didáctico sugerido



https://www.youtube.com/watch?v=SI7O81GMG2A

4.2.3.- Ventajas de la Programación Orientada a Objetos

- **Reutilización** del código.
- Convierte cosas complejas en **estructuras simples reproducibles**.
- Evita la **duplicación de código**.
- Permite **trabajar en equipo** gracias al encapsulamiento ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
- Al estar la clase bien estructurada permite la **corrección de errores** en varios lugares del código.
- **Protege la información** a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.
- La abstracción nos permite **construir sistemas más complejos** y de una forma más sencilla y organizada.

4.2.4.- Desventajas de Programación Orientada a Objetos

- Tecnología en continua evolución

- Tiempo para cambiar del paradigma de la programación estructurada.
- Mayor complejidad a la hora de entender el flujo de datos

Programación Estructurada	Programación Orientada A Objetos
Se centra en los algoritmos y procedimientos	Diseño enfocado a objetos
Reutilización de código limitada	Permite la reutilización de código
Es difícil el trabajo en equipo	Permite crear sistemas más complejos
Trabajos complejos dificultan su mantenimiento	Facilita el mantenimiento del software
Programas más sencillos y rápidos	Facilita el trabajo en equipo

Tabla 4 Comparación entre Programación Estructurada y programación Orientada a Objetos

Actividad 4. Mapa conceptual de "Metodologías de Programación"

Instrucciones: De manera individual, realiza un mapa conceptual, de acuerdo con la lectura anterior. Puedes incluir imágenes en tu mapa. Puedes apoyarte de herramientas digitales para la creación de tu mapa (GoConqr, CANVA, Mindmeister, etc.).

Instrumento de Evaluación LISTA DE COTEJO Actividad 4 “Mapa conceptual” Metodologías de la programación”					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s):**		
Producto: Mapa conceptual			**Fecha:**		
Materia:			**Periodo:**		
Nombre del Docente:			**Firma del Docente:**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	El mapa contiene el nombre del tema.				
1	Parte de un concepto central la elaboración del mapa.				
2	Jerarquiza las ideas primarias y secundarias, según sea el caso.				
1	Presenta conceptos, palabras de enlace y/o propocisio0nes.				
1	Se establece la relación entre los conceptos.				
1	Los conceptos están en recuadros/nubes/óvalos.				
1	Sintetiza adecuadamente el tema propuesto.				
1	La actividad impacta visualmente y no presenta errores ortográficos				
1	Entrega en tiempo y forma.				
	CALIFICACIÓN				

Referencias

CUAED. (1 de Enero de 2017). Coordinación de Universidad Abierta y Educación a Distancia de la UNAM. Recuperado el 20 de abril de 2022, de CUAED: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

Pérez, U. (11 de Julio de 2002). LaResvistaInformàtica.com. Recuperado el 25 de Abril de 2022, de LaResvistaInformàtica.com: http://www.larevistainformatica.com/clasificacion-de-los-lenguajes-de-programacion.html

Byte-mind. (2017). Curso Python – tema 6 – Programación estructurada, trabajando con funciones. Recuperado 16 de abril de 2023, de byte-mind website: https://byte-mind.net/curso-python-programacion-estructurada/

Domingo Muñoz, J. (2022). Estructuras de control repetitivas: while. Recuperado 14 de abril de 2022, de PLEDIN "Plataforma Educativa Informática" website: https://plataforma.josedomingo.org/pledin/cursos/programacion_python3/curso/u17

Lozano Gómez, J. J. (2022). Programación orientada a objetos (POO) en Python. Recuperado 14 de abril de 2023, de j2logo.com website: https://j2logo.com/python/tutorial/programacion-orientada-a-objetos/

Lección 4 Decisiones y emociones

El reto es que examinen de qué manera las emociones, el contexto, los amigos, las experiencias previas y la sensibilidad a la inmediatez, pueden favorecer u obstaculizar la toma responsable de decisiones.

¿Cuántas veces te ha pasado que después de haber actuado con enojo piensas: "no debí haber dicho eso", "no debí haber actuado de esa manera" o "si tan solo pudiera regresar el tiempo…"? Si bien son muchos los **factores que favorecen u obstaculizan la toma de decisiones**, como las emociones, el contexto, las amistades o las experiencias, es necesario tomar en cuenta el impacto que tienen en nosotros y en quienes nos rodean. Por ello es importante que, al tomar una decisión, lo hagamos buscando el efecto más constructivo, y una forma de hacerlo es revisar nuestros estados emocionales (como factor interno) evitando que la impulsividad decida por nosotros.

Actividad 1

a. Lee el siguiente caso.

Raquel va camino a la plaza. El camión está lleno, pero la señora junto a ella deja un lugar libre. En cuanto Raquel se sienta, saca el celular y abre *WhatsApp*. Lo primero que observa es el siguiente mensaje:

Se trata de un video íntimo en el cual aparece Alejandra, una chica guapa que todos admiran en el salón y con quien Raquel siente una particular rivalidad.

Para tu vida diaria

Piensa en qué consejos le darías a tu mejor amigo si, al momento de tomar una decisión importante, lo ves dominado por sus emociones.

b. Imagina qué emoción estaba experimentando Raquel en cada una de las siguientes situaciones y contesta aquí o en tu cuaderno.

Situación 1

Raquel se sintió triunfante. El corazón le palpitó fuertemente mientras pensaba "a ver si después de esto le sigues gustando a Roberto".

- ¿Qué emoción estaba experimentando Raquel?

_____

- ¿Qué te imaginas que hizo con el video Alejandra?

_____

- ¿Estaba pensando en las consecuencias que traería para ambas? ¿Por qué?

_____

¿Quieres saber más?

En esta conferencia encontrarás una explicación que hace el neurocientífico Facundo Manes sobre la importancia del lóbulo frontal en los seres humanos. En particular te recomendamos que veas a partir del minuto 8. Lo puedes encontrar en: https://bit.ly/1uoiAik

Situación 2

Después de ver el video, Raquel experimentó una opresión en el pecho mientras decía "no me cae bien, pero nadie merece que se viole su privacidad de esta manera. Esto no está bien".

- ¿Qué emoción estaba experimentando Raquel?

_____

- ¿Qué te imaginas que hizo con el video de Alejandra?

_____

- ¿Consideras que Raquel estaba pensando en las consecuencias que le traería a ambas? ¿Por qué?

_____

Actividad 2

a. Dialoga con tu grupo.

- ¿En general, qué tan influenciadas están sus decisiones por las emociones que experimentan? Anota las conclusiones a las que llegaron.

_____

_____

Concepto clave

Factores que favorecen u obstaculizan la toma de decisiones. Agentes internos y externos a la persona que toma la decisión, y que impactan de manera benéfica o perjudicial en ella misma y en quienes la rodean.

Reafirmo y ordeno

Todo lo que hacemos o decimos genera un impacto en nosotros y en nuestro entorno. Por ello, es muy importante hacernos conscientes de los factores que favorecen u obstaculizan la toma de decisiones, por ejemplo, ante un cambio de panorama o situación, es necesario identificar las diferentes alternativas y sus posibles consecuencias, pasar de la decisión a la acción, considerando información confiable para tomar decisiones pertinentes. Cabe destacar, que las emociones desempeñan un papel particularmente significativo por la fuerza y la frecuencia con la que afectan nuestro comportamiento. Si ignoramos la manera en la que nos impactan, será más fácil que la impulsividad determine nuestro quehacer, generando circunstancias que puedan dañarnos a nosotros mismos y a los que nos rodean.

Escribe en un minuto qué te llevas de la lección

_____

_____

_____

_____

Lectura 5. Programación utilizando un lenguaje de alto nivel.

Instrucciones: *Realiza el análisis de la siguiente lectura posteriormente realiza la Práctica 1.*

Python es un lenguaje de programación de propósito general muy poderoso y flexible, a la vez que sencillo y fácil de aprender. Es un lenguaje de alto nivel que permite procesar fácilmente todo tipo de estructuras de datos, tanto numéricos como de texto. Fue creado por Guido Van Rossum iniciando su desarrollo en 1989 y su implementación en febrero de 1991 con la publicación de la primera versión pública: la 0.9.0 que incluía clases con herencias, manejo de excepciones, funciones y una de las características principales de Python: funcionamiento modular. La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008.

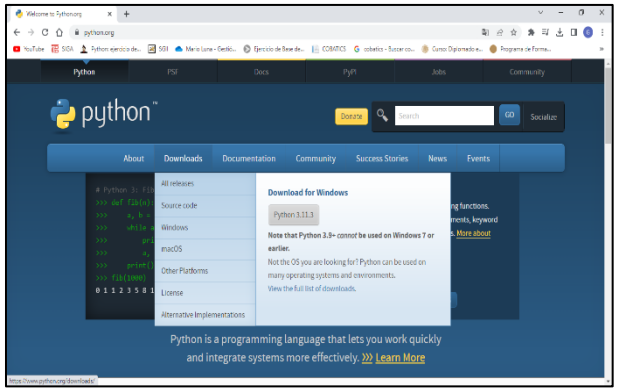
Características

Tipado dinámico	•Es cuando una misma variable puede tomar valores de distinto tipos en distintos momentos y en cualquier lugar de su código fuente.
Código abierto y libre	• No requiere licencias de pago y permite que cualquier desarrollador contribuya al código
Multiplataforma	• Se puede ejecutar en diferentes sistemas operativos como: Linux, Windows, UNIX, Mac con diferentes intérpretes para cada sistema.
Programación Oreintas a objetos (POO)	•Se organiza en clases y objetos, por lo cual es posible representar conceptos de la vida diaria en el lenguaje.
Interpretado	• Se ejecuta directamente el código línea por línea y en caso de contener errores en el código del programa, su ejecución se detiene.

5.1 Entornos de desarrollo

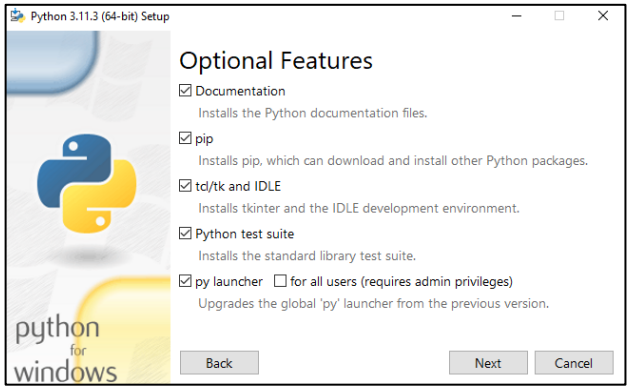
Instalación de Python para Windows 10

1. Ir a la URL de Python en el siguiente enlace: https://www.python.org/
2. Ir a la sección de **descargas** dar clic en **Python 3.11.3** para descargar el ejecutable para Windows (Desde ahí también puedes descargar la versión de Python de acuerdo con el sistema operativo de tu equipo).

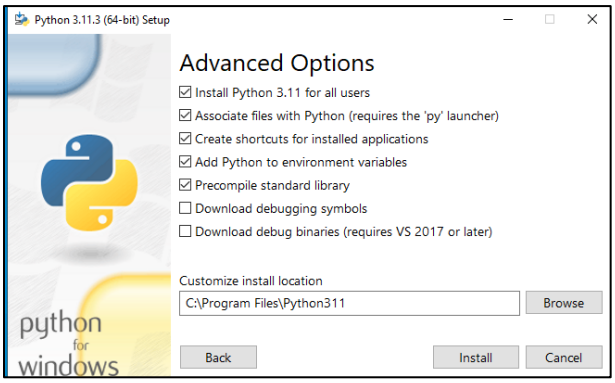


3. Una vez descargado inmediatamente después será abierto el asistente de instalación de Python en Windows 10 y en primer lugar activar la casilla **Add python.exe to PATH.**

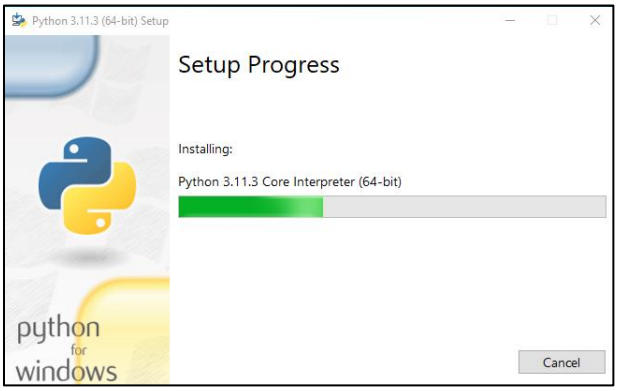
4. Luego seleccionar la opción **Customize installation** y se verá la pantalla como la siguiente y dar clic en **Next.**

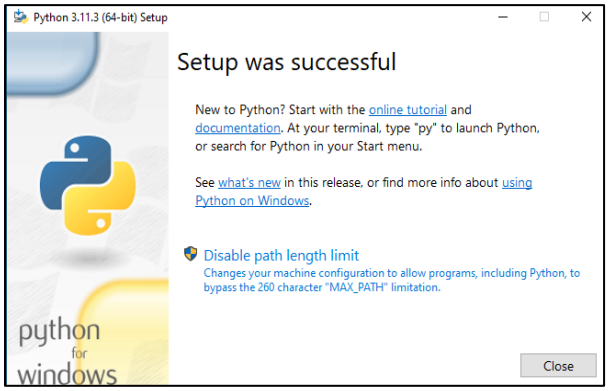


5. En la siguiente ventana se verán algunas opciones adicionales y se debe de activar la casilla **Install for all users** y esto modificara la ruta donde se instala Python en el sistema.



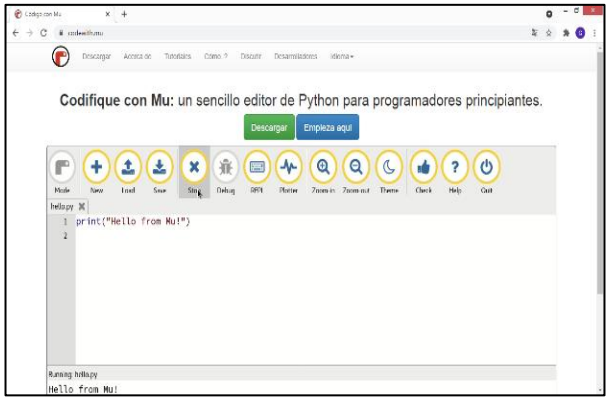
6. Después dar clic en **Install** para llevar a cabo el proceso de instalación de Python 3.11.3 en Windows 10. Una vez finalizado el proceso de instalación se debe dar clic en **Close (cerrar)** para salir del asistente.





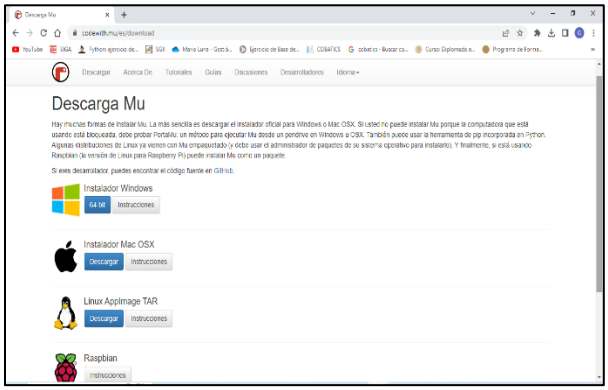
Instalación de Mu

Mu es un editor de código abierto que busca facilitar a los estudiantes aprender a codificar con Python, es una creación de Nicholas Tollervey, es una aplicación de código abierto (*con licencia bajo GNU GPLv3*) escrita en Python**. Originalmente fue desarrollado para trabajar con la minicomputadora Micro bit. Gracias** a los comentarios y las solicitudes de otros Docentes, impulsaron a su creador a reescribir Mu como un editor genérico de Python.



Si eres un usuario de Windows, puedes instalar Mu Editor con un instalador .exe de la siguiente manera:

1. Ir a https://codewith.mu/es/download
2. Descarga el instalador de Windows que corresponda a la arquitectura de tu procesador.
3. Ábrelo y acepta el aviso de seguridad.
4. Completa los pasos del instalador y haz clic en instalar.



5.2. Variables

Una variable es un contenedor que permite identificar y almacenar valores de datos en la memoria de la computadora.

La forma de dar valor a una variable se realiza por asignaciones y en Python pueden ser:

- **Simples:** se asigna un valor a una única variable.
- **Múltiples:** se asignan valores a varias variables a la vez.

En la operación de asignación de las variables se ven involucradas tres partes:

1. Un identificador o nombre de variable, a la izquierda del operador.
2. El operador de asignación igual **"= "**
3. Una literal, una expresión, una llamada a una función o una combinación de todos ellos a la derecha del operador de asignación.

identificador = [literal o expresión]

Una variable se crea en el momento en que se le asigna un valor por primera vez, en función del tipo de información que guarde (texto, números, booleanas, etc.), la variable será de uno u otro tipo, es decir se definen de forma dinámica.

Cada variable debe de tener un nombre con el que referirnos a ella.

- Python tiene en cuenta si escribimos en mayúsculas o minúsculas la variable **(case sensitive).**
- El nombre de la variable no puede coincidir con los nombres de los **«comandos»** de python (if, for, etc.).
- No se pueden usar nombres de variables con tildes o con ñ.

Para la comprender el ejemplo de variables que viene más adelante se requiere el uso de la función print() que se explica a continuación.

Función print()

Es una función, por lo que el contenido siempre debe estar entre paréntesis y sirve para mostrar texto o incluso el valor de alguna variable.

# Ejemplo 1 	
print('texto')
	# Ejemplo 2
	cadena = 'Esto es una cadena'
	print(cadena)

Ejemplo: Asignación e impresión de variables.

```
# VARIABLES
# Asigna a la variable <a1> el valor 1
a = 1
# Asigna a la variable <a2>  el resultado de la expresión 3 * 4
a2 = 3 * 4
# Asigna a la variable <a3> la cadena de caracteres 'Pythonista'
a3 = 'pysthonista'
# Se imprime el valor de cada variable
print (a)
print (a2)
print (a3)
print ("Esto es un texto")

```

```
Ejecutando: ejemplo1.py
1
12
pysthonista
Esto es un texto
>>>
```

Función input()

Permite la introducción de datos mediante el teclado, al ejecutarse esta función, el programa se detiene esperando que se escriba algo y se pulse la tecla **Intro.** De forma predeterminada, la función **input()** convierte la entrada en una cadena, aunque escribamos un número. Si intentamos hacer operaciones, se producirá un error.

Ejemplo:

```
# Función input()
nombre = input('Digite su Nombre :')
edad = input('Digite su Edad :')
estatura = input('Digite su Estatura :')
print(nombre, edad, estatura)
```

```
Ejecutando: input.py
Digite su Nombre : Juan
Digite su Edad :25
Digite su Estatura :1.75
 Juan 25 1.75
>>>
```

Si se quiere que Python interprete la entrada como un número entero, se debe utilizar la función **int()** y si se quiere que interprete la entrada como un número decimal, se debe utilizar la función **float().** Pero si el usuario no escribe un número, las funciones int() o float() producirán un error.

Ejemplo:

```
edad = int(input('Teclear edad: '))  # entrada de entero
peso = float(input('Teclear peso: '))  # entrada de flotante
nombre = input('Teclear nombre: ')  # entrada de cadena
print(nombre, edad, 'años', peso, 'kg')  # muestra datos
```

Comentarios

Son cadenas de caracteres las cuales constituyen una ayuda esencial tanto para quien está desarrollando el programa, como para otras personas que lean el código.

Se pueden agregar de dos formas:

- Escribiendo el carácter numeral **#** delante de la línea de texto donde está el comentario.
- Escribiendo triple comilla doble («»») al principio y al final del comentario (que puede ocupar más de una línea).

Ejemplo:

```
# Esto es un comentario en Python de una línea
print("Ejemplo de Comentario de una linea")
print("Ejemplo de Comentario después de una sentencia") # Después de una sentencia
"""
Esto es un comentario multilínea
"""
print("Comentario multilínea")
```

5.3 Constantes

Son contenedores de datos que no tienen que variar durante la ejecución del programa, son usualmente declaradas y asignadas en un módulo. El módulo significa un nuevo archivo que contiene variables, funciones, etc.; el cual es importada en el archivo principal.

Dentro del módulo, las constantes son escritas en letras **MAYÚSCULAS** y separadas las palabras con el carácter underscore **_**.

Ejemplos:

PI = 3.1416

HORAS_DIA = 24

DIAS_SEMANA = 7

Ahora bien, al igual que ocurre con tipos, algunas constantes vienen integradas en el lenguaje de Python. A continuación, te especificamos cada una de ellas:

1. **None:** el tipo asignado con la constante none tiene un valor único aplicable a un solo objeto. Por lo tanto, su valor no podrá ser usado en diferentes situaciones.
2. **NoteImplement:** al igual que None, esta constante tiene un solo valor y aplica a un solo objeto. Sin embargo, cuenta con algunos métodos numéricos y de comparación enriquecidos para indicar que no se está implementando respecto a otro tipo.
3. **Ellipsis:** esta constante también tiene un solo valor aplicable a un solo objeto. Su presencia indica que hay una sintaxis, normalmente englobada entre comillas.
4. **False:** al igual que true, esta constante es de tipo booleano. Únicamente puede tener dos valores, y este, como es el nombre lo indica representa una condición falsa.
5. **True:** valor de tipo booleano qué representa que una expresión condicional o un bucle es verdadera.
6. **_debug_:** sí Python no se inició con la opción 0, entonces esta constante tendrá un valor predeterminado de true.

5.4 Palabras reservadas de Python

A continuación, se presentan cada una de **las palabras reservadas.**

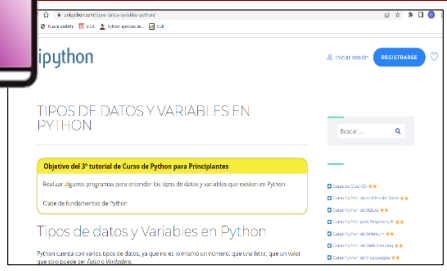
1. **False** – Valor booleano, resultado de operaciones de comparación u operaciones lógicas en Python
2. **None** – Representa a un valor nulo
3. **True** – Valor booleano, igual que false, resultado de operaciones de comparación u operaciones lógicas en Python
4. **__peg_parser__** – Llamado huevo de pascua, relacionado con el lanzamiento del nuevo analizador PEG no está definido aún.
5. **And** – Operador lógico
6. **As** – Se utiliza para crear un alias al importar un módulo.
7. **Assert** – Se utiliza con fines de depuración
8. **Async** – Proporcionada por la biblioteca 'asyncio' en Python. Se utiliza para escribir código concurrente en Python
9. **Await** – Proporcionada por la biblioteca 'asyncio' en Python. Se utiliza para escribir código concurrente en Python
10. **Break** – Se utiliza en el interior de los bucles for y while para alterar su comportamiento normal
11. **Class** – Se usa para definir una nueva clase definida por el usuario
12. **Continue** – Se utiliza en el interior de los bucles for y while para alterar su comportamiento normal
13. **Def** – se usa para definir una función definida por el usuario
14. **Del** – Para eliminar un objeto
15. **Elif** – Se usa en declaraciones condicionales, igual 'else' e 'if'
16. **Else** – Se usa en declaraciones condicionales, igual 'elif' e 'if'
17. **Except** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que 'raise' y 'try'
18. **Finally** – Su uso garantiza que el bloque de código dentro de él se ejecute incluso si hay una excepción no controlada

19. **For** – Utilizado para hacer bucles. Generalmente lo usamos cuando sabemos la cantidad de veces que queremos que se ejecute ese bucle
20. **From** – Para importar partes específicas de un módulo
21. **Global** – Para declarar una variable global.
22. **If** – Se usa en declaraciones condicionales, igual 'else' y 'elif'
23. **Import** – Para importar un módulo
24. **In** – Para comprobar si un valor está presente en una lista, tupla, etc. Devuelve True si el valor está presente, de lo contrario devuelve False
25. **Is** – Se usa para probar si las dos variables se refieren al mismo objeto. Devuelve True si los objetos son idénticos y False si no
26. **Lambda** – Para crear una función anónima
27. **Nonlocal** – Para declarar una variable no local
28. **Not** – Operador lógico
29. **Or** – Operador lógico
30. **Pass** – Es una declaración nula en Python. No pasa nada cuando se ejecuta. Se utiliza como marcador de posición.
31. **Raise** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que 'except y 'try'
32. **Return** – Se usa dentro de una función para salir y devolver un valor.
33. **Try** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que 'raise' y 'except
34. **While** – Se usa para realizar bucles.
35. **With** – Se usa para simplificar el manejo de excepciones
36. **Yield** – Se usa dentro de una función al igual que 'return', salvo que 'yield' devuelve un generador.

Recurso Didáctico Sugerido

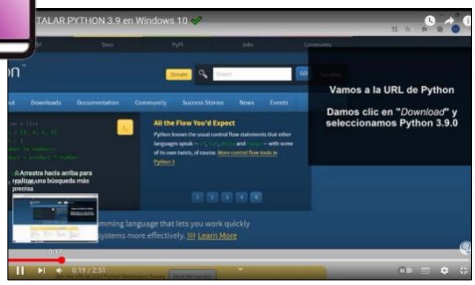
https://www.youtube.com/watch?v=lc5JJTQa4r8&t=150s

Recurso Didáctico Sugerido



https://unipython.com/tipos-datos-variables-python/

Recurso Didáctico Sugerido



https://www.youtube.com/watch?v=fySn0TIikoU&t=1s

Práctica 1. Mi primer programa

Propósito: Lograr que aplique los conocimientos básicos para la creación de un programa secuencial en un lenguaje de programación de alto nivel.

Conocimientos:

- ✓ Variables
- ✓ Comentarios
- ✓ Función input()
- ✓ Función print()

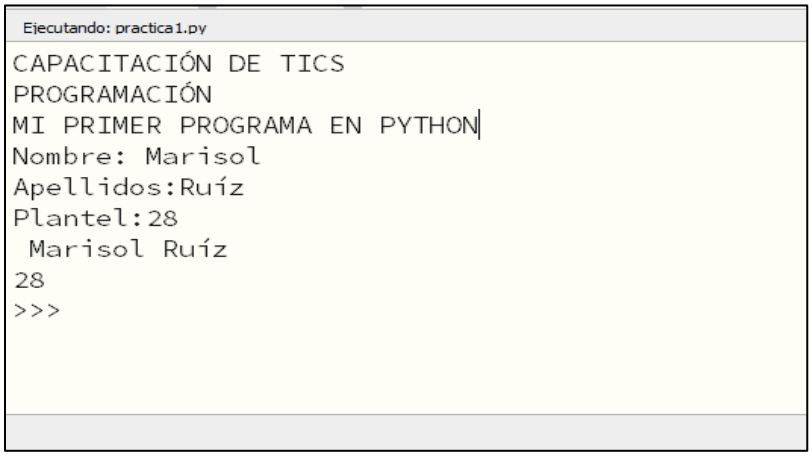
Desarrollo:

1. Realiza un programa en Python que permita imprimir mensajes con la función print() y dar entrada de datos con la función input(), así como agregar comentarios.
2. Nota: en caso de no contar con dispositivo digital realiza la práctica en tu libreta.
3. Abre el editor de programación el sugerido por tu Docente.
4. Escribe el código del programa con el apoyo de tu Docente de clase.

practica1.py

```
1  # Primer programa en Python
2  print("CAPACITACIÓN DE TICS")
3  print("PROGRAMACIÓN")
4  print("MI PRIMER PROGRAMA EN PYTHON")
5  nombre = input("Nombre:")
6  apellidos = input("Apellidos:")
7  plantel = input("Plantel:")
8  print(nombre, apellidos)
9  print(plantel)
```

5. Al terminar el código guarda el archivo en tu carpeta con la siguiente estructura: Practica01, siglas de tu nombre, grupo y turno. Ejemplo: **"Practica1_GEHM5B"** son las siglas de tu nombre, 5 es el semestre y B el grupo. En caso haberlo realizado en tu libreta, entrégalo a tu Docente para su revisión.
6. A continuación, presiona el botón de ejecutar y verifica que la salida sea la correcta.



7. **CONCLUSIONES:** Al finalizar la práctica, con tus palabras contesta en tu libreta las siguientes preguntas:

Menciona que instrucciones utilizaste en la práctica y escribe su sintaxis.

Instrumento de Evaluación LISTA DE COTEJO Práctica 1. Mi primer programa					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s)**		
Producto: Primer programa.			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente			**Firma del Docente**		
VALOR DEL REACTIVO	**CARACTERÍSTICAS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
2	Codifica de manera correcta y respeta la estructura del programa.				
1	Edita el comentario correctamente.				
1.5	Aplica la función print() adecuadamente.				
1.5	Aplica la función input() adecuadamente.				
2	Ejecuta el programa correctamente.				
1	Guarda el archivo con la nomenclatura solicitada.				
1	Entrega en tiempo y forma el programa.				
	CALIFICACIÓN				

Referencias

ADR(2021). *La función Print() en Python.* https://www.adrformacion.com/knowledge/programacion/2___print__.html

AWS. (s.f.). *¿Qué es Python? | Guía de Python para principiantes de la nube.* Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/python/

Codigofuente(2021). *Operadores en Python.* https://www.codigofuente.org/operadores-en-python/?vlogger_serie_in=1122

Fácil, P. (2022). *Las constantes en Python. Blog Programación Fácil .* https://programacionfacil.org/blog/las-constantes-en-python/

J2. (2022, 16 enero). *Variables en Python - Cómo asignar y modificar variables.* https://j2logo.com/python/tutorial/variables-python/#variables-intro

Peña MJ (2022). *Palabras reservadas en Python y su significado.* https://eiposgrados.com/blog-python/palabras-reservadas-python/

Python Software Foundation (2020). *Suport.* https://www.python.org/

Sintes B.(2018). *Entrada por teclado: la función input().* https://www.mclibre.org/consultar/python/lecciones/python-entrada-teclado.html

Unypyton (2020). *Tipos de datos y Variables en Python.* https://unipython.com/tipos-datos-variables-python/

Lectura 6. Condicionales

Instrucciones: Realiza la siguiente lectura subrayando las ideas principales, al finalizar completa la actividad sugerida dependiendo de las instrucciones de tu Docente.

Cada vez que realizamos una acción o resolvemos algún problema podemos hacerlo de forma secuencial, sin embargo, en muchos de los casos, estos requieren de decisiones, donde podemos encontrarnos con dos o más resultados.

En otras palabras, a diferencia de la estructura secuencial donde las instrucciones se siguen de manera consecutiva y lógica, ya que solo son definidas por un solo flujo de datos, la estructura condicional presenta bifurcaciones con una o más instrucciones que se ejecutarán en caso de que la condición o comparación sean verdadera o falsa.

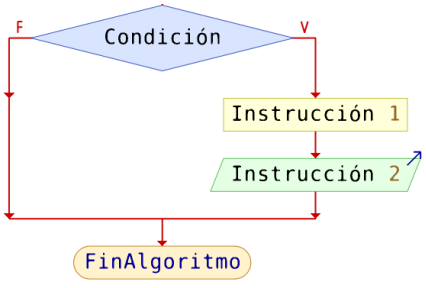
"Una instrucción (o estructura) condicional o de selección es aquella que establece qué instrucciones deben de ejecutarse o no, en función del valor de una condición." (Grupo Docente ISCyP, s/f.)

La estructura condicional se divide en:

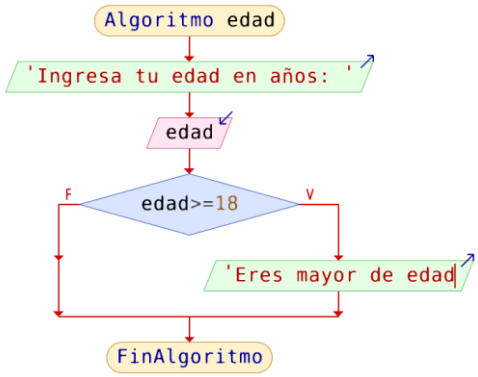
- Condicionales simples
- Condicionales dobles
- Condicionales múltiples

Nota: Es recomendable practicar con cada uno de los ejemplos o bien proponer algunos que considere, para consolidar el aprendizaje esperado.

Condicionales simples (sentencia if):
Este tipo de estructura condicional ejecuta una instrucción o un bloque de instrucciones siempre que la condición sea verdadera, pero no ejecutará ninguna instrucción si la condición es falsa.

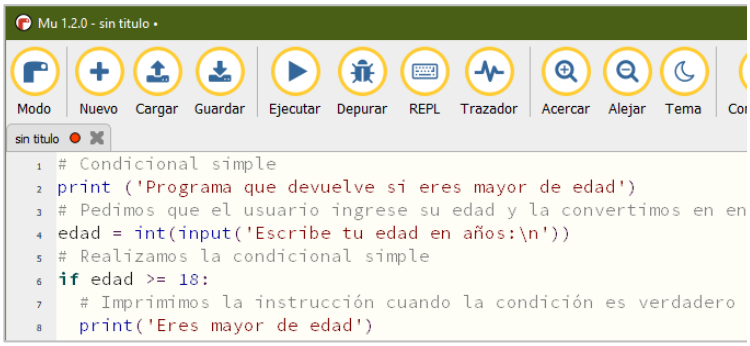


Ejemplo de diagrama de flujo:



En Python esta estructura se lleva a cabo mediante la palabra reservada **if.**

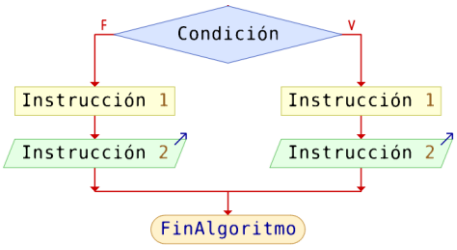
Código Python



Resultado

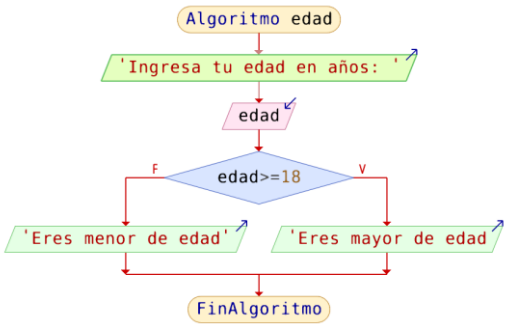
```
Programa que devuelve si eres mayor de edad
Escribe tu edad en años:
18
Eres mayor de edad

Process finished with exit code 0
```

Condicionales dobles (sentencia if – else):
En el caso de las condicionales dobles, si la condición es verdadera ejecutará una instrucción o un bloque de instrucciones, en tanto que si no se cumple o es falsa ejecutará otra instrucción u otro bloque de instrucciones.



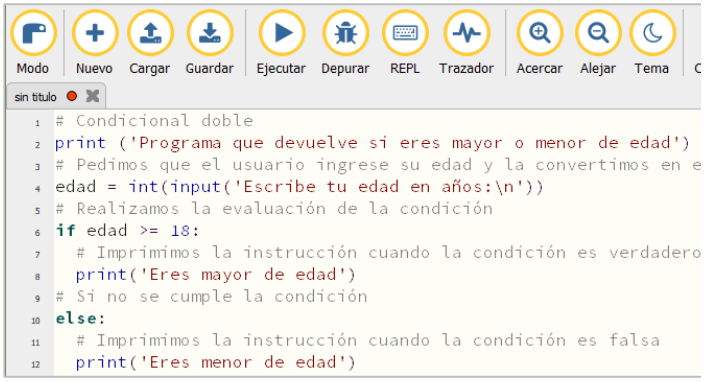
Ejemplo:

Diagrama de flujo:



En Python se compone de las palabras reservadas **if** y **else**. Es decir, a una estructura condicional simple le agregamos la palabra reservada **else** seguida del bloque de instrucciones que queremos ejecutar en el caso de que sea falsa la condición.

Código en Python



Resultado

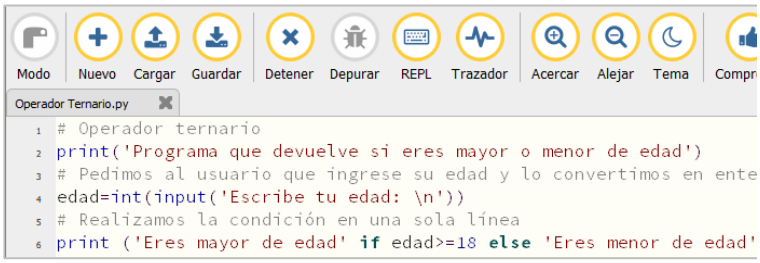
```
Programa que devuelve si eres mayor o menor de edad
Escribe tu edad en años:
17
Eres menor de edad

Process finished with exit code 0
```

Operador ternario:

> "El **operador ternario** o **ternary operator** es una herramienta muy potente que muchos lenguajes de programación tienen. En Python se trata de una cláusula if, else que se define en una sola línea y puede ser usado, por ejemplo, dentro de un print ()." (De Python, E. L., n.d.).

Su sintaxis es:
[código si es verdadero] **if** [condición] **else** [código si es falso]

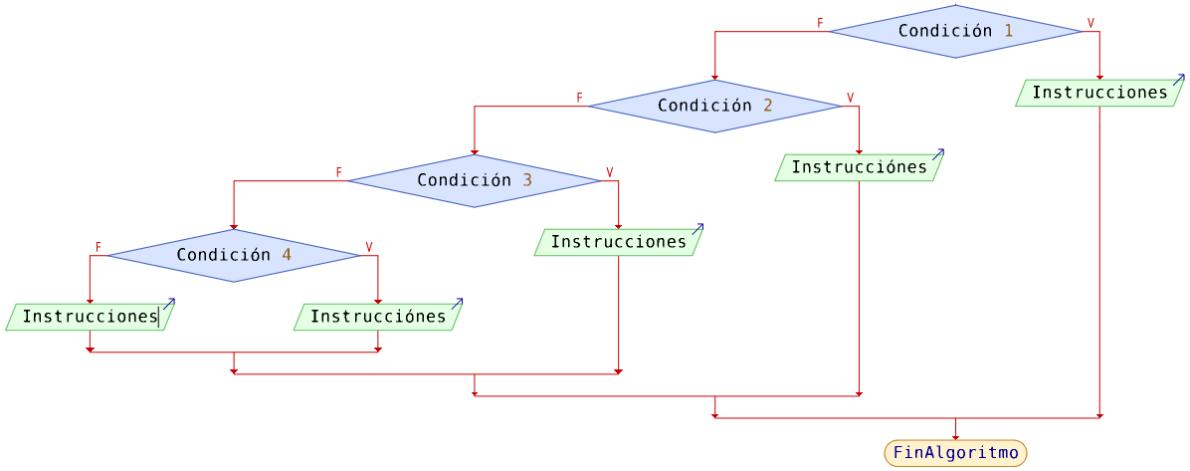
Código de Python:



Resultado:

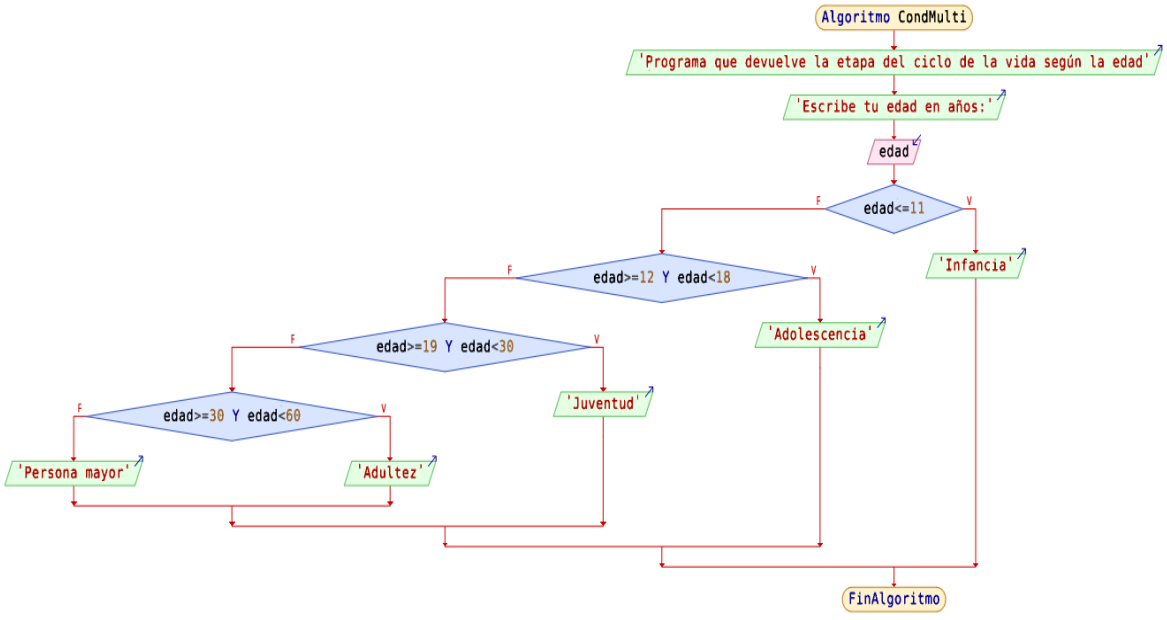
```
Programa que devuelve si eres mayor o meno
Escribe tu edad:
15
Eres menor de edad
```

Condicionales múltiples (sentencia if – elif – else):
El anidamiento se refiere a que podemos encontrar estructuras condicionales dentro de otras estructuras condicionales como bloque de instrucciones, es decir, que, permite que encadenemos varias condiciones.



Por ejemplo:

Diagrama de flujo:



Para esto Python contrae el anidamiento SiNo-Si o **else-if** en **elif** y la última contradicción o valor falso de la condición se utiliza la palabra **else**.

Código de Python:


```python
# Condicional múltiple
"""Infancia (0 - 11 años); Adolescencia (12 - 18 años); Juventud (14 - 26 años);
Adultez (27- 59 años); Persona Mayor (60 años o mas)"""
print("Programa que devuelve la etapa del ciclo de la vida según la edad")
# Pedimos que el usuario ingrese su edad y la convertimos en entero
edad = int(input("Escribe tu edad en años:\n"))
# Realizamos la evaluación de la condición 1
if edad <= 11:
    # Imprimimos la instrucción cuando la condición 1 es verdadera
    print("Infancia")
# Si no se cumple la condición 1, realizamos la condición 2
elif edad >= 12 and edad < 18:
    # Imprimimos la instrucción cuando la condición 2 es verdadera
    print("Adolescencia")
# Si no se cumple la condición 2, realizamos la condición 3
elif edad >= 19 and edad < 30:
    # Imprimimos la instrucción cuando la condición 3 es verdadera
    print("Juventud")
# Si no se cumple la condición 3, realizamos la condición 4
elif edad >= 30 and edad < 60:
    # Imprimimos la instrucción cuando la condición 4 es verdadera
    print("Adultez")
# Si no se cumple la condición 4
else:
    # Ejecutamos la última instrucción o bloque de instrucciones
    print("Persona mayor")
```

Resultado:

```
Programa que devuelve la etapa del ciclo de la vida según la edad
Escribe tu edad en años:
58
Adultez
```

Puedes practicar con los códigos dados en los ejemplos de esta lectura a través de los siguientes links, para hacerlo debes estar registrado en replit.com

Material disponible para la guía de Programación	
Condicional simple - https://replit.com/join/gzeiberrcj-miltonhernande7	

Material disponible para la guía de Programación	
Condicional doble - https://replit.com/join/egwndxpqyw-miltonhernande7	

Material disponible para la guía de Programación	
Condicional múltiple - https://replit.com/join/dapgywiztv-miltonhernande7	

Material disponible para la guía de Programación	
Aprende a crear un menú de opciones - https://replit.com/join/tnyttukyps-miltonhernande7	

Práctica 2 – En la búsqueda del código

Propósito: Corrige en un equipo de 4 personas el siguiente código en Python que imprime un menú de las 4 operaciones aritméticas básicas y realiza la operación, al terminar prográmalo en Python (computadora o dispositivo móvil) y socialízalo en plenaria para su Evaluación.

```
# Imprimir el menú de operaciones aritméticas
print("""
Opciones para elegir:
1: Suma
2: Resta
3: Multiplicación
4: División
""")

# Pedir al usuario elija una opción del menú
op = Int(Input('Ingrese el número de la opción deseada:\n'))

# Evaluamos la condición 1
if op = 1:
    # Pedir los datos que se necesitan
    num1 = Int(Input('Escribe el primer número:\n'))
    num2 = Int(Input('Escribe el segundo número:\n'))
    # Realizar la operación
    print(f'La suma es {Num1 + Num2}')

# Evaluamos la condición 2
else
    if op = 2:
    # Pedir los datos que se necesitan
    num1 = Int(Input('Escribe el primer número:\n'))
    num2 = Int(Input('Escribe el segundo número:\n'))
    # Realizar la operación
    print(f'La resta es {Num1 - Num2}')

# Evaluamos la condición 3
elif op = 3:
    # Pedir los datos que se necesitan
    num1 = Int(Input('Escribe el primer número:\n'))
    num2 = Int(Input('Escribe el segundo número:\n'))
    # Realizar la operación
    print(f'La multiplicación es {Num1 * Num2}')

# Evaluamos la condición 4
elif op = 4:
    # Pedir los datos que se necesitan
    num1 = Int(Input('Escribe el primer número:\n'))
```

```
    num2 = Int(Input('Escribe el segundo número:\n'))
    # Realizar la operación
    print(f'La división es {Num1 / Num2}')

# Evaluamos si no se cumple ninguna condición
else
    print('La opción elegida no es válida')
```

Instrumento de Evaluación LISTA DE COTEJO Práctica 2 En la búsqueda del código					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s)**		
Producto: Práctica 4 Listas			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente			**Firma del Docente**		
REACTIVO	**CRITERIOS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Utiliza un entorno de programación (IDE) para diseñar el programa solicitado.			10	
2	Reconoce el entorno de trabajo del IDE ya sea en computadora o dispositivo móvil			10	
3	Conoce y comprende la sintaxis y la semántica de las construcciones en Python			20	
4	Realiza programas de aplicación en Python aplicándolo a la solución de problemas reales			20	
5	El programa se ejecuta sin presentar fallos			20	
6	Trabaja de manera colaborativa con sus compañeros de equipo.			10	
7	Entrega en tiempo y forma la práctica cumpliendo con lo solicitado en la práctica.			10	
	CALIFICACIÓN				

Fuentes:

Bustamante, S. J. (2021, January 1). Sentencias If, Elif y Else en Python. freecodecamp.org. https://www.freecodecamp.org/espanol/news/sentencias-if-elif-y-else-en-python/

De Python, E. L. (n.d.). if en Python. El Libro De Python. Retrieved April 10, 2023, from https://ellibrodepython.com/if-python

Grupo Docente ISCyP. (n.d.). Estructuras de Control Condicional. Webs.um.es. Retrieved March 20, 2023, from https://webs.um.es/ldaniel/iscyp17-18/12-estructuraCondicional.html

if ... elif ... else. (n.d.). Mclibre.org. Retrieved March 29, 2023, from https://www.mclibre.org/consultar/python/lecciones/python-if-else.html

Lectura 7. Estructuras repetitivas

Instrucciones: Realiza la siguiente lectura, anotando los puntos más importantes para realizar cada uno de los ejemplos propuesto.

Probablemente lo más poderoso de las computadoras es que pueden repetir cosas una y otra vez muy rápidamente. Hay varias formas de repetir cosas en Python, la más común es mediante las *estructuras repetitivas*, también conocidas como *bucles* o *ciclos.*

Python tiene dos estructuras cíclicas:

1. El bucle while
2. El bucle for

El bucle *for* toma una colección de elementos y ejecuta un bloque de código una vez para cada elemento de la colección. Por el contrario, el bucle while se ejecuta mientras una determinada condición sea verdadera.

7.1 Estructura repetitiva while

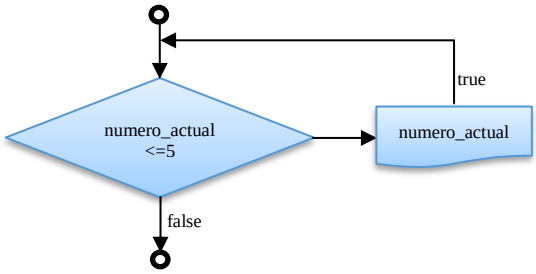
```
while <condición>:          # test del ciclo
<código a ejecutar>         # cuerpo del ciclo
else:                       # else opcional
<código a ejecutar>         # Se ejecuta si no sale del ciclo con un break
```

7.1.1 La estructura while en acción

Puede usar un bucle while para contar una serie de números. Por ejemplo, el siguiente ciclo while cuenta de 1 a 5:

while 1.py

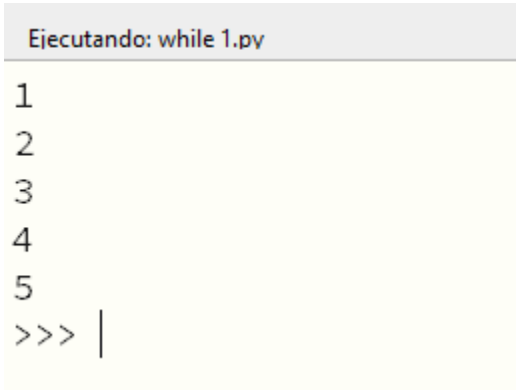
```
1  numero_actual = 1
2  while numero_actual <= 5:
3      print(numero_actual)
4      numero_actual+=1
```



En la primera línea, comenzamos a contar desde 1 estableciendo el valor de la variable ***numero_actual*** en 1. El ciclo *while* se configura para seguir ejecutándose siempre que el valor de ***numero_actual*** sea menor o igual a 5. El código dentro del ciclo imprime el valor de *numero_actual* y luego suma 1 a ese valor con

***numero_actual**+=1*. (El operador += es una forma abreviada de ***numero_actual**=**numero_actual**+1*).

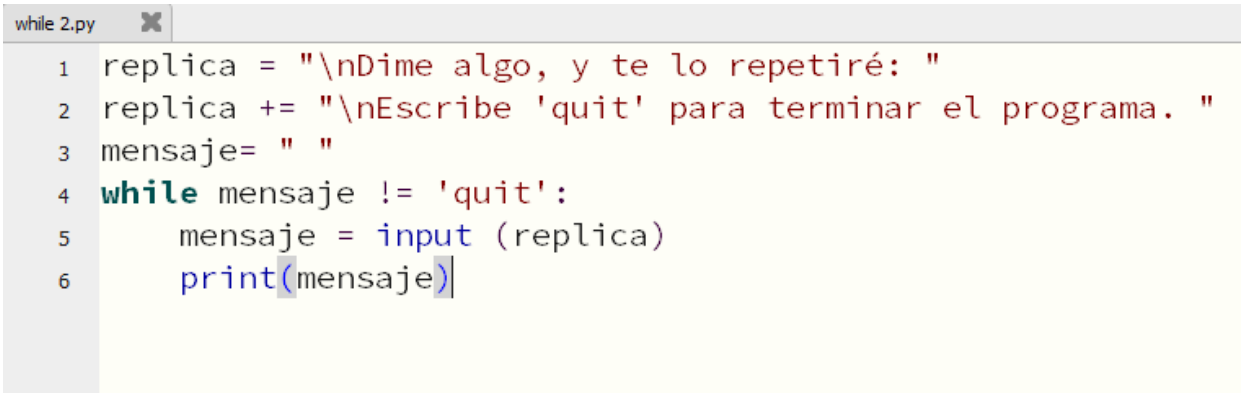
Python repite el bucle siempre que la condición *numero_actual<=5* sea verdadera. Debido a que 1 es menor que 5, Python imprime 1 y luego incrementa 1, lo que hace que el número actual sea 2. Como 2 es menor que 5, Python imprime 2 e incrementa 1 nuevamente, lo que hace que el número actual sea 3, y así sucesivamente. Una vez que el valor de *numero_actual* es mayor que 5, el ciclo deja de ejecutarse y el programa finaliza:



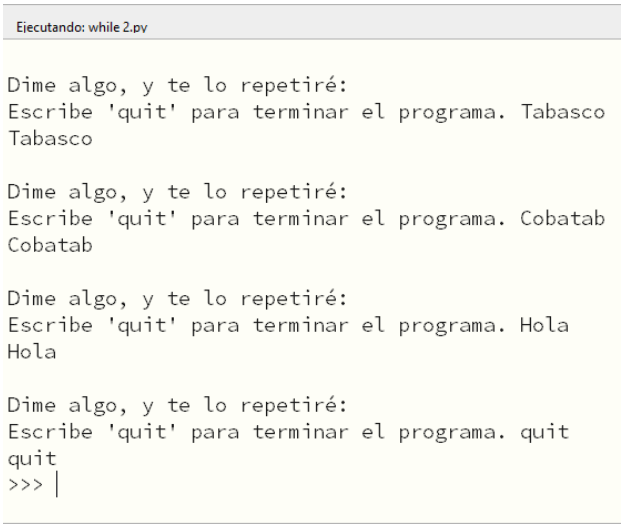
Es muy probable que los programas que usas todos los días contengan ciclos while. Por ejemplo, un juego necesita un ciclo *while* para seguir ejecutándose todo el tiempo que desee seguir jugando y, por lo tanto, puede dejar de ejecutarse tan pronto como le pida que lo abandone.

Los programas no serían divertidos de usar si dejaran de ejecutarse antes de que se lo indiquemos o si siguieran ejecutándose incluso después de que quisiéramos salir, por lo que los ciclos while son muy útiles.

Veamos este ejemplo que se ejecuta todo el tiempo que el usuario desee, mientras no ingrese un valor de salida.



La ejecución del programa sería así:



En el ejemplo anterior, hicimos que el programa realizara ciertas tareas mientras una condición determinada era verdadera. Pero ¿qué pasa con los programas más complicados en los que muchos eventos diferentes pueden hacer que el programa deje de ejecutarse? Por ejemplo, en un juego, varios eventos diferentes pueden terminar el juego.

Cuando el jugador se queda sin barcos, se acaba el tiempo o las ciudades que se suponía que debían proteger son destruidas, el juego debería terminar.

Debe terminar si ocurre cualquiera de estos eventos. Si pueden ocurrir muchos eventos posibles para detener el programa, tratar de probar todas estas condiciones en una instrucción while se vuelve complicado y difícil. Para un programa que debe ejecutarse siempre que se cumplan muchas condiciones, puede definir una variable que determine si todo el programa está activo o no.

Esta variable, llamada *bandera*, actúa como una señal para el programa. Podemos escribir nuestros programas para que se ejecuten mientras el indicador esté establecido en *True* y dejen de ejecutarse cuando cualquiera de varios eventos establezca el valor del indicador en *False*. Como resultado, nuestra declaración *while* general necesita verificar solo una condición: si la bandera es actualmente *True* o no. Luego, todas nuestras otras pruebas (para ver si ha ocurrido un evento que debería establecer el indicador en Falso) se pueden organizar ordenadamente en el resto del programa.

Agreguemos una bandera al ejemplo anterior. Esta bandera, la identificamos con la variable *active* (aunque puede llamarlo como quiera), monitoreará si el programa debe continuar ejecutándose o no:

while 3.py

```
replica = "\nDime algo, y te lo repetiré: "
replica += "\nEscribe 'quit' para terminar el programa\n"
active = True
while active:
    mensaje = input(replica)

    if mensaje == 'quit':
        active = False
        print ("Fin del programa")
    else:
        print(mensaje)
```

Este programa tiene la misma salida que el ejemplo anterior donde se colocó la prueba condicional directamente en la instrucción *while*. Pero ahora que tenemos un indicador para verificar si el programa general está en un estado activo, sería fácil agregar más pruebas (como declaraciones *elif*) para eventos que deberían causar que la variable *active* se convierta en *False*.

Esto es útil en programas complicados como juegos en los que puede haber muchos eventos que deberían hacer que el programa deje de ejecutarse. Cuando cualquiera de estos eventos hace que la bandera activa se vuelva *False*, el bucle principal del juego se cerrará, se puede mostrar un mensaje de “Game Over” y se le puede dar al jugador la opción de volver a jugar.

La ejecución del programa sería:

Ejecutando: while 3.py

```
Dime algo, y te lo repetiré:
Escribe 'quit' para terminar el programa
Hola
Hola

Dime algo, y te lo repetiré:
Escribe 'quit' para terminar el programa
Verano
Verano

Dime algo, y te lo repetiré:
Escribe 'quit' para terminar el programa
quit
Fin del programa
>>>
```

7.1.2 Usando break para salir de un ciclo

Para salir de un ciclo *while* inmediatamente sin ejecutar ningún código restante en el ciclo, independientemente de los resultados de cualquier prueba condicional, use la instrucción *break*.

La instrucción *break* dirige el flujo de su programa; puede usarlo para controlar qué líneas de código se ejecutan y cuáles no, por lo que el programa solo ejecuta el código que desea, cuando lo desea.
Por ejemplo, considere un programa que le pregunta al usuario sobre los lugares que ha visitado. Podemos detener el ciclo while en este programa llamando a *break* tan pronto como el usuario ingrese el valor *'quit'*:

while4.py

```
encuesta = "\nEscribe el nombre de una ciudad que hayas visitado: "
encuesta += "\n(Escribe 'quit' cuando hayas terminado)\n"

while True:
    ciudad = input(encuesta)

    if ciudad == 'quit':
        break
    else:
        print("Me encantaría ir a " + ciudad.title() + "!")
```

Un ciclo que comienza con *while True* se ejecutará para siempre a menos que llegue a una declaración de ruptura. El bucle en este programa continúa pidiéndole al usuario que ingrese los nombres de las ciudades en las que ha estado hasta que ingrese '*quit*'. Cuando se ingresa '*quit*', se ejecuta la declaración de interrupción, lo que hace que Python salga del ciclo.

```
Ejecutando: while4.py

Escribe el nombre de una ciudad que hayas visitado:
(Escribe 'quit' cuando hayas terminado)
Francia
Me encantaría ir a Francia!

Escribe el nombre de una ciudad que hayas visitado:
(Escribe 'quit' cuando hayas terminado)
Mérida
Me encantaría ir a Mérida!

Escribe el nombre de una ciudad que hayas visitado:
(Escribe 'quit' cuando hayas terminado)
Comalcalco
Me encantaría ir a Comalcalco!

Escribe el nombre de una ciudad que hayas visitado:
(Escribe 'quit' cuando hayas terminado)
quit
>>>
```

Nota: Puedes usar la instrucción break en cualquiera de los bucles de Python. Por ejemplo, podría usar break para salir de un bucle *for* que está trabajando en una lista o un diccionario.

7.1.2 Usando continue en un ciclo

En lugar de salir de un bucle por completo sin ejecutar el resto de su código, puedes usar la instrucción *continue* para volver al principio del bucle en función del resultado de una prueba condicional. Por ejemplo, considere un ciclo que cuenta del 1 al 10 pero imprime solo los números impares en ese rango:

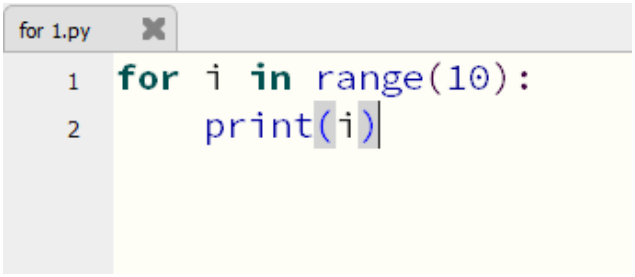
```
while 5.py
1  numero_actual = 0
2  while numero_actual < 10:
3      numero_actual+=1
4      if numero_actual %2 == 0:
5          continue
6      print(numero_actual)
```

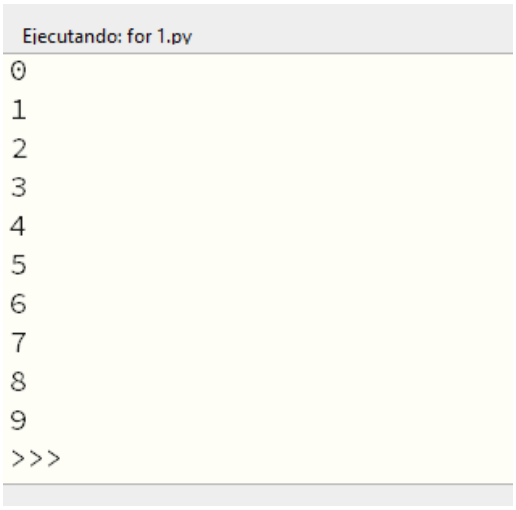
```
Ejecutando: while 5.py
1
3
5
7
9
>>>
```

Ejecución del programa usando *continue*:

7.2 Estructura repetitiva *for*

La estructura de repetición maneja todos los detalles de la repetición controlada por contador. El bucle *for* es un iterador de secuencia genérico en Python: puede recorrer los elementos en cualquier objeto de secuencia ordenado. La declaración *for* funciona en cadenas, listas (arreglos o vectores), tuplas, y otros iterables incorporados. Para ilustrar el poder de *for*, analicemos este programa:





La función *range* () puede tomar uno, dos o tres argumentos. Si solo le pasamos un argumento a la función, ese argumento, llamado *fin*, es un dato que tiene como valor "una unidad mayor al límite superior" (o valor más alto) de la secuencia.

Si pasamos tres argumentos, los primeros dos argumentos indican el *inicio* y *fin*, respectivamente, y el tercer argumento, llamado *incremento*, es el valor del incremento. La secuencia producida por una llamada a *rango* con un valor de *incremento* progresa de principio a fin en múltiplos del valor de incremento. Estas variables también se llaman *"variables de control"*. Las siguientes tres llamadas a *range()* producen la misma secuencia del ejemplo anterior.

range (10)
Se invoca la función con un solo argumento, Python lo entiende como valor final, los valores predeterminados de valor inicial e incremento serán 1.

range (0, 10)
Se invoca la función con dos argumentos: valor inicial y valor final, el incremento predeterminado será 1.

range (0, 10, 1)
Se invoca con tres argumentos: valor inicial, valor final e incremento.

En el siguiente ejemplo pasaremos a la función *range()* los tres argumentos en el orden que este lo interpreta, en esta ocasión para que nos devuelva una secuencia con decremento:

for 2.py

```
1  for i in range(10, 0, -1):
2      print(i)
3
```

Ejecutando: for 2.py

```
10
9
8
7
6
5
4
3
2
1
>>>
```

En este ejemplo las variables de control enviadas a la función *range()* son las siguientes: valor inicial: 10; valor final: 0; incremento: -1. Si observas, el valor final que se utilizó fue 0, pero en pantalla se imprimió hasta el 1, esto es porque Python considera este argumento con un valor superior al límite indicado, como mencionamos anteriormente.

Ahora veamos un ejercicio un poco más complejo, pero si observas, utiliza pocas líneas de código.

for 3.py

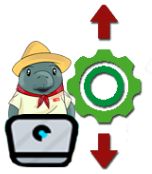
```
1  num = int(input("Ingresa un número para generar su tabla de multiplicar: \n"))
2
3  #usamos un bucle for para generar la tabla de multiplicar del número ingresado
4  for i in range(1, 11):
5      print(num, "x", i, "=", num*i)
```

Este código solicita al usuario un número a través de la función *input(),* lo convierte a un entero usando la función *int()* y luego usa un bucle *for* para imprimir la tabla de multiplicar del número ingresado. La variable i en el bucle *for* representa el multiplicador de la tabla de multiplicar y va desde 1 hasta 10.

Ejecutando: for 3.py

```
Ingresa un número para generar su tabla de multiplicar:
6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
>>>
```

Práctica 3: Calculadora de comisiones de ventas



Instrucciones: De manera individual resuelve el siguiente problema propuesto: (**Calculadora de comisiones de ventas**)

Una empresa paga a sus vendedores mediante comisiones. Los vendedores reciben $200 base por semana, más el 9% de sus ventas brutas durante esa semana. Por ejemplo, un vendedor que vende $5,000 de mercancía en una semana, recibe $200 más el 9% de 5,000, o un total de $650. Usted acaba de recibir una lista de los artículos vendidos por cada vendedor. Los valores de estos artículos son los siguientes:

Articulo	Valor
1	$239.99
2	$129.75
3	$99.95
4	$350.89

La siguiente tabla muestra un ejemplo de cómo se calcularía el pago del salario base más las comisiones de un vendedor en base a la lista de artículos que facturó.

Código. Articulo	Cantidad	Suma:
1	3	$719.97
3	6	$599.70
2	5	$648.75
4	1	$350.89
3	2	$199.90
	SUMA	$2,519.21
	COMISIÓN (9%)	$226.72
	SALARIO BASE	$200.00
	BASE + COMISIÓN	**$426.72**

Desarrolle una aplicación en Python que calcule la comisión a pagar a un vendedor. No hay límites en cuanto al número de artículos vendidos. Para resolver este problema debes usar el ciclo While y una de las técnicas de salida de este.

Ejemplo de ejecución:

```
Escribe el código del articulo (1, 2, 3 o 4), (otro número para terminar): 1
Indica la cantidad de piezas vendidas del (articulo# : 1): 5

Escribe el código del articulo (1, 2, 3 o 4), (otro número para terminar): 2
Indica la cantidad de piezas vendidas del (articulo# : 2): 9

Escribe el código del articulo (1, 2, 3 o 4), (otro número para terminar): 3
Indica la cantidad de piezas vendidas del (articulo# : 3): 2

Escribe el código del articulo (1, 2, 3 o 4), (otro número para terminar): 4
Indica la cantidad de piezas vendidas del (articulo# : 4): 7

Escribe el código del articulo (1, 2, 3 o 4), (otro número para terminar): 6
El salario del vendedor es $ 652.0636999999999
>>>
```

Instrumento de Evaluación LISTA DE COTEJO Práctica 3 Calculadora de comisiones de ventas					
DATOS GENERALES :					
Nombre(s) del estudiante(s)			**Matrícula(s)**		
Producto: Práctica 7			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente			**Firma del Docente**		
REACTIVO	**CRITERIOS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Usa el ciclo while como ciclo principal			**2**	
2	Define una condición clara para terminar el ciclo			**2**	
3	Declara las variables necesarias y las utiliza todas.			**2**	
4	Realiza correctamente el cálculo de los ingresos			**2**	
5	Propone una alternativa de solución fiable.			**1**	
6	Entrega en tiempo y forma			**1**	
	CALIFICACIÓN				

Referencias.

Matthes, Eric (2016) Python Crash Course San Francisco No Starch Press, Inc. p.122-126 ISBN-10: 1-59327-603-6

Lectura 8. Aprendamos sobre Listas en Python

Instrucciones: Realiza la siguiente lectura y algunos ejemplos en Python para ir aprendiendo sobre las listas y sus métodos. Al finalizar con la guía de tu Docente realiza la práctica solicitada.

Una lista en Python es una colección de datos o un tipo de datos parecido a un arreglo "arrays" o matriz de otros lenguajes de programación, una lista es una estructura de datos flexible en donde se almacena valores numéricos, tipo cadena, booleanos y otras listas. Las listas se delimitan **nombreLista** un signo **=** por **[]** y los elementos se separan con comas.

nombreLista = [] #declara una lista

Lista con n=5 elementos

nombreLista =
Indice (posición)

10	20	30	Hola	Python
0	1	2	3	n-1 = 4

Ejemplo en python:

Lista = [] #Declara una Lista Vacía
Lista = [1, 2, 3, 4] #declara e inicializa una lista con elementos del mismo tipo de datos
lista = [10,20, 30, "Hola", "Python"] #Lista con diversos tipos de datos
Lista =["Python", 3.1416, 50, [1, 2]] a lista dentro de otra lista está anidada

Propiedades de la lista

- Son ordenadas, mantienen el orden en el que han sido definidas 0, 1, 2, 3, …, n-1
- Pueden ser formadas por varios tipos de datos
- Pueden ser indexadas con [i], un índice que va desde **0** a **n-1** siendo **n** el tamaño de la lista
- Se pueden anidar, es decir, contener una lista dentro de la otra.
- Son mutables, ya que sus elementos pueden ser modificados.
- Son dinámicas, ya que se pueden añadir o eliminar elementos.

¿Cómo crear una lista en Python y Visualizarla?

Ejemplo 1: Crear una lista de 5 elementos y visualiza la lista con todos sus elementos

print (lista) → visualiza la lista con todos sus elementos

lista 1.py

```
1 #Módulo de Programación
2 #Crear una lista con 5 elementos
3
4 lista = [10,20, 30, "Hola", "Python"] #declara e inicializa una lista con 5 elementos
5 print(lista) #imprime la lista con todos sus elementos
```

Ejecutando: lista 1.py

```
[10, 20, 30, 'Hola', 'Python']
>>>
```

Ejemplo 2: Crear una lista con 5 elementos y visualiza a cada elemento de acuerdo con su índice (posición del elemento) n=5

10	20	30	Hola	Python

Lista = Indice(posición)	0	1	2	3	4

print(nombreLista[index]) → visualiza el valor de acuerdo a su indice o posición

lista 2.py

```
#Módulo de Programación

lista = [10,20, 30, "Hola", "Python"] #declara e inicializa la lista con valores
print(lista[0]) #Visualiza el elemento del indice 0 = 10
print(lista[1]) #Visualiza el elemento del indice 1 = 20
print(lista[2]) #Visualiza el elemento del indice 2 = 30
print(lista[3]) #Visualiza el elemento del indice 3 = "Hola"
print(lista[4]) #Visualiza el elemento del indice 8 = Indice no existe
```

Ejecutando: lista 2.py

```
10
20
30
Hola
Python
>>>
```

Ejemplo 3: Crea una lista con 5 elementos y visualiza un elemento que no existe

lista 2.py

```
#Módulo de Programación

lista = [10,20, 30, "Hola", "Python"] #declara e inicializa la lista con valores
print(lista[0]) #Visualiza el elemento del indice 0 = 10
print(lista[1]) #Visualiza el elemento del indice 1 = 20
print(lista[2]) #Visualiza el elemento del indice 2 = 30
print(lista[3]) #Visualiza el elemento del indice 3 = "Hola"
print(lista[8]) #Visualiza el elemento del indice 8 = Indice no existe
```

Ejecutando: lista 2.py

```
10
20
30
Hola
Traceback (most recent call last):
  File "c:\users\karina\documents\2023a\listas - programacion\programas\lista 2.py", line 8, in <module>
    print(lista[8]) #Visualiza el elemento del indice 8 = Indice no existe
IndexError: list index out of range
>>>
```

Nota: *Al colocar un índice que no existe, el programa genera una Excepción de tipo* ***IndexError****: La lista está fuera de rango*.

Ejemplo 4: Visualizar elementos de una lista indicando *solo los elementos que se desea mostrar*, por medio de la siguiente sentencia >> print(listaDias[2:5]) →Lo señalado por [] indica la posición del índice inicial "2" a la posición del índice final "n" no mostrando el valor de n. Es decir, solo se visualiza el elemento del índice 2, 3 y 4.

lista_rango.py

```
# Modulo de programación

listaDias = ["lunes","martes","miercoles","jueves","viernes","sabado","domingo"]
#Visualiza solo los elementos del indice (2, 3, 4)no mostrando el n=5 elemento
print(listaDias[2:5])# del 2do al 4to elemento de la lista
```

Ejecutando: lista_rango.py

```
['miercoles', 'jueves', 'viernes']
>>>
```

Ejemplo 5: Imprime el último elemento.

En Python, los índices también pueden ser números negativos, contará de derecha a izquierda con índices negativos. Ejemplo obtener el último elemento de la lista >>> lista=[-1], el penúltimo elemento >>>lista=[-2], etc.

lista3.py

```
1  #Módulo de Programación
2
3  lista = [10,20, 30, "Hola", "Python"]
4  print(lista[-1]) #imprime el ultimo elemento de la lista
```

Ejecutando: lista3.py

```
Python
>>>
```

Ejemplo 6: Crear una lista y visualizar todos los elementos utilizando un ciclo for . La cantidad de ciclos se determina por el numero elementos que contenga la lista
En el for la variable “elemento” es una variable iteractiva que apoya en el conteo de cada ciclo

lista for.py

```
1  #Módulo de Programación
2
3  lista = [10,20, 30, "Hola", "Python", [1, 2, "Python3"], "etc."]
4  #Visualización de todos los elementos mediante un for
5  for elemento in lista: #En cada ciclo tengo un "elemento" de en lista
6      print("Elemento de la lista = ", elemento) #imprime el elemento de acorde al ciclo
```

Ejecutando: lista for.py

```
Elemento de la lista =  10
Elemento de la lista =  20
Elemento de la lista =  30
Elemento de la lista =  Hola
Elemento de la lista =  Python
Elemento de la lista =  [1, 2, 'Python3']
Elemento de la lista =  etc.
>>>
```

Ejemplo 7: Calcular la suma de los elementos de una lista

Lista for01.py

```
1  #Módulo de Programación
2  #Sumar los elementos de una lista
3  lista = [10,20,30,40,50,60,70]
4  count=0  #inicializamos un contador de los numeros de la lista
5  suma=0; #inicializamos el sumador de los numeros
6  for numero in lista:
7      count=count + 1 #incrementa el contador en cada bucle
8      suma= suma + numero  # por cada bucle o ciclo suma un valor de la lista
9      print("Numero ",count," = ", numero) #imprime el elemento de acorde al ciclo
10 print("El total de la suma de la lista es ", suma)
```

Ejecutando: Lista for01.py

```
Numero  1  =  10
Numero  2  =  20
Numero  3  =  30
Numero  4  =  40
Numero  5  =  50
Numero  6  =  60
Numero  7  =  70
El total de la suma de la lista es  280
>>>
```

Listas mutables

las listas son mutables porque pueden cambiar el orden de los elementos en una lista o reasignar un elemento en una lista. Cuando el operador corchete aparece en el lado izquierdo de una asignación, éste identifica el elemento de la lista que será asignado.

Ejemplo 8: Crea una lista de 5 elementos y reasigna un nuevo valor en el índice 2 de la lista.

listaMutable.py

```
1  #Módulo de Programación
2  #Lista mutable
3
4  lista = [10,20, 30, "Hola", "Python"] #declara e inicializa una lista con elementos
5  print(lista) #visualiza todos los elementos de la lista
6  lista[2]=100  #asigna un nuevo valor a la posición 2 de la lista
7  print (" reasignamos el valor de lista[2]=100 ")
8  print(lista)  #visualización de los elementos de la lista con la reasignación
```

Ejecutando: listaMutable.py

```
[10, 20, 30, 'Hola', 'Python']
 reasignamos el valor de lista[2]=100
[10, 20, 100, 'Hola', 'Python']
>>>
```

Métodos de listas

Python tiene 11 métodos para trabajar con listas, de los cuales se menciona los más importantes.

- **Insert** agrega un elemento en una posición o índice determinado.

Ejemplo 9: Utiliza el ciclo for para llenar una lista de n elementos numéricos mediante el método insert

nombrelista.insert(indice, elemento)→ Para insertar debe colocar el índice y el valor del elemento entre ().

lista_metodos.py

```
# Módulo Programación

lista=[] #declara e inicializa la lista vacia
n= int(input("¿Cuántos elementos tendrá la lista? ")) #ingresa a pantalla el numero de datos en la lista
for i in range(0,n):#introducción de cada elementos con for
    elemento= int(input("ingresa el valor : ")) #ingresa por cada ciclo un numero
    lista.insert(i,elemento) #Metodo insert con dos parametros(index,elemento)
print (lista) #imprime todos los elementos de la lista
```

Ejecutando: lista_metodos.py

```
¿Cuántos elementos tendrá la lista? 3
ingresa el valor : 34
ingresa el valor : 2
ingresa el valor : 7
[34, 2, 7]
>>>
```

Nota: En el ciclo **for i** in **range(0,n)** se define el índice del elemento inicial "0" hasta el índice del elemento que contiene la lista "n" .

- **append** agrega un nuevo elemento al final de una lista.

nombreLista.append(elemento) →Agrega al final de la lista el **elemento** señalado entre comillas si el elemento es de tipo texto, en caso contrario sin comillas.

Ejemplos:

Ejemplo 10: Crea una lista con los primeros 3 meses del año "enero, febrero, marzo" y agrega con el método append el mes de "abril".

lista_append01.py

```
#Módulo de Programación
#Metodo append en la lista "meses"

meses = ["Enero", "Febrero", "Marzo"]
print (meses)
#metodo append
print ("utilizando el método Append")
meses.append("Abril") #agrega al final de la lista un valor
print (meses)
```

Ejecutando: lista_append01.py

```
['Enero', 'Febrero', 'Marzo']
utilizando el método Append
['Enero', 'Febrero', 'Marzo', 'Abril']
>>>
```

Ejemplo 11: En una lista vacía insertar un nombre con el método append

lista_append02.py

```
#Módulo de Programación
#Metodo append para insertar un nombre

lista_nombre= [] #declara lista vacia
nombre=str(input("Ingresa un nombre : ")) #variable para ingresar el nombre en la pantalla
#metodo append
lista_nombre.append(nombre) #en la lista se asigan el nombre
print (nombre)# Imprime el nombre de la lista

```

Ejecutando: lista_append02.py

```
Ingresa un nombre : Karina
Karina
>>>
```

Nota: Para agregar varios nombres podemos utilizar la estructura de control de ciclo **for**

- **extend** toma una lista como argumento y agrega todos los elementos

nombreLista.extend([elemento1, elemento2, elemento 3])

Ejemplo:

Ejemplo 12: Crea una lista con los primeros 3 meses del año "enero, febrero, marzo" y agrega con el método extend los 3 siguientes meses "abril, mayo, junio".

lista_extend.py

```
1  #Módulo de Programación
2  #Metodo extend en la lista "meses"
3
4  meses = ["Enero", "Febrero", "Marzo"]
5  print (meses)
6  #metodo extend
7  print ("utilizando el método extend")
8  meses.extend(["Abril","Mayo","Junio"])
9  print (meses)
10
```

Ejecutando: lista_extend.py

```
['Enero', 'Febrero', 'Marzo']
utilizando el método extend
['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio']
>>>
```

- **sort** ordena los elementos de la lista de menor a mayor:

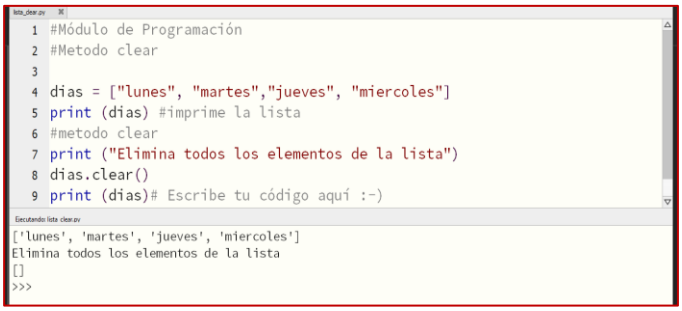
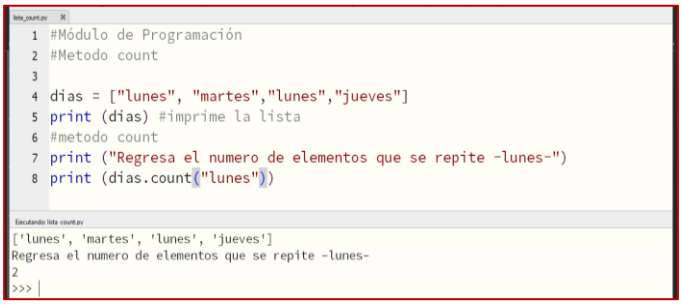
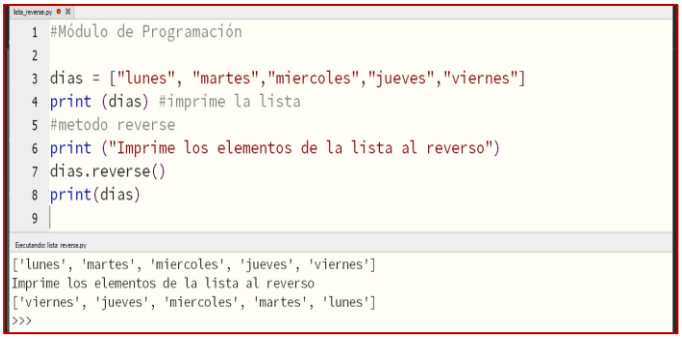
Ejemplo 13: Ordenar los elementos de menor a mayor.

lista_sort.py

```
1  #Módulo de Programación
2  #Metodo sort - ordena los elementos de menor a mayor
3
4  numeros = [5,3,7,9,0,1,4,8,6]
5  print (numeros) #imprime la lista
6  #metodo sort
7  print ("Ordena la lista con el método sort")
8  numeros.sort()
9  print (numeros)
10
```

Ejecutando: lista_sort.py

```
[5, 3, 7, 9, 0, 1, 4, 8, 6]
Ordena la lista con el método sort
[0, 1, 3, 4, 5, 6, 7, 8, 9]
>>>
```

Método	Función	Sintaxis	Ejemplo Visual
.clear	Elimina todos los elementos de la lista	**lista.clear**	 *Nota: Se eliminan todos los elementos de la lista, sin embargo, la lista aun existe y visualizarla con **print** aparecen los corchetes vacios.*
.index	Devuelve el indice del elemento que coloquemos como parámetro. Si el elemento no existe enviá un mensaje de ValueError.	**lista.index(elemento)**	
.count	Muestra el número de veces que se repite el elemento que usemos como parámetro.	**lista.count(elemento)**	
reverse	Invierte lo elementos de la lista.	**lista.reverse()**	

Eliminando elementos

Hay varias formas de eliminar elementos de una lista. Si sabes el índice del elemento que quieres, puedes

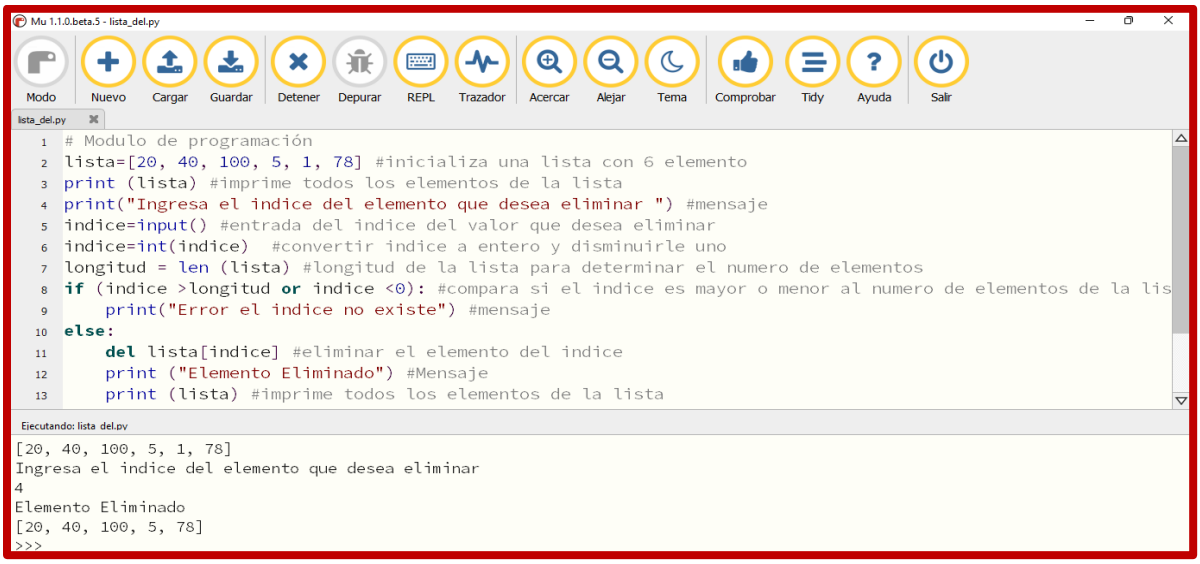
usar pop:

COMANDO	CONCEPTO	EJEMPLO
POP	Modifica la lista y regresa el elemento que fue removido. Si no provees un índice, la función elimina y retorna el último elemento.	**>>> lista = ['a', 'b', 'c']** **>>> x = lista.pop(1)** **>>> print(lista) ['a', 'c']** **>>> print(x) b**
DEL	Eliminar un elemento de la lista conociendo su index (índice)	**>>> lista = ['a', 'b', 'c']** **>>> del lista[1]** **>>> print(lista) ['a', 'c']**
REMOVE	Si sabes qué elemento quieres remover (pero no sabes el índice), puedes usar remove.	**>>> lista= ['a', 'b', 'c']** **>>> lista.remove('b')** **>>> print(lista) ['a', 'c']**

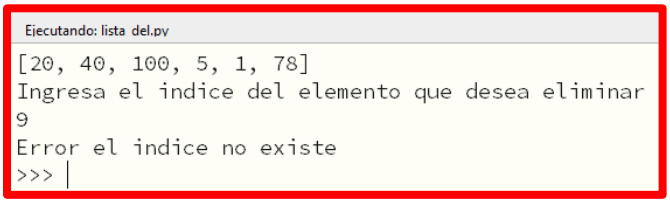
Ejemplo 14:

Eliminar un elemento de la lista conociendo su index y utilizando el operador del en caso de no existir la posición del valor mandar mensaje de "Error el índice no existe"

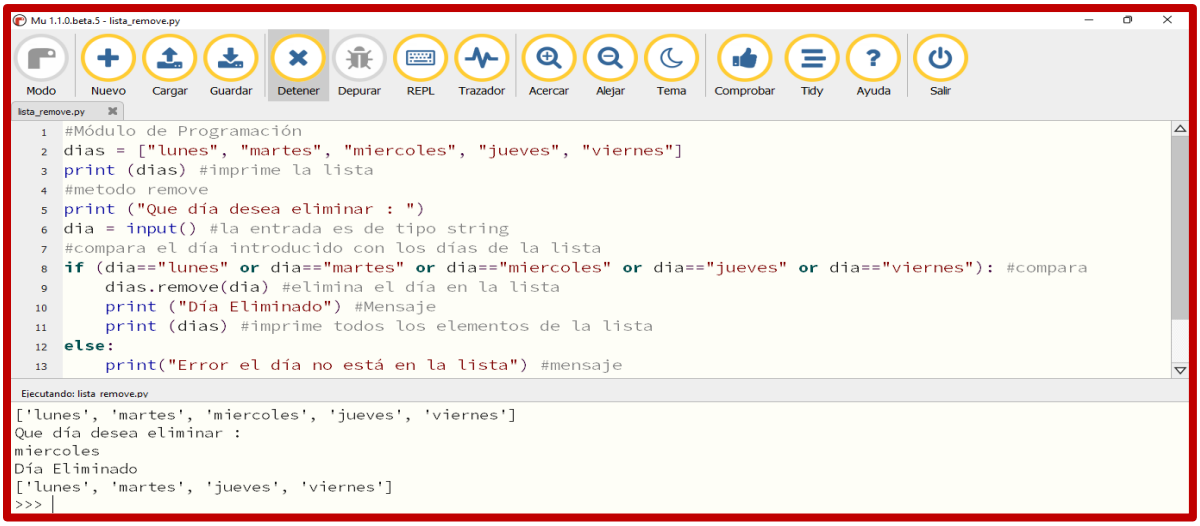
Lista =	20	40	100	5	1	78
Indice =	0	1	2	3	4	n-1 = 5



Si el índice colocado está fuera de rango



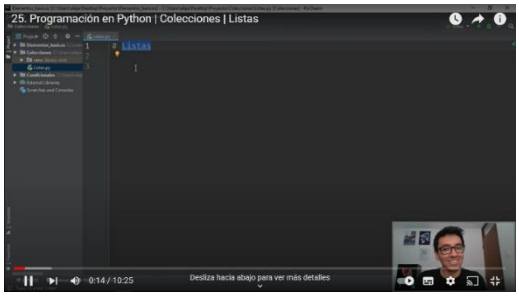
Ejemplo 15: Eliminar un valor de una lista, utilizando el operador remove



En caso de no existir el día en la lista

```
Ejecutando: lista remove.py
['lunes', 'martes', 'miercoles', 'jueves', 'viernes']
Que día desea eliminar :
Sabado
Error el día no está en la lista
>>>
```

Recurso Didáctico Sugerido



https://www.youtube.com/watch?v=xdCJa2QXmJ8

Recurso Didáctico Sugerido

https://www.youtube.com/watch?v=MHvrJhshEU0

Práctica 4 Manipulación de listas

Propósito: Crear una lista que almacena información la cual por medio de los métodos estudiados pueda manipularse para agregar y eliminar elementos de esta; así como hacer uso de las estructuras de control **for, case, if,** etc. En caso necesario.

Conocimientos:

- ✓ listas
- ✓ métodos para manipulación de listas
- ✓ Estructuras de control: if, case, for, while, etc.

Calcular las comisiones de ventas de "n" números de trabajadores, en donde los vendedores reciben $500 por semana, más el 12% de sus ventas brutas durante esa semana. Por ejemplo, si un vendedor vende $3500 de mercancía en una semana, recibe $500 más 12% de su comisión $420 de la venta, cobrando un total de $920.

Realice un programa en Python que calcule comisión y calcule el pago total por ventas y las guarde en una lista de pagoVentas.

El programa deberá mostrar el pago total del trabajador

- **venta semanal**
- **comisión de las ventas**
- **Pago total al vendedor**

Además, **el programa debe ser capaz de agregar un nuevo vendedor, eliminar y mostrar los pagos semanales de todos los vendedores.**

Instrucciones:

1. Realiza el código propio para generar la lista de pagosSemanales y utiliza de acuerdo con tu lógica de programación las sentencias de control que se ajusten a tu programa.
2. Considera en el programa agregando las diversas sentencias de control como: ciclos, condicionales, etc. para poder realizar las operaciones del programa.
3. Considera que el programa tome como base una lista vacía, deberá pedir la cantidad de **n** número de vendedores.
4. Deberá agregar y eliminar de al menos un vendedor
5. Deberá mostrar la venta, comisión por venta y pago total para 1 o varios trabajadores.
6. Deberá tener la opción para salir del programa.

Ejemplo:

```
-- MENU DE VENDEDOR --
1: AGREGAR VENDEDOR Y CALCULAR SU PAGO TOTAL
2: MOSTRAR PAGOS SEMANALES DE LOS VENDEDOR
3: ELIMINAR VENDEDOR Y SU VENTA SEMANAL
Elija una OPC
1
¿Cuántos Vendedores desea agregar ?
1
Ingresa la venta semanal del Vendedor
3500
Pago Semanal 500 + Comisión de ventas  420
El pago total al Vendedor:  920.0
 Desea continuar [S/N] :
```

NOTA: Considere que la solución depende en gran parte del uso de cada sentencia vista anteriormente en las lecturas, la aplicación de estas depende de tu capacidad de razonamiento y tu creatividad.

Instrumento de Evaluación LISTA DE COTEJO Práctica 4 Manipulación de Listas					
DATOS GENERALES					
Nombre(s) del estudiante(s)			**Matrícula(s)**		
Producto: Práctica 4 Listas			**Fecha**		
Materia:			**Periodo**		
Nombre del Docente			**Firma del Docente**		
REACTIVO	**CRITERIOS A CUMPLIR**	**VALOR OBTENIDO**		**CALIF**	**OBSERVACIONES Y/O SUGERENCIAS DE MEJORA**
		SI	**NO**		
1	Identifica y aplica la estructura de control de **ciclos y decisiones** para realizar la visualización de la lista de vendedores.			**2**	
2	Realiza los cálculos correspondientes para obtener la comisión de las ventas y el pago total del vendedor de manera correcta.			**2**	
3	Identifica y Aplica el método **insert** o **append** para insertar un elemento en la lista.			**2**	
4	Identifica y Aplica el método **remove** o **del** para eliminar un elemento de la lista.			**2**	
5	Visualiza la lista de pago de los vendedores.			**1**	
6	Muestra en pantalla la opción de continuar			**1**	
	CALIFICACIÓN				

Referencias

Listas en Python. (s/f). El Libro De Python. https://ellibrodepython.com/listas-en-python

MasterHeHeGar [@MasterHeHeGar]. (2013, julio 12). 30 - Ejercicio Práctico con Listas (Python). Youtube. https://www.youtube.com/watch?v=MHvrJhshEU0

Programación en Python | Colecciones | Listas. (2018, diciembre 6). Youtube. https://www.youtube.com/watch?v=xdCJa2QXmJ8