EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

SEMS
SUBSECRETARÍA DE EDUCACIÓN MEDIA SUPERIOR

DG ETI

Dirección General de Educación Tecnológica Industrial y de Servicios

Dirección Académica e Innovación Educativa
Subdirección de Innovación Educativa
Departamento de Planes, Programas y Superación Académica

Cuadernillo de Aprendizajes Esenciales, Estrategias de Aprendizaje y Productos

Programación

Aprendizajes esenciales			
Carrera:	Programación	Semestre:	3
Módulo/Submódulo:	**Módulo II. Aplica Metodologías de Desarrollo de Software con Herramientas de Programación Visual.** **Submódulo 1: Aplica metodología espiral con programación orientada a objetos**		

Aprendizajes esenciales o Competencias esenciales 1er parcial	Estrategias de Aprendizaje	Productos a Evaluar
I.- Identifica los elementos del paradigma de la programación orientada a objetos 1.- Aplica los conceptos de clases y objetos	1.1 El estudiante resuelve el Cuestionario Conceptos básicos **Cuestionario: Conceptos básicos** a) ¿Qué es un paradigma? b) ¿Qué entiende por programación orientada a objetos? c) ¿Conoces los elementos de la programación orientada a objetos y si es así cuáles son? d) ¿Qué beneficios obtienes al emplear la programación orientada a objetos? 1.2 El estudiante elabora un resumen de los conceptos principales del paradigma orientado a objetos a partir de las lecturas sugeridas por el docente. 1.3 El estudiante identifica en su entorno cotidiano los principales conceptos del paradigma orientado a objetos, los registra en un cuadro y redacta una definición propia de los conceptos de forma simple y entendible. 1.4 Investiga en los documentos proporcionados por el docente, la estructura de una **clase** y crear al menos 3 clases de los ejemplos proporcionados en la investigación de su entorno personal, registrando en su cuaderno de trabajo la estructura de cada uno de ellos. 1.5 De los ejemplos de clase elaborados, selecciona uno, y escribe cuáles serían los comportamientos que pudiese tener el objeto creado.	1.1 Cuestionario Conceptos básicos 1.2 Resumen de conceptos principales del paradigma orientado a objetos 1.3 Cuadro de identificación de conceptos y su definición personal 1.4 Redacción de tres ejemplos de clase con su respectiva estructura 1.5 Descripción del comportamiento

	1.6 Diseña, en su cuaderno de trabajo, clases relacionadas a objetos de su entorno considerando únicamente la identificación de los atributos del objeto e implementar las clases en un lenguaje de programación orientado a objetos. 1.7 Diseña clases protegiendo los datos con modificadores de acceso privado o protegido y agregar métodos públicos para obtener acceso seguro a los mismos. 1.8 En base a lo ya realizado deberá realizar un programa práctico empleando un lenguaje de programación.	1.6 Práctica realizada 1.7 Práctica realizada 1.8 Programa realizado
2. Utiliza las propiedades de herencia y polimorfismo	2.1 Crea clases que reúnan los miembros necesarios para resolver un problema y así implementar el encapsulamiento. 2.2 Identifica la estructura de un método y crea un programa que permita su uso en la resolución de problemas específicos. 2.3 Crea programas que contengan métodos sobrecargados y probar la utilidad de dichos métodos empleando herencia y polimorfismo. 2.4 Elabora reporte de prácticas. 2.5 Elabora un cuadro sinóptico en el que se muestren las definiciones de herencia y su clasificación. 2.6 Realiza la siguiente actividad: - Identifica los atributos y comportamientos propios de una especie que comparten los animales pertenecientes a ella. - Identifica los atributos y comportamientos propios de una categoría de objetos que compartan todos sus miembros. Con los datos obtenidos crea programas prácticos que manejen el concepto de herencia implementando redefinición de constructores y métodos.	2.1. Práctica realizada. 2.2. Ejercicio realizado. 2.3. Código de los Programas realizados. 2.4. Portafolio de Evidencia de Prácticas. 2.5. Cuadro sinóptico de herencia y clasificación 2.6. Programas Realizados.

	2.7. Realiza su portafolio de evidencias para su evaluación final del parcial.	2.7. Portafolio de evidencias: Final del parcial
Aprendizajes esenciales o competencias esenciales 2° parcial	Estrategias de Aprendizaje	Productos a Evaluar
II.- Aplica el modelo de espiral para el desarrollo de software 1.- Identifica los objetivos del sistema	1.1 El estudiante responde el cuestionario N°2 Objetivos del Sistema (**Anexo 1**) 1.2 Realiza un mapa mental de los elementos básicos de un proceso de desarrollo de software, de la siguiente información. **Elementos básicos de un proceso de desarrollo de software** Es definir los papeles que juegan los trabajadores, las actividades que desarrollan y los productos que deben generarse. En un plan de desarrollo cada trabajador debe tener su papel dentro de él, lo que define las actividades que debe realizar y los productos que debe generar. Las actividades son las tareas que deben realizar los trabajadores para cumplir sus obligaciones. A alto nivel, estas actividades son concebidas como las fases del proceso (especificación, análisis, ...), mientras que a más bajo nivel son tareas más concretas (crear cierto diagrama, escribir código, ...). Los productos son los documentos o información que debe ser creada como consecuencia de la actividad que se desarrolla. El producto último es el sistema que se desarrolla, pero en las fases intermedias deben generarse una amplia gama de documentos intermedios. Cada actividad debe tener siempre como principal objetivo generar ciertos productos bien definidos y especificados. Los procesos deben estar condicionados por el tipo de producto que se desarrolla y por la tradición y experiencia de la empresa que lo desarrolla 1.3 A partir de la lectura *Actividades para un proyecto de software* (**Anexo 2**) elabora una infografía que contenga imágenes relacionadas con el contenido. 1.4 A partir de la lectura *Tipos de modelos de desarrollo de software* (**Anexo 3**) elabora un glosario de al menos 10 conceptos sobre el tema	1.1 Cuestionario N°2 Objetivos del Sistema 1.2 Mapa mental: Elementos básicos de un proceso de desarrollo de software 1.3 Infografía: Actividades para un proyecto de software. 1.4 Glosario de conceptos sobre modelos de desarrollo de software

2.- Realiza el análisis de riesgos	2.1 Resuelve la actividad Análisis de riesgos **Actividad Análisis de riesgos** 1.- El alumno contestará el siguiente cuestionario: a) ¿Qué entiendes por la palabra "riesgo"? b) Da un ejemplo de un riesgo al que estés expuesto c) Al día, ¿a cuántos riesgos te enfrentas? d) Haz un listado de los riesgos a los que están expuestos tú y tu familia con la pandemia e) ¿Alguna vez te has puesto a analizar los riesgos a los que estás expuesto? f) De la lista de los riegos que detectaste, escoge uno y da una solución. g) Explica cómo fue tu experiencia al analizar el riesgo y da tus razones. 2.- Pregúntale a tu papá o tu mamá o a una persona mayor lo qué entiende por análisis de riesgo y anótalo en tu cuaderno. 3.- Utilizando tu celular, computadora, o yendo a un ciber, investiga en cualquier navegador lo que significa de análisis de riesgo. 4.- Realiza un resumen con lo más importante de tu investigación. 5.- Con base a lo que identificaste con tus papás, el listado de riesgos, y la investigación que hiciste, contesta las siguientes preguntas. a. ¿Mi definición de riesgo concuerda con la de mis papás y la de la investigación? b. ¿Qué ventajas tengo si hago un análisis de riesgo antes de tomar una decisión' c. ¿Qué desventajas tengo si hago un análisis de riesgo antes de tomar una decisión?	2.1 Actividad Análisis de riesgos

	2.2 - En base a la lectura *Análisis de riesgos en 6 pasos* (**Anexo 4**), selecciona un riesgo que veas en la escuela, en tu casa, o en tu entorno y siguiendo los 6 pasos, resuélvelo. 2.3 A partir de la actividad anterior contesta el siguiente cuestionario de reflexión: 1. ¿Fue fácil llegar a la solución? 2. Para ti, ¿es importante tener un análisis de riesgo? 3. Escribe tu experiencia al realizar el ejercicio anterior Piensa en la página de Internet de tu escuela y contesta lo siguiente. Haz de cuenta que tú tienes que diseñar la página y estar al frente de la administración: 4. ¿Qué harías para realizar la planeación de la página de tu escuela? 5. ¿A qué riesgos te vas a enfrentar (realiza un listado)? 6. Tomando en cuenta los 6 pasos de análisis de riesgos, escribe las actividades que se deben realizar en cada etapa. 7. Explica si fue fácil realizar este análisis de riesgo.	2.2 Uso del método de los 6 pasos para resolver el riesgo seleccionado. 2.3 Cuestionario resuelto de reflexión
3.- Desarrolla el código del sistema	3.1 Responde el siguiente cuestionario 1. ¿Qué es un programa? 2. ¿Qué define la sintaxis de un programa? 3. ¿Cuáles son elementos típicos de un programa? 4. ¿Qué son los identificadores? 5. ¿Cuáles son las reglas para el uso de identificadores? 6. ¿Qué son las palabras reservadas y da 5 ejemplos? 7. ¿Qué son los comentarios en un programa y para qué sirven? 8. ¿Qué es una sentencia y un bloque de programación? 9. ¿Cuáles son los tipos de datos que se manejan en un programa de manera común? 10. Explica cada tipo de operación que se puede realizar con las cadenas o string? 11. ¿Qué es una variable y que tipos hay en la programación?	3.1 Cuestionario

12. ¿Cuál es la diferencia entre una variable local y variable global?

3.2 Investiga las estructuras de control de los programas y elabora un mapa mental

3.3 Realiza el siguiente ejercicio de operadores

Dadas las variables de tipo int con valores A = 5, B = 3, C = -12 indicar si la evaluación de estas expresiones daría como resultado **verdadero o falso**

EXPRESION	RESULTADO
a) A > 3	
b) A > C	
c) A < C	
d) B < C	
e) B != C	
f) A == 3	
g) A * B == 15	
h) A * B == -30	
i) C / B < A	
j) C / B == -10	
k) C / B == -4	
l) A + B + C == 5	
m) (A+B == 8) && (A-B == 2)	
n) (A+B == 8) || (A-B == 6)	
o) A > 3 && B > 3 && C < 3	
p) A > 3 && B >= 3 && C < -3	

3.2 Mapa mental de las estructuras de control de los programas

3.3 Ejercicio de operadores

	3.4 Elabora códigos de programación a partir de las siguientes indicaciones I. Elabora el código de los siguientes programas en un lenguaje de programación 1.- Escriba el código que lee 3 números enteros y muestra cuál es el mayor y cuál el menor 2.- Escriba el código en lenguaje Java que lea un número entero, y muestre por pantalla el doble y el triple de ese número. E indicar si es par o impar 3.- Escriba el código que lea una cantidad de grados centígrados y la pase a grados Fahrenheit. La fórmula correspondiente es: F = 32 + (9 * C / 5) 4.- Escriba el código que lea por teclado el valor del radio de una circunferencia, calcula y muestra por pantalla la longitud y el área de la circunferencia. Utiliza las fórmulas siguientes: Longitud de la circunferencia = 2*PI*Radio, Área de la circunferencia = PI*Radio^2 5.- Escriba el código que lea un número entero que se introduce por teclado, determinar si es negativo o superior a 100 6.- Elaborar un programa en su cuaderno para calcular el área de un triángulo equilátero, A=(base * altura)/2	3.4 Código de programación

Aprendizajes esenciales o Competencias esenciales 3er parcial	Estrategias de Aprendizaje	Productos a Evaluar
4.- Realiza el proceso de comprobación 5.- Planifica su actualización 6.- Desarrolla Programa de manera individual o en equipo	1.- El estudiante atiende a la explicación del docente sobre el tema *uso de la metodología espiral en programación orientada a objetos y ejemplos prácticos para la aplicación del tema.* 2.- El estudiante siguiendo el cuadernillo de apuntes resuelve una serie de ejercicios prácticos proporcionados por el docente en documento. 3.- El alumno estudia en el cuadernillo el tema *búsquedas especiales*, así como ejemplos prácticos para aplicación del tema. 4.- El Estudiante siguiendo el cuadernillo, resuelve una serie de ejercicios prácticos proporcionados en documento. El docente realiza un repaso de los temas de *POO,* proporcionando cuadernillo de trabajo con resumen y actividades prácticas. 5.- El estudiante desarrolla en su cuaderno *una aplicación en un lenguaje de programación, implementando los recursos aprendidos en los temas vistos, siendo capaz de realizar comprobación, corrección y puesta en marcha de la aplicación generada.* 6.- El Estudiante escribe en su cuaderno *programas que contienen otras aplicaciones* poniendo en práctica todo lo aprendido en el presente Submódulo 1.	1.- Programas realizados 2.- Portafolio de evidencias 3.- Aplicaciones desarrolladas en un lenguaje de Programación Orientada a Objetos

Módulo II. Aplica Metodologías de Desarrollo de Software con Herramientas de Programación Visual.
Submódulo 1: Aplica metodología espiral con programación orientada a objetos

Anexo 1

Cuestionario N°2 Objetivos del Sistema

Escribe en el paréntesis la letra que se asocie a la respuesta correcta:

1. ¿Qué es un sistema de información? ( )

A) Es un conjunto de caracteres que se reúnen de una manera ilógica, para ejecutar una acción.

B) Es un conjunto de elementos que interactúan entre sí, organizados para llevar a cabo algunos métodos, procedimientos o control mediante el proceso de información.

C) Es un conjunto de operaciones y dispositivos que proporcionan capacidad de cálculos y funciones rápidas.

2. Indica cuál es el objetivo del proceso de Diseño del Sistema de Información ( )

A) Es la formalización de las actividades relacionadas con el desarrollo del software de un sistema informático

B) Es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

C) Todas son correctas

3. Son los elementos para un sistema de información ( ):

A) Usuarios, Comandos, Resultados, Información, Dispositivos de Salida

B) Hardware, Software, Recursos Humanos, Datos o Bases de Datos y Procedimientos.

C) Usuarios, Datos, Dispositivos de Entrada, Comandos y Operadores

4. Actividades de un sistema de información que se requieren para producir Información ( )

A) Identificación, Acción, Impresión y Recolección

B) Ingreso, Operación, Selección y Manipulación

C) Recopilación, Almacenamiento, Procesamiento y Distribución

5. ¿Qué es el diseño de sistemas? ( )

D) Es un proceso creativo en el que el analista repite a través de varias actividades o procedimientos de trabajo, uno a la vez, investigando mentalmente a través del proceso completo.

E) Es un proceso en el que el usuario ejecuta las acciones del sistema informático desarrollado

F) Es un proceso donde el programador realiza el pseudocódigo

Módulo II. Aplica Metodologías de Desarrollo de Software con Herramientas de Programación Visual.
Submódulo 1: Aplica metodología espiral con programación orientada a objetos

Anexo 2

Actividades para un proyecto de software:

1.- Planificación
Una vez que se hayan recopilado los requisitos del cliente, se debe realizar un análisis del ámbito del desarrollo. Este documento se conoce como especificación funcional. La Planificación es el paso previo al inicio de cualquier proyecto de desarrollo y sin dudas el más importante. En este se definen los requerimientos y funcionalidades que debe tener el software, mediante el trabajo en conjunto entre los desarrolladores, el departamento de ventas, los estudios de mercado y, fundamentalmente, el contacto con el cliente. En este punto se realizan asimismo los análisis de riesgo para el emprendimiento y se fijan los requisitos de aseguramiento de la calidad.

2.- Implementación, pruebas y documentación
La *implementación* es parte del proceso en el que los ingenieros de software programan el código para el proyecto de trabajo que está en relación de las demandas del software, en esta etapa se realizan las pruebas de caja blanca y caja negra.
Las *pruebas de software* son parte esencial del proceso de desarrollo del software. Esta parte del proceso tiene la función de detectar los errores de software lo antes posible.
La *documentación* del diseño interno del software con el objetivo de facilitar su mejora y su mantenimiento se realiza a lo largo del proyecto.

3.- Despliegue y Mantenimiento
El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.
Entrenamiento y soporte para el software es de suma importancia y algo que muchos desarrolladores de software descuidan. Los usuarios, por naturaleza, se oponen al cambio porque conlleva una cierta inseguridad, es por ello que es fundamental instruir de forma adecuada a los futuros usuarios del software.
El mantenimiento o mejora de un software con problemas recientemente desplegado, puede requerir más tiempo que el desarrollo inicial del software. Es posible que haya que incorporar código que no se ajusta al diseño original con el objetivo de solucionar un problema o ampliar la funcionalidad para un cliente. Si los costes de mantenimiento son muy elevados puede que sea oportuno rediseñar el sistema para poder contener los costes de mantenimiento.

Módulo II. Aplica Metodologías de Desarrollo de Software con Herramientas de Programación Visual.
Submódulo 1: Aplica metodología espiral con programación orientada a objetos

Anexo 3

Tipos de modelos de desarrollo de software

Ingeniería de software es la aplicación de enfoques sistemáticos y disciplinados al desarrollo de software, para esto se han creado modelos y metodologías para la correcta utilización del tiempo y recursos que una empresa o entidad disponen.
Los modelos de desarrollo de software ofrecen un marco de trabajo usado para controlar el proceso de desarrollo de sistemas de información, estos marcos de trabajo consisten en una filosofía de desarrollo de programas la cual debe de contar con las herramientas necesarias para la asistencia del proceso de desarrollo.

Modelo en espiral
Es el modelo en el cual las actividades se desarrollan en espiral, estas actividades se realizan conforme se van seleccionando de acuerdo con el análisis de riesgo. En cada iteración en este modelo, se deberán de tomar en cuenta los objetivos, las alternativas que se deberán de tomar de acuerdo con las características, estas son experiencia personal, requisitos a cumplir, las formas de gestión del sistema, entre otros. Este modelo tiene dos formas en las cuales se debe de planificar el proyecto, la forma angular, la cual indica únicamente el avance del software dentro del proyecto y la forma radial, la cual indica el aumento del costo dado que cada iteración conlleva más tiempo de desarrollo.

Modelo en cascada
Este es el modelo en el cual se ordenan rigurosamente las etapas del desarrollo del software, de esto se obtiene que el inicio de una etapa de desarrollo deba de esperar el fin de la etapa anterior. De esto se obtiene que cualquier error detectado lleve al rediseño del área de código afectado, lo cual aumenta de costo el desarrollo del proyecto.

Modelo de prototipos
Pertenece a los modelos evolutivos, en el cual el prototipo debe de ser construido rápidamente y con la utilización escasa de recursos. El prototipo es construido para mostrárselo al cliente, obtener críticas y retroalimentación, con lo cual se obtendrán los requisitos específicos para la aplicación a partir de las metas gráficas que son mostradas.

RAD o desarrollo de aplicaciones rápidas
Como su nombre lo indica permite la construcción rápida de sistemas utilizables. Está compuesto por un grupo reducido de personas incluyendo desarrolladores y probadores del sistema. También se debe de hacer énfasis al desarrollo de la aplicación cumpliendo correctamente las funcionalidades

principales, dejando a un lado a las implementaciones secundarias. Este modelo toma principalmente en cuenta las características de usabilidad, utilidad y rapidez de la ejecución de la aplicación.

Desarrollo concurrente

Se conoce como ingeniería concurrente y es utilizado en su mayoría para aplicaciones cliente servidor, en el cual se describen los múltiples procesos que ocurren simultáneamente en la aplicación. Una de las características de este proceso es que está orientado a las necesidades del usuario, las decisiones de la gestión y los resultados de las revisiones. Las ventajas que se pueden mencionar es que está orientado a grupos de trabajo independientes, proporcionando una visión exacta de lo que se lleva desarrollado del proyecto. Las desventajas que tienen es que se necesitan de grupos de trabajo y de las condiciones necesarias para su implementación.

Desarrollo por etapas

Es similar al modelo en cascada, sin embargo, se diferencia en que al momento de la crítica o bien retro alimentación por parte del usuario final, no se obtendrán completamente las características del software. Estas se irán descubriendo en el proceso del avance del software, mediante la creación de las diferentes versiones del código.

En este modelo, se distinguen las siguientes fases:

- Especificación conceptual
- Análisis de requisitos
- Diseño inicial
- Codificación y depuración

Módulo II. Aplica Metodologías de Desarrollo de Software con Herramientas de Programación Visual.
Submódulo 1: Aplica metodología espiral con programación orientada a objetos

Anexo 4

Análisis de Riesgo en 6 pasos

Fase 1. Definir el alcance

El primer paso a la hora de llevar a cabo el análisis de riesgos, es establecer el alcance del estudio. Vamos a considerar que este análisis de riesgos forma parte del Plan Director de Seguridad (PDS). Por lo tanto, recomendamos que el análisis de riesgos cubra la totalidad del alcance del PDS, dónde se han seleccionado las áreas estratégicas sobre las que mejorar la seguridad. Por otra parte, también es posible definir un alcance más limitado atendiendo a departamentos, procesos o sistemas. Por ejemplo, análisis de riesgos sobre los procesos del departamento de Administración, análisis de riesgos sobre los procesos de producción y gestión de almacén o análisis de riesgos sobre los sistemas TIC relacionados con la página web de la empresa, etc. En este caso práctico consideramos que el alcance escogido para el análisis de riesgos es "Los servicios y sistemas del Departamento de Informática".

Fase 2. Identificar los activos

Una vez definido el alcance, debemos identificar los activos más importantes que guardan relación con el departamento, proceso, o sistema objeto del estudio. Para mantener un inventario de activos sencillo puede ser suficiente con hacer uso de una hoja de cálculo o tabla como la que se muestra a continuación a modo de ejemplo:

ID	Nombre	Descripción	Responsable	Tipo	Ubicación	Crítico
ID_01	Servidor 01	Servidor de contabilidad.	Director Financiero	Servidor (Físico)	Sala de CPD1	Si
ID_02	RouterWifi	Router para la red WiFi de cortesía a los clientes.	Dept. Informática	Router (Físico)	Sala de CPD1	No
ID_03	Servidor 02	Servidor para la página web corporativa.	Dept. Informática	Servidor (Físico)	CPD externo	Si
...						

Fase 3. Identificar / seleccionar las amenazas

Habiendo identificado los principales activos, el siguiente paso consiste en identificar las amenazas a las que estos están expuestos. Tal y como imaginamos, el conjunto de amenazas es amplio y diverso por lo que debemos hacer un esfuerzo en mantener un enfoque práctico y aplicado. *Por ejemplo*, si nuestra intención es evaluar el riesgo que corremos frente a la destrucción de nuestro servidor de ficheros, es conveniente, considerar las averías del servidor, la posibilidad de daños por agua (rotura de una cañería) o los daños por fuego, en lugar de plantearnos el riesgo de que el CPD sea destruido por un meteorito.
A la hora de identificar las amenazas, puede ser útil tomar como punto de partida el catálogo de amenazas que incluye la metodología MAGERIT v3.

Fase 4. Identificar vulnerabilidades y salvaguardas

La siguiente fase consiste en estudiar las características de nuestros activos para identificar puntos débiles o vulnerabilidades. *Por ejemplo,* una posible vulnerabilidad puede ser identificar un conjunto de ordenadores o servidores cuyo sistemas antivirus no están actualizados o una serie de activos para los que no existe soporte ni mantenimiento por parte del fabricante. Posteriormente, a la hora de evaluar el riesgo aplicaremos penalizaciones para reflejar las vulnerabilidades identificadas.

Por otra parte, también analizaremos y documentaremos las medidas de seguridad implantadas en nuestra organización. *Por ejemplo*, es posible que hayamos instalado un sistema SAI (Sistema de Alimentación Ininterrumpida) o un grupo electrógeno para abastecer de electricidad a los equipos del CPD. Ambas medidas de seguridad (también conocidas como salvaguardas) contribuyen a reducir el riesgo de las amenazas relacionadas con el corte de suministro eléctrico.
Estas consideraciones (vulnerabilidades y salvaguardas) debemos tenerlas en cuenta cuando vayamos a estimar la probabilidad y el impacto como veremos en la siguiente fase.

Fase 5. Evaluar el riesgo

Llegado a este punto disponemos de los siguientes elementos:

1. Inventario de activos.
2. Conjunto de amenazas a las que está expuesta cada activo.
3. Conjunto de vulnerabilidades asociadas a cada activo (si corresponde).
4. Conjunto de medidas de seguridad implantadas

Con esta información, nos encontramos en condiciones de calcular el riesgo. Para cada par activo-amenaza, estimaremos la probabilidad de que la amenaza se materialice y el impacto sobre el negocio que esto produciría. El cálculo de riesgo se puede realizar usando tanto criterios cuantitativos como cualitativos. Pero para entenderlo mejor, veremos a modo de *ejemplo* las tablas para estimar los factores probabilidad e impacto.

Tabla para el cálculo de la probabilidad

Cualitativo	Cuantitativo	Descripción
Baja	1	La amenaza se materializa a lo sumo una vez cada año.
Media	2	La amenaza se materializa a lo sumo una vez cada mes.
Alta	3	La amenaza se materializa a lo sumo una vez cada semana.

Tabla para el cálculo del impacto

Cualitativo	Cuantitativo	Descripción
Bajo	1	El daño derivado de la materialización de la amenaza no tiene consecuencias relevantes para la organización.
Medio	2	El daño derivado de la materialización de la amenaza tiene consecuencias reseñables para la organización.
Alto	3	El daño derivado de la materialización de la amenaza tiene consecuencias graves reseñables para la organización.

Cálculo del riesgo

A la hora de calcular el riesgo, si hemos optado por hacer el análisis cuantitativo, calcularemos multiplicando los factores probabilidad e impacto:

RIESGO = PROBABILIDAD x IMPACTO.

Si por el contrario hemos optado por el análisis cualitativo, haremos uso de una matriz de riesgo como la que se muestra a continuación:

		IMPACTO		
		Bajo	Medio	Alto
PROBABILIDAD	Baja	Muy bajo	Bajo	Medio
	Media	Bajo	Medio	Alto
	Alta	Medio	Alto	Muy alto

Tal y como indicábamos en el apartado anterior, cuando vayamos a estimar la probabilidad y el impacto debemos tener en cuenta las vulnerabilidades y salvaguardas existentes. *Por ejemplo,* la caída del servidor principal podría tener un impacto alto para el negocio. Sin embargo, si existe una solución de alta disponibilidad (*Ej*. Servidores redundados), podemos considerar que el impacto será medio ya que estas medidas de seguridad harán que los procesos de negocio no se vean gravemente afectados por la caída del servidor. Si por el contrario hemos identificado vulnerabilidades asociadas al activo, aplicaremos una penalización a la hora de estimar el impacto. *Por ejemplo*, si los equipos de climatización del CPD no han recibido el mantenimiento recomendado por el fabricante, incrementaremos el impacto de amenazas como "condiciones ambientales inadecuadas" o "malfuncionamiento de los equipos debido a altas temperaturas".

Fase 6. Tratar el riesgo

Una vez calculado el riesgo, debemos tratar aquellos riesgos que superen un límite que nosotros mismos hayamos establecido. *Por ejemplo,* trataremos aquellos riesgos cuyo valor sea superior a "4" o superior a "Medio" en caso de que hayamos hecho el cálculo en términos cualitativos. A la hora de tratar el riesgo, existen cuatro estrategias principales:

5. **Transferir** el riesgo a un tercero. *Por ejemplo*, contratando un seguro que cubra los daños a terceros ocasionados por fugas de información.
6. **Eliminar** el riesgo. *Por ejemplo,* eliminando un proceso o sistema que está sujeto a un riesgo elevado. En el caso práctico que hemos expuesto, podríamos eliminar la wifi de cortesía para dar servicio a los clientes si no es estrictamente necesario.
7. **Asumir** el riesgo, siempre justificadamente. *Por ejemplo,* el coste de instalar un grupo electrógeno puede ser demasiado alto y por tanto, la organización puede optar por asumir.
8. **Implantar** medidas para mitigarlo. *Por ejemplo,* contratando un acceso a internet de respaldo para poder acceder a los servicios en la nube en caso de que la línea principal haya caído.

Por último, cabe señalar que como realizamos este análisis de riesgos en el contexto de un **PDS**, las acciones e iniciativas para tratar los riesgos pasarán a formar parte de este. Por lo tanto, deberemos clasificarlas y priorizarlas considerando el resto de los proyectos que forman parte del PDS. Asimismo, tal y como indicábamos en la introducción, llevar a cabo un análisis de riesgos nos proporciona información de gran valor y contribuye en gran medida a mejorar la seguridad de nuestra organización. Dada esta situación, animamos a nuestros lectores a llevar a cabo este tipo de proyectos ya bien sea de forma aislada o dentro del contexto de un proyecto mayor como es el caso del Plan Director de Seguridad

Aprendizajes esenciales			
Carrera:	**Programación**	Semestre:	**3**
Módulo/Submódulo:	**Módulo II Aplica metodologías de desarrollo de software con herramientas de programación visual** **Submódulo 2.- Aplica la metodología de desarrollo rápido de aplicaciones en programación orientada a eventos**		

Aprendizajes esenciales o Competencias esenciales 1er parcial	Estrategias de Aprendizaje	Productos a Evaluar
1.- Metodología de Desarrollo Rápido de Aplicaciones RAD	1.1.- Metodología de desarrollo rápido de aplicaciones: Después de consultar la documentación proporcionada por el docente. (**Anexo 1**) Construye un cuadro con Ventajas y desventaja de la implementación de esta metodología. RAD	El alumno entrega cada evidencia en libreta por medio de una fotografía 1.1 Cuadro de ventajas y desventajas de RAD 10%
	1.2 Realiza un mapa mental representando las fases y la filosofía de la metodología RAD.	1.2 Mapa mental Metodología RAD 10%
2.- Identifica los elementos característicos del paradigma de la programación orientada a eventos	2.1.- Paradigma de la Programación Orientada a Eventos: Después de consultar la documentación proporcionada por el docente (**Anexo 2**) identifica los elementos del paradigma de programación orientada a eventos y lo representa en un Diagrama Conceptual	2.1 Diagrama Conceptual paradigma de programación orientada a eventos 20%
	2.2 Elabora una tabla comparativa después de revisar información sobre herramientas visuales de desarrollo y relaciona 3 herramientas visuales de desarrollo. Analizando las especificaciones técnicas de cada software	2.2 Tabla Comparativa 10%
	2.3 Resuelve el Cuadro de relación de columnas de los siguientes conceptos: Evento, método, propiedad, formulario, botón, etiqueta, cuadro de texto, Estética, Usabilidad y Diseño.	2.3 Cuadro de relación resuelto **50%**

RELACIONA LAS COLUMNAS

AX	Al hacer clic en un botón, arrastrar el dedo tocar en la pantalla, se pueden ejecutar de forma automática o ser iniciados por el usuario	( )	Evento
BF	Son pequeños módulos de código que se diseñan con propósitos específicos. Realizan tareas particulares y especificas	( )	Método
VG	Son características que definen un objeto. es una asignación que describe algo sobre un componente o elemento que presta un servicio de comunicación cuando se diseñan interfaces	( )	Propiedad
GY	Es el primer objeto o control que se visualiza y constituye la pantalla o ventana sobre la que se colocan otros objetos o controles como etiquetas, controles de texto, botones, etc. y por supuesto el código necesario de nuestros programas	( )	Formulario
JU	Es un componente en el que el usuario lo selecciona para desencadenar cierta acción	( )	Botón
KE	Proporcionan una forma de colocar texto estático en una ventana gráfica	( )	Etiqueta

	DF Son elementos gráficos en donde el usuario puede escribir información en la aplicación. Escribe datos sensibles que permitirán alimentar datos para ejecutar o resolver, o controlar determinados propósitos del programa ( ) Cuadro de Texto RT Consiste en facilitar al usuario la utilización de una interfaz de la forma más fácil e intuitiva ( ) Usabilidad TY Es el proceso de visionado y definición de soluciones software a uno o más conjuntos de problemas por medio de la estrategia lógica uniendo códigos y elementos gráficos en una interfaz ( ) Diseño YU Armonía en el diseño de la interfaz ( ) Estética	
Aprendizajes esenciales o Competencias esenciales 2º parcial	Estrategias de Aprendizaje	Productos a Evaluar
3. Desarrolla de retos cortos en el Paradigma de la Programación Orientada a Eventos:	3.1 Implementación de eventos: **1.** Desarrolla un proyecto igual al de la imagen. Para practicar con Label (un label para cada uno de tus datos generales), PictureBox (Donde agregarás tu foto) y Botones. Junto con las propiedades tamaño de fuente, color de fuente, donde programarás el evento click del botón (para que muestre la información y otro para salga del proyecto). Una vez terminado subir el proyecto a la plataforma. Además, Tomar Fotografías del formulario y código hecho en tu cuaderno con el resultado de la operación. Integra la información a tu carpeta de	El alumno entrega cada evidencia en computadora por medio de una fotografía. Fotografía de Código y fotografía de evidencia de resultados en tiempo de corrida NOTA: (En el caso que el alumno no cuente con internet, solo enviara fotografía)

	evidencias.  2. Realiza un nuevo proyecto como el formulario de la imagen adjunta. Para continuar practicando con **PictureBox**, botones y el evento click. Cambiar la propiedad Name de los controles poniendo los prefijos o nomenclaturas utilizados (ver imagen del proyecto 8). El evento click de los botones debe modificar el valor de la propiedad visible a True o False respectivamente. 3. Tomar fotografías del formulario y código hecho en tu cuaderno. Integra la información a tu carpeta de evidencias.	• A cada Reto de programación en seg parcial se asigna una ponderación del 12.85%, son 7 retos. 2 retos por semana • Y 10% de calificación para estimular que cumpla en las fechas de entrega establecidas.

4. Crear el proyecto de la imagen adjunta donde deberás cambiar la propiedad Name de los controles con los siguientes prefijos:

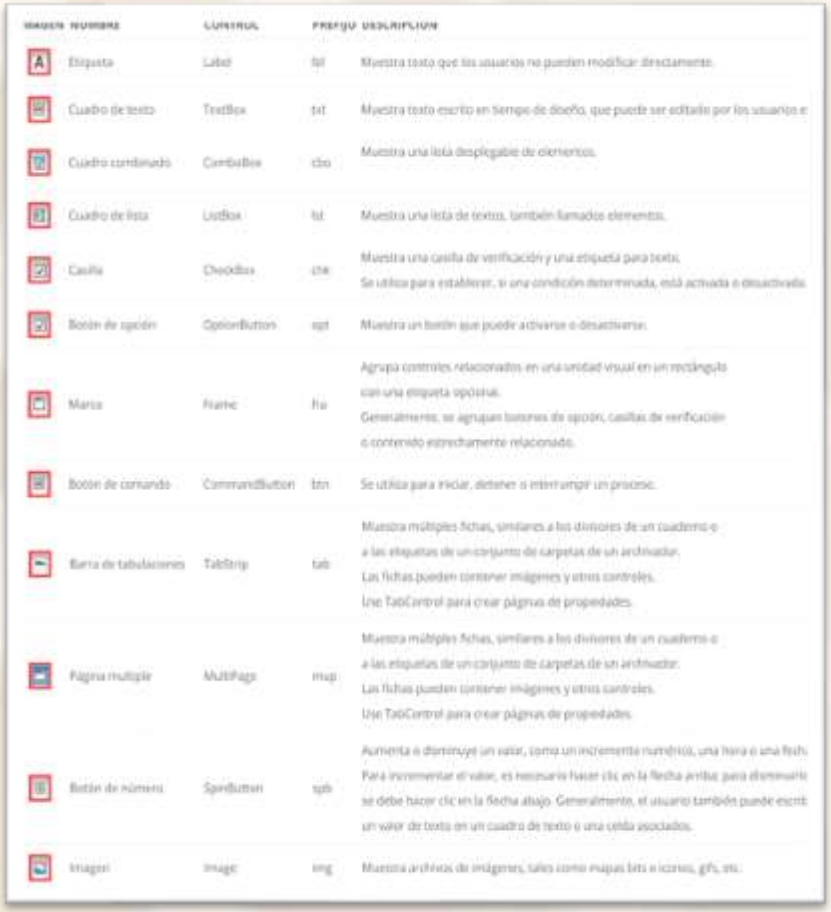


IMAGEN	NOMBRE	CONTROL	PREFIJO	DESCRIPCION
A	Etiqueta	Label	lbl	Muestra texto que los usuarios no pueden modificar directamente.
	Cuadro de texto	TextBox	txt	Muestra texto escrito en tiempo de diseño, que puede ser editado por los usuarios e
	Cuadro combinado	ComboBox	cbo	Muestra una lista desplegable de elementos.
	Cuadro de lista	ListBox	lst	Muestra una lista de textos, también llamados elementos.
	Casilla	CheckBox	chk	Muestra una casilla de verificación y una etiqueta para texto. Se utiliza para establecer, si una condición determinada, está activada o desactivada.
	Botón de opción	OptionButton	opt	Muestra un botón que puede activarse o desactivarse.
	Marco	Frame	fra	Agrupa controles relacionados en una unidad visual en un rectángulo con una etiqueta opcional. Generalmente, se agrupan botones de opción, casillas de verificación o contenido estrechamente relacionado.
	Botón de comando	CommandButton	btn	Se utiliza para iniciar, detener o interrumpir un proceso.
	Barra de tabulaciones	TabStrip	tab	Muestra múltiples fichas, similares a los divisores de un cuaderno o a las etiquetas de un conjunto de carpetas de un archivador. Las fichas pueden contener imágenes y otros controles. Use TabControl para crear páginas de propiedades.
	Página múltiple	MultiPage	mup	Muestra múltiples fichas, similares a los divisores de un cuaderno o a las etiquetas de un conjunto de carpetas de un archivador. Las fichas pueden contener imágenes y otros controles. Use TabControl para crear páginas de propiedades.
	Botón de número	SpinButton	spb	Aumenta o disminuye un valor, como un incremento numérico, una hora o una flech Para incrementar el valor, es necesario hacer clic en la flecha arriba; para disminuirlo se debe hacer clic en la flecha abajo. Generalmente, el usuario también puede escrib un valor de texto en un cuadro de texto o una celda asociados.
	Imagen	Image	img	Muestra archivos de imágenes, tales como mapas bits e iconos, gifs, etc.

La operación es sobre números de tipo Entero (integer).

-El usuario introducirá dos números enteros (uno en cada cuadro de texto), al dar click sobre uno de los botones mostrará el resultado de la suma, resta, multiplicación o división según corresponda de los dos números introducidos.

Una vez terminado, Tomar Fotografías del formulario y código hecho en tu cuaderno con el resultado de la operación. Integra la información a tu carpeta de evidencias.

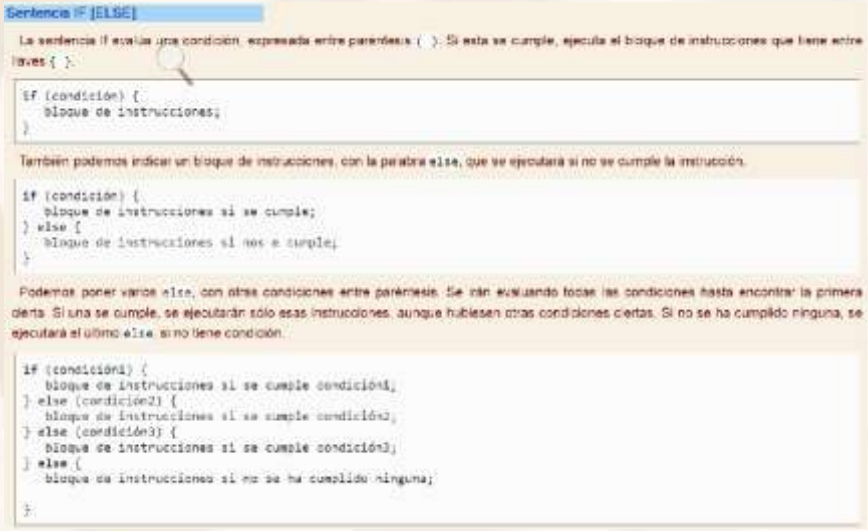
5. Crear el proyecto con un formulario que pida “Login” y “Password” a un usuario. Mostrar un mensaje de “Bienvenida” si los datos son correctos, o un mensaje de “Rechazo” si no lo son.

• Datos correctos
– Login: “ALUMNO”
– Password: “Cetis156”
• Ocultar los caracteres tecleados del Password con “*” (modificar la propiedad PasswordChar del TextBox).

Una vez terminado, Tomar Fotografías del formulario y código hecho en tu cuaderno. Integra la información a tu carpeta de evidencias.

6. Desarrollar un proyecto que solicite al usuario dos números enteros diferentes. Compare cuál es Menor de los dos y que despliegue un mensaje indicando cual es menor

(utilizar MessageBox). Para practicar el uso de la Sentencia if( )


Sentencia IF [ELSE]

La sentencia if evalúa una condición, expresada entre paréntesis ( ). Si esta se cumple, ejecuta el bloque de instrucciones que tiene entre llaves { }.

```
if (condición) {
   bloque de instrucciones;
}
```

También podemos indicar un bloque de instrucciones, con la palabra else, que se ejecutará si no se cumple la instrucción.

```
if (condición) {
   bloque de instrucciones si se cumple;
} else {
   bloque de instrucciones si no se cumple;
}
```

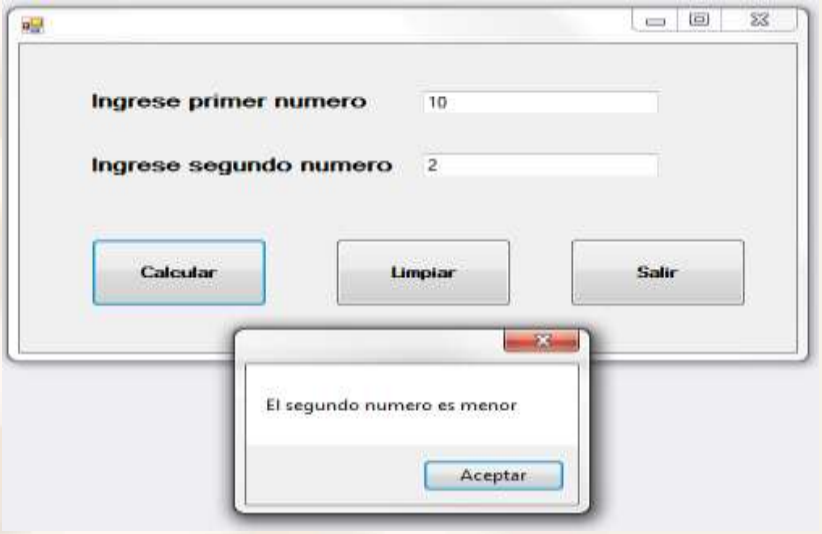
Podemos poner varios else, con otras condiciones entre paréntesis. Se irán evaluando hasta encontrar la primera cierta. Si una se cumple, se ejecutarán sólo esas instrucciones, aunque hubiesen otras condiciones ciertas. Si no se ha cumplido ninguna, se ejecutará el último else, si no tiene condición.

```
if (condición1) {
   bloque de instrucciones si se cumple condición1;
} else (condición2) {
   bloque de instrucciones si se cumple condición2;
} else (condición3) {
   bloque de instrucciones si se cumple condición3;
} else {
   bloque de instrucciones si no se ha cumplido ninguna;
}
```

• Recuerda los requisitos

- Tu nombre completo en la barra de titulo
- Utilizar los prefijos a los controles en la propiedad Name.

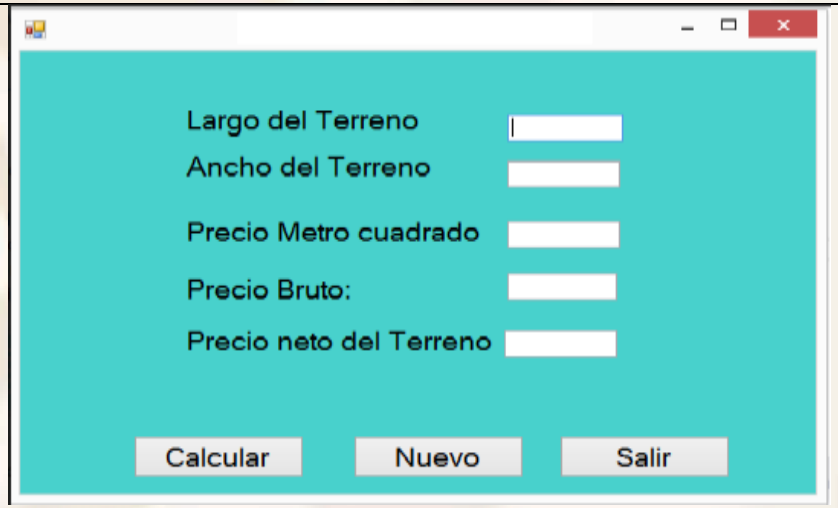
Una vez terminado, Tomar Fotografías del formulario y código hecho en tu cuaderno. Integra la información a tu carpeta de evidencias.



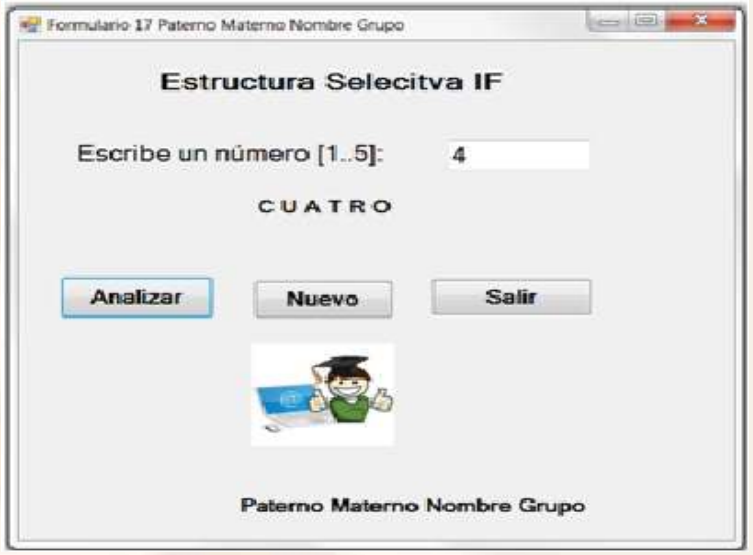
7. Realizar un Proyecto que calcule el Área y precio de un terreno rectangular. Solicitando como datos de entrada la medida del Largo y ancho, así como el precio por metro cuadrado. Si el terreno mide más de 1000 metros se otorga un 25% de descuento; si mide más de 500 metros un 17 % de descuento; y si mide más de 400 metros un 10% de descuento.

• Recuerda los requisitos:

- Tu nombre completo en la barra de título
- Utilizar los prefijos a los controles en la propiedad Name.
- Una vez terminado Tomar Fotografías del formulario y código hecho en tu cuaderno. Integra la información a tu carpeta de evidencias.



8. Elaborar un Proyecto que permita la captura de un número del 1 al 5. Deberá desplegar el número en letra. Debe validar que realmente se introduzca un número del 1 al 5.

	- Una vez terminado, Tomar Fotografías del formulario y código hecho en tu cuaderno. Integra la información a tu carpeta de evidencias. 	
Aprendizajes esenciales o Competencias esenciales 3er parcial	Estrategias de Aprendizaje	Productos a Evaluar
4.- Identifica los elementos del paradigma de la programación orientada a eventos 5.- Aplica el modelo de desarrollo rápido de aplicaciones (RAD) para el desarrollo de software	4.1 El alumno investiga (en el **Anexo 3**) sobre: • Las metodologías ágiles. • Metodología SCRUM: o Introducción: ¿por qué utilizar SCRUM? o Flujo SCRUM: principios, aspectos y procesos o Inicio del proyecto o Planificación y Estimación 4.2 Selecciona 5 razones del por qué utilizar SCRUMy explica el por qué. Explica el esquema del Flujo de SCRUM 5.1 Realiza un caso práctico con las fases vistas de SCRUM	4.1 Resumen: ¿qué son las metodologías ágiles? 4.2 Justificación de las 5 razones del por qué utilizar SCRUM y explicación del Flujo de SCRUM. 5.1 Caso práctico

Módulo II Aplica metodologías de desarrollo de software con herramientas de programación visual
Submódulo 2.- Aplica la metodología de desarrollo rápido de aplicaciones en programación orientada a eventos

Anexo 1

DOCUMENTACION PARA RESOLVER LAS ACTIVIDADES

METODOLOGÍA RAD (DESARROLLO RAPIDO DE APLICACIONES)

El desarrollo rápido de aplicaciones o RAD (acrónimo en inglés de rapid application development) es un proceso de desarrollo de software, desarrollado inicialmente por James Martin en1980.
El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE (Computer Aided Software Engineering).
El desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.
Una de las primeras decisiones a la que nos enfrentamos cuando comenzamos a enfocar un proyecto es elegir la metodología más adecuada para el mismo. Generalmente no es una decisión sencilla, siendo necesario debatir los requerimientos del proyecto antes de elegir la más apropiada.
Las dos metodologías más utilizadas en el desarrollo de proyecto son:
Waterfall: También denominada en "cascada", es el método que se ha utilizado tradicionalmente. Consiste en desarrollar un proyecto de forma secuencial, comenzando con las fases de análisis y diseño y terminando con las de testeo y puesta en producción.
Agile: Una metodología de tipo RAD (Rapid Application Development), siendo Scrum el método más utilizado.
Qué es RAD
El desarrollo rápido de aplicaciones es un enfoque de desarrollo de software ágil que se centra más en proyectos de software en curso y comentarios de los usuarios y menos en seguir un plan estricto. Por tanto, prioriza la creación rápida de prototipos sobre la planificación costosa. Este modelo permite tratar los proyectos de software como arcilla, en lugar de acero, que es como los tratan las prácticas tradicionales de desarrollo como Waterfall.
RAD es menos charla y más trabajo, es decir, menos palabras y más acciones. Para ello se realizan muchas pruebas y se siguen una serie de fases o pasos, a pesar de que RAD desestima la planificación estricta.

FASES DEL RAD

Modelado de gestión: el flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la proceso?
Modelado de datos: el flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.
Modelado de proceso: los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.
Generación de aplicaciones: El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.

Pruebas de entrega: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

LA METODOLOGIA RAD

Ventajas y Desventajas RAD

Ventajas:

- Velocidad: Con RAD, es más probable que los proyectos terminen a tiempo y para satisfacción del cliente en el momento de la entrega.
- Costo: Con el desarrollo rápido de aplicaciones, los desarrolladores crean los sistemas exactos que requiere el cliente, y nada más.
- En general:
    - El desarrollo se realiza a un nivel de abstracción mayor.
    - Visibilidad temprana.
    - Menor codificación manual.
    - Posiblemente menos fallas.
    - Posiblemente menor costo.
    - Ciclos de desarrollo más pequeños.
    - Tiene mayor probabilidad de dejar “a gusto” al cliente ya que se pueden generar con mayor facilidad y rapidez aplicaciones prototipo con una GUI vistosa y lista para su uso prematuro, con una diferencia de tiempo de entrega abismal a software desarrollado con otros modelos de desarrollo enfocados en la robustez del sistema.

Desventajas:

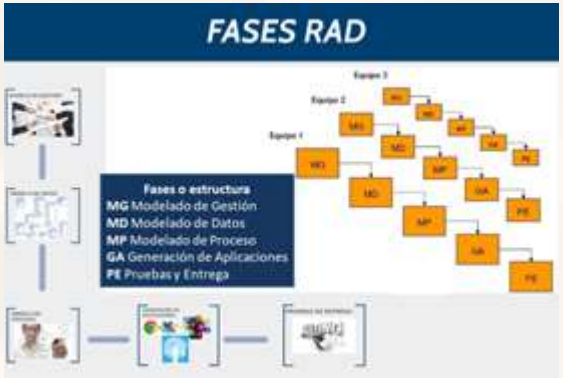
- Escala: Las prácticas de RAD se complican cuando se expanden más allá de un solo equipo o requieren comunicación entre equipos.
- Compromiso: El ciclo frecuente de prototipos RAD requiere que los desarrolladores y clientes se comprometan a reuniones frecuentes que, al principio, pueden parecer consumir ciclos innecesarios para ambas partes.
- Enfoque de interfaz: La metodología RAD motiva a los desarrolladores a encontrar la solución perfecta para el cliente. El cliente juzga la calidad de la solución en su interacción y, a menudo, todo con lo que interactúa es una fachada.
- Como consecuencia, algunos desarrolladores se enfocan menos en las prácticas back-end para acelerar el desarrollo del prototipo enfocado en el front-end.
- Y en general:
    - Progreso más difícil de medir.
    - Riesgo de revertirse a las prácticas sin control de antaño.

- Más fallas (por síndrome de “codificar a lo bestia”).

Hoy en día se suele utilizar las siguientes herramientas para referirnos al desarrollo rápido de interfaces gráficas de usuario tales como Glade, o entornos de desarrollo integrado completos. Algunas de las plataformas más conocidas son java NetBeans eclipse, Visual Studio, OpenXava, Lazarus, Gambas, RapidClipse, Delphi, Foxpro, Anjuta, Game Maker, Velneo o Clarion.

JAVA (RAD)
Algunas herramientas de software de desarrollo rápido de aplicaciones en JAVA son los siguientes:

Ebase Xi : Es una herramienta de desarrollo rápido de aplicaciones plataforma que combina el navegador web basado en la interfaz de usuario de desarrollo, gestión de procesos empresariales y la integración de datos en una sola tecnología IDE .
IBM Rational Application Developer es un IDE rápido desarrollo de aplicaciones para la creación de aplicaciones empresariales y web y servicios para Windows, Linux y Unix
m-Power es una herramienta de desarrollo de software que automatiza el desarrollo de aplicaciones y crea rápidamente de clase empresarial de aplicaciones web a través de cualquier base de datos o la plataforma.
MyEclipse es un entorno de desarrollo rápido de aplicaciones, centrándose en Java empresarial y el desarrollo de aplicaciones web.
NetBeans es un entorno multiplataforma, RAD IDE para la creación visual de escritorio, móviles, web y aplicaciones para Linux, Windows y Mac OS X.
El IDE es oficialmente compatible con Java, PHP, JavaScript y C / C + + lenguajes de programación.



MODELO DE GESTIÓN

MODELO DE DATOS

MODELO DE PROCESO

GENERACIÓN DE APLICACIONES

PRUEBAS Y ENTREGA

CARACTERÍSTICAS RAD

LENGUAJE DE PROGRAMACIÓN

- PERL
- Python
- TCL
- PHP
- Java
- Fortran
- C

VENTAJAS Y DESVENTAJAS

Módulo II Aplica metodologías de desarrollo de software con herramientas de programación visual
Submódulo 2.- Aplica la metodología de desarrollo rápido de aplicaciones en programación orientada a eventos

Anexo 2

2.1.- ¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A EVENTOS?

La programación dirigida por eventos es un paradigma de programación en el que el flujo del programa está determinado por eventos o mensajes desde otros programas o hilos de ejecución.
Las aplicaciones desarrolladas con programación dirigida por eventos implementan un bucle principal donde se ejecutan las dos secciones principales de la aplicación: El selector de eventos y el manejador de eventos.
La mayoría de las librerías para el desarrollo de aplicaciones con GUI están diseñadas para ser dirigidos por eventos.

Un poco de historia
A finales de los 70, los sistemas estaban pensados para trabajar como cadenas de ensamblaje donde un programa usaba una entrada y producía una salida que sería utilizada por otro programa como entrada para producir otra salida y así hasta finalizar el proceso. Este proceso mental de construir software es la base del desarrollo estructurado.
El padre del desarrollo estructurado (NO de la programación estructurada ojo) fue Larry LeRoy Constantine bajo el ala del Instituto de Investigación de Sistemas de IBM. Uno de los mayores expertos y defensores de los métodos estructurados es Edward Yourdon, tanto que las expresiones “Yourdon” y “métodos de análisis y diseño estructurado” son sinónimos.
Constantine y Yourdon definieron nuevos modelos de control del flujo de datos implementando lo que llamaron transacciones que en realidad son un patrón de diseño de manejadores de eventos.

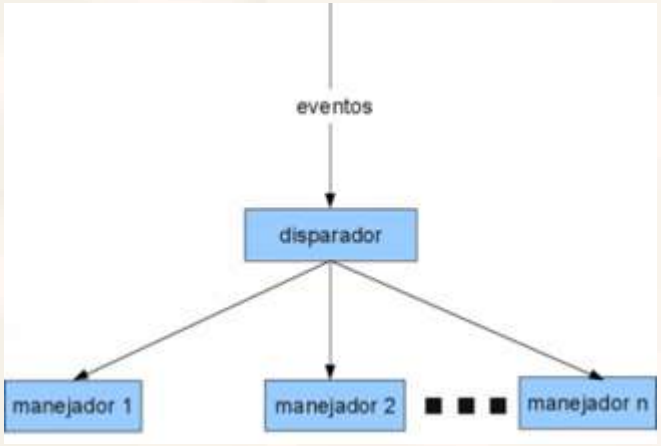
En el siguiente diagrama se muestra el siguiente proceso:

Un flujo de datos invoca eventos o lo que Constantine y Yourdon llamaron “transacciones “

Un disparador los envía a manejadores especializados, Constantine y Yourdon lo llamaron “centro de transacciones “

un conjunto de manejadores que se encargan de realizar operaciones sobre el flujo de datos

El trabajo del disparador es analizar los eventos para determinar su naturaleza y entonces enviarlos al manejador adecuado que es capaz de trabajar con eventos de esa naturaleza. El disparador tiene que procesar un flujo de eventos, así que su lógica debe incluir un bucle de eventos para poder enviar un evento a un manejador y volver a escuchar a la espera de nuevos eventos que disparar.



Es común que exista un evento especial que rompa el bucle y salga de la aplicación, a ese evento se le llama evento finalizador y es muy común en todas las librerías para escribir aplicaciones GUI. También puede ocurrir que el disparador capture un evento de naturaleza desconocida o para el que no exista un manejador adecuado, en esos casos, el disparador debe descartar el evento o lanzar una excepción.

En algunas ocasiones es común que el disparador y los manejadores no sean capaces de procesar los eventos con la suficiente premura conforme van llegando por lo que la mayoría de las aplicaciones basadas en GUI implementan una cola de eventos. Esta lógica es muy delicada y si no se implementa bien puede ser producto de cuellos de botella fatales.

Cuando ocurre un evento, la entidad observada mira en su lista de observadores y notifica a aquellos que se registraron para recibir eventos de ese tipo. Los observadores dejaron instrucciones detalladas de cómo puede la entidad observada ponerse en contacto con ellos para recibir los eventos. A este patrón también se le llama El Principio de Hollywood como parodia de la típica frase de Hollywood “te llamaremos en cuanto tengamos un papel para ti “.

Otro aspecto a tener en cuenta en la programación orientada a eventos es el comprender los siguientes términos:

- Evento
- Propiedades
- Métodos

¿Qué son los eventos?
Los Eventos son las acciones sobre el programa, como, por ejemplo:

- Clic sobre un botón
- Doble clic sobre el nombre de un fichero para abrirlo
- Arrastrar un icono
- Pulsar una tecla o una combinación de teclas
- Elegir una opción de un menú
- Escribir en una caja de texto
- simplemente mover el mouse

Cuando se produce o dispara un evento en programación permite al usuario realizar una serie de acciones lógicas dentro de un determinado programa y sobre un determinado componente o elemento que presta un servicio de comunicación cuando se diseñan interfaces. Se da inicio a un conjunto de acciones programadas por el programador para ese evento concreto.

¿Qué es una propiedad?
Una propiedad son características que definen un objeto. es una asignación que describe algo sobre un componente o elemento que presta un servicio de comunicación cuando se diseñan interfaces., como, por ejemplo:

- Un formulario
- Un botón de comando
- Una caja de texto
- Una etiqueta

Dependiendo de la propiedad Acciones que definen un objeto., esta se puede asignar en tiempo de diseño usando la ventana Propiedades y/o en tiempo de ejecución en el momento de programar el control de objetos que poseen propiedades y métodos que facilitan el desarrollo de aplicaciones informáticas.

Las propiedades se usan para cambiar la forma de los componentes o control de objetos que poseen propiedades y métodos que facilitan el desarrollo de aplicaciones, por ejemplo, el tamaño de la letra de un control, el tipo de letra, la alineación, etc.

¿Qué es un método?
Un método es una función que es llamada desde el programa, Son pequeños módulos de código que se diseñan con propósitos específicos, los métodos realizan tareas particulares y específicas. Los métodos solo pueden ser ejecutados en tiempo de ejecución no en tiempo de diseño. Algunos ejemplos de métodos son:

En una ventana, el método MOVE, que mueve un formulario en un espacio de dos dimensiones en la pantalla
Otros SetFocus, LostFocus, AddItem etc…

Las aplicaciones reaccionan a eventos; en programación permite al usuario realizar una serie de acciones lógicas para un determinado programa. Al hacer clic en un botón, arrastrar el dedo o tocar la pantalla es lo que conocemos como eventos. Cuando se produce un evento, la aplicación reacciona llamando a una secuencia de instrucciones como establecer el color de fondo de un botón a azul o cambiar el texto de una etiqueta.

Los eventos pueden ser divididos en 2 tipos diferentes:

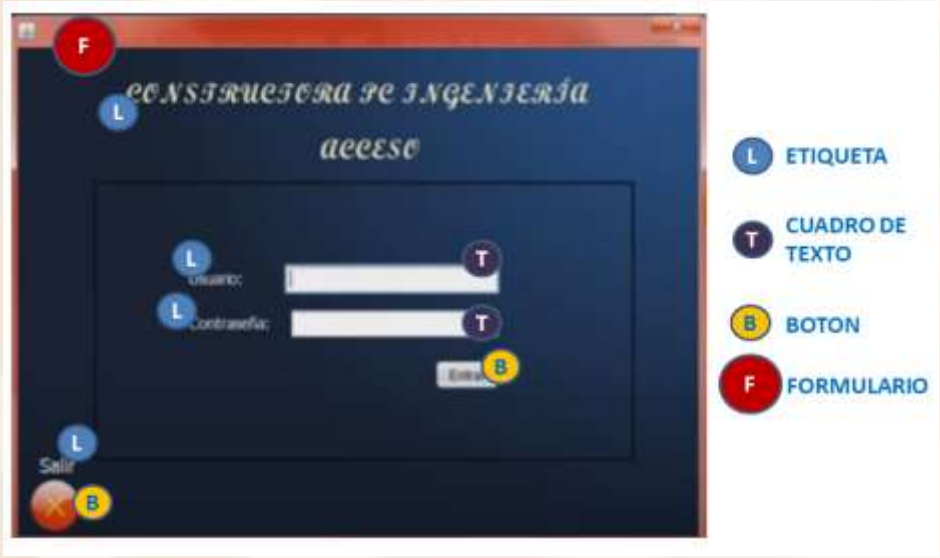
- Automáticos como al abrir una ventana.
- Iniciados por el usuario como hacer clic en un botón, tocar o arrastrar en la pantalla, inclinar el teléfono, etc.
-

El formulario es el primer objeto o control que se visualiza y constituye la pantalla o ventana sobre la que se colocan otros objetos o controles como etiquetas, controles de texto, botones, etc. y por supuesto el código necesario de nuestros programas, por lo tanto, constituirán la Interfaz de usuario.

Un botón es un componente en el que el usuario lo selecciona para desencadenar cierta acción. Una aplicación de Java puede utilizar varios tipos de botones, incluyendo botones de comando, casillas de verificación, botones interruptores y botones de opción

Las etiquetas (Label) proporcionan una forma de colocar texto estático en una ventana gráfica, para mostrar información que no varía, normalmente se limita a presentar textos en pantalla

Los cuadros de texto son elementos gráficos en donde el usuario puede escribir información en la aplicación. Escribe datos sensibles que permitirán alimentar datos para ejecutar o resolver, o controlar determinados propósitos del programa. El componente de texto más usado es TextField el cual puede mostrar varias líneas, ajustar el texto al tamaño del control y agregar formato básico



En el mundo del desarrollo de las aplicaciones orientadas a eventos. La aplicación, primero debe de funcionar y siguiente elemento a valorar es cuidar la estética, es decir que tenga armonía en el diseño de la interfaz.

El neologismo usabilidad (del inglés usability —facilidad de uso—) se refiere a la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.

La usabilidad en el diseño de interfaz; consiste en facilitar al usuario la utilización de una interfaz de la forma más fácil e intuitiva. Pero ¿cómo diseñamos aplicaciones usables?

Diseño:
El diseño de software es el proceso de visionado y definición de soluciones software a uno o más conjuntos de problemas por medio de la estrategia lógica uniendo códigos y elementos gráficos en una interfaz tipo (Frame (ventana) que puedan ayudar al usuario de forma intuitiva a interactuar con una aplicación informática. Prueba de ello es que las grandes marcas intentan renovarse constantemente y mejorar el aspecto de su software

Reglas y principios de usabilidad y Diseño
A la hora de diseñar una interfaz para un proyecto, es de especial importancia tener en consideración las reglas básicas de usabilidad. A continuación, se muestran algunas de ellas.

	Simetría Mediante la simetría podemos conseguir diseños equilibrados
	La regla de los tres clics El usuario debe poder acceder de forma sencilla a todo el contenido de una plataforma. De este modo, se considera que el contenido que se encuentra a más de tres clics no es importante.

	Correspondencia entre los contenidos y el mundo real Es decir, el contenido debe seguir las convenciones del mundo real y el diseñador debe ser capaz de mostrar la información de forma natural y lógica.
	Navegación rápida entre secciones Significa que el usuario se encuentre cómodo visitando todo el contenido de nuestra aplicación. El menú principal siempre debe estar visible o utilizar adecuadamente los sistemas de paginado, entre otras opciones

Módulo II Aplica metodologías de desarrollo de software con herramientas de programación visual
Submódulo 2.- Aplica la metodología de desarrollo rápido de aplicaciones en programación orientada a eventos

Anexo 3

Metodologías ágiles para la gestión de proyectos

Podemos definir las **metodologías ágiles** como un conjunto tareas y procedimientos dirigidos a la **gestión de proyectos**. Son aquellos métodos de desarrollo en los cuales tanto las necesidades como las soluciones a estas evolucionan con el pasar del tiempo, a través del trabajo en equipo de grupos multidisciplinarios que se caracterizan por tener las siguientes cualidades:

1. Desarrollo evolutivo y flexible.
2. Autonomía de los equipos.
3. Planificación.
4. Comunicación.

Existen diferentes opciones ágiles entre las cuales podemos destacar las siguientes: **Scrum**, programación extrema (**XP**) y **Kanban**, siendo estas tres (03) las alternativas más utilizadas. Es importante mencionar, que todas las metodologías ágiles cumplen con el **Manifiesto ágil**, el cual se encuentra compuesto por doce (12) principios agrupado en cuatro (04) valores fundamentales:

1) Individuos e interacciones sobre procesos y herramientas.
2) Software funcionando sobre documentación extensiva.
3) Colaboración con el cliente sobre negociación contractual.
4) Respuesta ante el cambio sobre seguir un plan.

1.- PROGRAMACIÓN EXTREMA (XP)

Conocida por sus siglas XP (**eXtreme Programming**), es una metodología basada en un conjunto de reglas y buenas prácticas para el desarrollo de software en ambientes muy cambiantes con requisitos imprecisos, por ende, está enfocada en la retroalimentación continua entre el equipo de desarrollo y el cliente.

Es por ello que iniciando el proyecto se deben definir todos los requisitos, para luego invertir el esfuerzo en manejar los cambios que se presenten y así minimizar las posibilidades de error. **XP** tiene como base la **simplicidad** y como objetivo la **satisfacción** del cliente.

CARACTERÍSTICAS DE XP

En resumen, las principales características de la **programación extrema** son:

1. Desarrollo iterativo e incremental.
2. Programación en parejas.
3. Pruebas unitarias continuas.
4. Corrección periódica de errores.
5. Integración del equipo de programación con el cliente.
6. Simplicidad, propiedad del código compartida y refactorización del código.

La **programación extrema** optimiza los tiempos y se adapta al desarrollo de sistemas grandes y pequeños sin mayor documentación, el código es claro y simple, así mismo complementa los conocimientos entre los miembros del equipo, gracias a la programación en parejas. Sin embargo, una desventaja de esta metodología ágil es que luego de cada entrega el sistema puede ir creciendo según sean las peticiones del cliente.

2.- SCRUM

Esta metodología, es un **marco de trabajo de procesos ágiles** que trabaja con el **ciclo de vida iterativo e incremental**, donde se va liberando el producto por pares de forma periódica, aplicando las buenas prácticas de trabajo colaborativo (en equipo), facilitando el hallazgo de soluciones óptimas a los problemas que pueden ir surgiendo en el proceso de desarrollo del proyecto.

Con **Scrum** se realizan entregas regulares y parciales (**sprint**) del producto final, todas ellas con una prioridad previamente establecida que nace según el beneficio que aporten al cliente, minimizando los riesgos que pueden surgir de desarrollos extremadamente largos. Es por tal motivo, que Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesitan obtener resultados de manera inmediata y donde son fundamentales los siguientes aspectos: la innovación, la productividad, la flexibilidad y la competitividad.

¿QUIÉN CONFORMA EL EQUIPO SCRUM?

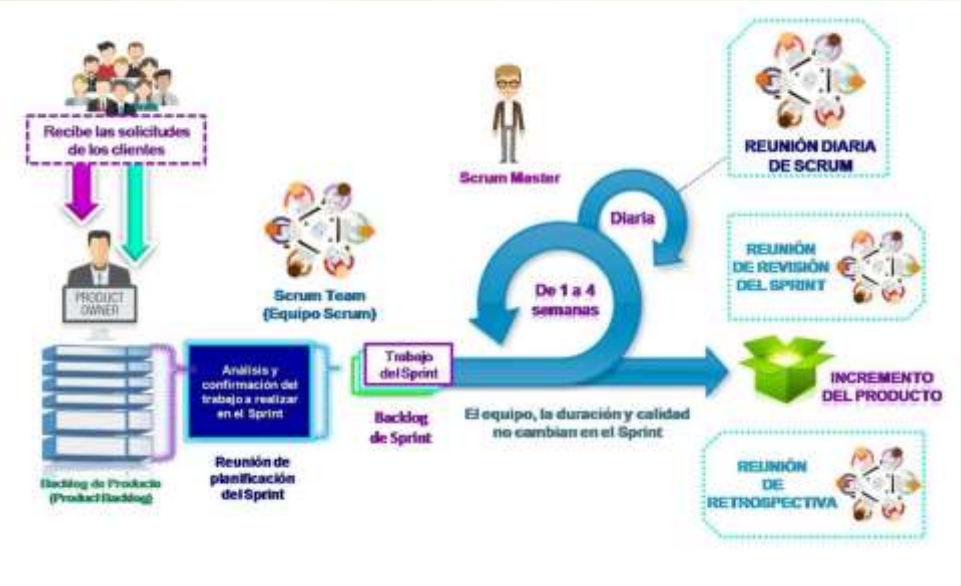
En los Equipos Scrum, se cuenta con roles específicos y cada uno de ellos es imprescindible para que se lleve a cabo el proceso de forma satisfactoria:

1. **Stakeholder:** Es el cliente, su responsabilidad radica en definir los requerimientos (**Product Backlog**), recibir el producto al final de cada iteración y proporcionar el feedback correspondiente.
2. **Product Owner:** Es el intermediario de la comunicación entre el cliente (**stakeholder**) y el equipo de desarrollo. Este debe priorizar los requerimientos según sean las necesidades de la solicitud.
3. **Scrum Master:** Actúa como facilitador ante todo el equipo de desarrollo, elimina todos aquellos impedimentos que identifique durante el proceso, así mismo se encarga de que el equipo siga los valores y los principios ágiles, las reglas y los procesos de Scrum, incentivando al grupo de trabajo.
4. **Scrum Team (Equipo de desarrollo):** Se encarga de desarrollar los casos de uso definidos en el Product Backlog, es un equipo auto gestionado lo que quiere decir que no existe un de jefe de equipo, motivo por el cual todos los miembros se deben de encargar de realizar las estimaciones y en base a la velocidad obtenida en las iteraciones irán construyendo el **Sprint Backlog**.

LAS REUNIONES UN PILAR IMPORTANTE

Un punto fundamental en el proceso que conlleva **Scrum** son las **revisiones (reuniones)**, con ellas se fomenta la comunicación y transparencia del proceso, las reuniones que aplican son:

1) **Reunión de planificación:** Se debe realizar al **inicio de cada sprint**, esto con el objetivo de **planificar la cantidad de trabajo** a la que el equipo se va a comprometer a construir durante el próximo sprint.
2) **Reunión diaria:** Son reuniones cuyo lapso tiene un **máximo 15 minutos**, en ellas se realiza una **retroalimentación** de qué se hizo el día ayer, qué se hará hoy y cuáles han sido los problemas que han surgido hasta el momento. El objetivo, es que el equipo establezca un plan para las próximas 24 horas.
3) **Reunión de revisión:** Se lleva a cabo **al final de cada sprint**, en ellas se exponen los puntos completados y los que no.
4) **Reunión de retrospectiva:** Una vez **culminado un sprint** se efectúa esta reunión, que tiene como objetivo que el equipo reflexione y saque como resultado **posibles acciones de mejora**. A ella, debe asistir todo el Equipo Scrum (Dueño de Producto, Equipo de Desarrollo y Scrum Master). Es una de las reuniones más importantes ya que es un espacio de reflexión y mejora continua.



3.- KANBAN

Proveniente de una palabra japonesa cuyo significado es **"Tarjeta Visual"** es un marco de trabajo que requiere una comunicación en tiempo real sobre la capacidad del equipo, utilizado para controlar el avance de trabajo en una línea de producción, en la cual se clasifican las tareas en sub-estatus, esto con la intención de determinar los niveles de productividad en cada fase del proyecto.

Para el desarrollo de software, gracias a su sencillez **KANBAN**, simplifica la planificación y la asignación de responsabilidades, en un tablero se representan los **procesos del flujo de trabajo**, cómo mínimo deben existir tres columnas (Pendiente, En Progreso, Terminado), la cantidad de tarjetas en estatus pendiente forma parte de lo solicitado por el cliente, aquellas colocadas en progreso dependerán de la capacidad del equipo de trabajo.

Las **tarjetas Kanban**, se deben desplazar por cada una de las diversas etapas de su trabajo hasta su finalización.

VENTAJAS KANBAN

A continuación, se listan las principales ventajas:

1. Planificación de tareas.
2. Tiempos de ciclos reducidos.
3. Rendimiento del equipo de trabajo.

4. Métricas visuales.
5. Menos cuellos de botella.
6. Entrega continua.

Las metodologías ágiles comparten ciertas características, buscan la interacción de los miembros del grupo de trabajo, siempre con la meta de satisfacer los requisitos del cliente. Estas no se limitan tan sólo a desarrollos de software, con ellas se pueden gestionar cualquier tipo de proyectos. Es recomendable que las empresas apliquen estos métodos para eliminar el desperdicio que generan los esfuerzos sin planificación, las reuniones que consumen tiempo y no generan productividad ante alguna iniciativa, entre otros aspectos.

¿Por qué utilizar metodologías ágiles?

Simplificar en la sofisticación
Los Software de metodología ágil son realmente complejos, con lo que uno de los principios que se han de implantar en una empresa es ofrecer un **enfoque hacia la simplicidad.** Eliminar lo que no es esencial facilita el trabajo por parte de todos los miembros del equipo.

En resumen, las aplicaciones de los sistemas Agile deben parecer simples por parte de todos, sino es que realmente se está haciendo algo mal.

Demasiadas reuniones derivan en menos trabajo hecho
En la gestión de un proyecto, en el desarrollo de una estrategia, las reuniones han de ser breves y productivas. No resulta productivo para nadie asistir a un sinfín de reuniones en el que no se obtengan respuestas. **El tiempo es oro y hay que aprovecharlo.**

Pensar en global antes que actuar en lo particular
Dentro del desarrollo de los proyectos con la implantación de los sistemas Agile, uno de los puntos más importantes que destacó Holler fue la **necesidad de pensar en global**, en la visión de un todo, antes de actuar en lo particular.

Hacer cambios en pequeñas partes puede provocar durante todo el proceso demasiados riesgos que afecten al trabajo de todo el equipo.

Se trabaja por sprints, pero el objetivo es un producto final
Una de las ventajas de la implementación de los sistemas agile es la posibilidad de ir efectuando pruebas sobre el desarrollo del producto. Esto repercute en ir sobre seguro y comprobar la calidad durante todo el proceso y no depender de una prueba final que pueda poner en riesgo todo el proyecto.

METODOLOGÍA SCRUM

La **metodología Scrum** es un marco de trabajo o framework que se utiliza dentro de equipos que manejan proyectos complejos. Es decir, se trata de una **metodología de trabajo ágil** que tiene como finalidad la entrega de valor en períodos cortos de tiempo y para ello se basa en tres pilares: la transparencia, inspección y adaptación. Esto permite al cliente, junto con su equipo comercial, insertar el producto en el mercado pronto, rápido y empezar a obtener ventas.

¿En qué se basa la metodología Scrum?
Al estar enmarcada dentro de las metodologías agile, Scrum se basa en aspectos como:

1. La **flexibilidad** en la adopción de cambios y nuevos requisitos durante un proyecto complejo.
2. El factor **humano.**
3. La **colaboración** e interacción con el cliente.
4. El desarrollo iterativo como forma de asegurar buenos **resultados.**

Los pilares o **características de la metodología Scrum** más importantes son 3:

1. Transparencia. - Con el método Scrum todos los implicados tienen conocimiento de qué ocurre en el proyecto y cómo ocurre. Esto hace que haya un entendimiento "común" del proyecto, una visión global.
2. Inspección. - Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera autoorganizada.
3. Adaptación. - Cuando hay algo que cambiar, el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir el éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

Roles en el equipo Scrum

Con la **metodología Scrum**, el equipo tiene como foco entregar valor y ofrecer resultados de calidad que permitan cumplir los objetivos de negocio del cliente.

Para ello, los equipos de Scrum son **auto-organizados y multifuncionales**. Es decir, cada uno es responsable de unas tareas determinadas y de terminarlas en los tiempos acordados. Esto garantiza la entrega de valor del equipo completo, sin necesidad de ayuda o la supervisión minuciosa de otros miembros de la organización.

En Scrum existen 3 roles muy importantes: **Product Owner, Scrum Master y Equipo de desarrollo.**

1. Product owner.- Es el responsable de maximizar el valor del trabajo del equipo de desarrollo. La maximización del valor del trabajo viene de la mano de una buena gestión del Product Backlog, el cual explicaremos más adelante.

   El **Product owner** es el único perfil que habla constantemente con el cliente, lo que le obliga a tener muchos conocimientos sobre negocio.

   Para finalizar, un **equipo Scrum** debe tener solo un Product Owner y este puede ser parte del equipo de desarrollo.

2. Scrum Master. - Es el responsable de que las técnicas Scrum sean comprendidas y aplicadas en la organización. Es el manager de Scrum, un líder que se encarga de eliminar impedimentos o inconvenientes que tenga el equipo dentro de un *sprint* (que ya revisaremos en detalle más adelante), aplicando las mejores técnicas para fortalecer el equipo de marketing digital.

   Dentro de la organización, el Scrum Master tiene la labor de ayudar en la adopción de esta metodología en todos los equipos.

3. Equipo de desarrollo. - Son los encargados de realizar las tareas priorizadas por el Product Owner. Es un equipo multifuncional y auto-organizado. Son los únicos que estiman las tareas del product backlog, sin dejarse influenciar por nadie.

Los **equipos de desarrollo** no tienen sub-equipos o especialistas. La finalidad de esto es transmitir la responsabilidad compartida si no se llegan a realizar todas las tareas de un sprint.

Los hitos de la Metodología de trabajo Scrum

La gráfica describe los hitos dentro del **proceso Scrum**. El desarrollo iterativo se realiza en un sprint, que contiene los siguientes eventos: **sprint planning, daily meeting, sprint review y sprint retrospective.**

1. Sprint. - El sprint es el corazón de Scrum, es el contenedor de los demás hitos del proceso. Todo lo que ocurre en una iteración para entregar valor está dentro de un *sprint*. La duración máxima es de un mes, el tiempo se determina en base al nivel de comunicación que el cliente quiere tener con el equipo. Los sprints largos pueden hacer que se pierda feedback valioso del cliente y poner en peligro el proyecto.

2. Sprint planning.- En esta reunión todo el equipo Scrum define qué tareas se van a abordar y cuál será el objetivo del *sprint*. La primera reunión que se hace en el *sprint* puede llegar a tener una duración de 8 horas para sprints de un mes.

   El equipo se hace las siguientes preguntas:
   a. **¿Qué se va a hacer en el *sprint*?** En base a ello, se eligen tareas del Product backlog.
   b. **¿Cómo lo vamos a hacer?** El equipo de desarrollo define las tareas necesarias para completar cada ítem elegido del Product Backlog.

   La definición de qué se va a hacer implica que **el equipo tenga un objetivo y se encuentre comprometido** con la entrega de valor que se hará al cliente al final del sprint. A esto se le llama ***sprint goal.***

   El resultado de esta reunión es el ***sprint goal y un sprint backlog***.

3. Daily meeting. - Es una reunión diaria dentro del sprint que tiene como máximo 15 minutos de duración. En ella debe participar, sí o sí, el equipo de desarrollo y el Scrum Master. El Product Owner no tienen necesidad de estar presente.

   En esta reunión diaria el equipo de desarrollo hace las siguientes tres preguntas:
   - ¿Qué hice **ayer**?
   - ¿Qué voy a hacer **hoy**?
   - ¿Tengo algún **impedimento** que necesito que me solucionen?

   Esta reunión es la más oportuna para poder inspeccionar el trabajo y poder adaptarse en caso de que haya cambio de tareas dentro de un *sprint*.

4. Sprint review.- La review del valor que vamos a entregar al cliente se hace en esta reunión, al final de cada sprint. Su duración es de 4 horas para sprints de un mes, y es la única reunión de Scrum a la que puede asistir el cliente. En ella el Product Owner presenta lo desarrollado al cliente y el equipo de desarrollo muestra su funcionamiento. El cliente valida los cambios realizados y además brinda feedback sobre nuevas tareas que el Product Owner tendrá que agregar al Product backlog.

5. Sprint retrospective.- La retrospectiva es el último evento de Scrum, tiene una duración de 3 horas para Sprints de un mes, y es la reunión del equipo en la que se hace una evaluación de cómo se ha implementado la metodología Scrum en el último *sprint*.

   Es una gran oportunidad para el **equipo Scrum** de inspeccionarse a sí mismo, proponiendo mejoras para el siguiente sprint.

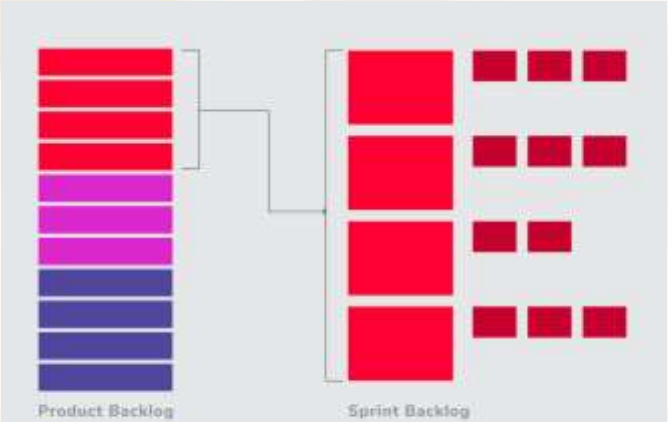
   El resultado: una lista de mejoras que debe aplicar el siguiente día, ya que, al finalizar la retrospectiva, inmediatamente comienza un nuevo *sprint*, que incluye el *sprint planning, daily meeting, sprint review* y el ya mencionado sprint *retrospective*.

Herramientas para la metodología Scrum

Las herramientas que se utilizan en Scrum están definidas para maximizar la **transparencia** dentro del equipo; es decir, que todos tengan una misma visión de lo que está ocurriendo en el proyecto.

Las **herramientas principales de Scrum** son: *product backlog y sprint backlog*.

1. Product backlog



Product Backlog & Sprint Backlog: Scrum Methodology

Básicamente, el product backlog es el **listado de tareas que engloba todo un proyecto**. Cualquier cosa que debamos hacer debe estar en el *product backlog* y con un tiempo estimado por el equipo de desarrollo.

La responsabilidad exclusiva de ordenar el *product backlog* es del Product Owner, que se encuentra en constante comunicación con el cliente para asegurarse de que las prioridades están bien establecidas.

La ordenación también es 100% responsabilidad del Product Owner, **por lo que las tareas que están más arriba deben de ser las de mayor prioridad.**

El equipo de desarrollo elige tareas del *product backlog* en el *sprint planning* para generar tanto el *sprint backlog* como el *sprint goal*.

2. Sprint backlog. - Es el grupo de tareas del *product backlog* que el equipo de desarrollo elige en el *sprint planning* junto con el plan para poder desarrollarlas. Debe ser conocido por todo el equipo, para asegurarse de que el foco debe estar en este grupo de tareas.

   El *sprint planning* no cambia durante el ***sprint,*** solo se permite cambiar el plan para poder desarrollarlas.

Ventajas y desventajas de la metodología Scrum

Una vez que sabemos cómo funciona esta metodología, hablemos de sus ventajas y desventajas:

- Ventajas de la metodología Scrum
  - Scrum es muy **fácil de aprender**: los roles, hitos y herramientas son claros y tienen un objetivo por lo que es un método muy relacionado con nuestra manera diaria de trabajar.

- El cliente puede comenzar a **usar el producto rápidamente.**
- Se agiliza el proceso, ya que la entrega de valor es muy frecuente.
- Menor probabilidad de sorpresas o imprevistos, porque el cliente está viendo frecuentemente el proyecto.

- Desventajas de la metodología Scrum
  - Aunque Scrum sea fácil de aprender, es muy **difícil implementarlo**. Esto supone una predisposición y un cambio de cultura de la organización que debe ir desde los altos mandos hasta los clientes.
  - La necesidad de tener **equipos multidisciplinares** puede ser un problema, ya que es difícil encontrar personas que sean capaces de hacer todo el trabajo de un equipo.
  - El equipo puede tender a realizar el camino más corto para conseguir el objetivo de un *sprint*, el cual no siempre ofrece resultados de calidad.

En definitiva, **Scrum** es especialmente interesante para proyectos en los que el objetivo es la entrega de valor continua al cliente para poder empezar a ver resultados lo antes posibles. Además, esta metodología permite agilizar procesos, practicar la transparencia y motivar al equipo a través de la autonomía y la independencia.

A continuación, vamos a ver como la metodología SCRUM rompe con el sistema de trabajo tradicional jerárquico y se enfoca en la auto-organización de un equipo para realizar proyectos de alta complejidad con la posibilidad de realizar pequeñas entregas parciales al cliente que le den una **idea del producto final**. De esta forma se consigue realizar proyectos complejos en el menor tiempo posible y con garantía de satisfacción. Mejora la productividad del equipo y establece una relación directa entre el cliente y el proveedor.

Ejemplo:

A continuación, vamos a ver como la metodología SCRUM rompe con el sistema de trabajo tradicional jerárquico y se enfoca en la auto-organización de un equipo para realizar proyectos de alta complejidad con la posibilidad de realizar pequeñas entregas parciales al cliente que le den una **idea del producto final**. De esta forma se consigue realizar proyectos complejos en el menor tiempo posible y con garantía de satisfacción. Mejora la productividad del equipo y establece una relación directa entre el cliente y el proveedor

Fase de planificación

1. Un cliente propietario de una Oficina privada que se encarga de las gestiones o trámites administrativos de particulares, profesionales, sociedades y empresas. necesita una aplicación que ayude a sus empleados a agilizar los procesos administrativos y de presentación de impuestos de sus empresas clientes.

2. El cliente deberá reunirse con el "Product Owner". La tarea del dueño del producto consiste en tomar nota de lo que el cliente necesita para ajustar el producto final lo más posible a la idea que el cliente tiene en su cabeza. Después, en base a esa información, priorizará las tareas a realizar.

3. El "Product Owner" o dueño del producto realiza entonces una DIVISIÓN del producto en PILAS DE PRODUCTO. En el caso de este ejemplo, cada pila podría representar la parte específica de la aplicación que se encargaría de una gestión administrativa determinada: Nóminas, Impuestos, Facturas, Contabilidad, Plan financiero, etcétera.

4. En este paso ya el "Scrum Master", el cual es un miembro del equipo, se encarga de comunicar las necesidades trasmitidas por el "Product Owner" y de estimar el coste de creación en tiempo y recursos de cada pila de producto.

5. Una vez entregado el presupuesto al cliente, este ordena las pilas de producto según su orden de prioridad. Será el cliente quien deberá decidir qué pila de producto tiene interés porque se realice primero según su urgencia o importancia.

Fase de creación del proyecto

6. Un vez aprobadas y reordenadas las pilas de producto por el cliente el equipo comienza su trabajo desglosando cada pila de producto en tareas menores llamadas "Pilas de Sprint".

7. Las "Pilas de Sprint" tienen como objetivo fraccionar el trabajo en tareas más pequeñas que **agilicen el proceso** elaboración del proyecto. La propia pila de sprint se ordena por prioridad por el "Product Owner", el cual a su vez ha consultado de nuevo las preferencias con el cliente antes de que el equipo comience el trabajo.

8. Comienza entonces el trabajo real y se convoca con la mayor frecuencia posible una reunión de equipo donde se comentan las tareas realizadas el día anterior y las que se van a realizar a continuación.

9. Después de la finalización de todas las pilas que componen un Sprint, el cliente ya puede ver el resultado de una pila de producto. De esta forma el cliente ya tiene un primer contacto con el producto final y puede dar su opinión. Además, puede reordenar de nuevo la pila de producto si así lo desea.

10. Por último, es de buena costumbre que el equipo celebre una reunión donde se analice lo ocurrido durante las fases que componen el Sprint y la realización de las pilas de producto del proyecto. Normalmente esta reunión se celebra fuera de las oficinas y con comida y bebida de por medio. El objetivo es promover las buenas relaciones personales y la complicidad del equipo de trabajo.

Aprendizajes esenciales			
Carrera:	Programación	Semestre:	**5**
Módulo/Submódulo:	**Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos** **Submódulo 1: Construye Bases de datos para aplicaciones Web**		

Aprendizajes esenciales o Competencias esenciales 1er parcial	Estrategias de Aprendizaje	Productos a Evaluar
1.- Implementa el diseño conceptual y diseño lógico de la base de datos	1.1 El estudiante identifica los *conceptos básicos de bases de datos* y elabora un resumen utilizando el manual de apoyo de CBDD (**Anexo 1**). 1.2 Haz una lectura comprensiva del tema *modelo entidad relación* apoyándose en el manual de apoyo CBDD, elabora uno de los ejercicios propuestos para este tema. **Ejercicios Modelo entidad relación** a) Se desea almacenar información sobre personas y los autos que eventualmente posean. Una misma persona puede tener varios autos, estos se identifican con el número de la matrícula y las personas mediante un el IFE. Se ha de almacenar la fecha en que una determinada persona adquirió un determinado auto**.** b) Se desea almacenar información sobre empresas y sucursales de empresas. Una empresa puede tener varias sucursales repartidas geográficamente. Una sucursal determinada debe pertenecer a una sola empresa. Las sucursales se numeran correlativamente para cada empresa. 1.3 Hacer una lectura comprensiva del tema *modelo relacional* apoyándose del manual de apoyo CBDD, posteriormente utilizando el resultado del tema	1.1 Resumen conceptos básicos de base de datos 10% 1.2 Ejercicio Modelo E-R 20% 1.3 Ejercicio Modelo Relacional 20%

	modelo entidad relación, elaborado en la actividad anterior, realiza el modelo relacional correspondiente, resaltando la normalidad mínima de las tablas. 1.4 Desarrolla la Práctica Integradora donde aplique los conocimientos adquiridos sobre los temas abordados, instrucciones: a) En la Biblioteca del plantel, se desea llevar el control de la existencia de material bibliográfico, usuarios que ingresan a la misma, así como registrar cada uno de los préstamos que el mismo usuario realiza sobre el material bibliográfico, diseña el modelo entidad relación, modelo relacional y su normalidad mínima. 1.5 Elabora en tu cuaderno una síntesis de reflexión sobre lo que se aprendió, como se aprendió y que estrategias aplicará para mejorar su proceso de aprendizaje. 1.6 Integra el portafolio de evidencias en tu cuaderno de trabajo.	1.4 Práctica integradora 30 % 1.5 Síntesis 10% 1.6 Portafolio 10%
Aprendizajes esenciales o Competencias esenciales 2º parcial	Estrategias de Aprendizaje	Productos a Evaluar
2.- Implementa el diseño físico de la base de datos	Responde la siguiente pregunta: ¿Cómo han influenciado el uso de las bases de datos a lo largo de la historia, en nuestra vida cotidiana y laboral? 2.1 Utiliza el manual de apoyo 2 (Anexo 2), haz una lectura comprensiva para identificar los siguientes conceptos: *Creación de una base de datos en MySQL ¿Qué es MYSQL? ¿Qué es SQL?,* al finalizar la lectura elabora un mapa conceptual del tema. 2.2 Apoyándose en el manual de apoyo 2, realiza una lectura comprensiva del tema *creación de bases de datos y tablas en MySQL*, utilizando la consola, elabora un cuadro sinóptico para recuperar la información más importante del tema.	Participación durante el 2° parcial 10% 2.1 Mapa conceptual 10% 2.2 Cuadro sinóptico 10% 2.3 Prácticas 30%

	2.3 Elabora las prácticas guiadas y autónomas en tu cuaderno, simulando el uso del símbolo del sistema cmd, para elaborar bases de datos y tablas en MySQL. Si tienes dudas sobre el tema consulta tu manual de apoyo. 2.4 Elabora en tu cuaderno una práctica integradora, simulando el uso de la consola cmd, donde involucres como mínimo 3 tablas en una base de datos, identifiques correctamente los campos de cada tabla, su campo llave y llave foránea, modifiques su estructura, insertes registros, elimines registros y se desarrollen consultas básicas. 2.5 Elabora una síntesis de reflexión sobre lo que aprendió, cómo lo aprendió y qué estrategias aplicará para mejorar su proceso de aprendizaje el próximo parcial. 2.6 Integración del portafolio de evidencias (en el cuaderno de trabajo).	2.4 Practica integradora 20% 2.5 Síntesis 10% 2.6 Portafolio de evidencias 10%
Aprendizajes esenciales o Competencias esenciales 3er parcial	Estrategias de Aprendizaje	Productos a Evaluar
3.- Administra la información de la base de datos	3.1 Utilizando el manual de apoyo N°3 CBBD (**Anexo 3**) realiza una lectura compresiva del tema *lenguaje de consulta y las diversas cláusulas que puedes incorporar para filtrar la información que contienen las tablas de las bases de datos,* utiliza un organizador gráfico para identificar la información más importante del tema. 3.2 Desarrolla las prácticas guiadas y autónomas del tema *Lenguaje de consulta*. 3.2.1 Realiza una lectura comprensiva del tema *Manejadores gráficos de bases de datos con PHPmyadmin,* identificando su entorno de trabajo. En su cuaderno de trabajo desarrolle el código que permita realizar las practicas guiadas y autónomas simulando el entorno de PHPmyadmin.	3.1 Organizador gráfico 10% 3.2 Prácticas desarrolladas 30% 3.3 Práctica integradora 40%

	3.3 Elaborar práctica integradora donde administre una base de datos con mínimo de 4 tablas, con sus respectivos campos, tipos de datos y consultas, con phpmyadmin. 3.4 Síntesis de reflexión de lo que aprendió en el semestre y estrategias para fortalecer su aprendizaje. 3.5 Integración del portafolio de evidencias **(en el cuaderno de trabajo)**.	3.4 Síntesis 10% 3.5 Portafolio de evidencias 10%

Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos
Submódulo 1: Construye Bases de datos para aplicaciones Web

Anexo 1

Manual de apoyo CBDD

Conceptos básicos

En la conformación de una base de datos **se pueden seguir diferentes modelos y paradigmas**, cada uno dotado de características, ventajas y dificultades, haciendo énfasis en su estructura organizacional, su jerarquía, su capacidad de transmisión o de interrelación

Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso, análisis y/o transmisión. Existen actualmente muchas formas de bases de datos, que van desde una biblioteca hasta los vastos conjuntos de datos de usuarios de una empresa de telecomunicaciones.

Las bases de datos son el **producto de la necesidad humana de almacenar la información**, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnética n, etc. Esto se conoce como *modelos de base de datos* y permite el diseño y la implementación de algoritmos y otros mecanismos lógicos de gestión, según sea el caso específico.

Toda **base de datos** está formada por uno o varios bloques de información llamados TABLAS (Inicialmente denominados FICHEROS o ARCHIVOS) que normalmente tendrán una característica en común.

Una **TABLA** o archivo de datos es un conjunto de información del mismo tipo; por ejemplo, en una base de datos de una biblioteca, una tabla estará constituida por la información relativa a todos los libros de la misma, otra tabla contendrá información sobre los lectores, etc.

Cada tabla está formada por REGISTROS. Un **registro** es la unidad elemental de información de la tabla o fichero. En la tabla o fichero de libros, un registro estaría constituido por la información correspondiente a cada libro concreto, con su título, autor, editorial, etc. Cada registro está formado por uno o más elementos llamados CAMPOS.

Un **campo** es cada una de las informaciones que interesa almacenar en cada registro y es, por tanto, la unidad elemental de información del registro.

El manejo de las bases de datos se lleva mediante sistemas de gestión (llamados *DBMS* por sus siglas en inglés: *Database Management Systems* o Sistemas de Gestión de Bases de Datos), actualmente digitales y automatizados, que **permiten el almacenamiento ordenado y la rápida recuperación de la información**.

Gracias a la aparición de estos programas de usuario es posible realizar la gestión de tablas de una base de datos, sin tener que realizar programas que procesen esos datos, facilitando todas las operaciones de creación, actualización, consulta y creación de informes con los datos recogidos.

MySQL es un sistema gestor de bases de datos relacionales, que además ofrece compatibilidad con PHP, C y HTML, y funciones avanzadas de administración y optimización de bases de datos para facilitar las tareas habituales.

Modelo Entidad – Relación

Los diagramas o modelos entidad-relación son una herramienta para el modelado de datos de un sistema de información.

Elementos del modelo entidad-relación

Entidad

Las entidades representan *cosas* u *objetos* (ya sean reales o abstractos), que se diferencian claramente entre sí.

Para poder seguir un ejemplo durante el artículo añadiré ejemplos sobre un taller mecánico, donde se podría crear las siguientes entidades:

- **Coches** (objeto físico): contiene la información de cada taller.
- **Empleado** (*objeto* físico): información de los trabajadores.
- **Cargo del empleado** (*cosa* abstracta): información de la función del empleado.

Estas entidades se representan en un diagrama con un rectángulo, como los siguientes.

Atributos

Los atributos definen o identifican las características de entidad (**es el contenido de esta entidad**). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...).

Siguiendo el ejemplo de antes podemos analizar los atributos de nuestra entidad "**Coches**", que nos darán información sobre los coches de nuestro supuesto taller.

Unos posibles atributos serían los siguientes: *número de chasis*, *matrícula*, *DNI del propietario*, *marca*, *modelo* y muchos otros que complementen la información de cada coche.

Los atributos se representan como círculos que descienden de una entidad, y no es necesario representarlos todos, sino los más significativos, como a continuación.

Relación

Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.

Por ejemplo, los empleados del taller (de la entidad "**Empleados**") tienen un cargo (según la entidad "**Cargo del empleado**"). Es decir, un atributo de la entidad "*Empleados*" especificará que cargo tiene en el taller, y tiene que ser idéntico al que ya existe en la entidad "*Cargo del empleado*".

Las relaciones se muestran en los diagramas como rombos, que se unen a las entidades mediante líneas.

Modelo relacional

Lo primero que hay que tener presente a la hora de diseñar una base de datos relacional es el propio concepto de modelo relacional, que organiza los datos en una base de datos como una colección de tablas teniendo inicialmente:

- Cada tabla un nombre que la identifica unívocamente.
- Cada tabla tiene una o más columnas, que están dispuestas en un orden específico de izquierda a derecha.
- Cada tabla tiene cero o más filas, conteniendo cada una un único valor en cada columna. Las filas están desordenadas.

A su vez las tablas están relacionadas unas con otras por los datos que contienen. El modelo de datos relacional utiliza claves primarias y claves secundarias (externas o foráneas) para representar estas relaciones entre tablas. A la hora de definir las claves primarias y secundarias es necesario tener presente de entrada lo siguiente:

- Una **clave primaria** es una clave única elegida entre todas las candidatas que define unívocamente a todos los demás atributos de la tabla, para especificar los datos que serán relacionados con las demás tablas. La forma de hacer esto es por medio de claves foráneas o secundarias.

- Una **clave foránea** es una referencia a una clave en otra tabla, determina la relación existente en dos tablas. Las claves foráneas no necesitan ser claves únicas en la tabla donde están y sí a donde están referenciadas.

Por ejemplo, el código de departamento puede ser una clave foránea en la tabla de empleados. Se permite que haya varios empleados en un mismo departamento, pero habrá uno y sólo un departamento por cada clave distinta de departamento en la tabla de departamentos.

NORMALIZACIÓN DE UNA BASE DE DATOS

La normalización es un proceso que pone las cosas en su sitio, haciéndolas normales. En una base de datos, este término tiene un significado matemático específico, realizando una separación de elementos de datos (tales como nombres, direcciones u oficios) en grupos afines y definiendo las relaciones normales o "correctas" entre ellos.

La normalización es una técnica eficaz para el diseño de bases de datos que puede aplicarse tanto a sistemas relacionales como a otros modelos. Con la técnica de la normalización se trata de evitar la dependencia entre inserciones, actualizaciones y borrado de elementos de las tablas de la base de datos. La normalización tiene tres etapas que transforman las relaciones no normales en normalizadas y que se denominan primera, segunda y tercera formas normales.

PRIMERA FORMA NORMAL

El primer paso en la normalización es poner los datos en la primera forma normal. Esto se hace situando los datos en tablas separadas, de manera que los datos de cada tabla sean de un tipo similar, y dando a cada tabla una clave primaria y un identificador o etiqueta única. Esto elimina los grupos repetidos de datos.

Supongamos que, inicialmente, tenemos toda la información a incluir en una base de datos en una sola tabla de nombre EMPLEADO cuyos campos pueden ser Nombre, Edad, Alojamiento, Responsable, Dirección, Oficio1, Oficio2 y Oficio3.

SEGUNDA FORMA NORMAL

El segundo paso, es la segunda forma normal, se centra en aislar los datos que sólo dependen de una parte de la clave. Para obtener la segunda forma normal, se deben sacar Oficio y Descripción a una tercera tabla. La clave primaria de la tercera tabla es Oficio y su larga descripción aparece sólo una vez. Si echamos un vistazo a la tabla OFICIO de la primera forma normal, las descripciones detalladas se repiten para cada trabajador que tenga ese "oficio". Además, cuando el último trabajador con conocimientos de herrero se marcha de la ciudad, la descripción del oficio de herrero se desvanece.

TERCERA FORMA NORMAL

El tercer paso, la tercera forma normal, implica deshacerse de cualquier cosa de las tablas que no dependa únicamente de la clave primaria. La información de los trabajadores sobre el alojamiento depende de dónde estén viviendo (si se trasladan habrá que actualizar la fila con el nuevo nombre del alojamiento en el que vive), pero el patrón de la posada y su dirección son independientes de si el trabajador vive ahí o no. Por eso la información de alojamiento se lleva a una tabla separada de nombre VIVIENDA y, por conveniencia, se usa una versión taquigráfica del nombre de la casa de alojamiento como clave primaria y el nombre total se guarda en el campo NombreCompleto.

Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos
Submódulo 1: Construye Bases de datos para aplicaciones Web

Anexo 2

Manual de apoyo 2

CREACIÓN DE UNA BASE DE DATOS en MySQL

Para crear bases de datos en MySQL, podemos utilizar la consola de comandos, independientemente del sistema operativo que se esté utilizando.
Antes de comenzar se debe tener instalado MySQL en un ordenador local o un servidor remoto.
MYSQL es un sistema de gestor de bases de datos relacionales, desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, que además ofrece compatibilidad con PHP, C y HTML y funciones avanzadas de administración y optimización de bases de datos incluye capacidades de análisis integradas, servicios de transformación y duplicación de datos y funciones de programación mejoradas.

Lenguaje formal estructurado SQL

(Structured Query Language) Es un lenguaje estructurado de integración y administración de bases de datos relacionales. Numerosas firmas comerciales han introducido productos de bases de datos relacionales orientado al SQL, entre ellos Transact-SQL, que es el lenguaje de SQL Server 2000, y SQL de Microsoft Yet, que es el lenguaje de SQL de Microsoft Access. Ambos lenguajes pueden utilizarse para el desarrollo de aplicaciones Web junto con el lenguaje SQL de MySQL, que se utilizaran más adelante.

Creación de una base de datos

Práctica 1

1. Abrir el panel de control de XAMPP en el menú inicio (start) o en el escritorio como acceso directo.
2. En el cuadro de diálogo hacer clic sobre el botón Start situado a la derecha del nombre MySQL, al presionarlo, éste cambiará a Stop y el nombre MySQL se remarcará de color verde, tal como se muestra en el cuadro de diálogo siguiente:
3. Minimizar el cuadro de diálogo anterior.

Acceder a mysql desde la consola.

Lo primero que tenemos que hacer para poder ejecutar comandos es ejecutar el programa '**mysql**', si ya está instalado en el ordenador o servidor, bastaría con ejecutar desde la consola

4**.-** Seleccionar de la barra de tareas de Windows la opción buscar y escribir cmd, aparecerá un cuadro de diálogo como el siguiente selecciona símbolo del sistema.

Al hacer esto aparece un cuadro de diálogo similar al que se muestra a continuación:

Escribir enseguida del o los usuarios lo siguiente:

cd c:\xampp\mysql\bin y dar enter indicamos donde se encuentra XAMPP y MySQL, ahí se guardaran las bases de datos creadas

Aparece c:\xampp\mysql\bin> escribir en esta parte lo siguiente:
MySQL -h localhost -u root -p y presiona Enter para indicar el uso de MySQL
Aparece Enter Password: No se debe de escribir, sólo presionar Enter.
Al finalizar el paso 4, aparece un cuadro de diálogo como el que se muestra a continuación:
5. Ahora llega el momento de crear una nueva base de datos MySQL CREATE DATABASE agenda;
6. Teclear la instrucción para el uso de la base de datos. Use agenda;
7. Ejecuta **SHOW DATABASES** para visualizarla
Crear tablas
8. Crear una tabla sencilla llamada contactos, con los campos nombre, domicilio, y celular, indicando su tipo de dato y la longitud del mismo
Create table contactos (nombre varchar(30), direccion varchar(50), celular (int);
9. Para comprobar los campos de la tabla que acabamos de crear podemos utilizar el comando **DESCRIBE** con el nombre de la tabla
DESCRIBE contactos;
10. Para terminar vamos a añadir contenido a nuestra base de datos desde consola.
INSERT INTO contactos VALUES ('Jose Sanchez', 'Calle 3 Av. 15', 66);
11. Ahora simplemente vamos a comprobar que la base de datos o mejor dicho, la tabla contactos tiene los datos que acabamos de introducir.
SELECT * FROM contactos;
12. Cerrar MySQL con presionar dos veces exit en cada línea de comandos, seguidamente dar clic en Stop, y por último cerrar presionando el botón en el cuadro de diálogo de XAMPP.

Práctica 2:
Objetivo: En una práctica guiada, el estudiante utilizará la sintaxis correcta para la creación de una base de datos en MySQL.
Pasos:

1. Abrir XAMPP
2. Seleccionar de la barra de tareas la opción buscar y escribir cmd (**Command), símbolo del sistema**
3. Abrir MySQL como se muestra en la pantalla siguiente:

4. Enseguida de MariaBD> se teclean las instrucciones para crear la base de datos
alumnos nombrebdd
6. Crear una tabla llamada alumnos, con los campos no_control, nombre, especialidad y edad, indicando su tipo de dato y la longitud del mismo, además asigna un campo llave
create table alumno (no_control varchar(5), nombre varchar(30), especialidad varchar(20), edad int, primary key (no_control));
7. Para comprobar los campos de la tabla que acabamos de crear podemos utilizar el comando **DESCRIBE** con el nombre de la tabla
8. Añadir 5 registros a nuestra base de datos desde consola.

INSERT INTO nombretabla VALUES ('columna1', 'columna2', …..);
El primer registro debe corresponder a tus datos
9. Ahora simplemente vamos a comprobar que la base de datos o mejor dicho, la tabla contactos tiene los datos que acabamos de introducir.
SELECT * FROM nombretabla;
10. muestra el resultado de tu práctica al docente.
11. Cerrar MySQL con presionar dos veces exit en cada línea de comandos, seguidamente dar clic en Stop, y por último cerrar presionando el botón stop en el cuadro de diálogo de XAMPP.

Práctica. 3
Nombre del Alumno: _____
Objetivo: En una práctica autónoma, el estudiante utilizará la sintaxis correcta para la creación de una base de datos en MySQL, tablas, insertar registros, borrar tablas, borrar bases de datos.
Instrucciones: Realiza los pasos en tu cuaderno simulando el uso del símbolo del sistema
Pasos:

1. Abrir XAMPP
2. Seleccionar de la barra de tareas la opción buscar y escribir cmd (**Command), símbolo del sistema**
3. Abrir Mysql
4. Crear una base de datos en MySQL con el nombre de abarrotes
5. Crear una tabla llamada **productos**, la cual deberá contener 4 campos diferentes (id_producto, nombre_producto, precio_producto, proveedor) se deberá asignar **una clave primaria** a esta tabla, la cual debe ser **id_producto**, en este campo no se puede repetir el contenido, es decir, si en un registro se insertó el valor 1234 no se podrá utilizar para otro registro.
6. Crear una tabla diferente, la cual se deberá llamar **clientes** la cual contará con 3 campos en esta tabla (id_producto, id_cliente, nombre_cliente,) se asignará también **una clave primaria,** la cual debe ser **id_cliente**, Asimismo se debe asignar **una clave foránea**, la cual servirá para crear la relación entre las tablas, para esto se utilizará el campo **id_producto**, y se hará referencia a su uso en la tabla productos en el campo **id_producto,** la creación de estas dos tablas se muestra a continuación:
7. Insertar 5 registros en cada tabla

Nota: el primer registro sea tu nombre

8. Mostrar las bases de datos existentes para la confirmación de la creación de la base de datos **abarrotes**. Show
9. Mostrar las tablas que se crearon en la base de datos abarrotes. Show
10. Mostrar los registros que se insertaron en las tablas creadas (clientes y productos). Recuerda que tus tablas deben tener 5 registros cada una
11. Utiliza el comando count para contar los registros.
12. Utiliza el comando sum. **Sumar los registros de una tabla, con el campo precio_producto**
13. Cierra cmd y XAMPP

sentencias para crear y borrar una BBDD:

- CREATE DATABASE
- DROP DATABASE

Modificar la estructura de las tablas

```
ALTER TABLE `contactos` CHANGE COLUMN`nombre` `nombre_c` varchar (50) NOT NULL;
 ALTER TABLE `contactos` MODIFY `celular` varchar (10) NOT NULL;
DROP TABLE nombre_tabla
DELETE FROM nombre_tabla where (condición)
DELETE FROM usuarios WHERE edad >35
```

Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos
Submódulo 1: Construye Bases de datos para aplicaciones Web

Anexo 3

Manual de apoyo N° 3 CBBD

Claúsula Select

Una **cláusula SELECT** se usa para especificar los nombres de los campos que contienen los datos que quiere usar en una consulta. ... Si la instrucción **SQL** tiene dos o más campos con el mismo nombre, debe agregar el nombre del origen de datos de cada campo al nombre del campo en la **cláusula SELECT**.

Las **cláusulas** son las condiciones que modifican nuestras consultas y son utilizadas para definir los datos que desea seleccionar o manipular.

La sentencia **SELECT** se utiliza para recuperar información de la base de datos, y puede proyectar las columnas seleccionadas, es decir, realizar un filtro sobre la tabla o tablas originales y recuperar solamente datos de las columnas filtradas.

Claúsula From

En una instrucción SELECT, la **cláusula** FROM especifica las tablas o consultas **que** contienen los datos **que** se usarán en la **cláusula** SELECT. Puede usar corchetes para delimitar el nombre.
Sintaxis

Select * from nombre_tabla;

ORDER BY que tiene como finalidad ordenar los resultados de las consultas por columnas en vez del campo índice por defecto.

La cláusula GROUP BY es un comando SQL que se usa para **agrupar filas que tienen los mismos valores**.
La cláusula GROUP BY se utiliza en la instrucción SELECT. Opcionalmente se usa junto con funciones agregadas para producir informes resumidos de la base de datos.
Eso es lo que hace, **resumiendo los datos** de la base de datos.
Las consultas que contienen la cláusula GROUP BY se denominan consultas agrupadas y solo devuelven una sola fila para cada elemento agrupado.

GROUP BY Sintaxis

Ahora que sabemos cuál es la cláusula GROUP By, veamos la sintaxis para un grupo básico por consulta.
Sentencias SELECT ... GROUP BY column_name1 [, column_name2, ...] [HAVING condition];

- "Sentencias SELECT …" es la consulta estándar del comando SQL SELECT.
- " **GROUP BY** *column_name1* " es la cláusula que realiza la agrupación basada en column_name1.
- "[, column_name2, …]" es opcional; representa otros nombres de columna cuando la agrupación se realiza en más de una columna.

- "[TENER condición]" es opcional; se usa para restringir las filas afectadas por la cláusula GROUP BY. Es similar a la cláusula WHERE.

Claúsula Where

Este es uno de un conjunto de artículos sobre Access SQL. En este artículo se describe cómo escribir una cláusula WHERE y se usan ejemplos para mostrar diferentes técnicas que puede usar en una cláusula WHERE.
En una instrucción SQL, la cláusula WHERE especifica criterios que tienen que cumplir los valores de campo para que los registros que contienen los valores se incluyan en los resultados de la consulta.
Sintáxis
Select nombre_campos from nombre_tabla where condición;

La **cláusula** "**limit**" se usa para restringir los registros que se retornan en una consulta "select". Recibe 1 ó 2 argumentos numéricos enteros positivos; el primero indica el número del primer registro a retornar, el segundo, el número máximo de registros a retornar.
Sintaxis
SELECT nombre_campos FROM nombre_tabla LIMIT no_registros;

La claúsula **DISTINCT** nos devuelve valores únicos. En una tabla, una columna puede contener valores duplicados; y algunas veces sólo se necesita un listado de los valores diferentes.

SINTAXIS SELECT DISTINCT SQL
La claúsula DISTINCT se añade a las sentencias SELECT, justo después de la palabra clave SELECT.

SELECT DISTINCT column_name,column_name FROM table_name;

Columnas calculadas

Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un campo de una tabla

En la imagen siguiente tenemos una tabla con los siguientes campos, con la cual obtendremos una columna nueva ganancia con la información obtenida de los campos precio_lista y precio_publico

Para lo anterior aplicamos la siguiente instrucción

Se obtiene el siguiente resultado:

Consultas con condiciones de búsqueda

Ejemplos:
SELECT nombre, apellidos FROM empleados WHERE estado_civil != 'soltero'
SELECT id_producto, total FROM ventas WHERE id_producto= '345pasta';
SELECT id_producto, total FROM ventas WHERE precio_publico<= 35;

Consultas multitabla

Es posible seleccionar datos de diferentes tablas mediante una solo consulta, esto se realiza con la sentencia SELECT
Consultando 2 tablas a la vez:
1

JOIN de dos tablas

INNER JOIN selecciona todas las filas de las dos columnas siempre y cuando haya **una coincidencia entre las columnas en ambas tablas**. Es el tipo de JOIN más común.

```
SELECT nombreColumna(s)
FROM tabla1
INNER JOIN tabla2
ON tabla1.nombreColumna=table2.nombreColumna;
```

Manejadores gráficos de bases de datos

PHP myadmin

phpMyAdmin es una herramienta de software gratuita escrita en PHP , destinada a manejar la administración de MySQL a través de la Web. phpMyAdmin admite una amplia gama de operaciones en MySQL y MariaDB. Las operaciones de uso frecuente (administración de bases de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) se pueden realizar a través de la interfaz de usuario, mientras aún tiene la capacidad de ejecutar directamente cualquier declaración SQL.

Creación de bases de datos

Para iniciar con la sesión en phpmyadmin, debemos igual activar el panel de control de xampp, después debemos de teclear en el navegador **https://localhost/phpmyadmin/ .**

Al pulsar sobre el nombre de la base, en el panel de la izquierda, se despliegan los nombres de todas las tablas.
En la parte superior que analizaremos posteriormente. Las funciones de estos botones son:
Examinar, estructura, buscar, insertar, vaciar y eliminar.
Los enlaces (botones) superiores del panel de la derecha tienen funciones específicas, para manejar toda la base. Estas son las funciones:

Crear una base de datos
Para crear una base de datos en phpmyadmin, se activa mysql del panel de control XAMPP , se indica en el navegador **https://localhost/phpmyadmin/**, luego damos clic en la parte izquierda de la pantalla donde dice nueva, en la parte derecha se abre un cuadro de texto para indicar el nombre de la base de datos y el cotejamiento

Ejemplo:
Dentro del campo de texto escribamos cursos (en minúsculas), pues ese será el nombre de nuestra **nueva base de datos.**
Luego de escribir el nombre, elegimos el juego de caracteres que almacenaremos (para textos en español será el utf8_spanish_ci, que corresponde al español tradicional, y permite que utilicemos la ñ y la ch y ll).
Finalmente, pulsamos el **botón Crear** y, a continuación, el nombre de la base recién creada aparecerá en la **columna de la izquierda**, como también aparece el nombre de la base de datos activa en la ruta superior (breadcrumb o migas de pan) que siempre nos indica donde estamos parados:
Si hemos usado el XAMPP podremos entrar con el programa Mi PC (o cualquier otro explorador de archivos), hasta llegar a C:/servidor/XAMPP/mysql/data/ y allí encontraremos una carpeta por cada base de datos que hayamos creado; en este caso, vemos, al lado de las bases que vienen por defecto, nuestra nueva base "cursos":

Crear Tablas

En este punto, ya estamos listos para crear nuestra **primera tabla** dentro de nuestra flamante base de datos (recordemos que una base de datos es una **simple carpeta** que organiza nuestras tablas, pero los lugares donde se almacenan realmente los datos son **las tablas**).
Para ello, primero haremos un clic en la columna izquierda, sobre el nombre de la base dentro de la cual queremos crear una tabla (nuestra base llamada "cursos" aún no tiene ninguna tabla creada).
Esto recargara la parte derecha de la pantalla, y veremos un mensaje avisando que todavía no hay tablas en la base.
Ahora podremos crear una tabla muy fácilmente en la base de datos, simplemente escribiendo el **nombre de la tabla** que creamos y la **cantidad de campos** (columnas) que deseamos que posea.
Crearemos una tabla llamada **"mensajes"** cuyo fin será almacenar el nombre, el correo electrónico y un mensaje que irán dejando los usuarios: id, nombre, email y mensaje.
Número de columnas = significa la cantidad de campos que tendrá nuestra tabla
Presionar el botón continuar.
Luego de pulsar el botón Continuar, aparecerá la siguiente pantalla, en la que tendremos que escribir los nombres de cada uno de los cuatro campos o columnas que tendrá nuestra tabla.
1. En el primer campo de texto, justo debajo del título que dice **"Campo"**, escribiremos el **nombre de cada campo** (id, nombre, email, mensaje), en ese orden, uno debajo de otro, todos en la primera columna.
2. En la **segunda columna**, denominada **Tipo**, elegiremos el tipo de dato que podrá almacenar cada uno de estos campos. Ya veremos muy pronto otros tipos de datos posibles, pero por ahora adelantemos que los tipos de datos normalmente más utilizados son **INT** (integer, es decir, números enteros, sin decimales, como

los que precisa el campo id), **VARCHAR** (variable carácter o caracteres variables, que almacena letras y números, hasta un máximo de **255 caracteres**, como los necesarios para los campos nombre y email), y **TEXT** (para textos **mayores de 255 caracteres**, como los de nuestro campo mensaje). Así que elegiremos estos tipos de datos en la **columna "Tipo"**.
3. En la **tercera columna**, definiremos la **cantidad máxima de caracteres** que almacenará cada campo (cuatro dígitos para el id- suponemos que no tendremos nunca más de 9999 mensajes-), 60 dígitos para cada "nombre" y cada "mail", agregaremos que en los campos de tipo TEXT como "mensaje" no deberemos poner nada en longitud, ya que debe quedar vacía.
4. Ahora nos desplazamos hacia la derecha de pantalla. En la **columna Nulo**, si dejamos de sin marcar la casilla de selección, haremos que es campo sea **NOT NUL**L; es decir, **será obligatorio que le completemos algún valor** cuando agreguemos un registro.
Si no queremos que esto sea obligatorio y que se pueda dejar vacío y que se añada igual el registro completo con el resto de los campos que si se hubieran completado, entonces marcamos esa casilla de selección, lo que equivale a definir ese campo como potencialmente **NULL**, o sea que pueda ser nulo o vacío.
5. Ahora, exclusivamente en el reglón pertenece al campo id (el primero) deberemos elegir en la **columna Indice** la opción **Primary**, tal como vamos en la imagen anterior, lo que indica que ese campo será el que identificará cada registro **de forma única será su clave primaria**.
Además de lado del menú de selección, marcaremos la casilla con la abreviatura **A_I** (Auto Increment), que hace que el contenido o "valor" de este campo id, sea completado automáticamente cada vez que agreguemos un registro, con números que se irán incrementando de uno en uno, sin repetirse nunca.
6. Ahora pulsemos el botón **Grabar** o **Guardar**.
Ahora veamos cómo **modificar campos** en tablas de una base de datos.
Primero selecciona la opción estructura, la cual permite modificar los campos de una tabla.
Identifica el campo a modificar y presiona en la columna acción cambiar
Guarda los cambios presionando el botón guardar

Agregar una columna nueva (campo)
En agregar se indica la cantidad de campos a añadir y después de que campo ya existente y das click en botón continuar
Se indica el nombre y características del campo
Presiona el botón guardar

Eliminar una columna (campo) de una tabla
Selecciona el campo y en la columna acción, selecciona eliminar
Se confirma la eliminación

A continuación, realizaremos con ella los procesos más necesarios en una base de datos: **agregar**, **modificar** y **borrar datos**.

Agregar datos en una tabla, Se selecciona la opción insertar
En la pantalla que aparece rellenar los cuadros de texto con la información correspondiente

Presionar el botón continuar.

En Examinar se muestra el contenido de la tabla

Modificar el contenido de la tabla
Seleccionamos la opción editar del registro seleccionado

En la pantalla que aparece cambia la información que desees y presiona el botón continuar

Eliminar registros de una tabla
Se da clic en la acción borrar
Confirmar la eliminación
Exportar bases de datos
Consultas SQL indica la instrucción y presiona el botón

Aprendizajes esenciales			
Carrera:	Programación	Semestre:	**5**
Módulo/Submódulo:	**Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos** **Submódulo 2: Desarrolla aplicaciones Web con conexión a bases de datos**		

Aprendizajes esenciales o Competencias esenciales 1er parcial	Estrategias de Aprendizaje	Productos a Evaluar
1.- Integra conceptos básicos de un servidor web.	1.1 Realiza investigación documental sobre*: Servidores HTTP, Lenguajes de Servidor y Sistemas Gestores de Bases de Datos,* escribe en tu cuaderno las definiciones de los conceptos (citar la fuente). 1.2 En base a la información recabada, desarrolla un mapa mental. 1.3 Realiza investigación documental sobre: *el modelo cliente servidor y del Hardware mínimo necesario para un servido*r. 1.4 En base a la información recabada, elaborar una infografía que explique el modelo Cliente servidor y el Hardware mínimo necesario para servidores.	1.1 Anotaciones de la investigación documental de Servidor HTTP, Lenguajes de Servidor y Sistemas Gestores de Bases de Datos (20%) 1.2 Mapa mental (30%) 1.3 Anotaciones de la investigación documental de modelo Cliente Servidor y Hardware mínimo necesario (20%) 1.4 Infografía (30%)
Aprendizajes esenciales o Competencias esenciales 2º parcial	Estrategias de Aprendizaje	Productos a Evaluar
2.- Prepara una computadora como un webserver	2.1 Realiza la práctica N°1 Preparar una computadora como un webserver (enfocado netamente a Windows), (**Anexo 1**)	2.1 Práctica N°1
Aprendizajes esenciales o Competencias esenciales 3er parcial	Estrategias de Aprendizaje	Productos a Evaluar
3.- Usa PHP para insertar datos en MySQL utilizando MySQLi y PDO	3.1 Realiza la práctica N°2 Usa PHP para insertar datos en MySQL utilizando MySQLi y PDO	3.1 Práctica N°2

Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos
Submódulo 2: Desarrolla aplicaciones Web con conexión a bases de datos

Anexo 1

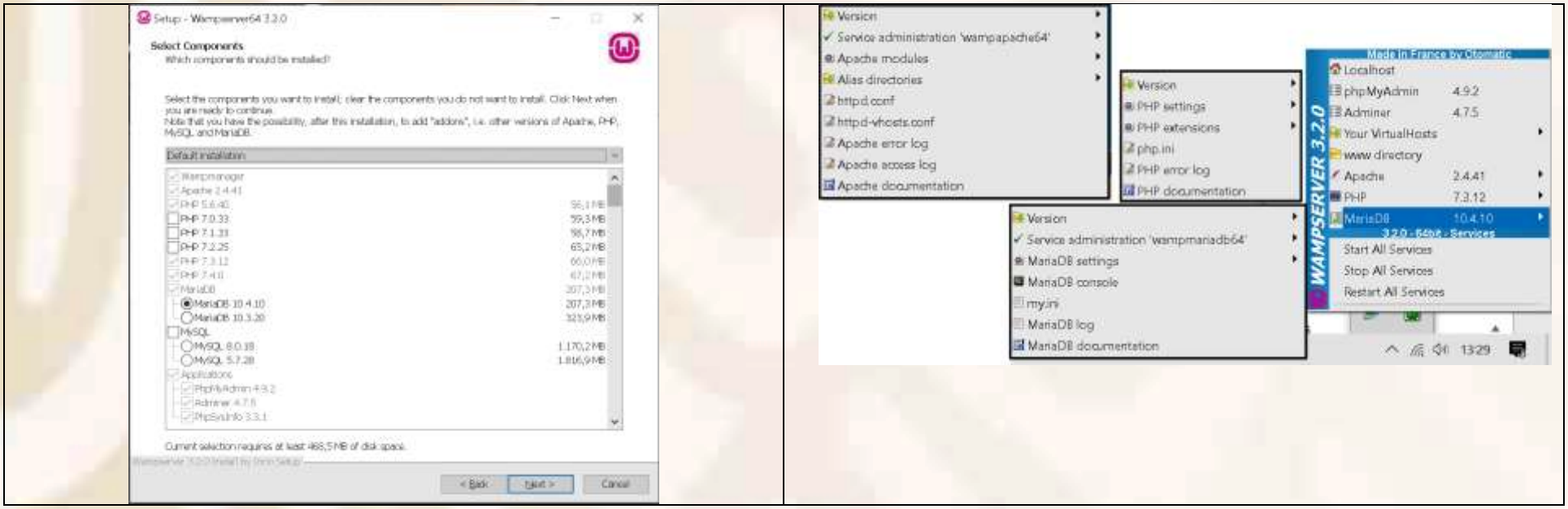
Práctica N°1 Preparar una computadora como un webserver (enfocado netamente a Windows)

En cualquier computadora o en un cibercafé próximo a tu casa o escuela, realiza las siguientes actividades, con la finalidad de preparar una computadora como un webserver

Wampserver es un paquete asistido para la instalación sencilla de un servidor web Apache, con MySQL y PHP en Windows. Permite a usuarios sin conocimientos de sistemas instalar un servidor web de manera simple. Todo el proceso de instalación se realiza a través de un asistente por lo que las tareas de configuración del webserver pasan desapercibidas.
Una vez instalado podemos crear nuestros propios sitios web en nuestro ordenador. Incluso podemos instalar gestores de contenidos web como WordPress para facilitar las tareas de gestión de nuestro site.

A continuación, estudiaremos como instalar el servidor WAMP, dar de alta nuevos sites, a crear servidores con varios dominios web, y para finalizar haremos referencia a administración de bases de datos MySQL y personalización del código PHP tan utilizado en las sites actuales. Como es habitual, también haremos a los mejores vídeos online relacionados …

Instalación inicial



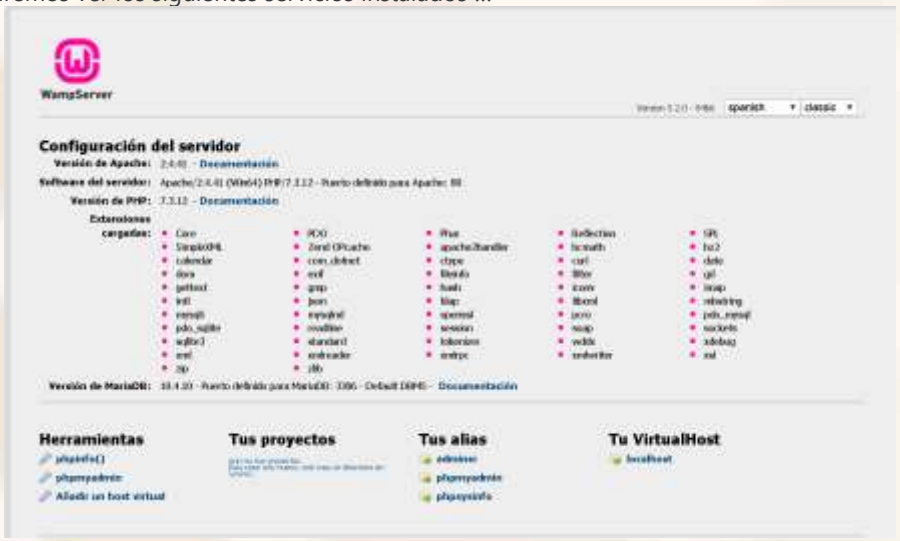
Principales funcionalidades.
A través del botón derecho del ratón sobre el icono de servidor Wamp podemos mostrar en el menú contextual:

- Gestionar sus servicios de Apache* y MySQL.
- Poner en línea (para todo el público) y offline su servidor (localhost).
- Instalar y cambiar la configuración de los servidores Apache2, MySQL y PHP.
- Acceder a sus registros.
- Crear alias para disponer diferentes sites en mismo servidor.
- Cambiar el idioma por defecto.

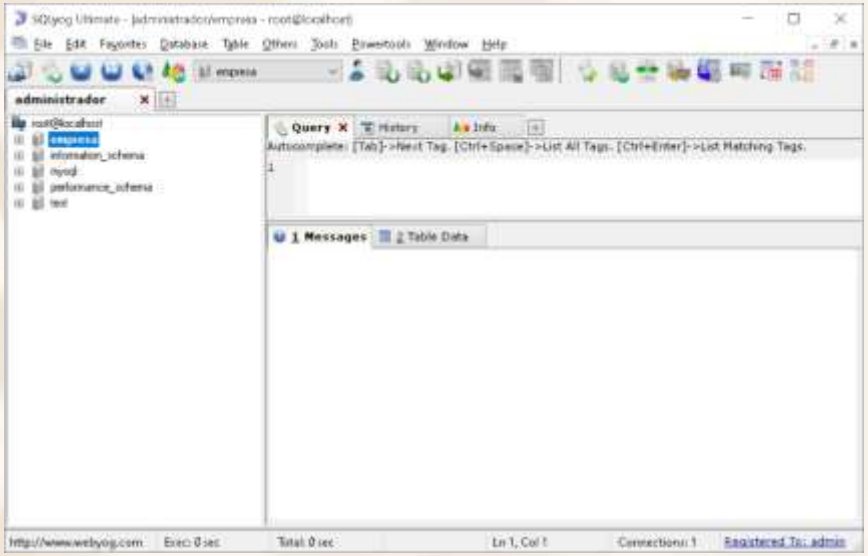
Notas: El servidor web Apache es el más utilizado en el mundo, se calcula su cuota está entorno al 70%. Incluso su predominio ha calado en entornos empresariales.

Primeros pasos …

Una vez instalado Wampserver, en localhost podremos ver los siguientes servicios instalados …



Con un monitor (GUI) para bases de datos SQL tipo SQLyog podemos gestionar de forma visual nuestra base de datos MariaDB o MySQL …



Descarga e instalación de Wampserver.

Haz doble clic en el archivo descargado y sigue las instrucciones. Todo es automático El paquete WampServer se entrega con las últimas versiones de Apache, MySQL y PHP.

Una vez que WampServer está instalado, puede agregar manualmente versiones adicionales de Apache, PHP o MySQL (solo VC9, VC10 y VC11 compiladas). Se proporcionarán explicaciones en el foro.

Cada versión de Apache, MySQL y PHP tiene su propia configuración y sus propios archivos (datos para MySQL).

Usando wampserver: El directorio «www» se creará automáticamente (generalmente c: \ wamp \ www)

Crea un subdirectorio en «www» y coloca tus archivos PHP dentro.

Haz clic en el enlace «localhost» en el menú de WampSever o abra tu navegador de Internet y vaya a la URL: http: // localhost

Con esto preparamos la computadora para realizar bases de datos y vincularlas a programas utilizando programación dinámica usando el lenguaje de programación PHP (el alumno debe tener conocimientos previos de HTML, CSS y JavaScript, y muy preferentemente jQuery.

Como interactuar con una base de datos MySQL usando PHP

Existe una variedad de lenguajes de *script* y motores de bases de datos, pero a una de las mejores alternativas en cuanto a costo y velocidad la conforman PHP y MySQL. Además de ser extremadamente rápido, utilizo este sistema de base de datos para mis sitios por estar bien documentado. Con solo pocas líneas de código PHP se pueden administrar datos guardados en una base MySQL, agregando, eliminando o actualizando campos y registros.

Para comenzar con los ejemplos de este artículo debemos crear una nueva base de datos. Esto se puede hacer de diferentes maneras, ya que existen distintos programas para administrar bases de datos, como el conocido phpMyAdmin; otra forma es desde la línea de comandos del Server haciendo un Telnet.

En este ejemplo asumo que tienes instalado y configurado phpMyAdmin adecuadamente para manejar bases y tablas. La mayoría de ISP y proveedores de hosting ofrecen este servicio ya instalado para los clientes. Siguiendo las instrucciones del PhpMyAdmin fácilmente podrás crear la base que utilizaremos para los ejemplos, el nombre de la base será **ejemplo**.

Tabla de usuarios

Con la primera base creada pasamos a construir la tabla usuarios que estará compuesta como se muestra a continuación en la tabla:

Campo
Tipo
id
Entero
Autonumérico
Valor Default 0
Clave principal
nombre
Varchar 50
apellido
Varchar 50
dni
Entero

Las propiedades de los campos se configuran en phpMyAdmin al crear la tabla. Con la base y la tabla creada adecuadamente podemos pasar a escribir código PHP.

Código PHP

A continuación, crearemos cinco *scripts* que se encargarán de realizar las funciones básicas sobre una base de datos MySQL, estos archivos serán: conexion.php, cerrar_conexion.php, guardar.php, ver.php y actualizar.php.

Como primer paso crearemos la conexión con el MySQL, el script correspondiente será conexion.php, el cual estará incluido utilizando la función include (nombre_archivo) de PHP, en todos los demás *scripts* del sitio que trabajen con la base de datos para establecer la conexión ahorrando líneas de código. Sin una conexión establecida no se podrá trabajar con la base de datos.

En el archivo conexion.php deben ser configuradas un par de variables correspondientes a nuestro servidor: el host, usuario y password para acceder al MySQL, además indicar que base se utilizara en la conexión.

Entonces podemos abrir el bloc de notas, escribir las siguientes líneas de código y guardarlas con el nombre conexion.php:

Archivo conexion.php

```
<?$dbhost="localhost";

// host del MySQL (generalmente localhost)$dbusuario="agustin"; // aqui debes ingresar el nombre de usuario // para acceder a la base$dbpassword="mipass"; // password de acceso para el usuario de la // linea anterior$db="ejemplo"; // Seleccionamos la base con la cual trabajar$conexion = mysql_connect($dbhost, $dbusuario, $dbpassword);mysql_select_db($db, $conexion);?>
```

Con este archivo creado podemos empezar a trabajar con la base desde nuestros siguientes *scripts*.

Sería correcto cerrar la conexión abierta al terminar de trabajar con la base, para ello creamos un nuevo archivo con solamente una línea de código:

Archivo cerrar_conexion.php

```
<? mysql_close($conexion); ?>
```

Esta línea cierra la conexión con el motor MySQL abierta en el archivo conexion.php, este archivo será incluido en el final de todos los *scripts* siguientes.

Es el turno ahora del archivo guardar.php que se encargara de insertar registros en la base mostrando un formulario mediante el cual el usuario ingresara los nuevos datos. Entonces creamos un nuevo documento de texto con el siguiente contenido:

Archivo guardar.php

```
<?include "conexion.php";if (!isset($accion)){
    echo "<html> <head><title>Guardar datos en la base</title></head> <body><h3>Guardar datos en la base</h3><form name="form1" method="post" action="guardar.php?accion=guardar"> <p>Nombre:<br> <input type="text" name="nombre"> </p> <p>Apellido:<br> <input type="text" name="apellido"> </p> <p>DNI:<br> <input type="text" name="dni"> </p> <p> <input type="submit" name="Submit" value="Guardar Datos"> </p></form></body></html>";}elseif($accion=="guardar"){ include"conexion.php"; $result=mysql_query("INSERT INTO usuarios (id,nombre, apellido, dni) VALUES ('',$nombre,$apellido,$dni) ",$conexion); echo" <html> <head></head> <body> <h3>Los datos han sido guardados</h3> </body> </html>";}include "cerrar_conexion.php";?>
```

Claro está que se omitieron muchas líneas de código destinadas a checar que los datos ingresados por el usuario fueron correctos y solamente nos limitamos a guardar los datos en la base.

El *script* que escribiremos a continuación toma los datos de la base y mediante una iteración forma una tabla en la cual se representara la lista de datos ordenada por el campo nombre.

Archivo ver.php

```
<?include "conexion.php";$result=mysql_query("SELECT * FROM usuarios ORDER BY nombre", $conexion);echo"<table width=300><tr><td><b>Nombre</b></td><td><b>Apellido</b></td><td><b>DNI</b></td></tr>";while($row=mysql_fetch_row($result)){ echo"<tr><td>$row[1]</td><td>$row[2]</td><td>$row[3] <a href="actualizar.php?id=$row[0]">Actualizar</a></td> </tr>";}echo"</table>";include "cerrar_conexion.php";?>
```

Con los archivos creados hasta ahora podemos guardar datos en la base y ver los registros guardados. Al formarse la tabla con los resultados obtenidos de la base, notaremos que al lado de cada registro aparece un link llamado Actualizar, el cual hace referencia al archivo que crearemos a continuación. El link pasa como parámetro la variable id que contiene el número correspondiente al campo autonumérico del registro que actualizaremos.

El archivo siguiente muestra un formulario con los datos que actualmente contiene la base en el lugar indicado por la variable id, una vez llenado el formulario, el script actualiza la tabla de la base de datos con la nueva información ingresada por el usuario.

Archivo actualizar.php

```
<?include "conexion.php";if (!isset($accion)){ $result=mysql_query("SELECT * FROM usuarios WHERE id=$id", $conexion); $row=mysql_fetch_row($result); echo"<html> <head><title>Actualizar datos de la base</title></head> <body> <form action="actualizar.php?accion=guardar" method="POST"> Nombre:<br> <input type="text" value="$row[1]" name="nombre"><br> Apellido:<br> <input type="text" value="$row[2]" name="apellido"><br> DNI:<br> <input type="text" value="$row[3]" name="dni"><br> <input type="hidden" name="id" value="$row[0]"> <input type="submit" value="Guardar"> </form> </body> </html>";}elseif($accion==guardar){ $result=mysql_query("UPDATE usuarios SET nombre=$nombre, apellido=$apellido, dni=$dni WHERE id = $id",$conexion); echo" <html> <body> <h3>Los registros han sido actualizados</h3> </body> </html>";}include "cerrar_conexion.php";?>
```

Estos *scripts* muestran como rápidamente se puede administrar una base de datos on-line. El PHP trae incorporadas funciones para interactuar con distintos tipos de bases de datos, como Oracle, SQL Server, y más.

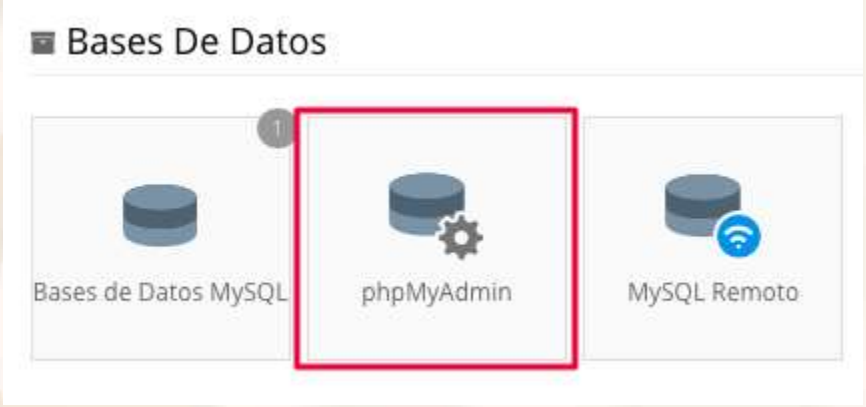
Módulo IV: Desarrolla software de aplicación web con almacenamiento persistente de datos
Submódulo 2: Desarrolla aplicaciones Web con conexión a bases de datos

Anexo 2

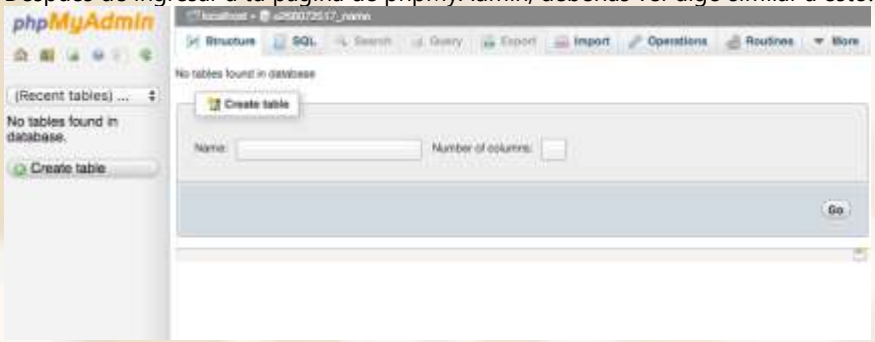
Práctica N°2 Cómo usar PHP para insertar datos en MySQL a través de PHPMYADMI

En cualquier computadora del cibercafé más próximo a tu casa o escuela realiza las siguientes actividades.

En este tutorial, aprenderá a **INSERTAR** datos en tu base de datos MySQL desde scripts **PHP**. Hay dos métodos que puedes usar, MySQLi y PDO. Crear una tabla (Opcional)
En primer lugar, debes crear una tabla para tus datos. Si ya has creado una, desplázate hacia abajo hasta la siguiente sección. Crear una tabla es un proceso simple que puedes hacer con la opción **phpMyAdmin**, que encuentras en tu panel de control de hosting.



Después de ingresar a tu página de phpMyAdmin, deberías ver algo similar a esto:



Crearemos una tabla llamada **Students** para nuestra base de datos **u266072517_name**. Puedes crear una nueva tabla haciendo clic en el botón **Create table**. Después de eso, verás esta nueva página donde puedes ingresar toda la información necesaria para tu tabla:

Esta es la configuración más simple que puedes usar para una tabla, para obtener más información sobre la estructura de la tabla/base de datos y qué tipo de configuración puedes usar con esas columnas, consulta la documentación oficial de **phpMyAdmin**.
Por ahora, aquí hay algunas explicaciones sencillas de las columnas que usamos:

- **Name**: Este es el nombre de tu columna. Se mostrará en la parte superior de tu tabla.
- **Type**: Puedes establecer un tipo de columna aquí. Por ejemplo, seleccionamos **varchar** porque necesitamos ingresar un tipo de cadena caracteres de nombre aquí (que tenga letras, no números).
- **Length/Values**: Se usa para especificar la longitud máxima que tu entrada en esta columna puede tener.
- **Index**: Usamos el índice «Principal» para nuestro campo «ID». Al crear una tabla, se recomienda tener una columna de ID. Se utiliza para enumerar las entradas de la tabla y se requiere para configurar las relaciones de la tabla. También marqué «**A_I**«, lo que significa **Auto Incremento**. Esto servirá para enumerar automáticamente las entradas (1,2,3,4 ...).

Haz clic en **Save** para guardar y se creará tu tabla.

Código PHP para INSERTAR datos en una base de datos MySQL

Hay dos métodos que puedes usar para INSERTAR datos en tu base de datos MySQL. El método PHP MySQLi y el método PHP Data Object o PDO.

Método MySQLi

En primer lugar, debes establecer una conexión con una base de datos tomando como referencia nuestro tutorial anterior. Una vez hecho esto, podemos proceder con la consulta MySQL **INSERT**. Aquí hay un código de ejemplo completo con la conexión básica y los métodos de inserción:

- <?php
- $servername = "mysql.hostinger.co.uk";
- $database = "u266072517_name";
- $username = "u266072517_user";

```
$password = "buystuffpwd";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $database);
// Check connection
if (!$conn) {
die("Connection failed: " . mysqli_connect_error());
}

echo "Connected successfully";

$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')";
if (mysqli_query($conn, $sql)) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Entonces, la primera parte del código (líneas **3 a 18**) tiene como objetivo la conexión a la base de datos. No vamos a analizar esta parte de nuevo, pero si quieres saber qué significa cada línea del código, mira nuestra guía anterior sobre **cómo conectarte a una base de datos**.
Comencemos con la línea número **19**:

```
$sql = "INSERT INTO Students (name, lastname, email) VALUES ('Test', 'Testing', 'Testing@tesing.com')";
```

Esta es la línea más importante del código PHP, ya que es la que logra insertar datos en la base de datos MySQL. **INSERT INTO** es una declaración que agrega datos a la tabla de la base de datos especificada. En nuestro ejemplo, estamos agregando datos a la tabla **Students**.

Si continuamos, entre los corchetes, tenemos columnas de tabla específicas a las que queremos agregar los valores: **(name, last name, email)**. Los datos se agregarán en el orden especificado. Si escribiéramos **(email, last name, name)**, los valores serían agregados en un orden diferente.

La siguiente parte es la declaración de **VALUES**. Aquí especificamos nuestros valores para las columnas previamente especificadas. De esta forma, cada columna representa un valor específico. Por ejemplo, en nuestro caso sería así: **name = Test, lastname = Testing, email = Testing@testing.com**.

Algo más que vale la pena destacar es que acabamos de ejecutar una **consulta SQL** (SQL query, por su nombre en inglés) usando código PHP, las consultas SQL deben establecerse entre comillas. En nuestro ejemplo, todo lo que está entre comillas y después de **$sql =** es una consulta SQL.

La siguiente parte del código (líneas **20 a 22**) verifica si nuestra consulta fue exitosa:

- **if** (**mysqli_query**($conn, $sql)) {
- echo "New record created successfully";
- }

Simplemente muestra un mensaje de éxito si la consulta que ejecutamos fue exitosa.

Y la parte final (líneas **22 a 24**) muestra un mensaje diferente en caso de que nuestra consulta no fuera exitosa:

- else {
- echo "Error: " . $sql . "
" . **mysqli_error**($conn);
- }

Esto nos mostrará un mensaje de error en caso de que algo esté mal.

Método PHP Data Object (PDO)

Al igual que en el ejemplo anterior, primero necesitamos una conexión a la base de datos lo cual se realiza creando un nuevo objeto PDO; **este tutorial** te mostrará cómo hacerlo. Como la conexión a la base de datos MySQL es un objeto PDO, debes usar varios **métodos** PDO (cualquier función que sea parte de cualquier objeto) para preparar y ejecutar consultas. Los métodos de los objetos se llaman así:

- $the_Object->**the_Method**();

PDO te permite **preparar** el código SQL antes de que se ejecute. La consulta SQL se evalúa y se **corrige** antes de ejecutarse. Un ataque de inyección SQL simplificado podría hacerse simplemente escribiendo código SQL en un campo de un formulario. Por ejemplo:

- // User writes this in the username field of a login form
- john"; DROP DATABASE user_table;
-
- // The final query becomes this
- "SELECT * FROM user_table WHERE username = john"; DROP DATABASE user_table;

Como hay un código SQL sintácticamente correcto, el punto y coma hace que **DROP DATABASE user_table** sea una nueva consulta SQL, y tu tabla de usuario se borra. Las declaraciones preparadas no permiten los caracteres **«** y **;** para finalizar la consulta original y la instrucción maliciosa **DROP DATABASE** nunca se ejecutará.

Siempre deberías usar declaraciones preparadas al enviar o recibir datos de la base de datos con PDO.

Para usar declaraciones preparadas, debes escribir una nueva variable que llame al método **prepare()** del objeto de la base de datos.

En el código correcto:

- <?php
- $servername = "mysql.hostinger.com";
- $database = "u266072517_name";

```
$username = "u266072517_user";
$password = "buystuffpwd";
$sql = "mysql:host=$servername;dbname=$database;";
$dsn_Options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION];
// Create a new connection to the MySQL database using PDO, $my_Db_Connection is an object
try {
$my_Db_Connection = new PDO($sql, $username, $password, $dsn_Options);
echo "Connected successfully";
} catch (PDOException $error) {
echo 'Connection error: ' . $error->getMessage();
}
// Set the variables for the person we want to add to the database
$first_Name = "Test";
$last_Name = "Testing";
$email = "Testing@testing.com";
// Here we create a variable that calls the prepare() method of the database object
// The SQL query you want to run is entered as the parameter, and placeholders are written like this :placeholder_name
$my_Insert_Statement = $my_Db_Connection->prepare("INSERT INTO Students (name, lastname, email) VALUES (:first_name, :last_name, :email)");
// Now we tell the script which variable each placeholder actually refers to using the bindParam() method
// First parameter is the placeholder in the statement above - the second parameter is a variable that it should refer to
$my_Insert_Statement->bindParam(:first_name, $first_Name);
$my_Insert_Statement->bindParam(:last_name, $last_Name);
$my_Insert_Statement->bindParam(:email, $email);
// Execute the query using the data we just defined
// The execute() method returns TRUE if it is successful and FALSE if it is not, allowing you to write your own messages here
if ($my_Insert_Statement->execute()) {
echo "New record created successfully";
} else {
echo "Unable to create record";
}
```

```
// At this point you can change the data of the variables and execute again to add more data to the database
$first_Name = "John";
$last_Name = "Smith";
$email = "john.smith@email.com";
$my_Insert_Statement->execute();
// Execute again now that the variables have changed
if ($my_Insert_Statement->execute()) {
echo "New record created successfully";
} else {
echo "Unable to create record";
}
```

En las líneas **28, 29** y **30**, usamos el método **bindParam()** del objeto de la base de datos. También está el método **bindValue()** que es muy diferente.

- **bindParam()** – Este método evalúa los datos cuando se alcanza el método **execute()**. La primera vez que el script llega a un método **execute()**, ve que **$first_Name** corresponde a «Test», vincula ese valor y ejecuta la consulta. Cuando el script llega al segundo método **execute()**, ve que **$first_Name** ahora corresponde a «John», vincula ese valor y ejecuta nuevamente la consulta con los nuevos valores. Lo que es importante recordar es que definimos la consulta una vez y la reutilizamos con diferentes datos en diferentes puntos del script.
- **bindValue()** – Este método evalúa los datos tan pronto como se llega a **bindValue()**. Como el valor de **$first_Name** se definió como «Test» cuando se llegó a **bindValue()**, este se usará cada vez que se llame a un método **execute()** para **$my_Insert_Statement**.

Observa que reutilizamos la variable **$first_Name** y le damos un nuevo valor la segunda vez. Si revisas tu base de datos después de ejecutar este script, tienes los dos nombres definidos, a pesar de que la variable **$first_Name** equivale a «John» al final del script. Recuerda que PHP evalúa un script completo antes de ejecutarlo.
Si actualizas el script para reemplazar **bindParam** con **bindValue**, insertarás en MySQL «Test Testing» dos veces en la base de datos y John Smith será ignorado.

Hay que confirmar que todo funcione y resolver problemas comunes.
Si la consulta que ejecutamos e insertamos en la base de datos MySQL fue exitosa, veremos el siguiente mensaje:
Connect SuccessfullyNew record created successfully

Solución de errores comunes

Sin embargo, hay momentos en que el nuevo registro puede mostrar un error al insertar de SQL. Pero no te preocupes, hay algunas maneras en que puede solucionar estos errores de MySQL.

MySQLi

Si te aparece un mensaje de error de MySQLi, puedes aplicar los siguientes métodos para solucionarlo. Por ejemplo, si hay un error de sintaxis en tu código, verás algo similar a esto:

Connect successfullyError: INSERT INTO students {name, lastname, email} VALUES ('Test', 'Testing', 'Testing@testing.com')You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '{name, lastname, email} VALUES ('Test', 'Testing', 'Test@testingcom')' at line 1"

Como puedes ver, la primera parte del código está bien, la conexión se estableció con éxito, pero nuestra consulta SQL se encontró con un muro.

"Error: INSERT INTO Students {name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com') You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '{name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')' at line 1"

Hay un error de sintaxis que, lamentablemente, provocó el error en nuestro script. El error fue estuvo aquí:

$sql = "INSERT INTO Students {name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')";
Utilizamos corchetes en lugar de paréntesis simples. Como esto no está bien, el script arrojó un error de sintaxis.
PDO

En la línea 7 de la conexión PDO, el modo de error está configurado para **mostrar todas las excepciones**. Si esto quedara fuera del script y la consulta fallara, no recibirías ningún mensaje de error. Con las excepciones habilitadas, se muestra el problema específico.

Por lo general, esto solo debería utilizarse al desarrollar un script, ya que puedes exponer la base de datos y los nombres de tablas, cosa que quizás prefieras ocultar de cualquiera que intente acceder maliciosamente a tus datos. En el caso anterior donde se usaron corchetes en lugar de paréntesis normales, el error sería similar a este:

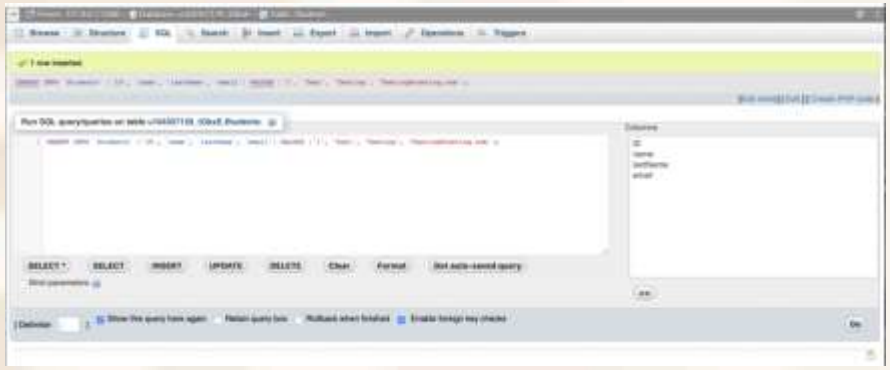
Fatal error: Uncaught exception 'PDOException' with message 'SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; <code>check the manual that corresponds to your MySQL server version for the right syntax to use near '{name, lastname, email} VALUES ('Thom', 'Vial', 'thom.v@some.com')' at line 1"</code>

Otros posibles problemas que puedes encontrar:

- Columnas incorrectas especificadas (columnas inexistentes o un error de ortografía).
- Un tipo de valor asignado a otro tipo de columna. Por ejemplo, si tratamos de asignar un número **47** a una columna de nombre (**Name**), obtendríamos un error porque se supone que debería ser un valor de texto. Pero si asignamos un número entre comillas, por ejemplo, **«47»**, eso funcionaría porque nuestro número se asignaría como un texto a la columna.
- Intentar ingresar datos en una tabla que no existe o cometer un error ortográfico en la tabla.

Todos esos errores se pueden solucionar fácilmente siguiendo las pautas del mensaje de error o **verificando el error log**.

Después de una entrada de datos exitosa, deberíamos ver la información agregada a nuestra base de datos. Aquí hay un ejemplo de la tabla a la que agregamos nuestros datos cuando los vemos desde **phpMyAdmin**.



Conclusión

En este tutorial, has aprendido a usar **PHP** para **INSERTAR** datos en una base de datos **MySQL** utilizando **MySQLi** y **PDO**. También aprendiste a solucionar los errores de conexión más comunes. Saber cómo usar PHP para insertar datos a una base de datos MySQL es útil ya sea que estés aprendiendo a programar o construyendo tu sitio web.