REPÚBLICA DEL ECUADOR

Ministerio de Educación

UNIDAD EDUCATIVA

“BAÑOS”

Baños de Agua Santa – Tungurahua

PROYECTO INTERMODULAR

“Temas de Tercero de Bachillerato del Currículo del Módulo Formativo Nº3: Programación y Base de Datos”

Figura Profesional: Informática

Elaborado por: Anahi Sailema

Pamela Aponte

Elkyn Zambrano

Curso: Segundo “H”

Módulo formativo:

Módulo 3: Programación y Base de Datos

Docentes Técnicos: Mg. Diana Fuentes

Año Lectivo: 2024-202

EL NUEVO ECUADOR

Índice

Índice 2
1. Estructuras de control: 3
1.1. Clases y funciones 7
1.2. Librerías 8
1.3. Código fuente 10
1.4. Desarrollo de programas 12
1.5. Documentación de programas 12
2. Herramientas de desarrollo 15
2.1. Generadores de pantallas (Interfaces de usuario) 15
2.2. Generadores de informes 17
2.3. Generadores de consultas 19
2.4. Generador de aplicaciones 20
3. **Generación y desarrollo de aplicaciones con herramientas CASE:** 22
3.1. Características de las Herramientas CASE 22
3.2. Estructura de las Herramientas CASE 25
3.3. Actualización y Mantenimiento del Proyecto 27
3.4. Generación de Código 29
3.5. Documentación generada 32
3.6. Tutoriales y Reportes de Desarrollo 33
3.7. Ventajas de Usar Herramientas CASE 36

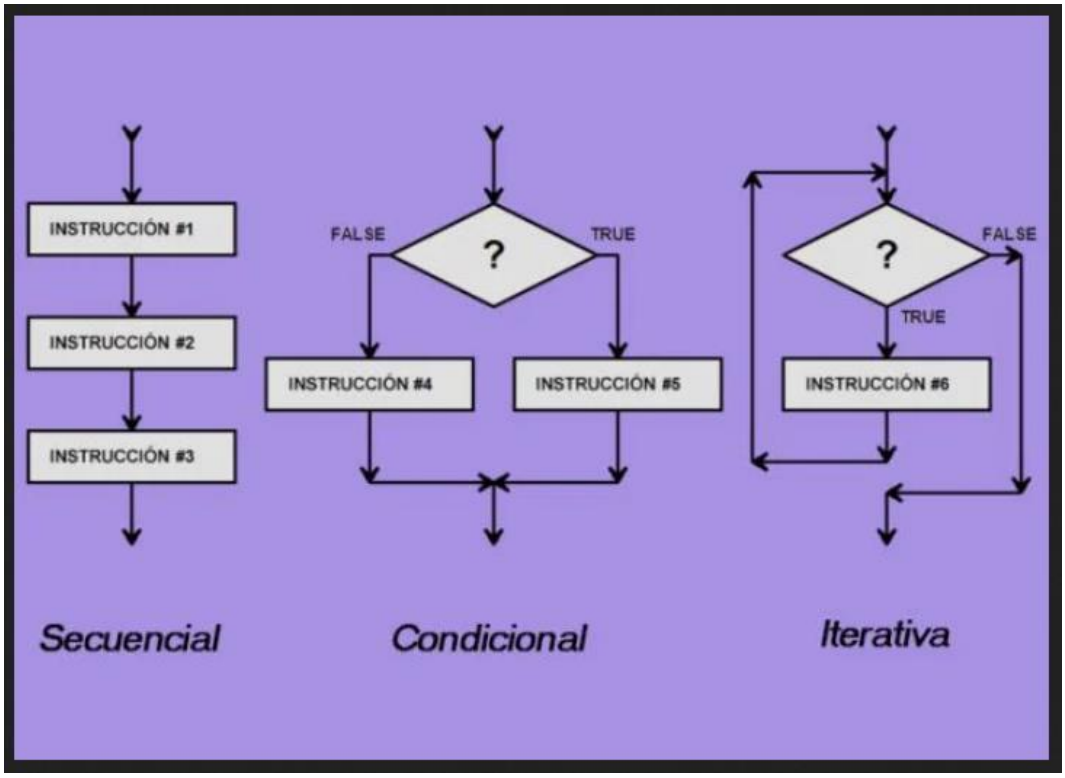
1. Estructuras de control:

Las estructuras de control en programación son herramientas que hacen posible que los algoritmos sigan un flujo lógico de acontecimientos y que gestionen el comportamiento de un programa, dadas ciertas condiciones específicas. En el día de hoy queremos explicarte más a fondo qué son las estructuras de control, cómo funcionan, sus tipos y cómo puedes usarlas en programación.

Las estructuras de control son un conjunto de reglas que hacen posible la gestión del flujo de ejecución de los programas, esto lo logran determinando el orden en que las instrucciones se ejecutan. Grosso modo, las estructuras de control permiten que el código tome decisiones, repita acciones o bien siga su flujo de ejecución secuencial.

Figura 1

Estructuras de control



Nota: Visualizacion de la imagen acerca de las estructuras de control. Fuente: Isuu (2025)

Tipos de estructuras de control

Las estructuras de control se dividen en tres categorías: secuenciales, condicionales e iterativas. Veamos en qué consiste cada una.

Estructuras de control secuenciales

Las estructuras de control secuenciales son las más básicas. En este caso, las instrucciones se ejecutan una tras otra en el mismo orden en que están escritas. Piensa en este tipo de estructuras como una fila de acciones que siguen un patrón fijo. Estas son la base de cualquier algoritmo, en donde cada acción sigue a la anterior de modo predecible.

Estructuras de control condicionales

Las condicionales son estructuras de control que permiten que el programa vaya tomando decisiones con base en ciertas condiciones. Existen varios tipos de estructuras condicionales, como las simples, las dobles y las múltiples, y cada una de ellas se utiliza para ejecutar diferentes bloques de código, esto dependiendo de si una condición es verdadera o falsa.

Condicionales simples

Una estructura condicional simple ejecuta un bloque de código solo si una condición se cumple:

```
if (edad >= 18) {
   console.log("Eres mayor de edad.");
}
```

En este caso, si la variable edad es mayor o igual a 18, el programa imprimirá «Eres mayor de edad». Si no, simplemente saltará ese bloque de código.

Condicionales dobles

Aquí se añaden dos posibles caminos: si la condición es verdadera, se ejecuta un bloque, y si es falsa, se ejecuta otro:

```
if (edad >= 18) {
   console.log("Puedes votar.");
} else {
   console.log("No puedes votar.");
}
```

Este código ofrece dos respuestas posibles, dependiendo del valor de la variable edad. Si es mayor, o tiene 18 años, aparecerá el mensaje «Puedes votar». De lo contrario, aparecerá el mensaje «No puedes votar».

Condicionales múltiples

Las condicionales múltiples o anidadas se usan para evaluar más de dos condiciones, ya que nos permiten manejar varios casos y tomar decisiones más complejas. Veamos un ejemplo:

```
if (edad < 18) {
   console.log("No puedes votar.");
} else if (edad >= 70) {
   console.log("Votar es opcional.");
} else {
   console.log("Debes votar obligatoriamente.");
}
```

Aquí, dependiendo de la edad, el programa imprimirá uno de tres posibles mensajes:

- ✓ Si es menor de 18 años, aparecerá el mensaje «No puedes votar».
- ✓ Si es mayor, o tiene 70 años, aparecerá el mensaje «Votar es opcional».

- ✓ Si no se cumple ninguna de las dos condiciones anteriores (que sea menor de 18 años o tenga 70 o más), aparecerá el mensaje «Debes votar obligatoriamente».

Estructuras de control iterativas

Las estructuras de control iterativas nos permiten repetir un conjunto de instrucciones varias veces. Se presenta cuando necesitamos realizar la misma acción en diferentes valores, o cuando no sabemos cuántas veces una condición será verdadera. Algunas de las más conocidas son:

Bucle for

Este tipo de bucle for repite un bloque de código un número fijo de veces:

```
for (let i = 0; i < 5; i++) {
    console.log("Iteración " + i);
}
```

En este ejemplo el bucle imprimirá cinco veces el texto «Iteración», cambiando el valor de i en cada repetición.

Bucle while

El bucle while sigue repitiendo un bloque de código mientras que una condición sea verdadera. Si no sabes de antemano cuántas veces se va a repetir una acción, debes usarlo:

```
let contador = 0;
while (contador < 5) {
    console.log("Contador: " + contador);
    contador++;
}
```

Este código imprimirá el valor de contador hasta que llegue a 5.

Bucle do-while

A diferencia del while, el bucle do-while siempre ejecuta el bloque de código al menos una vez, ya que evalúa la condición después de la primera ejecución:

```
let contador = 0;
do {
    console.log("Contador: " + contador);
    contador++;
} while (contador < 5);
```

Este bucle se asegurará de ejecutar el código una vez antes de comprobar si debe seguir repitiéndolo.

En programación existen muchísimos conceptos que son de gran importancia si estás en el proceso de convertirte en desarrollador en cualquier área. Para entender mejor de qué van todos estos términos, hemos creado el bootcamp de programación inicial, un curso en donde te beneficiarás del aprendizaje con los mejores profesores y podrás acceder a excelentes ofertas laborales. (Maldonado, 2025)

1.1.Clases y funciones

En programación, las funciones son bloques de código que realizan tareas específicas. Son una estructura esencial de código que permiten escribir programas más organizados y fáciles de entender.

¿Para qué sirven las funciones?

Dividen y organizan el código en partes más sencillas, encapsulan el código que se repite a lo largo de un programa para ser reutilizado, permiten escribir código más limpio y organizado, permiten calcular un valor de manera independiente al resto del programa.

¿Cómo funcionan las funciones?

- Reciben valores de entrada, llamados parámetros.
- Realizan operaciones, llamadas código de la función.
- Entregan un valor final, llamado resultado o valor de retorno.

¿Qué se debe tener en cuenta al escribir funciones?

- El uso que se le va a dar a la función.
- La definición de los datos que van a ingresar en la función.
- La definición de las variables que se van a necesitar dentro de la función.

El conjunto de pasos que se van a realizar en la función. (Abraira, 2024)

1.2.Librerías

Es un conjunto de funciones, clases, y recursos que puedes utilizar para hacer que tu código sea más eficiente. En lugar de escribir cientos de líneas de código desde cero, una librería te proporciona fragmentos ya hechos que puedes reutilizar. Esto ahorra tiempo y esfuerzo, lo cual es muy importante cuando trabajas en proyectos complejos.

Imagina que quieres construir una aplicación. Puedes escribir cada línea de código tú mismo o puedes apoyarte en librerías que otros desarrolladores han creado para tareas específicas como conectar con una base de datos o generar gráficos.

Facilitan el proceso de programación y reducen los posibles errores, ya que brindan código que ya fue probado y verificado por otros desarrolladores. También te permiten enfocarte en la parte más creativa e innovadora de tu proyecto, sin tener que preocuparte por reinventar cada componente básico desde cero. Además, se actualizan constantemente, lo cual te permite aprovechar mejoras y parches de seguridad.

Tipos de librerías

Las librerías se pueden clasificar en diferentes tipos según el lenguaje de programación y su propósito. Hay librerías específicas para el desarrollo web, para trabajar con datos, para videojuegos, entre muchas otras. Conocer las opciones te permite decidir cuáles se adaptan mejor a lo que necesitas.

Algunos de los tipos más comunes de librerías son:

Librerías gráficas: Permiten generar gráficos y elementos visuales en aplicaciones. Ejemplos de estas son matplotlib en Python o OpenGL en C++. Estas librerías son útiles para crear gráficos estadísticos, simulaciones en 3D, o incluso visualizaciones de datos que mejoren la experiencia del usuario.

Librerías matemáticas: Proporcionan funciones avanzadas para realizar cálculos complejos, como NumPy en Python o Math.h en C. Cuando necesitas realizar cálculos científicos, análisis de datos, o simplemente resolver ecuaciones matemáticas complicadas, estas librerías se vuelven esenciales.

Librerías de redes: Te permiten realizar conexiones y gestionar la comunicación entre sistemas a través de protocolos como HTTP, TCP/IP, entre otros. Un ejemplo es Requests en Python. Si estás desarrollando una aplicación que necesita enviar o recibir información desde otro servidor, las librerías de redes son fundamentales para hacer todo el trabajo detrás de la conexión.

Librerías de programación en C

El lenguaje C, uno de los más antiguos y potentes, también cuenta con muchas librerías. Algunas son estándar y vienen integradas al lenguaje, como la librería stdio.h, que se usa para gestionar operaciones de entrada y salida como leer y escribir en la consola.

- Otras librerías populares incluyen:

math.h para funciones matemáticas avanzadas.

string.h para manejar cadenas de caracteres.

time.h, que facilita la gestión de fechas y horarios, permitiendo desarrollar funciones de temporización o medir el rendimiento de ciertos fragmentos de código.

stdlib.h, que ofrece funciones generales como conversión de datos, gestión de memoria dinámica, y generación de números aleatorios.

Estas librerías te ayudarán a simplificar mucho tu código y a mejorar tu eficiencia. Al ser un lenguaje de bajo nivel, el desarrollo en C suele ser más complejo y detallado que en otros lenguajes. Tener librerías ya listas para usar permite evitar errores y escribir código más limpio.

¿En qué lenguajes se pueden usar?

Se puede usar en estos lenguajes de programación JavaScript, R, Python. Y muchos otros. (Onmex, 2025)

1.3. Código fuente

Comenzamos explicando en términos generales qué es el código fuente. Con este término nos referimos al contenido de los archivos de texto que forman parte de un programa informático, escritos en un lenguaje de programación. Dicho de otra manera, es el código que especifica el funcionamiento de las aplicaciones.

Sin embargo, de un modo más amplio, el término de código fuente puede aplicarse también para expresar otros tipos de documentos como archivos HTML, CSS o XML, que no son específicamente lenguajes de programación.

Por tanto, el código fuente es el responsable de definir tantos programas como otros documentos usados en el día a día de la informática, atendiendo a una serie de reglas léxicas y gramaticales especificadas por el propio lenguaje con el que se está trabajando.

¿Cuál es la estructura de un código fuente?

Como decimos, todo código fuente debe ajustarse a unas reglas, además de una estructura bien definida por el lenguaje en el que se escribe. Si nos centramos en los lenguajes de programación encontraremos estos elementos básicos.

Variables

Permiten almacenar datos en espacios de memoria a los que se les asigna un nombre para referirnos a ellos. Ese dato puede variar a lo largo del tiempo de ejecución de un programa, por lo que le llamamos variable.

Operadores

Permiten realizar operaciones relacionando una o varias variables, para efectuar cálculos o transformaciones de los datos introducidos por el usuario.

Sentencias

Son agrupaciones de uno o más operadores sobre una o más variables, formando instrucciones que el lenguaje de programación debe de procesar para llevar a cabo los objetivos de los programas.

Comparaciones y condicionales

Son las sentencias que nos permiten tomar decisiones en los lenguajes de programación, para hacer unas cosas u otras en función de las condiciones que se dan en los programas.

Bucles

Nos permiten repetir una o varias sentencias un número determinado o indeterminado de veces, generalmente mientras se cumpla una condición.

Comentarios

Son mensajes que los programadores pueden dejar en el código para especificar el funcionamiento del mismo, las decisiones que han tomado para realizar cierto algoritmo, la

función que cumple el código, etc. Los comentarios no se procesan por el lenguaje, por lo que pueden escribirse en el lenguaje escrito entendible por las personas. (Garcia de Zuñiga, 2024)

1.4.Desarrollo de programas

Es el proceso completo de crear un software, desde la planificación, escritura de código, pruebas y documentación.

El desarrollo de programas es el proceso de crear aplicaciones informáticas, programas y sistemas. Esto se hace mediante la planificación, diseño, implementación, prueba y mantenimiento de software.

El desarrollo de programas es un proceso iterativo y colaborativo que involucra a varios profesionales.

Las etapas del desarrollo de programas son:

Planificación, Análisis, Diseño, Programación, Pruebas, Implementación, Mantenimiento, Documentación.

El software es el conjunto de instrucciones o programas que indican a una computadora lo que debe hacer. Los lenguajes de programación son las herramientas que se utilizan para crear programas. Algunos ejemplos de lenguajes de programación son C, Java, Python y JavaScript.

El desarrollo de software permite expandir y adaptar las aplicaciones según la demanda y el crecimiento del negocio.

(StarkCloud, 2024)

1.5. Documentación de programas

La documentación de software se define como la información enfocada en la descripción del sistema o producto para quienes se encargan de desarrollarlo, implementarlo y utilizarlo. Por lo que también puedes determinar como una documentación de procesos.

Además, esta documentación software incorpora los aspectos de los manuales, como las funciones de ayuda, sus versiones en línea y demás.

Es importante destacar también que la documentación de software o documentación de procesos puede indicar los datos relacionados con las notas de sesión y comentarios de código del sistema.

Asimismo, este elemento permite la comprensión del producto de software, su interfaz, sus capacidades, propiedades principales y demás. Además, la documentación de software será clave para facilitar encontrar de manera rápida una determina sección.

Características de la documentación de software

Dentro de las principales características de la documentación de software o documentación de procesos, se encuentra que se utiliza con el fin de describir el producto tecnológico y los procesos que se implementaron para su creación, todo ello escrito de manera formal.

Otra de las características de la documentación de software es que esta permite registrar el proceso de desarrollo de software del sistema, lo que también contribuye en labores como la solución de problemas.

De manera que la documentación de software o la documentación de programas también debe incluir la descripción adecuada para que ayude a los usuarios finales a entender mejor el producto.

De la misma manera, esta documentación debe ser lo suficientemente detallada para servir de ayuda en el hallazgo de la información que se requiera por los clientes y el equipo de soporte del software.

Además de esto, la documentación de software debe actualizarse cada vez que se lleven a cabo modificaciones en el sistema, debido a que tener una documentación desactualizada o

que no corresponda con la versión más reciente es incluso peor que no tenerla, pues puede ocasionar confusiones e inconvenientes.

Cabe destacar sobre la documentación en software también que es importante llegar a un compromiso en los equipos de trabajo para que se incluya la documentación de software en el proceso de desarrollo de software de las nuevas versiones antes de poder liberar la versión.

Tipos de documentación de software

Gracias a las características de este tipo de documentación en software, se puede hablar de una serie de instrucciones acerca de cómo llevar a cabo ciertas labores, así como de la documentación de diseño enfocada en el proceso y en el producto. En esta última, la primera es la relacionada con la información del desarrollo de software o documentación de programas y mantenimiento del sistema, y la segunda, la descripción del producto que se desarrolla dentro de la documentación en software. (Mallón , 2024)

2. Herramientas de desarrollo

Las herramientas de desarrollo de software son programas y aplicaciones diseñadas para facilitar y optimizar el proceso de creación, prueba y mantenimiento de software. Son esenciales para mejorar la productividad de los desarrolladores, asegurar la calidad del código y gestionar eficientemente los proyectos de software.

El desarrollo de software, es una de las ramas de la ingeniería que se en enfoca principalmente a lo que es la creación de sistemas informáticos.

El desarrollo de software, también conocido como el ciclo del software, se compone por diversas etapas que dependen precisamente de qué es lo que se está llevando a cabo, cada una de esas etapas cuenta con distintas Herramientas de Desarrollo de Software y veremos cada una de ellas, para que sin importar en qué fase de desarrollo te encuentres, tengas la posibilidad de usar distintas herramientas de software que te faciliten la vida en gran manera. (OK HOSTING, 2024)

2.1. Generadores de pantallas (Interfaces de usuario)

Un generador de pantallas es una herramienta para crear interfaces gráficas de usuario (GUI) sin escribir mucho código. Permite arrastrar y soltar elementos para diseñar aplicaciones, facilitando el proceso a quienes tienen poca experiencia en diseño.

La interfaz de usuario es el medio por el cual una persona controla una aplicación de software o dispositivo de hardware. Es decir, el programa incluye controles gráficos que optimizan la experiencia de usuario al emplear un mouse o teclado, lo que posibilita la interacción con los procesadores para realizar un trabajo.

Características de la interfaz de usuario

El balance de estos elementos resulta en una interfaz que ayuda al usuario a realizar el trabajo para el cual fue diseñada una aplicación o un programa de cómputo.

Atractivo visual: La apariencia debe ser una prioridad de los equipos de desarrollo de interfaz del usuario, ya que permite que el usuario se sienta identificado y cómodo con el programa.

Claridad: Es la manera de transmitir la información al usuario debe ser clara y concisa para evitar errores o confusiones al momento de interactuar el software.

Coherencia: Todos los elementos de una aplicación deben mantener unidad en su diseño y propósito. Al conseguirlo, los usuarios pueden crear patrones de uso de forma intuitiva, sin la necesidad de aprender procesos muy complejos que podrían desinhibir su uso de tecnología

Flexibilidad: Este concepto es cada vez más importante para los usuarios: una herramienta que puede adaptarse a las necesidades del usuario favorecerá su éxito en el futuro.

Tipos de interfaz de usuario

Interfaz de lenguaje natural: Una interfaz de usuario consta de dos componentes: el lenguaje de presentación, o sea, la transacción de la computadora hacia la persona, y el lenguaje de acción, que se caracteriza por ser la interacción de la persona con la computadora.

Las interfaces de lenguaje natural son el sueño de los usuarios no especializados, ya que permiten la comunicación entre humanos y máquinas mediante un lenguaje cotidiano o natural. En otras palabras, el usuario no requiere habilidades especiales para controlarla. Un ejemplo es Alexa, que cuenta con un software basado en modelos acústicos y del lenguaje.

Elementos más comunes de lenguaje natural: Procesan el lenguaje humano mediante muestras de texto y voz.

Comprenden de la mejor manera posible las intenciones de los usuarios y simplifican la actividad comercial de las empresas.

Interfaz de preguntas y respuestas: Esta es una de las interfaces más utilizadas por los consumidores: esta les muestra una pregunta en la pantalla y, a través del teclado o haciendo

clic con el mouse, la aplicación les permite responder. Según la respuesta recibida, la computadora actúa de una manera preprogramada.

Los asistentes que sirven para instalar software son un ejemplo común de este tipo de interfaz, el usuario responde las preguntas acerca del proceso de instalación, tales como dónde instalar el software.

Elementos más comunes de preguntas y respuestas: Simplifica procesos, responde a las expectativas de los clientes.

Interfaz gráfica de usuario: Conocida también como GUI (del inglés Graphical User Interface), utiliza imágenes, iconos y menús para mostrar las acciones disponibles en un dispositivo, entre las que un usuario puede escoger.

La interfaz gráfica de usuario de escritorio de Windows o Mac OS son ejemplos populares de GUI, las cuales reemplazaron comandos de texto y código binario por elementos gráficos, hoy utilizados en la mayoría de equipos.

Elementos más comunes de la interfaz de usuario: Controles de entrada: permiten introducir información en el sistema por parte de los usuarios. Componentes de navegación: ayudan a los usuarios a moverse. Componentes informativos: brindan información a los usuarios. Contenedores: mantienen el contenido organizado, como paneles, ventanas, marcos, etc. (Lenis, 2023)

2.2. Generadores de informes

Un generador de informes es un programa que toma datos de una fuente y los utiliza para crear un documento. Los datos pueden provenir de bases de datos, hojas de cálculo o flujos XML.

Estos informes pueden incluir solo texto o gráficos, tablas y otros elementos visuales. De forma totalmente personalizada por el cliente. Puedes representarlo de la forma que quieras, utilizando tantos gráficos o tablas, como necesites.

Algunos generadores de informes son:

Report Builder

Un generador de informes de Microsoft que permite crear informes y conjuntos de datos compartidos.

ReportGenerator

Un generador de informes que convierte informes de cobertura en informes legibles en varios formatos.

TFORMer SDK

Un generador de informes multiplataforma que está disponible para Microsoft Windows, Mac OS X, Linux y UNIX.

Dotnet Report

Un generador de informes ad-hoc para desarrolladores .NET.

Reportei

Una herramienta que permite crear análisis automáticos en base a integraciones de CRM, automatización de marketing, tráfico pagado o redes sociales.

Algunas herramientas de informes son:

- ★ Piktochart
- ★ Holistics
- ★ Microsoft Power BI
- ★ Supermetrics
- ★ Hubspot Marketing Analytics
- ★ Thoughtspot
- ★ Octoboard
- ★ Whatagraph
- ★ Google Data Studio

Como resultado, la información se expone de manera clara, mediante un informe generado y personalizado para utilizar en presentaciones o compartir con otros responsables de tu negocio. Es así como se pueden tomar las mejores decisiones, a partir de la información obtenida y representada de manera adecuada.

Estos programas se utilizan comúnmente en empresas y organizaciones para monitorear y analizar los resultados de las operaciones. No basta con obtener los datos, sino que también es fundamental exponerlos y exportar gráficos conforme esos datos, para ver de manera más clara la información. (TEC-IT, 2025)

2.3. Generadores de consultas

Un generador de consultas SQL es una biblioteca de programación o herramienta visual que simplifica la construcción de consultas SQL. Conoce la estructura de una base de datos y ofrece a los usuarios sugerencias y procedimientos para ayudarlos a escribir la consulta necesaria para lograr su objetivo de extracción o manipulación de datos.

Los desarrolladores utilizan bibliotecas de generadores de consultas para generar consultas SQL programáticamente en su código, lo que les permite acceder a una sintaxis más estructurada y, a menudo, más intuitiva. Por otro lado, los generadores de consultas visuales están diseñados tanto para desarrolladores como para usuarios sin conocimientos técnicos.

Las herramientas de construcción de consultas SQL ofrecen una forma más abstracta de interactuar con las bases de datos. En concreto, permiten crear consultas complejas sin tener que lidiar con las complejidades de los comandos SQL. Generalmente, proporciona métodos, funciones o procedimientos visuales para construir una consulta mediante la integración de componentes básicos, como la selección de columnas, la determinación de condiciones, la unión de tablas, la ordenación de resultados y la agrupación de datos.

Tipos de generadores de consultas SQL

Ahora que sabes qué son los constructores de consultas SQL, debes saber que estas herramientas se pueden clasificar en dos tipos:

- Bibliotecas de creación de consultas
- Constructores de consultas visuales

Ejemplos de generadores de consultas:

SQL Server Management Studio (SSMS): La herramienta de Microsoft para trabajar con SQL Server incluye un generador de consultas visual que permite crear consultas sin escribir código manualmente.

Toad para MySQL/Oracle: Herramientas que permiten generar consultas SQL de manera visual.

MySQL Workbench: Ofrece un generador de consultas SQL visual para bases de datos MySQL.

Ejemplo de uso:

Si tienes una base de datos de empleados y deseas obtener todos los empleados que trabajan en el departamento de "Ventas", un generador de consultas puede permitirte seleccionar la tabla "empleados", elegir los campos "nombre" y "departamento", y aplicar un filtro con "departamento = 'Ventas'" para generar la consulta SQL automáticamente. (Zanini, 2024)

2.4. Generador de aplicaciones

Un generador de aplicaciones es una herramienta que permite crear aplicaciones completas sin escribir todo el código desde cero. Incluye módulos predefinidos para tareas comunes, facilitando el trabajo a desarrolladores y usuarios sin experiencia técnica gracias a interfaces de "arrastrar y soltar".

Ejemplos de generadores de aplicaciones:

AppGyver: Plataforma para crear aplicaciones móviles sin código (no-code), con una interfaz visual.

OutSystems: Un entorno de desarrollo rápido de aplicaciones que permite crear aplicaciones empresariales a gran escala sin necesidad de escribir mucho código.

FileMaker: Herramienta de creación de aplicaciones empresariales que facilita la creación de soluciones personalizadas, como bases de datos, con poco o ningún código.

Ejemplo de uso:

Usando una herramienta como OutSystems, puedes crear una aplicación de gestión de empleados arrastrando módulos predefinidos, como formularios para agregar empleados, listas para ver todos los empleados, y pantallas para editar la información de los empleados, todo sin necesidad de escribir mucho código.

Beneficios:

- ✓ **Ahorro de tiempo:** Permiten desarrollar aplicaciones y sistemas mucho más rápidos, ya que automatizan tareas complejas o repetitivas.
- ✓ **Facilidad de uso:** Muchas de estas herramientas están diseñadas para que personas con pocos conocimientos de programación puedan usarlas, facilitando la creación de aplicaciones e informes.
- ✓ **Reducción de errores:** Al generar automáticamente el código necesario, se reducen los errores humanos que podrían ocurrir al escribirlo manualmente.
- ✓ **Flexibilidad:** Aunque estas herramientas facilitan el trabajo, aún permiten que los desarrolladores agreguen código personalizado cuando sea necesario para adaptar las soluciones a necesidades específicas. (AWS, 2024) - (Euroinnova, 2025)

3. **Generación y desarrollo de aplicaciones con herramientas CASE:**

Las herramientas CASE (Computer-Aided Software Engineering) son software que ayudan en el desarrollo de aplicaciones de software, mejorando el ciclo de vida desde la planificación hasta el mantenimiento. Se explicará la generación y desarrollo de aplicaciones con estas herramientas, destacando características, estructura, generación de código, documentación y más.

Herramienta CASE

Las herramientas CASE son un conjunto de aplicaciones informáticas, utilizadas para automatizar actividades del ciclo de vida de desarrollo de sistemas (SDLC). Las herramientas CASE son usadas por los directores de proyectos de software, analistas e Ingenieros para desarrollar sistemas de software. (tutorialspoint, 2025)

3.1. Características de las Herramientas CASE

Las herramientas CASE tienen características diseñadas para optimizar y hacer más eficiente el proceso de desarrollo de software. Algunas de sus características más comunes incluyen:

Automatización de tareas repetitivas: Ayudan a automatizar muchas de las tareas involucradas en el desarrollo, como la creación de diagramas, la generación de código o la creación de documentación.

Soporte para Modelado: Ofrecen herramientas para diseñar diagramas de flujo, diagramas de casos de uso, diagramas de clases, diagramas de actividad y otros modelos que representan el sistema a desarrollar.

Generación de código: Pueden generar automáticamente código a partir de los modelos desarrollados, reduciendo el tiempo y los errores humanos al escribir código manualmente.

Manejo de la documentación: Ayudan a generar documentación del software y proporcionar informes sobre el desarrollo y el estado del proyecto.

Control de versiones y seguimiento de cambios: Muchas herramientas CASE permiten gestionar versiones del proyecto, lo que facilita la actualización y el mantenimiento de los sistemas.

Características deseables

Una herramienta CASE cliente/servidor provee modelo de datos, generación de código, registro del ciclo de vida de los proyectos , múltiples repositorios de usuarios, comunicación entre distintos ingenieros. Las principales herramientas son KnowledgeWare's Application Development Workbench, TI's Information Engineering Facility (IEF), and Andersen Consulting's Foundation for Cooperative Processing.

Por otra parte, una herramienta CASE Cliente/Servidor debe ofrecer:

- Proporcionar topologías de aplicación flexibles. La herramienta debe proporcionar facilidades de construcción que permita separar la aplicación (en muchos puntos diferentes) entre el cliente, el servidor y más importante, entre servidores.
- Proporcionar aplicaciones portátiles. La herramienta debe generar código para Windows, OS/ 2, Macintosh, Unix y todas las plataformas de servidores conocidas. Debe ser capaz, a tiempo de corrida, desplegar la versión correcta del código en la máquina apropiada.
- Control de Versión. La herramienta debe reconocer las versiones de códigos que se ejecutan en los clientes y servidores, y asegurarse que sean consistentes. También, la herramienta debe ser capaz de controlar un gran número de tipos de objetos incluyendo texto, gráficos, mapas de bits, documentos complejos y objetos únicos, tales como definiciones de pantallas y de informes, archivos de objetos y datos de prueba y

resultados. Debe mantener versiones de objetos con niveles arbitrarios de granularidad; por ejemplo, una única definición de datos o una agrupación de módulos.

- Crear código compilado en el servidor. La herramienta debe ser capaz de compilar automáticamente código 4GL en el servidor para obtener el máximo performance.
- Trabajar con una variedad de administradores de recurso. La herramienta debe adaptarse ella misma a los administradores de recurso que existen en varios servidores de la red; su interacción con los administradores de recurso debería ser negociable a tiempo de ejecución.
- Trabajar con una variedad de software intermedios. La herramienta debe adaptar sus comunicaciones cliente/servidor al software intermedio existente. Como mínimo la herramienta debería ajustar los temporizadores basándose en, si el tráfico se está moviendo en una LAN o WAN.
- Soporte multiusuarios. La herramienta debe permitir que varios diseñadores trabajen en una aplicación simultáneamente. Debe gestionarse los accesos concurrentes a la base de datos por diferentes usuarios, mediante el arbitrio y bloqueos de accesos a nivel de archivo o de registro.
- Seguridad. La herramienta debe proporcionar mecanismos para controlar el acceso y las modificaciones a los que contiene. La herramienta debe, al menos, mantener contraseñas y permisos de acceso en distintos niveles para cada usuario. También debe facilitar la realización automática de copias de seguridad y recuperaciones de las mismas, así como el almacenamiento de grupos de información determinados, por ejemplo, por proyecto o aplicaciones.
- Desarrollo en equipo, repositorio de librerías compartidas. Debe permitir que grupos de programadores trabajen en un proyecto común; debe proveer facilidades de check-in/ check-out registrar formas, widgets, controles, campos, objetos de negocio, DLL, etc;

- Debe proporcionar un mecanismo para compartir las librerías entre distintos realizadores y múltiples herramientas; gestiona y controla el acceso multiusuario a los datos y bloquea los objetos para evitar que se pierdan modificaciones inadvertidamente cuando se realizan simultáneamente. (Lycos Tripod, 2025)

3.2. Estructura de las Herramientas CASE

Las herramientas CASE están divididas en varios módulos, cada uno para una fase del ciclo de vida del software. Incluyen:

- Módulo de análisis y diseño: crea modelos del sistema.
- Módulo de generación de código: genera código fuente automáticamente.
- Módulo de pruebas: gestiona pruebas automatizadas.
- Módulo de documentación: genera documentación técnica.
- Módulo de gestión de proyectos: planifica y sigue actividades del proyecto.

Componentes de una herramienta CASE

De una forma esquemática podemos decir que una herramienta CASE se compone de los siguientes elementos:

- Repositorio (diccionario): donde se almacenan los elementos definidos o creados por la herramienta, y cuya gestión se realiza mediante el apoyo de un Sistema de Gestión de Base de Datos (SGBD) o de un sistema de gestión de ficheros.
- Meta modelo (no siempre visible): que constituye el marco para la definición de las técnicas y metodologías soportadas por la herramienta.
- Carga o descarga de datos: son facilidades que permiten cargar el repertorio de la herramienta CASE con datos provenientes de otros sistemas, o bien generar a partir de la propia herramienta esquemas de base de datos, programas, etc. que pueden, a su vez, alimentar otros sistemas. Este elemento proporciona así un medio de comunicación con otras herramientas.

- Comprobación de errores: facilidades que permiten llevar a cabo un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.
- Interfaz de usuario: que constará de editores de texto y herramientas de diseño gráfico que permitan, mediante la utilización de un sistema de ventanas, iconos y menús, con la ayuda del ratón, definir los diagramas, matrices, etc. que incluyen las distintas metodologías.

Estructura General de una Herramienta CASE

La estructura CASE se basa en la siguiente terminología:

- CASE de alto nivel: son aquellas herramientas que automatizan o apoyan las fases finales o superiores del ciclo de vida del desarrollo de sistemas como la planificación de sistemas, el análisis de sistemas diseño de sistemas.
- CASE de bajo nivel: son aquellas herramientas que automatizan o apoyan las fases finales o inferiores del ciclo de vida como el diseño detallado de sistemas, la implantación de sistemas y el soporte de sistemas.
- CASE cruzado de ciclo de vida: se aplica a aquellas herramientas que apoyan actividades que tienen lugar a lo largo de todo el ciclo de vida, se incluyen actividades como la gestión de proyectos y la estimación.

Integración de las Herramientas CASE en el futuro

Las herramientas CASE evolucionan hacia tres tipos de integración:

1. La integración de datos permite disponer de herramientas CASE con diferentes estructuras de diccionarios locales para el intercambio de datos.
2. La integración de presentación confiere a todas las herramientas CASE el mismo aspecto.
3. La integración de herramientas permite disponer de herramientas CASE capaces de invocar a otras CASE de forma automática. (Herramientas Case , 2025)

3.3. Actualización y Mantenimiento del Proyecto

Las herramientas CASE permiten actualizar y mantener el software a lo largo de su ciclo de vida. Algunas de las funcionalidades relacionadas con la actualización incluyen:

Refactorización automática: Permite actualizar y mejorar el código de manera automática sin alterar su funcionalidad.

Control de versiones: Las herramientas CASE suelen incluir un sistema para gestionar las versiones del software y realizar un seguimiento de los cambios a lo largo del tiempo.

Mantenimiento de modelos: Si se realizan cambios en los requisitos o el diseño, las herramientas CASE permiten actualizar los diagramas y modelos, lo que actualiza automáticamente la documentación y el código correspondiente.

¿Cuáles son los 4 tipos de mantenimiento de software?

Cada uno de los cuatro tipos diferentes de mantenimiento de software se realiza por diferentes razones y propósitos. Es posible que una determinada pieza de software deba someterse a uno, dos o todos los tipos de mantenimiento a lo largo de su vida útil.

Los cuatro tipos son:

- Mantenimiento correctivo de software
- Mantenimiento preventivo de software
- Mantenimiento perfectivo de software
- Mantenimiento adaptable de software
- Mantenimiento correctivo de software

El mantenimiento correctivo del software es la forma clásica y típica de mantenimiento (para el software y cualquier otra cosa). El mantenimiento de software correctivo es necesario cuando algo sale mal en una pieza de software, incluidos fallos y errores. Estos pueden tener un impacto generalizado en la funcionalidad del software en general y, por lo tanto, deben abordarse lo antes posible.

Muchas veces, los proveedores de software pueden abordar problemas que requieren mantenimiento correctivo debido a los informes de errores que envían los usuarios. Si una empresa puede reconocer y solucionar las fallas antes de que los usuarios las descubran, esta es una ventaja adicional que hará que su empresa parezca más respetable y confiable (después de todo, a nadie le gusta un mensaje de error).

Mantenimiento preventivo de software

El mantenimiento preventivo de software está mirando hacia el futuro para que su software pueda seguir funcionando como se desee durante el mayor tiempo posible.

Esto incluye realizar los cambios necesarios, actualizaciones, adaptaciones y más. El mantenimiento preventivo del software puede abordar pequeños problemas que en un momento dado pueden carecer de importancia, pero pueden convertirse en problemas mayores en el futuro. Estos se denominan fallas latentes que deben detectarse y corregirse para asegurarse de que no se conviertan en fallas efectivas.

Mantenimiento perfectivo de software

Al igual que con cualquier producto en el mercado, una vez que el software se lanza al público, surgen nuevos problemas e ideas. Los usuarios pueden ver la necesidad de nuevas características o requisitos que les gustaría ver en el software para convertirlo en la mejor herramienta disponible para sus necesidades. Es entonces cuando entra en juego el mantenimiento perfectivo del software.

El mantenimiento perfectivo de software tiene como objetivo ajustar el software agregando nuevas características según sea necesario y eliminando características que son irrelevantes o no efectivas en el software dado. Este proceso mantiene el software relevante a medida que el mercado y las necesidades del usuario cambian.

Mantenimiento adaptativo de software

El Mantenimiento adaptativo de software tiene que ver con las tecnologías cambiantes, así como con las políticas y reglas relacionadas con su software. Las cuales incluyen cambios en el sistema operativo, almacenamiento en la nube, hardware, etc. Cuando se realizan estos cambios, su software debe adaptarse para cumplir adecuadamente los nuevos requisitos y continuar funcionando bien. (Thales, 2025)

3.4. Generación de Código

La generación de código es una de las funcionalidades clave de las herramientas CASE. Estas herramientas permiten generar código de forma automática a partir de los modelos creados, lo que ayuda a reducir la cantidad de código repetitivo que debe escribirse manualmente. Esto también ayuda a garantizar que el código sea consistente y esté alineado con los requisitos del sistema.

Generación desde diagramas: Por ejemplo, si se crea un diagrama de clases, la herramienta CASE puede generar las clases correspondientes en el lenguaje de programación deseado (por ejemplo, C++, Java, Python).

Generación de código de bases de datos: También pueden generar código para la creación de bases de datos y la manipulación de datos (por ejemplo, sentencias SQL).

Herramientas CASE

La ingeniería del software asistida por computadora, conocida normalmente como CASE (Computer Aided Software Engineering), es un conjunto de herramientas que permiten la automatización del proceso de producción de desarrollo de software.

Bibliografía sugerida

El objetivo principal de una aplicación CASE es proporcionar un conjunto de herramientas bien integradas que ahorren trabajo, enlazando y automatizando todas las fases del ciclo de ciclo del software.

Existe una gran variedad de herramientas CASE en el mercado, cada una con una orientación diferente y aplicable a una o más fases del ciclo de vida del desarrollo de software. Algunas sólo apoyan al ingeniero o analista en el diseño de diagramas para modelizar procesos o flujos de datos; otras ofrecen prestaciones incluso para generar código o bases de datos.

La mayoría de los diagramas y documentos que generan estas herramientas se utilizan como documentación en el diseño y desarrollo de las soluciones informáticas. Algunos de los modelos que permiten generar dichas herramientas son los siguientes:

- Diagramas genéricos. Este tipo de diagramas se utiliza para crear diseños y esquemas para brainstormings, planificación, comunicación, generar la documentación de apoyo a un proyecto informático, etc.
- Diagramas de auditoría. Este tipo de diagramas se utilizan para documentar y analizar procesos que incluyen transacciones financieras y gestión de inventarios.
- Diagramas de modelos conceptuales de bases de datos. Existen diferentes notaciones para modelizar "conceptualmente" bases de datos. La mayoría de las herramientas ofrecen prestaciones para diseñar estos modelos y generar a partir de éstos los modelos relacionales de las bases de datos. Algunas van incluso más allá y generan la base de datos para diferentes sistemas gestores de bases de datos. Algunas de estas notaciones son la notación Bachman, Chen ERD, etc.
- Diagramas de modelos orientados a objetos y notación UML. Siguiendo diferentes notaciones (Booch OOD, los casos de uso de Jacobson, la notación ERD Martin, el lenguaje de modelado ROOM o la metodología OMT de Rumbaugh) este tipo de diagramas permiten diseñar aplicaciones orientadas a objetos. Al igual que en el caso de los diagramas de bases de datos, la mayoría de las herramientas CASE disponen de funciones para generar código de programación a partir de los diseños.

- Diagramas conceptuales de web sites. Utilizados en el diseño y desarrollo de webs, este tipo de diagramas permite esquematizar la organización en páginas de una web, la navegación entre ellas, su contenido, etc.
- Diagramas de procesos y flujos de datos. Permiten diseñar y reflejar los procesos de negocio y los flujos de datos y documentos de los diferentes departamentos de una compañía y la interacción entre éstos. Las herramientas CASE avanzadas disponen de funcionalidades para simular el comportamiento de dichos procesos y prever situaciones futuras modificando los diferentes parámetros de los procesos. Un método de generación de modelos de flujos de datos es el de Gane-Sarson.
- Diagramas GANTT y PERT. Se utilizan para planificar proyectos y procesos. Existen herramientas específicas, los sistemas gestores de proyectos, para tratar este tipo de información.
- Diagramas de redes. Este tipo de diagramas permite crear y diseñar el diagrama tanto lógico como físico de una red de ordenadores y dispositivos. Una vez diseñados los citados diagramas, algunas aplicaciones permiten guardar esta información en bases de datos y/o realizar simulaciones de tráfico por la red, etc.
- Diagramas de interfaz de usuario, formularios e informes. La mayoría de los programas y aplicaciones disponen de una interfaz gráfica para el usuario. Cuando trabajamos con un programa introducimos datos, leemos información, seleccionamos opciones de menús... Durante la fase de diseño de un software, se utiliza este tipo de diagramas para definir cómo serán sus diferentes interfaces. Muchas herramientas CASE ofrecen funcionalidades para diseñar formularios de entrada de datos e informes, y más adelante podrán generar esos formularios e informes en código de programación de diferentes lenguajes. (moduls, 2025)

3.5. Documentación generada

Las herramientas CASE también son esenciales para la creación y mantenimiento de la documentación de software. Pueden generar documentación técnica y de usuario de forma automática y actualizada en función del estado del proyecto.

Documentación técnica: A partir de los diagramas y el código generado, las herramientas CASE pueden crear documentación sobre el diseño, las clases, los métodos, las relaciones entre componentes y las decisiones arquitectónicas.

Manuales de usuario: Las herramientas CASE también pueden generar manuales de usuario para explicar cómo interactuar con el software, con instrucciones sobre cómo usar las funcionalidades.

Manuales de programador: Además de los manuales de usuario, pueden generar documentación dirigida a los programadores, que describe la estructura del código, las clases y métodos, y cómo se puede modificar o extender el software.

Herramienta CASE de generación de documentación mediante ingeniería inversa

Resumen

Para el desarrollo de este proyecto se han de aplicar los conocimientos tanto metodológicos como tecnológicos adquiridos durante el transcurso del Máster en Ingeniería Web. Para ello se propone USTUML, una aplicación web que permite la generación de diagramas UML tanto por ingeniería directa como por ingeniería inversa. La ingeniería directa se ha realizado mediante técnica declarativa y la ingeniería inversa a partir de código fuente en java, alojado en GitHub. Respecto a metodologías, al principio se aplicó RUP (Rational Unified Process) y en el momento en el cual se llegó a comprender cómo implementar el proyecto se comenzó a desarrollar de manera ágil. Las tecnologías usadas han sido el framework Spring para el desarrollo de la capa de negocio, el framework Angular para el desarrollo de la capa de presentación y MongoDb para la capa de datos. Como ecosistema se utiliza GitActions para la

integración continua desde el cual se llama a SonarCloud para realizar el análisis de calidad de código y realiza el despliegue continuo en Heroku, la capa de negocio, y en Firebase, la capa de presentación. Abstract: For the development of this project, the methodological and technological knowledge acquired during the course of the Master's Degree in Web Engineering has to be applied. For this purpose, USTUML is proposed, a web application that allows the generation of UML diagrams by both direct and reverse engineering. Direct engineering has been carried out using declarative techniques and reverse engineering from java source code, hosted on GitHub. Regarding methodologies, RUP (Rational Unified Process) was applied at the beginning and when it was possible to understand how to implement the project, agile development began. The technologies used were the Spring framework for the development of the business layer, the Angular framework for the development of the presentation layer and MongoDb for the data layer. As an ecosystem, GitActions is used for the continuous integration from which SonarCloud is called to perform the code quality analysis and performs the continuous deployment in Heroku, the business layer, and in Firebase, the presentation layer. (Muñoz, Héctor, Guijarro , & Pablo, 2025)

3.6. Tutoriales y Reportes de Desarrollo

Las herramientas CASE suelen incluir tutoriales integrados e informes de desarrollo que ayudan a los desarrolladores a entender cómo usar la herramienta ya seguir el progreso del proyecto. Algunos ejemplos hijo:

Tutoriales interactivos: Proporcionan guías paso a paso para aprender a usar la herramienta, lo que puede ser útil para los nuevos usuarios.

Reportes de desarrollo: Incluyen información sobre el estado del proyecto, los componentes desarrollados, las tareas completadas, las pruebas realizadas, etc. Estos informes ayudan en la gestión del proyecto y permiten a los desarrolladores y líderes de equipo mantenerse al tanto de los avances.

Componentes de las Herramientas CASE

Las herramientas CASE se pueden dividir en las siguientes partes en base a su uso en una etapa concreta del SDLC:

- **Depósito central** - Las herramientas CASE requieren un depósito central, el cual nos puede servir como fuente de común, consistente e información integrada. El depósito central, es un lugar central de almacenamiento, donde los requisitos del producto, los documentos requeridos, los informes y diagramas relacionados, y otra información útil sobre la gestión se almacena. El Depósito central también sirve como Diccionario de datos.
- Herramientas Upper CASE - Las Herramientas Upper CASE se usan en las etapas de planificación, análisis y diseño del SDLC.
- Herramientas Lower CASE - Las Herramientas Lower CASE se usan en la implementación, las pruebas y en el mantenimiento.
- Herramientas Integrated CASE - Las Herramientas Integrated CASE son de utilidad en todas las fases del SDLC, des de la educación de requisitos y las pruebas hasta la documentación.

Tipos de herramientas CASE

Herramienta CASE Diagrama

Estas herramientas se usan para representar componentes del sistema, datos, y controlar la fluidez de varios componentes y estructura del software de manera gráfica. Por ejemplo, la herranmienta 'Flow Chart Maker' para crear los más novedosos Diagramas de flujos.

Herramientas para modelado de procesos.

El modelado de procesos es un método para crear modelos de proceso de software y se usa para desarrollar el software. Las herramientas para el modelado de procesos ayudan a los Directores a elegir un modelo de proceso o para modificarlo según los requisitos del producto software. Por ejemplo, el 'EPF Composer'

Herramientas de administración de procesos.

Estas herramientas se usan para la planificación del proyecto, el costo y esfuerzo estimados, la temporalización y la organización de los recursos. Los Directivos deben coordinar de manera muy estricta la ejecución del proyecto con cada uno de los pasos mencionados con anterioridad para la buena gestión del proyecto software. Herramientas de administración de procesos ayudan a almacenar y compartir información sobre el proyecto en tiempo real durante su organización. Ejemplos de este tipo de herramienta son 'Creative Pro Office', 'Trac Project' o 'Basecamp'.

Herramientas de documentación

La documentación de un proyecto de software empieza antes que el proceso de software, pasa por todas las fases del SDLC y se concluye con la terminación del proyecto.

Las Herramientas de documentación generan documentos tanto para el consumidor final como para consumidores de soporte técnico. Estos últimos son en su mayoría profesionales internos del equipo de desarrollo que consultan manuales de sistemas, manuales de referencia, manuales de formación, de instalación, etc. El consumidor final describe el funcionamiento e instrucciones del sistema como por ejemplo el manual para el usuario. Ejemplos de este tipo de herramientas son: Doxygen, DrExplain, Adobe RoboHelp para documentación.

Herramientas de análisis

Estas herramientas ayudan a cumplir con los requisitos, de manera automática examinan si hay alguna inconsistencia, o información no curada en los diagramas, buscan posibles redundancias u omisiones incorrectas. Ejemplos de este tipo de herramienta son Accept 360, Accompa, CaseComplete para análisis de requisitos y Visible Analysts para análisis total. (tutorialspoint, 2025)

3.7. Ventajas de Usar Herramientas CASE

Aumento de la productividad: La automatización de tareas repetitivas como la generación de código y la documentación reducen significativamente el tiempo de desarrollo.

Mejora de la calidad: Al generar código automáticamente desde modelos bien definidos, las herramientas CASE reducen la probabilidad de errores humanos en la codificación.

Facilita la colaboración: Proporcionan una representación visual de la arquitectura y los componentes del sistema, lo que facilita la colaboración entre los miembros del equipo de desarrollo.

Mejora la gestión del proyecto: Al integrar la planificación, el diseño y la gestión de versiones en una sola herramienta, los equipos pueden llevar un control más efectivo del proyecto.

Ejemplos de herramientas CASE:

Algunas de las herramientas CASE más utilizadas en el desarrollo de software incluyen:

IBM Rational Rose: una herramienta de modelado que permite crear diagramas UML y generar código.

Enterprise Architect: Utilizada para el modelado y diseño de sistemas, incluye funciones de generación de código, pruebas y documentación.

Visual Paradigm: Ofrece soporte para el modelado UML, la generación de código y la creación de diagramas de bases de datos.

PowerDesigner: Enfocada en el diseño de bases de datos y la ingeniería de software.

Entre las más significativos de las herramientas CASE se enumeran los siguientes:

Ventajas

1. Facilidad para la revisión de aplicaciones

   La experiencia muestra que una vez que las aplicaciones se implementan, se emplean por mucho tiempo. Las herramientas CASE proporcionan un beneficio substancial para las organizaciones al facilitar la revisión de las aplicaciones. Contar con un depósito central agiliza el proceso de revisión ya que éste proporciona bases para las definiciones y estándares para los datos. Las capacidades de generación interna, si se encuentran presentes, contribuyen a modificar el sistema por medio de las especificaciones más que por los ajustes al código fuente.

2. Soporte para el desarrollo de prototipos de sistemas

   Se suelen desarrollar diseños para pantallas y reportes con la finalidad de mostrar la organización y composición de los datos, encabezados y mensajes. Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interface. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo. Como disyuntiva, el desarrollo de prototipos puede producir un sistema que funcione. Las características de entrada y salida son desarrolladas junto con el código orientado hacia los procedimientos y archivos de datos.

3. Generación de código

   La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación del código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el

mantenimiento) y disminuye la ocurrencia de varios tipos de errores, mejorando de esta manera la calidad. Las características de la generación del código permiten volver a utilizar el software y las estructuras estándares para generar dicho código, así como el cambio de una especificación modular, lo que significa volver a generar el código y los enlaces con otros módulos.

4. Mejora en la habilidad para satisfacer los requerimientos del usuario

Es bien conocida la importancia de satisfacer los requerimientos del usuario, ya que esto guarda relación con el éxito del sistema. De manera similar, tener los requerimientos correctos mejora la calidad de las prácticas de desarrollo. Las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios. Las herramientas afectan la naturaleza y cantidad de interacción entre los encargados del desarrollo y el usuario. Las descripciones gráficas y los diagramas, así como los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo.

5. Soporte interactivo para el proceso de desarrollo

La experiencia ha demostrado que el desarrollo de sistemas es un proceso interactivo. Las herramientas CASE soportan pasos interactivos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Como resultado de esto, se anticipa que los analistas repasarán y revisarán los detalles del sistema con mayor frecuencia y en forma más consistente. (Wikiversidad, 2023)