Tema 1: Entornos de desarrollo

1.- Introdución a los entornos de desarrollo.

Los entornos de desarrollo son un tipo de software que nos permite desarrollar aplicaciones facilitando el labor desarrollar aplicaciones facilitando la labor a través de herramientas y tareas predefinidas. Con el tiempo algunos han quedado com mas generales, pudiendo abarcar diversos lenguajes y tipos de aplicaciones, mientras que otros se han vuelto más específicos.

La utilización de un IDE nos permite desarrollar nuestro proyecto software de una forma mucho más ágil. Además, podemos utilizar las herramientras que incluya nuestro entorno de trabajo, como el depurador de código o el autoformateo del mismo.

Los entornos de desarrollo pueden clasificarse en base a numerosos factores. Por ejemplo podríamos clasificarlos en base al tipo de liciencia, por ejemplo software libre o software privado, en función de la plataforma para la que esté disponible, si estamos ante un IDE multiplataforma o no o en función del lenguaje de programación que soporte, puesto que hay entornos de desarrollo para un lenguaje especifico, como PHP Storm, o para múltiples lenguajes de programación, como Sublime text. En esta unidad vamos a clasificar los distintos entornos de desarrollo de aplicaciones web.

En la siguiente tabla podemos ver los distintos entornos de desarrollo que vamos a estudiar a lo largo de la unidad, en base a la clasificación antriormente citada.

2.- Entornos integrados de desarrollo comunes

2.1.-Netbeans

Netbeans es un entorno de desarrollo integrado libre desarrollado para lenguaje de programación Java, aunque extisten módulos que permiten desarrollar en otros lenguajes de programación como PHP.

Netbeans fue desarrollado por Sun MicroSystems y tras la adquisicion de esta empresa por Oracle Corporation, pasó a ser propiedad de Oracle que es la empresa encargada de su mantenimiento.

Netbeans IDE que fue desarrollado para todos los tipos de JAVA y despues para generación de esquemas XML, UML y soporte para otros lenguajes de programacion como PHP o C++.

3.Entornos de desarrollo multiplataforma

Los entornos de desarrollo multiplataforma son aquellos conocidos por ser específicos para aplicaciones de escritorio o de dispositivos móviles. Actualmente se estan diferenciando de los entornos para web y mixtos, debido tambien a la evolución de las distintas lineas de mercado, que están llevando a los programadores y desarrolladores de software a ser mas especializados y por tanto, requerir herramientas con este fin.

3.1.-Android Studio

Existen en el mercado diversos entornos orientado a la diversidad de aplicaciones que se pueden realizar (web,apps, etc…).

En este punto vamos a introducir algunos de los mas importantes.

Android Studio es el IDE desarrollo por Google para la implementación de aplicaciones para el SO Android.

Algunas de las carracteristicas mas importantes son:

* Editor de codigo inteligente.
* Plantillas de codigo e integración GitHub
* Desarrollo de aplicaciones multidispositivo (tablet, móvil, dispositivo wearables, etc…).
* Dispositivos virtuales para todos los tamaños y orientaciones.
* Sistema de generación de apk basado en Gradle

Como requisitos mínimos del sistema para su uso:

* 2GB de ram mínimo, 4GB de RAM recomendado
* 400MB de espacio en disco duro
* Al menos 1 GB para Android SDK, imágenes del sistema emulador, y cachés
* 1280 x 800 resolución de pantalla mínima
* Java Development kit (JDK) 7

3.2.- XCode

Xcode es un entorno de desarrollo integrado para macOs que contiene las herramientas necesarias para desarrollar aplicaciones para las distintas plataformas de Apple. Con XCode podemos desarrollar aplicaciones para MacOs, iOs, watchOs y tvOs

Xcode puede descargarse de forma gratuita desde la App Store, para ello tenemos que utilizar un sistema operativo macOs.

3.3.- Otros entornos de desarrollo Multiplataforma.

Android, sistena operativo basado en el kernel de linux, se ha convertido en el sistema operativo dominante de los dispositivo móviles. Actualmente copa cerca del 88% del mercado de dispositivos móviles, lo que ha hecho que se haya convertido en todo un referente. Desde que Android Studio se convirtió en el IDE oficial de Androidm muchos otros entornos de desarrollo que eran compatibles con Android han perdido cuota de mercado.

Por este motivo, Android Studio es el IDE por excelencia en el desarrollo de aplcicaicones multiplataforma.

A pesar de que Android domine el mercado de los dispositivos móviles, estudios recientes concluyen que desarrollar aplicaciones par iOS es mucho más rentable. Por este motivo en el epigrafe anterior hicimos una pequeña mención de XCode, el IDE para desarrillar aplicaciones para las distintos dispositivos de Apple.

Una vez hayamos visto los dos principales entornos de desarrollo para multiplataforma, vamos a hacer una breve descripción de deos framework destinados al desarrollo de videojuegos que contienen su propio IDE.

3.3.1.- Unity

Unity es una herramienta que nos permite desarrollar videojuegos para diversas plataformas. Unity cuenta cins us propio editor integrado que, junto con las librerías y API que proporciona el framework de Unitye, nos permite desarrollar juegos par dispositivos móviles, videoconsilas y diversos sistemas informáticos.

3.3.2.- JMonkey Engine

JMonkey es un motor para el desarrollo de juegos 3D basado en java. Es compatible con LWJGL y la potencialidad unida a la comunidad de desarrolladores que lo sostienen lo hacen una opción muy interesante.

Además del motor de juegos, JMonkey Engine contiene su propio IDE para trabajar con JMonkeyEngine.

4.- Entornos de desarrollo web.

La programación y el desarrollo web en sus inicios, se basaba en un bloc de notas que a partir de código básado en etiquetas, generaba un archivo interpretado que los navegadores podían entender. La cantidad de lenguajes web actuales, así como la implantación total de la web hace necesario el apoyo en herramientas que permitan facilitar el desarrollo.

4.1.-Notepad++

Notepad++ en uno de los editores de código más conocidos en la actualidad. Distribuido bajo licencia GNU, estamos ante un editor de texto que permite trabajar con una gran cantidad de lenguajes. Su instalación es muy sencilla y liviana.

4.2.-Bracket.io

Bracekts es un editor de texto ligero, potente y moderno que nos permite escribir código en una gran variedad de lenguajes de programación. Estamos ante un proyecto open-source (código abierto) distribuido bajo la licencia MIT y que ha sido desarrollado en HTML, CSS y JavaScript.

Aunque estamos ante un proyecto open-source, fue creado en sus orígenes por Adobe System ® y actualmente está disponible para Mac, Windows y GNU/Linux.

4.3.- Sublime text

Sublime text está considerado por muchos desarrolladores web como el mejor editor de código. Estamos ante un IDE de pago, con un coste por licencia de unos 70 dólares, aunque existe la posibilidad de descargar una versión de prueba de forma gratuita desde la web oficial del proyecto.

4.4.- Otros IDEs

4.4.1.- Dreamweaver

Dreamweaver es un editor de páginas web desarrollado originalmente por Macromedia y que tras la adquisición de esta empresa por Adobe, pasó a formar parte de la Suite Creative de Adobe.

La caracteristica de este editor es que no necesita saber ningún lenguaje de programación.

4.4.2.- Atom.

Es un edito gratuito de código abierto lanzado desde la plataforma de github.

Es compatible con linux, windows y macOs, permite la integración de node.js, el framework de javascript de google chome.

4.4.3.- Editra.

Editra es un editor de texto multiplataforma distribuido bajo Licencia LGPL y que cuenta con una liviana interfaz, un buscador y un asistente para la programación en código lo cual facilita enormemente el proceso de desarrollo.

Permite aumentar su funcionalidad mediante la instalación de plugins adicionales.

4.4.4.- Aptana Studio

Aptana Studio es un entorno de desarrollo integrado de software libre basado en eclipse y que funciona bajo Windows, MacOs y GNU/Linux. Soporta una amplia variedad de lenguajes de programación como PHP, Python, Ruby, CSS, HTML y JavaScript.

Incluye soporte para la mayoría de librerías de JavaScript, como jQuery, Ext JS, DOJO y YUI entre otras. Al estar basado en Eclipse es compatible con una amplia variedad de extensiones.

4.5.5.- PHP Storm.

PHP Storm es un IDE comercial para PHP. Soporta la mayoría de framework basados en PHP, como Symfony, Drupal, Wordpress, Zend framework, Yii y CakePHP entre otros.

Actualmente es uno de los IDE más completos para trabajar con PHP.

5.- Herramientas CASE para el desarrollo, prueba y documentación del código.

Las herramientas CASE (Computer Aided Software Engineering) son aplicaciones software que nos facilitan el desarrollo del software, reduciendo el tiempo y el coste del mismo.

5.1.-Herramientas de los navegadores web.

A través de dichas herramientas podemos trabajar con nuestros desarrollos web de forma dinámica, permitiéndonos habilitar y deshabilitar distintos lenguajes utilizados, como JavaScript y CSS, modificar la hoja de estilos en tiempo real e incluso modificar líneas de código HTML de forma dinámica, para ver como quedaría nuestra web con dichas modificaciones y con la ventaja de que los cambios serán deshechos al recargar la página.

5.1.1.- Safari.

Safari es el navegador web desarrollado por Apple. Dispone de un amplio conjunto de herramientas para desarrolladores web. Desde la pestaña Desarrolladores, se puede acceder a un amplio abanico de herramientas para el desarrollo web.

Podremos analizar todo el código HTML y CSS de cualquier página web, además podremos realizar modificaciones sobre el mismo en tiempo real, modificaciones que solo estarán visibles en nuestro navegador y de forma temporal, hasta que recarguemos la página.

5.1.2.- Microsoft Edge.

Microsoft Edge es el nuevo navegador web de Microsoft Windows. Disponible a partir de la versión 10 de Windows. Para acceder a las herramientas de desarrollo basta con pulsar la tecla F12 y se abrirá el menú de herramientas para desarrolladores.

5.1.3.- Mozilla Firefox.

Originalmente se utilizaba como herramienta para inspeccionar el código fuente en HTML y CSS de la web la extensión Firebug.

5.1.4.- Firebug.

Firebug es un complemento para el navegador web FIrefox que integra diversas herramientas de desarrollo web. Permite la edición, depuración y monitorización de los lenguajes CSS, HTML y JavaScript en cualquier página web y en tiempo real.

Para instalar la extensión es tan sencillo como acceder aquí y hacer clic en el botón de "Agregar a Firefox".

* Características fundamentales:
  + Muestra de forma jerarquica el arbol de etiquetas (lo que lo hace mas accesible).
  + Pestaña DOM: Permite recorrer el Documento de Modelado de Objetos de la web en ejecución asi como modificar parámetros.
  + Pestaña Consola: Permite la ejecución de comandos en tiempo real.
  + Pestaña CSS. Para la edición y modificación de los archivos CSS.
  + Pestaña Script: Para visualizar y modificar los documentos o métodos JavaScript cargados en la ejecución de la web.
  + Pestaña Net: Permite la visualización de todos y cada uno de los recursos de la página.

5.1.5- Inspector de Chrome.

Chrome es un navegador que provee de diversas herramientas para la edición y depuración de la web en tiempo real. Esto quiere decir que podemos incluir modificaciones y visualizarlas o ejecutarlas en tiempo real.

5.3.- Software para generar documentación del código fuente.

Uno de los principales objetivos que debe de satisfacer el software desarrollado es su reutilización. Uno de los axiomas principales radica en no reinventar la rueda. Para ello es fundamental reutilizar el software que ya hemos desarrollado, o bien nosotros u otros programadores, siempre y cuando su licencia lo permita.

Existen herramientas software que facilitan el proceso de generar la documentación a partir de nuestro código fuente. En este epígrafe vamos a centrarnos en JavaDoc, que permite generar la documentación del código fuente escrito en Java y está disponible en NetBeans. Decir que existen programas orientados a otros lenguajes de programación como PHPDoc.

JavaDoc

JavaDoc nos permite obtener la documentación a partir de nuestro código fuente. Para ello en el código tenemos que insertar una serie de etiquetas, las cuales vamos a ver a continuación:

* @author - El nombre del autor del proyecto por ejemplo Javier Mejías
* @version - La versión del proyecto, por ejemplo 2.3
* @see - Añade una referencia a una clase, método o enlace web
* @param - Nombre de parámetro utilizado en una función indicando para que sirve.
* @return - El resultado devuelto por la función.
* @exception - Excepciones que lanza nuestro función o clase.
* @throws - Nombre de la excepción junto con su descripción.
* @deprecated - Añade una alerta al usuario de que el método que sigue a continuación ya no debe usarse y que será eliminado en versiones posteriores.

Anexo III.- CMS Gestores de Contenidos.

La web ha experimentado cambios vertiginosos, dando lugar a los gestores de contenidos o CMS que permiten desarrollar una aplicación web o una web típica con pocos conocimientos de programación. Especialmente implantados como soluciones para tiendas online, conocerlos es básico de cara al mercado actual. Un plantea, una flecha, dinero y un ordenador todo enlazado. Son dibujos.

Existen gestores de contenido tanto instalables en servidores propios como en alojamientos de terceros, y tanto libres como de pago recurrente.

Vamos a introducirnos en los tres mas famosos, si bien existen otras opciones como Shopify o Virtue-Mart entre otros.

A.III.1.- Prestashop.

PrestaShop es un software gratuito y de código abierto dedicado a la creación de comercios electrónicos online.

Prestashop integra más de 310 funcionalidades que facilitan el comercio electrónico y está en continuo crecimiento. Todas las funcionalidades son 100% gratuitas y pueden ser instaladas y desinstaladas con un sólo clic.

A.III.2.- Magento.

Magento es un gestor de contenidos web opensource para comercio electrónico. Es una solución flexible y escalable con la que se pueden desarrollar prácticamente todo tipo de proyectos e-commerce.

Ofrece muchas funcionalidades entre las que destacan especialmente la gran flexibilidad en el diseño y un panel de control. Además Magento fue adquirida en 2011 por Ebay, una empresa de comercio electrónico.

Dependiendo de la modalidad y el producto escogido, podremos obtener funcionalidades tales como:

* Flexibilidad para crear enriquecedoras experiencias a los consumidores.
* flujos de trabajo a medida back-end para su negocio.
* Cientos de extensiones.
* Arquitectura abierta.
* Integración con otras soluciones de comercio electrónico.
* Red de socios y desarrolladores certificados.
* Apoyo de expertos, capacitación y servicios de consultoría.
* Vista y gestión global de inventario.
* Modular, flexible e integrada.
* Velocidad rápida al mercado.
* Opciones de pago seguro y flexible.
* El software de código abierto para el comercio electrónico.
* Descarga gratuita.
* Apoyo basado en la Comunidad.

A.III.3.- WordPress - WooCommerce.

¿Qué es WordPress?

WordPress es un sistema de gestión de contenidos o CMS enfocado a la creación de cualquier tipo de sitio, aunque ha alcanzado una gran relevancia usado para la creación de blogs. Ha sido desarrollado en el lenguaje PHP para entornos que ejecuten MySQL y Apache, bajo licencia GPL y es software libre.

¿Qué es WooCommerce?

WooCommerce es un plugin de tienda online para WordPress muy popular que está teniendo un gran crecimiento. WooCommerce está construido para integrarse perfectamente con WordPress, por lo que es la elección obvia de eCommerce para usuarios de WordPress y que conecta con el creciente ecosistema de WordPress.

Tema 2. Desarrollo de software.

1.-Software y programa. Tipos de software.

Es de sobra conocido que el ordenador se compone de dos partes bien diferenciadas: Hardware y Software.

El software es el conjunto de programas informáticos que actúan sobre el hardware para ejecurae lo que el usuario desee.

Hay tres tipos de software:

El sistema operativo es el software base que ha de estar instalado y configurado en nuestro ordenador para que las aplicaciones puedan ejecutarse y funcionar.

El software de programación es el conjunto de herramientas que nos permiten desarrollar programas informaticos, y las aplicaciones informáticas son un conjunto de programas que tienen una finalidad más o menos concreta.

Un programa es un conjunto de intrucciones escritas en un lenguaje de programación.

2.- Relación hardware-software

Esta relación software-hardware la podemos poner de manifiesto desde puntos de vista:

1. Desde el punto de vista del sistema operativo

El sistema operativo es el encargo de coordinar al hardware durante el funcionamiento del ordenador, actuando como intermediario entre éste y las aplicaciones que este corriendo en un momento dado.

El sistema controla el hardware.

1. Desde el punto de vista de las aplicaciones

Las aplicaciones son un conjunto de programas escritos en algún tipo de lenguaje de programación para que los hymanos puedan entenderlo. Esta parte se vera en esta unidad.

3.- Desarrollo de software.

El proceso de desarrollo, que en un principio puede parecer una tarea simple, consta de una serie de pasos de obligado cumplimiento, pues sólo así podremos garantizar que los programas creados son eficientes, fiables, seguros y responden a las necesidades de los usuarios finales (aquellos que van a utilizar el programa).

Como veremos con más detenimiento a lo largo de la unidad, el desarrollo de software es un proceso que conlleva una serie de pasos. Genéricamente, estos pasos se denominan Etapas en el desarrollo de software.

3.1.- Ciclos de vida del software.

Los ciclos de dividen en etapas que se explicaran cuando más avanzemos en el temario.

Diversos autores han planteado distintos modelos de ciclos de vida, pero los más conocidos y utilizados son los que aparecen a continuación:

* Modelo en Cascada

Es el modelo de vida clásico del software.

Es prácticamente imposible que se pueda utilizar, ya que requiere conocer de antemano todos los requisitos del sistema. Sólo es aplicable a pequeños desarrollos, ya que las etapas pasan de una a otra sin retorno posible

* Modelo en Cascada con Realimentación

Es uno de los modelos más utilizados. Proviene del modelo anterior, pero se introduce una realimentación entre etapas de forma que podamos volver atrás en cualquier momento para corregir, modificar o depurar algún aspecto. No es el más idoneo cuando se prebeen cambio en el desarrollo.

* Modelos Evolutivos

Son parecidos a los anteriores solo que tienen en cuenta la naturelza cambiante del diseño de software.

Hay dos modelos:

* + Modelo Iterativo Incremental

Está basado en el modelo en cascada con realimentación, donde las fases se repiten y refinan, y van propagando su mejora a las fases siguientes.

* + Modelo espiral

Es una combinación del modelo anterior con el modelo en cascada. En él, el software se va construyendo repetidamente en forma de versiones que son cada vez mejores, debido a que incrementan la funcionalidad en cada versión. Es un modelo bastante complejo.

3.2.- Herramientas de apoyo al desarrollo del software.

Las herramientas CASE son un conjunto de aplicaciones que se utilizan en el desarrollo de software con el objetivo de reducir costes y tiempo del proceso, mejorando por tanto la productividad del proceso.

El desarrollo rápido de aplicaciones o RAD es un proceso de desarrollo de software que comprende el desarrollo iterativo, la construcción de prototipos y el uso de utilidades CASE.

La tecnología CASE trata de automatizar las fases del desarrollo de software para que mejore la calidad del proceso y del resultado final. En concreto, estas herramientas permiten:

* Mejorar la planificación del proyecto.
* Darle agilidad al proceso.
* Poder reutilizar partes del software en proyectos futuros.
* Hacer que las aplicaciones respondan a estándares.
* Mejorar la tarea del mantenimiento de los programas.
* Mejorar el proceso de desarrollo, al permitir visualizar las fases de forma gráfica.

CLASIFICACIÓN

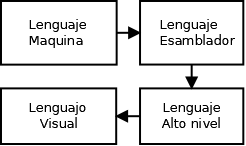
Normalmente, las herramientas CASE se clasifican en función de las fases del ciclo de vida del software en la que ofrecen ayuda:

* U-CASE: ofrece ayuda en las fases de planificación y análisis de requisitos.
* M-CASE: ofrece ayuda en análisis y diseño.
* L-CASE: ofrece ayuda en la programación del software, detección de errores del código, depuración de programas y pruebas, y en la generación de la documentación del proyecto.

4.- Lenguajes de programación.

Podemos definir un Lenguaje de Programación como un idioma creado de forma artificial, formado por un conjunto de símbolos y normas que se aplican sobre un alfabeto para obtener un código, que el hardware de la computadora pueda entender y ejecutar.

Hay multitud de lenguajes de programación, cada uno con unos símbolos y unas estructuras diferentes. Además, cada lenguaje está enfocado a la programación de tareas o áreas determinadas. Por ello, la elección del lenguaje a utilizar en un proyecto es una cuestión de extrema importancia.



Lenguaje máquina:

* Sus instrucciones son combinaciones de unos y ceros.
* Es el único lenguaje que entiende directamente el ordenador (no necesita traducción).
* Fue el primer lenguaje utilizado.
* Es único para cada procesador (no es portable de un equipo a otro).
* Hoy día nadie programa en este lenguaje.

Lenguaje ensamblador:

* Sustituyó al lenguaje máquina para facilitar la labor de programación.
* En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas).
* Necesita traducción al lenguaje máquina para poder ejecutarse.
* Sus instrucciones son sentencias que hacen referencia a la ubicación física de los archivos en el equipo.
* Es difícil de utilizar.

Lenguaje de alto nivel basados en código:

* Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación.
* En lugar de mnemotécnicos, se utilizan sentencias y órdenes derivadas del idioma inglés. Necesita traducción al lenguaje máquina.
* Son más cercanos al razonamiento humano.
* Son utilizados hoy día, aunque la tendencia es que cada vez menos.

Lenguajes visuales:

* Están sustituyendo a los lenguajes de alto nivel basados en código.
* En lugar de sentencias escritas, se programa gráficamente usando el ratón y diseñando directamente la apariencia del software.
* Su correspondiente código se genera automáticamente.
* Necesitan traducción al lenguaje máquina.
* Son completamente portables de un equipo a otro.

4.1.- Concepto y características.

CONCEPTO

* Alfabeto: conjunto de símbolos permitidos.
* Sintaxis: normas de construcción permitidas de los símbolos del lenguaje.
* Semántica: significado de las construcciones para hacer acciones válidas.

CARACTERÍSTICAS

Podemos clasificar los distintos tipos de Lenguajes de Programación en base a distintas características:

* Según lo cerca que esté del lenguaje humano
  + Lenguajes de Programación De alto nivel: por su esencia, están más próximos al razonamiento humano.
  + Lenguajes de Programación De bajo nivel: están más próximos al funcionamiento interno de la computadora:
    - Lenguaje Ensamblador.
    - Lenguaje Máquina.
* Según la técnica de programación utilizada:
  + Lenguajes de Programación Estructurados: Usan la técnica de programación estructurada. Ejemplos: Pascal, C, etc.
  + Lenguajes de Programación Orientados a Objetos: Usan la técnica de programación orientada a objetos. Ejemplos: C++, Java, Ada, Delphi, etc.
  + Lenguajes de Programación Visuales: Basados en las técnicas anteriores, permiten programar gráficamente, siendo el código correspondiente generado de forma automática. Ejemplos: Visual Basic.Net, Borland Delphi, etc.

4.2.-Lenguajes de programación estructurados.

La programación estructurada se define como una técnica para escribir lenguajes de programación que permite sólo el uso de tres tipos de sentencias o estructuras de control:

* Sentencias secuenciales.
* Sentencias selectivas (condicionales).
* Sentencias repetitivas (iteraciones o bucles).

VENTAJAS DE LA PROGRAMACIÓN ESTRUCTURADA

* Los programas son fáciles de leer, sencillos y rápidos.
* El mantenimiento de los programas es sencillo.
* La estructura del programa es sencilla y clara.

INCONVENIENTES

* Todo el programa se concentra en un único bloque (si se hace demasiado grande es difícil manejarlo).
* No permite reutilización eficaz de código, ya que todo va "en uno". Es por esto que a la programación estructurada le sustituyó la programación modular, donde los programas se codifican por módulos y bloques, permitiendo mayor funcionalidad.

4.3.- Lenguajes de programación orientados a objetos.

En la P.O.O. los programas se componen de objetos independientes entre sí que colaboran para realizar acciones. Los objetos son reutilizables para proyectos futuros.

Razones:

* El código es reutilizable.
* Si hay algún error, es más fácil de localizar y depurar en un objeto que en un programa entero.

Características:

* Los objetos del programa tendrán una serie de atributos que los diferencian unos de otros.
* Se define clase como una colección de objetos con características similares.
* Mediante los llamados métodos, los objetos se comunican con otros produciéndose un cambio de estado de los mismos.
* Los objetos son, pues, como unidades individuales e indivisibles que forman la base de este tipo de programación.

5.- Fases en el desarrollo y ejecución del software.

Estas etapas son:

1. ANÁLISIS DE REQUISITOS.

Se especifican los requisitos funcionales y no funcionales del sistema.

1. DISEÑO.

Se divide el sistema en partes y se determina la función de cada una.

1. CODIFICACIÓN.

Se elige un Lenguajes de Programación y se codifican los programas.

1. PRUEBAS.

Se prueban los programas para detectar errores y se depuran.

1. DOCUMENTACIÓN.

De todas las etapas, se documenta y guarda toda la información.

1. EXPLOTACIÓN.

Instalamos, configuramos y probamos la aplicación en los equipos del cliente.

1. MANTENIMIENTO.

Se mantiene el contacto con el cliente para actualizar y modificar la aplicación el futuro.

5.1.- Analisis

¿Qué se hace en esta fase? Se especifican y analizan los requisitos funcionales y no funcionales del sistema.

Requisitos:

* Funcionales: Qué funciones tendrá que realizar la aplicación. Qué respuesta dará la aplicación ante todas las entradas. Cómo se comportará la aplicación en situaciones inesperadas.
* No funcionales: Tiempos de respuesta del programa, legislación aplicable, tratamiento ante la simultaneidad de peticiones, etc.

Lo fundamental es la buena comunicación entre el analista y el cliente para que la aplicación que se va a desarrollar cumpla con sus expectativas.

La culminación de esta fase es el documento ERS (Especificación de Requisitos Software). En este documento quedan especificados:

* La planificación de las reuniones que van a tener lugar.
* Relación de los objetivos del usuario cliente y del sistema.
* Relación de los requisitos funcionales y no funcionales del sistema.
* Relación de objetivos prioritarios y temporización.
* Reconocimiento de requisitos mal planteados o que conllevan contradicciones, etc.

5.2.- Diseño.

Se debe dividir el sistema en partes y establecer qué relaciones habrá entre ellas. Decidir qué hará exactamente cada parte.

En este punto, se deben tomar decisiones importantes, tales como:

* Entidades y relaciones de las bases de datos.
* Selección del lenguaje de programación que se va a utilizar.
* Selección del SGBD.
* Definición de diagrama de clases.
* Definición de diagrama de colaboración.
* Definición de diagrama de paso de mensajes.
* Definición de diagrama de casos de uso.

5.3.- Codificación. Tipos de código.

Esta tarea la realiza el programador y tiene que cumplir exhaustivamente con todos los datos impuestos en el análisis y en el diseño de la aplicación.

Las características deseables de todo código son:

* Modularidad: que esté dividido en trozos más pequeños.
* Corrección: que haga lo que se le pide realmente.
* Fácil de leer: para facilitar su desarrollo y mantenimiento futuro.
* Eficiencia: que haga un buen uso de los recursos.
* Portabilidad: que se pueda implementar en cualquier equipo.

Durante esta fase, el código pasa por diferentes estados:

* Código Fuente: es el escrito por los programadores en algún editor de texto.
* Código Objeto: es el código binario resultado de compilar el código fuente.
* Código Ejecutable: Es el código binario resultante de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias.

Los programas interpretados no producen código objeto. El paso de fuente a ejecutable es directo.

5.4.- Fases en la obtención de código.

5.4.1.- Fuente.

El código fuente es el conjunto de instrucciones que la computadora deberá realizar, escritas por los programadores en algún lenguaje de alto nivel. Este conjunto de instrucciones no es directamente ejecutable por la máquina, sino que deberá ser traducido al lenguaje máquina, que la computadora será capaz de entender y ejecutar.

Para obtener el código fuente de una aplicación informática:

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

Puesto que, como hemos dicho antes, este código es inteligible por la máquina, habrá que TRADUCIRLO, obteniendo así un código equivalente pero ya traducido a código binario que se llama código objeto. Que no será directamente ejecutable por la computadora si éste ha sido compilado.

Un aspecto importante a tener en cuenta es su licencia. Así, en base a ella, podemos distinguir dos tipos de código fuente:

* Código fuente abierto. Es aquel que está disponible para que cualquier usuario pueda estudiarlo, modificarlo o reutilizarlo.
* Código fuente cerrado. Es aquel que no tenemos permiso para editarlo.

5.4.2.- Objeto.

El código objeto es un código intermedio. Es el resultado de traducir código fuente a un código equivalente formado por unos y ceros que aún no puede ser ejecutado directamente por la computadora.

Sólo se genera código objeto una vez que el código fuente está libre de errores sintácticos y semánticos.

El proceso de traducción de código fuente a código objeto puede realizarse de dos formas:

1. Compilación: El proceso de traducción se realiza sobre la totalidad del código fuente, en un solo paso. Se crea código objeto que habrá que enlazar. El software responsable se llama compilador.
2. Interpretación: El proceso de traducción del código fuente se realiza línea a línea y se ejecuta simultáneamente. No existe código objeto intermedio. El software responsable se llama intérprete.

El código objeto es código binario, pero no puede ser ejecutado por la computadora.

5.4.3.- Ejecutable.

El código ejecutable, resultado de enlazar los archivos de código objeto, consta de un único archivo que puede ser directamente ejecutado por la computadora. No necesita ninguna aplicación externa. Este archivo es ejecutado y controlado por el sistema operativo.

En el esquema de generación de código ejecutable, vemos el proceso completo para la generación de ejecutables:

1. A partir de un editor, escribimos el lenguaje fuente con algún Lenguaje de programación. (En el ejemplo, se usa Java).
2. A continuación, el código fuente se compila obteniendo código objeto o bytecode.
3. Ese bytecode, a través de la máquina virtual (se verá en el siguiente punto), pasa a código máquina, ya directamente ejecutable por la computadora.

5.5.- Máquinas virtuales.

Una máquina virtual es un tipo especial de software cuya misión es separar el funcionamiento del ordenador de los componentes hardware instalados. Esta capa de software desempeña un papel muy importante en el funcionamiento de los lenguajes de programación, tanto compilados como interpretados.

Las funciones principales de una máquina virtual son las siguientes:

* Conseguir que las aplicaciones sean portables.
* Reservar memoria para los objetos que se crean y liberar la memoria no utilizada.
* Comunicarse con el sistema donde se instala la aplicación (huésped), para el control de los dispositivos hardware implicados en los procesos.
* Cumplimiento de las normas de seguridad de las aplicaciones.

Características de las máquinas virtuales:

* La máquina virtual aísla la aplicación de los detalles físicos del equipo en cuestión. Cuando el código fuente se compila se obtiene código objeto (bytecode, código intermedio).Para ejecutarlo en cualquier máquina se requiere tener independencia respecto al hardware concreto que se vaya a utilizar. La máquina virtual funciona como una capa de software de bajo nivel y actúa como puente entre el bytecode de la aplicación y los dispositivos físicos del sistema.
* La máquina virtual verifica todo el bytecode antes de ejecutarlo.
* La máquina virtual protege direcciones de memoria.

5.5.1.- Frameworks.

Esquema formado por un óvalo y tres círculos azulados. En la parte central del esquema se encuentra el óvalo, más grande que los círculos, en cuyo interior se puede leer

Un framework es una estructura de ayuda al programador, en base a la cual podemos desarrollar proyectos sin partir desde cero.

* Ventajas de utilizar un framework:
  + Desarrollo rápido de software.
  + Reutilización de partes de código para otras aplicaciones.
  + Diseño uniforme del software.
  + Portabilidad de aplicaciones de un computador a otro, ya que los bytecodes que se generan a partir del lenguaje fuente podrán ser ejecutados sobre cualquier máquina virtual.
* Inconvenientes:
  + Gran dependencia del código respecto al framework utilizado (sin cambiamos de framework, habrá que reescribir gran parte de la aplicación).
  + La instalación e implementación del framework en nuestro equipo consume bastantes recursos del sistema.

5.5.2.- Entornos de ejecución.

Un entorno de ejecución es un servicio de máquina virtual que sirve como base software para la ejecución de programas. En ocasiones pertenece al propio sistema operativo, pero también se puede instalar como software independiente que funcionará por debajo de la aplicación.

Se denomina runtime al tiempo que tarda un programa en ejecutarse en la computadora.

Durante la ejecución, los entornos se encargarán de:

* Configurar la memoria principal disponible en el sistema.
* Enlazar los archivos del programa con las bibliotecas existentes y con los subprogramas creados.
* Depurar los programas: comprobar la existencia (o no existencia) de errores semánticos del lenguaje (los sintácticos ya se detectaron en la compilación).

Funcionamiento del entorno de ejecución:

* El entorno de ejecución está formado por la máquina virtual y los API's (bibliotecas de clases estándar, necesarias para que la aplicación, escrita en algún Lenguaje de Programación, pueda ser ejecutada).
* El entorno funciona como intermediario entre el lenguaje fuente y el sistema operativo, y consigue ejecutar aplicaciones.
* Sin embargo, si lo que queremos es desarrollar nuevas aplicaciones, no es suficiente con el entorno de ejecución.
* Adelantándonos a lo que veremos en la próxima unidad, para desarrollar aplicaciones necesitamos algo más.

5.5.3.- Java runtime environment.

Concepto

Se denomina JRE al Java Runtime Environment (entorno en tiempo de ejecución Java).

El JRE se compone de un conjunto de utilidades que permitirá la ejecución de programas java sobre cualquier tipo de plataforma.

Componentes

JRE está formado por:

Una máquina virtual Java (JMV o JVM si consideramos las siglas en inglés), que es el programa que interpreta el código de la aplicación escrito en Java.

Bibliotecas de clase estándar que implementan el API de Java.

Las dos: JMV y API de Java son consistentes entre sí, por ello son distribuidas conjuntamente.

5.6.- Pruebas.

La realización de pruebas es imprescindible para asegurar la validación verificación del software construido.

Entre todas las pruebas que se efectúan sobre el software podemos distinguir básicamente:

PRUEBAS UNITARIAS

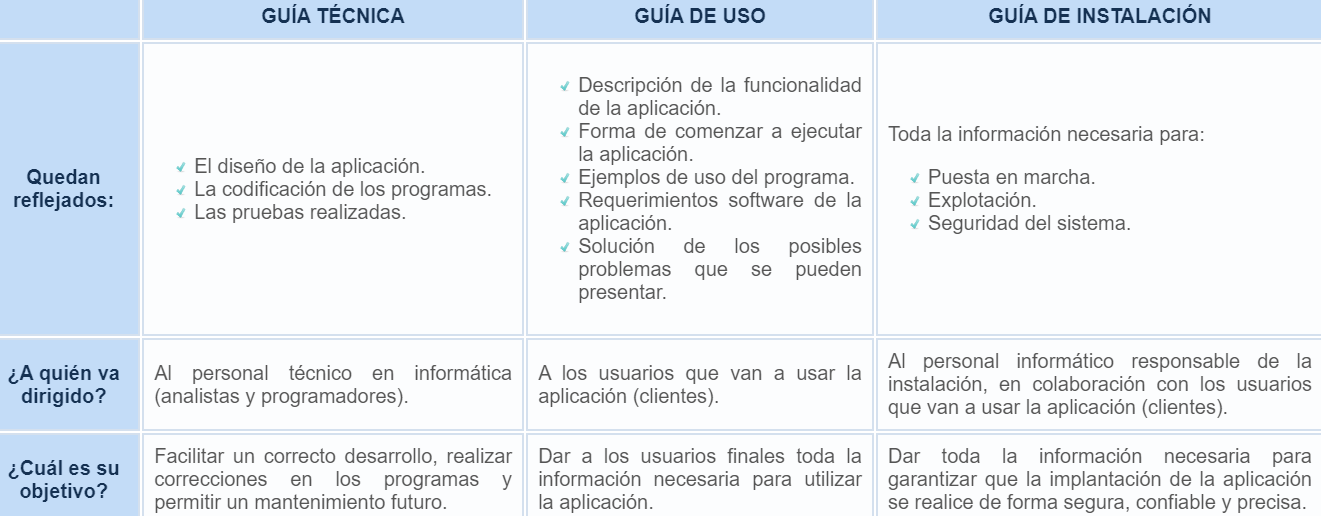
Consisten en probar, una a una, las diferentes partes de software y comprobar su funcionamiento (por separado, de manera independiente). JUnit es el entorno de pruebas para Java.

PRUEBAS DE INTEGRACIÓN

Se realizan una vez que se han realizado con éxito las pruebas unitarias y consistirán en comprobar el funcionamiento del sistema completo: con todas sus partes interrelacionadas.

5.7.- Documentación.

Todas las etapas en el desarrollo de software deben quedar perfectamente documentadas.



5.8.- Explotación.

La explotación es la fase en que los usuarios finales conocen la aplicación y comienzan a utilizarla.

La explotación es la instalación, puesta a punto y funcionamiento de la aplicación en el equipo final del cliente.

* En el proceso de instalación, los programas son transferidos al computador del usuario cliente y posteriormente configurados y verificados. Es recomendable que los futuros clientes estén presentes en este momento e irles comentando cómo se va planteando la instalación.
* En este momento, se suelen llevan a cabo las Beta Test, que son las últimas pruebas que se realizan en los propios equipos del cliente y bajo cargas normales de trabajo.
* Una vez instalada, pasamos a la fase de configuración. En ella, asignamos los parámetros de funcionamiento normal de la empresa y probamos que la aplicación es operativa. También puede ocurrir que la configuración la realicen los propios usuarios finales, siempre y cuando les hayamos dado previamente la guía de instalación. Y también, si la aplicación es más sencilla, podemos programar la configuración de manera que se realice automáticamente tras instalarla. (Si el software es "a medida", lo más aconsejable es que la hagan aquellos que la han fabricado).
* Una vez se ha configurado, el siguiente y último paso es la fase de producción normal. La aplicación pasa a manos de los usuarios finales y se da comienzo a la explotación del software.

5.9.- Mantenimiento.

El mantenimiento se define como el proceso de control, mejora y optimización del software. Su duración es la mayor en todo el ciclo de vida del software, ya que también comprende las actualizaciones y evoluciones futuras del mismo.

Los tipos de cambios que hacen necesario el mantenimiento del software son los siguientes:

* Perfectivos: Para mejorar la funcionalidad del software.
* Evolutivos: El cliente tendrá en el futuro nuevas necesidades. Por tanto, serán necesarias modificaciones, expansiones o eliminaciones de código.
* Adaptativos: Modificaciones, actualizaciones... para adaptarse a las nuevas tendencias del mercado, a nuevos componentes hardware, etc.
* Correctivos: La aplicación tendrá errores en el futuro (sería utópico pensar lo contrario).

6.- Entornos de desarrollo.

Un entorno integrado de desarrollo (IDE), es un tipo de software compuesto por un conjunto de herramientas de programación. En concreto, el IDE se compone de:

* Editor de código de programación.
* Compilador.
* Intérprete.
* Depurador.
* Constructor de interfaz gráfico.

Los primeros entornos de desarrollo integrados nacieron a principios de los años 70, y se popularizaron en la década de los 90. Tienen el objetivo de ganar fiabilidad y tiempo en los proyectos de software.

6.1.-Evolucion historia

En las décadas de utilización de la tarjeta perforada como sistema de almacenamiento el concepto de Entorno de Desarrollo Integrado sencillamente no tenía sentido.

El primer lenguaje de programación que utilizó un IDE fue el BASIC (que fue el primero en abandonar también las tarjetas perforadas o las cintas de papel).

Éste primer IDE estaba basado en consola de comandos exclusivamente (normal por otro lado, si tenemos en cuenta que hasta la década de los 90 no entran en el mercado los sistemas operativos con interfaz gráfica). Sin embargo, el uso que hace de la gestión de archivos, compilación y depuración; es perfectamente compatible con los IDE actuales.

A nivel popular, el primer IDE puede considerarse que fue el IDE llamado Maestro. Nació a principios de los 70 y fue instalado por unos 22.000 programadores en todo el mundo. Lideró este campo durante los años 70 y 80.

6.2.- Funciones de un entorno de desarrollo.

Las funciones de los IDE son:

* Editor de código: coloración de la sintaxis.
* Auto-completado de código, atributos y métodos de clases.
* Identificación automática de código.
* Herramientas de concepción visual para crear y manipular componentes visuales.
* Asistentes y utilidades de gestión y generación de código.
* Archivos fuente en unas carpetas y compilados a otras.
* Compilación de proyectos complejos en un solo paso.
* Control de versiones: tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de auto-recuperación a un estado anterior estable.
* Soporta cambios de varios usuarios de manera simultánea.
* Generador de documentación integrado.
* Detección de errores de sintaxis en tiempo real.

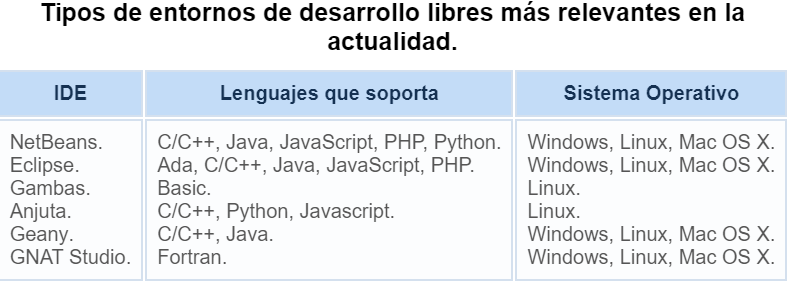
Otras funciones importantes son:

* Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo cambiar el nombre a una variable).
* Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
* Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
* Aumento de funcionalidades a través de la gestión de sus módulos y plugins.
* Administración de las interfaces de usuario (menús y barras de herramientas).
* Administración de las configuraciones del usuario.

6.3.- Entornos integrados libres y propietarios.

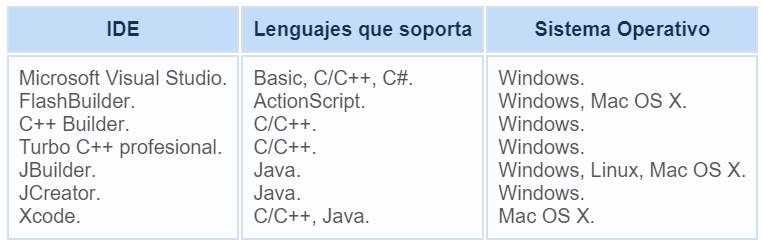
Entornos Integrados Libres

Son aquellos con licencia de uso público.



Entornos Integrados Propietarios

Son aquellos entornos integrados de desarrollo que necesitan licencia. No son free software, hay que pagar por ellos.



6.4.- Estructura de entornos de desarrollo.

Estos componentes son:

* Editor de textos: Resalta y colorea la sintaxis, tiene la función de autocompletar código, ayuda y listado de parámetros de funciones y métodos de clase. Inserción automática de paréntesis, corchetes, tabulaciones y espaciados.
* Compilador/intérprete: Detección de errores de sintaxis en tiempo real. Características de refactorización.
* Depurador: Botón de ejecución y traza, puntos de ruptura y seguimiento de variables. Opción de depurar en servidores remotos.
* Generador automático de herramientas: Para la visualización, creación y manipulación de componentes visuales y todo un arsenal de asistentes y utilidades de gestión y generación código.
* Interfaz gráfica: Nos brinda la oportunidad de programar en varios lenguajes con un mismo IDE. Es una interfaz agradable que puede acceder a innumerables bibliotecas y plugins, aumentando las opciones de nuestros programas.