Tutoría 17 de Noviembre 2020

- ✓ Tarea Evaluable
- ✓ Exámen
- ✓ Resumen teoría.

- ✓ Tarea Evaluable: Ojo elegir solo una
- ✓ tienen distinta **nota máxima**: Eclipse sobre 8 e IntelliJ IDEA sobre 10

Tareas Evaluables

Las tareas evaluables son voluntarias, aunque suponen un 10% de la nota final del trimestre.
La parte teórica se puntúa sobre un 90%

Tarea Evaluable Primer Trimestre

Creación de un manual siguiendo las pautas establecidas en el guión.

- Las tareas evaluables son voluntarias, aunque suponen un 10% de la nota final. **Eclipse sobre 8, IntelliJ sobre 10.**
- La parte teórica se puntúa sobre un 90%.
- **Fecha tope de entrega: 10 Diciembre.**

Elegir una de las dos

Tarea Evaluable Guión Eclipse

Tarea Evaluable Guión IntelliJ IDEA

Entrega de Tarea PRIMER Trimestre

Examen

Examen 1º Trimestre 1 Diciembre 16:00h

- Acudir con **DNI válido**
- Se ruega **puntualidad y no se permitirá la salida del exámen durante los 10 min posteriores al inicio de la prueba.**
- Acudir con **usuario y contraseña de Aulas Virtuales**
- **Acordarse de bolígrafo, no se puede prestar material.**

- ✓ DNI y boli (no lápiz)
- ✓ Justificantes:
  - ✓ Perimetrales *(previo)*
  - ✓ Por trabajo *(el día del examen)*
- ✓ Confirmar asistencia en el cuestionario
- ✓ ¿Dónde veo las calificaciones?

Ut1.- Desarrollo de software.

Objetivos

Reconocer la relación de los programas con los componentes del sistema informático.

Identificar el conjunto de etapas por las que debe pasar una aplicación informática, desde que se recibe la petición por parte de un cliente hasta que llega a sus manos y se mantiene, introduciendo cuantas actualizaciones o modificaciones sean necesarias.

Diferenciar los conceptos de código fuente, código objeto y código ejecutable.

Clasificar los lenguajes de programación.

Siguiente »

Concepto de Software

Conjunto de **programas** informáticos que actúan sobre el hardware para ejecutar lo que el usuario desee.

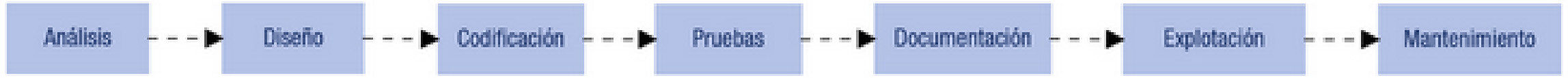
- Según la **tarea** que realizan:
    1. De sistemas
    2. De programación
    3. De aplicación
- Según el tipo de **distribución**:
    1. Shareware
    2. Freeware
    3. Adware
    4. De uso específico
- Según el tipo de **licencia**:
    1. Libre
    2. Propietario
    3. De Dominio público

Ciclo de Vida

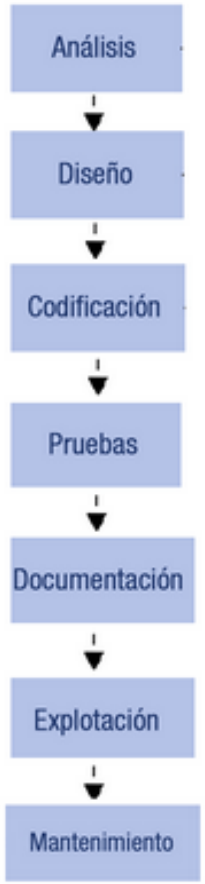
Proceso que incluye todos los pasos que se dan desde que se concibe una idea hasta que una aplicación informática está implementada en el ordenador y funcionando.

Pasos:

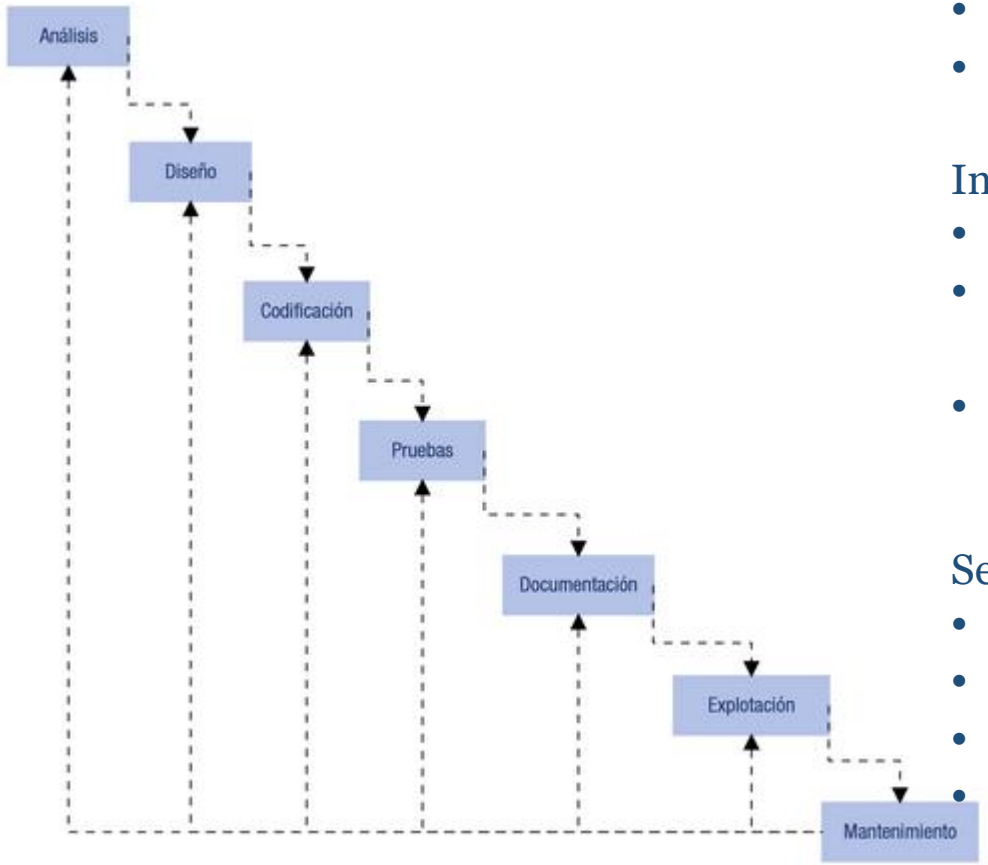
- **Análisis**: especificación de requisitos funcionales (comportamiento) y no funcionales (tiempos, legislación).
- **Diseño**: división del sistema en partes y sus relaciones.
- **Codificación**: elegir lenguaje de programación y programar.
- **Pruebas**: detección de errores y depuración.
- **Documentación**: documentar todas la etapas.
- **Explotación**: instalación, configuración y prueba en el equipo del cliente (en producción).
- **Mantenimiento**: actualización y depuración.



Modelos de Ciclo de Vida



En Cascada



En Cascada con Realimentación

Ventajas:

- Fácil de comprender, planificar y seguir.
- Calidad del producto alta.

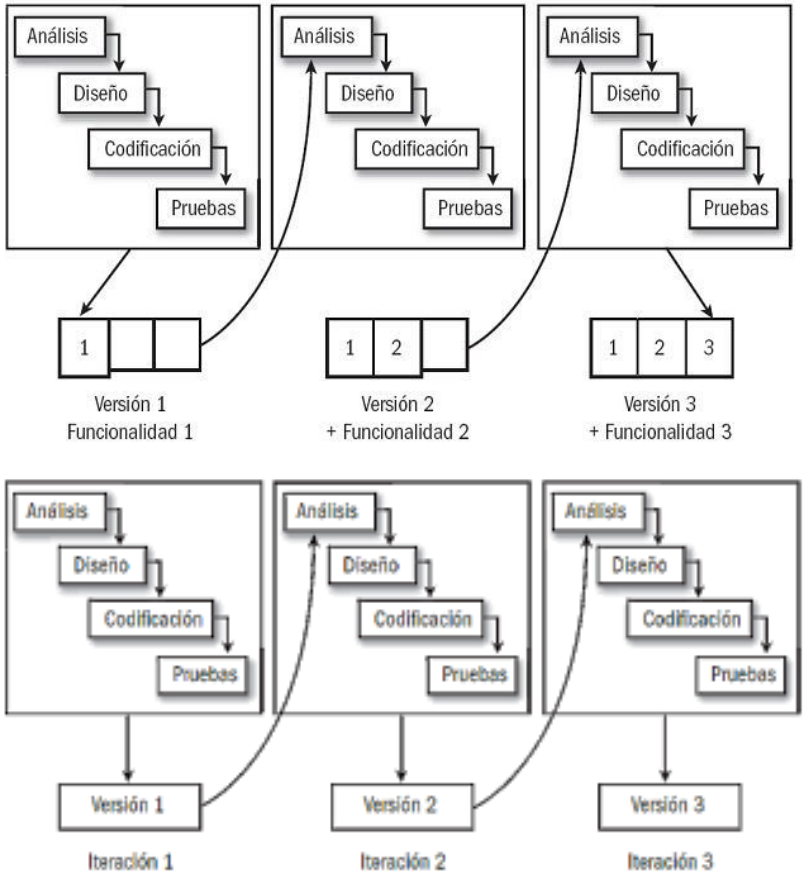
Inconvenientes:

- Necesita definir todos los requisitos al inicio.
- Es difícil volver atrás si se cometen errores en una etapa.
- El producto no está disponible hasta que no se termina.

Se recomienda cuando:

- El proyecto es similar a otro finalizado con éxito
- Requisitos estables y bien definidos.
- Clientes no necesitan versiones intermedias.
- Proyecto muy pequeño

Modelos de Ciclo de Vida



Evolutivo - 1 Incremental 2 Iterativo

Ventajas:

- No se necesitan reconocer lo requisitos al comienzo
- Permite la entrega temprana de partes operativas.

Inconvenientes:

- Es difícil estimar el esfuerzo y el coste final.
- Tiene el riesgo de no acabar.
- No recomendable para sistemas en tiempo real, de alto nivel de seguridad, de procedimiento distribuido y/o alto índice de riesgos.

Se recomienda cuando:

- Los requisitos o el diseño no están completamente definidos y es posible que haya grandes cambios.
- Se están probando o introduciendo nuevas tecnologías.

Modelos de Ciclo de Vida

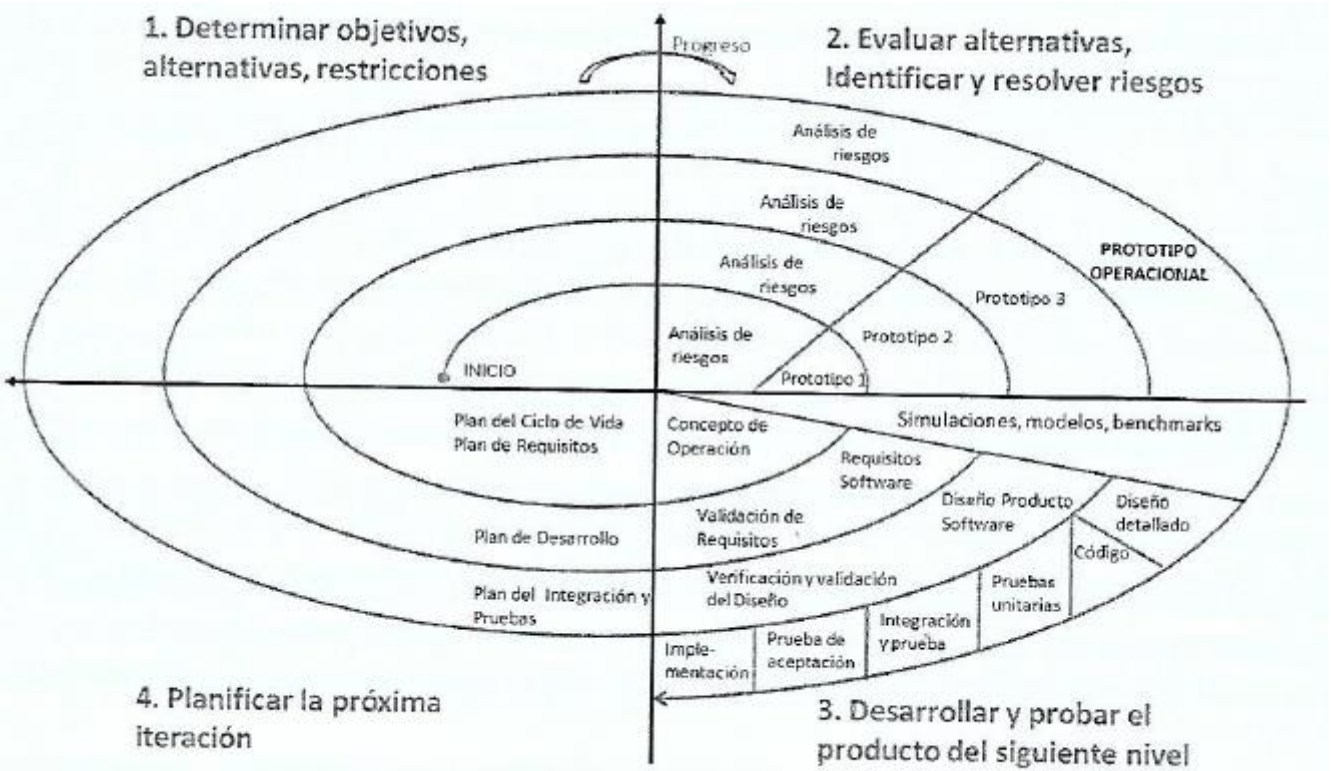


Figura 1.8. Ejemplo de Modelo en espiral.

Evolutivo - en Espiral

Ventajas:

- No se necesitan tener una definición completa de los requisitos para empezar a funcionar.
- Análisis de riesgos en todas las etapas.
- Incorpora objetivos de calidad.
- Reduce riesgos del proyecto.

Inconvenientes:

- Es difícil evaluar los riesgos.
- El coste del proyecto aumenta a medida que la espiral pasa por las sucesivas iteraciones.
- El éxito del proyecto depende en gran medida de la fase de análisis de riesgos.

Se recomienda cuando:

- Proyectos de gran tamaño y necesitan grandes cambios.
- Proyectos donde sea importante el factor riesgo.

Lenguajes de Programación

Idioma creado de forma artificial, formado por un conjunto de símbolos (**léxico**) y normas (**sintaxis, semántica**) que se aplican sobre un alfabeto para obtener un código que el hardware pueda ejecutar.

- Según el nivel de **abstracción:**
  1. Lenguaje Máquina.
  2. Lenguaje Ensamblador.
  3. Lenguaje Alto Nivel.
  4. Lenguaje. Visual.
- Según el **paradigma de programación**:
  1. Estructurados
  2. POO
  3. Visuales
- Según la forma de **ejecución:**
  1. Lenguajes Interpretados.
  2. Lenguajes Compilados.

Fases en el desarrollo de una aplicación:

1. **ANÁLISIS DE REQUISITOS.:** Se especifican los requisitos funcionales y no funcionales del sistema.
   - **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.
     Ejemplo Agenda: R1.- El usuario puede agregar un contacto
   - **No funcionales**: Tiempos de respuesta del programa, legislación aplicable, tratamiento ante la simultaneidad de peticiones, etc.
     Ejemplo Agenda: R1.- La aplicación debe funcionar en sistemas Windows y Linux.
2. **DISEÑO.** Se divide el sistema en partes y se determina la función de cada una.
3. **CODIFICACIÓN.** Se elige un Lenguaje de Programación y se codifican los programas.

Fases en el desarrollo de una aplicación:

4. **PRUEBAS.** Se prueban los programas para detectar errores y se depuran.
   - **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. **DOCUMENTACIÓN.** De todas las etapas, se documenta y guarda toda la información.

Documentos a elaborar en el proceso de desarrollo de software

	GUÍA TÉCNICA	GUÍA DE USO	GUÍA DE INSTALACIÓN
Quedan reflejados:	✓ El diseño de la aplicación. ✓ La codificación de los programas. ✓ Las pruebas realizadas.	✓ Descripción de la funcionalidad de la aplicación. ✓ Forma de comenzar a ejecutar la aplicación. ✓ Ejemplos de uso del programa. ✓ Requerimientos software de la aplicación. ✓ Solución de los posibles problemas que se pueden presentar.	Toda la información necesaria para: ✓ Puesta en marcha. ✓ Explotación. ✓ Seguridad del sistema.
¿A quién va dirigido?	Al personal técnico en informática (analistas y programadores).	A los usuarios que van a usar la aplicación (clientes).	Al personal informático responsable de la instalación, en colaboración con los usuarios que van a usar la aplicación (clientes).
¿Cuál es su objetivo?	Facilitar un correcto desarrollo, realizar correcciones en los programas y permitir un mantenimiento futuro.	Dar a los usuarios finales toda la información necesaria para utilizar la aplicación.	Dar toda la información necesaria para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa.

Fases en el desarrollo de una aplicación:

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

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

   El mantenimiento se define como el proceso de control, mejora y optimización del software.

   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).

Tipos de Código

- Código **fuente**: instrucciones escritas en lenguaje de alto nivel (editor de texto).
- Código **objeto**: resultado de compilar (traducir) el código fuente. (bytecode).
- Código **ejecutable**: código binario resultante de enlazar el código objeto con rutinas y bibliotecas necesarias. (código máquina). Ejecutado y controlado por el S.O.

Máquinas Virtuales

Una máquina virtual es un tipo especial de software cuya misión es separar el funcionamiento del ordenador de los componentes hardware instalados.

1.- Máquinas Virtuales de Sistema: Permiten ejecutar en la misma máquina física varias máquinas virtuales cada una con un sistema operativo, de esta manera pueden coexistir diferentes sistemas operativos sobre un mismo equipo real.
Ejemplo: VMWare, VirtualBox….

2.- Máquinas Virtuales de Proceso: A veces llamada Máquina virtual de Aplicación se ejecuta como un proceso normal dentro de un sistema operativo y soporta un solo proceso. Su objetivo es proporcionar un entorno de ejecución independiente de la plataforma hardware y del sistema operativo y que oculte los detalles de la plataforma real de forma que se pueda ejecutar sobre cualquier plataforma.
La máquina se inicia cuando se inicia el proceso que se quiere ejecutar y se detiene cuando finaliza.
Ejemplo: Máquina virtual de Java.

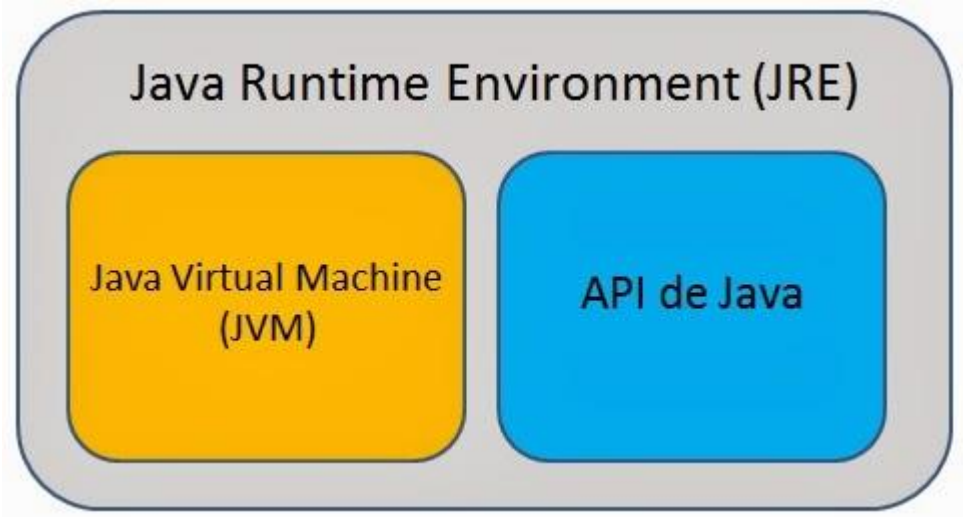
Framework

Un framework es una estructura de ayuda al programador, en base a la cual podemos desarrollar proyectos sin partir desde cero.
Se trata de una plataforma software donde están definidos programas soporte, bibliotecas, lenguaje interpretado, etc., que ayuda a desarrollar y unir los diferentes módulos o partes de un proyecto.

Entornos de Ejecución

Un entorno de ejecución es un servicio de máquina virtual que sirve como base software para la ejecución de programas.

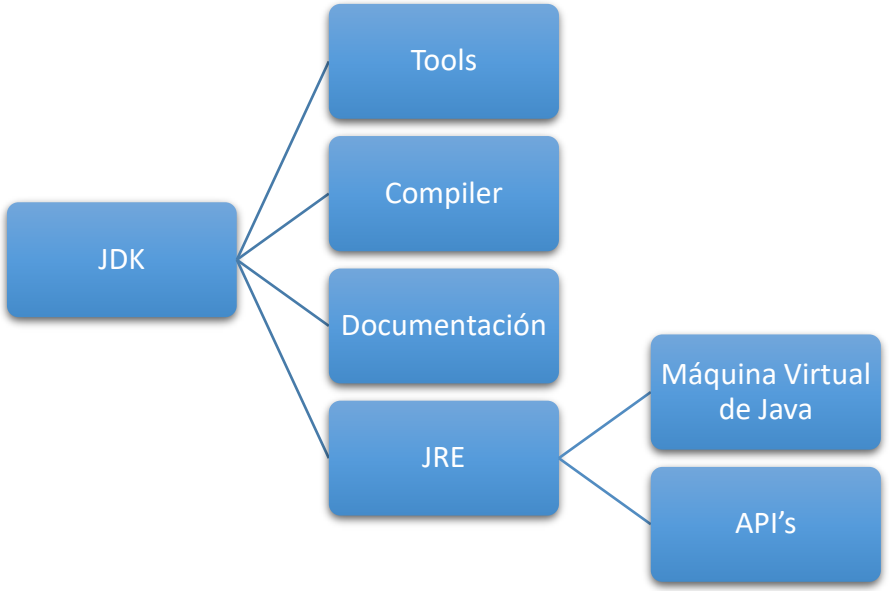
Entorno de Ejecución está formado por la máquina virtual y los API's (bibliotecas de clases estándar, necesarias para que la aplicación, escrita en algún Lenguaje de Programación pueda ser ejecutada). Estos dos componentes se suelen distribuir conjuntamente, porque necesitan ser compatibles entre sí.



Parte Práctica de UT1:

1. **Virtual Box:** Se ha utilizado VirtualBox para generar una máquina virtual de sistema con Windows 10 para pruebas.

2. Instalación del JDK



3. Compilar desde línea de Comandos.

Ut2 (I).- Entornos de Desarrollo.

Objetivos

Conocer las carácterísticas de los entornos de desarrollo.

Evaluar entornos integrados de desarrollo, analizando sus características para editar código fuente y generar ejecutables.

Instalar y configurar entornos de desarrollo.

Utilizar entornos de desarrollo para crear modelos de datos y desarrollo de programas.

Instalaremos dos tipos de entornos de desarrollo: SQL Developer, orientado al uso de bases de datos, y BlueJ y Eclipse orientados al desarrollo de programas. Además instalaremos plugins para añadir funcionalidad al entorno y trabajar con otras herramientas.

Concepto de Entorno de Desarrollo

- **IDE** son las siglas de *Integrated Development Environment*, en español **Entorno Integrado de Desarrollo.**

- Se define como una **aplicación informática** que está compuesta por un conjunto de **herramientas de programación** que van a facilitar la tarea al programador y obtener una mayor rapidez en el desarrollo de aplicaciones.

- Puede estar pensada para un lenguaje concreto o puede dar cabida a varios lenguajes.

Los entornos de desarrollo están compuestos por una serie de herramientas software de programación, necesarias para la consecución de sus objetivos. Estas herramientas son:

- Un editor de código fuente.
- Un compilador y/o un intérprete.
- Automatización de generación de herramientas.
- Un depurador.

Duda Frecuente:

Diferencia entre IDE y Framework

El **IDE** es el software que utiliza para desarrollar; por ejemplo, Eclipse es un IDE *(editor de código, depurador, herramientas de compilación ...)*
El **Framework** es un conjunto de bibliotecas y prácticas recomendadas que le ayudan reutilizar código y proporciona un conjunto de indicaciones o pautas sobre cómo desarrollar una aplicación.

Citando *Wikipedia*, un **IDE** :
es una **aplicación de software** que proporciona a los programadores informáticos instalaciones completas para el desarrollo de software. Un IDE normalmente consiste en:

- ✓ un editor de código fuente
- ✓ un compilador y / o un intérprete
- ✓ construir herramientas de automatización
- ✓ un depurador

Mientras que un **Framework** :
es una **abstracción** en la que el código de usuario puede anular o especializar selectivamente el código común que proporciona una funcionalidad genérica, lo que proporciona una funcionalidad específica. Los frameworks son un caso especial de bibliotecas de software, ya que son abstracciones reutilizables de código envuelto en una interfaz de programación de aplicaciones (API) bien definida, pero contienen algunas características distintivas clave que las separan de las bibliotecas normales.

Evolución Histórica

Tipos de entornos de desarrollo más relevantes en la actualidad.

Entorno de desarrollo	Lenguajes que soporta	Tipo de licencia
NetBeans.	C/C++, Java, JavaScript, PHP, Python.	De uso público.
Eclipse.	Ada, C/C++, Java, JavaScript, PHP.	De uso público.
Microsoft Visual Studio.	Basic, C/C++, C#.	Propietario.
C++ Builder.	C/C++.	Propietario.
JBuilder.	Java.	Propietario.

No hay unos entornos de desarrollo más importantes que otros. La elección del IDE más adecuado dependerá del lenguaje de programación que vayamos a utilizar para la codificación de las aplicaciones y del tipo de licencia con la que queramos trabajar.

Tipos de Entornos

Tipos de entornos de desarrollo libres más relevantes en la actualidad.

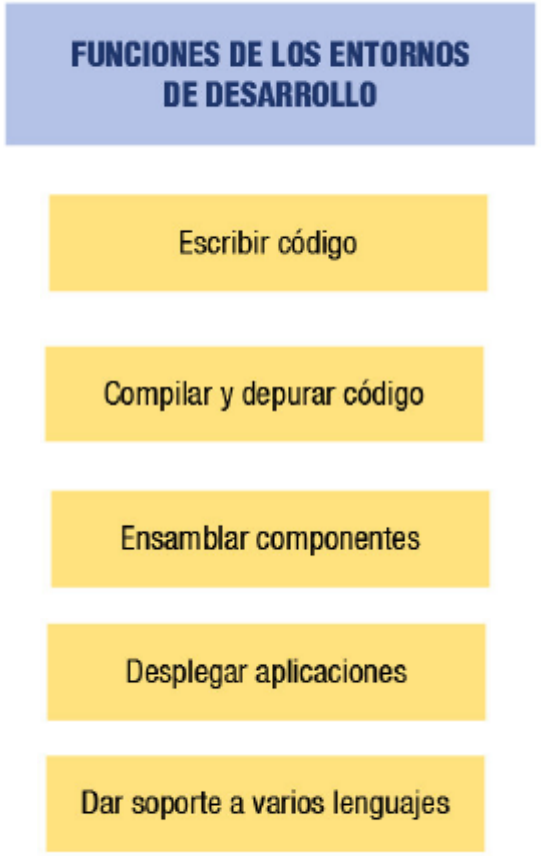
IDE	Lenguajes que soporta	Sistema Operativo
NetBeans.	C/C++, Java, JavaScript, PHP, Python.	Windows, Linux, Mac OS X.
Eclipse.	Ada, C/C++, Java, JavaScript, PHP.	Windows, Linux, Mac OS X.
Gambas.	Basic.	Linux.
Anjuta.	C/C++, Python, Javascript.	Linux.
Geany.	C/C++, Java.	Windows, Linux, Mac OS X.
GNAT Studio.	Fortran.	Windows, Linux, Mac OS X.

Tipos de entornos de desarrollo propietarios más relevantes en la actualidad.

IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio.	Basic, C/C++, C#.	Windows.
FlashBuilder.	ActionScript.	Windows, Mac OS X.
C++ Builder.	C/C++.	Windows.
Turbo C++ profesional.	C/C++.	Windows.
JBuilder.	Java.	Windows, Linux, Mac OS X.
JCreator.	Java.	Windows.
Xcode.	C/C++, Java.	Mac OS X.

El aspecto de la licencia del IDE que se elija para el desarrollo de un proyecto es una cuestión de vital importancia. En su elección prevalecerá la decisión de los supervisores del proyecto y de la dirección de la empresa.

Funciones de un Entorno de Desarrollo



- ✓ 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.

Estructura de un Entorno de Desarrollo

Editor de textos: Se 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.

Parte Práctica de UT2:

1. Eclipse:
    - Instalación.
    - Uso básico del Entorno.
    - Instalación de Plugins
    - Ejercicios de Depuración.
2. Bases de Datos ORACLE:
    - Instalación de la base de datos ORACLE
    - Manipulación de la BD desde línea de comandos SQL
    - Instalar e iniciar SQL Developer
        1) Partes del Entorno.
        2) Creación de usuarios.
        3) Introducir y ejecutar consultas SQL
        4) Actualizar Datos.
    - Ejemplo práctico.