Desarrollo de Software Informático

TEMA 1.2

Ciclo de Vida del Software.

Fases del Ciclo de Vida del Software

- **Análisis** (lo que se debe hacer): estudio de las necesidades que se habrán de satisfacer.
  - **Definición de requisitos** (funcionales y no funcionales) del sistema.
  - **Documentación de análisis**, con la especificación de requisitos del software (ERS).
- **Diseño** (como se debe hacer): elaboración de la solución que cubra los requisitos.
  - **Estructurado** (clásico).
    - **Diseño de Datos**: descripción detallada de datos a partir de los DER.
    - **Diseño de arquitectónico**: estructura modular a partir de los DFD.
    - **Diseño de interfaz**: formatos de pantalla a partir de la E/S.
    - **Diseño de componentes**: procedimientos internos.
  - **Orientado a Objetos**.
- **Codificación**: elaboración de programas en el lenguaje de programación seleccionado, resolviendo los errores de los procedimientos
- **Pruebas**: verificación y validación de los elementos desarrollados con respecto a los requisitos, resolviendo errores de análisis y diseño en los procedimientos principalmente, o también en la codificación.
  - **Pruebas de caja blanca:** sobre aspectos procedimentales (operativa interna).
  - **Pruebas de caja negra:** sobre aspectos funcionales (operativa externa).
- **Documentación**: elaboración de la documentación funcional, técnica y de usuario.
  - **Documentación del proceso**: desarrollo y mantenimiento.
  - **Documentación del producto**.
    - **Documentación (técnica) del sistema**.
    - **Documentación (guía) del usuario**.
- **Producción**: fase operativa de uso del software desarrollado, atendiéndose todas las necesidades que pudieran surgir durante toda la vida útil del producto software.
  - **Implantación** (preparativos, instalación, inicialización): software ya desarrollado puesto en marcha en un entorno real de trabajo.
  - **Explotación**: software plenamente operativo para su aprovechamiento dando el servicio para el que ha sido desarrollada.
  - **Mantenimiento** (preventivo, correctivo, adaptativo, evolutivo): fase en la que se toman las medidas necesarias para asegurar el correcto funcionamiento y el mejor rendimiento del software, en condiciones óptimas de calidad durante todo el tiempo de vida de este (fase más importante por larga y costosa, con problemas propios inherentes).
  - **Actualización**: proceso de sustitución del software por una nueva versión mejorada o adaptada a un nuevo contexto de trabajo.

Técnicas de análisis: diagrama de flujo de datos (DFD), diagrama de flujo de control (DFC), diagrama de transición de estados (DTE), diagrama Entidad / Relación (DER), diccionarios de Datos (DD).

Técnicas de definición de requisitos: entrevistas, brainstorming, prototipos, desarrollo conjunto de aplicaciones (JAD), planificación conjunta de requisitos (JRP), caso de usos (UML).

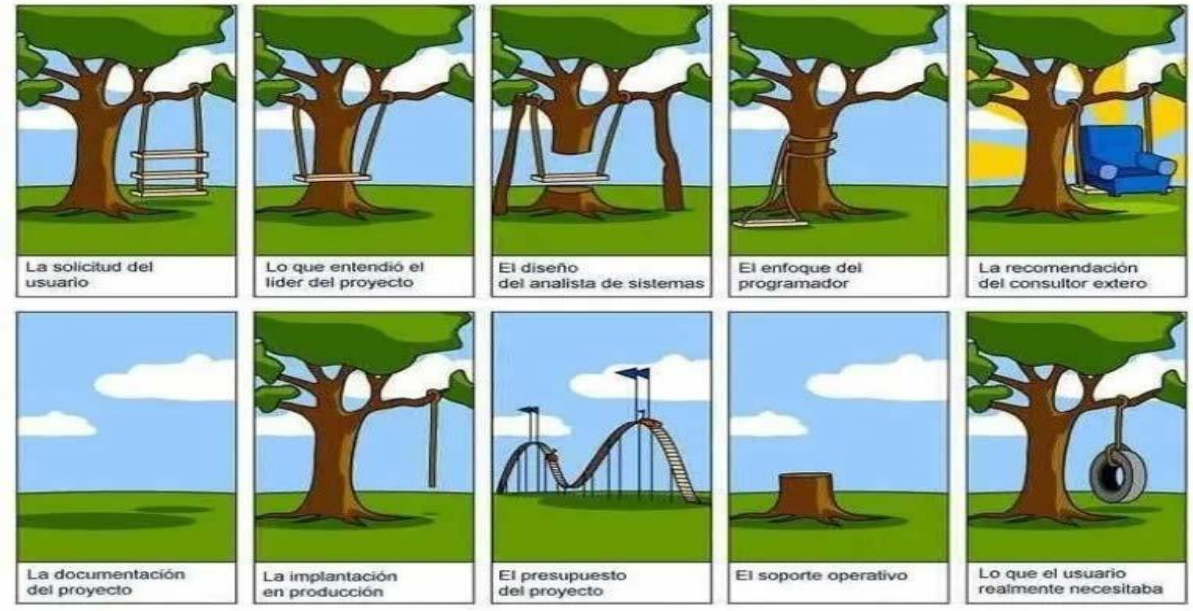
Técnicas de diseño de componentes: pseudocódigo, diagrama de flujo u ordinograma, diagrama de cajas, tablas de decisión.

El estándar **IEEE std 830-1998** define el formato de documento de especificación de requisitos del software.
https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Ciclo de Vida del Software.

Roles del Desarrollo

- **Jefe de Proyecto:** dirige todo el proyecto de desarrollo de software, velando por el cumplimiento de plazos, el mantenimiento de los costes presupuestados, y la calidad del producto final.
- **Analista funcional:** encargado de realizar la especificación de requisitos del producto software a construir.
- **Diseñador de software:** encargado de confeccionar la solución a desarrollar a partir de los requisitos especificados.
- **Arquitecto:** encargado de establecer las tecnologías y herramientas a utilizar, empleando los recursos apropiados, y supervisando que todo el proceso se lleva a cabo de la mejor forma posible.
- **Analista programador** (comúnmente llamado desarrollador)**:** encargado de organizar el trabajo de los programadores, además de participar en tareas de análisis, al tener una experiencia y visión amplias en el desarrollo de software.
- **Programador:** encargado de codificar los programas en el lenguaje fuente del software en desarrollo.



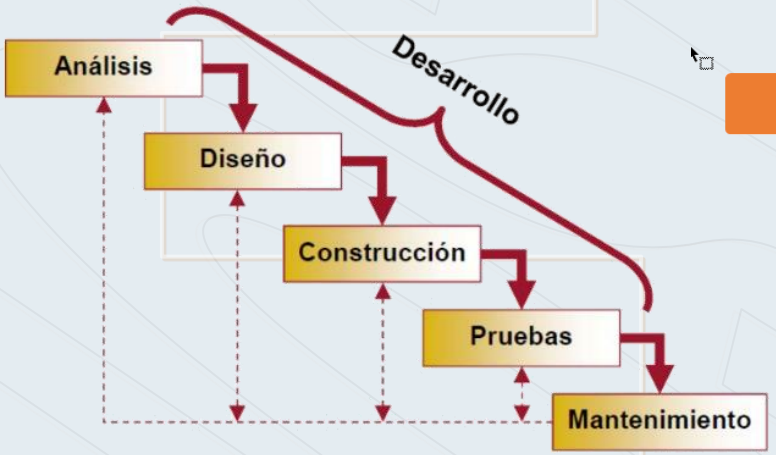
Metodologías de Desarrollo de Software.

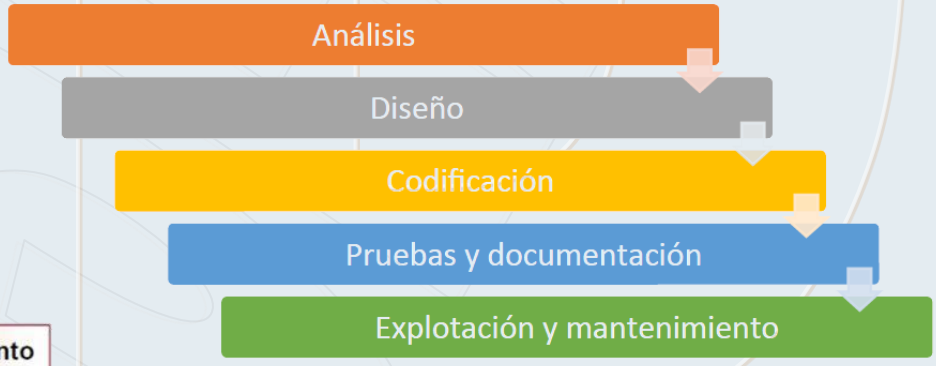
Las **Metodologías de Desarrollo de Software** son un conjunto integrado de técnicas y métodos sistemáticos, predecibles y repetibles, que permiten abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo de software, a fin de mejorar la productividad en el desarrollo y la calidad del software, según distintos modelos.

METRICA es una metodología de desarrollo de sistemas de información promovida por el estado español para la sistematización de las actividades que abarcan el ciclo de vida de proyectos software en el ámbito de las administraciones públicas, con varias versiones en su evolución (Métrica v1 - 1989, Métrica v2 - 1993, Métrica v2.1 - 1995, Métrica v3 - 2001).

Clasificación

El **Modelo en Cascada (clásico)** es un modelo en el que las fases se suceden en orden secuencial, pasando a la siguiente cuando ha finalizado la anterior, y volviéndose atrás solo si se detecta algún error, hasta obtener el producto final, con el riesgo de no mostrar nada al cliente hasta la fase de pruebas. Se suele usar en grandes proyectos, cuando el volumen de personas involucradas en los equipos es elevado y el contacto con el cliente está limitado tanto en tiempo como en personas.

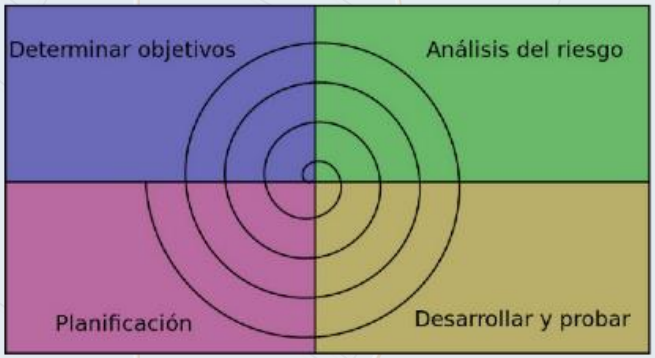




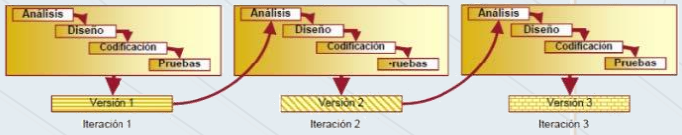
Metodologías de Desarrollo de Software.

Clasificación

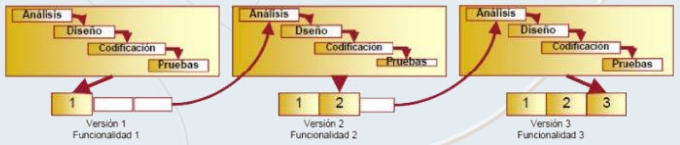
El **Modelo en Espiral (evolutivo**) es un modelo en el que las fases se suceden en orden secuencial repetidamente, en varias pasadas recurrentes de refinamiento, obteniéndose varias aproximaciones del producto hasta alcanzar el resultado final, con la ventaja de contar con diversas versiones del producto revisables por el cliente. El recorrido de fases puede ser iterativo (se van entregando versiones de la totalidad del producto) o incremental (se van entregando versiones de partes del producto).



Iterativo



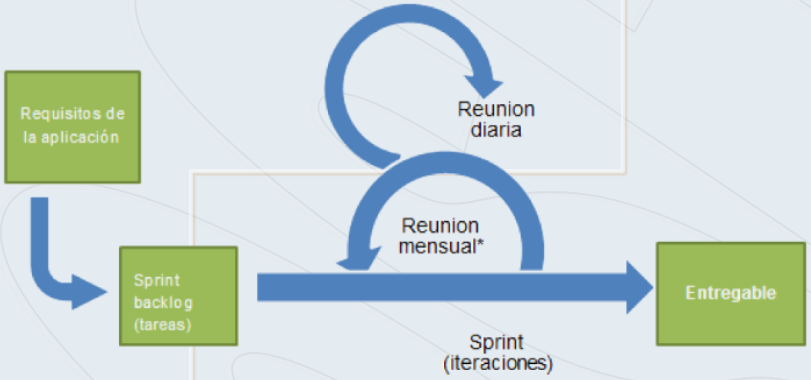
Incremental

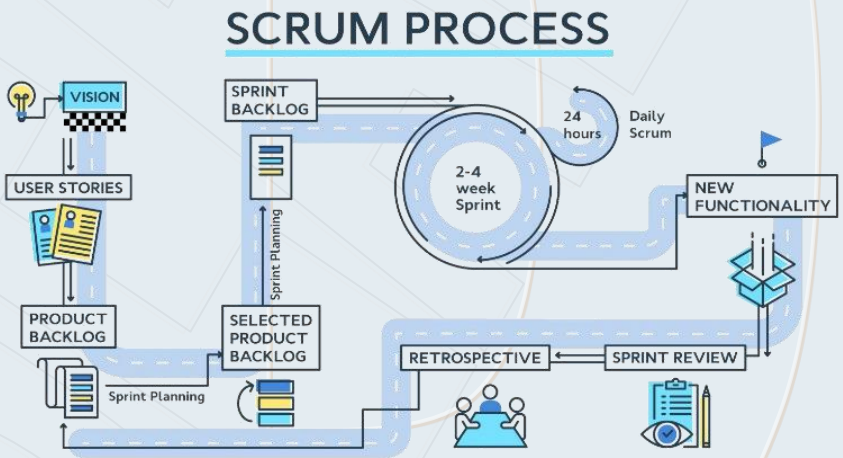


Metodologías de Desarrollo de Software.

Clasificación

El **Modelo ágil (a partir de 2001)** es un modelo basado en el modelo en espiral de tipo incremental, en ciclos muy cortos, y poniendo especial énfasis en el compromiso de los miembros integrantes de los equipos de trabajo, las comunicaciones cara a cara en lugar de la comunicación escrita, y la implicación y colaboración con el cliente. Este modelo busca minimizar el riesgo de no mostrar nada al cliente hasta una fase muy avanzada del desarrollo mediante un esquema incremental y el contacto continuo con el cliente. El enfoque de las metodologías ágiles resulta de gran efectividad en proyectos pequeños/medianos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.





SCRUM es uno de los modelos de desarrollo ágil de software más usado actualmente, que se ha extendido a otras industrias. Se basa en pequeños incrementos en iteraciones de 2 semanas. Al inicio de cada iteración el equipo decide que va a desarrollar en función de las prioridades del cliente, y durante 2 semanas trabaja en su análisis, diseño y codificación. Al final de las dos semanas se prueba y valida con el usuario todos los requisitos finalizados para dar por terminada la iteración y continuar con la siguiente. Este modelo requiere un equipo con cierta experiencia y conocimiento del sistema, ya que los ciclos cortos y sensación de dispersión o baja eficiencia en los trabajos puede ocasionar malestar en el cliente.

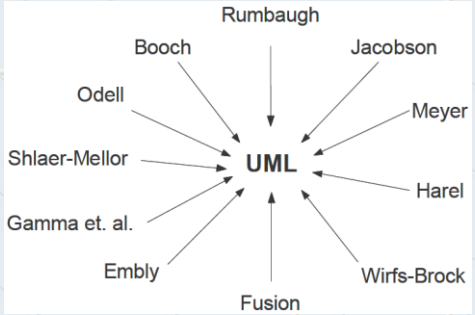
Lenguaje Unificado de Modelado.

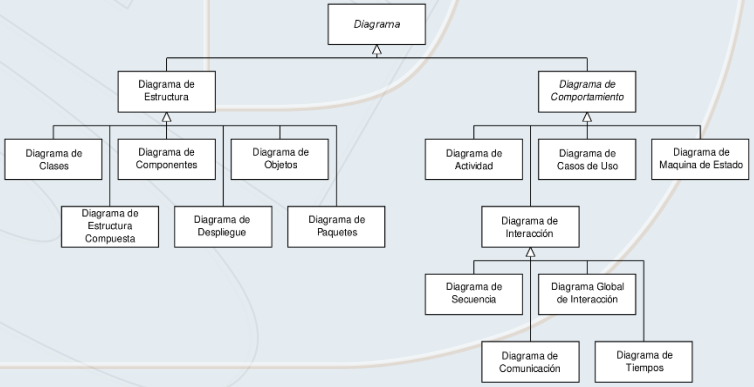
El **Lenguaje Unificado de Modelado** o **Unified Modeling Language** (**UML**) es un lenguaje gráfico de modelado de sistemas de software (el más conocido y utilizado en la actualidad, aunando diversas notaciones previas), estandarizado (ISO/IEC 19501:2005) y respaldado por el Object Management Group (OMG). Es utilizado por los desarrolladores de software para especificar, construir, visualizar y documentar sistemas software orientados a objeto. Proporciona distintos tipos de diagrama para representar, desde distintos puntos de vista, diversos aspectos del proceso de desarrollo de un proyecto software, referentes a la estructura (organización del sistema de software) y comportamiento (dinámica del sistema de software), incluyendo las interacciones entre los diferentes componentes.

UML puede llegar a resultar excesivamente complejo ("el 80% de los problemas puede modelarse usando alrededor del 20% de UML"), y no exento de ambigüedad, sin que además llegue a integrar la totalidad de las técnicas utilizadas en el desarrollo del software (no se define un método de diseño de interfaces de usuario por ejemplo).

El **Object Management Group** (**OMG**) es un consorcio sin ánimo de lucro dedicado al cuidado y promoción del uso de las tecnologías orientadas a objetos, mediante el establecimiento de diversos estándares, a través de guías y especificaciones.

Un modelo es una simplificación de la realidad, con el que se pretende capturar las partes esenciales de un sistema. Representa "los planos" del sistema con independencia del procedimiento de implementación, y con determinado grado de abstracción, con más o menos detalle en función de los elementos que sean relevantes en cada momento. Los modelos visuales utilizan notación gráfica para simplificar la complejidad de los sistemas a analizar o diseñar.





Herramientas de Desarrollo de Software.

Las **herramientas de desarrollo** son herramientas CASE (Ingeniería del Software Asistida por Ordenador o Computer Aided Software Engineering), encaminadas a automatizar el desarrollo de proyectos software en las distintas etapas del ciclo de vida de este, con el objetivo principal de aumentar la eficiencia y productividad, reduciendo costes económicos y temporales.

Clasificación

- **Upper CASE** (fases de análisis y diseño).
  - **Análisis** (reconocimiento de requisitos, detección de inconsistencias, redundancias de datos, errores e inexactitudes, ...): CaseComplete, VisibleAnalyst.
  - **Diseño** (construcción de bocetos, simulación de procesos, interfaces de usuario, ...): Balsamiq, Mockup Builder.
- **Lower CASE** (fases de implementación, pruebas, mantenimiento).
  - **Entornos Integrados de Desarrollo** orientados a **programación** (edición, traducción, depuración, ...): Eclipse, Netbeans, BlueJ, IntelliJ; **bases de datos** (esquemas, consultas, ...): SQLDeveloper, SQLNavigator, Erwin; **web** (textos, gráficos, controles, formularios, ...): Kompozer, Brackets.
  - **Documentación** (aspectos funcionales y orgánicos, guías de usuario, manuales técnicos, ...): Dosygen, DrExplain.
  - **Calidad** (cumplimiento de estándares, monitorización de procesos, inspección de código, realización de pruebas, control de cambios, ...): Jmeter, Selenium, SonarQube, Git, Subversion.
- **Integrated CASE** (todas las fases).
  - **Modelado** (procesos de negocio, de aplicación, de datos): SAP PowerDesigner, Eclipse Process Framework.
  - **Proyectos** (programación temporal, planificación de recursos, estimación de costes, ...): Microsoft Project, OpenProject.
  - **Diagramas** (mapas, flujos de datos, de control, ...): Dia, Draw.io, Visio, Gliffy, FlowChart Maker, Lucid Chart.
  - **UML** (modelado de sistemas, administración de proyectos, confección de prototipos, generación de código, ingeniería inversa, ...): Visual Paradigm, ArgoUML.

Entornos Integrados de Desarrollo.

Un **Entorno Integrado de Desarrollo** o **Integrated Development Environment** (**IDE**) se puede definir como un software cuyo objetivo es el de asistir al programador en la tarea de codificación de programas, por medio de múltiples herramientas y funciones orientadas a tal fin, con una amplia diversidad de versiones y variantes para distintos lenguajes de programación y plataformas, tanto de escritorio (ECLIPSE, NETBEANS, BLUEJ, INTELLIJ, VISUAL STUDIO, SQL NAVIGATOR, ANDROID STUDIO, …), como en línea (COMPILEJAVA, CODECHEF, CODING GROUND FOR DEVELOPERS, …).

Los entornos integrados de desarrollo suelen soportar varios lenguajes de programación, y suelen disponer de versiones para distintas plataformas, permitiendo así la traducción de código de diversos lenguajes para diversas plataformas.

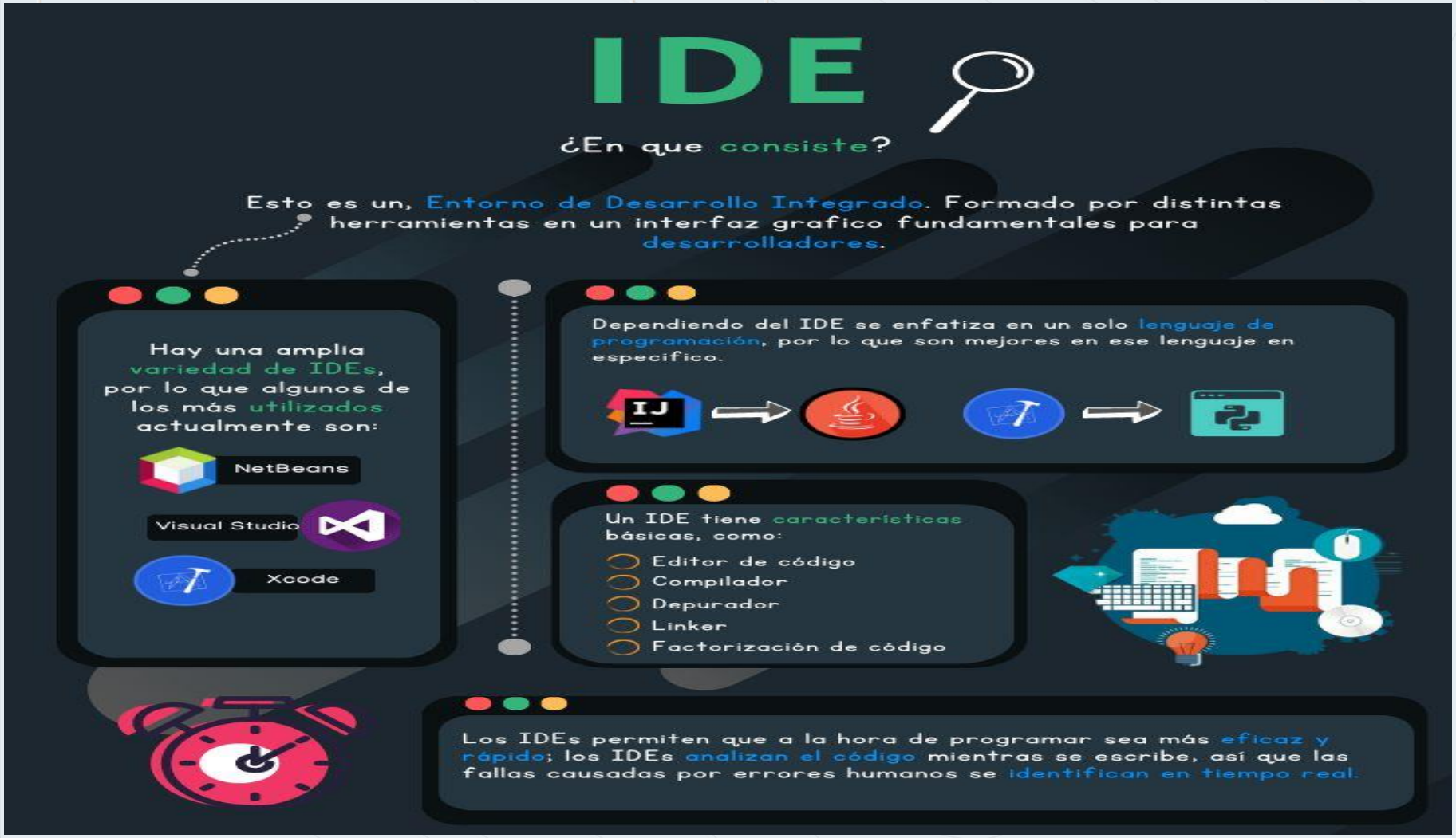
Cada entorno integrado de desarrollo incorpora de facto, o mediante módulos de extensiones o plugins aparte, características y funcionalidades específicas que lo definen, incluyendo todos por regla general una serie de componentes comunes para la labor de programar, que abarcan las distintas fases de elaboración de programas (editor de textos, compilador, intérprete, depurador, cliente). Usualmente, también permiten insertar marcos de trabajo o frameworks, que proveen un conjunto de funcionalidades específicas cuyo objetivo es agilizar el proceso de desarrollo dentro de un contexto particular. El conjunto de funcionalidades definen un ambiente de trabajo enfocado en un aspecto concreto de desarrollo (aplicaciones gráficas, aplicaciones web, aplicaciones móviles, bases de datos, etc).

Entornos Integrados de Desarrollo.

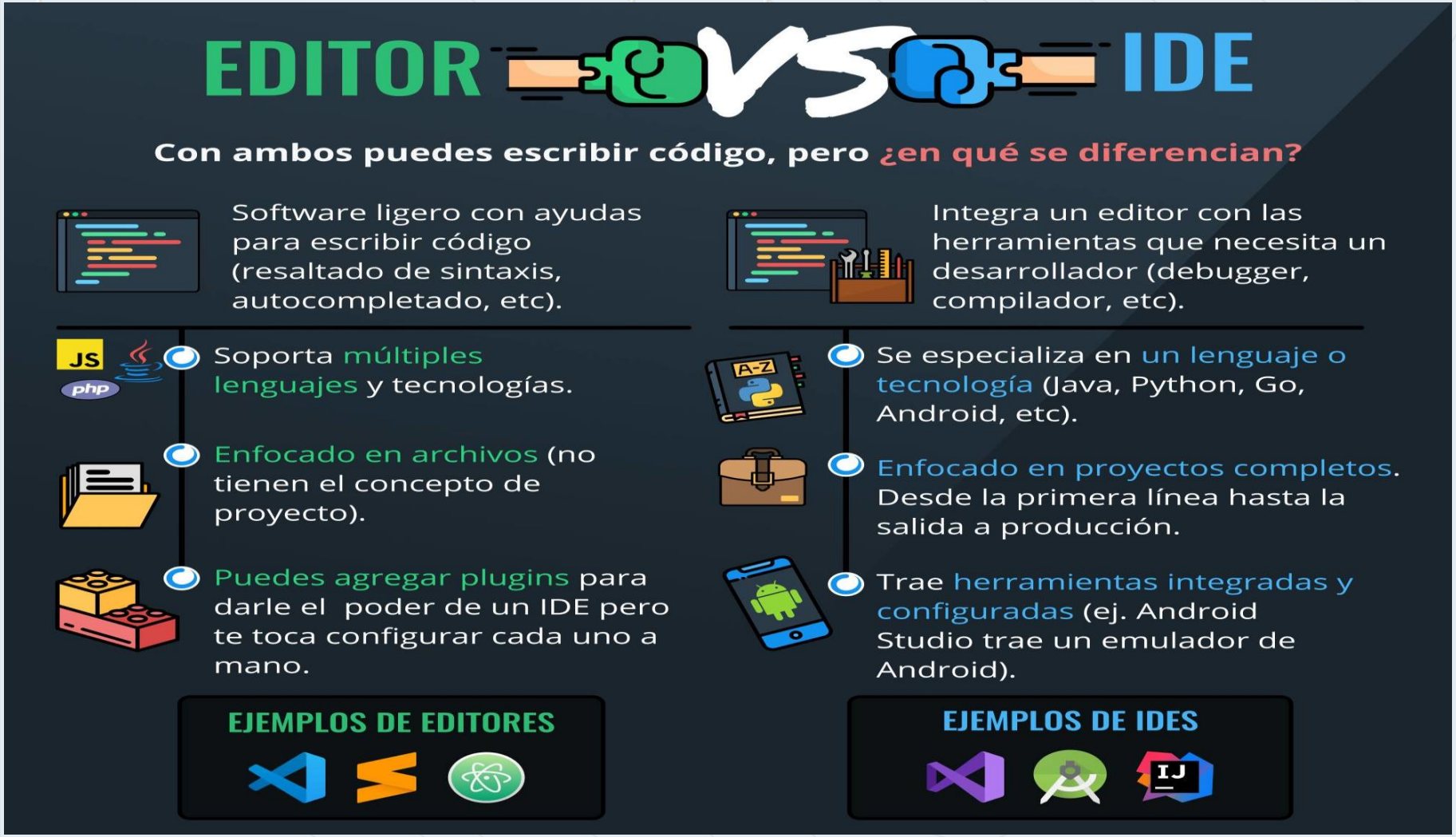
Funciones

- **Personalización del entorno** según las preferencias de configuración de usuario (añadir y modificar las barras de herramientas, definir comandos personalizados y atajos de teclado para cada una de ellas, ...).
- **Gestión de proyectos**.
  - Organización de ficheros en carpetas (fuente, objeto, ...).
  - Mantenimiento de dependencias y enlaces entre secciones de código.
  - Acceso a la documentación y ayuda contextual sobre los lenguajes de programación soportados.
- **Edición de código**.
  - Coloreado o resaltado sintáctico para facilitar la identificación de los elementos clave del lenguaje de programación.
  - Autocompletado de código sensible al contexto (propiedades y métodos) o combinaciones de tecla (fragmentos de código), ...
  - Indentación automática anidando los distintos elementos y estructuras de programación.
  - Expansión / Contracción de fragmentos de código para facilitar su manejo y visualización.
  - Detección de errores de sintaxis en tiempo real.
- **Manejo de Código**.
  - Asistentes y utilidades de generación de código.
  - Ejecución virtual de código sin despliegue real de las aplicaciones.
  - Compilación en un paso de proyectos completos o solo secciones modificadas y conectadas.
  - Refactorización de código mediante reestructuración automática del código fuente para facilitar su legibilidad o eficiencia.
  - Realización de pruebas para comprobación del correcto funcionamiento de los programas.
  - Control de versiones y gestión de cambios en archivos de código compartido por los miembros de un equipo de trabajo.
- **Depuración de código**.
  - Ejecución paso a paso de instrucciones o bloques de instrucciones.
  - Inclusión de puntos de ruptura o parada en la ejecución.
  - Inspección y seguimiento, e incluso modificación durante la ejecución en algunos IDEs, de variables y estructuras de datos.
  - Identificación y localización de códigos y mensajes de error en los programas.
- **Gestión de módulos** (extensiones o plugins), como por ejemplo:
  - Constructor de interfaces gráficas de usuario (GUI).
  - Sistema de acceso a bases de datos y gestión de archivos.

Entornos Integrados de Desarrollo.

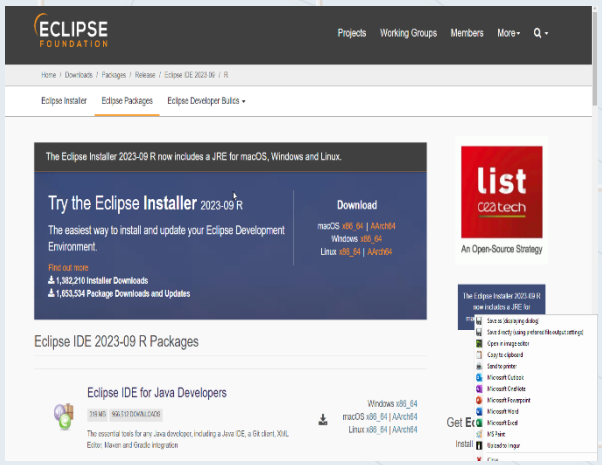


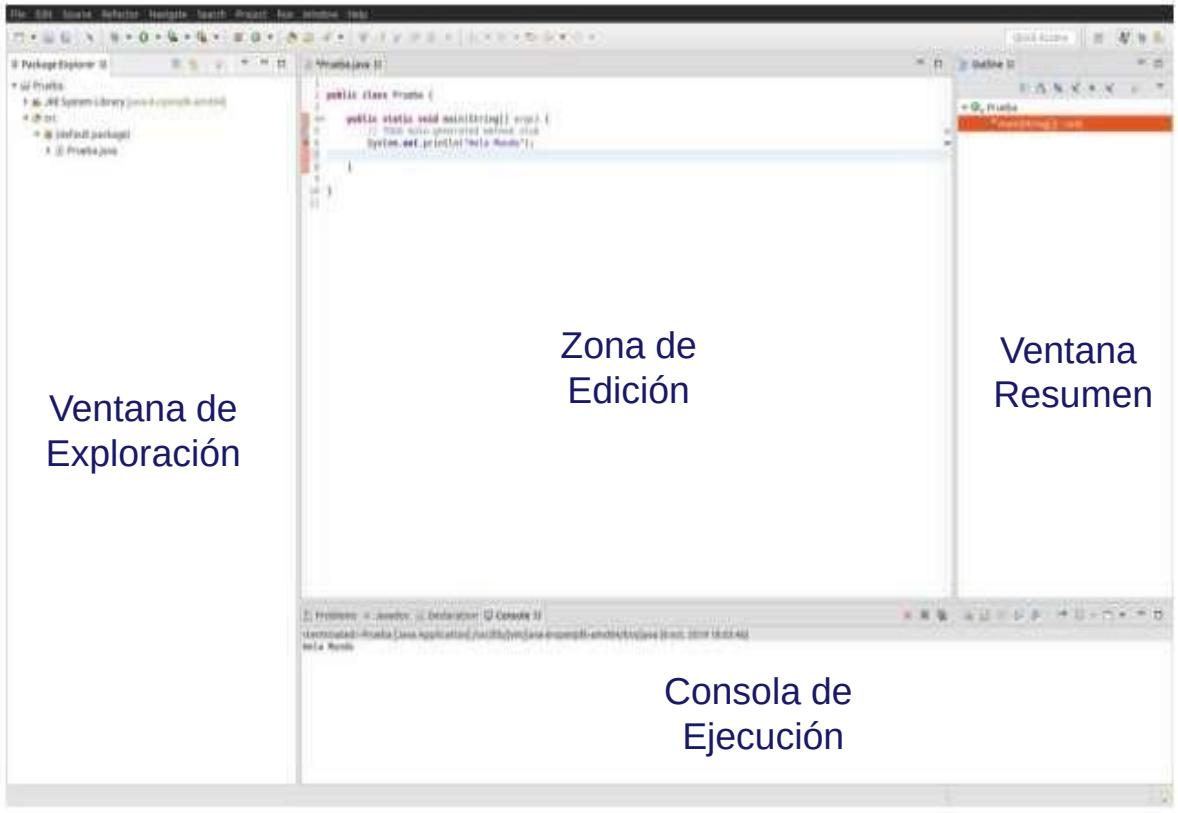
Entornos Integrados de Desarrollo.



Eclipse.

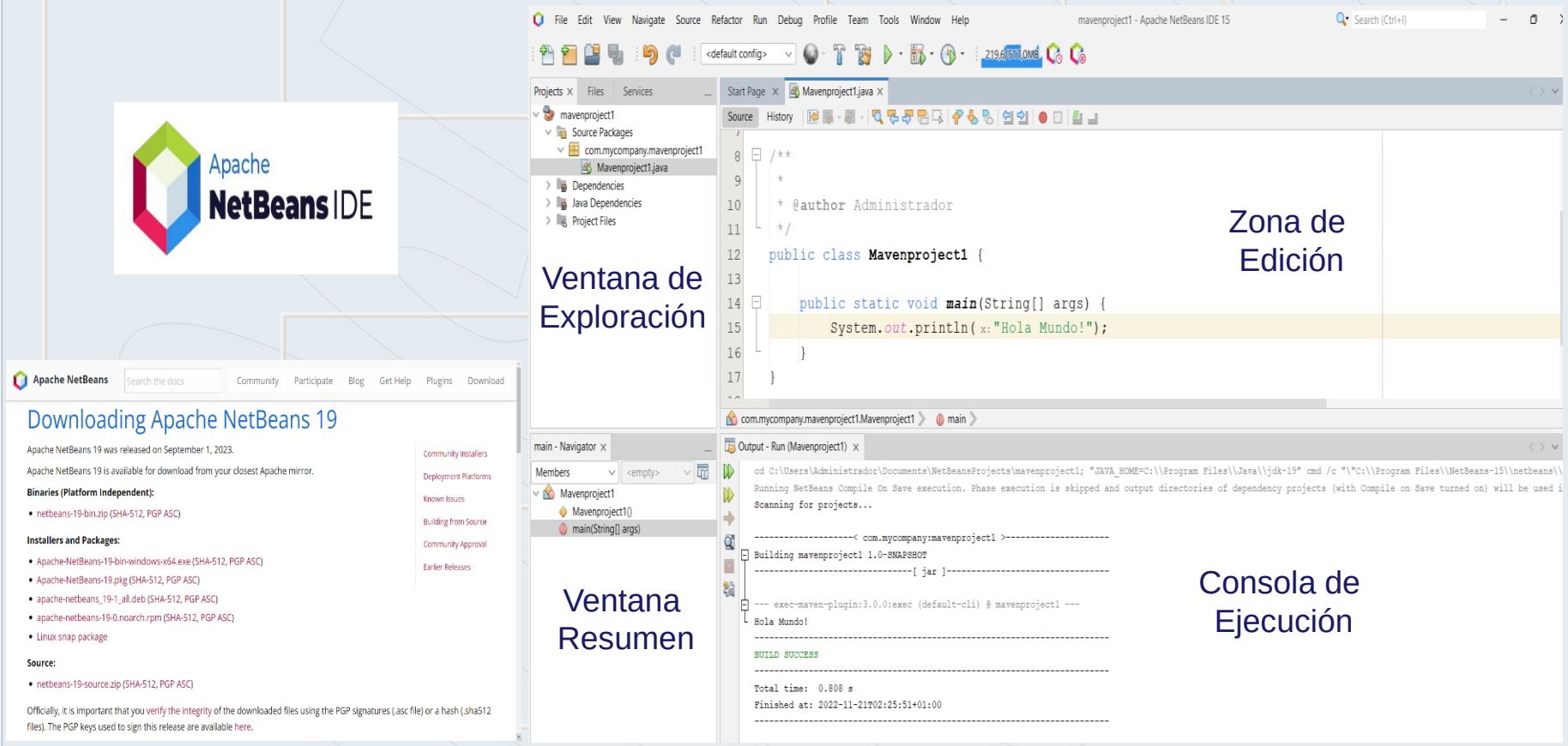
Eclipse es, según la definición del proyecto Eclipse, “una especie de herramienta universal- un IDE abierto y extensible para todo y nada en particular”. Es un entorno de código abierto bajo licencia pública, de uso ampliamente extendido para el desarrollo en Lenguaje Java, soportado por una comunidad independiente, sin ánimo de lucro (Eclipse Foundation) encargada de mantener el proyecto y su ecosistema de productos y servicios complementarios.





NetBeans.

NetBeans es un entorno de desarrollo integrado libre y gratuito sin restricciones de uso, principalmente pensado para el lenguaje de programación Java, aunque también admite otros lenguajes de programación, con un número importante de módulos para extenderlo. Es un proyecto de código abierto iniciado por Sun Microsystems, y continuado en la actualidad por Oracle.



Referencias.

¿Qué es un IDE de Programación? Significado de Entorno de Desarrollo Integrado

https://www.youtube.com/watch?app=desktop&v=_WKWpJEv9UY

Herramientas de desarrollo

https://www.tutorialspoint.com/es/software_engineering/case_tools_overview.htm