

<b>1. Historia y contexto de IntelliJ IDEA - Punto 1.....</b>	<b>2</b>
• 1.1. Origen:.....	2
• 1.2. Lenguaje y Tecnologías:.....	2
• 1.3. Evolución y Hitos:.....	2
<b>2. Instalación en una máquina virtual - Punto 2.....</b>	<b>3</b>
<b>3. Partes y usos de IntelliJ IDEA - Punto 3.....</b>	<b>4</b>
• 3.1. Editor de código:.....	4
• 3.1.1. Compilador / Intérprete:.....	4
• 3.1.2. Depurador / Debugger:.....	4
• 3.1.3. Control de versiones:.....	4
• 3.1.4. Refactorización:.....	4
• 3.1.5. Plugins:.....	5
<b>4. Ejemplo práctico de código en IntelliJ - Punto 4.....</b>	<b>6</b>
• 4.1. Foto del código (heladería):.....	6
<b>5. Valoración personal - Punto 5.....</b>	<b>7</b>
5.1. ¿Cuándo se recomendaría su uso?.....	7
5.1.1.....	7
• 5.1.2. Desarrollo en JAVA y KOTLIN:.....	7
• 5.1.3. Proyectos Empresariales:.....	8
• 5.1.4. Necesidades de refactorización:.....	8
• 5.1.5. Desarrollo WEB:.....	8
• 5.1.6. Uso de control de versiones:.....	8
• 5.1.7. Productividad:.....	8
• 5.1.8. Soporte y actualización:.....	8

# 1. Historia y contexto de IntelliJ IDEA - Punto 1

## ● 1.1. Origen:

- IntelliJ IDEA fue creado por la empresa **JetBrains**, fundada en el año 2000 en Praga (República Checa) por tres desarrolladores: Sergey Dmitriyev, Valentin Kipiatkov y Eugene Beyaev
- Su lema desde el principio fue claro: “Desarrolladores felices escriben mejor código”
- Mientras otros IDEs de Java como Eclipse eran algo toscos y lentos, IntelliJ apareció como una revolución silenciosa, con una interfaz más fluida y funciones inteligentes que ayudan al programador en lugar de entorpecer

## + Curiosidad:

- El nombre “IntelliJ” viene de “Intelligent Java”, porque desde sus primeras versiones el objetivo era crear un entorno que entendiera tu código: sugerencias automáticas, refactorización en tiempo real, y detección de errores sin tener que compilar. En resumen fue el primer IDE que parecía leer la mente de los trabajadores

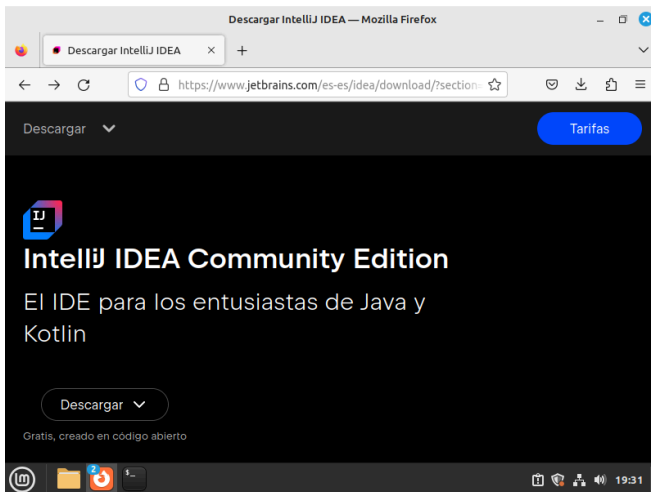
## ● 1.2. Lenguaje y Tecnologías:

- Aunque nació en principio solo para Java, hoy IntelliJ IDEA soporta Kotlin, Groovy, Scala, SQL, HTML, CSS, JavaScript, Python, PHP, e incluso Android (de hecho Android Studio está basado en IntelliJ)
- JetBrains lo diseñó con una arquitectura modular, así que cualquier lenguaje puede añadirse mediante plugins

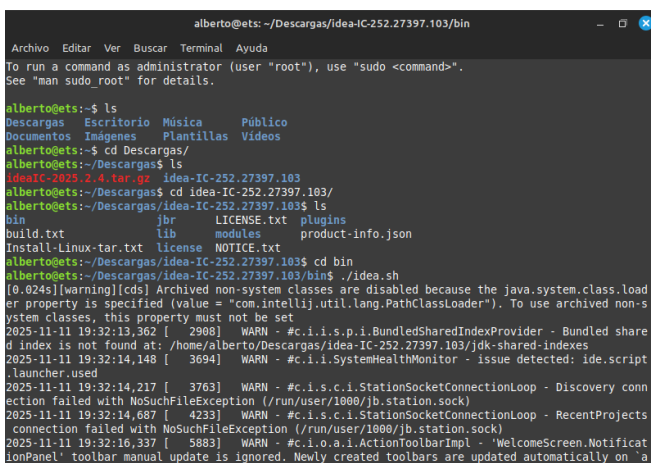
## ● 1.3. Evolución y Hitos:

- **2001:** Primera versión pública. Se ganó fama por su autocompletado inteligente (cuando eso no era común)
- **2009:** JetBrains lanza la versión Community, gratuita y de código abierto (basada en Apache 2.0)
- **2011:** Google selecciona IntelliJ IDEA como base oficial para su IDE Android Studio
- **2016:** JetBrains introduce su propio lenguaje de programación, Kotlin, totalmente integrado en IntelliJ (y hoy oficial en Android)
- **2020 en Adelante:** Soporta inteligencia artificial, integración con GitHub Copilot, y funciones predictivas cada vez más “autónomas”

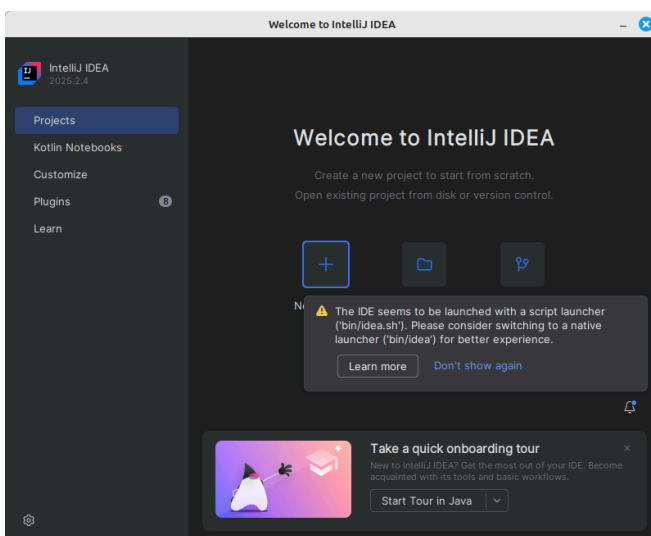
## 2. Instalación en una máquina virtual - Punto 2



- Buscamos la página oficial de IntelliJ y descargamos la versión gratuita (open source)



- Posteriormente lo que haremos será ir a la terminal de nuestro equipo.
- Ahí nos dirigiremos a la carpeta donde hayamos descargado el archivo. Dentro de esta carpeta haremos lo mismo para entrar dentro de la carpeta del archivo.
- Ahora que estamos dentro de la carpeta lo que haremos será instalar el programa en nuestro sistema.



- Una vez ya tengamos el programa instalado solo faltaria iniciarlo y empezar a usarlo para las actividades que queramos realizar.

## 3. Partes y usos de IntelliJ IDEA - Punto 3

- **3.1. Editor de código:**

- Podríamos destacar las siguientes secciones:

- **3.1.1. Compilador / Intérprete:**

- IntelliJ no tiene un compilador de por sí integrado en el software. Este usa otros compiladores ya existentes, principalmente el compilador estándar de Java llamado “*javac*”. Además, no sólo se puede utilizar este sino que IntelliJ te permite usar compiladores de más lenguajes como Kotlin, Scala, Groovy, etc.
- Una de las mayores fortalezas del compilador que usa IntelliJ IDEA es su compilación incremental. Esto significa que cada vez que se realice un cambio, en vez de recompilar todo el código desde cero, IntelliJ solo compila los archivos modificados.

- **3.1.2. Depurador / Debugger:**

- Este está completamente integrado dentro de IntelliJ IDEA. Permite la ejecución de líneas concretas de código, con puntos de interrupción de distintos tipos:
  - **Condicionales:**
    - Permiten que el código se detenga si se cumple una condición booleana.
  - **Dependientes:**
    - Esto depende de un primer checkpoint. Si el primero no se completa el dependiente no llegará a activarse.
  - **De excepción:**
    - Permiten detener la ejecución de un programa en el punto exacto donde se genere una excepción (error). Esto lo hace sin necesidad de ir línea a línea mirando donde podemos encontrar un error.

- **3.1.3. Control de versiones:**

- El control de versiones o VCS de IntelliJ IDEA soporta la integración nativa de sistemas populares como Git o Subversion.

- **3.1.4. Refactorización:**

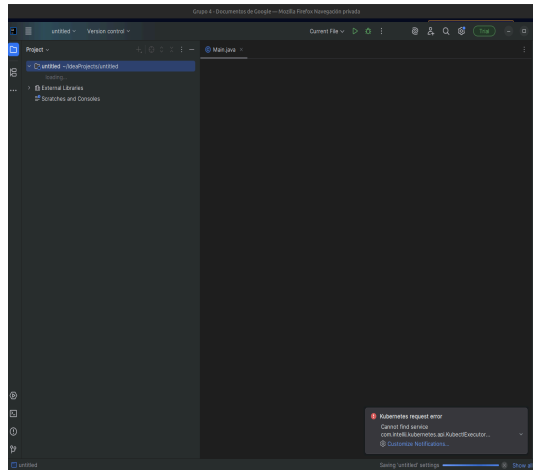
- Las Herramientas de refactorización dentro de IntelliJ IDEA permiten modificar la estructura del código de forma segura y eficiente.
- Podemos renombrar elementos de forma segura. Extraer métodos para simplificar el código. Eliminar código muerto o no utilizado.

- **3.1.5. Plugins:**

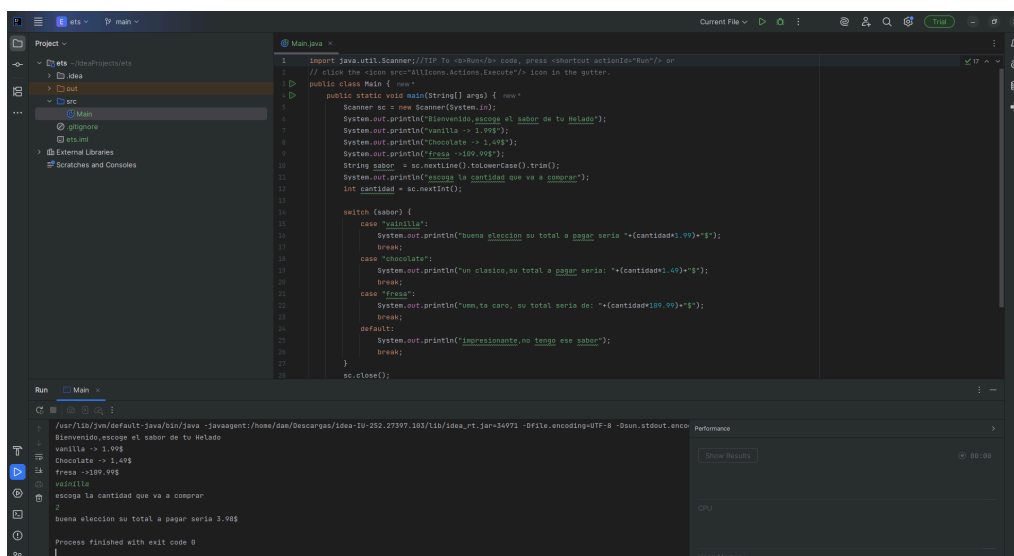
- IntelliJ IDEA tiene una amplia variedad de plugins que podemos utilizar. Los para mencionar algunos los podemos dividir en algunas secciones principales como:
    - **De productividad:**
      - Git Toolbox. Información adicional sobre Git.
      - SonarLint. Analiza la calidad del código.
    - **De desarrollo:**
      - Cucumber for Java/Gherkin. Esenciales para los proyectos que usen metodologías de Desarrollo guiado por comportamiento.
      - .env Files support. Permite reconocer correctamente los archivos .env dentro del IDE
    - **De temas:**
      - Atom material Icons. Reemplaza los iconos del software para hacerlos más coloridos, visibles y entendibles.
-

## 4. Ejemplo práctico de código en IntelliJ - Punto 4

- interfaz gráfica de IntelliJ IDEA:



- 4.1. Foto del código (heladería):



## 5. Valoración personal - Punto 5

IDE	Ventajas de IntelliJ IDEA	Desventajas de IntelliJ IDEA
NetBeans	<ul style="list-style-type: none"><li>-Excelente soporte para Java y Kotlin.</li><li>-Integración sólida de control de versiones.</li><li>-Potente refactorización y sugerencias de código.</li><li>-Buenas herramientas para desarrollo web.</li></ul>	<ul style="list-style-type: none"><li>-Puede ser más lento y pesado que NetBeans.</li><li>-Algunos usuarios prefieren la simplicidad de Netbeans.</li></ul>
Visual Studio Code	<ul style="list-style-type: none"><li>-Visual Carece de algunas características avanzadas de IntelliJ IDEA como la refactorización profunda.</li><li>- No requiere configuración adicional para ciertos lenguajes.</li></ul>	<ul style="list-style-type: none"><li>-Gran número de extensiones disponibles que facilitan el trabajo.</li><li>-Ligero y rápido.</li><li>-Soporte para distintos lenguajes de programación debido a las extensiones variadas.</li><li>-Integración con sistema de control de versiones</li></ul>
Eclipse	<ul style="list-style-type: none"><li>-Es más intuitiva para nuevos usuarios.</li><li>-La interfaz de Eclipse se considera a veces anticuada.</li></ul>	<ul style="list-style-type: none"><li>-Muy flexible y personalizable</li><li>-Buena plataforma para el desarrollo de JAVA.</li><li>-Muchas herramientas y plugins adicionales.</li></ul>
Android Studio	<ul style="list-style-type: none"><li>-Es mucho más adecuado para otros tipos de lenguajes o tipos de aplicaciones.</li><li>-Es más ligero y adaptable para equipos con recursos limitados.</li></ul>	<ul style="list-style-type: none"><li>-IDE oficial para el desarrollo de Android.</li><li>-Muy buena herramienta para crear interfaces de usuarios.</li><li>-Integración optimizada con emuladores y dispositivos.</li></ul>

### 5.1. ¿Cuándo se recomendaría su uso?

#### 5.1.1.

- **5.1.2. Desarrollo en JAVA y KOTLIN:**

- Es el IDE más completo y robusto para desarrollar en JAVA, ofreciendo herramientas avanzadas y soporte extensivo.

- Ideal si estás desarrollando aplicaciones en KOTLIN ya que está diseñado para aprovechar al máximo este lenguaje.

- **5.1.3. Proyectos Empresariales:**

- Aplicaciones a gran escala: Excelente para proyectos grandes que requieren una gestión de código sofisticada y herramientas de refactorización.
- integración con Frameworks: Soporte para Frameworks populares como Spring y Hibernate facilita el desarrollo de aplicaciones empresariales.

- **5.1.4. Necesidades de refactorización:**

- Código mantenible: Las herramientas de refactorización de IntelliJ IDEA te ayudan a mantener el código limpio y escalable.

- **5.1.5. Desarrollo WEB:**

- Aplicaciones WEB modernas: Para aplicaciones que utilizan tecnologías como Spring MVC o Java EE
- Soporte para JavaScript y otros lenguajes: Aunque su enfoque está en JAVA, IntelliJ IDEA ofrece buen soporte para lenguajes como JavaScript, HTML y CSS.

- **5.1.6. Uso de control de versiones:**

- Integración sencilla: Si estás trabajando en un equipo grande y necesitas funcionalidades robustas para control de versiones, IntelliJ IDEA facilita la integración con Git o SVN.

- **5.1.7. Productividad:**

- Características de Inteligencia artificial: Auto-completación inteligente y sugerencias contextuales.
- Plugins y herramientas integradas: Tener todo en un mismo IDE reduce la necesidad de cambiar entre diferentes herramientas y mejora el flujo de trabajo.

- **5.1.8. Soporte y actualización:**

- Actualizaciones frecuentes: Se mantiene actualizado con las últimas tendencias y prácticas de desarrollo.