Programación II – ALGORTIRMOS Y ESTRUCTURAS DE DATOS II

**Docente:** Esp. Lic. PEREZ, Nicolás Ignacio

[nicoperez@uade.edu.ar](mailto:nicoperez@uade.edu.ar)

Repositorio de la materia: <https://github.com/NicolasPerezUNLaSMN/PROG_II_UADE_JAVA>

# Facultad de Ingeniería y Ciencias Exactas

Universidad Argentina de la Empresa



**Repaso de Programación I y Transición a Java**

Los lenguajes de programación permiten a los desarrolladores escribir instrucciones que una computadora ejecuta. Se pueden clasificar según su nivel de abstracción y su método de ejecución.

### **Clasificación según el Nivel de Abstracción**

**Lenguajes de Bajo Nivel:**

* Cercanos al hardware, dependen de la arquitectura del procesador.
* Más rápidos, pero más difíciles de programar.
* Ejemplos: Lenguaje de máquina (binario), Ensamblador.

**Lenguajes de Alto Nivel:**

* Más cercanos al lenguaje humano, permiten mayor abstracción.
* Código más legible y portable.
* Ejemplos: C, Java, Python, JavaScript.

**Lenguajes de Nivel Intermedio:**

* Combinan características de bajo y alto nivel.
* Ejemplo: C (permite manipular memoria como bajo nivel, pero con estructuras de alto nivel).



### **Clasificación según el Método de Ejecución**

**Lenguajes Compilados:**

* Se traducen completamente a código máquina antes de ejecutarse.
* Mayor velocidad y eficiencia.
* Ejemplos: C, C++, Rust.

**Lenguajes Interpretados:**

* Se traducen y ejecutan línea por línea en tiempo real.
* Más flexibles pero generalmente más lentos.
* Ejemplos: Python, JavaScript, PHP.

**Lenguajes de Ejecución Intermedia (Bytecode + Máquina Virtual):**

* Se compilan a un código intermedio ejecutado por una Máquina Virtual.
* Buscan balancear rendimiento y portabilidad.
* Ejemplos: Java (JVM), C# (CLR), Python (Pyc).



### **Comparación: Python vs Java**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Python** | **Java** |
| Tipo | Interpretado | Intermedio (Bytecode + JVM) |
| Nivel | Alto nivel | Alto nivel |
| Paradigma | Multiparadigma | Principalmente orientado a objetos |
| Tipado | Dinámico y débilmente tipado | Estático y fuertemente tipado |
| Velocidad | Más lento | Más rápido |
| Uso | Scripts, IA, web, ciencia de datos | Aplicaciones empresariales, móviles, sistemas grandes |

### **Clasificación por “lados”**

Otra forma de clasificar los lenguajes de programación es según su uso en el desarrollo de software, dividiéndolos en **frontend** y **backend**:

### **Frontend (Lado del Cliente)**

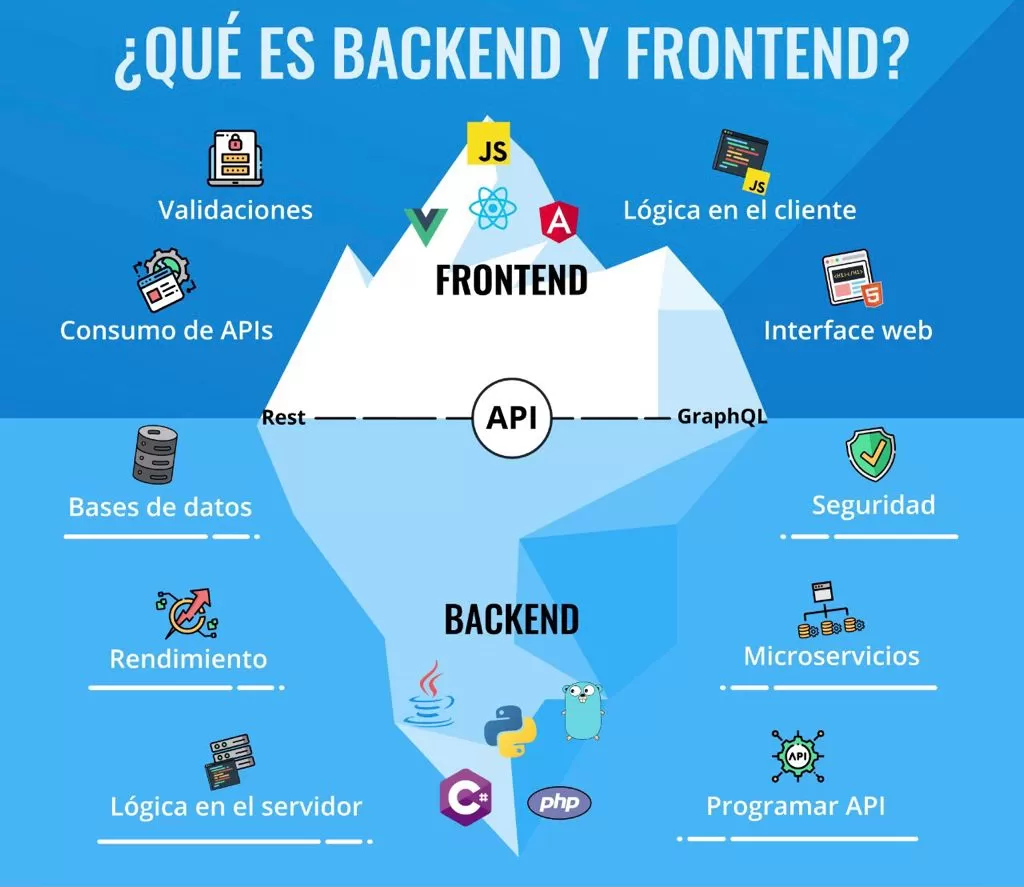
El **frontend** se refiere a la parte visible y con la que interactúa el usuario en una aplicación o sitio web. Los lenguajes de programación usados en el frontend están diseñados para manejar la interfaz gráfica y la experiencia del usuario.

* **Lenguajes principales**:
  + **HTML** (estructura de la web)
  + **CSS** (estilos y diseño)
  + **JavaScript** (interactividad y dinámica)
* **Frameworks y bibliotecas populares**:
  + **React** (JavaScript)
  + **Vue.js** (JavaScript)
  + **Angular** (TypeScript)

### **Backend (Lado del Servidor)**

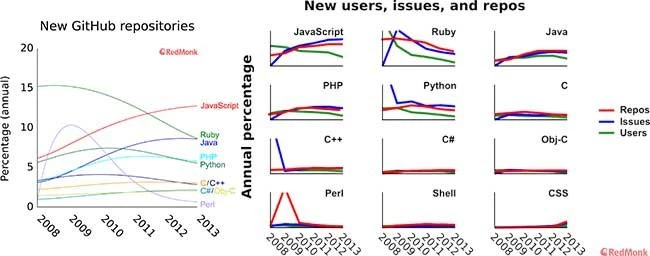
El **backend** es la parte del software que maneja la lógica de negocio, la base de datos y la comunicación con el frontend. Se encarga de procesar solicitudes, almacenar información y gestionar la seguridad.

* **Lenguajes comunes**:
  + **Java** (Spring Boot)
  + **Python** (Django, Flask)
  + **C#** (ASP.NET)
  + **JavaScript** (Node.js)
  + **PHP** (Laravel)
* **Bases de datos utilizadas**:
  + **SQL** (MySQL, PostgreSQL)
  + **NoSQL** (MongoDB, Firebase)



### **Conclusión**

* **Python**: Simplicidad y flexibilidad, ideal para desarrollo rápido y aprendizaje.
* **Java**: Más estructurado y robusto, eficiente para aplicaciones grandes y multiplataforma.





### **Instalación y Uso de Herramientas**

**Hasta el momento saben programar algoritmos sencillos con Python, sin utilizar ninguna IDE (próximo tema). A continuación, pueden ver un pequeño notebook con lo más sencillo que ya deberían saber de programación.**

**Notebook de Python:** Accede a un cuaderno interactivo con instrucciones básicas de Python: [Notebook](https://colab.research.google.com/drive/1h7B0u9XusLaDJIdgMdLP7Gk7cjW_f3zg?usp=sharing)

Una **IDE (Entorno de Desarrollo Integrado)** es una aplicación que proporciona herramientas para escribir, depurar y compilar código en distintos lenguajes de programación. Generalmente, incluye un editor de código, un compilador o intérprete, y un depurador en una interfaz unificada.

En la materia usaremos dos IDEs, (una opcional y la otra obligatoria), Code::Blocks la usaré para explicar algunos conceptos puntuales y no será obligatorio que la usen en la materia y luego Eclipse para el resto de la materia en Java (Esto si es obligatorio, aunque pueden usar otras alternativas como NetBeans, Visual Studio Code, etc, que son más complejas de configurar).

### **Code::Blocks**

Es una IDE liviana y de código abierto, ideal para C y C++. Destaca por su rapidez y flexibilidad, permitiendo el uso de múltiples compiladores.

Aquí un ejemplo sencillo de lo explicado sobre lenguajes tipados y principios de TDA. Recomiendo fuertemente probarlo en un IDE: [Acceso](https://drive.google.com/file/d/1Q5px-dgPKicu4bnm3ZmKh1SIvINScW_Z/view?usp=sharing)

En caso de tener inconvenientes con la instalación, no es central de la materia, así que pueden probarlo con **OnlineGDB** [(Acceso)](https://www.onlinegdb.com/)

### **Eclipse**

Es una IDE potente y extensible, utilizada principalmente para Java, aunque admite otros lenguajes mediante plugins. Su ecosistema lo hace ideal para grandes proyectos y desarrollo empresarial.



#### **Instalación de Code::Blocks y Uso de C**

* **Windows:** [Ver Video](https://www.youtube.com/watch?v=-cbaqg7lB6o&utm_source=chatgpt.com)
* **Linux:** [Guía Oficial](https://wiki.codeblocks.org/index.php/Installing_Code::Blocks_from_source_on_Linux)
* **Mac:** [Descargar Code::Blocks](https://sourceforge.net/projects/codeblocks/files/Binaries/13.12/MacOS/CodeBlocks-13.12-mac.zip/download)

#### **Alternativas en Línea para C**

* **Replit:** Soporta proyectos colaborativos con .h y .c ([Acceder](https://replit.com/))
* **OnlineGDB:** Depurador online con soporte para múltiples archivos ([Acceder](https://www.onlinegdb.com/))
* **JDoodle:** Rápido y liviano para ejecutar C ([Acceder](https://www.jdoodle.com/))
* **Ideone:** Para probar código rápidamente ([Acceder](https://ideone.com/))

#### **Instalación de Eclipse para Java**

* **Windows y Mac:** [Video Tutorial](https://www.youtube.com/watch?v=EnAA88eTw5Y)
* **Linux:** [Guía de Instalación](https://www.youtube.com/watch?v=XguyDdhScL8)



#### **Alternativas en Línea para Java**

* **Replit:** ([Acceder](https://replit.com/))
* **JDoodle:** ([Acceder](https://www.jdoodle.com/))
* **OnlineGDB:** ([Acceder](https://www.onlinegdb.com/))

Aquí un primer ejemplo de lo que estuvimos trabajando: [Acceso](https://drive.google.com/file/d/1vvhYN8XgorccQgw9bSx4JsEV_aUe_LqW/view?usp=sharing)

#### **Objetivos de la clase**

1. Conocer las limitaciones de Python. Diferenciar entre lenguajes, compilados, interpretados, intermedios, bajo y alto nivel, back y front.
2. Conocer las ventajas y desventajas de los lenguajes, tipados, no tipados y débilmente tipados.
3. Comprender las diferencias entre funciones y procedimientos.
4. Entender qué es un puntero, como se usa para reservar memoria y liberarla y cómo gracias a la JVM ese problema quedará resuelto en JAVA.
5. Aprender comandos básicos de JAVA, variables, decisiones, bucles, funciones, procedimientos.
6. Entender los fundamentos básicos de la Teoría del Dato Abstracto.
7. Comprender la primera intuición de Interfaz e Implementación.