

# Manual de Usuario - JavaBridge

Proyecto 2 - Lenguajes Formales y de Programación  
Universidad de San Carlos de Guatemala

---

## Información del Proyecto







- Nombre: JavaBridge - Traductor de Java a Python
  - Estudiante: Christian Javier Rivas Arreaga
  - Carnet: 202303204
  - Fecha: Octubre 2025
- 

## 1. Introducción

### 1.1 ¿Qué es JavaBridge?

JavaBridge es una aplicación web que traduce automáticamente código Java a Python. Está diseñada para procesar un subconjunto específico de Java y generar código Python equivalente, facilitando la migración de código entre estos dos lenguajes.

### 1.2 Características Principales

-  Interfaz web moderna: Diseño profesional con tema oscuro
  -  Editor de código integrado: Escribe o carga archivos .java
  -  Traducción automática: Convierte Java a Python instantáneamente
  -  Reportes HTML: Visualiza tokens, errores léxicos y sintácticos
  -  Simulación de ejecución: Visualiza la salida del código Python
  -  Gestión de archivos: Abre, guarda y exporta archivos
- 

## 2. Requisitos del Sistema

### 2.1 Requisitos Mínimos

- Sistema Operativo: Windows 10/11, macOS 10.15+, Linux (Ubuntu 20.04+)
- Navegador Web: Google Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- Node.js: Versión 18.x o superior

- RAM: Mínimo 4 GB
- Disco Duro: 500 MB de espacio libre

## 2.2 Dependencias

Backend:

- Node.js
- npm (incluido con Node.js)

Frontend:

- Navegador web moderno con JavaScript habilitado
- 

## 3. Instalación

### 3.1 Descarga del Proyecto

```
# Clonar el repositorio
git clone https://github.com/Arreagaaa/LFP_2S2025_202303204.git

# Navegar al directorio del proyecto
cd LFP_2S2025_202303204/Proyecto2
```

### 3.2 Instalación del Backend

```
# Navegar a la carpeta backend
cd backend

# Instalar dependencias
npm install

# Iniciar el servidor
node src/index.js
```

Salida esperada:

```
Servidor backend escuchando en http://localhost:3000
```

### 3.3 Instalación del Frontend

Abrir una nueva terminal y ejecutar:

```
# Navegar a la carpeta frontend
```

```
cd frontend

# Instalar dependencias
npm install

# Iniciar el servidor de desarrollo
npm run dev
```

Salida esperada:

```
VITE v5.1.4  ready in 500 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
```

## 3.4 Acceso a la Aplicación

Abrir el navegador y visitar: <http://localhost:5173>

---

## 4. Guía de Uso

### 4.1 Interfaz Principal

La interfaz está dividida en dos áreas principales:

1. Editor de Código (Izquierda): Para escribir o cargar código Java
2. Salida de Traducción (Derecha): Muestra el código Python generado

Barra de Menú Superior:

- ARCHIVO: Gestión de archivos
  - TRADUCIR: Operaciones de traducción y reportes
  - AYUDA: Información del desarrollador
- 

### 4.2 Menú ARCHIVO

#### 4.2.1 Nuevo

Función: Limpia el editor para empezar un nuevo archivo.

Pasos:

1. Hacer clic en ARCHIVO

2. Seleccionar Nuevo
3. El editor se limpiará automáticamente

#### **4.2.2 Abrir**

Función: Carga un archivo .java existente desde tu computadora.

Pasos:

1. Hacer clic en ARCHIVO
2. Seleccionar Abrir
3. Navegar y seleccionar un archivo .java
4. El contenido se cargará en el editor

Nota: Solo se aceptan archivos con extensión .java

#### **4.2.3 Guardar**

Función: Guarda el código Java actual en un archivo.

Pasos:

1. Escribir o modificar código en el editor
2. Hacer clic en ARCHIVO
3. Seleccionar Guardar
4. Elegir ubicación y nombre del archivo
5. Guardar con extensión .java

#### **4.2.4 Guardar Python Como**

Función: Guarda el código Python traducido en un archivo .py

Pasos:

1. Realizar una traducción exitosa primero
2. Hacer clic en ARCHIVO
3. Seleccionar Guardar Python Como
4. Elegir ubicación y nombre
5. Guardar con extensión .py

Requisito: Solo funciona si la traducción fue exitosa.

#### **4.2.5 Salir**

Función: Cierra la aplicación.

Pasos:

1. Hacer clic en ARCHIVO
2. Seleccionar Salir

3. La ventana del navegador se cerrará
- 

## 4.3 Menú TRADUCIR

### 4.3.1 Generar Traducción

Función: Traduce el código Java a Python.

Pasos:

1. Escribir o cargar código Java válido
2. Hacer clic en TRADUCIR
3. Seleccionar Generar Traducción
4. Esperar el procesamiento
5. Revisar el código Python en el panel derecho

Ejemplo de código válido:

```
public class MiPrograma {  
    public static void main(String[] args) {  
        int numero = 10;  
        if (numero > 5) {  
            System.out.println("Mayor que 5");  
        }  
    }  
}
```

Resultado Python:

```
numero = 10  
if numero > 5:  
    print("Mayor que 5")
```

### 4.3.2 Ver Tokens

Función: Muestra un reporte HTML con todos los tokens reconocidos.

Pasos:

1. Escribir código Java
2. Hacer clic en TRADUCIR
3. Seleccionar Ver Tokens
4. Se abrirá una nueva ventana con la tabla de tokens

Información mostrada:

- Número de token

- Tipo de token
- Valor (lexema)
- Línea y columna

#### 4.3.3 Ver Errores

Función: Muestra reportes de errores léxicos y sintácticos detectados.

Pasos:

1. Escribir código Java (válido o con errores)
2. Hacer clic en TRADUCIR
3. Seleccionar Ver Errores
4. Se abrirán ventanas con los reportes correspondientes

Tipos de errores mostrados:

Errores Léxicos:

- Caracteres no reconocidos (@, #, \$, etc.)
- Cadenas sin cerrar
- Números mal formados

Errores Sintácticos:

- Punto y coma faltante
- Llaves no balanceadas
- Estructura de clase incorrecta
- Tipos de datos no soportados

#### 4.3.4 Simular Ejecución

Función: Extrae y muestra las salidas `print()` del código Python traducido.

Pasos:

1. Generar una traducción exitosa primero
2. Hacer clic en TRADUCIR
3. Seleccionar Simular Ejecucion
4. Se abrirá una ventana mostrando:
  - Código Python completo
  - Salidas simuladas de `print()`

Ejemplo:

Si el código Python contiene:

```
print("Hola")
print(numero)
```

La simulación mostrará:

```
Hola  
numero
```

---

## 4.4 Menú AYUDA

### 4.4.1 Acerca De

Función: Muestra información sobre el proyecto y el desarrollador.

Pasos:

1. Hacer clic en AYUDA
2. Seleccionar Acerca de
3. Se navegará a la página "Acerca de"

Información mostrada:

- Nombre del proyecto
  - Descripción
  - Desarrollador
  - Carnet y sección
  - Universidad
- 

## 5. Casos de Uso

### 5.1 Caso de Uso 1: Traducción Simple

Objetivo: Traducir un programa Java básico a Python.

Código Java:

```
public class Ejemplo1 {  
    public static void main(String[] args) {  
        int a = 5;  
        int b = 10;  
        int suma = a + b;  
        System.out.println(suma);  
    }  
}
```

Pasos:

1. Copiar el código Java en el editor
2. Clic en TRADUCIR → Generar Traducción
3. Revisar el código Python generado

Código Python generado:

```
a = 5
b = 10
suma = a + b
print(suma)
```

## 5.2 Caso de Uso 2: Estructuras de Control

Objetivo: Traducir estructuras if-else y bucles.

Código Java:

```
public class Ejemplo2 {
    public static void main(String[] args) {
        int x = 10;

        if (x > 5) {
            System.out.println("Mayor");
        } else {
            System.out.println("Menor");
        }

        for (int i = 0; i < 3; i++) {
            System.out.println(i);
        }
    }
}
```

Código Python generado:

```
x = 10

if x > 5:
    print("Mayor")
else:
    print("Menor")

for i in range(0, 3):
    print(i)
```

## 5.3 Caso de Uso 3: Manejo de Errores



Objetivo: Identificar errores en código inválido.

Código Java con errores:

```
public class Ejemplo3 {  
    public static void main(String[] args) {  
        int x = 5 // Falta punto y coma  
        @#$ // Caracteres no reconocidos  
    }  
}
```

Resultado:

- Errores Léxicos: Caracteres @, #, \$ no reconocidos
  - Errores Sintácticos: Se esperaba ; después de `int x = 5`
- 

## 6. Resolución de Problemas

### 6.1 Problemas Comunes

#### Problema 1: No se puede conectar al backend

Síntoma: Error "Network Error" o "Cannot connect to server"

Solución:

1. Verificar que el servidor backend esté corriendo

Abrir terminal y ejecutar:

```
cd backend  
node src/index.js
```

- 2.
3. Confirmar que aparece: "Servidor backend escuchando en <http://localhost:3000>"

#### Problema 2: La traducción no funciona

Síntoma: No aparece código Python en el panel derecho

Posibles causas:

1. Código Java inválido: Verificar errores con "Ver Errores"

Estructura incorrecta: Asegurar que el código tenga:

```
public class NombreClase {  
    public static void main(String[] args) {
```

```
// código aquí  
}  
}
```

2.

### Problema 3: No se pueden abrir archivos .java

Síntoma: El botón "Abrir" no responde

Solución:

1. Verificar que el archivo tenga extensión .java
2. Cerrar y reabrir el navegador
3. Limpiar caché del navegador

### Problema 4: Reportes HTML no se abren

Síntoma: Las ventanas de reportes no aparecen

Solución:

1. Verificar que el navegador permita ventanas emergentes (pop-ups)
2. Configurar permisos del navegador para <http://localhost:5173>

---

## 7. Limitaciones

### 7.1 Constructos de Java Soportados

✓ Soportado:

- Tipos: int, double, char, String, boolean
- Operadores: +, -, \*, /, ==, !=, >, <, >=, <=, ++, --
- Estructuras: if-else, for, while
- Impresión: System.out.println()
- Comentarios: // y /\* \*/


✗ No Soportado:

- Múltiples clases
- Métodos personalizados (solo main)
- Arrays (excepto String[] args en main)
- Objetos y POO
- Try-catch y excepciones
- Import y packages
- Operadores lógicos (&&, ||, !)

## 7.2 Formato Obligatorio

Estructura requerida:

```
public class NombreClase {  
    public static void main(String[] args) {  
        // Todo el código debe estar aquí  
    }  
}
```

 Estructura incorrecta:

```
class MiClase { // Falta "public"  
    void main() { // Firma incorrecta  
        // código  
    }  
}
```

---

## 8. Consejos y Buenas Prácticas

### 8.1 Para Mejores Resultados

1. Usar nombres descriptivos: Evitar variables de una sola letra
2. Indentar correctamente: Facilita la lectura del código
3. Probar código simple primero: Antes de código complejo
4. Revisar reportes: Siempre verificar tokens y errores
5. Guardar frecuentemente: Usar "Guardar" regularmente

### 8.2 Ejemplos Recomendados

Declaraciones:

```
int edad = 25;  
double precio = 19.99;  
String nombre = "Juan";  
boolean activo = true;
```

Operaciones:

```
int suma = a + b;  
int producto = a * b;  
boolean mayor = a > b;
```

Estructuras:

```
if (condicion) {
```

```
// código
} else {
    // código alternativo
}

for (int i = 0; i < limite; i++) {
    // repetir
}

while (condicion) {
    // repetir mientras sea verdadero
}
```

---

## 9. Preguntas Frecuentes

### 9.1 ¿Puedo traducir cualquier código Java?

No, solo el subconjunto especificado en el proyecto. Código con características avanzadas (clases múltiples, herencia, interfaces) no será traducido correctamente.

### 9.2 ¿El código Python generado es ejecutable?

Sí, si la traducción fue exitosa (sin errores), el código Python es completamente funcional y puede ejecutarse con Python 3.x.

### 9.3 ¿Puedo editar el código Python generado?

El código Python se muestra en el panel de salida. Para editarlo, usa "Guardar Python Como" y ábrelo en un editor de Python.

### 9.4 ¿Cómo ejecuto el código Python generado?

Después de guardar el archivo .py:

```
python archivo.py
```

### 9.5 ¿Qué hago si encuentro un error no documentado?

1. Revisar la consola del navegador (F12)
  2. Verificar que backend y frontend estén corriendo
  3. Reportar el error con el código que lo causó
-

## 10. Contacto y Soporte

Desarrollador: Christian Javier Rivas Arreaga

Carnet: 202303204

Universidad: San Carlos de Guatemala

Facultad: Ingeniería

Carrera: Ingeniería en Ciencias y Sistemas

GitHub: [https://github.com/Arreagaaa/LFP\\_2S2025\\_202303204](https://github.com/Arreagaaa/LFP_2S2025_202303204)

---

## 11. Notas Finales

Este manual ha sido diseñado para facilitar el uso de JavaBridge. Si tienes dudas adicionales o encuentras problemas no documentados, no dudes en consultar el Manual Técnico para información más detallada sobre el funcionamiento interno del sistema.

¡Gracias por usar JavaBridge!

---

Elaborado por: Christian Javier Rivas Arreaga

Fecha: Octubre 2025

Versión: 1.0