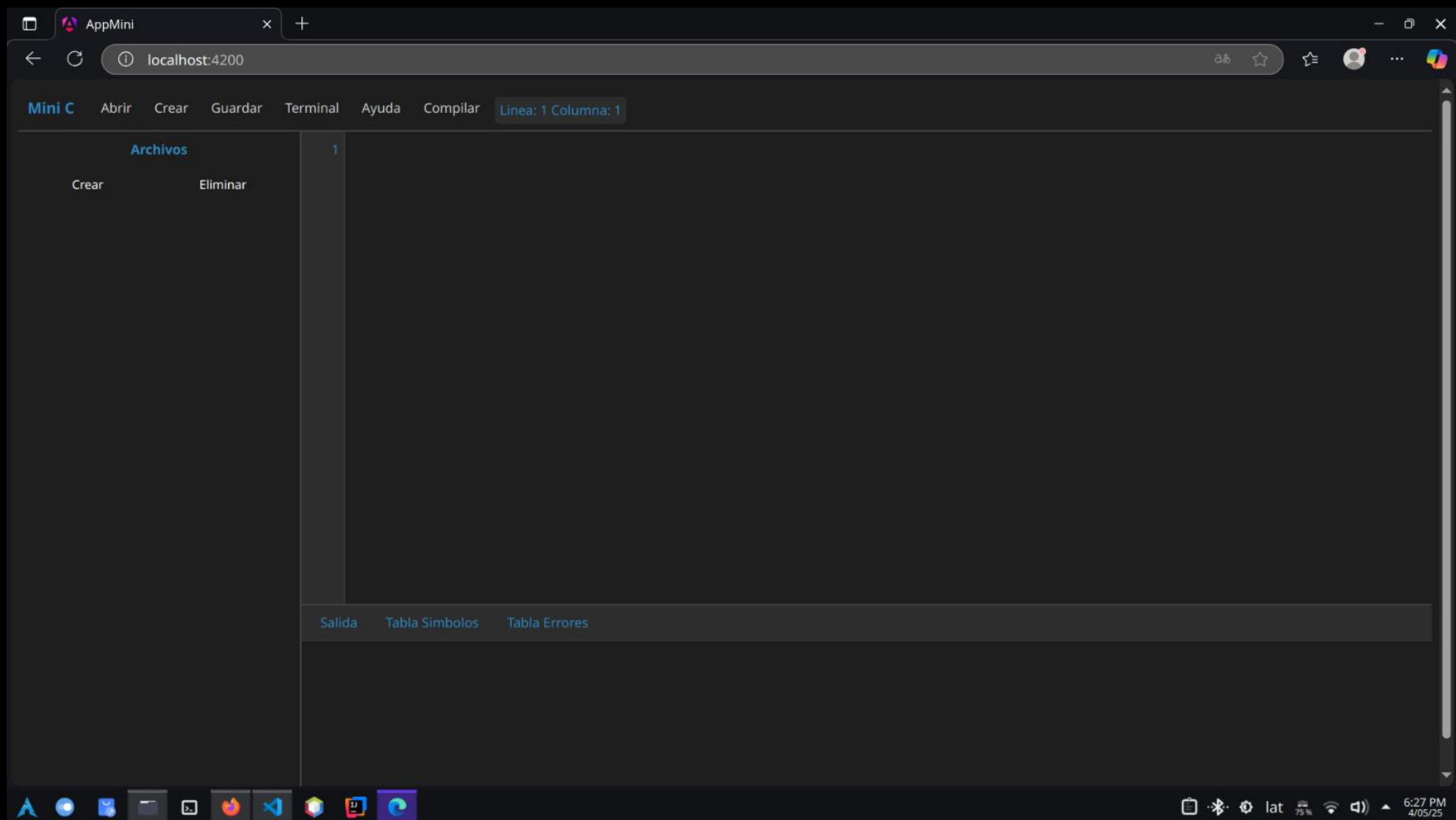


Manual Usuario

Es un entorno de desarrollo web diseñado para escribir, editar y compilar programas en un lenguaje propio inspirado en C. Desarrollado completamente en Angular, Mini-C ofrece una experiencia interactiva y accesible directamente desde el navegador, sin necesidad de instalaciones adicionales ni servidores backend.

Este entorno permite a los usuarios crear y gestionar proyectos fácilmente, escribir código en archivos .cmm, compilarlo, visualizar la salida del programa, y obtener herramientas útiles como la tabla de símbolos, la tabla de errores, y un gráfico visual del Árbol de Sintaxis Abstracta (AST).

Visualizar IDE Mini-C

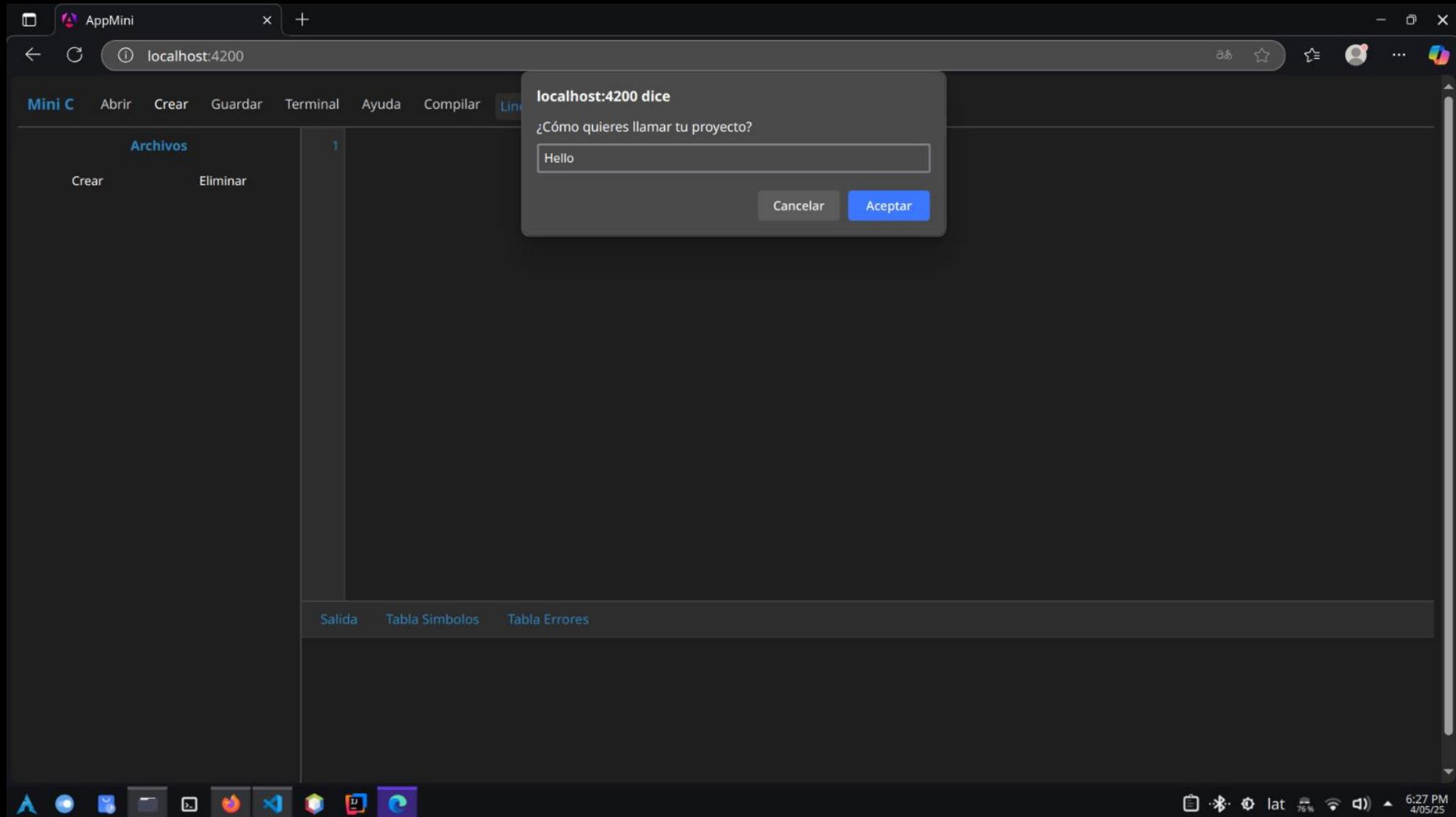


Creación de Proyectos

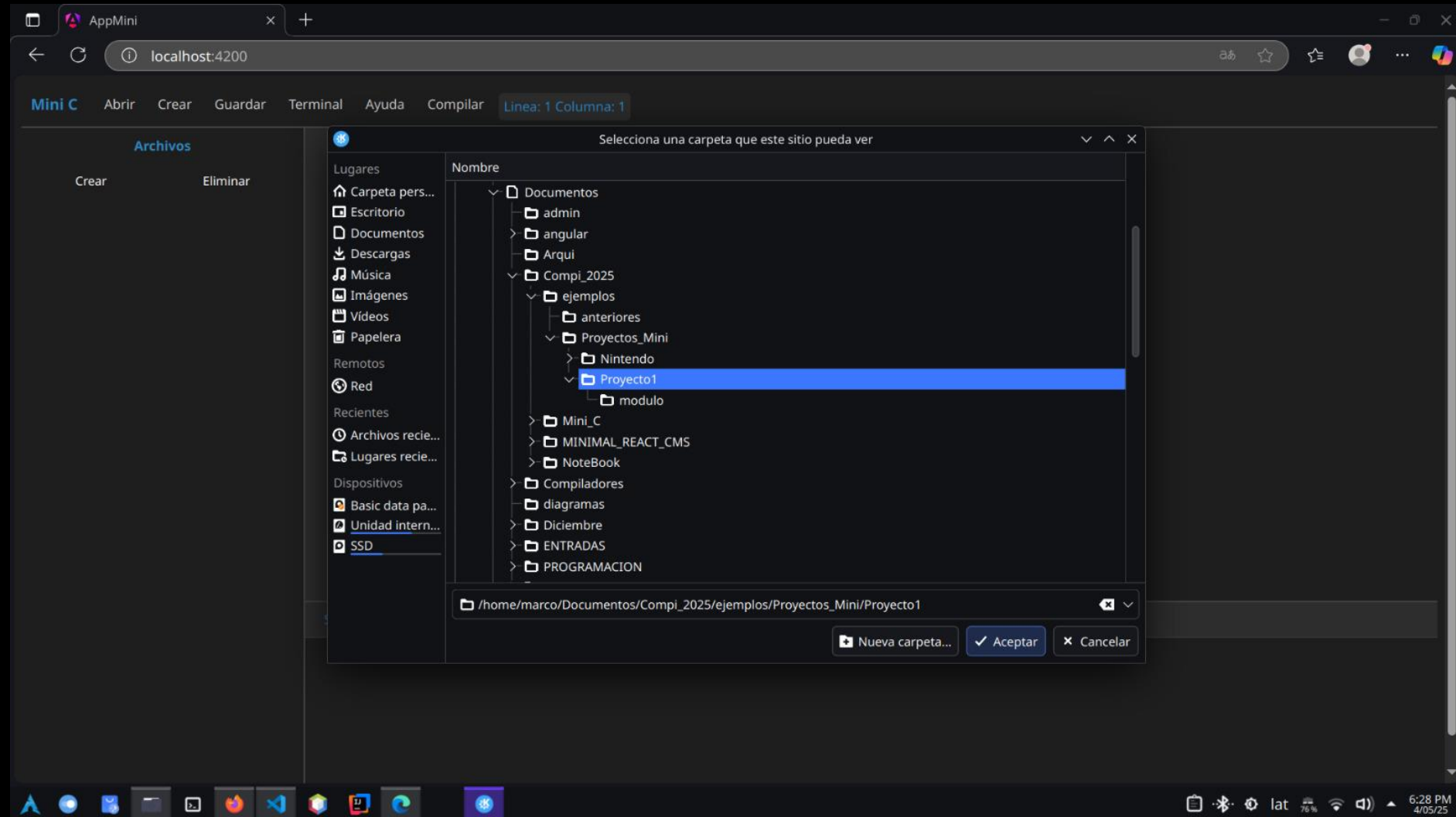
Para iniciar un nuevo proyecto en Mini-C, el usuario debe hacer clic en el botón "Crear Proyecto". Esta acción genera automáticamente una estructura básica de proyecto con los siguientes elementos:

- Una carpeta raíz con el nombre del proyecto.
- Un archivo fuente principal: `main.cmm`, que contiene la clase `main`, punto de entrada del programa.
- Un archivo de configuración: `config.yml`, que gestiona la estructura del proyecto y mantiene actualizados los archivos y carpetas creados o eliminados.

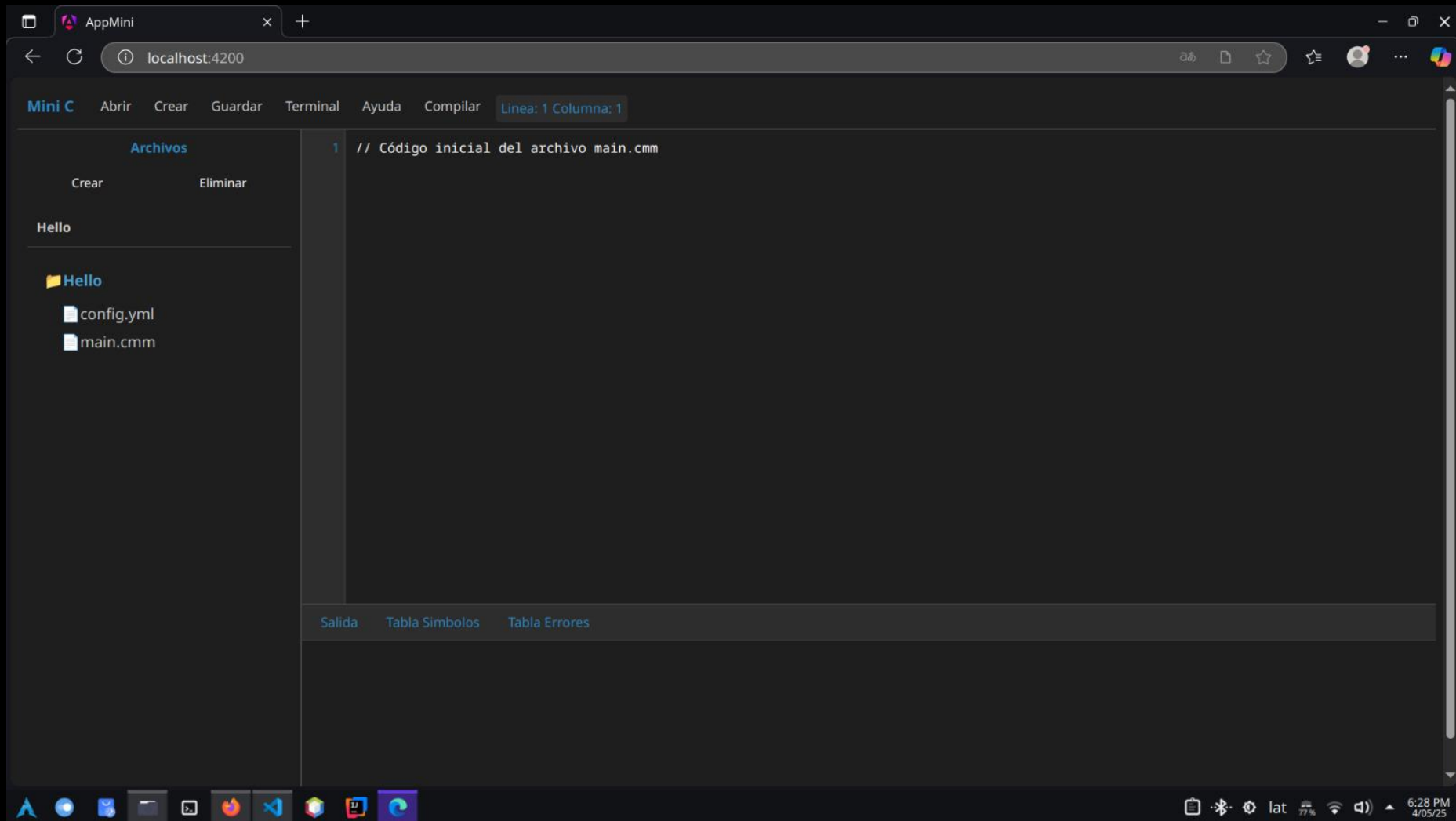
Crear proyecto



Selección donde guardar el proyecto



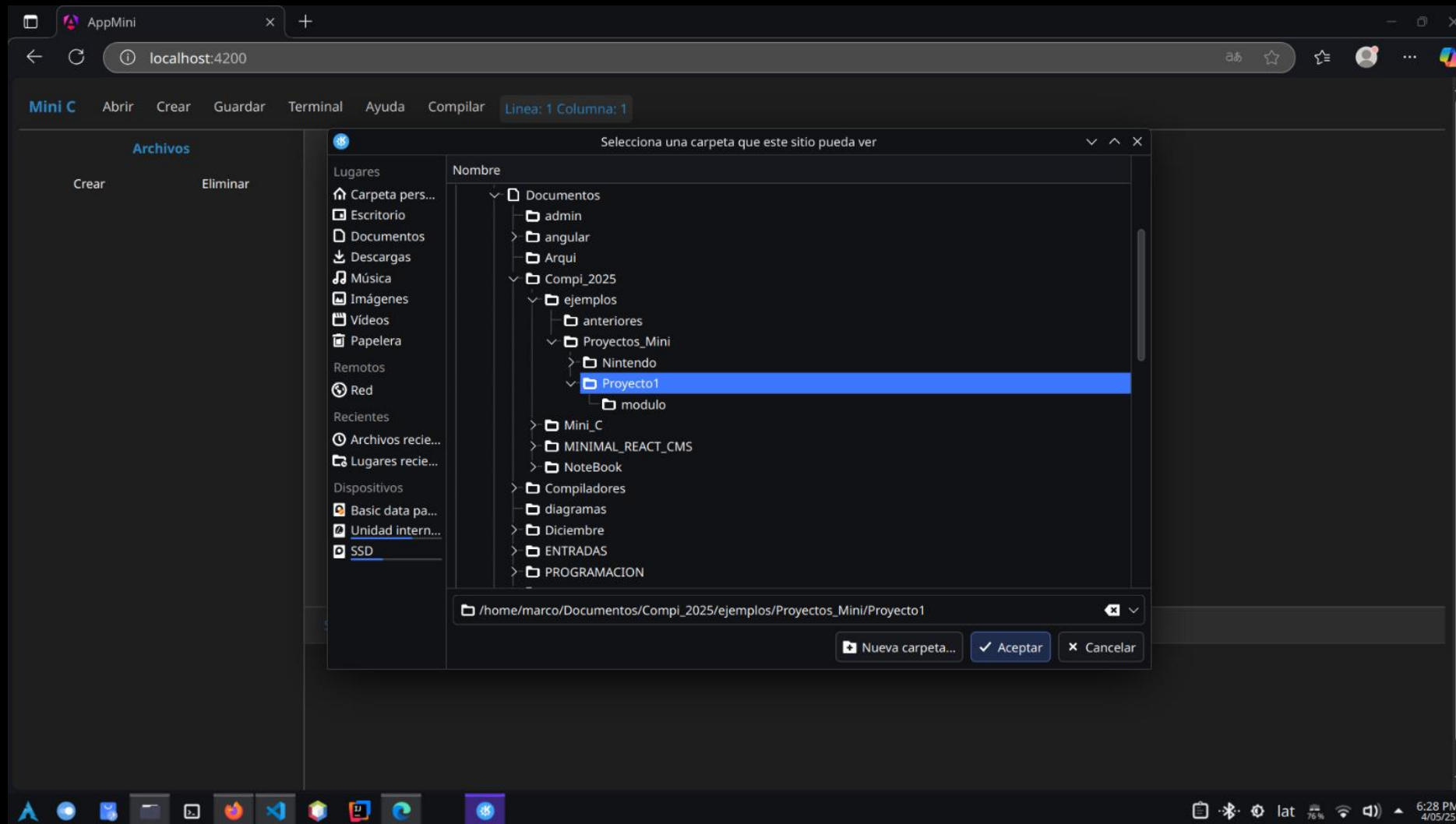
Ver el proyecto



Apertura de Proyectos Existentes

El sistema permite abrir proyectos previamente creados. Al seleccionar una carpeta de proyecto, Mini-C carga el contenido del config.yml y muestra en el editor todos los archivos .cmm y carpetas correspondientes a dicho proyecto.

Abrir un proyecto

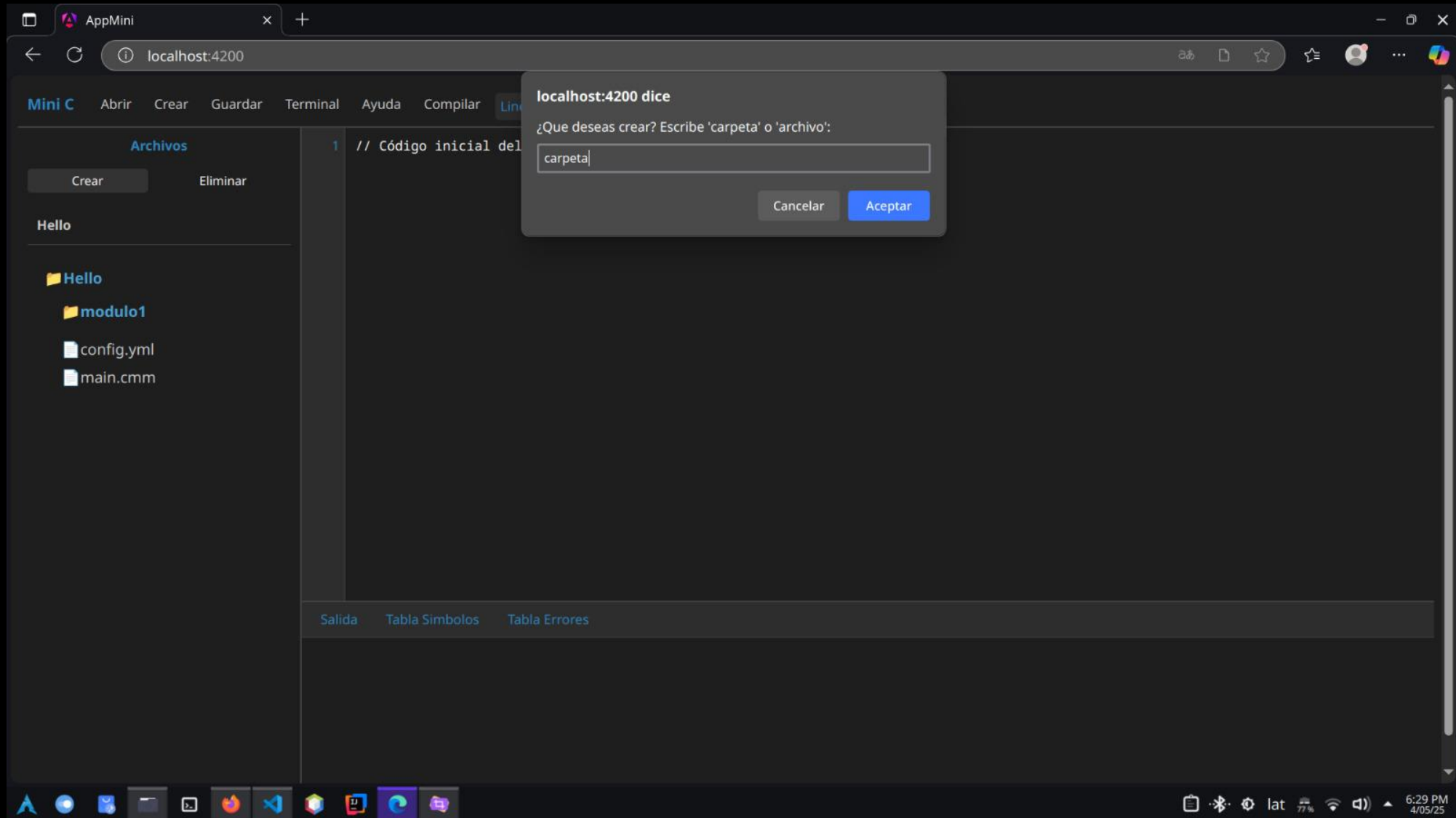


Gestión de Archivos y Carpetas

El usuario puede:

- Crear nuevas carpetas y archivos .cmm dentro del proyecto.
- Eliminar archivos o carpetas existentes.
- Todas estas operaciones actualizan automáticamente el archivo config.yml para reflejar la nueva estructura del proyecto.

Crear carpeta o archivos



Edición de Archivos

- Los archivos `.cmm` y el archivo `config.yml` se pueden visualizar y editar directamente desde el editor integrado.
- El usuario puede realizar cambios en el contenido de estos archivos y luego guardar sus modificaciones desde la interfaz.
- El lenguaje esta basado en C.



Imprimir



```
1 void main () {  
2     print("Hola Mundo");  
3 }
```

Declarar e Imprimir



```
1 void main () {  
2     string saludar = "Hola Mundo";  
3     int a = 10;  
4     float b = 12.2;  
5  
6     print(saludar);  
7     print(a + b);  
8 }
```

If

```
1 void main () {  
2     int a = 5;  
3     int b = 10;  
4  
5     if (a < b) {  
6         print("${a} es mayor a ${b}");  
7     } else {  
8         print("${a} es menor a ${b}");  
9     }  
10 }
```

For

```
1 void main () {  
2     for(int i = 0; i < 5; i++) {  
3         print(i);  
4     }  
5 }
```

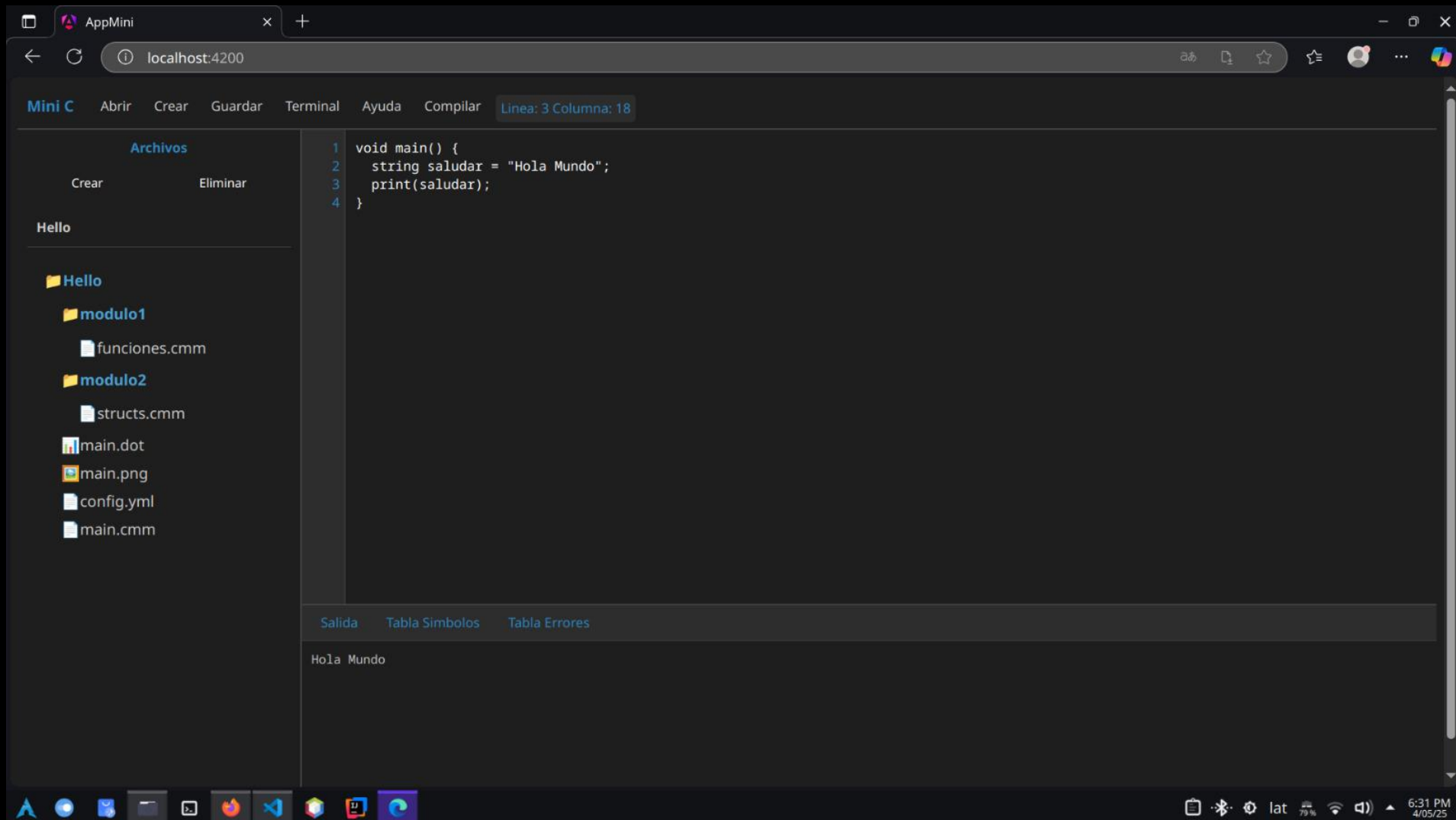
Struct

```
1 void main () {
2     struct Persona {
3         string nombre;
4         int edad;
5     }
6
7     struct Persona p1 = {"Marcos", 21};
8     print("Mi nombre es: ${p1.nombre}");
9     print("Mi edad es: ${p1.edad}");
10 }
```

Function

```
1 void main () {
2
3     int suma(num1, num2) {
4         return num1 + num2;
5     }
6
7     void condicion(variable) {
8         if (variable > 5) {
9             return;
10        }
11        print("Hola mundo");
12    }
13
14    int resultado = suma(4, 6);
15    print(resultado);
16    condicion(2);
17 }
```

Editar archivos .cmm

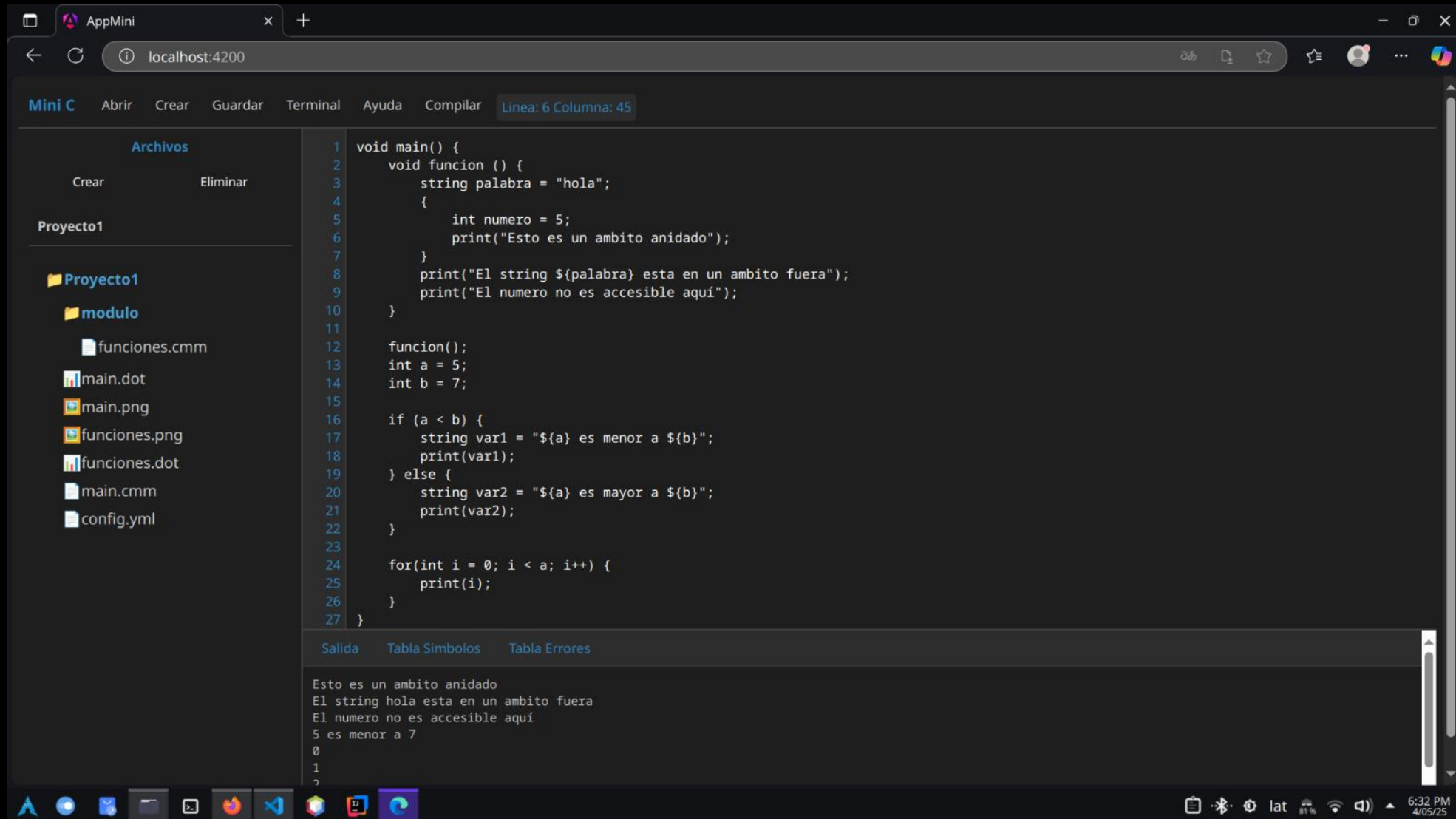


Compilación de Archivos

Al presionar el botón "Compilar", el sistema analiza el archivo actual .cmm, realiza el análisis sintáctico y semántico, y muestra los siguientes resultados en la terminal integrada:

- Salida del programa: Si el código se ejecuta correctamente, se muestra su salida estándar.
- Tabla de Símbolos: Muestra las variables declaradas, su tipo, valor y posición (línea y columna).
- Tabla de Errores: Lista de errores léxicos y sintácticos detectados durante la compilación.

Salida de código



The screenshot shows the AppMini IDE interface. The top bar includes a menu with 'Abrir', 'Crear', 'Guardar', 'Terminal', 'Ayuda', and 'Compilar'. The status bar indicates 'Linea: 6 Columna: 45'. The left sidebar shows a file explorer with a project named 'Proyecto1' containing a folder 'modulo' and several files: 'funciones.cmm', 'main.dot', 'main.png', 'funciones.png', 'funciones.dot', 'main.cmm', and 'config.yml'. The main editor displays C code with line numbers 1 through 27. The code defines a 'main' function and a nested 'funcion' function. The 'funcion' function prints a message about a nested scope and then calls 'funcion()' with local variables 'a' and 'b'. The 'main' function prints a message about a string scope and then calls 'funcion()' with 'a' and 'b'. The output window at the bottom shows the execution results: 'Esto es un ambito anidado', 'El string hola esta en un ambito fuera', 'El numero no es accesible aquí', '5 es menor a 7', and a loop output '0', '1', '2'.

```
1 void main() {
2     void funcion () {
3         string palabra = "hola";
4         {
5             int numero = 5;
6             print("Esto es un ambito anidado");
7         }
8         print("El string ${palabra} esta en un ambito fuera");
9         print("El numero no es accesible aquí");
10    }
11
12    funcion();
13    int a = 5;
14    int b = 7;
15
16    if (a < b) {
17        string var1 = "${a} es menor a ${b}";
18        print(var1);
19    } else {
20        string var2 = "${a} es mayor a ${b}";
21        print(var2);
22    }
23
24    for(int i = 0; i < a; i++) {
25        print(i);
26    }
27 }
```

Salida Tabla Simbolos Tabla Errores

Esto es un ambito anidado
El string hola esta en un ambito fuera
El numero no es accesible aquí
5 es menor a 7
0
1
2

Tabla de Símbolos

Archivos

Crear Eliminar

Proyecto1

Proyecto1

modulo

funciones.cmm

main.dot

main.png

main.cmm

funciones.png

funciones.dot

config.yml

1 void main() {

2 string saludar = "Hola mundo";

3 int a = 6;

4 float b = 13.6;

5 bool uno = true;

6 bool cero = false;

7 char carac = 'a';

8 }

Salida

Tabla Simbolos

Tabla Errores

ID	Tipo	Valor	Línea	Columna
saludar	string	Hola mundo	2	5
a	int	6	3	5
b	float	13.6	4	5
uno	bool	true	5	5

AppMini

localhost:4200

Linea: 8 Columna: 4

6:32 PM

4/05/25

Tabla de Errores

Archivos

Crear Eliminar

Proyecto1

Proyecto1

modulo

funciones.cmm

main.dot

main.png

funciones.png

funciones.dot

main.cmm

config.yml

1 void main() {

2 void funcion () {

3 string palabra = "hola";

4 {

5 int numero = 5;

6 print("Esto es un ambito anidado");

7 }

8 print("El string \${palabra} esta en un ambito fuera");

9 print("El numero no es accesible aquí");

10 }

11 }

12 funcion();

13 int a = 5.6;

14 int b = 7;

15 }

16 if (a < b) {

17 string var1 = "\${a} es menor a \${b}";

18 print(var1);

19 } else {

20 string var2 = "\${a} es mayor a \${b}";

21 print(var2);

22 }

23 }

24 for(int i = 0; i < a; i++) {

25 print(p);

26 }

27 }

Salida

Tabla Simbolos

Tabla Errores

Tipo	Descripción	Línea	Columna
SEMANTICO	Tipo FLOAT no compatible con INT	13	5
SEMANTICO	La Variable a no existe	16	9
SEMANTICO	La Variable a no existe	24	24

AppMini

localhost:4200

Linea: 13 Columna: 16

6:35 PM

4/05/25

Visualización del Árbol AST

Como parte del proceso de compilación, se genera automáticamente un archivo de imagen .png que representa el Árbol de Sintaxis Abstracta (AST) del código ingresado. Esta imagen es generada a partir del análisis estructural del código fuente y proporciona una visualización gráfica útil para depuración y análisis del programa.

Ver imagen AST

