

1 Open index.html

2 ¿Como abrir archivos?
Click "File"



3

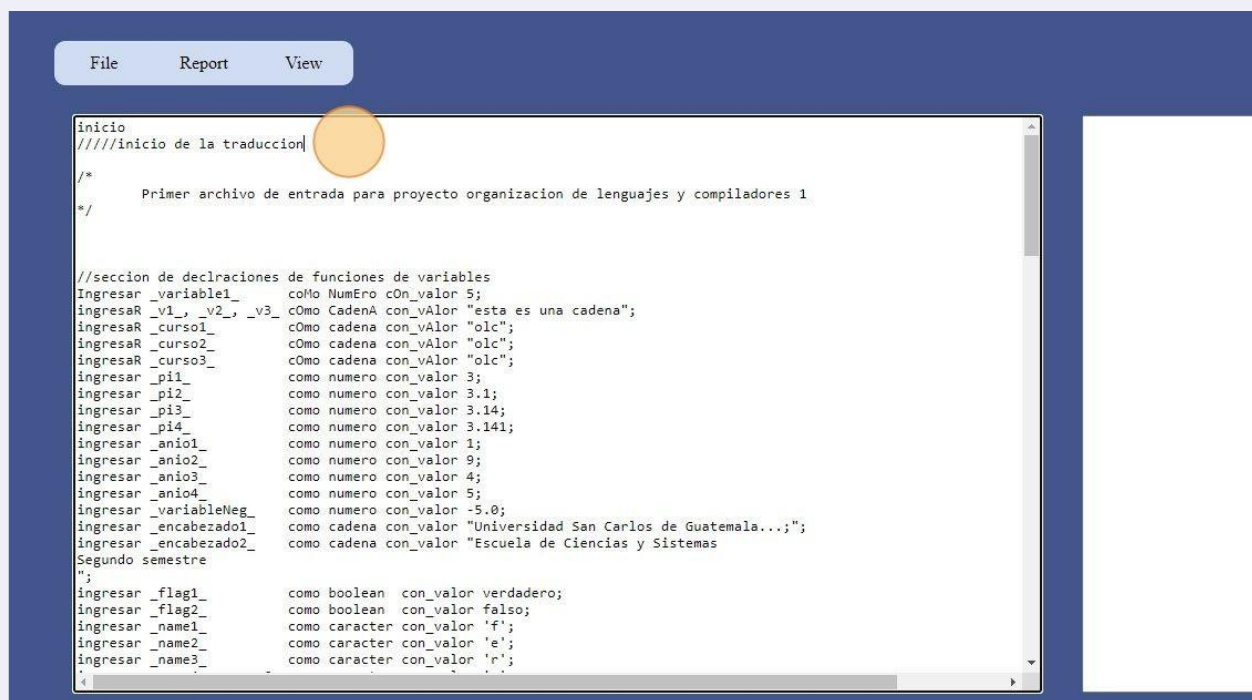
Click "Open file"

Le permitirá seleccionar archivos de pseudo código en su PC y verlos en el navegador



4

Se moverá el texto al cuadro de entrada del lado izquierda, el cual podrá editar el contenido o enviarlo a traducir

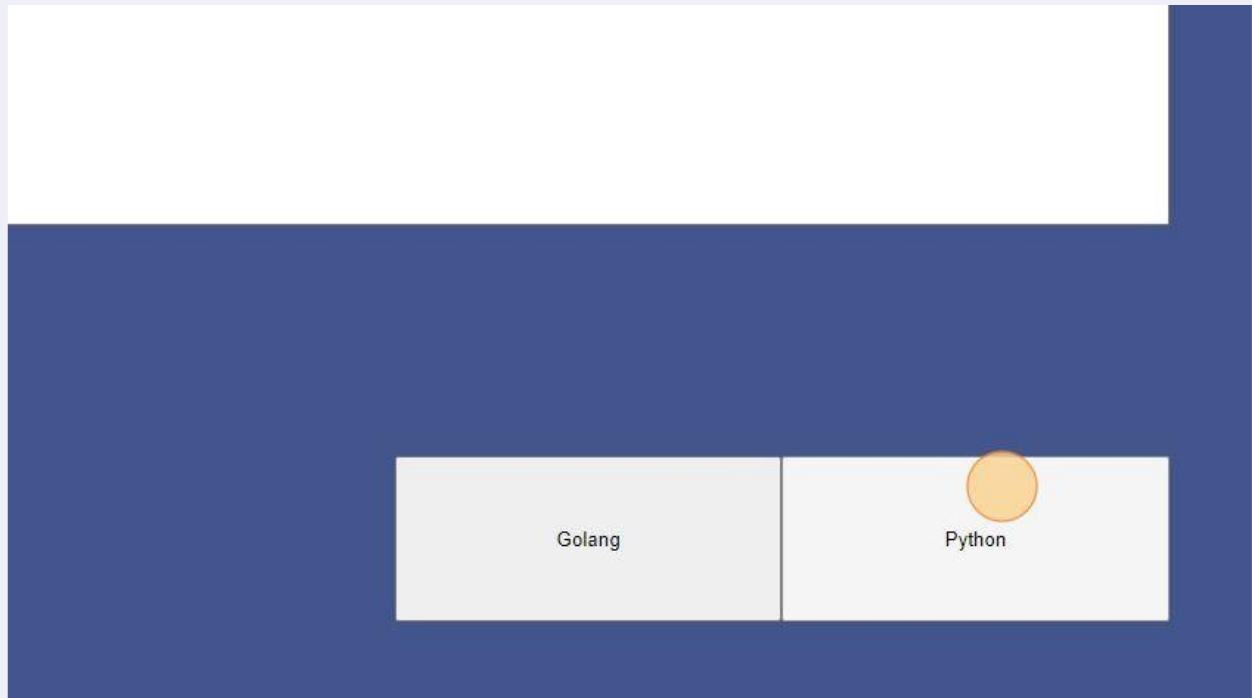


5

¿Como traducir?

Dependiendo del lenguaje al que desee traducir el pseudocódigo de

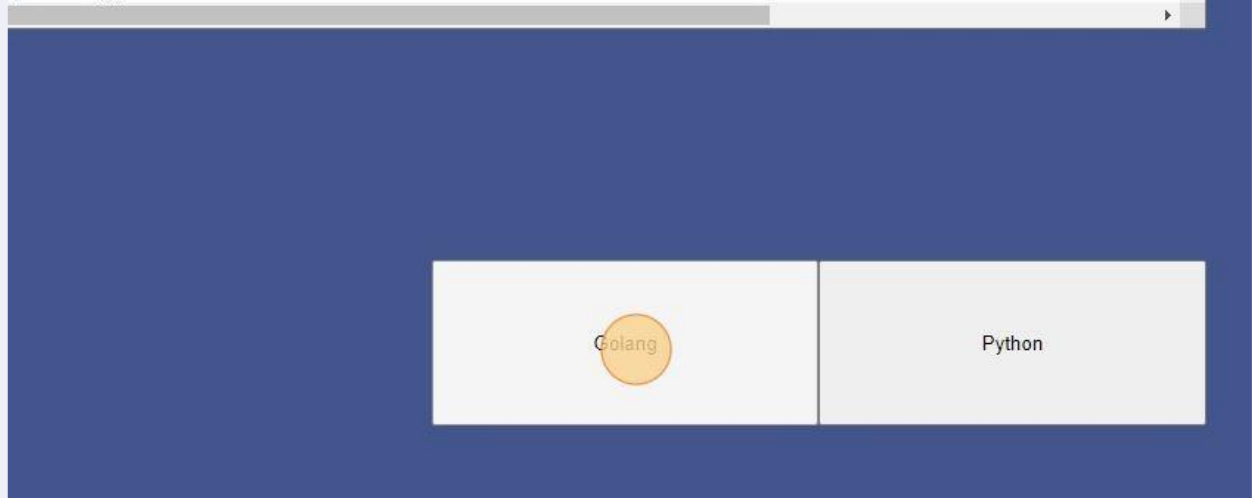
Click "Python"



6

o Click "Golang"

```
def _potenciafuncion(_base_,_exponente_):  
    _potencia_ = _base_**(_exponente_)  
    return _potencia_  
def _metodo1():  
    print("estamos entrando al metodo 1")  
    _potenciaManual_(3*1+4/2,3+2)  
    print(_potenciaFuncion_(3*1+4/2,3+2))  
    print(" Esta es la potencia Funcion")  
_metodo1()
```



7

Podrá observar la traducción en el lado derecho

OLC1_2S_2022
201700703

```
a_;  
do  
  
a_;  
a_+1;  
ca_)*(8+5);  
asica_;
```

```

if __name__ == '__main__':
    ##inicio de la traduccion

    ...

    Primer archivo de entrada para proyecto organizacion de lenguajes y compiladores 1

    ...

    #seccion de declaraciones de funciones de variables

    _variable1_ = 5
    _v1_, _v2_, _v3_ = "esta es una cadena", "esta es una cadena", "esta es una cadena"
    _curso1_ = "olc"
    _curso2_ = "olc"
    _curso3_ = "olc"
    _pi1_ = 3
    _pi2_ = 3.1
    _pi3_ = 3.14
    _pi4_ = 3.141
    _anio1_ = 1
    _anio2_ = 9
    _anio3_ = 4
    _anio4_ = 5
    _variableNeg_ = -5.0
    _encabezado1_ = "Universidad San Carlos de Guatemala..."
    _encabezado2_ = "" "Escuela de Ciencias y Sistemas

Segundo semestre
""

    _flag1_ = True
    _flag2_ = False
    _name1_ = 'f'
    _name2_ = 'e'
    _name3_ = 'n'
    _name4_, _name6_ = 'n', 'n'
    _name5_ = 'a'

```

8

En dado caso el texto ingresado tuviera errores, mostrara los errores en el siguiente recuadro indicando que tipo de tokens/simbolos estaba esperando

```
//seccion de asignaciones
_v1_          -> "esta es la cadena numero 1";
_v2_, _v3_    -> "estas cadenas deben ser diferentes";
_curso1_ , _curso2_ , _curso3_  "Organizacion de lenguajes y compiladores 1";

imprimir_n1 _encabezado1;
imprimir_n1 _encabezado2;
```

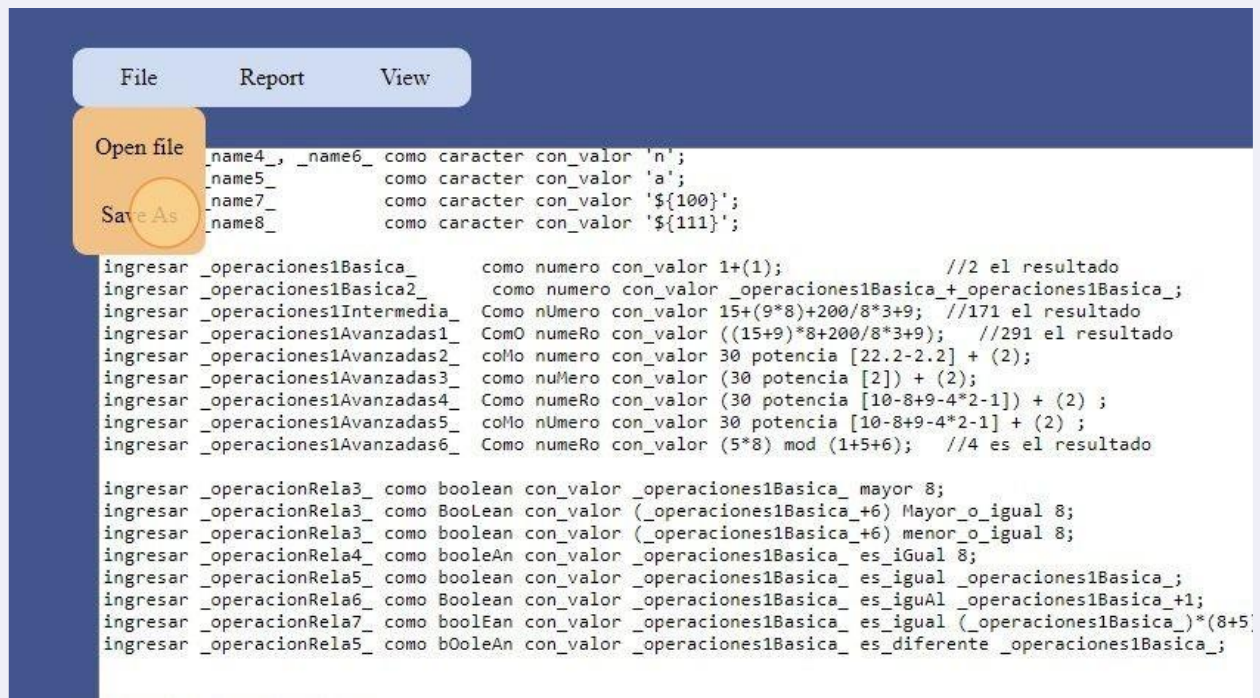
Fila: 62 Columna: 32

Errores

```
ERROR EN: Linea 62 Columna 33, texto: "Organizacion de lenguajes y compiladores 1"
Izquierda: IMPRIMIR , Derecha: TIPO ,
Expected tokens: [ASIGNACION]
```

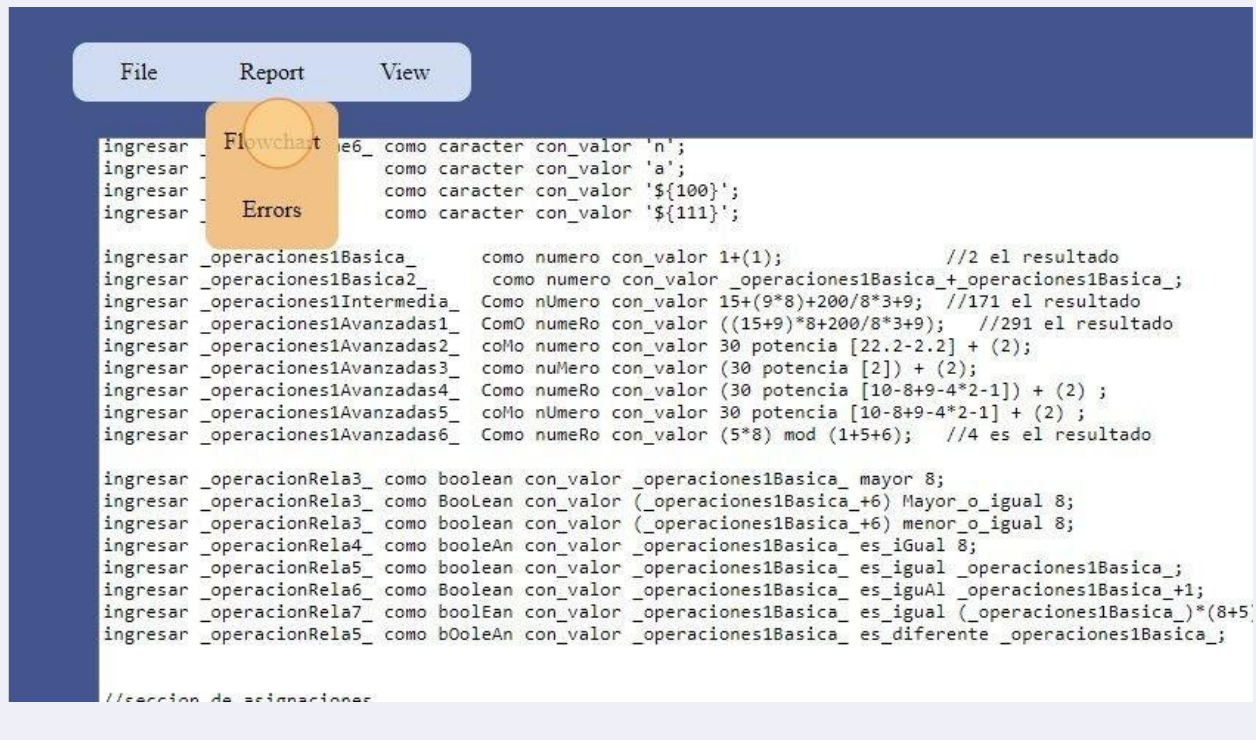
9

Si desea guardar el pseudocódigo a un archivo, puede dar Clic en el botón Save As, lo que descargara un archivo con el código automáticamente

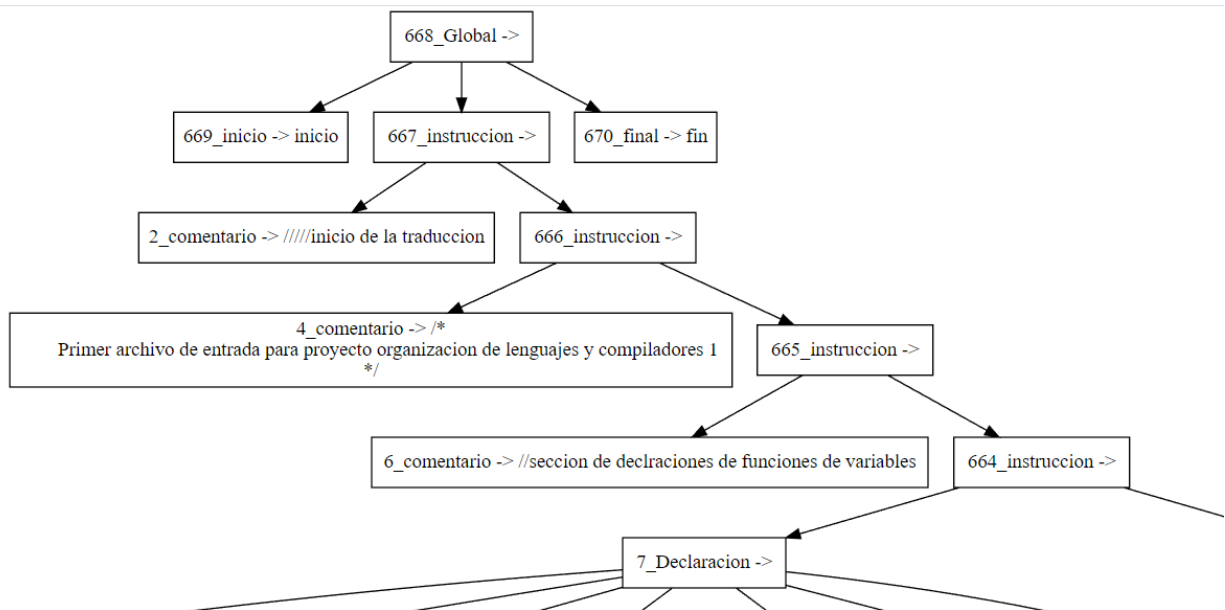


10 Click "Flowchart"

Le llevará a ver el árbol del código ingresado y como este fue estructurado en una nueva ventana

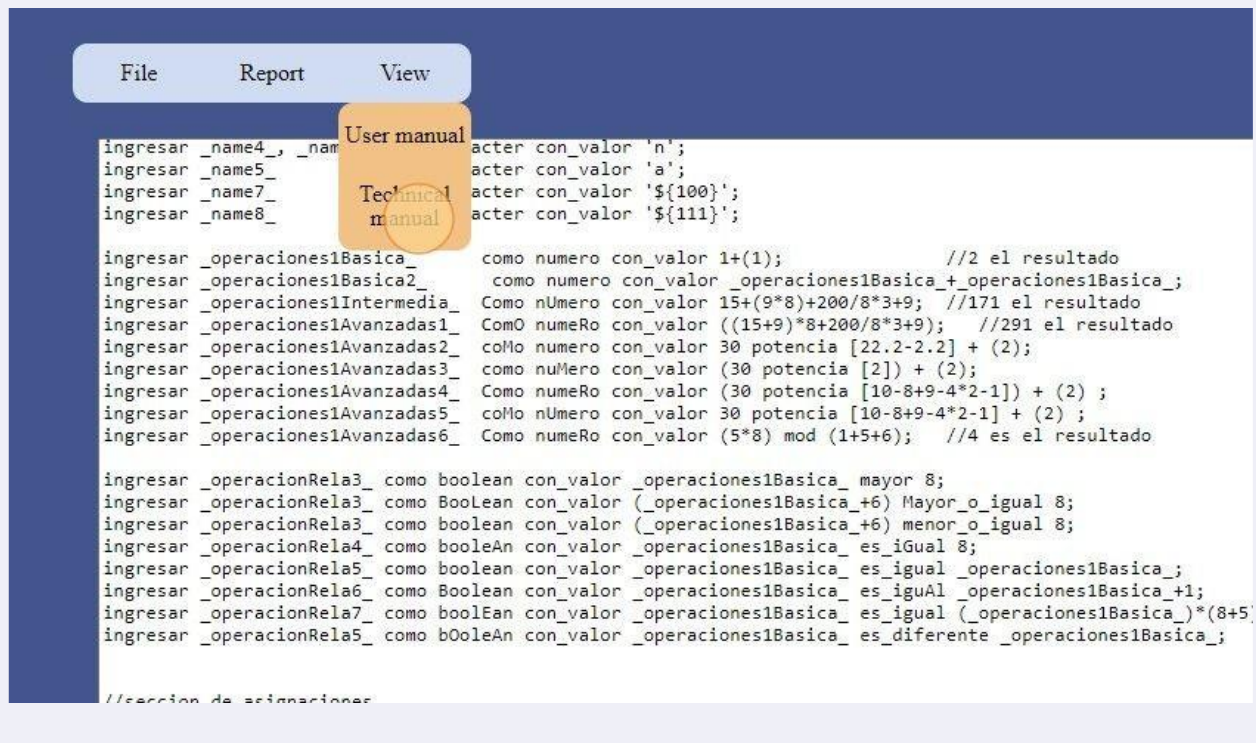


11 Switch to tab

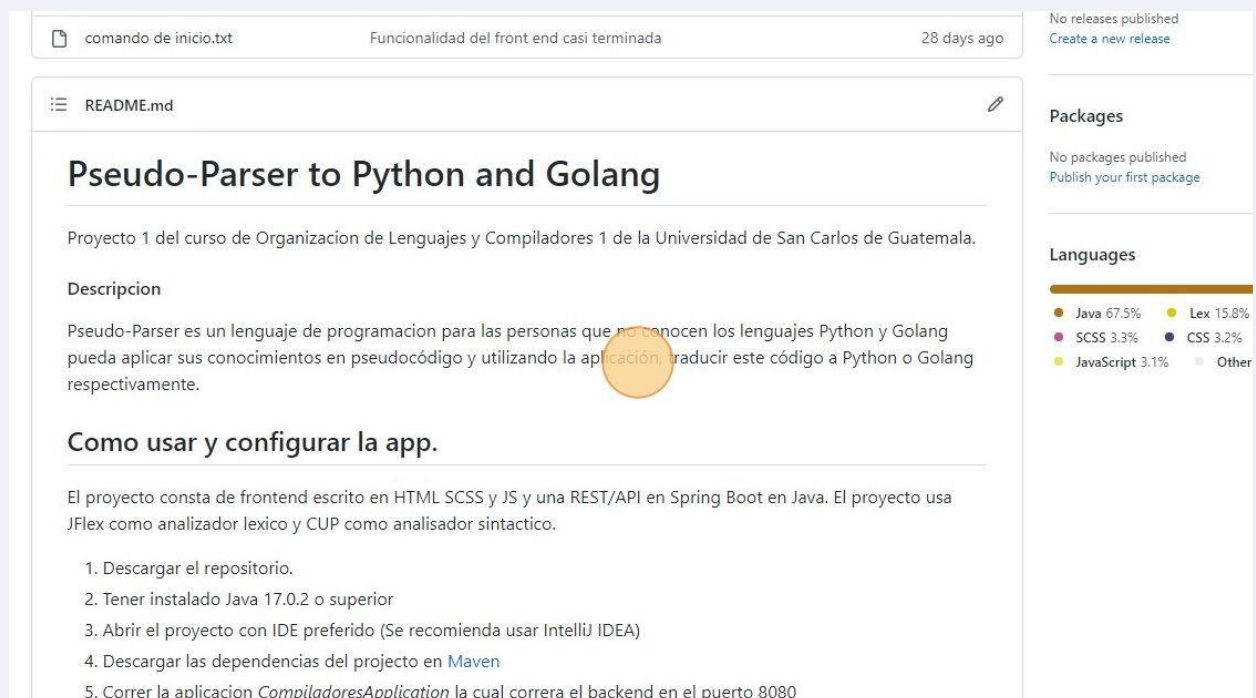


12 Click "Technical manual"

Lo llevará a ver la documentación técnica en GitHub



13 Documentacion tecnica



pueda aplicar sus conocimientos en pseudocódigo y utilizando la aplicación, traducir este código a Python o Golang respectivamente.

Como usar y configurar la app.

El proyecto consta de frontend escrito en HTML SCSS y JS y una REST/API en Spring Boot en Java. El proyecto usa JFlex como analizador lexico y CUP como analizador sintactico.

1. Descargar el repositorio.
2. Tener instalado Java 17.0.2 o superior
3. Abrir el proyecto con IDE preferido (Se recomienda usar IntelliJ IDEA)
4. Descargar las dependencias del proyecto en [Maven](#)
5. Correr la aplicacion *CompiladoresApplication* la cual corra el **backend** en el puerto 8080
6. Abrir el *index.html* para observar la interfaz. Dentro de *script.js* existe una variable que indica la direccion ip del backend, cambiarla a la direccion ip de su PC de lo contrario no funcionaran las peticiones al backend.
7. Listo para usarse.

Estructura del proyecto

Backend

src/main/

- */cup*

JavaScript 3.1%

Estructura del proyecto

Backend

src/main/

- */cup*

En esta carpeta encontraras dos archivos, *parser.cup* y *golang_helper.cup*

parser.cup: Archivo encargado de generar el parser para el analisis sintactico del pseudo codigo.

golang_helper.cup: Archivo encargado de generar un parser llamado *golang_helper* el cual ayuda al traductor de golang a reacomodar la sintaxis al momento de la traduccion. Este nuevo parser es utilizado en la clase *Traductor_Go* mas adelante.

• *java/com/backend/compiladores*

mainController: Encargado de recibir las peticiones POST del **front** para iniciar la traduccion a traves del *compilerController*

CompiladoresApplication: Clase principal para correr Spring boot

- */services*

compilerController: Encargado de crear el lexer, parser, el AST (Abstract Syntax Tree), traducir el codigo y determinar los errores lexicos o sintacticos que puedan surgir, anexando toda la informacion generada al objeto response, el arbol generado se guarda en la carpeta resources.

- */parserPackage*

Clases para la creacion de arboles y nodos

- */response*

16

Click "User manual"

Lo llevara a ver este manual

