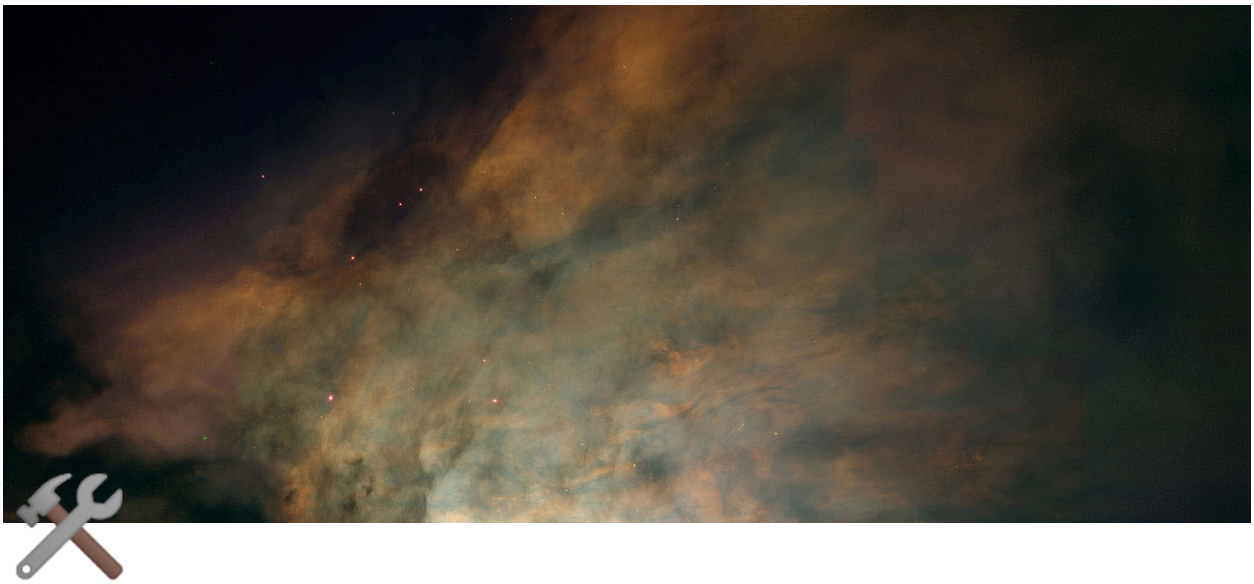

Manual Técnico SysCompiler

SysCompiler es un programa que utiliza como cliente una página web en donde se inserta y se ve toda la información que los usuarios necesitan saber saber. Además, SysCompiler utiliza un API la cual solamente maneja una ruta en la cual se manda la información necesaria para analizar y como respuesta devuelve una serie de datos para que se muestren en la página web.

Front-end

La parte front-end de Syscompiler está conformada por una página web, la cual se ubica en el puerto 4200 del localhost. El proyecto está construido en su totalidad con angular 8 y versión de node 14.17.2. La única librería externa utilizada en el proyecto es la de vis, la cual se utiliza para construir el AST de la gramática de entrada. La versión de vis utilizada es la número 4.21.0 y se importa en el componente ast.component.ts.

Para activar el front-end se hace uso del siguiente comando

```
ng serve
```

Componentes

▼ Menu

El componente menu es el componente principal y el único que está asociado con una ruta. En este componente se ubican los editores de texto, así como la consola y los botones. El menu se encarga de captar la gramática de entrada y mandarla al API a ser analizada. Cuando se obtiene una respuesta, se guarda en las variables globales del componente, como estructura, símbolos, errores y salidas. Cada una de estas variables sirven para proveer al usuario de la información del compilador al analizar la gramática introducida. Las salidas y los errores van a la consola. Los símbolos van al componente de Símbolos, los errores van al componente de errores y la estructura va al componente del AST.

▼ Símbolos

Cuando el botón de "Tabla de Símbolos" en el componente del menu es ejecutado, el componente de símbolos se activa y se muestra al usuario. La única variable que este componente utiliza es la de símbolos y es un arreglo de objetos de tipo símbolos. La parte visual del componente se trata de una tabla en la que se despliegan todos los símbolos encontrados por el analizador de manera ordenada para que los usuarios la entiendan.

▼ Errores

Cuando el botón de "Reporte de Errores" en el componente del menu es ejecutado, el componente de errores se activa y se muestra al usuario en la parte inferior de la pantalla. Este funciona similar al componente de símbolos, ya que maneja una sola variable y es un arreglo de objetos de tipo error, los cuales muestran el tipo de error, una descripción del error, y sus respectivas líneas y columnas. La parte visual del componente consiste en una tabla que muestra cada uno de los errores encontrados por el analizador y los muestra de una manera ordenada.

▼ AST

Cuando el botón de "Generar Árbol AST" en el componente del menu es ejecutado, el componente del AST se activa y se muestra al usuario en la parte inferior de la pantalla. Este componente solamente maneja un arreglo con el contenido de la gramática que se ingreso como entrada para el analizador. Luego de recibir el contenido, el componente inicia un procedimiento para convertir la data en un árbol AST con ayuda de la librería vis. Este es el único componente que hace uso de

esta librería. Después de terminar el procedimiento de generar el árbol AST, lo muestra en pantalla.

Servicio

El proyecto cuenta con un servicio que actúa como intermediario para conectar el front-end con el back-end. Solamente cuenta con una función que hace una petición http a la ruta del API y como cuerpo de la petición, se manda la entrada completa para ser analizada. Como respuesta se obtiene todos los datos descritos en el componente del menu.

Ruta

El proyecto cuenta solamente de una ruta, la cual es "menu" y se dirige directamente al componente del menu. Cualquier otra dirección ingresada en el navegador redirigirá a la ruta "menu" para evitar errores.

Back-end

El proyecto en el back-end consiste en una API construida enteramente con node versión 14.17.2. El API contiene solamente una ruta de tipo POST la cual recibe el código de entrada para ser analizada por el compilador. El proyecto está construido con el lenguaje TypeScript, sin embargo este proyecto es traducido a JavaScript a una carpeta en el proyecto llamada "build" y es esa la carpeta que se ejecuta para que funcione el proyecto. Es importante destacar que el proyecto esta disponible en el puerto 8080 del localhost.

El comando para iniciar el servidor es el siguiente:

```
npm run dev
```

El comando para detectar los cambios en el proyecto de TypeScript y traducirlos a JavaScript es:

```
npm run build
```

Paquetes Utilizados en el proyecto

- Express

  Express se encarga hacer posible la construcción del API ya que permite que el envío de solicitudes al puerto indicado y enviar respuestas devuelta. Express se encarga de esperar las solicitudes de tipo http y de esta manera comunicarse con el front-end.

- Cors

  Cors funciona en paralelo con Express para administrar las solicitudes de tipo http y mandar devuelta respuestas de tipo http, las cuales son compatibles con el front-end.

- Morgan

  Morgan es utilizado para que los desarrolladores del proyecto sepan en que momento se hizo una solicitud al servidor y de qué tipo fue la solicitud. Tambien informa si hubo algún tipo de error a la hora de mandar alguna respuesta.

- Jison

  Jison se utilizo para construir los analizadores léxicos y sintácticos del proyecto. Con ayuda de Jison se hizo posible recibir la información de manera ordenada para que funcione el analizador semántico. El archivo generado por Jison es el de gramatica.js y es este el que se encarga de leer la información de entrada recibida por la solicitud del front-end. Para construir la lógica que el analizador debe seguir, se elaboró el archivo gramatica.jison el cual contiene la definición de la gramática que se aceptará. Para construir el archivo gramatica.js se debe ejecutar el siguiente comando:

  ```
  npm run jison
  ```

Controlador

El controlador es el método que se ejecuta cuando se hace una solicitud a la única ruta disponible en el proyecto. Este recibe en el cuerpo de la solicitud el contenido de la

entrada que se require analizar y ejecutar. Lo primero que hace es mandar a analizar la entrada con ayuda del archivo gramatica.js y como resultado se obtiene un arreglo con todas las instrucciones detectadas. Después se recorre el arreglo para guardar las declaraciones de todos los métodos encontrados. Luego se recorre otra vez el arreglo para evaluar si existe una instrucción "start with", si se encuentra una entonces la ejecución del código empezará por ahí. Una vez finalizado el procedimiento de ejecución del código, se habrá recolectado todos los símbolos encontrados, así como errores semánticos y posibles salidas a la consola. Se recaudan todos los datos y se devuelven como respuesta de la solicitud http.