T6

Lenguaje y Autómatas I

06/06/2022

Actividad 5

Repositorio GITHUB
https://github.com/Ing-JuanMata/LYA_SMARTH_HEALTH

Video proyecto final
https://drive.google.com/file/d/1X45ES7zaEjcLCJtCmX_VBmjJuYQF03nE/view?usp=sharing

T6 – Actividad 5. Reporte final del compilador.

U6 – Máquinas de Turing.

Lenguaje y Autómatas I

Ibarra Carlos Francisco

Integrantes

Guerra Castillo Osmar Aldahir

Gutiérrez Aguirre Jesús Daniel

López Quintero Daniel

Santos Moya Guillermo Hilario

Villegas Samartin Ximena

Duran Sánchez Juan Manuel

Mata Solís Juan Jesús

1. Preliminares

1.1. Agradecimientos

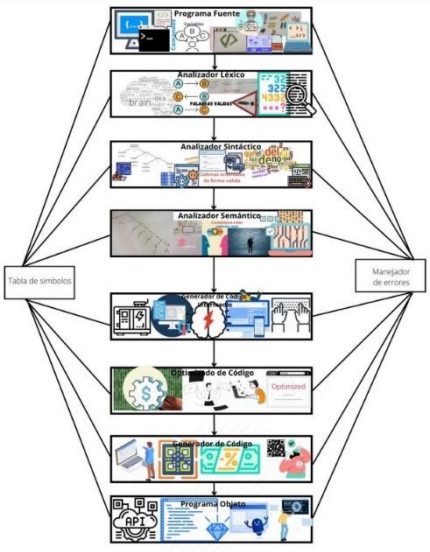
En primer lugar, deseamos expresar una muestra de agradecimiento al Mtro. Dr. Francisco Ibarra, por la dedicación y apoyo que se nos ha brindado a lo largo del desarrollo de nuestro proyecto. Por el respeto a las sugerencias e ideas propuestas además por la dirección y rigor que ha facilitado a las mismas. Gracias por la confianza ofrecida desde el inicio de este proyecto además de los conocimientos compartidos.

Un trabajo de investigación es siempre fruto de las, proyectos y esfuerzos previos que corresponden a otras personas las cuales nos ayudaron a mejorar nuestro proyecto con las ideas proporcionadas. Así mismo nos agradecemos a nosotros mismos por el trabajo en equipo realizado siendo que a pesar de las distintas formas de pensar, trabajar y desarrollar se la podido lograr hacer y terminar el proyecto.

1.2. Resumen

Se desarrollo un lenguaje con nombre “Smart Health”, consiste en funciones enfocadas al cumplimiento de protocolos de sanidad publica establecidos dentro de lugares cerrados con el fin de facilitar el cumplimiento de estas.

1.3. Colash de imágenes



Contenido

2. Generalidades del proyecto

2.1. Introducción

En el siguiente documento esta presentado el desarrollo realizado con nombre “Smart Health”. Este surge de la necesidad de hacer cumplir las medidas de sanidad pública presentadas en la contingencia de la pandemia del COVID-19. Las funciones se centran en medir la temperatura, alertar cuando se incumpla la cantidad de personas en un área cerrada, verificar el cumplimiento de la sana distancia y, además, aportaran un valor agregado a las medidas que se llevan a cabo en el país a nivel nacional.

2.2. Descripción de la empresa y área de trabajo

The project is centered within the ITTEPIC campus located in Tepic, Nayarit, Mexico. Its use as a work area in the classrooms or rooms located within the campus that are used by teachers or students is contemplated.

2.3. Problemas que resolver

Actualmente el instituto tecnológico de Tepic (ITTEPIC) cuenta con el equipo básico de prevención para el virus Covid-19, esto es dispensadores de gel en cada uno de los edificios así como también un termómetro en la entrada para comprobar que los alumnos están en condiciones para poder entrar al plantel, sin embargo medidas como la ventilación y la sana distancia entre los alumnos no está verificada del todo dentro de las aulas, dicha verificación o validación se limita a lo que el docente en turno determine permisivo en cuanto a la distancia y distribución de los alumnos dentro del aula en cuestión. Otra de las situaciones que se presentan dentro de las instalaciones es que cada aula es desinfectada unas pocas veces durante el día, la desinfección rutinaria después de cada clase está sujeta a que los alumnos y docente se tome el tiempo de rociar el líquido recomendado sobre toda aquella superficie utilizada tales como mesas y sillas, situación que por supuesto el cuerpo

estudiantil y docente no suele realizar al 100% por prisa de abandonar el aula e ir a seguir con la siguiente labor académica.

2.4. Objetivos

2.4.1. General

Create a mobile application for the student and administrative community of the Technological Institute of Tepic, which, through a voice assistant, allows control of the opening of doors, windows, as well as the turning on of fans and disinfection systems for classrooms without the need for touching physical items that may be potential carriers of the Covid-19 virus.

2.4.2. Específicos

- Asegurar la seguridad sanitaria en las aulas y las oficinas
- Garantizar el proceso de limpieza para los alumnos y personal del Instituto Tecnológico de Tepic previo al ingreso de las aulas u oficinas
- Prevenir saturación de aulas y asegurar el distanciamiento entre los alumnos durante actividades académicas

2.5. Justificación

Podemos definir que el control de sanidad es el conjunto de acciones preventivas que lleva a cabo el estado, para normar y controlar las condiciones sanitarias del hábitat humano, los establecimientos, las actividades, los productos, los equipos, los vehículos y las personas que puedan representar riesgo o daño a la salud de la población en general, así como a fomentar a través de prácticas de repercusión personal y colectiva, la protección a la salud. Hoy en día estamos atravesando un hecho mundial que nos afecta a todos en gran medida, tanto así que si nos exponemos y no nos resguardamos o tomamos las medidas necesarias para prevenir infectarse del virus, en este caso, se trata del coronavirus SARS-COV 2.

Apareció en China en diciembre pasado y provoca una enfermedad llamada COVID-19, que se ha extendido por el mundo y fue declarada pandemia global por la Organización Mundial de la Salud, gracias a este virus podríamos hasta perder la vida. Todos podemos hacer algo para mitigar y contener la expansión de las enfermedades. Es por esto, que, debido a esta problemática, hemos decidido enfocar nuestro proyecto a esta área tan importante como lo es el control de sanidad, gracias a esto podremos asegurar acciones tales como la distancia entre las personas en espacios cerrados y el acceso seguro a lugares de trabajo y estudio en la institución.

3. Marco teórico

Los coronavirus son una familia de virus que causan enfermedades (desde el resfriado común hasta enfermedades respiratorias más graves) y circulan entre humanos y animales.

En este caso, se trata del SARS-COV2. Apareció en China en diciembre pasado y provoca una enfermedad llamada COVID-19, que se extendió por el mundo y fue declarada pandemia global por la Organización Mundial de la Salud.

https://coronavirus.gob.mx/covid-19/

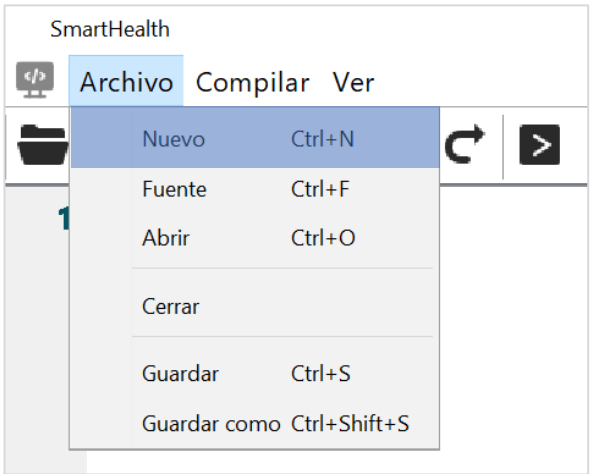
A raíz de esto hemos decidido crear y diseñar un lenguaje basado en JAVA que nos permite realizar ciertos filtros de sanidad para la universidad ITTEPIC, con el fin de mantener un lugar más seguro y libres de contagios dentro de las aulas del ITTEPIC, este permite sanitización del aula y de alumnos, ventilación, control de alumnos dentro del aula, toma de temperatura, automatización de apertura y cierre de puertas y ventanas y entre otras más funciones que se encuentran dentro del lenguaje, todo esto controlado por un sistema de voz a través de comandos.

4. Desarrollo

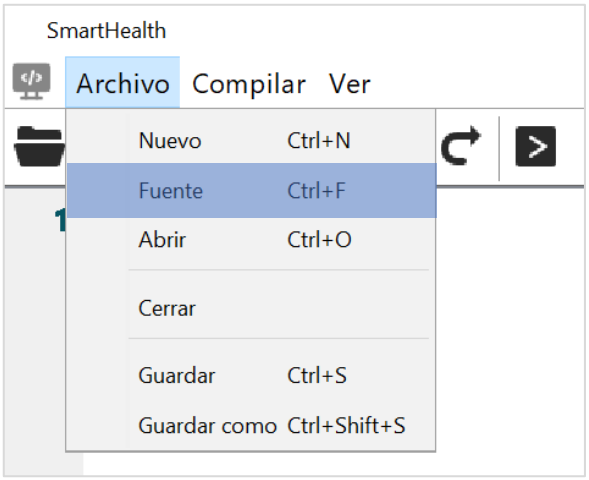
4.1. Manual de Usuario

Esta es la pantalla principal de Smart Health

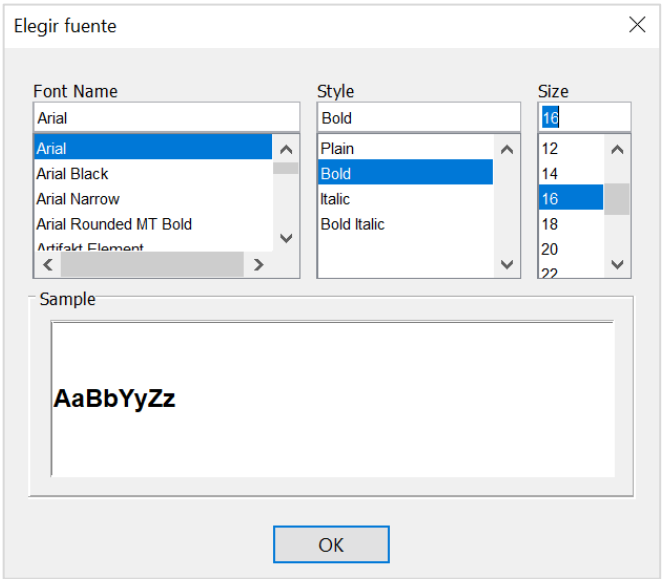
Al dar clic en la primera pestaña superior izquierda llamada “Archivo”, nos desplegará opciones, la primera opción “Nuevo”, el cual nos permite crear un nuevo archivo, para abrirlo también podemos utilizar Ctrl+N.



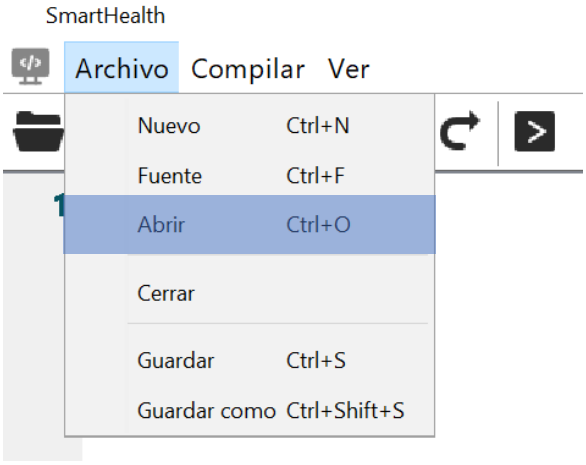
La segunda opción “Fuente” nos permite cambiar el estilo de la letra que aparece donde estamos escribiendo nuestro código, para abrirlo también podemos utilizar Ctrl+F.



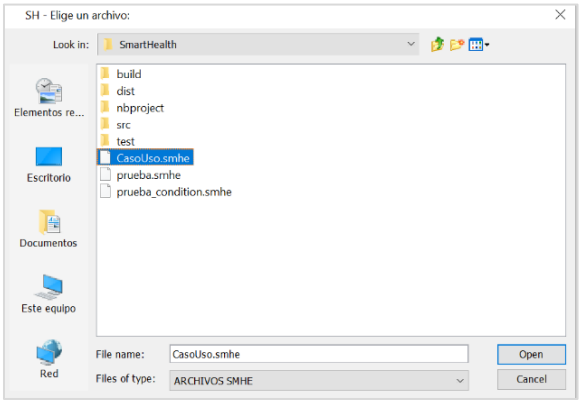
Nos aparece otra ventana y ahí las opciones “Font Name” la cual nos permite elegir entre muchas fuentes, después “Style” en la cual elegimos si queremos que nuestra letra sea: Plain, Bold, Italic o Bold Italic; en “Size” podemos elegir el tamaño de nuestra letra.Finalmente, en la parte inferior tenemos “Sample” en donde nos aparece un ejemplo de la letra que queda al final de elegir como la queremos.



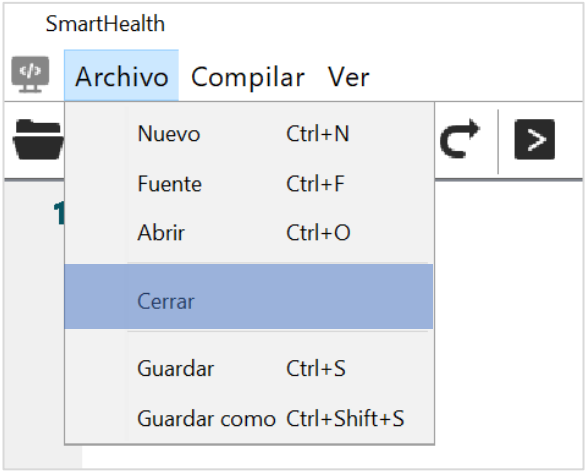
La tercera opción de la sección de “Archivo” es “Abrir”, como su nombre lo dice nos permite abrir un archivo generado con nuestro código (.smhe).



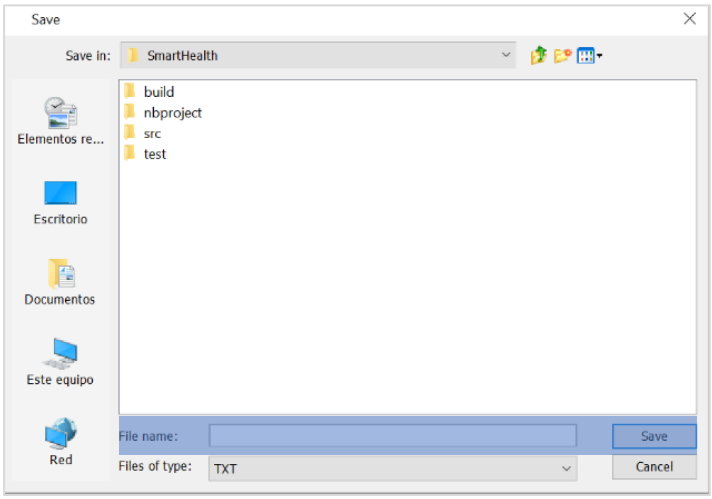
Nos abre una ventana donde podemos ver todos los documentos generados con nuestro lenguaje .smhe



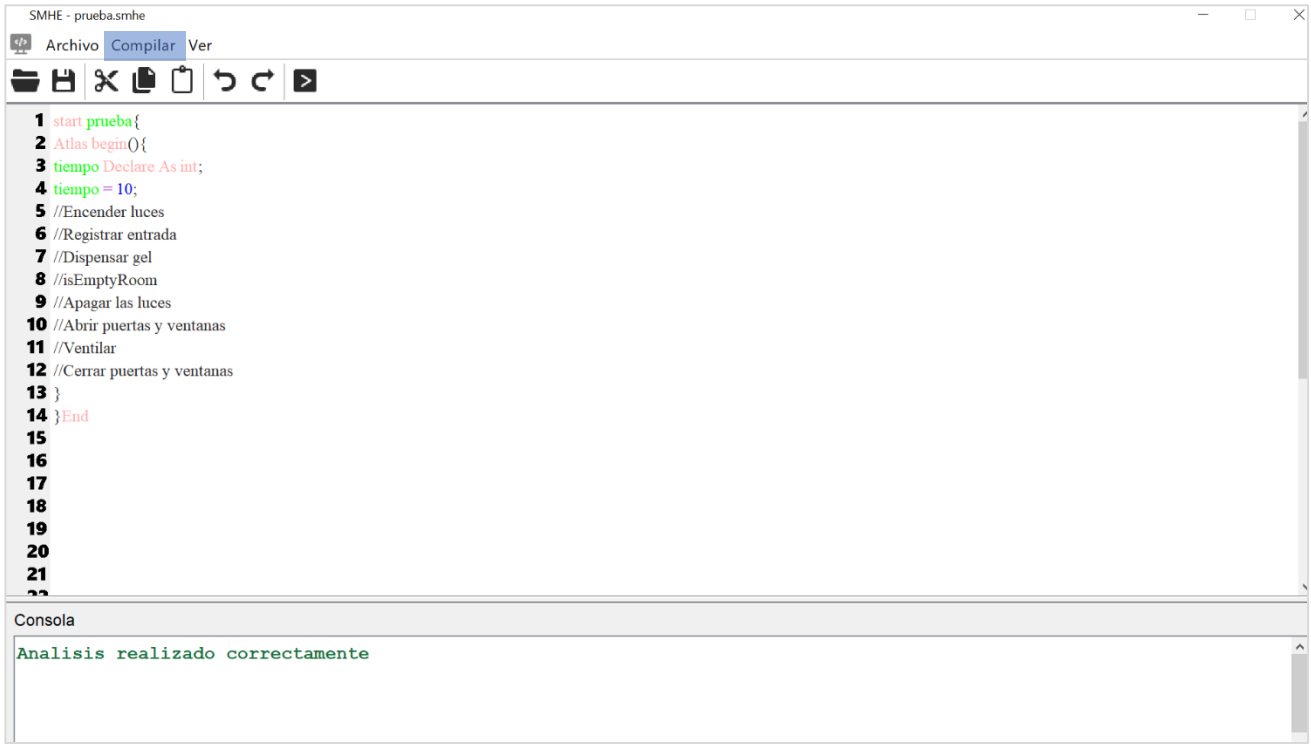
La cuarta opción de la sección de “Archivo” es “Cerrar”, como su nombre lo dice nos permite cerrar nuestro lenguaje.



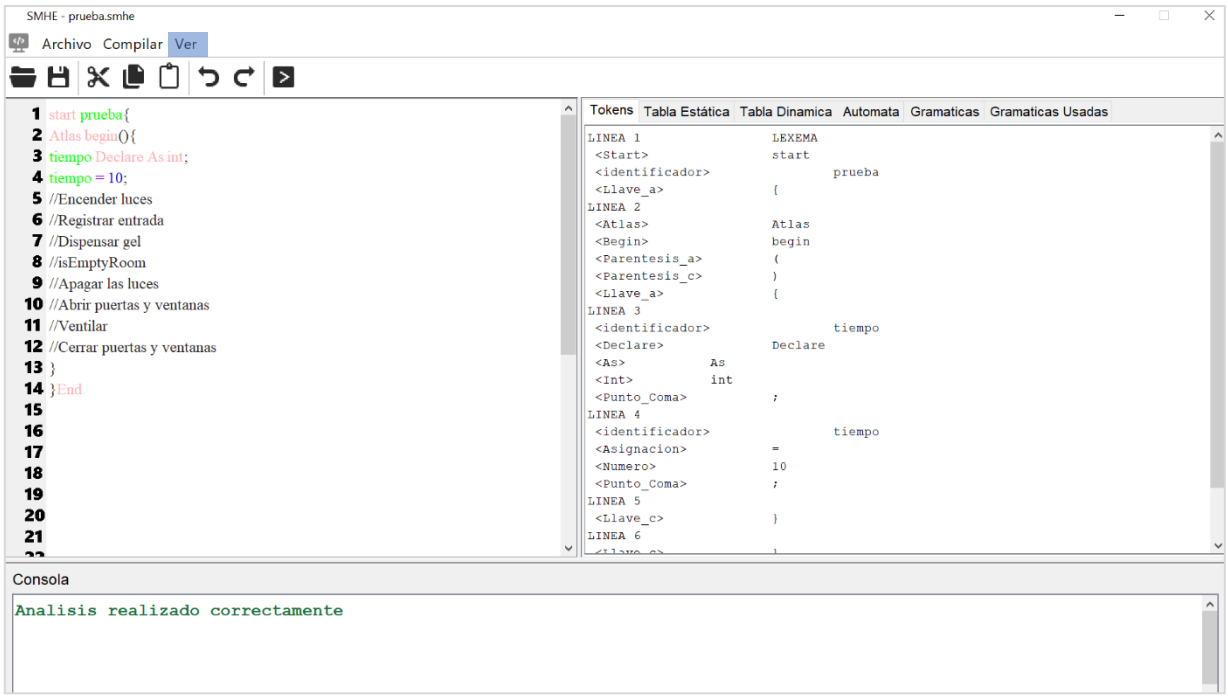
Como quinta y sexta opción de la sección de “Archivo” tenemos “Guardar” y “Guardar cómo”, al dar clic a una de estas dos nos abre una ventana la cual nos permiten guardar nuestro código realizado con el nombre que queramos.



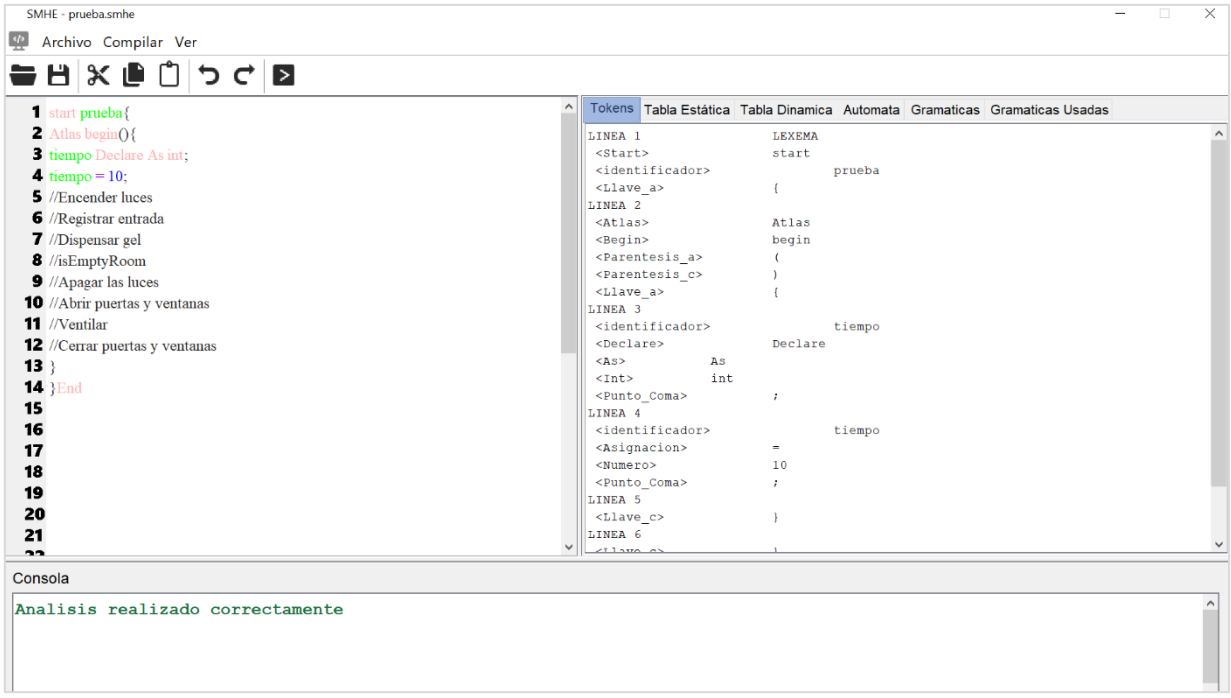
La siguiente opción de nuestra barra principal es “Compilar”, compila nuestro programa y nos muestra su nuestro análisis es correcto o incorrecto.



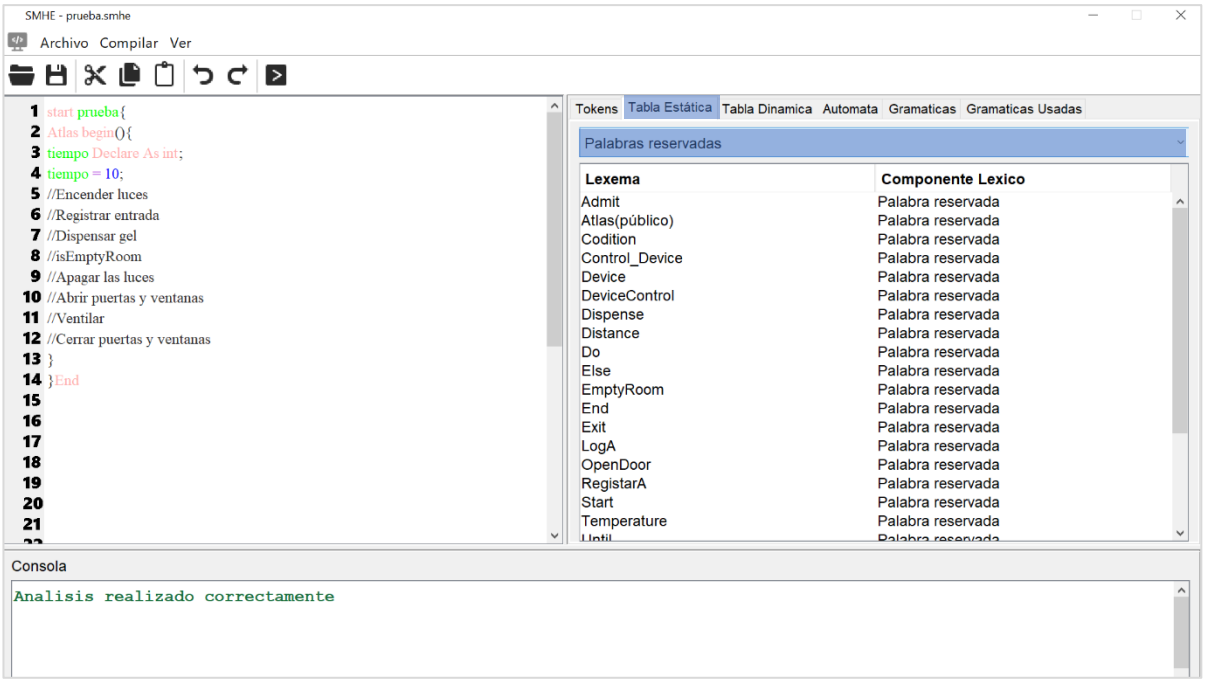
En nuestra última opción de nuestra barra principal está “Ver” el cual nos muestra: Tokens, Tabla Estática, Tabla Dinámica, Autómata, Gramáticas y Gramáticas Usadas, del lado derecho de nuestra pantalla.



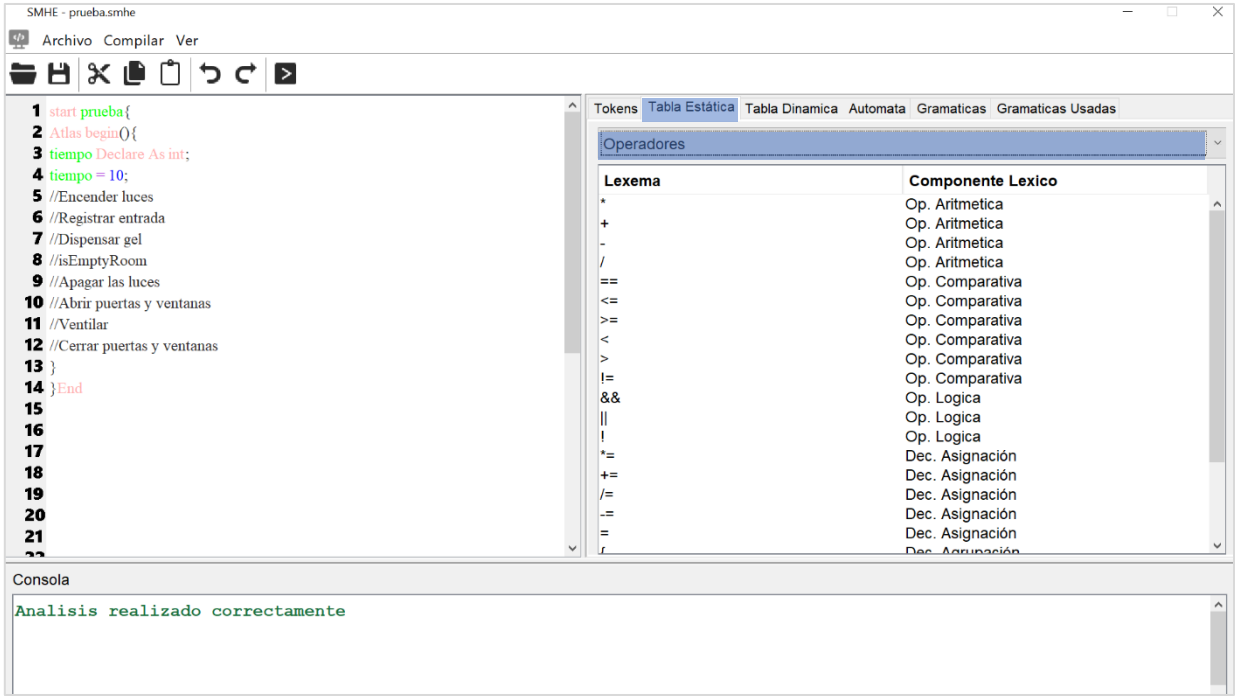
En el área de Tokens nos muestra los tokens que hay por cada línea.



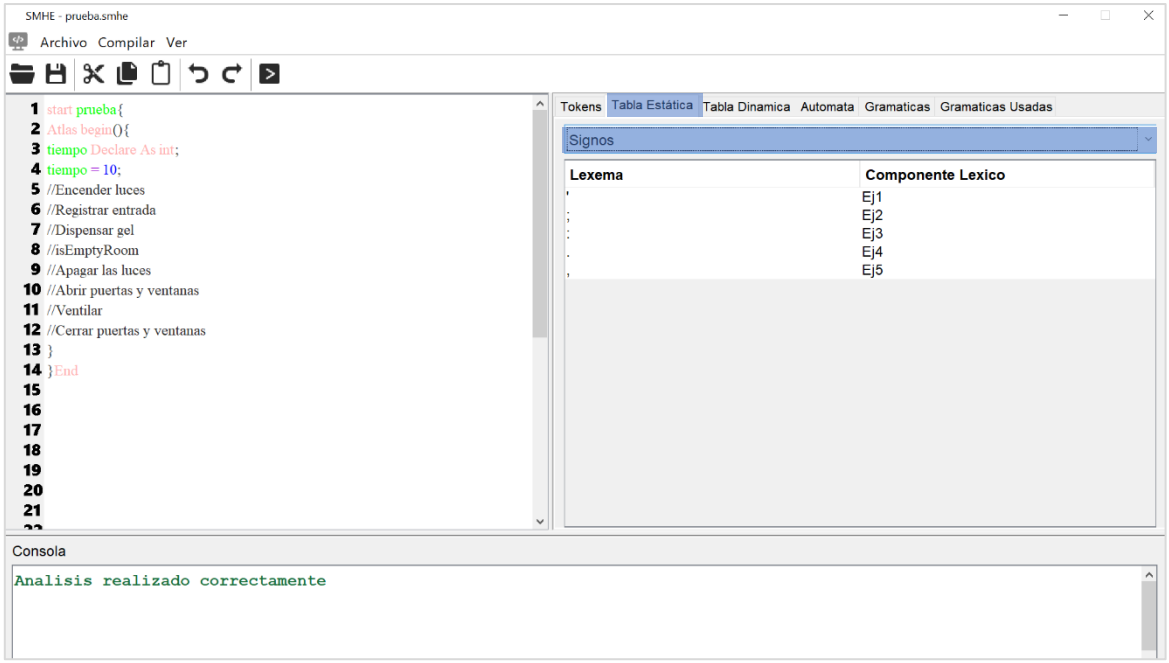
En el apartado “Tabla Estática” tenemos un ComboBox el cual nos despliega la primera opción “Palabras reservadas” y nos muestra el Lexema con su respectivo Componente



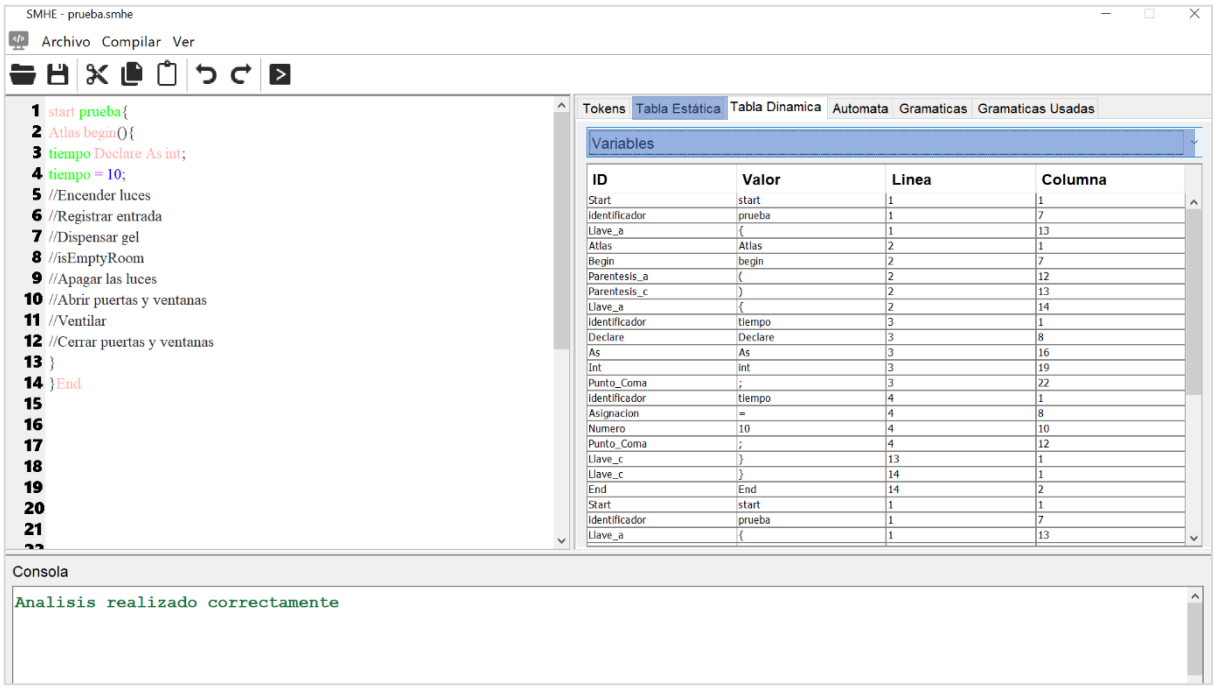
En el apartado “Tabla Estática” tenemos un ComboBox el cual nos despliega la segunda opción “Operadores”, el cual nos muestra el Lexema con su respectivo Componente Léxico.



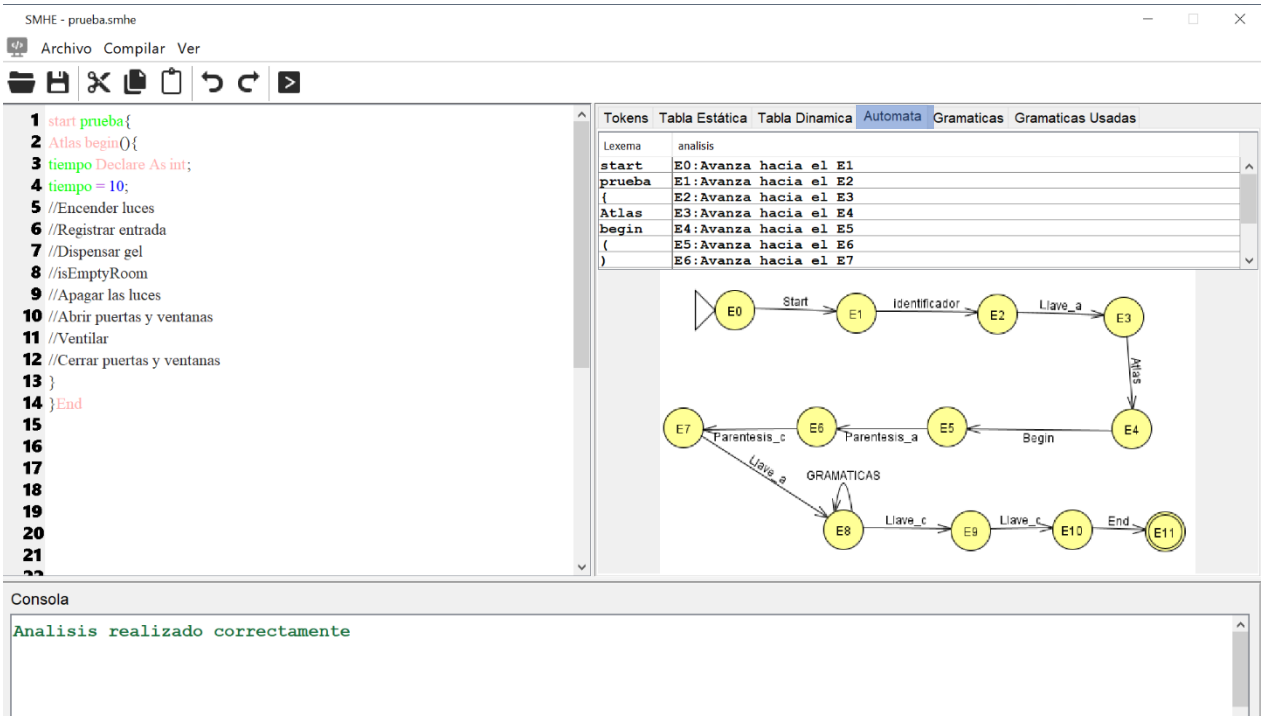
En el apartado “Tabla Estática” tenemos un ComboBox el cual nos despliega la segunda opción “Signos”, el cual nos muestra el Lexema con su respectivo Componente Léxico.



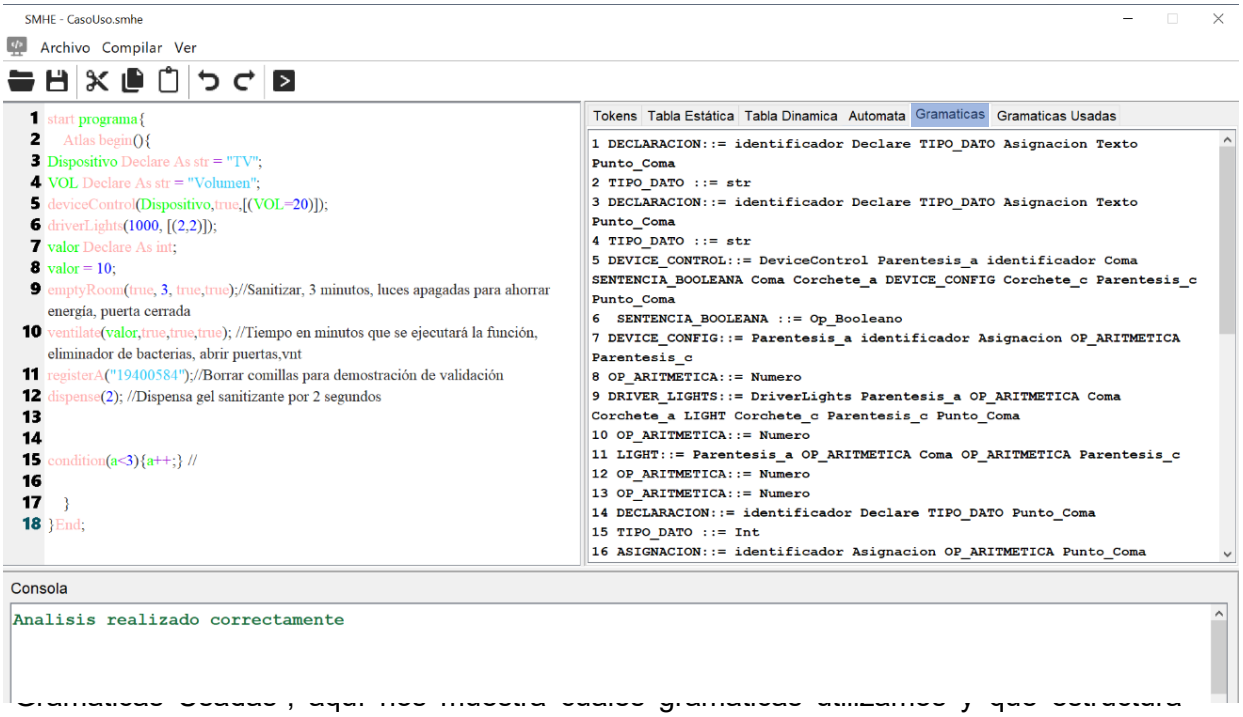
En “Tabla Dinámica” nos muestra las Variables que tenemos en nuestro código, el valor y en qué línea y columna se encuentran.



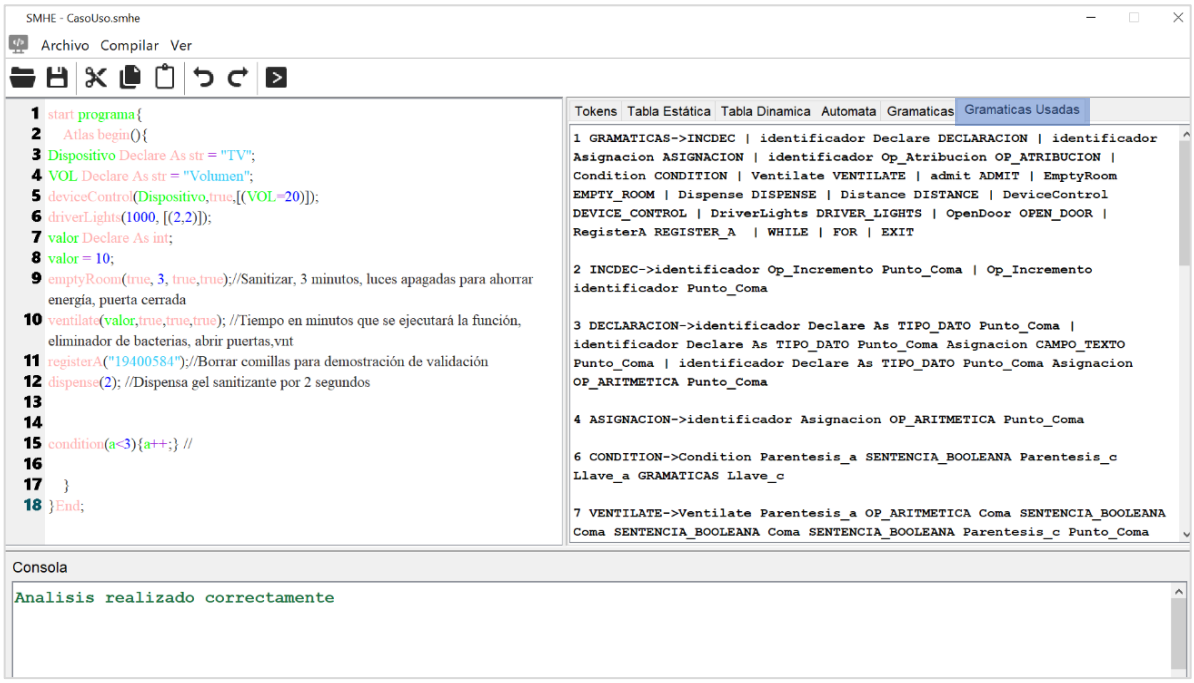
En el apartado “Autómata”, nos muestra en una tabla el lexema y el análisis, que queda mejor explicado con el autómata que nos muestra el cómo debe de ser la estructura.



“Gramáticas” no muestra todas las gramáticas de nuestro código.



Gramáticas Usadas”, aquí nos muestra cuáles gramáticas utilizamos y que estructura puede tener cada una de ellas.



En nuestra segunda barra tenemos iconos, los cuales nos facilitan aún más nuestra interacción, por ejemplo, el primer botón con el icono de carpeta nos permite abrir un documento .smhe.

El segundo botón nos permite guardar un archivo

El tercer botón nos permite cortar texto para después pegarlo en otra parte de nuestro código.

El cuarto botón nos permite copiar texto.

El quinto botón nos permite pegar texto.

El sexto botón (flecha de regreso) nos permite regresar al texto después de modificarlo.

El séptimo botón nos permite regresar al texto modificado.

El octavo y ultimo botón nos permite compilar nuestro código.

4.2. Manual técnico

4.2.1. Requerimientos técnicos

- Procesador: Intel Pentium III 800MHz
- Memoria RAM: 2 GB, 4 GB (2 GB para sist. 32 bits y 4 GB para sist. 64 bits).
- Disco duro: 2 GB.
- Tarjeta gráfica: GPU integrada en CPU o discreta.

4.2.2. Requerimientos mínimos de software

- Privilegios de administrador.
- Sistema operativo: Windows 10.

4.2.3. Herramientas utilizadas

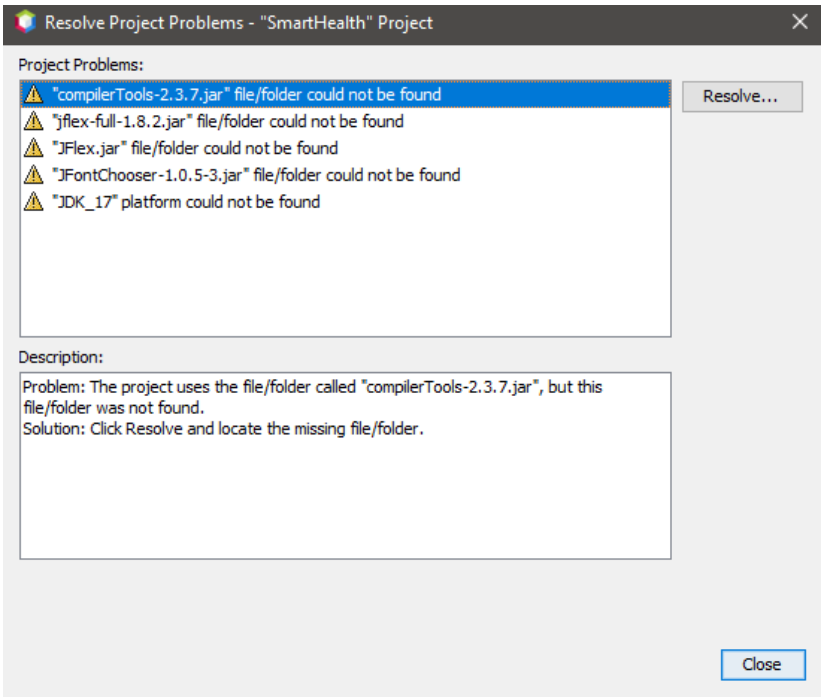
- Netbeans: NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.
- Java: Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.

4.2.4. Instalación

Para la instalación de Smart Health en un ambiente de desarrollo es necesario seguir los siguientes pasos:

1. Descargar e instalar Netbeans, de preferencia la versión más reciente para evitar problemas.
2. Descargar el proyecto desde el repositorio de github (https://github.com/Ing-JuanMata/LYA_SMARTH_HEALTH).
3. Una vez instalado el proyecto lo descomprimimos del zip en que fue descargado y lo movemos a un directorio deseado (de preferencia a la carpeta de documentos).

4. Abrimos netbeans y damos click en: **Archivo > Abrir proyecto**. (También podemos hacer uso del atajo **Ctrl + Shift + O**).
5. Al abrir el proyecto será necesario resolver algunos problemas de compatibilidad de versión de java además de agregar las librerías necesarias. Para esto hacemos clic derecho sobre el nombre del proyecto localizado en el menú izquierdo y buscamos la opción de “Resolver problemas del proyecto” o “Resolve project problems”.
6. Una vez hecho el paso anterior nos saldrá la siguiente ventana



Se nos pediré agregar las librerías faltantes, dichas librerías se ubican en la siguiente carpeta/dirección:

```
LYA_SMARTH_HEALTH-main\LYA_SMARTH_HEALTH main\CODIGO_FUENTE\SmartHealth\dist\lib
```

Y seguir los pasos indicados

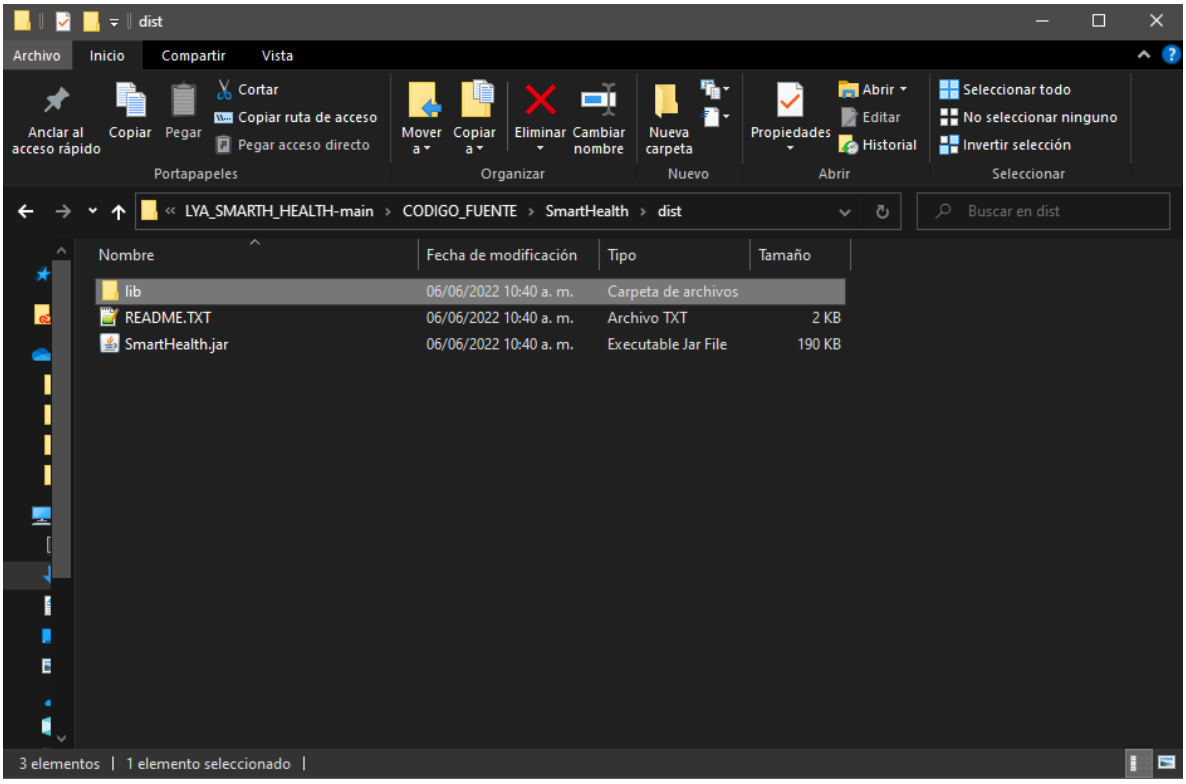
7. Habiendo completado todos los pasos anteriores ya podremos compilar/editar el proyecto.

4.3. Correr desde ejecutable

Una vez limpiado y compilado el proyecto desde el IDE Netbeans podremos encontrar un archivo .jar que corresponde al mismo. Este se ubica en:

`LYA_SMARTH_HEALTH-main\LYA_SMARTH_HEALTH main\CODIGO_FUENTE\SmartHealth\dist`

El archivo .jar tendrá el nombre de SmartHealth.jar



5. Resultados

Resumen del video

Para los tokes mandamos a validar nuestras estructuras complejas donde tenemos el condition el cual es una palabra reservada que debe cumplir sus parámetros este condicional es parecido a un if/else if, while y for, esto nos permite validar estas estructuras de control básicas adaptadas a nuestro lenguaje, así como verificar los caracteres esperados y que sean válidos para que no se produzca un error, dentro de nuestro programa tenemos los autómatas el cual nos muestra de manera textual y gráfica de manera estática, al igual nos muestra los paso a paso del autómata, este se genera una vez compilado el programa, al igual como interviene un error al momento de ejecutar en este.

Para la parte de las gramáticas se muestran una vez compilado el código, esté muestra todas las gramáticas existentes dentro de nuestro lenguaje al igual que también sus parámetros para que sea válida la gramática y sus validaciones, al igual tenemos la interfaz de gramáticas usadas o que se ejecutaron en el código puesto, esta nos mostrará las gramáticas usadas y sus parámetros necesarios para que sea válida nuestra gramática, al igual que nos muestra los errores de parámetros.

A la hora de compilar nuestro programa nos permite visualizar nuestras distintas tablas como es la de Tokens, donde nos muestra la descripción de las líneas y que tipo de dato es, como palabras reservadas, identificadores o palabras/símbolos válidas dentro de este, también nos muestra una tabla estática el cual nos muestra todas nuestras palabras reservadas,operadores y signos y que componente léxico pertenece, al igual nos muestra la tabla dinámica, esta nos muestra todos nuestros identificadores, su valor y en qué línea se muestra, también la tabla de de automatas, gramaticas y gramáticas usadas

Dentro del código se muestra cómo se programó el autómata, este tiene una estructura para seguir una secuencia que es necesaria para que sea correcto, este se hace

con tokens y validados para poder avanzar, en caso de ser válido arroja un boolean “true” al ser incorrecto a la validación o secuencia este retornara el token, una vez esté validado y sea correcto este continuará con el identificador y así hasta que no se encuentren más tokens.

Al igual para el código de la gramática funcionan por tokens, el cual se encarga de realizar los pasos dependiendo al token dependiendo del tipo de token es como si se ignora o continua con el proceso.

6. Conclusiones

En conclusión, este proyecto nos deja muchas enseñanzas ya que aplicamos todo lo que vas a lo largo del semestre como son autómatas gramática Y procesos para llevar a cabo este proyecto además de eso nos dejan nos deja el aprendizaje del trabajo en equipo por lo tanto fue fundamental para realizar este esté compilador Al igual también tuvimos nuevos aprendizajes dentro de lenguaje aplicado dentro del programa y sus nuevas implementaciones que se implementaron.

Estamos muy contentos ya que se alcanzaron todas las metas esperadas y logramos implementar nuevas funciones que nos esperaban en un principio del proyecto. Con el estudio que se realizó nos ha dado la oportunidad de descubrir nuevos métodos y librerías del lenguaje utilizado

7. Competencias desarrolladas

1 introducción a la Teoría de Lenguajes Formales

- 1.1 Alfabeto.
- 1.2 Cadenas.
- 1.3 Lenguajes, tipos y herramientas.
- 1.4 Estructura de un traductor
- 1.5 Fases de un compilador

2 expresiones Regulares.

- 2.1. Definición formal de una ER.
- 2.2. Diseño de ER.
- 2.3. Aplicaciones en problemas reales.

3 autómatas Finitos.

- 3.1 Conceptos: Definición y Clasificación de Autómata Finito (AF).
- 3.2 Conversión de un Autómata Finito No Determinista (AFND) a Autómata Finito Determinista (AFD).
- 3.3 Representación de ER usando AFND
- 3.4 Minimización de estados en un AF
- 3.5 Aplicaciones (definición de un caso de estudio).

4 análisis Léxico.

- 4.1 Funciones del analizador léxico.
- 4.2 Componentes léxicos, patrones y lexemas.
- 4.3 Creación de Tabla de tokens.

4.4 Errores léxicos.

4.5 Generadores de analizadores Léxicos.

4.6 Aplicaciones (Caso de estudio).

5 análisis Sintáctico.

5.1 Definición y clasificación de gramáticas.

5.2 Gramáticas Libres de Contexto (GLC).

5.3 Árboles de derivación.

5.4 Formas normales de Chomsky.

5.5 Diagramas de sintaxis

5.6 Eliminación de la ambigüedad.

5.7 Tipos de analizadores sintácticos

5.8 Generación de matriz predictiva (cálculo first

y follow)

5.9 Manejo de errores

5.10 Generadores de analizadores sintácticos

8. Fuentes de información

- TechTarget Contributor. (2016). compiler. Recabado febrero 9, 2022, de WhatIs.com sitio web: https://whatis.techtarget.com/definition/compiler#:~:text=A%20compiler%20is%20a%20special,that%20a%20computer%27s%20processor%20uses.&text=The%20programmer%20then%20runs%20the,that%20contains%20the%20source%20statements
- un. (2015). 1.7. Fases de un compilador - TEORIA DE LENGUAJES FORMALESS. Recabado febrero 9, 2022, de Google.com sitio web: https://sites.google.com/site/teoriadelenguajesformaless/1-7-fases-de-un-compilador
- 1.5. Tipos de Compiladores. (2022). Recabado febrero 9, 2022, de Uaeh.edu.mx sitio web: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro32/15_tipos_de_compiladores.html
- Monus, A. (2020, noviembre 10). Los 13 principales Lenguajes de Scripts a los que Deberías Prestar Atención en 2022. Recabado febrero 9, 2022, de Kinsta sitio web: https://kinsta.com/es/blog/lenguajes-script/
- Águila, J. (n.d.). Recabado de: https://kataix.umag.cl/~jaguila/Compilers/T01_Fases_Compilador.pdf
- un. (2015). 1.7. Fases de un compilador - TEORIA DE LENGUAJES FORMALESS. Recabado febrero 9, 2022, de Google.com sitio web: https://sites.google.com/site/teoriadelenguajesformaless/1-7-fases-de-un-compilador
- Unknown. (2022, febrero 10). 1.7 Fases de un compilador. Recabado febrero 9, 2022, de Blogspot.com sitio web: http://lenguajesyautomatasitsh.blogspot.com/2015/02/17-fases-de-un-compilador.html
- Glosario informático - Definición de términos informáticos. (2022). Recabado febrero 9, 2022, de Glosarioit.com sitio web: https://www.glosarioit.com/Cadena

9. Glosario

AFN: Es un autómata finito que, a diferencia de los autómatas finitos deterministas (AFD), posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición $\delta(q,a)$ posible.

Alfabeto: Conjunto ordenado de sus letras.

Analizador léxico: Es un módulo destinado a leer caracteres del archivo de entrada, donde se encuentra la cadena a analizar, reconocer subcadenas que correspondan a símbolos del lenguaje y retornar los tokens correspondientes y sus atributos

Autómata finito determinista: Es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

Cadena: Es una secuencia ordenada de elementos que pertenecen a un cierto lenguaje formal o alfabeto análogas a una fórmula o a una oración.

Carácter: Es una unidad de información que corresponde aproximadamente con un grafema o con una unidad o símbolo parecido, como los de un alfabeto o silabario de la forma escrita de un lenguaje

Cerradura de Kleene: se refiere a la operación unitaria de lenguajes que identifica a la concatenación sucesiva de ninguna o más veces de todas y cada una de las cadenas que conforman al lenguaje en cuestión.

Clases: Grupo de elementos de un conjunto que tiene características comunes.

Código objeto: Conjunto de instrucciones y datos escritos en un lenguaje que entiende el ordenador directamente.

Componente Léxico: Es un pequeño análisis sintáctico del tipo de lexema al que se está haciendo referencia.

Conjunto: Agrupación de personas, animales o cosas considerados como un todo homogéneo, sin distinguir sus partes.

Depuración: Es el proceso de identificar y corregir errores de programación.

Diagrama de transiciones: Muestra el comportamiento dependiente del tiempo de un sistema de información

Diagrama: Representación gráfica de las variaciones de un fenómeno o de las relaciones que tienen los elementos o las partes de un conjunto.

DOS: Sistema operativo basado en consola para computadoras personales.

Ecuación: Igualdad entre dos expresiones que contiene una o más variables.

Estado: Es un modelo computacional que realiza cómputos en forma automática sobre una entrada.

Expresión regular: Es una cadena de caracteres que es utilizada para describir o encontrar patrones dentro de otras cadenas, en base al uso de delimitadores y ciertas reglas de sintaxis.

Expresión: Es la representación, con palabras o con otros signos externos, de un pensamiento, una idea, un sentimiento, etc.

GFNCH: Todo lenguaje independiente del contexto que no posee a la cadena vacía, es expresable por medio de una gramática en forma normal de Chomsky (GFNCH) y recíprocamente. Además, dada una gramática independiente del contexto, es posible algorítmicamente producir una GFNCH equivalente, es decir, que genera el mismo lenguaje.

Gramática Equivalente: Dos gramáticas que describan el mismo lenguaje se llaman gramáticas equivalentes.

Gramáticas Libres de Contexto: es una gramática formal en la que cada regla de producción es de la forma:

Identificador: Es un símbolo que se utiliza para formar un nombre.

Instrucción: Es el conjunto de datos en secuencia estructurada o específica que el procesador interpreta y, posteriormente, ejecuta.

Lema de Arden: El Lema de Arden, en lenguajes formales, indica una solución particular a la ecuación con expresiones regulares.

Lenguaje estructurado: Es un lenguaje natural limitado en palabras y construcciones, lo que le da más precisión y claridad, evitando ambigüedades.

Lenguaje fuente: Es el código escrito en algún lenguaje de programación previo a ser interpretado o compilado.

Lenguaje universal: cualquier sistema de signos que permite componer e interpretar mensajes.

Lenguaje vacío: El lenguaje que es aceptado por un autómata finito se llama lenguaje regular

Lenguaje: Es la capacidad propia del ser humano para expresar pensamientos y sentimientos por medio de la palabra.

Lenguaje: Es la capacidad propia del ser humano para expresar pensamientos y sentimientos por medio de la palabra

Lexema: es la palabra que se escribe en el lenguaje de programación previo a compilar. Es el componente que es pasado como token.

Longitud: Dimensión de una línea o de un cuerpo considerando su extensión en línea recta.

Operador aritmético: Los operadores aritméticos realizan operaciones matemáticas, como sumas o restas con operandos.

Operador lógico: Los operadores lógicos permiten combinar más de una prueba relacionar en una comparación. Los operadores lógicos devuelven el valor TRUE (1) o FALSE (0). Los operadores lógicos tienen una menor precedencia que los operadores aritméticos.

Operador relacional: Los operadores relacionales son: < Menor que, > Mayor que, <= Menor o igual que, >= Mayor o igual que.

Operadores Atribución: Los operadores de este tipo permiten guardar en una variable el valor de otra variable, una expresión o una constante, +=, -=, *=, /=, %=.

Operadores Booleanos: Forman la base de los conjuntos matemáticos y la lógica para la búsqueda en las bases de datos, (true, false).

Operadores Incremento y decremento: Los Operadores de incremento y decremento son operadores unarios que agregan o sustraen uno de sus operandos, respectivamente. ++, --.

Palabra reservada: Es una palabra que tiene un significado gramatical especial para ese lenguaje y no puede ser utilizada como un identificador de objetos en códigos de este, como pueden ser las variables.

Patrón: Es una expresión regular que nos muestra las cadenas válidas para ciertas frases.

Prefijo: Combinación de cifras que se añade a los números de teléfono de una zona, ciudad o país para distinguirlos de los de otro lugar**.**

Regla: Principio que se impone o se adopta para dirigir la conducta o la correcta realización de una acción o el correcto desarrollo de una actividad

Símbolo: Es una entidad abstracta, que no se va a definir. Normalmente los símbolos son letras (a,b,c,… z), dígitos (0,1,2…9) y otros caracteres (+,*,/,-,?…).

Simplificar: Consiste en escribirla de la forma más sencilla posible.

Subcadenas: Una subcadenas es una secuencia continua de caracteres dentro de una cadena.

Teorema: Proposición matemática demostrable a partir de axiomas o de proposiciones ya demostradas.

Tiempo de ejecución: Es el intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo, este tiempo se inicia con la puesta en memoria principal del programa, por lo que el sistema operativo comienza a ejecutar sus instrucciones.

Tipo de dato: Es un atributo de los datos que indica al ordenador sobre la clase de datos que se va a manejar.

Token: Son palabras reservadas de un lenguaje.