Jonatan Josue Vasquez Pastor
202007092
ingeniería en ciencias y sistemas.
LENGUAJES FORMALES Y DE PROGRAMACION
Sección A

Proyecto 1 LFP

Índice

Introducción

La empresa “Grafos Guatemala” busca innovar en el campo del análisis y visualización de grafos mediante la implementación de un software avanzado. Este software deberá ser capaz de analizar archivos de entrada con una sintaxis específica y generar grafos utilizando la herramienta Graphviz. Para llevar a cabo esta tarea, la empresa ha contratado mis servicios como desarrollador para crear esta aplicación.

El desafío radica en desarrollar una aplicación que no solo cumpla con los requisitos técnicos de análisis y generación de grafos, sino que también ofrezca una interfaz gráfica intuitiva y accesible para el público general. Además, es crucial que el software maneje adecuadamente los errores léxicos que los usuarios puedan cometer, proporcionando retroalimentación clara y precisa.

Objetivos del Software

- La aplicación debe contar con las siguientes funcionalidades clave:
- Cargar archivo: El usuario debe poder seleccionar un archivo con extensión ".code" que será cargado en la memoria del sistema, listo para ser procesado.
- Ejecutar Archivo: Esta función analizará el archivo de entrada y generará las imágenes de los grafos descritos en el archivo. Es esencial validar que se haya cargado un archivo previamente, de lo contrario, se deberá notificar al usuario sobre la necesidad de cargar un archivo de entrada antes de proceder.
- Visualización de imágenes: Tras generar las imágenes de los grafos, la aplicación debe permitir la visualización de estas imágenes dentro de la interfaz, facilitando así la interpretación y análisis por parte del usuario.
- Reportes: La aplicación debe proporcionar un área dedicada a la generación de reportes, los cuales serán exportados en formato HTML. El usuario debe poder elegir la ubicación para almacenar estos reportes. Los reportes incluirán:
- Reporte de tokens: Detallando el nombre del token, lexema, fila y columna.
- Reporte de errores: Incluyendo el token en proceso de reconocimiento, lexema, fila, columna y el carácter del error.

Soluciones

Desarrollar un software de análisis y generación de grafos para "Grafos Guatemala" implica abordar varios desafíos técnicos y de experiencia de usuario. A continuación, se amplía la información sobre los desafíos identificados y las soluciones propuestas para cada uno.

1. Interfaz gráfica intuitiva

Desafíos:

- **Diseño de la interfaz**: La interfaz debe ser clara y fácil de usar, con una estructura que permita a los usuarios realizar todas las acciones necesarias sin confusión.
- **Accesibilidad**: Asegurarse de que la aplicación sea accesible para usuarios con diferentes niveles de habilidad técnica y necesidades especiales.
- **Feedback al usuario**: Proveer retroalimentación inmediata y útil a las acciones del usuario, especialmente en caso de errores.

Soluciones:

- **Prototipos y pruebas de usabilidad**: Crear prototipos de la interfaz y realizar pruebas con usuarios para asegurar que la navegación sea intuitiva. Utilizar herramientas de diseño de interfaz como Figma o Adobe XD.
- **Organización clara**: Agrupar funcionalidades relacionadas (como cargar y ejecutar archivos) en secciones claramente etiquetadas y visibles. Utilizar iconos y colores distintivos para cada sección.
- **Mensajes de error informativos**: Implementar mensajes de error detallados que expliquen claramente qué salió mal y cómo puede corregirse. Utilizar pop-ups o áreas dedicadas en la interfaz para estos mensajes.

2. Manejo de errores léxicos

Desafíos:

- **Detección de errores**: Implementar un analizador léxico que pueda identificar y reportar una amplia variedad de errores en el archivo de entrada.
- **Reporte de errores detallado**: Proveer información suficiente sobre cada error para que los usuarios puedan identificar y corregir los problemas en sus archivos.

Soluciones:

- **Analizador léxico robusto**: Desarrollar o integrar una biblioteca de análisis léxico que pueda manejar la sintaxis específica del archivo .code. Esto incluye la identificación de tokens válidos y la detección de errores como caracteres inválidos o estructuras incorrectas.
- **Reporte de errores detallado**: Implementar una funcionalidad que no solo indique que hubo un error, sino que también especifique el tipo de error, el token que se estaba reconociendo, el lexema, la fila y la columna, y el carácter de error. Este reporte debe ser accesible y fácil de entender para los usuarios.

3. Integración con Graphviz

Desafíos:

- **Generación precisa de grafos**: Asegurarse de que los grafos generados a partir del archivo de entrada sean precisos y reflejen correctamente la información proporcionada por el usuario.
- **Rendimiento**: Mantener un rendimiento adecuado, incluso con archivos de entrada grandes o complejos, para evitar tiempos de espera largos que puedan frustrar a los usuarios.

Soluciones:

- **Uso de PyGraphviz o similar**: Integrar PyGraphviz (una interfaz de Python para Graphviz) para convertir los datos del archivo .code en representaciones gráficas. Asegurarse de que el archivo .dot generado sea correcto antes de pasarlo a Graphviz.
- **Optimización del rendimiento**: Implementar técnicas de optimización para el procesamiento de archivos grandes, como la generación de grafos en segundo plano o el uso de estructuras de datos eficientes. Además, utilizar el canvas de Tkinter para renderizar las imágenes generadas de manera eficiente.

4. Visualización de imágenes

Desafíos:

- **Desplazamiento y redimensionamiento**: Manejar imágenes que pueden ser más grandes que el área visible en la interfaz, permitiendo a los usuarios desplazarse y ajustar el tamaño de las imágenes.
- **Claridad y detalle**: Asegurarse de que las imágenes sean claras y detalladas, incluso cuando se redimensionan o se desplazan.

Soluciones:

- **Uso de canvas y scrollbars**: Utilizar un canvas de Tkinter junto con scrollbars para manejar imágenes grandes. Esto permite a los usuarios desplazarse por la imagen y verla en detalle.
- **Redimensionamiento de imágenes**: Implementar la funcionalidad de redimensionamiento automático de las imágenes para ajustarlas al tamaño de la ventana, con la opción de verlas en tamaño completo si es necesario.

5. Generación de reportes

Desafíos:

- **Exportación a HTML**: Implementar una funcionalidad que convierta los datos de tokens y errores en reportes HTML bien formateados y fáciles de leer.
- **Selección de ubicación de almacenamiento**: Permitir a los usuarios elegir dónde guardar los reportes generados.

Soluciones:

- **Librerías de generación de HTML**: Utilizar librerías como `html` en Python para crear documentos HTML a partir de los datos de los reportes. Asegurarse de que el formato sea claro y profesional.
- **Diálogos de archivo**: Utilizar diálogos de selección de archivo (`tkinter.filedialog`) para permitir a los usuarios seleccionar la ubicación de almacenamiento de los reportes. Proporcionar opciones predeterminadas y recordar la última ubicación utilizada
-

Requisitos mínimos

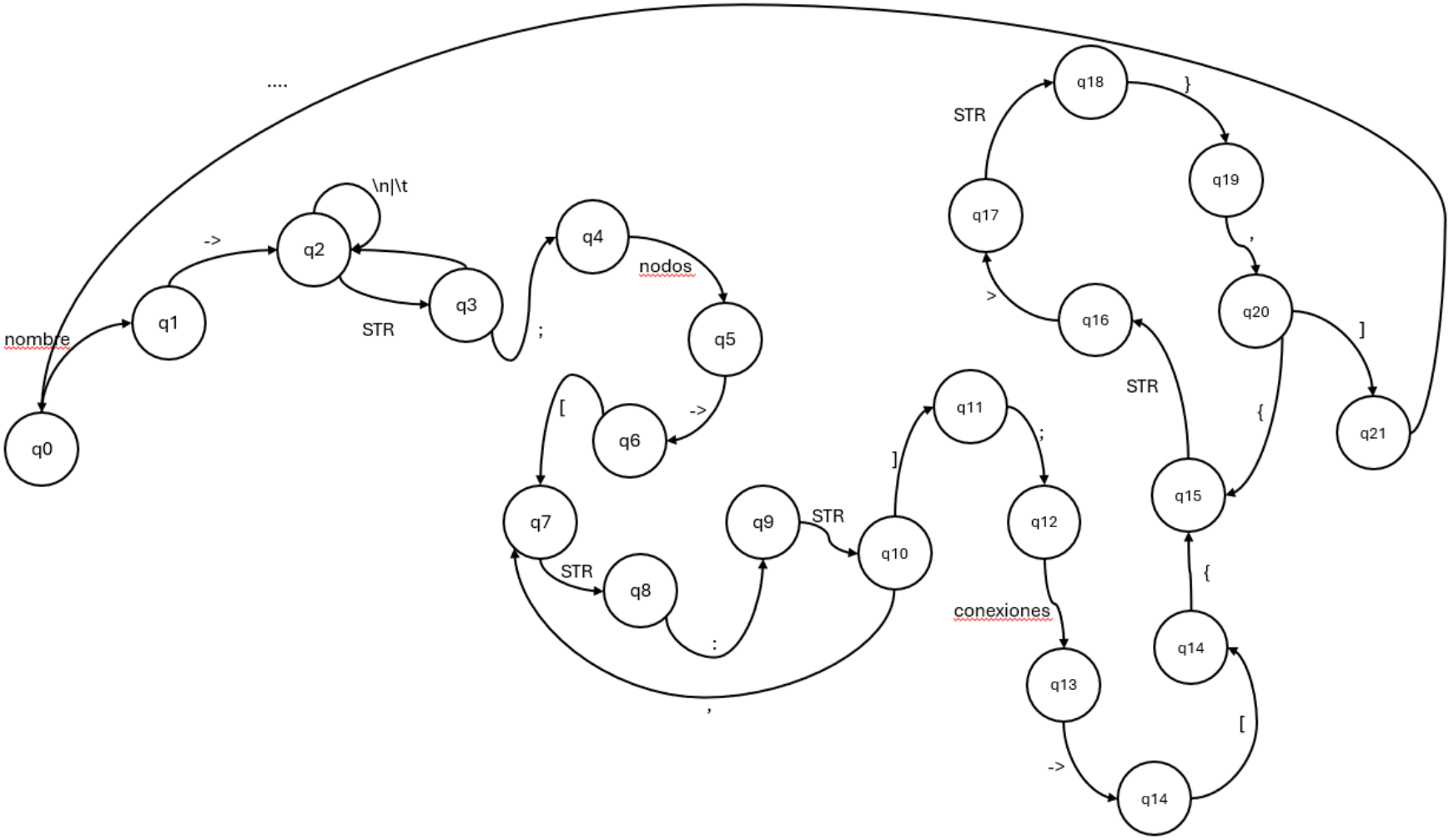
Hardware:

- Procesador de 1 GHz o superior.
- Al menos 2 GB de RAM (4 GB recomendado).
- 500 MB de espacio libre en disco.
- Tarjeta gráfica compatible con DirectX 9.
- Resolución de pantalla mínima de 1024x768 píxeles.

Software:

- Windows 10 o superior.
- Python 3.8 o superior.
- Dependencias de Python: tkinter, Pillow, graphviz, pygraphviz.
- Graphviz instalado y configurado.

Imagen del autómata utilizado



Conclusiones

Abordar estos desafíos con soluciones adecuadas garantizará que el software desarrollado para “Grafos Guatemala” sea robusto, fácil de usar y eficiente. La combinación de una interfaz gráfica intuitiva, un manejo preciso de errores, una integración eficaz con Graphviz, y capacidades avanzadas de visualización y generación de reportes permitirá a los usuarios aprovechar al máximo las funcionalidades del software, mejorando su experiencia y productividad en el análisis y visualización de grafos.