

Título del Proyecto: Analisis y diseño de algoritmos

Universidad de Caldas

Fase: Frontend

Participantes:

- Jeronimo Toro C.
- Marlon Stiven A.

Tabla de Contenidos

- [Introducción](#)
- [Definición de Herramientas de Desarrollo](#)
- Etapa de Análisis
 - Análisis de Requerimientos
 - Casos de Uso
 - Casos de Prueba
- Etapa de Diseño
 - Diseño de Datos
 - Diseño de Arquitectura
 - Diseño de Interfaz
 - Diseño de Software
- Etapa de Desarrollo
 - Configuración del Ambiente de Desarrollo
 - Desarrollo en Python
 - Creación de Documentación Técnica
- Etapa de Pruebas
 - Pruebas de Integración
 - Pruebas de Requerimientos
 - Pruebas de Estrés
- Etapa de Implementación
 - Creación de Ejecutable
 - Documentación de Instalación
 - Manual de Usuario
- Documentación de Desarrollo
 - Estructura del Proyecto
 - Componentes Desarrollados

1. Introducción

El proyecto de Análisis y Diseño de Algoritmos se propone como una solución integral para el estudio, diseño, y manipulación de grafos, dirigido a una amplia audiencia que incluye estudiantes, académicos, y profesionales interesados en la teoría de grafos y sus aplicaciones prácticas. A través de una serie de fases interconectadas, este proyecto abarca el desarrollo de una aplicación web que servirá como herramienta educativa y de investigación, facilitando la visualización de algoritmos y la resolución de problemas complejos de optimización.

En su núcleo, el proyecto busca simplificar la comprensión de conceptos avanzados en teoría de grafos, ofreciendo una plataforma intuitiva y accesible para la experimentación y el análisis. A medida que el proyecto avance a través de sus diversas fases, desde el diseño y desarrollo del frontend hasta la implementación de lógicas de backend complejas, se incorporarán funcionalidades y herramientas avanzadas para enriquecer la experiencia del usuario y ampliar las capacidades de la aplicación.

2. Definición de Herramientas de Desarrollo

Construcción y Empaquetado: Se utiliza Vite como nuestra herramienta de construcción y empaquetado, destacando por su rapidez y eficiencia en el desarrollo de proyectos modernos de JavaScript. Vite aprovecha las capacidades del módulo nativo ES para un recargado rápido y un empaquetado optimizado.

Framework de Desarrollo Frontend: React, en su versión 18.2.0, es el framework elegido para el desarrollo del frontend. Combinado con TypeScript y la compilación SWC, React nos permite construir una interfaz de usuario reactiva y eficiente, con tipado estático para mejorar la calidad del código y facilitar el mantenimiento.

Gestión del Estado: Redux se implementa para la gestión del estado global de la aplicación, permitiendo un flujo de datos predecible y una fácil integración con React para el manejo de estados complejos a lo largo de la interfaz de usuario.

Estilización y Diseño de Interfaz: La combinación de Tailwind CSS y Next UI forma la base de nuestro sistema de diseño. Tailwind CSS facilita la creación de diseños personalizados y responsivos con una sintaxis de clases utilitarias, mientras que Next UI proporciona un conjunto de componentes de React estilizados y listos para usar, que se integran perfectamente con Tailwind.

Rutas y Navegación: React Router es la librería seleccionada para gestionar la navegación dentro de nuestra aplicación SPA (Single Page Application), ofreciendo un sistema robusto y flexible para el manejo de rutas.

Llamadas a API y Manejo de Datos Externos: Axios se utiliza para realizar solicitudes HTTP, dada su interfaz sencilla y capacidad para trabajar con promesas, facilitando la comunicación con el backend y el manejo de datos externos.

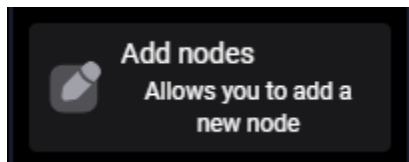
Control de Versiones y Colaboración: Git, junto con GitHub, forma el ecosistema de control de versiones y colaboración, permitiendo un desarrollo iterativo y colaborativo entre los miembros del equipo, además de facilitar la integración y despliegue continuos (CI/CD).

3. Etapa de Análisis

3.1 Casos de Uso

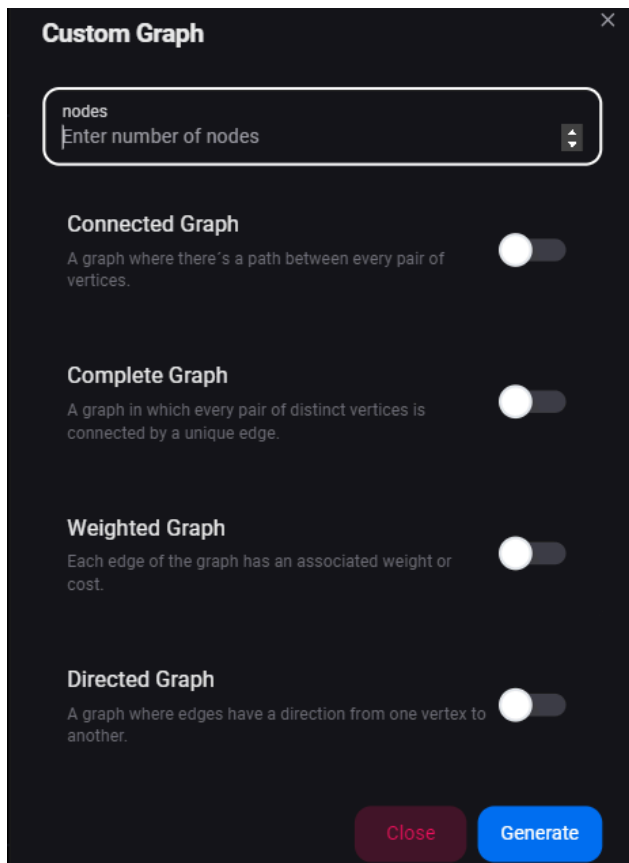
- **CU01: Crear Grafo Personalizado**

- El usuario crea un grafo personalizado, especificando nodos y aristas manualmente.



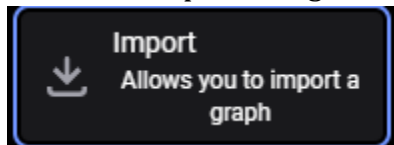
- **CU02: Crear Grafo Aleatorio**

- El sistema genera un grafo aleatorio basado en parámetros definidos por el usuario.



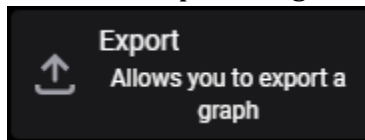
- **CU03: Importar Grafo**

- El usuario importa un grafo desde un archivo de texto.



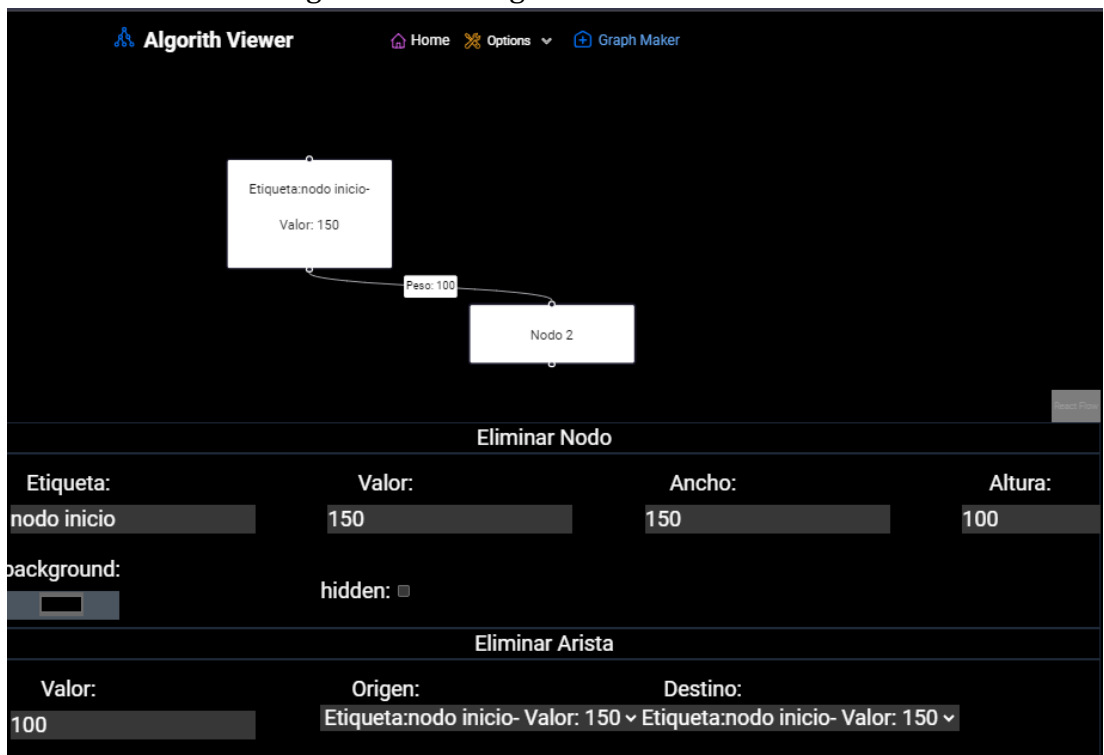
- **CU04: Exportar Grafo**

- El usuario exporta el grafo actual en formatos como JSON, Excel o imagen.



- **CU05: Visualizar Grafo:**

- El usuario visualiza el grafo en modo gráfico o como matriz.



3.2 Casos de Prueba

- **CP01: Verificación de Creación de Grafo Personalizado**

- Probar que el usuario puede crear un grafo personalizado correctamente.

- **CP02: Verificación de Generación de Grafo Aleatorio**

- Probar que el sistema genera grafos aleatorios acorde a los parámetros definidos por el usuario.

- **CP03: Verificación de Importación de Grafo**

- Probar que los grafos se pueden importar correctamente desde archivos de texto.
- **CP04: Verificación de Exportación de Grafo**
 - Probar que el grafo se puede exportar correctamente en los formatos especificados.
- **CP05: Verificación de Visualización de Grafo**
 - Probar que el usuario puede visualizar grafos tanto en modo gráfico como en matriz sin errores.

4. Etapa de Diseño

4.1 Diseño de Datos

(Descripción de la estructura de datos utilizada para el manejo de grafos y otras funcionalidades de la aplicación.)

4.2 Diseño de Arquitectura

(Descripción del diseño arquitectónico de la aplicación, incluyendo patrones de diseño y la estructura de componentes.)

4.3 Diseño de Interfaz

(Descripción y esbozos del diseño de la interfaz de usuario, incluyendo la navegación y la experiencia del usuario.)

4.4 Diseño de Software

(Descripción de la estructura general del software, incluyendo módulos, librerías y dependencias.)

5. Etapa de Desarrollo

5.1 Configuración del Ambiente de Desarrollo

(Descripción de la configuración del ambiente de desarrollo, incluyendo herramientas y versiones.)

5.2 Desarrollo en Python

(Detalles sobre el desarrollo de la lógica de backend en Python, incluyendo algoritmos y la integración con el frontend.)

5.3 Creación de Documentación Técnica

(Proceso para la creación de documentación técnica sobre el código y la arquitectura de la aplicación.)

6. Etapa de Pruebas

6.1 Pruebas de Integración

(Descripción de las pruebas de integración realizadas para asegurar la compatibilidad entre diferentes partes del sistema.)

6.2 Pruebas de Requerimientos

(Detalles sobre las pruebas realizadas para validar que la aplicación cumple con los requerimientos especificados.)

6.3 Pruebas de Estrés

(Descripción de las pruebas de estrés realizadas para evaluar el rendimiento y la estabilidad de la aplicación bajo carga.)

7. Manual de Usuario

8. Documentación de Desarrollo

(Contenido existente)