UNIVERSIDAD
CATÓLICA
BOLIVIANA

Plan de Pruebas

Campero Morales José Antonio

Campohermoso Berdeja Oscar

Carrasco Cespedes Miguel Alejandro

Martinez Acarapi Fabiola Alejandra

Montero Garrido Diana Aneliz

Zizold Sempertegui Gabriela Zulema Britta

Universidad Católica Boliviana

SIS-312: Gestión de Calidad de Sistemas

Lic. Cecilia Alvarado Monrroy

15 de diciembre de 2024

Índice

Índice de figuras

Índice de tablas

Introducción

Este plan de pruebas tiene como objetivo definir y organizar las actividades de prueba para los módulos principales de la aplicación *Bajo Esquina*. Esta aplicación es un editor de grafos que ofrece una variedad de herramientas avanzadas para la creación, manipulación y análisis de estructuras de grafos, así como funcionalidades orientadas a la resolución de problemas reales en el ámbito de los grafos y las estructuras de datos.

Entre sus características destacadas, la aplicación incluye un módulo optimizado para calcular rutas óptimas en la red de teleféricos de la ciudad de La Paz, Bolivia. Este módulo está diseñado para ayudar a los usuarios a optimizar trayectos considerando criterios como costo y tiempo. Además, la aplicación incluye diversas funcionalidades para la generación de grafos personalizados, la representación de matrices de adyacencia y costos, y una interfaz amigable para facilitar la interacción con sus herramientas.

El propósito de este plan de pruebas es asegurar que los módulos desarrollados de distintos algoritmos implementados cumplan con los requisitos funcionales, ofrezcan una experiencia de usuario fluida y proporcionen resultados precisos y eficientes. Este enfoque busca garantizar la calidad del sistema mediante pruebas que evalúen su funcionalidad, robustez y usabilidad, asegurando que cumpla con los objetivos establecidos y satisfaga las expectativas de los usuarios finales.

Alcance del Plan de Pruebas

El presente plan de pruebas plantea evaluar de manera integral el sistema desarrollado, abarcando múltiples módulos y funcionalidades clave, con el fin de garantizar su correcto funcionamiento, accesibilidad, usabilidad y cumplimiento de los objetivos especificados. A continuación, se detallan los alcances específicos de las pruebas:

Áreas incluidas en las pruebas

- **Creación y visualización de grafos:** Se evaluará la capacidad del editor para permitir la creación precisa de grafos, incluyendo la adición, edición y conexión de nodos y aristas, asegurando que los grafos generados se representen de manera

adecuada en el entorno gráfico del editor.

- **Guardado y carga de grafos:** Se comprobará que los datos de los grafos se almacenen y recuperen correctamente, permitiendo la continuidad del análisis y la manipulación en futuras sesiones.
- **Generación de matrices:** Las pruebas cubrirán:
  - **Matriz de adyacencia:** Verificar que refleje con precisión las conexiones y relaciones entre los nodos del grafo.
  - **Matriz de costos:** Evaluar su construcción precisa basada en los nodos y sus conexiones, garantizando que represente correctamente las relaciones de costos entre puntos de suministro y demanda.
- **Optimización:** Se revisará la funcionalidad del módulo de optimización, asegurando que pueda resolver problemas de maximización y minimización de manera consistente con los objetivos definidos.
- **Interfaz de usuario y usabilidad:** Se evaluará la facilidad de uso y la accesibilidad de la interfaz desarrollada en Vue.js, evaluando que cumpla con los criterios de accesibilidad de la WCAG versión 2.2 y que los resultados del sistema sean presentados de forma clara y comprensible para los usuarios.
- **Algoritmos implementados:** Se evaluará la correcta ejecución y funcionalidad de los siguientes algoritmos:
  - **Algoritmo de Asignación:** Incluyendo la interacción entre el backend en Spring Boot y la base de datos MySQL, para garantizar que los datos se almacenen y recuperen adecuadamente.
  - **Compet:** Se verificará la funcionalidad de su dashboard de edición de grafos, permitiendo la adición de nodos y la resolución eficiente del algoritmo mediante accesos directos.

    - **Dijkstra:** Se evaluarán opciones avanzadas como la selección de nodos inicial y destino, así como la ejecución del algoritmo para minimizar o maximizar rutas.
    - **Kruskal:** Verificación de la implementación y resultados del algoritmo en el contexto del sistema de gestión de grafos.
    - **Algoritmo de Johnson:** Se comprobará la correcta implementación del algoritmo para resolver problemas de caminos más cortos en grafos dirigidos con otras funcionalidades del editor.
    - **Método de la esquina noroeste (North-West):** Se evaluará su capacidad para resolver problemas de transporte, verificando que las soluciones iniciales se calculen correctamente y se representen de manera adecuada en las matrices de costos.
- **Sorts:** Evaluar la ejecución de los algoritmos de ordenamiento (Selection Sort, Insertion Sort, Merge Sort, Shell Sort), verificando la correcta representación del proceso y su animación a diferentes velocidades.
- **Árboles Binarios:** Se probará la creación, visualización gráfica y recorridos (inorden, preorden, postorden) de los árboles binarios, garantizando que los datos se representen adecuadamente.

Áreas excluidas de las pruebas

- **Componentes externos al editor:** No se evaluarán funcionalidades relacionadas con la manipulación de archivos de grafos creados previamente o la descripción del algoritmo en la interfaz inicial, ya que se consideran fuera del alcance del editor.
- **Pruebas de estrés:** No se realizarán pruebas de desempeño bajo condiciones extremas de carga en esta fase del proyecto. Estas evaluaciones serán consideradas para iteraciones futuras.

Objetivos

Objetivo General

Garantizar la correcta funcionalidad, usabilidad y calidad de los diferentes módulos y algoritmos implementados en el sistema, validando que cumplan con las especificaciones técnicas y funcionales definidas, y que proporcionen una experiencia de usuario eficiente y minimizando de errores.

Objetivos Específicos

- Validar la precisión y correcta ejecución de los algoritmos implementados, asegurando resultados optimizados y libres de errores.
- Comprobar que la interfaz de usuario desarrollada en Vue.js sea accesible, intuitiva y cumpla con los estándares de accesibilidad WCAG 2.2.
- Confirmar la integración precisa y estable entre los componentes del sistema, como la interfaz, el backend y la base de datos.
- Detectar y documentar errores o áreas de mejora en el sistema, permitiendo su corrección antes de la implementación en producción.
- Validar la funcionalidad de módulos específicos, como el editor de grafos y la gestión de estructuras de datos, asegurando su correcto desempeño.
- Evaluar la experiencia del usuario, garantizando una interacción fluida, estable y libre de errores críticos o caídas inesperadas.

Limitaciones

El presente plan de pruebas contempla diversas limitaciones que restringen el alcance y el entorno en el cual se ejecutarán las pruebas. A continuación, se describen dichas limitaciones:

- **Datos de prueba limitados:** La disponibilidad de datos de prueba representativos es limitada, lo que podría restringir la validación a configuraciones específicas de oferta y demanda, dejando fuera algunos escenarios posibles que podrían presentarse en situaciones reales. Además, los datos de prueba utilizados son simulados y pueden no representar la diversidad de datos procesados en producción.

- **Entorno de pruebas no desplegado:** Las pruebas se realizarán en un entorno de desarrollo, sin acceso a un ambiente de staging ni a una instancia de deploy. Por lo tanto, los resultados pueden no reflejar completamente el rendimiento o la experiencia de usuario final en un entorno de producción. Esto también implica que no se realizarán pruebas de carga, estrés o escalabilidad a gran escala.

- **Exclusión de componentes específicos:** Los contenedores de *Keycloak*, *MinIO* y *Python* no serán objeto de pruebas directas. Sin embargo, cualquier incidencia en su funcionamiento que afecte a los módulos bajo prueba será documentada.

- **Funcionalidades no finalizadas:** Algunas funcionalidades del sistema están en etapas de desarrollo o modificación, como la autenticación con Google. Estas características no serán evaluadas en las pruebas actuales debido a su estado de implementación.

- **Pruebas de rendimiento excluidas:** No se realizarán pruebas de rendimiento o de carga a gran escala debido a las limitaciones del entorno de desarrollo. Las pruebas de rendimiento deberán ejecutarse en un entorno de staging o producción para garantizar su validez.

- **Pruebas de seguridad excluidas:** Este plan no contempla evaluaciones sobre la gestión de autenticación, autorización, cifrado de datos ni posibles vulnerabilidades de seguridad. Estas pruebas deben ser consideradas en futuros planes de validación.

- **Limitaciones en funcionalidades del frontend:** Las pruebas funcionales en el

frontend estarán limitadas a verificar la interacción del *dashboard* de nodos y arcos con los algoritmos de *Compet* y *Dijkstra*. Otras funcionalidades del *dashboard* y la interfaz de usuario no serán evaluadas.

- **Compatibilidad limitada de accesibilidad:** Las pruebas de accesibilidad se limitarán a elementos básicos, como la navegación con teclado y el contraste visual en componentes relacionados con los algoritmos de *Compet* y *Dijkstra*. No se evaluarán estándares adicionales de accesibilidad en otros módulos.

Base de la Prueba

Descripción del Sistema

El proyecto "Bajo Esquina.[es] un sistema web que permite a los usuarios crear, editar y visualizar grafos, así como realizar análisis a través de algoritmos clave en el ámbito de la teoría de grafos. Los usuarios pueden interactuar con el sistema a través de un editor visual, el cual les permite construir grafos al agregar, modificar o eliminar nodos y aristas. El objetivo es ofrecer una plataforma que no solo facilite la creación y manipulación de grafos, sino que también permita aplicar algoritmos relevantes, manteniendo la integridad y precisión en los cálculos y la representación visual. Este sistema está diseñado para estudiantes, profesionales y cualquier usuario interesado en la teoría de grafos y su aplicación práctica, brindando una experiencia de usuario interactiva, intuitiva y funcional.

Requisitos del Sistema

Los requisitos del sistema definen las necesidades y expectativas que los distintos módulos del sistema deben cumplir para asegurar una funcionalidad completa y efectiva. Estos incluyen:

- **Creación de cuentas y acceso seguro**: Los usuarios deberán poder realizar la creación de cuentas personales vinculando su cuenta a un correo electrónico y una contraseña segura, así como deberán poder acceder al sistema mediante un inicio de sesión seguro que gestione credenciales.

- **Creación y visualización de grafos**: El editor deberá permitir a los usuarios crear grafos, incluyendo la adición de nodos y aristas, y visualizar los grafos de forma clara e intuitiva dentro de la interfaz del editor.

- **Carga y descarga de grafos**: El editor permitirá a los usuarios cargar grafos guardados desde su computador y descargar los grafos creados para su almacenamiento local, asegurando que la información se maneje de forma precisa y sin pérdidas.

- **Generación automática de la matriz de adyacencia**: A partir de los grafos creados en el editor, este deberá generar automáticamente una matriz de adyacencia que refleje correctamente las conexiones entre los nodos definidos.

- **Generación automática de la matriz de costos**: El editor deberá generar de manera automática una matriz de costos basada en los nodos y conexiones definidas por el usuario, asegurando que la información de costos esté alineada con los datos ingresados.

- **Manejo de oferta y demanda**: El editor deberá gestionar situaciones donde la oferta y la demanda no estén balanceadas. Si es necesario, ajustará automáticamente las configuraciones para asegurar que no se produzcan errores en los cálculos de transporte.

- **Funcionalidades de optimización**: El editor deberá ofrecer opciones que permitan al usuario seleccionar entre maximización o minimización de soluciones, proporcionando resultados válidos y ajustados a los objetivos especificados en las configuraciones de entrada.

- **Resolución de grafos complejos**: El editor deberá proporcionar herramientas avanzadas para resolver grafos con características complejas, como valores extremos o negativos, y múltiples relaciones. Los algoritmos implementados deberán manejar

estas condiciones sin generar errores, asegurando que los resultados sean precisos y comprensibles. Además, el sistema deberá notificar al usuario si los datos ingresados no son válidos para el análisis.

- **Visualización interactiva de procesos de ordenamiento**: El editor deberá permitir a los usuarios observar de manera interactiva el proceso de ordenamiento de listas de datos. La visualización debe mostrar cada paso del algoritmo en tiempo real, permitiendo ajustar la velocidad según las preferencias del usuario.

- **Construcción y visualización de árboles binarios**: El editor deberá permitir a los usuarios construir árboles binarios mediante la inserción de nodos con valores definidos. Además, ofrecerá la visualización gráfica de la estructura del árbol, mostrando claramente sus niveles, conexiones y jerarquías. También deberá proporcionar opciones para realizar y mostrar recorridos preorden, inorden y postorden, asegurando que los datos se presenten de manera precisa y comprensible para el usuario.

- **Resolución de problemas de asignación**: El editor deberá permitir a los usuarios resolver problemas de asignación óptima mediante el uso de matrices de asignación. Los usuarios podrán definir manualmente o cargar los datos necesarios, y seleccionar criterios de maximización o minimización para obtener soluciones óptimas.

- **Accesibilidad del editor**: Aunque no se ha pensado en un nivel profundo de accesibilidad, se espera que el editor, como mínimo, tenga un buen contraste de colores y cumpla con un nivel básico de accesibilidad.

Especificaciones Funcionales

Estas especificaciones detallan cada función del distintos editores y el comportamiento esperado, proporcionando instrucciones claras para los desarrolladores:

- **Creación de cuentas y acceso seguro**:

- **Función**: Gestión de cuentas de usuario y accesos.
- **Requisitos funcionales**:
  - Los usuarios deberán poder registrarse proporcionando un nombre de usuario, un correo electrónico y contraseña segura.
  - El acceso al sistema deberá estar protegido por un inicio de sesión que gestione las credenciales de los usuarios.
- **Proceso**:
  - Los usuarios podrán registrarse o iniciar sesión a través de una interfaz segura.

- **Creación y visualización de grafos**:
  - **Función**: Permitir la creación y visualización de grafos.
  - **Requisitos funcionales**:
    - El editor deberá permitir a los usuarios definir nodos y conexiones con pesos dentro del editor.
    - Los grafos creados deben ser visualizados de forma intuitiva en la interfaz gráfica del editor.
    - Se notificará al usuario si hay inconsistencias en la entrada de datos para asegurar la precisión en la creación del grafo.
  - **Proceso**:
    - Al definir nodos y conexiones, el editor mostrará visualmente el grafo en tiempo real.
    - Cualquier cambio en los nodos o aristas reflejará una actualización inmediata en la visualización.

- **Carga y descarga de grafos**:

- **Función**: Permitir la carga de grafos desde el computador y la descarga de grafos creados en el editor.
- **Requisitos funcionales**:
  - El editor deberá permitir al usuario cargar grafos guardados en su computador en formato JSON, para que preserve toda la información de los nodos y conexiones.
  - El editor deberá permitir al usuario descargar los grafos creados en formato JSON, garantizando así su integridad y que se puedan volver a cargar sin pérdida de
  - El editor deberá notificar al usuario sobre el estado de carga y descarga, indicando si estas operaciones se realizaron con éxito o si se presentaron problemas.
- **Proceso**:
  - El usuario podrá cargar grafos guardados en su computador en cualquier momento y descargará grafos generados para su almacenamiento local.

- **Generación de la matriz de adyacencia**:
  - **Función**: Generación automática de la matriz de adyacencia a partir del grafo.
  - **Requisitos funcionales**:
    - El editor deberá calcular automáticamente la matriz de adyacencia que refleje las conexiones entre los nodos del grafo.
    - El editor deberá permitir al usuario visualizar la matriz de adyacencia generada.
  - **Proceso**:
    - Una vez creado el grafo, el editor generará automáticamente la matriz de adyacencia y permitirá su visualización al usuario.

    - La matriz de adyacencia deberá reflejar correctamente las relaciones entre los nodos y las conexiones.

- **Generación de la matriz de costos**:
    - **Función**: Generación automática de la matriz de costos basada en los pesos de las conexiones.
    - **Requisitos funcionales**:
        - El editor deberá permitir al usuario definir nodos y conexiones con pesos.
        - El editor calculará la matriz de costos utilizando los valores de los pesos de las conexiones.
        - La matriz de costos deberá reflejar correctamente las relaciones entre los puntos de suministro y demanda.
    - **Proceso**:
        - Al definir nodos y conexiones, el editor generará automáticamente la matriz de costos.
- **Ajuste de oferta y demanda**:
    - **Función**: Ajuste automático cuando la oferta y demanda no coinciden.
    - **Requisitos funcionales**:
        - El editor deberá añadir un nodo ficticio cuando los valores de oferta y demanda no estén balanceados.
        - El editor deberá ajustar la matriz de costos para reflejar el nodo ficticio y balancear la matriz.
        - El usuario deberá ser notificado sobre el ajuste realizado y cómo afecta a la solución.
    - **Proceso**:

    - Si hay desequilibrio entre oferta y demanda, se añadirá un nodo ficticio para balancear la matriz.
    - La matriz de costos se ajustará automáticamente para reflejar el nodo ficticio y los cambios en la solución.

- **Optimización**:
    - **Función**: Cálculo de soluciones optimizadas.
    - **Requisitos funcionales**:
        - El editor permitirá al usuario elegir entre criterios de maximización y minimización.
        - El editor calculará la solución óptima basada en el criterio seleccionado.
        - El editor mostrará la solución optimizada al usuario de forma clara y comprensible.
    - **Proceso**:
        - El usuario seleccionará un criterio de optimización y el editor calculará la solución optimizada.
        - La solución se mostrará al usuario de forma clara y comprensible, en una tabla.

- **Resolución de grafos complejos**:
    - **Función**: Proporcionar herramientas para analizar y resolver grafos que incluyen valores extremos, negativos y múltiples relaciones complejas.
    - **Requisitos funcionales**:
        - El sistema deberá soportar grafos con valores extremos y negativos, garantizando la correcta interpretación de estos.
        - Los resultados obtenidos deberán ser visualizados de manera clara, con gráficos o tablas que expliquen las soluciones generadas.

- **Proceso**:
  - El usuario ingresará los nodos y conexiones del grafo, incluyendo valores extremos o negativos.
  - El sistema analizará los datos ingresados y ejecutará los algoritmos correspondientes para encontrar soluciones viables.
  - Una vez procesados, los resultados se presentarán al usuario con explicaciones y representaciones gráficas de las soluciones.

- **Visualización interactiva de procesos de ordenamiento**:
  - **Función**: Representar visualmente el proceso de ordenamiento.
  - **Requisitos funcionales**:
    - El editor permitirá al usuario seleccionar algoritmos de ordenamiento y observar cada paso del proceso en tiempo real.
    - El usuario podrá ajustar la velocidad de la animación para un aprendizaje personalizado.
  - **Proceso**:
    - El usuario seleccionará un algoritmo de ordenamiento y visualizará el proceso paso a paso en la interfaz.
    - El sistema permitirá al usuario ajustar la velocidad de la visualización.

- **Construcción y visualización de árboles binarios**:
  - **Función**: Permitir la creación y manipulación visual de árboles binarios.
  - **Requisitos funcionales**:
    - El editor deberá permitir al usuario insertar nodos en un árbol binario y mostrar la estructura de forma clara.

    - El editor deberá permitir al usuario realizar recorridos preorden, inorden y postorden, mostrando los resultados.
  - **Proceso**:
    - El usuario ingresará valores para los nodos y observará cómo se estructura el árbol binario.
    - Los resultados de los recorridos serán mostrados directamente en la interfaz.

- **Resolución de problemas de asignación**:
  - **Función**: Proveer herramientas para resolver problemas de asignación óptima considerando criterios de maximización o minimización.
  - **Requisitos funcionales**:
    - El sistema deberá permitir la definición de matrices de asignación que representen relaciones entre tareas y recursos.
    - Los usuarios podrán ingresar manualmente los valores de la matriz o importarlos desde un archivo.
    - El sistema deberá implementar algoritmos de asignación para calcular soluciones óptimas basadas en los criterios seleccionados (maximización o minimización).
    - Los resultados deberán incluir las asignaciones óptimas y el costo total asociado, presentados de forma clara y comprensible.
  - **Proceso**:
    - El usuario definirá una matriz de asignación ingresando los datos necesarios o cargándolos desde un archivo.
    - El sistema ejecutará el algoritmo de asignación correspondiente basado en el criterio seleccionado por el usuario.

    - Una vez procesada la asignación, el sistema mostrará los resultados en una tabla con las asignaciones óptimas y el costo total.

- **Accesibilidad del editor**:
    - **Función**: Garantizar la accesibilidad básica para todos los usuarios.
    - **Requisitos funcionales**:
        - El editor deberá contar con un contraste de colores adecuado para asegurar la visibilidad de todos los elementos gráficos, especialmente en los grafos, matrices y estructuras visualizadas.
        - Los controles interactivos, como botones y menús, deberán ser lo suficientemente grandes y claros para facilitar su uso en dispositivos con diferentes tamaños de pantalla.
        - El sistema deberá proporcionar descripciones textuales (etiquetas o tooltips) en los elementos visuales clave para guiar a los usuarios en su interacción.
    - **Proceso**:
        - Al cargar el editor, los elementos visuales y controles serán evaluados automáticamente para garantizar un contraste adecuado.
        - Los tooltips o descripciones se mostrarán al pasar el cursor sobre elementos clave, ayudando a los usuarios a comprender su propósito.

Historias de Usuario

Las siguientes historias de usuario documentan los requisitos específicos y su relación con las especificaciones funcionales descritas en la sección de **Especificaciones Funcionales**. Cada historia de usuario puede consultarse en el *Anexo A*, donde se detallan los parámetros y contexto para cada funcionalidad. Estas historias serán fundamentales para la creación de casos de prueba

HU001-SignUp/SignIn-01

Describe los requisitos para que un usuario nuevo pueda registrarse proporcionando un nombre de usuario, un correo electrónico y una contraseña segura. Esta historia se relaciona con la especificación funcional de **Creación de cuentas y acceso seguro**, garantizando que el usuario pueda crear una cuenta para acceder a la aplicación. *Ver Anexo HU001-SignUp/SignIn-01*

HU002-SignUp/SignIn-02

Documenta la necesidad de proporcionar mensajes de error claros si se introduce un correo electrónico inválido o una contraseña débil. Relacionada con la especificación funcional de **Creación de cuentas y acceso seguro**, asegurando una experiencia de usuario clara al corregir errores. *Ver Anexo HU002-SignUp/SignIn-02*

HU003-SignUp/SignIn-03

Describe los requisitos para que un usuario registrado pueda iniciar sesión proporcionando su correo electrónico y contraseña. Relacionada con la especificación funcional de **Creación de cuentas y acceso seguro**, asegurando el acceso a las funcionalidades de la aplicación. *Ver Anexo HU003-SignUp/SignIn-03*

HU004-Johnson-01

Documenta los requisitos para construir un problema de ruta crítica (CPM) como un grafo con actividades en las aristas. Relacionada con la especificación funcional de **Optimización**, describiendo cómo debe implementarse el algoritmo de Johnson. *Ver Anexo HU004-Johnson-01*

HU005-Johnson-02

Define los requisitos para que el usuario pueda descargar un grafo como archivo local para continuarlo más tarde. Relacionada con la especificación funcional de **Carga y descarga de grafos**, asegurando la preservación del trabajo del usuario. *Ver Anexo HU005-Johnson-02*

HU006-GraphEditor-01

Describe los requisitos para que el usuario pueda agregar nodos y aristas, y modificar sus valores dentro del editor gráfico. Esta historia de usuario se relaciona directamente con la especificación funcional de **Creación y visualización de grafos** ya que define cómo debe construirse y modificarse el grafo en la interfaz gráfica del editor. *Ver Anexo HU006-GraphEditor-01*

HU007-AdjacentMatrix-01

Documenta la necesidad del usuario de obtener la matriz de adyacencia del grafo generado. Esta historia respalda la especificación funcional de **Generación de la matriz de adyacencia** y establece que el sistema debe mostrar las conexiones entre nodos en un formato estructurado. *Ver Anexo HU007-AdjacentMatrix-01*

HU008-FileManagement-01

Define los requisitos para guardar el grafo en formato JSON y cargarlo posteriormente con todos sus elementos intactos. Esta historia está vinculada a la especificación funcional de **Carga y descarga de grafos**, asegurando que el usuario pueda conservar su trabajo sin pérdida de información. *Ver Anexo HU008-FileManagement-01*

HU009-NorthWest-01

Documenta los requisitos para la generación automática de la matriz de costos con base en los pesos de las conexiones definidas por el usuario. Relacionada con la especificación funcional de **Generación de la matriz de costos**, esta historia establece cómo se debe reflejar la estructura de costos entre puntos de suministro y demanda. *Ver Anexo HU009-NorthWest-01*

HU010-NorthWest-02

Detalla la necesidad de manejar desequilibrios entre oferta y demanda, añadiendo un nodo ficticio cuando sea necesario. Este requisito se relaciona con la especificación

funcional de **Ajuste de oferta y demanda**, describiendo cómo el sistema debe ajustarse automáticamente para mantener el equilibrio en la matriz de costos. *Ver Anexo HU010-NorthWest-02*

HU011-NorthWest-03

Especifica la capacidad del sistema para generar soluciones válidas según diversas configuraciones y criterios de optimización. Esta historia se relaciona con la especificación funcional de **Optimización**, asegurando que los resultados sean precisos y que el usuario pueda seleccionar el criterio de optimización deseado. *Ver Anexo HU011-NorthWest-03*

HU012-Kruskal-01

Define los requisitos para que el usuario pueda aplicar el algoritmo de Kruskal en el editor de grafos para encontrar el árbol de expansión mínimo o máximo. Relacionada con la especificación funcional de **Optimización**. *Ver Anexo HU012-Kruskal-01*

HU013-Dijkstra-01

Documenta los requisitos para que el algoritmo de Dijkstra devuelva el camino más corto en el grafo utilizando la menor cantidad de recursos posibles. Relacionada con la especificación funcional de **Optimización**. *Ver Anexo HU013-Dijkstra-01*

HU014-InvalidData-01

Especifica la validación y manejo de datos no válidos en los algoritmos del sistema. Relacionada con la especificación funcional de **Creación y visualización de grafos**. *Ver Anexo HU014-InvalidData-01*

HU015-Dashboard-01

Describe los requisitos para que el dashboard se actualice dinámicamente cuando se modifiquen nodos o arcos en el grafo. Relacionada con la especificación funcional de **Creación y visualización de grafos**. *Ver Anexo HU015-Dashboard-01*

HU016-AccesibilityAndNavigation-01

Detalla los requisitos de accesibilidad para usuarios con discapacidades, asegurando que los elementos cumplan con estándares de contraste y sean intuitivos. Relacionada con la especificación funcional de **Accesibilidad del editor**, para garantizar que el sistema sea inclusivo. *Ver Anexo HU016-AccesibilityAndNavigation-01*

HU017-Compet-01

Define cómo el algoritmo Compet debe manejar grafos con valores extremos y negativos, garantizando resultados precisos. Relacionada con la especificación funcional de **Resolución de grafos complejos**. *Ver Anexo HU017-Compet-01*

HU018-AccesibilityAndNavigation-02

Documenta la necesidad de proporcionar textos de ayuda o "tooltips"que expliquen la función y pasos de uso de cada algoritmo. Relacionada con la especificación funcional de **Accesibilidad del editor**, asegurando que el usuario comprenda las funcionalidades del sistema. *Ver Anexo HU018-AccesibilityAndNavigation-02*

HU019-Sorts-01

Define los requisitos para que el usuario pueda ingresar una lista de números o cargar un archivo y ordenarlos usando diferentes algoritmos. Relacionada con la especificación funcional de **Visualización interactiva de procesos de ordenamiento**. *Ver Anexo HU019-Sorts-01*

HU020-Sorts-02

Especifica la capacidad del usuario de elegir entre diferentes algoritmos de ordenamiento (e.g., Selection Sort, Merge Sort) y observar sus pasos de ejecución. Relacionada con la especificación funcional de **Visualización interactiva de procesos de ordenamiento**. *Ver Anexo HU020-Sorts-02*

HU021-Sorts-03

Documenta la necesidad de incluir una animación paso a paso que muestre cómo los algoritmos ordenan los datos. Relacionada con la especificación funcional de **Visualización interactiva de procesos de ordenamiento**. *Ver Anexo HU021-Sorts-03*

HU022-Sorts-04

Define la funcionalidad de ajustar la velocidad de las animaciones para adaptarse a las necesidades del usuario. Relacionada con la especificación funcional de **Visualización interactiva de procesos de ordenamiento**. *Ver Anexo HU022-Sorts-04*

HU023-Sorts-05

Documenta la opción de generar automáticamente una lista de números aleatorios para probar los algoritmos de ordenamiento sin necesidad de ingresar los datos manualmente. Relacionada con la especificación funcional de **Visualización interactiva de procesos de ordenamiento**. *Ver Anexo HU023-Sorts-05*

HU024-BinaryTrees-01

Especifica los requisitos para guardar un arreglo y cargarlo en el módulo de árboles binarios, preservando los datos para análisis futuro. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU024-BinaryTrees-01*

HU025-BinaryTrees-02

Define cómo los usuarios pueden construir un árbol binario ingresando valores y visualizar su estructura de forma clara. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU025-BinaryTrees-02*

HU026-BinaryTrees-03

Documenta los requisitos para realizar recorridos preorden, inorden y postorden en un árbol binario, mostrando los resultados al usuario. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU026-BinaryTrees-03*

HU027-BinaryTrees-04

Especifica cómo los usuarios pueden construir y analizar árboles binarios sin necesidad de editar o eliminar nodos, facilitando la visualización. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU027-BinaryTrees-04*

HU028-BinaryTrees-05

Documenta la funcionalidad para centrar y ajustar automáticamente la vista del árbol binario, permitiendo una exploración completa sin desplazamientos excesivos. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU028-BinaryTrees-05*

HU029-BinaryTrees-06

Define los requisitos para guardar y cargar árboles binarios, asegurando que los usuarios puedan preservar su trabajo y continuar en futuras sesiones. Relacionada con la especificación funcional de **Construcción y visualización de árboles binarios**. *Ver Anexo HU029-BinaryTrees-06*

HU030-Asignation-01

Define los requisitos para que el sistema resuelva problemas de asignación. Relacionada con la especificación funcional de **Resolución de problemas de asignación**. *Ver Anexo HU030-Asignation-01*

HU031-Asignation-02

Especifica los requisitos para visualizar los resultados de asignación de manera clara y

comprensible. Relacionada con la especificación funcional de **Resolución de problemas de asignación**. *Ver Anexo HU031-Asignation-02*

HU032-Asignation-03

Documenta la capacidad de elegir entre maximizar o minimizar la asignación, adaptándose a las necesidades del usuario. Relacionada con la especificación funcional de **Resolución de problemas de asignación**. *Ver Anexo HU032-Asignation-03*

HU033-Asignation-04

Define la funcionalidad de seleccionar y ejecutar el algoritmo de asignación, permitiendo un cálculo eficiente. Relacionada con la especificación funcional de **Resolución de problemas de asignación**. *Ver Anexo HU033-Asignation-04*

Arquitectura del Software

Esta arquitectura del sistema es fundamental para identificar los objetos de prueba dentro del plan de pruebas. La configuración actual consiste en una mezcla de arquitectura monolítica con microservicios debido al uso de múltiples servicios externos. Para ver el diagrama de arquitectura detallado, consulte el Anexo B: Diagrama de Arquitectura del Sistema.

La interfaz de usuario, desarrollada en Vue.js, se encarga de descargar y cargar el grafo, tareas independientes del almacenamiento en MinIO. FastAPI maneja las solicitudes del modelo de transporte a través de la API ‘http://localhost:8000/transportation/‘, a la cual se le envía el grafo, la oferta, la demanda, y el criterio de optimización (minimizar o maximizar), y devuelve la solución óptima. Por otra parte, el endpoint ‘http://localhost:8081/graph/adjMatrix‘ en el servicio Spring genera la matriz de adyacencia necesaria para las operaciones de grafo.

Supuestos y Limitaciones del Proyecto de Prueba

Supuestos

- **Vigencia de autenticación:** Los tokens de autenticación utilizados en el sistema tienen una duración de 30 minutos. Esto implica que las pruebas deben contemplar este límite de tiempo para mantener la validez de los tokens y garantizar la continuidad del proceso.

- **Funcionamiento continuo del sistema:** Se asume que el sistema operará de manera ininterrumpida durante las sesiones de prueba, asegurando estabilidad operativa y disponibilidad continua mientras el sistema esté activado.

- **Conocimiento del equipo de pruebas:** El equipo de pruebas cuenta con un conocimiento básico de los algoritmos implementados, como Dijkstra y Compet, así como de su funcionamiento teórico y práctico, lo que permitirá ejecutar pruebas más comprensivas y efectivas.

- **Disponibilidad de recursos y soporte:** Se garantiza acceso a los recursos necesarios, incluyendo herramientas de prueba (como Axe DevTools para accesibilidad), documentación del sistema y el soporte de los desarrolladores durante la fase de pruebas.

- **Estabilidad de los requisitos:** Los requisitos del sistema permanecerán estables durante todo el ciclo de pruebas. No se esperan cambios significativos en el alcance del proyecto que puedan impactar la cobertura o los resultados de las pruebas.

- **Disponibilidad del entorno y módulos:** Se presupone que todos los módulos, incluidos Sorts y Árboles Binarios, están completamente desarrollados y estables, así como que el entorno de pruebas será consistente en términos de hardware y software durante todo el ciclo de pruebas.

- **Representatividad de los datos de prueba:** Los datos de prueba utilizados serán representativos de los escenarios reales de uso del sistema, asegurando que los resultados de las pruebas sean aplicables al entorno de producción.

Limitaciones

- **Entorno de pruebas simulado:** Las pruebas se realizarán en un entorno de desarrollo local, lo que implica que no se simularán condiciones de producción o de alta carga. Además, la configuración del entorno puede variar entre máquinas, afectando la consistencia de los resultados.
- **Limitación en la configuración de grafos:** La configuración de grafos a probar estará predefinida, lo que restringe la flexibilidad para evaluar variaciones personalizadas y escenarios alternativos.
- **Limitaciones en el conocimiento del código:** Dado que el equipo de QA no participó en el desarrollo, su conocimiento del código y su estructura es limitado, lo que puede impactar la identificación de defectos y la ejecución de pruebas más detalladas.
- **Tiempo y recursos limitados:** Debido a restricciones de tiempo y recursos, se priorizarán los casos de prueba críticos y las funcionalidades principales, dejando algunos escenarios secundarios fuera del alcance.
- **Datos de prueba limitados:** La disponibilidad de datos de prueba se restringe a configuraciones representativas de oferta y demanda, excluyendo casos extremos o poco comunes.
- **No escalabilidad para grandes volúmenes de datos:** Este plan no incluye pruebas de escalabilidad ni rendimiento con grandes volúmenes de datos que excedan los límites definidos para el desarrollo.

Partes Interesadas

Equipo de Desarrollo

El equipo de desarrolladores responsables de la implementación de todo el sistema está compuesto por:

- Oscar Campohermoso Berdeja
- Miguel Alejandro Carrasco Céspedes
- Oscar Menacho Silva
- Sebastián Orias Bellido

Responsabilidades:

- Implementar funcionalidades del sistema según las especificaciones técnicas y funcionales.
- Resolver defectos reportados por el equipo de QA y asegurar que las correcciones no afecten otros módulos.
- Documentar el código y los cambios realizados para facilitar futuras actualizaciones.
- Colaborar con los testers para garantizar que los entornos de prueba estén configurados correctamente.

Impacto en el Proyecto: La calidad y estabilidad del sistema dependen directamente de la experiencia y desempeño del equipo de desarrollo. Su capacidad para implementar soluciones rápidas y efectivas es crucial para cumplir con los objetivos del proyecto.

Equipo de Pruebas (QA)

El equipo de QA tiene la responsabilidad de garantizar que el sistema cumpla con los requisitos funcionales y de calidad establecidos. Este equipo está compuesto por seis testers que trabajaron de manera colaborativa, sin un liderazgo centralizado, distribuyendo las tareas de forma equitativa y apoyándose mutuamente para cumplir los objetivos.

Miembros del Equipo:

- Campero Morales José Antonio
- Campohermoso Berdeja Oscar
- Carrasco Céspedes Miguel Alejandro
- Martínez Acarapi Fabiola Alejandra
- Montero Garrido Diana Aneliz
- Zizold Sempertegui Gabriela Zulema Britta

Responsabilidades:

- Diseñar, ejecutar y documentar casos de prueba para evaluar la funcionalidad y el rendimiento del sistema.
- Identificar y reportar defectos, clasificándolos según su severidad y priorización.
- Evaluar la interfaz del sistema para garantizar el cumplimiento de estándares de usabilidad y accesibilidad.
- Proponer mejoras para optimizar la experiencia de usuario, especialmente en términos de claridad e intuitividad de la interfaz.
- Colaborar estrechamente con los desarrolladores para validar las soluciones aplicadas a los defectos reportados.

Impacto en el Proyecto: El equipo de QA desempeña un papel clave al identificar problemas tempranos y asegurar que el producto final cumpla con las expectativas de calidad. Su enfoque colaborativo ha permitido una cobertura de pruebas más amplia y efectiva.

Usuarios Finales

Los usuarios finales del sistema son quienes interactuarán directamente con la plataforma. En este caso, los usuarios incluyen tanto estudiantes como docentes que utilizarán el sistema para fines educativos y prácticos.

Responsabilidades:

- Proporcionar retroalimentación sobre la usabilidad, funcionalidad y experiencia general del sistema.
- Identificar posibles inconvenientes o áreas de mejora durante el uso del sistema.

Impacto en el Proyecto: La retroalimentación de los usuarios finales es fundamental para garantizar que el sistema cumpla con sus expectativas y sea efectivo para los objetivos planteados.

Stakeholders Académicos

El profesor de la materia, Ing. Yamil Cárdenas, quien supervisa el uso del editor en el contexto académico. Su interés es que el editor cumpla con los objetivos educativos, permitiendo a los estudiantes experimentar con algoritmos de manera confiable e intuitiva.

Responsabilidades:

- Supervisar el uso del sistema en entornos académicos.
- Asegurar que las funcionalidades del sistema estén alineadas con los objetivos educativos.
- Proporcionar comentarios para futuras mejoras en base a las necesidades de los estudiantes y profesores.

Impacto en el Proyecto: Los stakeholders académicos influyen en la dirección del desarrollo y en la priorización de funcionalidades, asegurando que el sistema sea útil y relevante en un entorno pedagógico.

Comunicación

La comunicación de los resultados y hallazgos se llevará a cabo mediante un informe de resultados que incluirá todos los aspectos evaluados y observaciones relevantes.

Formularios y frecuencias de comunicación

Para garantizar una comunicación eficiente, se establecerán los siguientes formularios y frecuencias de comunicación:

- **Reuniones semanales:** Se realizarán reuniones de seguimiento, donde se discutirán los avances, problemas encontrados y planes de acción. (ver en Anexos).
- **Reuniones mensuales:** Se realizarán evaluaciones del progreso general del proyecto y ajustes en la estrategia o recursos. (ver en Anexos).
- **Revisión de Defectos:** Los defectos se informarán por correo electrónico, con un resumen del impacto y la prioridad asignada. (ver en Anexos).

Plantillas de documentación

Se utilizarán plantillas estandarizadas para la comunicación formal, controlando la uniformidad y claridad de los documentos. Las principales plantillas son las siguientes:

- **Plantilla de historias de usuario:** Esta plantilla define los requisitos desde la perspectiva del usuario final. Se utiliza la estructura: Çomo [tipo de usuario], quiero [acción o funcionalidad] para [beneficio o valor agregado]".
- **Plantilla de casos de prueba:** La plantilla de casos de prueba permite documentar cada una de las pruebas necesarias para validar el sistema.

- **Plantilla de reporte de defectos:** Esta plantilla se utiliza para registrar y controlar los errores o defectos identificados durante las pruebas.
- **Plantilla de reporte de riesgos:** La plantilla de reporte de riesgos permite identificar, evaluar y controlar los riesgos que podrían afectar el proyecto.

Registro de Riesgos

A continuación, se presenta un registro de riesgos que abarca tanto los riesgos del proyecto como del producto. Estos riesgos podrían impactar en la ejecución, calidad y resultados esperados del sistema.

Riesgos del Producto

Los riesgos del producto están relacionados con los requisitos funcionales y la experiencia del usuario al utilizar el sistema web. Estos riesgos pueden impactar directamente en la satisfacción del usuario final y en el rendimiento del sistema.

- **Resultados inexactos en algoritmos o cálculos complejos:**
  - **Descripción:** Los algoritmos implementados (como optimización o generación de matrices) pueden producir resultados erróneos debido a errores en el código, malas configuraciones o datos incorrectos.
  - **Consecuencias:** Resultados incorrectos podrían afectar la experiencia del usuario, generar desconfianza en el sistema y comprometer su utilidad académica o profesional.
  - **Controles:**
    - Revisiones detalladas del código y lógica de los algoritmos.
    - Validación de resultados mediante pruebas con datos representativos y casos límite.
    - Implementación de validaciones automáticas para verificar la precisión de los cálculos.

- **Fallas en la comunicación entre backend y frontend:**
  - **Descripción:** Errores en la comunicación entre las APIs del backend y el frontend pueden impedir que las funcionalidades del sistema operen correctamente, como la generación de matrices, resultado de los algoritmos o la carga de datos.
  - **Consecuencias:** La falta de comunicación afecta la funcionalidad del sistema, generando errores visibles para el usuario y afectando la experiencia.
  - **Controles:**
    - Pruebas de integración regulares para verificar la comunicación entre los componentes.
    - Manejo adecuado de errores en el frontend para informar al usuario sobre problemas de conexión.
    - Monitoreo de las solicitudes API con herramientas como Postman o logs.
- **Manejo inadecuado de datos en la interfaz del usuario:**
  - **Descripción:** La interfaz puede presentar datos incorrectos, desactualizados o mal procesados debido a fallos en el backend o en la manipulación de datos en el frontend.
  - **Consecuencias:** Confusión del usuario, reducción en la confianza del sistema y resultados inexactos al trabajar con datos erróneos.
  - **Controles:**
    - Validación y sanitización de datos en el backend antes de enviarlos al frontend.
    - Pruebas de usabilidad y validaciones en el frontend.
- **Rendimiento insuficiente del sistema con grandes volúmenes de datos (Grafos grandes):**

- **Descripción:** El sistema podría experimentar tiempos de carga lentos o fallos al procesar grafos grandes o matrices complejas.
- **Consecuencias:** Experiencia de usuario deficiente, posibles cuelgues del sistema y limitación para casos prácticos con grandes cantidades de datos.
- **Controles:**
  - Optimización del manejo de estructuras de datos y algoritmos.
  - Escalabilidad mediante paralelización de procesos y distribución de carga en el backend.

- **Cumplimiento insuficiente de estándares de accesibilidad:**
  - **Descripción:** La interfaz del sistema podría no cumplir estándares como WCAG 2.1, dificultando el uso a personas con discapacidades visuales, motoras o cognitivas.
  - **Consecuencias:** Exclusión de usuarios con necesidades especiales y posibles críticas o incumplimientos normativos.
  - **Controles:**
    - Auditorías de accesibilidad utilizando herramientas como Axe DevTools.
    - Implementación de prácticas de diseño inclusivo de la WCAG 2.1 (contrastes adecuados, navegación por teclado, etc).
    - Pruebas de accesibilidad con usuarios finales.

- **Vulnerabilidades en la seguridad de la gestión de datos y autenticación:**
  - **Descripción:** Aunque se implementan medidas de seguridad con Keycloak para la gestión de autenticación y protección de datos, siempre existe el riesgo de vulneraciones a través de ataques avanzados, como la explotación de fallos en el sistema de autenticación o ataques de tipo "man-in-the-middle.[en] conexiones no seguras.

- **Consecuencias:** Filtración de datos sensibles, acceso no autorizado al sistema y pérdida de confianza en la plataforma.
- **Controles:**
  - Continuar utilizando Keycloak para la autenticación con medidas de seguridad robustas, como la autenticación multifactor (MFA).
  - Asegurar la transmisión de datos mediante HTTPS con cifrado TLS.
  - Implementación de auditorías y escaneos de seguridad periódicos en el sistema de autenticación.
  - Revisión regular de configuraciones de seguridad de Keycloak y actualizaciones de seguridad.

- **Problemas en el almacenamiento de archivos al guardar un grafo:**
  - **Descripción:** El almacenamiento de archivos generados al guardar un grafo puede fallar si el formato del archivo es incompatible, el tamaño es excesivo, o si la infraestructura de almacenamiento no está optimizada para manejar grandes volúmenes de datos.
  - **Consecuencias:** Pérdida de datos, fallos en la visualización del grafo, y frustración para el usuario al no poder acceder al archivo correctamente.
  - **Controles:**
    - Validación del formato y tamaño de los archivos de grafo en el backend antes de su almacenamiento.
    - Uso de una infraestructura de almacenamiento optimizada para grandes archivos, como MinIO o AWS S3, con mecanismos de compresión y control de versiones.

Las consecuencias de estos riesgos incluyen insatisfacción del usuario, pérdida de confiabilidad en los resultados y posibles costos de mantenimiento adicionales, que podrían

impactar negativamente en la percepción y uso del sistema web.

Riesgos de Proyecto

Estos riesgos están asociados a factores de gestión y control del proyecto, lo que incluye problemas organizacionales, limitaciones de recursos y cuestiones técnicas.

- **Demoras en la comunicación y cronograma:**
  - **Descripción:** Las demoras en la comunicación entre los equipos de desarrollo y calidad, así como las dificultades en el cumplimiento de los plazos establecidos, pueden retrasar la implementación y las pruebas del sistema de grafos.
  - **Consecuencias:** Retrasos en el cronograma del proyecto, reducción de la calidad de las entregas y posibles inconsistencias entre las funcionalidades implementadas y las expectativas del cliente.
  - **Controles:**
    - Establecimiento de canales de comunicación claros y frecuentes entre los equipos de desarrollo y calidad.
    - Uso de herramientas de gestión de proyectos para hacer un seguimiento eficiente de los plazos y las tareas.
- **QAs desconocen la lógica interna del sistema:**
  - **Descripción:** Los QAs pueden no comprender completamente la lógica de los algoritmos y el sistema de grafos, lo que puede llevar a pruebas incompletas o incorrectas.
  - **Consecuencias:** Identificación inadecuada de errores, posibles fallos en la validación de algoritmos, y reducción de la cobertura de las pruebas.
  - **Controles:**
    - Documentación clara de la lógica interna de los algoritmos y su funcionamiento dentro del sistema.

    - Sesiones de capacitación y colaboración entre desarrolladores y QAs para garantizar la comprensión mutua del sistema.

- **Dependencia de QAs hacia developers para el entorno contenedorizado:**

    - **Descripción:** La dependencia de los QAs hacia los desarrolladores para la configuración y el manejo del entorno contenedorizado puede generar cuellos de botella en las pruebas y la integración continua.
    - **Consecuencias:** Retrasos en la ejecución de pruebas, dificultades en la automatización de pruebas y posibles inconsistencias entre entornos de desarrollo y pruebas.
    - **Controles:**
        - Capacitación a los QAs en la gestión y configuración de entornos contenedorizados, utilizando herramientas como Docker.
        - Creación de scripts y documentación detallada para la configuración y despliegue de entornos de prueba.

- **Falta de documentación del sistema:**

    - **Descripción:** La ausencia de documentación técnica adecuada por parte de los desarrolladores puede dificultar la comprensión, mantenimiento y escalabilidad del sistema de grafos.
    - **Consecuencias:** Aumento en el tiempo de resolución de problemas, dificultades en la implementación de nuevas características y posibles errores debido a la falta de claridad sobre la arquitectura y el flujo del sistema.
    - **Controles:**
        - Revisión de código periódica y mantenimiento de documentación actualizada sobre la arquitectura y algoritmos del sistema.

- Implementación de un sistema de documentación automática, como comentarios detallados en el código y herramientas como Javadoc o Swagger.

Estos riesgos de proyecto pueden afectar el cronograma, los recursos y el alcance del proyecto, comprometiendo potencialmente la capacidad del equipo para cumplir con todos los objetivos del plan de pruebas.

Matriz de Riesgos

Se incluyen matrices de riesgos para clasificar la probabilidad e impacto de cada riesgo, facilitando la priorización de estrategias de mitigación.

Figura 1

Matriz de Riesgos del Producto

- **Resultados inexactos en algoritmos o cálculos complejos** (Probabilidad Media, Impacto Alto): Este riesgo es **Alto** dado que los cálculos incorrectos pueden comprometer la utilidad y fiabilidad del sistema.

- **Fallas en la comunicación entre backend y frontend** (Probabilidad Media, Impacto Medio): Clasificado como riesgo **Medio**, aunque podría generar errores visibles para el usuario.
- **Manejo inadecuado de datos en la interfaz del usuario** (Probabilidad Baja, Impacto Bajo): Este riesgo se encuentra en el cuadrante de riesgo **Bajo** debido a la baja probabilidad de ocurrencia.
- **Rendimiento insuficiente del sistema con grandes volúmenes de datos** (Probabilidad Media, Impacto Medio): Este riesgo está clasificado como **Medio**, ya que afectaría significativamente el desempeño del sistema.
- **Cumplimiento insuficiente de estándares de accesibilidad** (Probabilidad Media, Impacto Bajo): Clasificado como riesgo **Bajo**, pero con la posibilidad de afectar la accesibilidad del sistema.
- **Vulnerabilidades en la seguridad de la gestión de datos y autenticación** (Probabilidad Media, Impacto Alto): Este riesgo es **Alto**, dado que las vulnerabilidades en la seguridad pueden comprometer la integridad del sistema.
- **Problemas en el almacenamiento de archivos al guardar un grafo** (Probabilidad Baja, Impacto Medio): Clasificado como riesgo **Bajo**, pero puede generar frustración si ocurre.

Figura 2

Matriz de Riesgos del Proyecto

- **Demoras en la comunicación y cronograma** (Probabilidad Media, Impacto Alto): Este riesgo se encuentra en el cuadrante de riesgo **Alto**.
- **QAs desconocen la lógica interna del sistema** (Probabilidad Media, Impacto Medio): Clasificado como riesgo **Medio**, este riesgo necesita medidas de mitigación, como capacitación o documentación adicional.
- **Dependencia de QAs hacia developers para el entorno contenedorizado** (Probabilidad Alta, Impacto Medio): Clasificado como riesgo **Alto**, este riesgo requiere capacitación y mejora en la gestión de entornos de pruebas.
- **Falta de documentación de los developers** (Probabilidad Alta, Impacto Bajo): Este riesgo se encuentra en el cuadrante de riesgo **Medio** y necesita controles

estrictos para garantizar la disponibilidad de documentación técnica adecuada.

Presupuesto y Cronograma

Para el desarrollo del proyecto, se empleó la técnica de estimación de tres puntos, utilizando valores optimistas, pesimistas y los más probables para cada tarea. Este método permitió calcular un rango confiable de tiempo y costo, reduciendo riesgos de sobrecostos o retrasos.

Así se obtuvo:

- **Horas estimadas**: 36 horas para el periodo inicial (1 de diciembre al 15 de diciembre).

La estimación final es de **36 horas-persona** con un **error estándar aproximado de 4.8 horas-persona**. Esto significa que el esfuerzo estimado es de 36 horas-persona, pero podría variar en un rango de ±4.8 horas debido a posibles incertidumbres. En base a esta estimación, se ha elaborado un cronograma detallado para el proyecto, que se muestra a continuación.

Cronograma del Proyecto

Cuadro 1

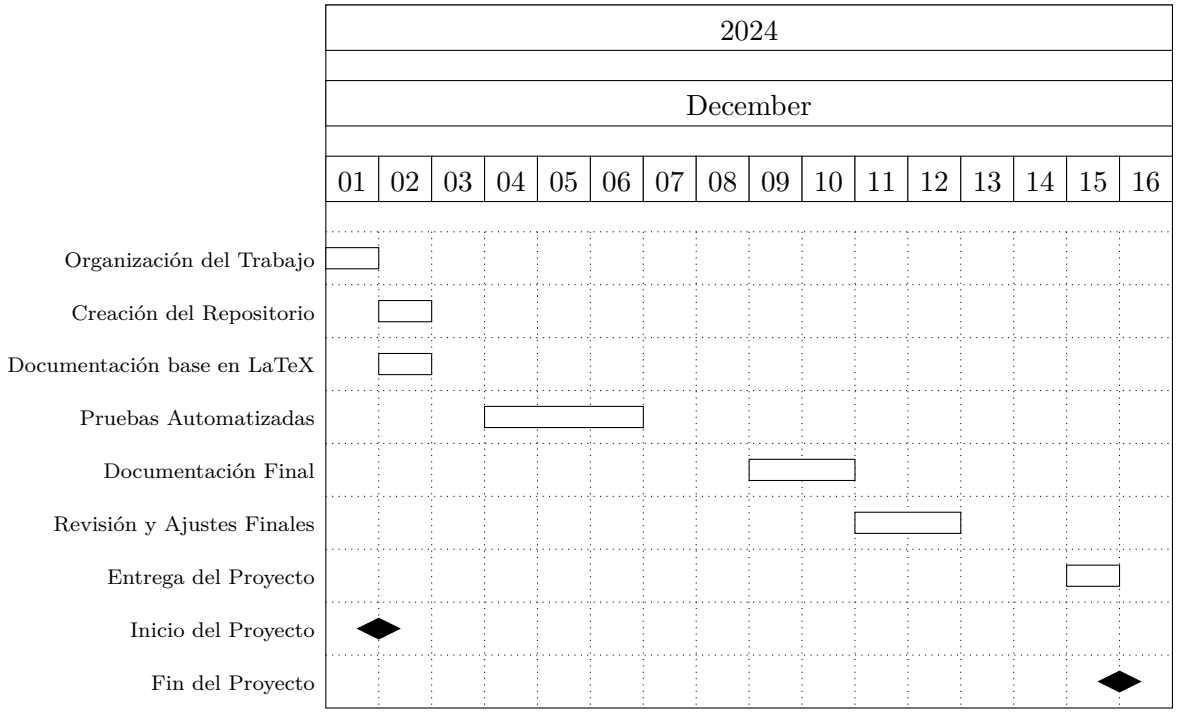
Cronograma del Proyecto

Actividad	**Descripción**	**Fecha prevista**	**Duración (días)**	**Duración (horas)**
Organización del Trabajo	Reunión del equipo para definir roles y tareas.	2024-12-01	1	2.50
Creación del Repositorio	Configuración inicial para pruebas y documentación.	2024-12-02	1	1.5
Documentación base en LaTeX	Subida de documentos base para el plan y el informe.	2024-12-02	1	2.5
Pruebas Automatizadas	Realización de pruebas (3 Postman, 3 Playwright).	2024-12-04 - 2024-12-06	3	16
Documentación Final	Completar y actualizar los documentos en LaTeX.	2024-12-09 - 2024-12-10	2	8
Revisión y Ajustes Finales	Revisión final de todos los documentos y pruebas.	2024-12-11 - 2024-12-12	2	6

Diagrama Gantt

Figura 3

Cronograma del Proyecto: Diagrama de Gantt



El cronograma fue diseñado para ser realista y permitir la realización de las tareas clave del proyecto dentro de un marco temporal adecuado. Las horas estimadas para cada actividad se basaron en la complejidad de las tareas y en la experiencia previa en proyectos similares. Además, se consideraron posibles retrasos o imprevistos, lo que justifica la inclusión del margen de error en la estimación final de horas.

Enfoque de Prueba

El enfoque de prueba se centra en validar tanto los aspectos funcionales como no funcionales del sistema, asegurando que cumpla con los requisitos definidos y proporcione una experiencia de usuario satisfactoria.

Niveles de Prueba

Para asegurar la calidad del sistema, se realizarán pruebas en los siguientes niveles:

- **Pruebas de Componentes**: Cada funcionalidad clave (como la creación de grafos, generación de matrices, y opciones de optimización) se probará de manera aislada para verificar su correcto funcionamiento.
- **Pruebas de Integración**: Validar la interacción entre componentes, especialmente entre la interfaz y los servicios backend, para asegurar que los datos fluyan correctamente.
- **Pruebas de Sistema**: Se evaluará la funcionalidad del editor, considerando la ejecución fluida de los procesos de prueba, generación de soluciones de optimización, y verificación de resultados.

Tipos de Prueba

En este proyecto, se implementarán diferentes tipos de prueba para asegurar una validación integral del editor *Northwest*, abarcando tanto la funcionalidad como la experiencia de usuario.

- **Pruebas Funcionales**: Estas pruebas aseguran que cada funcionalidad del editor cumpla con los requisitos establecidos, validando que el sistema se comporte como se espera en escenarios específicos. Las pruebas funcionales incluyen:
    - **Validación de Creación y Visualización de Grafos**: Verifica que los grafos se creen correctamente con nodos y conexiones, y se visualicen de manera clara e intuitiva en la interfaz.
    - **Carga y Descarga de Archivos**: Asegura que los grafos puedan ser guardados y recuperados sin pérdida de datos ni errores.
    - **Generación de Matrices de Adyacencia y Costos**: Comprueba que el editor construya matrices precisas basadas en las conexiones y valores de los nodos.

    - **Optimización de Soluciones para diferentes algoritmos**: Valida que el editor produzca soluciones correctas según los distintos algoritmos de optimización seleccionados.
    - **Verificación de uso de Interfaz de Usuario (UI)**: Se realizarán pruebas automáticas de la interfaz gráfica para validar la interacción del usuario con el sistema, garantizando la correcta visualización, usabilidad y funcionalidad de los elementos de la interfaz.
    - **Verificación de llamadas API**: Se ejecutarán pruebas de los servicios API mediante Postman para garantizar la correcta comunicación entre el cliente y el servidor.

- **Pruebas No Funcionales**: Estas pruebas evalúan los aspectos que afectan la experiencia del usuario, controlando que se cumpla con los criterios de calidad establecidos por la WCAG 2 AA.
    - **Pruebas de Usabilidad**: Basadas en las heurísticas de Nielsen, estas pruebas aseguran que la interfaz sea intuitiva y satisfactoria para el usuario.
    - **Pruebas de Accesibilidad**: Verifican el cumplimiento de estándares básicos de accesibilidad, asegurando que el editor sea utilizable por personas con distintas capacidades.

Técnicas de Prueba

Las técnicas de prueba aplicadas en este proyecto incluyen una combinación de técnicas de caja negra, caja blanca y herramientas específicas para la usabilidad y accesibilidad, para garantizar una validación exhaustiva del editor *Northwest*.

- **Técnicas de Caja Negra**: Se enfocan en la funcionalidad del software sin considerar la estructura interna. Estas pruebas evalúan si el software actúa según lo esperado en diferentes situaciones.

- **Partición de Equivalencias**: Se utilizará para validar que el sistema maneja adecuadamente diferentes categorías de entrada (por ejemplo, valores válidos e inválidos en campos como oferta y demanda), agrupando los datos en clases de equivalencia. Esta técnica permite reducir el número de casos de prueba necesarios al probar solo una entrada representativa por clase.
- **Valores Límite**: Para verificar que los algoritmos respondan correctamente en los extremos de los datos de entrada, asegurando que no ocurran errores con valores cercanos a los límites aceptables.

- **Técnicas para Usabilidad y Accesibilidad**:
  - **Pruebas Basadas en Listas de Verificación (Checklists)**: Enfocadas en usabilidad, estas pruebas se basarán en listas de verificación construidas a partir de heurísticas de usabilidad de Nielsen para evaluar la facilidad de uso de la interfaz. La lista incluirá aspectos como claridad de la interfaz, accesibilidad de los elementos y consistencia en el diseño.
  - **Pruebas de Accesibilidad con Axe DevTools**: Para verificar el cumplimiento de criterios de accesibilidad, se empleará la herramienta Axe DevTools, que permite realizar pruebas automáticas de accesibilidad en la interfaz y detectar posibles problemas, como bajo contraste de colores, falta de etiquetas en elementos interactivos y problemas de navegabilidad. Los resultados obtenidos con Axe DevTools se documentarán en el reporte de accesibilidad, asegurando que el sistema cumpla con estándares de accesibilidad propuestos por la WCAG 2 AA.

Criterios de Entrada y Salida

- **Criterios de Entrada**: Para iniciar la fase de pruebas, deben cumplirse las siguientes condiciones:

- Todos los servicios de backend estén configurados y ejecutándose en Docker.
- El entorno de frontend esté disponible, ya sea ejecutándose en Docker o localmente.
- La interfaz de usuario y los servicios backend estén disponibles y comunicándose correctamente.
- Todos los requisitos de datos de prueba estén definidos, incluyendo configuraciones específicas para nodos, matrices y otros parámetros requeridos.
- Acceso a las historias de usuario para revisar requisitos y objetivos específicos de cada funcionalidad.
- Se tienen habilitadas las herramientas necesarias para realizar las pruebas, como: Postman, herramientas de accesibilidad y de pruebas de interfaz.

- **Criterios de Salida**: Las pruebas se considerarán finalizadas cuando:
  - Se hayan ejecutado en su totalidad los casos de prueba propuestos y se hayan documentado sus resultados y observaciones.
  - Se haya finalizado el análisis de defectos encontrados y todos los hallazgos estén documentados en el informe de resultados.
  - Todos los criterios de éxito para alcanzar el nivel `AA` en la normativa WCAG de accesibilidad hayan sido evaluados y cumplidos.
  - Se haya completado el checklist de usabilidad basado en las heurísticas de Nielsen, asegurando una experiencia de usuario óptima.
  - Todas las métricas definidas se hayan recopilado para el análisis de resultados.

Independencia de las Pruebas

Dado que parte del equipo de desarrollo participa también en el proceso de pruebas, las pruebas no serán completamente independientes. Para mitigar posibles conflictos de

interés, se establecerán procedimientos claros para la documentación y revisión de resultados, asegurando que las pruebas sean objetivas y confiables.

Métricas a Ser Recopiladas

Para evaluar la efectividad y calidad de las pruebas, se recopilarán las siguientes métricas, incluyendo aquellas derivadas del análisis de resultados:

- **Tasa de éxito de Test Cases**: Porcentaje de pruebas exitosas sobre el total de casos ejecutados.
- **Cobertura de requisitos del usuario**: Porcentaje de historias de usuario cubiertas por pruebas realizadas.
- **Número y tipo de defectos identificados**: Total de defectos encontrados, categorizados según funcionalidad, usabilidad y accesibilidad.
- **Defectos resueltos vs. pendientes**: Proporción de defectos corregidos en relación con los detectados.
- **Tiempo total de ejecución de pruebas**: Tiempo en horas-persona requerido para completar las pruebas.
- **Automatización de Test Cases**: Resultados de pruebas automatizadas, diferenciando las herramientas utilizadas y los resultados obtenidos.
- **Tiempo Medio de Reparación (TMR)**: Tiempo promedio requerido para corregir defectos críticos, desde su detección hasta su resolución.
- **Impacto de accesibilidad**: Alertas detectadas y corregidas en análisis realizados con herramientas como *Axe Dev Tools*.

Requisitos de Datos de Prueba

Para llevar a cabo las pruebas, los datos de prueba deben incluir:

- Incluir diversas configuraciones de tamaño y complejidad de los grafos para abarcar un amplio espectro de escenarios.
- Incluir casos con restricciones específicas para evaluar la robustez y flexibilidad de los algoritmos.
- Los arcos deben contener valores que cubran rangos amplios, incluyendo pesos negativos, valores extremos y valores cercanos a los límites aceptables, para simular distintas situaciones de uso.
- Se deben incluir datos de entrada no válidos para probar la capacidad de los algoritmos de manejar errores de forma controlada y sin interrupciones.

Requisitos del Entorno de Prueba

Los servicios necesarios en el entorno de prueba incluyen:

- **Backend implementado en FastAPI**: Este servicio maneja las solicitudes de optimización de transporte y debe estar configurado para responder a peticiones de la interfaz y los cálculos de matrices, asegurando un procesamiento eficiente y consistente de datos.
- **Servicios de almacenamiento**: Configurados para gestionar y almacenar archivos de configuración en formato JSON. Estos servicios garantizan la persistencia de datos y permiten la carga y recuperación de grafos y matrices sin pérdida de información.
- **Frontend desarrollado en Vue.js**: La interfaz gráfica del editor, desarrollada en Vue.js, permite la interacción del usuario con los elementos del editor. El frontend puede ejecutarse en Docker o de manera local, asegurando accesibilidad a todas las funcionalidades requeridas para la prueba.
- **Buscador web**: Disponibilidad de un buscador de donde pueda realizarse el acceso al sistema.

- **Herramientas de prueba**: Estas herramientas (tales como Postman, Axe DevTools y Playwright) deben estar activas para la ejecución de los test cases.

Además, para asegurar la integridad del entorno, todos los servicios deben estar conectados mediante una red interna en Docker, permitiendo una comunicación fluida entre los contenedores. Esta configuración facilitará la ejecución de pruebas automáticas y manuales en un entorno controlado, simulado y con condiciones similares a las de producción.

Desviación de la Política de pruebas y estrategia de pruebas de la empresa

Este plan de pruebas sigue los lineamientos establecidos por la política y estrategia de pruebas de los propietarios del software, cumpliendo con los estándares de calidad y verificación esperados.

Referencias

Dennis, A., Wixom, B. H., y Roth, R. M. (2015). *System analysis and design* (5th ed ed.). John Wiley & Sons, Inc. Descargado de `http://www.wiley.com/college/dennis` (Indiana University, University of Virginia, University of Northern Iowa)

Gregory, J., y Crispin, L. (2023). *Holistic testing: Weave quality into your product*. Agile Testing Fellowship. Descargado de `https://agiletestingfellow.com` (Copyright 2023, Janet Gregory and Lisa Crispin)

Nielsen, J. (1994, abril). *10 heuristics for user interface design: Article by jakob nielsen.* Descargado de `https://www.nngroup.com/articles/ten-usability-heuristics/` (Recuperado 8 enero, 2020)

Nielsen (1994) Dennis, Wixom, y Roth (2015) Gregory y Crispin (2023)

Anexos

Anexo A: Historias de Usuario

Cuadro 2

Historia de Usuario HU001-SignUp/SignIn-01

HU001-SignUp/SignIn-01
Como usuario nuevo,
quiero poder registrarme proporcionando mi co-
rreo electrónico, nombre de usuario y contraseña,
para crear una cuenta y acceder a la aplicación.
Ver en GitHub

Cuadro 3

Historia de Usuario HU002-SignUp/SignIn-02

HU002-SignUp/SignIn-02
Como usuario que se está registrando,
quiero recibir mensajes de error claros si introduz-
co un correo electrónico inválido o una contraseña débil,
para saber qué debo corregir antes de completar mi registro.
Ver en GitHub

Cuadro 4

Historia de Usuario HU003-SignUp/SignIn-03

HU003-SignUp/SignIn-03
Como usuario registrado,
quiero ingresar mi correo electrónico y contraseña en una pantalla de inicio de sesión,
para acceder a mi cuenta y a las funcionalidades de la aplicación.
Ver en GitHub

Cuadro 5

Historia de Usuario HU004-Johnson-01

HU004-Johnson-01
Como usuario,
quiero construir una representación de un problema de ruta crítica (CPM) como un grafo con actividades en las aristas,
para identificar la ruta crítica utilizando el algoritmo de Johnson.
Ver en GitHub

Cuadro 6

Historia de Usuario HU005-Johnson-02

HU005-Johnson-02
Como usuario,
quiero descargar un grafo como archivo local,
para poder continuarlo más tarde.
Ver en GitHub

Cuadro 7

Historia de Usuario HU006-GraphEditor-01

HU006-GraphEditor-01
Como usuario de la aplicación,
quiero poder agregar nodos, aristas y editar sus valores,
para poder construir y modificar gráficos de manera interactiva y visual.
Ver en GitHub

Cuadro 8

Historia de Usuario HU007-AdjacentMatrix-01

HU007-AdjacentMatrix-01
Como usuario de la aplicación,
quiero obtener la matriz de adyacencia del grafo generado,
para poder visualizar las conexiones entre los nodos de forma estructurada y utilizar esta información posteriormente.
Ver en GitHub

Cuadro 9

Historia de Usuario HU008-FileManagement-01

HU008-FileManagement-01
Como usuario de la aplicación,
quiero poder guardar el grafo realizado en formato JSON,
para cargarlo posteriormente y abrirlo en el mismo estado en el que lo dejé, conservando todos los nodos, aristas y configuraciones.
Ver en GitHub

Cuadro 10

Historia de Usuario HU009-NorthWest-01

HU009-NorthWest-01
Como usuario del sistema,
quiero que la matriz de costos se genere automáticamente al crear nodos y conexiones,
para asegurarme de que los valores de los pesos reflejan correctamente las relaciones entre puntos de suministro y demanda.
Ver en GitHub

Cuadro 11

Historia de Usuario HU010-NorthWest-02

HU010-NorthWest-02
Como usuario del sistema,
quiero manejar desequilibrios entre oferta y demanda, validando los campos obligatorios en el formulario de entrada,
para asegurarme de que se ajusten las soluciones automáticamente, evitando errores en el cálculo del transporte, y que pueda ingresar mis datos sin problemas, recibiendo retroalimentación clara en caso de errores.
Ver en GitHub

Cuadro 12

Historia de Usuario HU011-NorthWest-03

HU011-NorthWest-03
Como usuario del sistema,
quiero generar soluciones válidas para las diferentes configuraciones que me da el sistema,
para poder confiar en la validez de los resultados y obtener diferentes resultados basados en el criterio de optimización que elija.
Ver en GitHub

Cuadro 13

Historia de Usuario HU012-Kruskal-01

HU012-Kruskal-01
Como usuario de la aplicación,
quiero poder aplicar el algoritmo de Kruskal en el editor de grafos para encontrar el árbol de expansión mínimo o máximo,
para visualizar y analizar la ruta óptima entre los nodos seleccionados y obtener el valor total de la suma de los pesos de dicha ruta.
Ver en GitHub

Cuadro 14

Historia de Usuario HU013-Dijkstra-01

HU013-Dijkstra-01

Como usuario del sistema,

quiero que el algoritmo de Dijkstra me devuelva el camino más corto en el grafo con la menor cantidad de recursos posibles,

para visualizar y analizar la ruta óptima entre los nodos seleccionados y obtener el valor total de la suma de los pesos de dicha ruta.

Ver en GitHub

Cuadro 15

Historia de Usuario HU014-InvalidData-01

HU014-InvalidData-01

Como usuario administrador del sistema,

quiero que el sistema valide y maneje los datos no válidos que se ingresan en los algoritmos,

para poder corregir los errores rápidamente y que el sistema no presente fallos graves.

Ver en GitHub

Cuadro 16

Historia de Usuario HU015-Dashboard-01

HU015-Dashboard-01

Como usuario del sistema,

quiero que la interfaz del dashboard se actualice al modificar los nodos o arcos en el grafo,

para poder ver reflejados mis cambios inmediatamente y asegurarme de que el algoritmo calcule con los datos actualizados.

Ver en GitHub

Cuadro 17

Historia de Usuario HU016-AccesibilityAndNavigation-01

HU016-AccesibilityAndNavigation-01

Como usuario con discapacidades,

quiero que el sistema sea accesible y que el contraste de los elementos de la interfaz cumpla con estándares de accesibilidad,

para poder utilizar el sistema sin barreras visuales y facilitar mi interacción.

Ver en GitHub

Cuadro 18

Historia de Usuario HU017-Compet-01

HU017-Compet-01

Como usuario del sistema,

quiero que el algoritmo de Compet procese grafos que incluyen nodos y arcos con valores extremos y negativos,

para poder verificar rutas que consideren correctamente estos valores y obtener soluciones precisas y optimizadas.

Ver en GitHub

Cuadro 19

Historia de Usuario HU018-AccesibilityAndNavigation-02

HU018-AccesibilityAndNavigation-02

Como usuario del sistema,

quiero que cada algoritmo cuente con textos de ayuda o "helps"que expliquen su función y pasos de uso,

para poder comprender cómo funciona cada algoritmo y cómo utilizarlo correctamente en el sistema.

Ver en GitHub

Cuadro 20

Historia de Usuario HU019-Sorts-01

HU019-Sorts-01

Como usuario que quiere ordenar datos,

quiero ingresar una lista de números (o cargar un archivo),

para que el sistema los ordene usando diferentes algoritmos de ordenamiento disponibles.

Ver en GitHub

Cuadro 21

Historia de Usuario HU020-Sorts-02

HU020-Sorts-02

Como usuario que está probando el módulo de ordenamiento,

quiero poder elegir entre varios algoritmos de ordenamiento (Selection Sort, Insertion Sort, Merge Sort, Shell Sort),

para observar y entender cómo funciona cada método visualmente.

Ver en GitHub

Cuadro 22

Historia de Usuario HU021-Sorts-03

HU021-Sorts-03

Como usuario interesado en el módulo de ordenamiento,

quiero ver una animación paso a paso que

me muestre cómo se ordenan los elementos,

para aprender cómo el algoritmo organiza los datos.

Ver en GitHub

Cuadro 23

Historia de Usuario HU022-Sorts-04

HU022-Sorts-04

Como usuario que observa la visualización del ordenamiento,

quiero ajustar la velocidad de la animación,

para adaptar el ritmo de la visualización a mis necesidades.

Ver en GitHub

Cuadro 24

Historia de Usuario HU023-Sorts-05

HU023-Sorts-05
Como usuario que desea probar los algoritmos de ordenamiento con diferentes conjuntos de datos, *quiero* generar automáticamente una lista de números aleatorios en el módulo de Sorts, *para* poder ordenar fácilmente un conjunto de datos sin tener que ingresarlos.
Ver en GitHub

Cuadro 25

Historia de Usuario HU024-BinaryTrees-01

HU024-BinaryTrees-01
Como usuario que trabaja con datos específicos, *quiero* poder guardar un arreglo y cargarlo en el módulo, *para* reutilizar datos sin tener que ingresarlos manualmente cada vez.
Ver en GitHub

Cuadro 26

Historia de Usuario HU025-BinaryTrees-02

HU025-BinaryTrees-02
Como usuario que quiere trabajar con estructuras de datos,
quiero ingresar valores para construir un árbol binario visual,
para entender y ver la estructura resultante.
Ver en GitHub

Cuadro 27

Historia de Usuario HU026-BinaryTrees-03

HU026-BinaryTrees-03
Como usuario que estudia los árboles binarios,
quiero ver los recorridos del árbol (preorden, inorden, postorden),
para entender cómo funcionan las diferentes formas de recorrer un árbol binario.
Ver en GitHub

Cuadro 28

Historia de Usuario HU027-BinaryTrees-04

HU027-BinaryTrees-04
Como usuario que interactúa con árboles binarios,
quiero poder ingresar valores para construir el árbol,
para visualizar y analizar la estructura resultante sin la necesidad de editar o eliminar nodos.
Ver en GitHub

Cuadro 29

Historia de Usuario HU028-BinaryTrees-05

HU028-BinaryTrees-05
Como usuario que quiere explorar la estructura completa del árbol,
quiero centrar y ajustar el gráfico,
para ver el árbol completo de forma clara sin tener que desplazarme mucho.
Ver en GitHub

Cuadro 30

Historia de Usuario HU029-BinaryTrees-06

HU029-BinaryTrees-06
Como usuario que quiere preservar su trabajo,
quiero guardar la estructura del árbol binario y cargarla en futuras sesiones,
para continuar trabajando en el árbol o analizarlo en otro momento.
Ver en GitHub

Cuadro 31

Historia de Usuario HU030-Asignation-01

HU030-Asignation-01
Como usuario,
quiero tener una opción,
para resolver la asignación.
Ver en GitHub

Cuadro 32

Historia de Usuario HU031-Asignation-02

HU031-Asignation-02
Como usuario,
quiero visualizar el resultado de la asignación,
para recibir una respuesta.
Ver en GitHub

Cuadro 33

Historia de Usuario HU032-Asignation-03

HU032-Asignation-03
Como usuario,
quiero elegir entre Maximizar o Minimizar la Asignación,
para recibir una respuesta acorde a mis necesidades.
Ver en GitHub

Cuadro 34

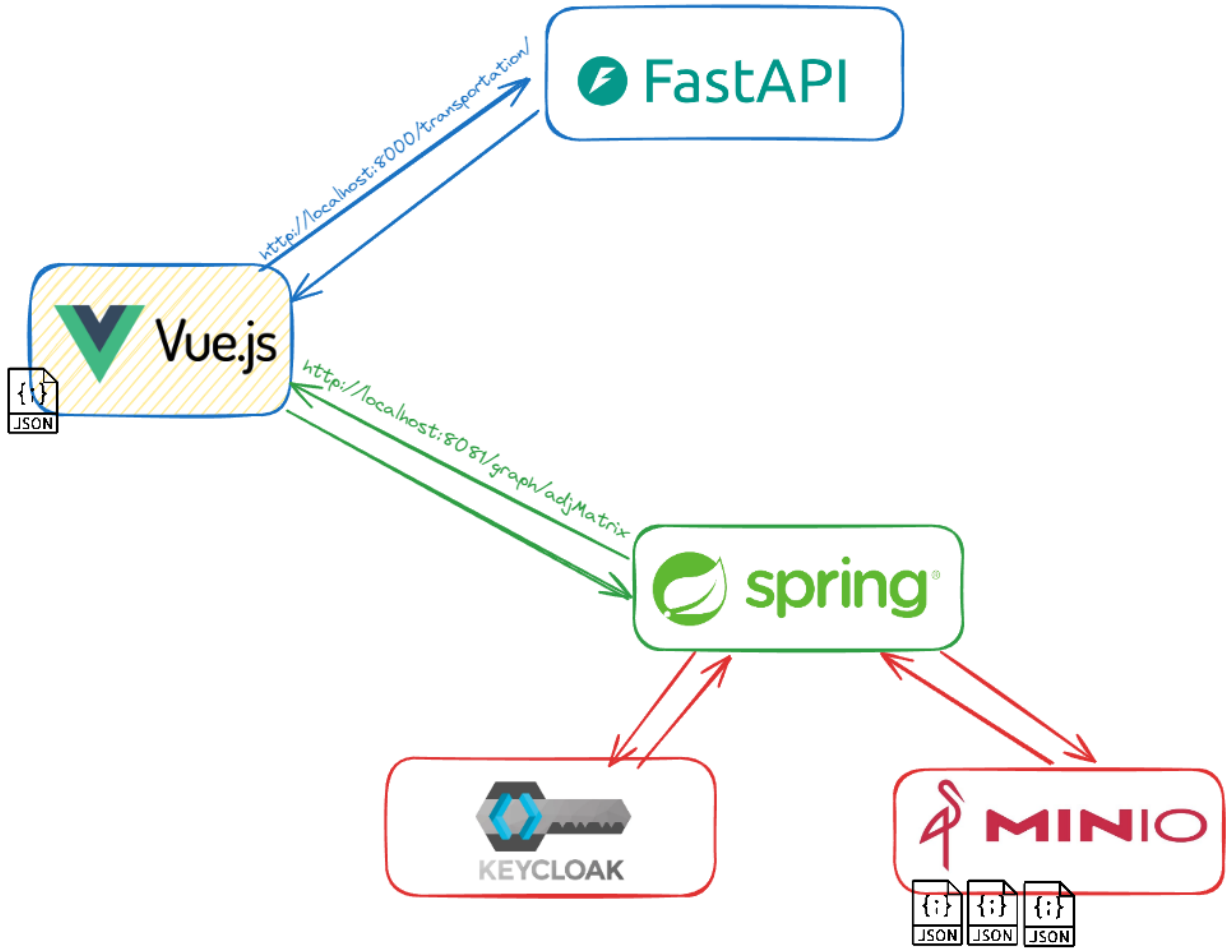
Historia de Usuario HU033-Asignation-04

HU033-Asignation-04
Como usuario,
quiero tener la opción del Algoritmo de Asignación,
para poder usarlo.
Ver en GitHub

Anexo B: Diagramas de Flujo y Arquitectura

Figura 4

Arquitectura del sistema y sus interacciones principales



Anexo C: Plantillas del Reporte de Defectos

Cuadro 35

Plantilla de Reporte de Defectos

Test Case Reference	
ID	
Title	
Description	
Steps to Reproduce	1. 2. 3.
Expected Result	
Actual Result	
Evidence	
Severity	
Priority	

Cuadro 36

Plantilla de Reporte de Usabilidad

Items	**Evaluation**
1.- Visibilidad del estado del sistema	
¿Cada parte de la interfaz comienza con un título que describa el contenido de la pantalla?	
¿El diseño de íconos y su estética es consistente en todo el sistema?	

Cuando se selecciona un icono que está rodeado de otros iconos, ¿Se distingue claramente el ícono seleccionado?	
Si se utilizan ventanas emergentes (pop-up) para mostrar mensajes de error, ¿Permiten esas ventanas que el usuario visualice el error en la interfaz cuando se despliegan?	
¿Hay algún tipo de feedback para cada acción u operación?	
Luego de que el usuario completa una acción o serie de acciones, ¿El "feedback"del sistema indica que el siguiente grupo de acciones puede completarse?	
El sistema provee algún tipo de feedback visual en menús o cajas de diálogo que indiquen qué opciones pueden seleccionarse.	
El sistema provee algún tipo de feedback visual en menús o cajas de diálogo que indiquen en cuál de las posibles opciones se halla posicionado el cursor.	
Si hay menús o caja de diálogo en donde pueden seleccionarse múltiples opciones, ¿El sistema provee algún tipo de "feedback"visual que indique cuáles son las opciones ya seleccionadas?	
¿El sitio web entrega información corporativa de la organización?	

Si existen demoras mayores a 15 segundos en las respuestas del sistema, ¿El usuario es informado del progreso en la concreción de la respuesta?	
¿Informa datos relevantes para quien no "navega"(Ej: Horas de atención)? ¿Y para hacer consultas web o no web (Ej: números de teléfono)?	
¿Los tiempos de respuesta son apropiados para cada tarea?	
Tiempo de escritura, movimiento del cursor o selección con el ratón: entre 0,5 y 1,5 milisegundos	
Tareas más comunes: 2 a 4 segundos	
Tareas complejas: 8 a 12 segundos	
No son necesarios altos niveles de concentración y no es requerido retener información: 2 a 15 segundos	
La terminología usada en los menús, ¿Es consistente con el dominio de conocimiento del usuario en relación a la tarea a realizar?	
¿El usuario conoce su ruta de ubicación?	
2.- Relación entre el sistema y el mundo real	
¿Los íconos son concretos y familiares para el usuario?	
¿Los colores seleccionados corresponden a los valores esperados?	
Cuando se ingresan datos en la pantalla, ¿La terminología utilizada para describir la tarea es familiar para los usuarios?	

Cuando la pantalla incluye preguntas, ¿El lenguaje de esas preguntas es claro y conciso?	
Las combinaciones de secuencias de letras o palabras extrañas o poco frecuentes, ¿Se evitan siempre que sea posible?	
El sistema ingresa/elimina de manera automática los signos de pesos o dólar y decimal cuando se insertan valores monetarios.	
¿Se utilizan nombres unívocos y descriptivos en todo momento?	
¿Se hace uso de los rastreadores de progreso?	
Los H1 están optimizados para SEO	
3.- Control y libertad por parte del usuario	
En sistemas que permitan el uso de ventanas superpuestas ¿Es fácil reacomodar reubicar esas ventanas en la pantalla?	
En sistemas que permitan el uso de ventanas superpuestas ¿Es fácil para los usuarios cambiar de una ventana a otra?	
Cuándo una tarea efectuada por el usuario se completa ¿el sistema espera alguna señal del usuario antes de procesar la tarea?	
¿Se pregunta al usuario que confime acciones que tendrán consecuencias drásticas, negativas o destructivas?	

¿Existe una función para "deshacer.ªl nivel de cada acción simple, cada entrada de datos y cada grupo de acciones completadas?	
¿Los usuarios pueden cancelar aacciones en progreso?	
¿Los usuarios pueden reducir el tiempo de entrada de datos copiando y modificando datos existentes?	
Los menús son anchos (muchos ítems), antes que profundos (muchos niveles)	
Si el sistema posee menús de niveles múltiples ¿Existe algún mecanismo que permita a los usuarios regresar al menú previo?	
Los usuarios pueden moverse hacia delante o hacia atrás entre las opciones de campos o cajas de dialogo.	
Si el sistema utiliza una interfaz de preguntas y respuestas ¿Pueden los usuarios regresar a la pregunta anterior o saltear hacia delante una pregunta?	
¿Los usuarios pueden revertir sus acciones de manera sencilla?	
Si el sistema permite a los usuarios revertir sus acciones , ¿Existe un mecanismo que permita "deshacer"varias acciones de manera simultánea?	
4.- Consistencia y estándares	
El abuso de letras en mayúscula en la pantalla se ha evitado	
No hay más de 12/20 tipos de íconos	

Existe algún elemento visual que identifique la ventana activa	
Cada ventana posee un título	
¿Es posible utilizar las barras de desplazamiento horizontal y vertical en cada ventana?	
Si una opción de un menú es la de "salir"¿Esta opción aparece como ultimo ítem en el menú?	
¿Los títulos de los menús están centrados o justificados a la izquierda?	
Fuentes: hasta tres tipos como máximo	
Hasta cuatro colores (usados ocacionalmente)	
Sonido: tonos suaves para dispositivos de retroalimentación ocacional y bruscos para condiciones críticas.	
¿Se provee una leyenda si los códigos de color son numeros o dificiles de interpretar?	
Se evitan los pares de colores espectralmente extremos y altamente cromáticos	
Los azules saturados no se utilizan para texto u otro elemento pequeño.	
La información más importante esta above the fold (la parte del sitio que los usuarios ven primero)	
¿La estructura de la entrada de datos es consistente entre las diferentes pantallas?	
5.- Prevención de errores	

¿Las entradas de datos no son sensibles a mayúsculas siempre que sea posible?	
Las pantallas para entrada de datos y cajas de diálogo indican el número de espacios en caracteres que estan disponibles para un campo	
Los campos en las pantallas de entrada de datos y las cajas de diálogo ¿contienen valores por defecto cuando corresponden?	
6.- Reconocer antes que recordar	
¿Las áreas de texto tienen .[es]pacios de respiración"que las rodeen?	
¿Se ha utilizado el mismo color para agrupar elementos relacionados?	
¿Existe buen contraste de brillo y de color entre los colores usados para imágines y fondos?	
Los colores suaves, brillantes y saturados se han utilizado para enfatizar datos, mientras que los colores oscuros, opacos y no saturados, han sido usados para des-enfatizar datos?	
¿Los ítems inactivos en un menú aprecen en gris o están omitidos?	
7.- Flexibilidad y eficiencia en el uso	
Los usuarios pueden reducir el tiempo de entrada de datos si se les permite copiar y pegar datos existentes.	

Si las listas de menú son cortas (siete ítem o menos) ¿Pueden los usuarios seleccionar un ítem moviendo el cursor?	
8.- Diseño estético y minimalista	
Los íconos son visuamente distinguibles de acuerdo a su significado conceptual	
¿Cada ícono esta resaltado con respecto a su fondo?	
Cada pantalla de entrada de datos incluye un título simple, corto, claro y suficientemente distintivo.	
Los títulos de los menús son breves pero lo suficientemente largos como para comunicar su contenido.	
9.- Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	
¿Los sonidos son utilizados para señalar errores?	
Si se usan mensajes de error con humor ¿Son apropiados y respetuosos para la comunidad de usuarios?	
¿Los mensajes de error son gramaticalmente correctos?	
¿Los mensajes de error evitan el uso de signos de admiración?	
Los mensajes de error evitan el uso de palabras violentas u hostiles	
Si se detecta un error en un campo de entrada de datos ¿El sistema posiciona el cursor en ese campo o lo resalta de alguna manera?	

¿Los mensajes de error sugieren la causa del problema que lo has ha ocacionado?	
¿Los mensajes de error indican que acción debe realizar el usuario para corregir el error correspondiente?	
10.- Ayuda y documentación	
¿Las instrucciones en linea se distnguen visualmente?	
Si las opciones de los menús son ambiguas ¿el sistema provee información aclaratoria adacional cuando un ítem es seleccionado?	
¿La función de ayuda del menú es visible? (Por ejemplo una tecla etiquetada AYUDA o un menú especial)	
Navegación: la información es facíl de encontrar	
¿La información es exacta, completa y comprensible? ¿La información es relevante?	
Tras haber accedido a la ayuda ¿Pueden los usuarios continuar con su trabajo desde donde ha sido interrumpido?	
¿Es fácil acceder y regresar del sistema de ayuda?	

Cuadro 37

Plantilla de Reporte de Accesibilidad

WCAG 2.1 AA	
Guideline	**Description of Violation**
1	
2	