Proyecto III – Graph API & Visualizer

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Primer Semestre 2019
Valor 15%

Objetivo General

- ➔ Implementar una herramienta que permita la visualización de grafos.

Objetivos Específicos

- ➔ Diseñar soluciones utilizando el paradigma orientado a objetos.
- ➔ Utilizar UML para modelar la solución de un problema.
- ➔ Aplicar **patrones de diseño** en la solución de un problema.
- ➔ Implementar estructuras de datos generales.
- ➔ Desarrollar un REST API para el modelado de grafos.
- ➔ Desarrollar las capacidades relacionadas con el trabajo individual y en equipo y el diseño de soluciones.

Descripción del Problema

El OIJ le ha solicitado a su equipo de trabajo, el diseño y la implementación de un REST API que les permita modelar grafos para el análisis criminológico de casos. El OIJ busca utilizar este API como la capa fundamental de sus sistemas.

A este punto, su equipo no tiene idea de lo que es un REST API ni cómo implementarlo. Sin embargo, uno de los miembros del equipo conocido como El Profesor, compartió un enlace muy relevante:

- www.google.com

Ud como líder del equipo, ha decidido que la única alternativa es investigar sobre REST, HTTP y Servidores de aplicaciones en Java. Además al ser un tema nuevo, deberá buscar buenas prácticas y patrones para la implementación del API. El OIJ es sumamente quisquilloso con las decisiones de diseño que el equipo contratado tome.

El API utiliza el formato JSON para el intercambio de información. Todos los errores del API deben contener el código de operación 500 en el HTTP Header Response Code.

El API debe proveer los siguientes EndPoints:

Endpoint	**Request**	**Response**
POST /graphs Crea un nuevo grafo	N/A	Objeto JSON con el siguiente atributo: id: contiene el id asignado al grafo En caso de error, se retorna 500
GET /graphs Recupera la lista de grafos creados en memoria	N/A	Array de objeto tipo Graph. En caso de no haber ningún grafo, retorna un array vacío y código 200.
DELETE /graphs		
GET /graphs/{id} Recupera el grafo identificado por **id**	N/A	Objeto tipo Graph. En caso de no encontrarse, retorna 404.
DELETE /graphs/{id} Elimina el grafo iden tificado por **id**.	N/A	En caso de no encontrarse el grafo retorna 404. En caso de éxito 200
POST /graphs/{id}/nodes Crea un nuevo nodo en el grafo indicado por id	Objeto que contiene los siguientes valores: entity: JSON object con el contenido de la entidad por almacenar	Objeto JSON con el siguiente atributo: id: contiene el id asignado al nodo
GET /graphs/{id}/nodes Recupera la lista de nodos del grafo identificado por **id**	N/A	Objeto tipo Node
PUT /graphs/{id}/nodes/{id} Actualiza la entidad almacenada en el nodo indicado.	Objeto que contiene los siguientes valores: entity: JSON object con el contenido de la entidad por almacenar	200 si la actualización fue exitosa, 500 en caso de error.
DELETE /graphs/{id}/nodes/{id} Elimina el nodo indicado	N/A	200 si la actualización fue exitosa, 500 en caso de error.
DELETE graphs/{id}/nodes Elimina todos los nodos del grafo identificado por **id**.	N/A	200 en caso de éxito, 500 en caso de error

GET /graphs/{id}/edges Recupera todas las aristas del grafo identificado por **id**	N/A	Array de objetos Edge
DELETE /graphs/{id}/edges: Elimina todas las aristas del grafo identificado por **id**	N/A	200 en caso de éxito, 500 en caso de error. 404 si el graph no tiene aristas
POST /graphs/{id}/edges: Crea un nuevo arista	Objeto con los siguientes atributos: startNode: id del nodo inicial endNode: id del nodo final weight: magnitud	En caso de error 500. En caso de exito 200, con el id del arista.
PUT /graphs/{id}/edges/{id}: Actualiza el arista	Objeto con los siguientes atributos: startNode: id del nodo inicial endNode: id del nodo final weight: magnitud	En caso de error 500. En caso de exito 200. Si el arista no existe, 404.
DELETE /graphs/{id}/edges/{id}: Elimina el arista	N/A	En caso de error 500. En caso de exito 200. Si el arista no existe, 404.
GET /graph/{id}/degree	Query String: ?sort=DESC o ?sort=ASC	Array de objetos Node ordenados por su grado promedio en el orden indicado por el parámetro sort
GET /graph/{id}/dijkstra Obtiene la ruta más corta entre dos nodos	QueryString ?startNode=id&endNode=id	200 en caso de exito, 500 en caso de error, 404 si no hay ruta entre los nodos. Objeto con los siguientes datos: valor total del camino array de objetos node con el camino

***Todos los Mensajes de error deben contener un JSON con el detalle indicando la razón del error**

Objeto	Atributos
Graph	id: Número único asignado para identificar el grafo nodes: Array de objetos tipo Node edges: Array de objetos tipo Edge
Node	id: Número único asignado al nodo. inDegree: Entero que indica el grado entrante outDegree: Entero que indica el grado saliente entity: Objeto JSON que contiene el dato de la entidad almacenada
Edge	id: Número único asignado a la arista start: Id del nodo inicial end: Id del nodo final weight: Valor numérico que contiene la magnitud del edge

El API debe almacenar todo en memoria, no se requiere almacenar datos en archivos.

Como parte de su trabajo, deberá implementar una aplicación a manera de PoC para el OIJ. Deberá desarrollar una aplicación web que permite importar un archivo CSV y construir el grafo implícito en el archivo utilizando el API. Debe verse el grafo visualmente utilizando alguna biblioteca en JS para este propósito. Por ejemplo http://sigmajs.org/ o cualquier otra biblioteca similar.

El CSV a importar en esta aplicación, debe contener la información que simula llamadas telefónicas entre personas. Cada línea tiene lo siguiente:

teléfono1,teléfono2,duración

El grafo se construye de forma tal que cada nodo es un teléfono y su arista tiene un magnitud designada por *duración*. Si hay varias llamadas del mismo telefono1 a un mismo telefono2 se debe sumar la duración de todas las llamadas para establecer el peso de la arista. El grafo es un grafo dirigido.

La aplicación debe permitir visualizar el grado entrante y saliente de cada nodo tanto en la visualización como en una tabla rankeada de menor a mayor (promediar grado entrante y saliente). También permite seleccionar un par de nodos y calcular la ruta más corta entre ambos.

La aplicación permite eliminar nodos, agregar nodos, actualizar nodos, agregar aristas, ,eliminar aristas, actualizar aristas. Es decir, debe permitir usar todos los APIs definidos.

Key technologies: WAS Liberty, REST API, Java, JavaScript.

Se le recomienda investigar buenas prácticas de APIs comerciales como Facebook, Twitter y otros.

Documentación requerida

1. Internamente, el código se debe documentar utilizando JavaDocs.
2. Dado que el código se deberá mantener en GitHub, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    a. Breve descripción del problema
    b. **Planificación y administración del proyecto:** se utilizará la parte de project management de GitHub para la administración de proyecto. Debe incluir:
        ➔ Lista de features e historias de usuario identificados de la especificación
        ➔ Distribución de historias de usuario por criticalidad
        ➔ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
        ➔ Descomposición de cada user story en tareas.
    c. Diagrama de clases en formato JPEG o PNG
    d. Descripción de las estructuras de datos desarrolladas.
    e. Descripción detallada de los algoritmos desarrollados.
    f. Problemas encontrados en forma de bugs de *github*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
3. Anexo. Ver las indicaciones al final de esta especificación.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 15% de la nota del curso.
3. El trabajo es **en grupos de 4 personas**.
4. Es obligatorio utilizar un GitHUB.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 85% y la documentación 15%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
10. La documentación se revisará según el día de entrega en el cronograma.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si la documentación no se entregan en la fecha indicada se obtiene una nota de 0.

    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.

ANEXO DEL PROYECTO

Objetivo General

- ➔ Elaborar un documento que permita la determinación de los requerimientos y los lineamientos para el trabajo en equipo.

Objetivos Específicos

- ➔ Describir los requerimientos del proyecto.
- ➔ Elaborar un modelo de propuesta para las restricciones de la arquitectura del sistema.
- ➔ Establecer los lineamientos adecuados para el trabajo en equipo.
- ➔ Describir los roles y responsabilidades de cada uno de los miembros del equipo.

Atributos de Acreditación

- ➔ Diseño (Inicial).
- ➔ Trabajo individual y en equipo (Inicial).

Descripción del Entregable

Cada grupo debe elaborar un documento que tenga la siguiente estructura:

1. Portada.
2. Tabla de contenidos.
3. Introducción.
4. Diseño.
    a. Listado de historias de usuario.
    b. Diagrama de arquitectura de la solución.
5. Trabajo en equipo.
    a. Plan de iteraciones que describa la distribución de las historias de usuario en las diferentes iteraciones y a su vez describa la carga de cada miembro del equipo durante las diferentes iteraciones.
    b. Bitácora con el trabajo realizado durante el proyecto por cada miembro del equipo.

Aspectos operativos y evaluación

1. El documento debe ser enviado en formato PDF.