Objetivos:
(OT1) Validar y controlar errores en los programas mediante el uso de pruebas unitarias y excepciones.
(OT2) Evaluar algoritmos de búsqueda y ordenamiento clásicos en estructuras de datos lineales y no lineales.
(OT3) Implementar soluciones a problemas que requieran el uso de listas enlazadas, árboles y recursión.

TAREA INTEGRADORA 1: Sistema de Gestión y Monitoreo de Movilidad y Seguridad (SGMMS)

Palmira, en el Valle del Cauca (Colombia), ha experimentado un rápido crecimiento económico y poblacional, lo que ha generado problemas de seguridad y movilidad. El aumento de la delincuencia, los robos y los accidentes, junto con la congestión vehicular y la falta de rutas optimizadas, preocupan a ciudadanos y autoridades. Para enfrentar estos retos, el gobierno municipal implementará un Sistema de Gestión y Monitoreo de Movilidad y Seguridad (SGMMS), que mejorará la vigilancia, agilizará la respuesta a emergencias y proporcionará información en tiempo real a la comunidad.
La Ciudad de Palmira enfrenta los siguientes desafíos:

1. **Seguridad:**
   - Altos índices de delincuencia y robos en zonas estratégicas de la ciudad.
   - Falta de un sistema centralizado para el registro y monitoreo de incidentes en tiempo real.
2. **Movilidad:**
   - Congestión vehicular en horas pico debido a rutas de transporte no optimizadas.
   - Falta de información en tiempo real sobre el estado de las vías y rutas disponibles.
3. **Información:**
   - Los ciudadanos no cuentan con herramientas para acceder a información actualizada sobre seguridad y movilidad.
   - Las autoridades carecen de datos consolidados para la toma de decisiones estratégicas.

El objetivo principal del **SGMMS** es reducir los índices de delincuencia, mejorar la movilidad y aumentar la seguridad de los ciudadanos en la Ciudad de Palmira.
Los objetivos específicos incluyen:

1. Monitorear y rastrear la ubicación de vehículos de emergencia y patrullas de seguridad en tiempo real.
2. Facilitar la respuesta rápida y eficiente a incidentes de seguridad y emergencias.
3. Proporcionar información en tiempo real sobre el estado del tráfico y la disponibilidad de transporte público.
4. Optimizar rutas de transporte público y privado basándose en datos históricos y en tiempo real.
5. Generar reportes y estadísticas para la toma de decisiones por parte de las autoridades.
6. Notificar a los ciudadanos sobre incidentes, rutas alternas y recomendaciones de seguridad.

Primera Fase del Proyecto: Diseño e Implementación Inicial
En esta primera fase, los estudiantes desarrollarán un prototipo inicial del SGMMS que incluirá:

1. **Lectura/escritura de archivos:** El sistema debe permitir leer y escribir datos en formato JSON, que contengan la siguiente información:
   - **Rutas de transporte**: ID de la ruta, distancia (en kilómetros), tiempo estimado de viaje (en minutos), y puntos de inicio y fin.
   - **Incidentes de seguridad**: ID del incidente, tipo (robo, accidente, incendio, etc.), ubicación (coordenadas o dirección), y fecha/hora del reporte, descripción, estado (pendiente, en proceso, resuelto)
   - **Pasajeros:** ID del pasajero, nombre, ruta asignada y contacto.
   - **Conductores:** Id del conductor, nombre, vehículo asignado y estado (disponible, en ruta)

(Imagen: Aimee Custis/Flickr)

2. **Manejo de excepciones:** El sistema debe manejar los errores comunes en ejecución y CREAR sus propias excepciones, proporcionando mensajes al usuario explicando la situación inesperada.
3. **Almacenamiento de datos:** Utilizar listas enlazadas simples, dobles o circulares, para almacenar y manipular la información de rutas e incidentes ***(NO debe usar ArrayList***).
4. **Ordenamiento:** El sistema debe permitir ordenar los datos de acuerdo con criterios específicos:
   - **Rutas**: Ordenar por distancia (de menor a mayor) o por tiempo estimado de viaje (de menor a mayor).
   - **Incidentes**: Ordenar por fecha/hora del reporte (del más reciente al más antiguo).
5. **Búsqueda**: El sistema debe permitir realizar búsqueda
   - Incidentes por Id
   - Conductores por nombre
   - La mejor ruta (ustedes definen el criterio para seleccionar la mejor ruta)**.**
6. **Generación de Reportes Básicos:** El sistema debe generar reportes simples en consola que muestren:
   - Las rutas ordenadas por distancia o tiempo.
   - Los incidentes ordenados por fecha/hora.
   - Resultados de búsquedas específicas.

Entregas

1. **Requerimientos + casos de prueba de funcionalidades + diseño UML (30%)**
   Especifique los requerimientos a partir de lo que aparece en este enunciado (usar este formato). A partir de los requerimientos, escriba al menos 2 pruebas por funcionalidad de cada requerimiento. Pueden ser test negativos o positivos que compongan una prueba completa de la aplicación. RECUERDE QUE SÓLO DEBE ESCRIBIR PRUEBAS DEL MODELO DE LA APLICACIÓN. No incluya pruebas que involucren a la consola o la capa de interacción con el usuario. Reporte la lista inicial de pruebas en el formato de este documento. **Entrega: Semana 5, sábado 8 de marzo 23:55**
2. **Implementación (50%)**
   Esta entrega debe tener la implementación funcionando perfectamente acorde a los requerimientos. Anexe el diseño UML, esta vez en su versión final.**Entrega: Semana 8, sábado 29 de marzo 23:55**
3. **Evolución del programa y seguimiento de Indicadores (20%)**
   Para llevar un buen rendimiento con la gestión de la configuración, se requiere poder medir la evolución de su repositorio. Para esto vaya reportando progresivamente indicadores de calidad en 10 commits del programa. Escoja 10 versiones equi-temporales. Escriba en el readme una sección de indicadores en el que irá acumulando los indicadores versión tras versión. **Entrega: A lo largo de todo el proyecto.**

   Indicadores

   Iteración 1 : <commit-sha>
   Densidad de errores-fallos = 0.2012
   Confiabilidad = 0.7988
   Completitud = 2.32

   Iteración 2: <commit-sha>
   Densidad de errores-fallos = 0.4043
   Confiabilidad = 0.5957
   Completitud = 3.01
   …
   Recuerde que las fórmulas de estas métricas son:
   - Densidad de errores-fallos = total de fallos / total de pruebas
   - Confiabilidad = 1 - densidad de fallos
   - Completitud = casos de prueba / total funcionalidades

ACLARACIÓN

Por favor indicar en el **readme** los ID de los **10 commits** en donde se hizo el reporte de los indicadores para facilitar la revisión.

(Imagen: Aimee Custis/Flickr)