RESUMEN DEL CICLO DE VIDA DEL DATO – DAVID GUTIÉRREZ

En este documento se describen las fases del ciclo de vida del dato implementadas en el proyecto GPS Bus:

1. **Captura / Generación**: Datos GPS simulados (aleatorios y rutas reales).

2. **Almacenamiento**: CSV (`gps_data.csv`), con posibilidad de archivar versiones antiguas.

3. **Procesamiento**: Filtrado por bus, rango horario, simulación de cambio de ruta.

4. **Análisis**: Velocidad media y conteo de paradas.

5. **Distribución**: Exportación de última posición a JSON.

6. **Visualización**: Menú interactivo en consola.

7. **Depuración / Limpieza**: Validaciones, manejo de errores y comentarios coherentes.

8. **Toma de decisiones / Próximos pasos**: Propuestas de mejoras como interfaz gráfica y base de datos.

📍 **FASE 1 – Captura o Generación**

En esta primera fase, el objetivo era generar los datos GPS con los que íbamos a trabajar durante todo el proyecto. Para empezar, creé una clase que simulaba recorridos aleatorios de tres autobuses ficticios (BUS01, BUS02 y BUS03), que generaban registros durante una hora completa (60 minutos) con coordenadas y velocidades simuladas.

Sin embargo, más adelante quise dar un paso más y sustituí esos datos por rutas realistas de tres líneas reales de TUSSAM (BUS22, BUS5 y BUS27). Para ello, busqué coordenadas reales en Google Maps y definí 9 paradas para cada línea, distribuidas a lo largo de Sevilla, con pasos intermedios entre cada una para simular el movimiento del autobús minuto a minuto.

Esta fue probablemente la parte más extensa del proyecto, pero también la que más me ayudó a comprender cómo representar un recorrido con datos, controlando el tiempo, las velocidades y los cambios de ubicación paso a paso. Al final de esta fase, todos los datos simulados quedaron listos para ser almacenados y analizados en las siguientes partes del trabajo.

🗂️ **FASE 2 – Almacenamiento de los datos**

Una vez generados los datos GPS (ya fueran aleatorios o realistas), había que almacenarlos en un formato que permitiera trabajar con ellos más adelante. En la propia actividad se pedía que los registros se guardaran en un archivo `.csv`, así que utilicé ese formato.

El CSV es simple, fácil de manejar desde Java y permite organizar bien los datos en columnas (ID del bus, fecha/hora, latitud, longitud y velocidad). Todo se guarda en un único archivo llamado `gps_data.csv`, que más adelante se puede volver a cargar para analizarlo o mostrarlo por pantalla.

También se nos pedía crear un sistema para archivar versiones anteriores del archivo, así que implementé una función que mueve automáticamente los CSV existentes a una carpeta `archivados` antes de generar uno nuevo. De esta forma se conserva una copia de sesiones anteriores y se evita que se sobrescriban sin querer.

Aunque lo ideal en un sistema real sería almacenar estos datos en una base de datos como MySQL o MongoDB, en este proyecto trabajar con archivos CSV ha sido suficiente para simular esa parte del proceso.

⚙️ **FASE 3 – Procesamiento**

Una vez cargados los datos desde el archivo `gps_data.csv`, se procesan para poder trabajar con ellos de forma útil. En esta fase se utilizan varias funciones que permiten filtrar y modificar la información de los autobuses.

Por ejemplo, se puede filtrar por línea (BUS22, BUS5 o BUS27) o por un rango horario concreto, lo que permite al usuario consultar solo los datos que le interesan. También se incluye una opción para simular un cambio en el recorrido de un autobús, modificando ligeramente sus coordenadas en varios registros.

Todas estas operaciones están organizadas en una clase propia llamada `ProcesadorDatos`, que agrupa estos métodos de forma clara. No hace falta crear objetos, ya que todos los métodos son estáticos y se usan directamente desde el programa principal.

Aunque todo se hace con listas en memoria, en un sistema real este procesamiento se haría probablemente a través de consultas en una base de datos, o incluso en un backend que respondiera a peticiones de una app o una interfaz web.

📊 **FASE 4 – Análisis**

En esta fase analizamos los datos ya filtrados o procesados. El objetivo no es solo mostrarlos, sino sacar algo de "información útil" a partir de ellos. Para ello se crearon dos funciones principales:

- Calcular la velocidad media de cada autobús.

- Contar el número de paradas que ha realizado (cuántas veces la velocidad fue 0).

Ambas funciones están en una clase llamada `AnalizadorDatos`, y se usan desde el menú técnico. El resultado se muestra por consola de forma clara, con los nombres de los autobuses y sus datos.

📌 En mi caso concreto, como la velocidad es constante (30 km/h) y las paradas están definidas desde el principio de forma fija, estos cálculos lucen menos o pierden algo de impacto visual o sorpresa. Aun así, se hace el proceso completo y se simula cómo funcionaría en un caso más variable.

🛑 Por otro lado, para la función de contar paradas, decidí no tener en cuenta la parada inicial (donde el bus parte por primera vez), porque no tiene mucho sentido considerarla como una parada "intermedia". Además, al mostrar los resultados, incluí también la hora exacta y las coordenadas de cada parada, para que no se viera solo un número suelto, sino que se notara que hay un proceso detrás de ese cálculo.

Aunque en este proyecto los datos no son muchos, todo este análisis sirve para simular cómo funcionaría en un sistema real, donde sí habría muchos registros y se podrían sacar conclusiones más completas.

🌎 **FASE 5 – Compartición o distribución**

Para esta fase, incluí una funcionalidad que permite exportar la última posición conocida de cada autobús en archivos `.json`, usando los datos cargados desde el CSV. Esto permite tener un resumen rápido del estado más reciente de cada bus.

Cada archivo generado se llama `bus22_status.json`, `bus5_status.json`, etc., y contiene el ID del bus, sus coordenadas actuales y la marca de tiempo. La idea era cumplir también con esta fase de la actividad, donde se nos pedía simular la compartición del dato usando formatos como JSON.

****Aunque nunca había trabajado antes con archivos JSON en Java****, el formato resultó bastante sencillo de construir para esta tarea, y me pareció útil dejar implementada esta parte para que se entienda cómo podrían compartirse los datos desde un sistema real. En un entorno profesional, esta información podría enviarse a través de una API o mostrarse en una app o panel de control.

👁️ **FASE 6 – Visualización**

En esta fase se presenta toda la información de forma ordenada y comprensible desde la consola.

Para ello, diseñé un menú principal que separa bien dos perfiles: el de usuario (que accede al seguimiento básico del bus) y el técnico (que puede generar, leer, analizar o exportar los datos).

📌 Reconozco que el menú me quedó bastante extenso, porque fui añadiendo mejoras y detalles durante el desarrollo, pero gracias a eso también pude integrar y reflejar todas las fases en un mismo lugar.

Aunque no implementé una interfaz gráfica, intenté cuidar los mensajes, la organización y algunos detalles visuales (como marcar 🅿️ las paradas detectadas) o el pedir que se pulse ENTER para continuar, cosa que ya hice en otros trabajos de programación para que la experiencia desde consola fuera clara y comprensible, y por qué no... algo más dinámica.

> Me habría gustado diseñar una interfaz más visual, pero por cuestión de tiempo y carga de trabajo me fue imposible.

🧠 **FASE 7 – Interpretación de los resultados**

Tras haber analizado los datos en la fase anterior, llega el momento de hacer una lectura más global y reflexiva sobre lo que muestran.

En este proyecto, aunque los trayectos están bastante definidos (con velocidades fijas y paradas preestablecidas), sí se pueden sacar algunas conclusiones: cada línea realiza un recorrido completo en una hora, y hace exactamente ocho paradas intermedias, todas ellas correctamente detectadas por el sistema.

En un entorno real, esta fase permitiría responder preguntas como: ¿qué línea tiene más paradas?, ¿en qué tramo del día hay más movimiento?, ¿algún bus se retrasa o se detiene más de la cuenta?... Aunque aquí no llegamos a ese nivel, el proyecto está preparado para avanzar hacia ese tipo de interpretaciones si los datos fueran más complejos o se conectara con sistemas reales.

Al final, todo este sistema de simulación no busca solo mostrar datos, sino sentar las bases para tomar decisiones: optimizar rutas, detectar incidencias o simplemente ofrecer información clara a usuarios y responsables del servicio.

> 🚌 Como usuario habitual de la línea 22, puedo confirmar que no le vendría mal un sistema así para mejorar sus tiempos y frecuencia.

🔐 **FASE 8 – Protección de los datos**

Aunque esta fase no se desarrolla como tal en el código, sí es importante tenerla en cuenta en un ciclo completo del dato.

En un sistema real, la información GPS puede llegar a ser sensible, ya que refleja rutas, horarios y patrones de movimiento. Por eso, se aplicarían medidas como:

- Transmisión segura (HTTPS, cifrado).

- Control de acceso según el tipo de usuario.

- Sistemas de backups y versiones antiguas (en este proyecto se simula al archivar CSVs).

- Cumplimiento con normativas como el RGPD.

📌 En este caso, no se trabaja con datos personales ni se exponen los registros fuera de consola, pero la idea general de proteger la información queda recogida en parte dentro del planteamiento general.

🧼 **FASE 9 – Depuración y limpieza**

Durante el desarrollo del proyecto, realicé bastantes pruebas con los menús, las rutas y los datos generados. Esto me permitió detectar algunos errores o comportamientos no deseados, como pulsaciones de ENTER que no hacían efecto, rutas que no cargaban bien si el CSV tenía datos incorrectos, o datos duplicados si se generaban varias veces sin limpiar antes.

También ajusté muchos mensajes para que fueran claros y usé validaciones como `leerOpcionMenu()` (incluida en la clase `Depurar`) para evitar que el programa se rompa si se introduce una letra en vez de un número.

Toda esta fase fue una mezcla de ensayo-error y revisión constante del funcionamiento del programa, intentando dejarlo lo más limpio posible antes de entregar. Además, eliminé clases que ya no se usaban y me aseguré de que el código estuviera comentado y estructurado de forma coherente y en un mismo formato todo para facilitar la revisión.

🛠️ No descarto, pese a ello, que hayan podido quedar cosas por depurar...

🔄 **FASE 10 – Toma de decisiones o próximos pasos**

Este proyecto me ha servido para recorrer todas las fases del ciclo de vida del dato de forma práctica y realista, desde la generación hasta el análisis y la visualización.

En el futuro, si quisiera seguir desarrollándolo, algunas posibles mejoras o decisiones serían:

- Añadir una ****interfaz gráfica****, aunque sencilla, que permita consultar las rutas o movimientos de los buses de forma más visual (por ejemplo, con un mapa o representaciones más intuitivas).

- Incluir una ****base de datos real**** en lugar de archivos CSV, para practicar también las conexiones JDBC y las consultas SQL con estos datos.

- Automatizar la lectura del archivo al iniciar el programa (sin tener que pulsar una opción cada vez), o incluso guardar el historial de trayectos.

- Mostrar los datos sobre un mapa real o ficticio, aunque sea en consola, usando símbolos o coordenadas estilizadas para representar movimiento (cosa que vi a algún compañero y me gustó bastante).

Algunas de estas ideas las comenté durante el desarrollo, pero por tiempo y carga de trabajo decidí centrarme en que todo funcionara bien desde consola. Me quedo contento con el resultado y siento que el programa refleja, paso a paso, lo que se nos pedía en la actividad.