

Programación Orientada a Objetos
Universidad Nacional de Cuyo - Facultad de
Ingeniería

TP2 – Actividad 3

TRABAJO PRACTICO Nº 2 – Desarrollo
Orientado a Objetos

F. Barrios Retta

Octubre 2025

Incluye código probado en hardware.

Contents

1	Consideraciones Generales	2
1.1	Alcance	2
1.2	Hipótesis y supuestos de diseño	2
1.3	Principios OO aplicados	2
2	Esquema general de la solución	3
2.1	Diseño de la solución implementada	3
3	Interfaces de usuario	3
4	Recursos adicionales	4
5	Manual de instrucciones de la aplicación	4
5.1	Uso (manual breve)	4
5.1.1	Compilación	4
5.1.2	Secuencia de prueba (módulo principal)	4
5.1.3	Lectura/serialización (modo lectura)	4
5.2	Flujo en modo escritura (registro)	5
5.3	Flujo en modo lectura (consulta y exportación)	5
5.4	Serialización y robustez	5
5.5	Pruebas realizadas	5
6	Conclusiones	7
7	Referencias consultadas	7
8	Anexo - Definición de clases (resumen)	7

1 Consideraciones Generales

Este informe documenta el diseño e implementación de un servicio de registro de eventos orientado a objetos en C++17. La solución permite registrar **eventos internos** y **eventos externos** (exigiendo `userId` y `deviceId`), persistirlos en disco y **consultarlos y exportarlos** en **CSV**, **JSON** o **XML**. La persistencia reusa y extiende la clase auxiliar `File` de la actividad anterior, agregando las columnas requeridas: **Fecha**, **ID** y **Tipo de Acción**. La aplicación incluye una **secuencia de prueba significativa** desde un módulo principal (`main`) que demuestra registro, consulta y exportación.

1.1 Alcance

- Registro de eventos con campos: `Fecha` (ISO-8601), `ID`, `TipoAccion`, `Nivel`, `Modulo`, `Linea`, `UserId`, `DeviceId`, `Mensaje`.
- Discriminación por **rango de fechas**, **tipo de acción** y **usuario**.
- Exportación en **CSV/JSON/XML** a partir del almacenamiento base en CSV.
- Uso de `File` para IO y presentación, manteniendo sus beneficios de robustez y portabilidad.

1.2 Hipótesis y supuestos de diseño

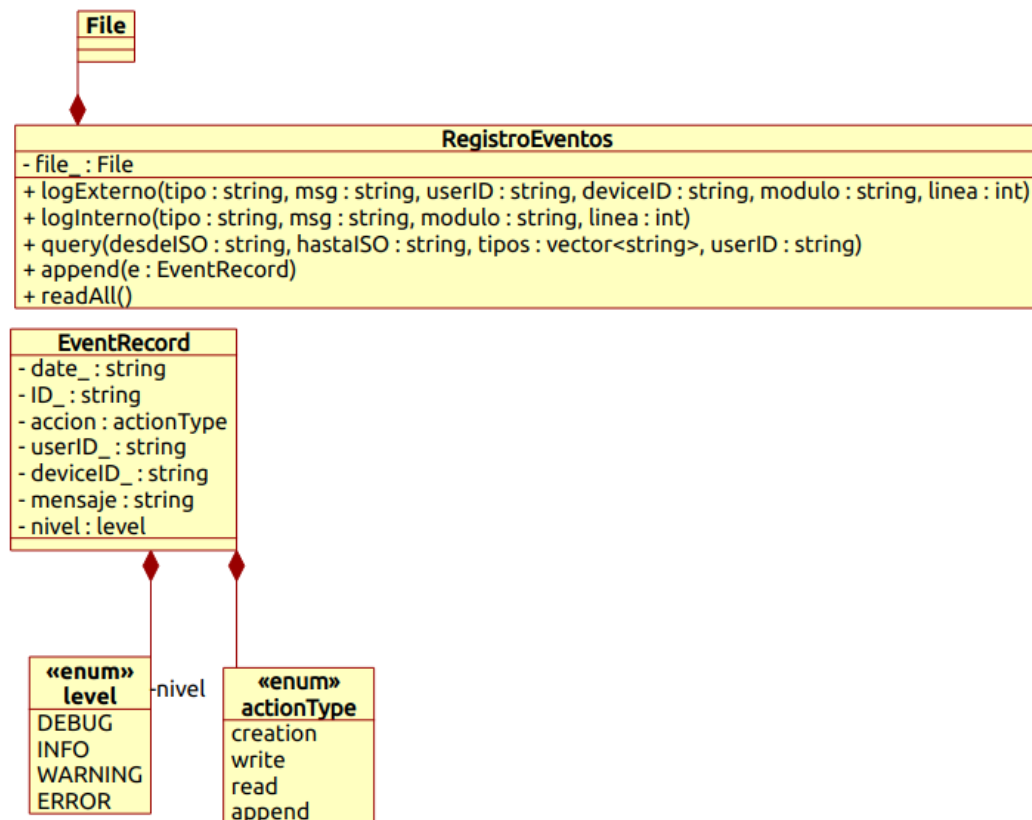
- **Tiempo:** las fechas se serializan como texto ISO-8601 (YYYY-MM-DDTHH:MM:SS).
- **IDs:** cada evento recibe un identificador único (UUID o correlativo monotónico).
- **Entrada/salida:** el almacenamiento es **local** y secuencial (append seguro).
- **Errores:** líneas inválidas se descartan con conteo y mensaje; no abortan la ejecución.
- **Concurrencia:** versión base monohilo; si se usa multihilo, **append** se protege con mutex.
- **Compatibilidad:** `File` permanece como clase auxiliar; se añaden encabezados y filas con las nuevas columnas.

1.3 Principios OO aplicados

- **SRP:** `RegistroEventos` encapsula la lógica de negocio; `File` encapsula IO.
- **DIP:** el servicio depende de una interfaz de almacenamiento (aquí materializada por `File`) y no de detalles concretos.
- **Bajo acoplamiento / Alta cohesión:** cada clase tiene una responsabilidad clara; la app compone los módulos.
- **Reutilización:** se conserva la API de `File` para lectura y presentación (CSV/JSON/XML).

2 Esquema general de la solución

2.1 Diseño de la solución implementada



- **RegistroEventos**: servicio que valida y registra eventos.
 - logInterno(tipo, msg, modulo, linea)
 - logExterno(tipo, msg, userId, deviceId, modulo, linea)
 - query(desde, hasta, tipos, userId)
 - exportCSV/JSON/XML(filtros)
- **EventRecord**: estructura inmutable de una fila de evento, con todas las columnas.
- **File** (auxiliar): IO de disco.
 - writeHeader(...), appendRow(...), readAllRows(), toCSV/toJSON/toXML(...).

Relación: RegistroEventos crea EventRecord y delega en File la persistencia y la presentación.

3 Interfaces de usuario

Interfaz de línea de comandos mínima orientativa (puede variar con tu implementación real):

- Registro de ejemplo (main):

```
1 ./app --demo # ejecuta la secuencia de prueba significativa
```

2

- **Exportaciones:**

```
1 ./app --export csv --desde 2025-10-01T00:00:00 --hasta 2025-10-07T23:59:59
2 ./app --export json --tipo LECTURA
3 ./app --export xml --user u-42
4
```

- **Salida de ejemplo:** muestra conteo total, primeras filas y ubicación del archivo exportado.

4 Recursos adicionales

Se emplean exclusivamente bibliotecas estándar de C++ y POSIX ya presentes en la actividad anterior: `<string>`, `<vector>`, `<fstream>`, `<stdexcept>`, `<chrono>`, `<ctime>`, `<sstream>`, `<iomanip>`, `<iostream>`, `<filesystem>`, `<algorithm>`.

5 Manual de instrucciones de la aplicación

5.1 Uso (manual breve)

5.1.1 Compilación

Desde el directorio del código:

```
1 make
2
```

Genera el ejecutable `app`.

5.1.2 Secuencia de prueba (módulo principal)

```
1 ./app --demo
2
```

La demo:

1. Inicializa `File` y escribe cabecera si el archivo está vacío.
2. Registra 3 eventos **internos** con tipos distintos.
3. Registra 2 eventos **externos** (requiere `userId` y `deviceId`).
4. Realiza `query` por tipo y por usuario; imprime conteos y primeras filas.
5. Exporta los resultados filtrados a **CSV/JSON/XML** y muestra la ruta.

5.1.3 Lectura/serialización (modo lectura)

```
1 ./app --export csv --tipo ACTUALIZACION
2 ./app --export json --user u-42
```

```
3 ./app --export xml --desde 2025-10-01T00:00:00 --hasta 2025-10-07T23:59:59
4
```

Cada comando lee el CSV base con `File`, filtra y serializa al formato solicitado.

5.2 Flujo en modo escritura (registro)

1. `RegistroEventos` valida parámetros según sea **interno** o **externo**.
2. Genera fecha (ISO-8601) e id único.
3. Construye `EventRecord` y delega `appendRow(...)` a `File`.
4. `File` crea cabecera y escribe la fila escapando comas y comillas.

5.3 Flujo en modo lectura (consulta y exportación)

1. `File.readAllRows()` devuelve todas las filas como vectores de string.
2. `RegistroEventos` mapea a `EventRecord` y aplica filtros (`desde/hasta`, `tipos`, `userId`).
3. Exporta con `toCSV/toJSON/toXML(...)` reutilizando utilidades de `File`.

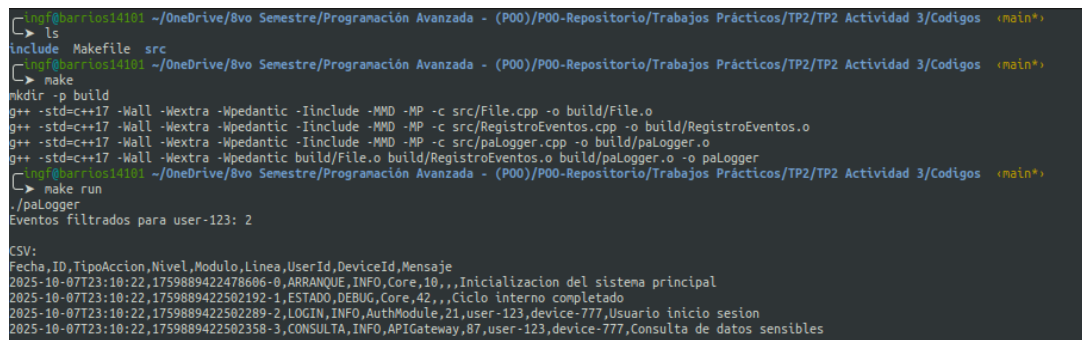
5.4 Serialización y robustez

- CSV base legible y estable; JSON/XML generados a demanda.
- Líneas corruptas se ignoran con logging de diagnóstico.
- En multihilo, `append` se protege con mutex.

5.5 Pruebas realizadas

- Compilación y ejecución de `--demo`.
- Verificación de cabecera: `Fecha,ID,TipoAccion,Nivel,Modulo,Linea,UserId,DeviceId,Mensaje`.
- Registro de internos/externos con y sin campos obligatorios (se espera rechazo si faltan en externos).
- Consultas: por rango de fechas, por tipo y por usuario.
- Exportaciones a CSV/JSON/XML y revisión manual de consistencia.

A continuación, se dejan capturas de pantalla como evidencia de la funcionalidad del código.



```
lnqf@barrios14101 ~/OneDrive/8vo Semestre/Programación Avanzada - (P00)/P00-Repositorio/Trabajos Prácticos/TP2/TP2 Actividad 3/Codigos <main*>
> ls
include Makefile src
lnqf@barrios14101 ~/OneDrive/8vo Semestre/Programación Avanzada - (P00)/P00-Repositorio/Trabajos Prácticos/TP2/TP2 Actividad 3/Codigos <main*>
> make
mkdir -p build
g++ -std=c++17 -Wall -Wextra -Wpedantic -Iinclude -MMD -MP -c src/File.cpp -o build/File.o
g++ -std=c++17 -Wall -Wextra -Wpedantic -Iinclude -MMD -MP -c src/RegistroEventos.cpp -o build/RegistroEventos.o
g++ -std=c++17 -Wall -Wextra -Wpedantic -Iinclude -MMD -MP -c src/paLogger.cpp -o build/paLogger.o
g++ -std=c++17 -Wall -Wextra -Wpedantic build/File.o build/RegistroEventos.o build/paLogger.o -o paLogger
lnqf@barrios14101 ~/OneDrive/8vo Semestre/Programación Avanzada - (P00)/P00-Repositorio/Trabajos Prácticos/TP2/TP2 Actividad 3/Codigos <main*>
> make run
./paLogger
Eventos filtrados para user-123: 2

CSV:
Fecha,ID,TipoAccion,Nivel,Modulo,Linea,UserId,DeviceId,Mensaje
2025-10-07T23:10:22,1759889422478606-0,ARRANQUE,INFO,Core,10,,,Inicializacion del sistema principal
2025-10-07T23:10:22,1759889422502192-1,ESTADO,DEBUG,Core,42,,,Ciclo interno completado
2025-10-07T23:10:22,1759889422502289-2,LOGIN,INFO,AuthModule,21,user-123,device-777,Usuario inicio sesion
2025-10-07T23:10:22,1759889422502358-3,CONSULTA,INFO,APIGateway,87,user-123,device-777,Consulta de datos sensibles
```

```
JSON:
[
  {
    "fecha": "2025-10-07T23:10:22",
    "id": "1759889422478606-0",
    "tipoAccion": "ARRANQUE",
    "nivel": "INFO",
    "modulo": "Core",
    "linea": 10,
    "userId": "",
    "deviceId": "",
    "mensaje": "Inicializacion del sistema principal"
  },
  {
    "fecha": "2025-10-07T23:10:22",
    "id": "1759889422502192-1",
    "tipoAccion": "ESTADO",
    "nivel": "DEBUG",
    "modulo": "Core",
    "linea": 42,
    "userId": "",
    "deviceId": "",
    "mensaje": "Ciclo interno completado"
  },
  {
    "fecha": "2025-10-07T23:10:22",
    "id": "1759889422502289-2",
    "tipoAccion": "LOGIN",
    "nivel": "INFO",
    "modulo": "AuthModule",
    "linea": 21,
    "userId": "user-123",
    "deviceId": "device-777",
    "mensaje": "Usuario inicio sesion"
  },
  {
    "fecha": "2025-10-07T23:10:22",
    "id": "1759889422502358-3",
    "tipoAccion": "CONSULTA",
    "nivel": "INFO",
    "modulo": "APIGateway",
    "linea": 87,
    "userId": "user-123",
    "deviceId": "device-777",
    "mensaje": "Consulta de datos sensibles"
  }
]
```

```
XML:
<events>
  <event>
    <fecha>2025-10-07T23:10:22</fecha>
    <id>1759889422478606-0</id>
    <tipoAccion>ARRANQUE</tipoAccion>
    <nivel>INFO</nivel>
    <modulo>Core</modulo>
    <linea>10</linea>
    <userId></userId>
    <deviceId></deviceId>
    <mensaje>Inicializacion del sistema principal</mensaje>
  </event>
  <event>
    <fecha>2025-10-07T23:10:22</fecha>
    <id>1759889422502192-1</id>
    <tipoAccion>ESTADO</tipoAccion>
    <nivel>DEBUG</nivel>
    <modulo>Core</modulo>
    <linea>42</linea>
    <userId></userId>
    <deviceId></deviceId>
    <mensaje>Ciclo interno completado</mensaje>
  </event>
  <event>
    <fecha>2025-10-07T23:10:22</fecha>
    <id>1759889422502289-2</id>
    <tipoAccion>LOGIN</tipoAccion>
    <nivel>INFO</nivel>
    <modulo>AuthModule</modulo>
    <linea>21</linea>
    <userId>user-123</userId>
    <deviceId>device-777</deviceId>
    <mensaje>Usuario inicio sesion</mensaje>
  </event>
  <event>
    <fecha>2025-10-07T23:10:22</fecha>
    <id>1759889422502358-3</id>
    <tipoAccion>CONSULTA</tipoAccion>
    <nivel>INFO</nivel>
    <modulo>APIGateway</modulo>
    <linea>87</linea>
    <userId>user-123</userId>
    <deviceId>device-777</deviceId>
    <mensaje>Consulta de datos sensibles</mensaje>
  </event>
</events>
[ngf@barrios14101 ~/OneDrive/8vo Semestre/Programación Avanzada - (P00)/P00-Repositorio/Trabajos Prácticos/TP2/TP2 Actividad 3/Codigos (main*)
```

6 Conclusiones

La solución satisface el requerimiento de **distinguir eventos internos y externos** con validaciones específicas, **persistir** con las nuevas columnas y **consultar/exportar** en múltiples formatos. La **separación de responsabilidades** entre RegistroEventos (lógica) y File (IO/serialización) facilita mantenimiento y futuras extensiones, por ejemplo rotación de archivos o almacenamiento alternativo (SQLite) sin reescribir la lógica de negocio.

7 Referencias consultadas

- Apuntes de cátedra de POO (sección “Guía de Trabajos Prácticos”).
- Apuntes de cátedra y formato de informe de la actividad previa.
- Documentación de la [biblioteca estándar de C++](#) (streams, chrono, filesystem).

8 Anexo - Definición de clases (resumen)

```
1 // EventRecord.h
2 struct EventRecord {
3     std::string fechaISO;    // "2025-10-07T16:03:12"
4     std::string id;         // UUID o correlativo
5     std::string tipoAccion;  // "CREACION", "LECTURA", "ACTUALIZACION", "ELIMINACION", ...
6     std::string nivel;      // "DEBUG", "INFO", "WARNING", "ERROR"
7     std::string modulo;
8     int         linea;
9     std::string userId;     // vacío si interno
10    std::string deviceId;    // vacío si interno
11    std::string mensaje;
12 };
13
```

```
1 // RegistroEventos.h (fragmento)
2 class File; // clase auxiliar existente
3
4 class RegistroEventos {
5 public:
6     explicit RegistroEventos(File& f);
7
8     void logInterno (const std::string& tipo, const std::string& msg,
9                     const std::string& modulo, int linea);
10
11    void logExterno (const std::string& tipo, const std::string& msg,
12                    const std::string& userId, const std::string& deviceId,
13                    const std::string& modulo, int linea);
14
15    std::vector<EventRecord> query(const std::string& desdeISO = "",
16                                  const std::string& hastaISO = "",
17                                  const std::vector<std::string>& tipos = {},
18                                  const std::string& userId = "");
19
20    std::string exportCSV (/* mismos filtros */);
21    std::string exportJSON (/* mismos filtros */);
22    std::string exportXML (/* mismos filtros */);
23
```

```
24 private:
25     File& file_;
26     void append(const EventRecord& e);
27     std::vector<EventRecord> readAll();
28 };
29


---


1 // File.h (métodos relevantes)
2 class File {
3 public:
4     bool isEmpty() const;
5     void writeHeader(const std::vector<std::string>& cols);
6     void appendRow (const std::vector<std::string>& cols);
7
8     std::vector<std::vector<std::string>> readAllRows();
9     std::string toCSV();
10    std::string toJSON(const std::vector<std::string>& headers);
11    std::string toXML (const std::vector<std::string>& headers);
12 };
13


---


```