



INFORME FINAL – SISTEMA DE MONITOREO ELECTROQUÍMICO

Universidad de Manizales – Facultad de Ingeniería

Estudiante: Camilo Rubio

Fecha: Junio de 2025

1. 🎯 Objetivo General del Proyecto

Desarrollar una plataforma visual amigable que permita al usuario cargar, consultar y analizar sesiones electroquímicas generadas por PStrace 5.9, y visualizar curvas, PCA y estimaciones de ppm. Todo esto con almacenamiento estructurado en una base de datos PostgreSQL y funcionalidades adicionales como exportación de resultados y búsqueda avanzada.

2. 📁 Estructura del Proyecto

pgsql

CopiarEditar

COINVESTIGACION1/

├── .venv/

├── data/

| ├── INFORME5.pdf

| ├── b_d.txt (script de la BD)

├── sdk/

| └── PSPythonSDK/

├── src/

| ├── __init__.py

- | |— db_connection.py
- | |— interfaz_grafica.py (GUI principal)
- | |— pstrace_session.py (lectura del .psession)
- | |— insert_data.py (carga en BD)
- | |— db_connection.py
- |— limits_ppm.json
- |— debug.log
- |— schema.sql

3. 🧠 Metodología Modular por “Bloques”

El desarrollo del código sigue una metodología modular por bloques, donde cada bloque representa una funcionalidad autónoma que puede mantenerse, corregirse o mejorarse sin romper el resto del sistema. Esta filosofía de trabajo facilita la depuración, pruebas y expansión futura.

Estructura de bloques:

Bloque	Funcionalidad
1	Carga de archivo .psession + inserción en BD
2	Consultas (búsqueda simple, avanzada, metadatos, tabla, exportaciones)
3	Detalle JSON técnico del session info

- 4 Gráfica de curvas individuales + promedio $\pm\sigma$
 - 5 Gráfico PCA con varianza acumulada y anotaciones
 - 6 Estimación de PPM con resaltado y exportación
-

4. Script de la Base de Datos (**b_d.txt**)

La estructura implementada incluye:

- Tabla **sessions**: ID, nombre de archivo, timestamp de carga.
- Tabla **measurements**: ID, FK a sessions, título, timestamp, serial, curva_count, pca_data, ppm_estimations.
- Tabla **curves**: FK a measurement_id, curva_index, num_points.
- Tabla **points**: FK a curve_id, potencial y corriente.

Relaciones completas, con índices y claves foráneas. Modelo relacional escalable.

5. Errores Presentados y Soluciones Aplicadas

Durante el desarrollo y refactorización se presentaron múltiples errores que fueron solucionados de forma estructurada, gracias al enfoque por bloques y la inclusión de **debug.log**. A continuación el resumen:


Error detectado

Causa

Solución aplicada

AttributeError: no attribute 'validate_int'	Se intentó registrar una función inexistente	Se eliminó el uso de <code>validatecommand</code> y se hizo validación manual dentro de <code>query_sessions()</code>
AttributeError: no attribute 'on_session_select'	Se enlazó método antes de estar definido	Se reorganizó la definición de bloques; se definió el bloque 2.4 completamente
AttributeError: no attribute 'load_devices'	Método no estaba definido aún	Se implementó el Bloque 2.5 para cargar seriales de dispositivos
AttributeError: no attribute 'update_overview'	Faltaba la definición de método	Se implementó Bloque 2.6 con conexión a BD y actualización de labels
AttributeError: no attribute 'set_default_date_range',	Método inexistente	Se implementó Bloque 2.7 correctamente para preseleccionar últimos 7 días
pg8000 error: Only %s and %% are supported	Error de sintaxis SQL con <code>%()</code>	Se reemplazó por placeholders tipo <code>%s</code> y se pasó tupla de parámetros
ERROR: no se puede convertir double precision[] a numeric	Se intentó usar <code>MAX</code> directamente sobre array	Se corrigió usando <code>JOIN LATERAL unnest(...) AS ppm(val)</code> para iterar arrays

6. Estado Actual del Sistema

- ✓ La interfaz gráfica se ejecuta sin errores.
 - ✓ Se consultan correctamente las sesiones por fecha o ID.
 - ✓ El sistema muestra la tabla, estadísticas generales y detalles técnicos.
 - ✓ Se integran los arreglos `ppm_estimations` correctamente.
 - ✓ Todos los paneles de la pestaña “ Consultas” están activos.
-

7. Recomendación sobre `device_combobox`

Aunque no es obligatorio en esta etapa inicial (Bloque 2), sí es altamente recomendable prever un `Combobox` para `device_serial`, debido a que:

- Está contemplado como futuro filtro avanzado ([INFORME5.pdf](#), sección 7 y 8).
- `load_devices()` ya obtiene los valores de la BD.
- Servirá para:
 - Filtrar por dispositivo específico.
 - Detectar dispositivos sin mediciones.
 - Generar estadísticas por hardware.

¿Dónde ubicarlo?

En el Bloque 2.2.2 – `_create_filters_panel()`, justo debajo de “ID Sesión”.

¿Cómo? Añadiendo:

python

CopiarEditar

```
ttk.Label(container, text="Dispositivo:").grid(row=2, column=0, sticky="e", padx=5)
```

```
self.device_combobox = ttk.Combobox(container, state="readonly", width=20)
```

```
self.device_combobox.grid(row=2, column=1, padx=5)
```

Y usando `self.device_combobox.get()` en el futuro para filtrar.

8. Próximos pasos

1. Implementar el **Combobox** de dispositivo para permitir filtro por hardware.
 2. Conectar la Mini PC para recibir datos remotos en tiempo real (streaming).
 3. Añadir tooltips explicativos en la GUI.
 4. Implementar exportación automática de PDF con tablas y gráficas.
 5. Mejorar la arquitectura multiusuario si se escala el sistema.
-

9. Recomendaciones Finales

- Mantener la metodología por bloques numerados.
 - En cada edición futura, reemplazar bloques completos como piezas de LEGO.
 - Anotar los cambios en un **CHANGELOG.md** para rastrear el historial.
 - Mantener los `print("[DEBUG] ...")` para seguimiento interno y guardar errores en `debug.log`.
-

10. Conclusión

Se ha construido un sistema estable, escalable y organizado, basado en una estructura modular clara. Los errores fueron detectados y solucionados uno a uno gracias a la depuración intensiva y los principios de separación de responsabilidades.

El siguiente ciclo puede enfocarse ya en visualización avanzada, exportaciones gráficas y conexión remota para cerrar el sistema completo de monitoreo.

