



INFORME 9 – IMPLEMENTACIÓN FINAL DE FUNCIONALIDADES, CONSULTAS Y TOOLTIPS

Nombre del autor: Camilo

Proyecto: Sistema de detección de metales pesados en agua potable

Versión del informe: 9.0

Fecha: 26 de junio de 2025

1. Objetivo del informe

Documentar exhaustivamente las mejoras, ajustes y funciones finales implementadas en la interfaz gráfica del sistema, enfocadas en:

- Bloques visuales (consultas, curvas, PCA, ppm)
 - Tooltips explicativos
 - Consultas intuitivas
 - Correcto filtrado de sesiones únicas
 - Preparación para visualización de contaminantes y exportaciones
-

2. Mejoras implementadas en este ciclo

2.1 Tooltips agregados

Se añadió la clase `ToolTip` al inicio del archivo `interfaz_grafica.py` para permitir ayudas contextuales flotantes. Esto se aplicó en:

- `self.id_entry` → Explicación sobre el ID de la sesión a consultar.
- `self.date_start` / `self.date_end` → Rango de fechas de carga de sesiones.
- `self.device_combobox` → Permite seleccionar el sensor específico.
- Botones de búsqueda, limpieza y últimos 7 días.

- Pestañas: *Curvas, PCA, ppm y Detalle*.

Esto mejora la usabilidad y la intuición del usuario sin alterar la lógica del sistema.

2.2 Visualización de resultados de búsqueda

Se actualizó el método `_create_results_table` para incluir una **columna de "Contaminantes"**, aunque aún no está poblada desde base de datos. También se implementó:

- Colores por estado: alerta (⚠️ ALERTA) o seguro (✅ SEGURO).
 - Etiquetado automático con `tag_configure()`.
-

2.3 Curvas voltamétricas

Método `show_curve()`:

- Gráfica promedia curvas de los ciclos 2–5 como se solicitó.
 - Añade curva promedio y sombra de desviación estándar.
 - Se recupera el nombre del sensor y se titula dinámicamente.
-

2.4 Análisis PCA

Método `show_pca()`:



- Se grafican los componentes principales.
 - Se anotan los tres primeros con su porcentaje de varianza explicada.
 - Se mantiene coherencia visual con fondo y ejes oscuros.
-

2.5 Estimaciones PPM

Método `show_ppm()`:

- Muestra tabla de estimaciones por cada metal.
 - Aplica color de alerta a filas con valores por encima del umbral.
 - Se guarda la tabla como `self.ppm_df` para exportaciones.
-

3. Correcciones aplicadas

- Se validó que `load_devices()` funciona correctamente y no lanza errores si no hay combobox presente.
 - Se confirmó que `query_sessions()` ejecuta correctamente SQL, y la consola devuelve resultados.
 - El problema de "consulta no hace nada" era únicamente de **falta de visual feedback**, lo cual se resolvió con:
 - Tooltips explicativos
 - Panel de resultados más claro
 - Iconografía visual (, )
 - Títulos dinámicos y tablas reactivas
-

4. Base de datos utilizada

```
-- schema.sql

-- 1) Tabla de sesiones
CREATE TABLE IF NOT EXISTS sessions (
    id          SERIAL PRIMARY KEY,
    filename     TEXT NOT NULL,      -- Nombre del archivo .psession
    loaded_at    TIMESTAMP NOT NULL  -- Fecha y hora de carga
);

-- 2) Tabla de mediciones (measurements)
```

```

CREATE TABLE IF NOT EXISTS measurements (
    id          SERIAL PRIMARY KEY,
    session_id  INTEGER NOT NULL REFERENCES sessions(id) ON DELETE
CASCADE,
    title       TEXT      NOT NULL,
    timestamp   TIMESTAMP NOT NULL,
    device_serial TEXT     NOT NULL,
    curve_count INTEGER NOT NULL
);

-- 3) Tabla de curvas (curves)
CREATE TABLE IF NOT EXISTS curves (
    id          SERIAL PRIMARY KEY,
    measurement_id INTEGER NOT NULL REFERENCES measurements(id) ON DELETE
CASCADE,
    curve_index  INTEGER NOT NULL,
    num_points   INTEGER NOT NULL
);

-- 4) Tabla de puntos (points)
CREATE TABLE IF NOT EXISTS points (
    id          SERIAL PRIMARY KEY,
    curve_id    INTEGER NOT NULL REFERENCES curves(id) ON DELETE CASCADE,
    potential    DOUBLE PRECISION NOT NULL,
    current     DOUBLE PRECISION NOT NULL
);

-- Índices adicionales para optimizar consultas
CREATE INDEX IF NOT EXISTS idx_measurements_session ON
measurements(session_id);
CREATE INDEX IF NOT EXISTS idx_curves_measurement ON curves(measurement_id);
CREATE INDEX IF NOT EXISTS idx_points_curve ON points(curve_id);

-- — Bloque: Añadir scan_rate a sessions —
ALTER TABLE sessions
    ADD COLUMN IF NOT EXISTS scan_rate DOUBLE PRECISION,
    ADD COLUMN IF NOT EXISTS start_potential DOUBLE PRECISION,
    ADD COLUMN IF NOT EXISTS end_potential DOUBLE PRECISION,
    ADD COLUMN IF NOT EXISTS software_version TEXT;

```

ESTRUCTURA GENERAL DEL PROYECTO

Carpeta principal: COINVESTIGACION1

```
- .venv/
- .vscode/settings.json
- data/
  | archivos .psession de prueba
  | matrices y muestras (.xlsx, .csv)
  | limits_ppm.json (umbrales ppm)
- sdk/PSPythonSDK/pspython (SDK PalmSens)
- src/
  | db_connection.py ← conexión y migración de BD
  | insert_data.py ← inserción de sesiones y mediciones
  | interfaz_grafica.py ← GUI con filtros, curvas, PCA, ppm
  | main.py ← orquestador (si aplica)
  | pstrace_connection.py
  | pstrace_session.py ← lectura y extracción de datos
- schema.sql ← script de creación y ALTER TABLE
- limits_ppm.json ← archivo de umbrales ppm
- requirements.txt
- debug.log, pstrace_debug.log
```

6. Recomendaciones futuras

- **Contaminantes por metal:** cargar desde `ppm_estimations` y mapear a los límites de `limits_ppm.json` para identificar qué metal excede los umbrales.
 - **Visualizar contaminantes en tabla:** nueva columna "`Contaminantes`" con texto como "`Cd, Zn`", por fila.
 - **Exportar sesión completa en .CSV** (incluyendo metadatos).
 - **Conexión IoT:** ya se tiene base para recibir datos en vivo desde la mini PC, aunque por ahora se sigue usando carga por archivo.
 - **Mejoras visuales:** tooltips por toda la interfaz, ajustes de íconos, gráficos más accesibles.
-

7. Próximo paso inmediato

✓ Todos los bloques han sido revisados, mejorados y confirmados en funcionamiento.

● **Siguiente paso:** Implementar el módulo que **muestra los contaminantes detectados por medición** comparando con los umbrales de `limits_ppm.json`. Esto implica:

1. Cargar los límites desde JSON.
2. Verificar qué metales sobrepasan valores.
3. Mostrar resultado por medición: "`Contaminada: Cd, Zn`" o "`Sin contaminación detectada`".

Esto se implementará en el bloque `query_sessions()` y se mostrará en la tabla como columna "`Contaminantes`"