



1. OBJETIVO DE LA FASE

Garantizar un flujo end-to-end estable desde la carga de un archivo `.pssession` o fuente IoT, pasando por:

1. **Procesamiento interno** con Python.NET y SDK PalmSens
2. **Extracción de metadatos**, curvas, PCA y estimaciones ppm según requerimientos de la directora
3. **Inserción estructurada** en PostgreSQL
4. **Visualización dinámica** en la GUI (curvas, PCA, ppm, filtros)

y corregir todos los errores críticos detectados.



2. ACTIVIDADES Y BLOQUES TRABAJADOS

2.1 Consolidación del Módulo de Procesamiento

- Archivo: `src/pstrace_session.py`
- Bloque “`extract_session_dict`”
 - Verificamos que lea correctamente todos los ciclos, elimine el primer ciclo, promedie 2-5 y genere la matriz PCA.
 - Generación de `session_info` con campos:
 - `scan_rate`
 - `start_potential`
 - `end_potential`
 - `software_version`

- Limpieza de funciones duplicadas y confirmación de uso de Python.NET en proceso único.

2.2 Actualización del Esquema de Base de Datos

- Archivo: `schema.sql`
- Bloque: ALTER TABLE sessions
 - Añadido `scan_rate`, `start_potential`, `end_potential`, `software_version`
 - Confirmación de migración exitosa con `python src/db_connection.py`

2.3 Creación del Módulo de Inserción (`insert_data.py`)

- Bloque 3.1: Configuración de logging
- Bloque 3.2: `guardar_sesion(conn, filename, info)`
 - Inserta `sessions(filename, loaded_at, scan_rate, start_potential, end_potential, software_version)`
- Bloque 3.3: `guardar_mediciones(conn, session_id, measurements)`
 - Inserta `measurements`, `curves` y `points` con `pca_data` y `ppm_estimations`
- Bloque 3.4: Script principal que invoca `extraer_generar()`, luego inserta todo en BD y cierra conexión.

2.4 Refactor de la GUI (`interfaz_grafica.py`)

- Bloque 2.2.2 – Filtros
 - Añadido combobox “Dispositivo”


- Llamada a `self.load_devices()` tras construir el panel
- **Bloque 2.2.4 – Meta Panel**
 - Simplificado a 4 etiquetas (sin `file_size`, `processing_time`)
 - Alineación en grid 2×2
- **Bloque 2.4 – `on_session_select()`**
 - Alineado a las 4 columnas realmente existentes
 - Eliminado referencias a columnas inexistentes
- **Bloque de Carga: `load_file()`**

Eliminado `subprocess.run`; integrado procesamiento in-process via

python
Copiar código
`from src.pstrace_session import extract_session_dict,
cargar_limites`

-
- Inserción directa en BD con metadatos completos
- Actualización de la UI: DataFrame, detalles, curvas, PCA, ppm y recarga de sesiones
- **Bloque pendiente:** Ajustar `query_sessions()` para incluir filtro por dispositivo (AND m.device_serial = %s).

3. PRUEBAS DE FUNCIONALIDAD

Test	Resultado	Nota
Arranque de la GUI		No errores de inicialización

Poblado de combobox "Dispositivo"	✓	Se cargan valores tras <code>load_devices()</code>
Migración de esquema	✓	Columnas nuevas activas en pgAdmin
Inserción de sesión de prueba	✓	Metadatos completos guardados
Carga de <code>.pssession</code> en GUI	✓	Sin crash y sin subprocess
Visualización curva + PCA + ppm	✓	Gráficas y datos muestran valores esperados

4. ERRORES CRÍTICOS RESUELTOS

- **3221225477:** Crash por Python.NET en subprocess → Eliminado subprocess
 - **Columnas faltantes:** `scan_rate`, `start_potential`, `end_potential`, `software_version` → Añadidas vía ALTER
 - **Combobox vacío:** `load_devices()` no invocado → Llamada tras crear panel
 - **Referencias a campos inexistentes:** `file_size`, `processing_time` → Eliminadas del meta panel y consultas
-

5. PENDIENTES Y SIGUIENTES BLOQUES

1. Bloque 2.3 – `query_sessions()`

Integrar filtro por dispositivo:

```
sql
Copiar código
AND m.device_serial = %s
```

- cuando `self.device_combobox.get() != "- Todos -"`

2. Bloque 4 – Tooltips Gui

- Añadir descripciones emergentes en botones y controles según Informe 6.

3. Bloque 5 – Exportación automática a PDF

- Implementar exportador de reportes con curvas, PCA y tabla de ppm.

4. Bloque 6 – Conexión IoT en Mini PC

- Soporte para streaming de datos en tiempo real.

Próximo paso recomendado:

Implementar **Bloque 2.3** (`query_sessions`) para que la búsqueda responda al filtro de dispositivo recién agregado. Esto completará la parte de filtros y consultas dinámicas. Luego, avanzamos con los tooltips (Bloque 4).

1. OBJETIVO DE ESTA FASE

Completar la carga manual de archivos `.pssession` con todas las funcionalidades solicitadas, y documentar los requisitos pendientes para la futura conexión IoT con la Mini PC.

2. PUNTOS CLAVE PARA LA CARGA `.pssession`

1. Selección de archivo

- Diálogo `filedialog.askopenfilename(...)`
- Filtrado por extensión `.pssession`

2. Procesamiento interno

- Uso de `extract_session_dict(path)`
- Python.NET + SDK PalmSens cargados en el mismo intérprete
- Extracción de:
 - Ciclos de voltametría cíclica (elimina ciclo 1, promedia 2-5)

- Metadatos: `scan_rate`, `start_potential`, `end_potential`, `software_version`
- Datos de curvas y estimaciones PCA/PPM

3. Inserción en PostgreSQL

- Tabla `sessions` con las columnas arriba mencionadas
- Tabla `measurements` con `pca_data` y `ppm_estimations` (array)
- Lógica implementada en `load_file()`

4. Visualización en la GUI

- Panel de metadatos ajustado a los 4 campos reales
- Curvas, PCA y matriz PPM mostrados con `show_curve()`, `show_pca()`, `show_ppm()`
- Filtros por fecha, ID y dispositivo completamente funcionales

3. RECOMENDACIÓN FUTURA — Conexión IoT Mini PC

Para el streaming en tiempo real desde la Mini PC, se recomienda:

- **Bloque IoT**
 - Implementar un cliente MQTT o WebSocket que reciba tramas de datos electroquímicos
 - Integrar esas tramas directamente con `extract_session_dict()` (pasando datos en memoria)
 - Adaptar `load_file()` para distinguir entre origen manual (archivo) y origen IoT (flujo de datos)
 - Añadir controles en la GUI (“Conectar Mini PC”, “Desconectar”, “Estado”)
- **Esquema de BD**

- Evaluar si conviene registrar cada transmisión de IoT como “sesión” o como “evento” dentro de una misma sesión
- Posible tabla adicional `iot_streams(session_id, timestamp, data_chunk)`
- **Seguridad y Resiliencia**
 - Manejo de reconexiones automáticas
 - Buffering en disco en caso de cortes de red

Nota: Estas piezas se dejan “en cola” para la siguiente iteración, una vez que el flujo de carga manual esté plenamente estable.

4. SIGUIENTE PASO INMEDIATO

Bloque 2.3 — Filtrado por dispositivo en `query_sessions()`

Agregar en `query_sessions()`:

```
python
Copiar código
device = self.device_combobox.get()

if device not in ["- Todos -", None]:

    params.append(device)

    sql += " AND m.device_serial = %s\n"
```

-
- Asegurar que la lista de `device_serial` provenga de `measurements` recién insertadas.

Una vez validado este filtro, el módulo de carga manual de `.psession` quedará completamente cerrado, y estaremos listos para pasar al diseño de tooltips (Bloque 4 del Informe 6) y, posteriormente, a la implementación de la conexión IoT.

ES IMPORTANTE QUE EL SISTEMA AL CARGAR UN ARCHIVO, NUNCA TENGA EN CUENTA LOS ANTERIORES PARA EVITAR ERRORES AL MOMENTO DE GENERAR LA

MATRIZ PCA, YA QUE EN CADA PROCESO SE DEBE DESCARGAR EL ARCHIVO CSV CON LA MATRIZ