

Monitor de Longevidad v4.0 - Resumen Completo de Sesión

Fecha: 4 de octubre de 2025

Proyecto: Dashboard de salud con datos de HealthConnect (Android)

GitHub: <https://github.com/lw2die/health-dashboard>

Dashboard: <https://lw2die.github.io/health-dashboard/>

🎯 Objetivo Principal

Modularizar el código monolítico (~1000 líneas en un solo archivo) para facilitar mantenimiento y agregar nuevas funcionalidades.

✅ Logros Completados

1. Modularización Exitosa

Antes:



SCRIPT/

 └— monitor_salud_CLAUDE_MODULAR_V4.py (1000+ líneas)

Después:



```

SCRIPT/
    config.py          # Constantes centralizadas
    monitor_salud.py  # Orquestador (~200 líneas)

    core/              # Procesamiento de datos
        __init__.py
        cache.py       # Gestión cache JSON
        limpieza.py   # Limpieza duplicados
        procesador.py # Lectura JSONs HealthConnect

    metricas/          # Cálculos de salud
        __init__.py
        pai.py         # PAI semanal
        fitness.py    # VO2max + TSB
        score.py       # Score longevidad

    outputs/           # Visualización
        __init__.py
        graficos.py   # Datos para Plotly
        dashboard.py  # HTML completo
        github.py      # Git push automático

    utils/
        __init__.py
        logger.py      # Sistema de logging

```

Ventajas:

- Cada archivo tiene 50-200 líneas (fácil de leer y mantener)
- Separación clara de responsabilidades
- Agregar funcionalidades es trivial (ej: VO2max manual)

2. Sistema Funcionando con Datos Reales

Métricas actuales del usuario:

- **PAI Semanal:** 141.2 (objetivo ≥ 100) 
- **Score Longevidad:** 63/100 (Bueno)
- **Peso:** 82.5 kg (objetivo 79 kg)
- **VO2max:** 27.7 ml/kg/min (calculado)
- **Entrenamientos:** 70 únicos procesados
- **Peso:** 27 registros
- **Sueño:** 53 sesiones

3. Estrategia Híbrida de Limpieza

El sistema detecta 3 tipos de archivos JSON:

Tipo	Ejemplo	Estrategia
AUTO_FULL	health_data_AUTO_FULL_2025-10-03_07-05.json	Limpieza agresiva: 1 sesión/día, elimina duplicados
AUTO_DIFF	health_data_AUTO_DIFF_2025-10-04_00-45.json	Sin limpieza: datos ya validados por la app
SAMSUNG_MANUAL	health_data_SAMSUNG_MANUAL_2025-10-04_18-02.json	Sin limpieza + campo adicional vo2_max_records

Ejemplo de limpieza:



MÚLTIPLES ENTRENAMIENTOS el 2025-09-21 (34 sesiones) → manteniendo solo:

- ✓ MANTENER: Swimming - 30 min, FC 123 bpm, PAI 14.9
- ✗ ELIMINAR: Swimming - 30 min, FC 98 bpm, PAI 5.9
- ... (32 sesiones más eliminadas)

4. Manejo Robusto de Errores

Problema detectado: Algunos entrenamientos tienen fc_promedio = null

Solución implementada en core/procesador.py:



python

```
def _calcular_pai(fc_promedio, duracion_min):
    # Validar que fc_promedio no sea None o 0
    if fc_promedio is None or fc_promedio <= 0:
        return 0.0

    if fc_promedio <= FC_REPOSO:
        return 0.0
    # ... resto del cálculo
```

Logging agregado:



python

```
if entrada["fc_promedio"] is None or entrada["fc_promedio"] == 0:
    logger.warning(
        f"⚠️ FC NULA/CERO en [{nombre_archivo}] - "
        f"Tipo: {entrada['tipo']}, Duración: {entrada['duracion']} min"
    )
```

Esto permite identificar exactamente qué archivo tiene el problema para arreglarlo en la app.

5. Git Limpio y Organizado

Problema inicial: El `client_secret_*.json` estaba en el historial de Git (secreto expuesto públicamente)

Solución aplicada:



bash

1. Borrar historial completo

```
rmdir /s .git
```

2. Crear .gitignore

```
echo client_secret_*.json > .gitignore
echo __pycache__/* >> .gitignore
echo cache_datos.json >> .gitignore
```

3. Repo limpio desde cero

```
git init
git config user.name "Tu Nombre"
git config user.email "tu@email.com"
git add .
git commit -m "v4.0 Modular - Clean restart without secrets"
```

4. Push al remoto

```
git branch -M main
git remote add origin https://github.com/lw2die/health-dashboard.git
git push --force origin main
```

Resultado:



- Historial limpio sin secretos
- .gitignore funcionando
- Push exitoso a GitHub



Dashboard Actual

URL: <https://lw2die.github.io/health-dashboard/>

Métricas mostradas:

1. **Score de Longevidad:** 63/100
2. **PAI Semanal:** 141.2 (ventana móvil 7 días)

3. **Peso Actual:** 82.5 kg
4. **VO2max:** 27.7 ml/kg/min
5. **TSB (Balance):** -0.3 (CTL: 4.1, ATL: 4.4)
6. **Sueño:** Promedio últimos 7 días

Gráficos interactivos (Plotly):

- PAI semanal (últimos 30 días) con línea de objetivo
- Evolución del peso con línea de objetivo
- TSB histórico con CTL y ATL

Bugs Detectados (Pendientes en la App)

1. Tipos de ejercicio salen "Desconocido"

Observado en logs:



2025-10-03: Desconocido - PAI=14.7, FC=102 bpm, 62 min

Causa: El campo `exercise_type_name` no se está guardando correctamente en los JSONs

Solución: Revisar la app de HealthConnect que exporta los datos

2. FC nula en algunos entrenamientos

No detectado en esta sesión, pero el código ahora está preparado para identificarlo con warnings detallados.

Configuración del Sistema

Parámetros Usuario (en config.py)



python

```
EDAD = 61
ALTURA_CM = 177
FC_REPOSO = 55
FC_MAX = 159 # 220 - EDAD
PESO_OBJETIVO = 79.0
PAI_OBJETIVO_SEMANAL = 100
```

Rutas



python

```
BASE_DIR = Path(r"H:\My Drive\HealthConnect Exports\SCRIPT")
INPUT_DIR = BASE_DIR.parent
CACHE_JSON = BASE_DIR / "cache_datos.json"
OUTPUT_HTML = BASE_DIR / "index.html"
```

Ejecución



bash

```
# Ejecutar una vez
python monitor_salud.py

# El script entra en loop automático ejecutándose cada 30 minutos
# Ctrl+C para detener
```

📝 Comandos Git Útiles



bash

```
# Ver estado  
git status  
  
# Ver cambios  
git diff  
  
# Commit de cambios  
git add .  
git commit -m "Descripción del cambio"
```

```
# Push a GitHub  
git push origin main
```

```
# Ver historial  
git log --oneline --graph
```

```
# Crear rama para experimentos  
git checkout -b feature/nueva-funcionalidad
```



Próximos Pasos

1. ⚠️ URGENTE: Regenerar client_secret

El `client_secret_545103984315-7sluur0m921263tjolpqstktjirkcnc.apps.googleusercontent.com.json` fue expuesto en el historial viejo de Git.

Pasos:

1. Ir a [Google Cloud Console](#)
2. Encontrar OAuth Client ID: `545103984315-7sluur0m921263tjolpqstktjirkcnc`
3. **Eliminarlo**
4. Crear uno nuevo
5. Descargar el nuevo archivo
6. Reemplazar en `SCRIPT/`

2. Agregar Soporte para VO2max Manual

Contexto: Los exports manuales de Samsung incluyen el campo `vo2_max_records` con mediciones reales del reloj (solo aparece en caminatas/carreras con GPS).

Archivo a modificar: `core/procesador.py`

Función a agregar:



`python`

```

def _procesar_vo2max(datos, cache, nombre_archivo):
    """
    Extrae mediciones de VO2max del reloj.
    Solo disponible en exports SAMSUNG_MANUAL.
    """
    vo2max_data = None

    if "vo2_max_records" in datos and "data" in datos["vo2_max_records"]:
        vo2max_data = datos["vo2_max_records"]["data"]

    if not vo2max_data:
        return False

    count_anteriores = len(cache.get("vo2max", []))

    for v in vo2max_data:
        cache["vo2max"].append({
            "fecha": v.get("timestamp"),
            "vo2max": v.get("vo2_max_ml_per_kg_per_min", 0),
            "metodo": v.get("measurement_method", "Desconocido")
        })

    logger.info(f" → VO2max registros agregados: {len(cache['vo2max']) - count_anteriores}")
    return True

```

En procesar_archivo(), agregar:



python

```

if _procesar_vo2max(datos, cache, nombre_archivo):
    campos_detectados.append("vo2max")

```

En core/cache.py, agregar al cache:



python

```
cache.setdefault("vo2max", [])
```

En el dashboard (outputs/dashboard.py):

- Mostrar "VO2max Medido vs. Calculado"
- Gráfico de evolución de VO2max real en el tiempo

3. Investigar Bug de "Desconocido"

Revisar por qué `exercise_type_name` no se guarda correctamente en la app de HealthConnect.

📁 Archivos Importantes

Cache (cache_datos.json)



```
{  
  "ejercicio": [...], // Lista de entrenamientos únicos  
  "peso": [...], // Registros de peso  
  "sueno": [...], // Sesiones de sueño  
  "procesados": [...] // Archivos ya procesados  
}
```

Limpieza del cache:



```
# Si necesitas reprocesar todo desde cero  
del cache_datos.json  
move procesados\*.json .  
python monitor_salud.py
```

.gitignore



```
client_secret_*.json  
__pycache__/  
*.pyc  
cache_datos.json  
procesados/
```

🔍 Debugging

Ver logs detallados

El script muestra logs en tiempo real:



```
2025-10-04 19:13:29 [INFO] - Procesando archivo: health_data_SAMSUNG_MANUAL_2025-10-04_18-02.json
2025-10-04 19:13:29 [INFO] - → Ejercicios agregados: 70
2025-10-04 19:13:29 [INFO] - PAI TOTAL SEMANAL: 141.2
```

Si encuentra FC nula:



⚠️ FC NULA/CERO en [health_data_AUTO_FULL_2025-10-04_18-42.json] -
Tipo: Swimming, Duración: 45 min, Fecha: 2025-10-04

Si hay duplicados masivos:



RESUMEN LIMPIEZA: Se eliminaron 227 entrenamientos duplicados
Entrenamientos únicos: 29 (de 256 originales)

💡 Consejos para Continuar

1. **Ejecuta el script regularmente** - Cada 30 minutos procesará automáticamente nuevos archivos
2. **Revisa el dashboard** - <https://lw2die.github.io/health-dashboard/>
3. **Monitorea los logs** - Identifica problemas de datos temprano
4. **Haz commits frecuentes** - Especialmente después de agregar funcionalidades
5. **Mantén el .gitignore actualizado** - Nunca versiones secretos o datos personales

📚 Recursos

- **Repositorio:** <https://github.com/lw2die/health-dashboard>
- **Dashboard:** <https://lw2die.github.io/health-dashboard/>
- **Google Cloud Console:** <https://console.cloud.google.com/apis/credentials>
- **Plotly Docs:** <https://plotly.com/javascript/>

Aprendizajes Clave

1. **Modularización mejora mantenibilidad** - 12 archivos de ~100 líneas son más fáciles que 1 de 1000
 2. **Validación defensiva es esencial** - Nunca confies en que los datos de entrada sean perfectos
 3. **Git requiere disciplina** - .gitignore desde el inicio evita problemas
 4. **Logging detallado ayuda debugging** - Saber exactamente qué archivo causó un error ahorra tiempo
 5. **Estrategia híbrida es óptima** - FULL con limpieza agresiva, DIFF sin limpieza
-

Métricas del Proyecto

- **Líneas de código:** ~2,200 (modularizado en 12 archivos)
 - **Commits:** Historial limpio desde commit f7da09a
 - **Datos procesados:** 70 ejercicios, 27 pesos, 53 sesiones de sueño
 - **Tiempo de ejecución:** ~2 segundos por ciclo
 - **Frecuencia:** Cada 30 minutos automático
-

Última actualización: 4 de octubre de 2025, 19:15 hs

Estado:  Sistema funcional y en producción

Próximo milestone: Agregar VO2max manual cuando aparezca en exports de caminata