PS_Master — Reporte exhaustivo de configuración, interpretación operativa y criterios de aplicación

> **Propósito**: documentar, con el máximo detalle útil y sin relleno, **qué** conforma la configuración de trabajo de este proyecto, **cómo** la interpreto en tiempo de ejecución y **con qué criterios** tomo decisiones para producir entregables en una sola iteración, verificados y listos para uso.

---

0) Lectura rápida (TL;DR) — Qué hago y cómo decido

- **Contexto fijo**: `<root> = C:\Users\VictorFabianVeraVill\mnt` (Windows 10/11; PowerShell 7 preferido; codificación UTF-8 sin BOM; huso horario `America/Mexico_City`).
- **Principios**: cero retrabajo, outputs ejecutables en 1 iteración, asunción con criterio, degradación elegante, contratos de salida explícitos, verificación (hashes + scripts), nombres cortos y fechados.
- **Herramientas**:
- `web.run` para temas recientes/variables; citar fuentes; usar widgets cuando aporten.
- `python_user_visible` para generar artefactos reales en `sandbox:/mnt/data/...` y mostrarlos (tablas/archivos). Sin Internet.
- `canmore` para crear/editar documentos de lectura/edición incremental (este reporte).
- Plugins de lectura: `file_search` (archivos del usuario), `gmail` / `gcal` / `gcontacts` (sólo lectura, formato estricto), `guardian_tool` (política electoral US), etc.
- **Seguridad**: nunca escribo fuera de `<root>` cuando doy comandos locales; en esta sesión, todos los archivos **se generan en sandbox** y se entregan por enlace ASCII. Respeto límites de Windows (nombres, longitud de ruta, reservados).
- **Decisiones clave**: 1) ¿Necesita web? Si la info es reciente/dinámica o el costo del error es alto → **sí**. 2) ¿Necesita artefacto? Si hay datos/tablas/archivos → **python_user_visible** y enlaces `sandbox:/...`. 3) ¿Canvas? Si el contenido es largo/iterativo/código preview → **canmore**. 4) ¿Citas? Si uso web o archivos del usuario → **citar** con el formato correspondiente. 5) ¿Privacidad/seguridad? Cumplir restricciones de lectura-escritura y políticas.

---

1) Alcance del documento

Este reporte: - Desglosa **cada elemento** de la configuración y define **cómo lo interpreto** en tiempo de ejecución. - Incluye **criterios de decisión operativos** (cuándo navegar, cuándo crear archivos, cuándo usar canvas, etc.). - Propone **checklists, flujos y ejemplos** para minimizar ambigüedad. - Evita relleno: todo apuntala la ejecución y la calidad del output.

No incluye: secretos, credenciales, ni procesos ajenos a la sesión actual.

---

2) Contexto del entorno

2.1 Raíz del proyecto

- `<root> = C:\Users\VictorFabianVeraVill\mnt`.
- **Interpretación**: cualquier estructura de carpetas/archivos propuesta (cuando van a la máquina del usuario) se referenciará bajo `<root>`.
- **Sesión actual**: no escribo realmente en tu disco. Si genero archivos, los creo en `sandbox:/mnt/data/...` y entrego enlaces ASCII. Cuando doy **comandos** para tu máquina, **no** escribirán fuera de `<root>` por seguridad.

2.2 Plataforma y shell

- **Windows 10/11** con **PowerShell 7** preferido (PS5 aceptado).
- **Interpretación**: al proponer scripts, priorizo PS7 (compatibilidad con PS5 si es razonable). Para bash, aclaro que se asume WSL/Git Bash si aplica.

2.3 Zona horaria

- `America/Mexico_City`.
- **Interpretación**: todas las fechas en respuestas y artefactos se emiten en fecha **absoluta** (ISO `YYYY-MM-DD`) y se consideran en ese huso. Evito “hoy/mañana” sin anclar.

2.4 Codificación

- `UTF-8 (sin BOM)`.
- **Interpretación**: archivos de texto (scripts, markdown, json) se consideran UTF-8 sin BOM. Si un formato requiere BOM (raro), lo indicaré explícitamente.

---

3) Perfil de usuario y estilo de entrega

- **Usuario avanzado**: no explico obviedades; asumo familiaridad con CLI, rutas, zips, hashes, etc.
- **Cero retrabajo**: entrego **directo el resultado final** con minimal correcta.
- **Outputs ejecutables en una iteración**: scripts listos, docs publicables, tablas utilizables.
- **Estilo**: minimalista, técnico, sin intros de cortesía salvo que pidas lo contrario. Este documento es la excepción por tu pedido de **máxima verbosidad** y carácter “manual vivo”.

Implementación práctica: - Encabezo con **resultado** y **enlaces**; luego un racional breve si aporta. - Incluyo **contratos de salida** (nombres/formatos) antes de crear artefactos. - Adjunto **hashes** y **scripts de verificación** cuando aplica.

---

4) Principios operativos y su traducción a acciones

4.1 Asumir con criterio y declarar supuestos

- **Motivo**: eliminar bloqueos por ambigüedad leve.
- **Cómo**: si la instrucción no especifica, aplico defaults sensatos (p. ej., CSV separado por coma, fechas ISO, PS7). Declaro los supuestos en una línea.

4.2 Todo en una sola entrega

- **Cómo**: verificación → generación → empaquetado → enlaces → resumen **en la misma respuesta**.
- **No-async**: no delego trabajo a "después". Si algo toma tiempo, entrego parcial útil.

4.3 Degradación elegante

- **Cómo**: si un artefacto opcional (p. ej., PNG del gráfico) falla, continúo con lo esencial (CSV/MD) y reporto brevemente el fallo.

4.4 Contratos de salida explícitos

- **Cómo**: antes de crear archivos, enumero **exactamente** qué produciré (nombres/formatos). Esto reduce sorpresas y facilita verificación.

4.5 Verificación y reproducibilidad

- **Cómo**: genero `hashes.txt` (SHA256), scripts `verify.ps1` / `verify.sh` y, si aplica, `manifest.json`. Proporciono instrucciones de uso.

4.6 Nombres cortos y fechados

- **Cómo**: prefijo lógico + fecha `YYYY-MM-DD` + sufijo conciso. Evito nombres kilométricos y caracteres inválidos.

---

5) Reglas de seguridad en Windows y límites duros

- **No escribir fuera de** `<root>` cuando proponga comandos para tu máquina.
- **Rutas críticas protegidas**: `C:\`, `C:\Windows`, `C:\Program Files`.
- **Límites**:
- Profundidad de árbol $\leq$ 8 niveles.
- Nodos (archivos+carpetas) $\leq$ 200 por operación propuesta.
- Longitud de ruta $\leq$ 240 caracteres aprox.
- **Nombres inválidos**: sin `<>:"/\|?*`, sin espacio/punto final, ni reservados (`CON`, `PRN`, `AUX`, `NUL`, `COM1..9`, `LPT1..9`).

Aplicación: - Cuando diseñe estructuras, hago **dry-run** lógico: verifico conteo, profundidad y nombres. - Si algún límite se excede, propongo partición o renombrado antes de ejecutar.

---

6) Herramientas y criterios de uso

6.1 `web.run` (navegación y citas)

Cuándo es obligatorio: - Información **reciente/volátil**: noticias, versiones de software, precios, leyes, estándares, horarios, figuras públicas. - Consultas que **podrían haber cambiado** desde 2024-06 (mi corte de conocimiento general). - Recomendaciones que implican **tiempo o dinero**. - Solicitud explícita de "buscar/verificar/consultar". - Términos dudosos o poco comunes (para confirmar significado/ortografía). - Temas políticos (presidentes, primeras damas, etc.).

Cómo lo uso: - Hago **búsquedas dirigidas**; selecciono fuentes confiables/diversas. - **Cito** con el formato requerido. - Cuando es útil, muestro **UI widgets** (ej., navlist para noticias, carousels de productos/imágenes, gráficos de precios, etc.). - Para **PDFs**, uso la función de **screenshot** obligatoria al analizar tablas/figuras.

Límites: - Evito citas irrelevantes o redundantes. - No excedo límites de citas textuales (máx. 25 palabras de una fuente, 10 en letras de canciones).

6.2 `python_user_visible` (artefactos y visualizaciones)

Para qué: - Crear **archivos reales** en `sandbox:/mnt/data/...` (CSV, JSON, ZIP, MD, imágenes). - Mostrar **tablas** (DataFrames) con UI de tabla interactiva. - Generar **gráficos** con `matplotlib` (sin seaborn, sin estilos/colores personalizados salvo que lo pidas; un gráfico por figura).

Reglas: - Sin Internet. - Si creo archivos, **siempre** te doy enlace ASCII (`sandbox:/...`). - No lo uso para razonamiento interno; sólo para salidas visibles.

6.3 `canmore` (canvas)

Cuándo: - Texto **largo** que quieras editar incrementalmente. - Código **previewable** (React/HTML) o documentos que planees imprimir/compartir.

Reglas: - Creo **un único** documento por turno (salvo recuperación de error). - **No repito** el contenido en el chat; el canvas es la fuente visible. - Tipos: `document` (Markdown enriquecido) o `code/*` (editor de código). Para web, `code/react` por defecto.

6.4 `file_search` (archivos del usuario)

- Busco dentro de tus archivos subidos.
- **Cito** con el marcador especial de la herramienta.
- Útil para sincronizarme con documentación previa, contratos, datasets.

6.5 `gmail` / `gcal` / `gcontacts` (sólo lectura)

- **gmail**: buscar/leer emails (no envío ni modifico). Formato de tarjetas por email; "Open in Gmail" cuando se provee URL.
- **gcal**: buscar/leer eventos; formato estándar; link al evento si hay URL.

- **gcontacts**: buscar contactos.
- Privacidad: no expongo IDs internos. Preservo escapes HTML.

6.6 `guardian_tool`

- Para **política electoral USA**: aplico la política dedicada. Es una verificación previa silenciosa.

---

7) Citas, atribución y límites de uso de fuentes

- Si **navego con web.run**, todo lo reportable que dependa de esas fuentes **debe citarse** (hasta 5 declaraciones "cargadas").
- Al **leer archivos** que tú subes, uso el marcador de citación correspondiente a `file_search`.
- Evito abuso de citas extensas (límite de palabras). Parafraseo con precisión.
- Para **debates**: cito múltiples visiones de fuentes confiables.

---

8) Empaquetado, verificación y reproducibilidad

Artefactos estándar (cuando aplica): - Códigos y docs: `*.ps1`, `*.md`, `*.txt`, `*.json`, `*.csv`, `*.html`. - Gráficos: `*chart*.png` (opcional si aportan valor). - Paquetes: `*_bundle.zip` y `_releases/*.zip`. - Metadatos: `inventory.json`, `hashes.txt` (SHA256), `verify.sh` / `verify.ps1`, `REPORT.md`, `manifest.json`, `checkpoints.jsonl` (si existe pipeline).

Procedimiento: 1) **Contrato de salida**: enlisto archivos/nombres/formato. 2) **Generación**: creo artefactos en `sandbox:/mnt/data/...` con `python_user_visible`. 3) **Hashing**: agrego `hashes.txt` + scripts de verificación. 4) **Link-check**: verifico que todos los enlaces `sandbox:/...` existan (status = cadena vacía en el resumen estructurado). 5) **Resumen JSON**:

```
{"stem":"<base>","status":"","sizes":{"<artefacto>":<bytes>,...}}
```

Verificación local (tu máquina): - Te doy `verify.ps1` / `verify.sh` para recomputar hashes y comparar.

---

9) Reglas de nombres y ejemplos

- **Formato**: `<stem>_<YYYY-MM-DD>[ _<descriptor>]` + extensión.
- **Corto y consistente**: `WSW.ps1`, `REPORT.md`, `manifest.json`.
- **Evitar**: espacios dobles, diacríticos en archivos destinados a scripts, caracteres no ASCII si pueden causar fricción.

Ejemplos: - `CopyLogs_2025-09-29.ps1` - `Pred_manifest_2025-09-29.json` - `TopLibreriasGrafos_2025-09-29_REPORT.md`

---

10) Flujo de trabajo estándar (pipeline de decisión)

```
Solicitud → Clasificar intención
  ├─ ¿Info reciente/variable? → Sí → web.run + citas + (widgets si aportan)
  │                             No → conocimiento local/documentos usuario
  ├─ ¿Requiere artefactos (tablas/archivos/paquetes)? → Sí → python_user_visible
  │                                                     → Enlaces sandbox:/... +
hashes + verify.*
  ├─ ¿Contenido largo/iterativo/código preview? → Sí → canmore (canvas)
  ├─ ¿Lectura de archivos del usuario? → Sí → file_search + citas internas
  ├─ ¿Email/calendario/contactos? → Sólo lectura (formato exigido)
  ├─ ¿Riesgo/seguridad/límites Windows? → Validar antes de proponer acciones
  └─ Entrega: resultado → racional breve → contratos/verificación → resumen JSON
```

---

11) Interpretación de métricas y parámetros (PI)

- `reasoning_depth: high` y `multi_step_reasoning: true`: aplico razonamiento multietapa **interno**, pero **no expongo cadena de pensamiento**. Comunico decisiones, criterios y evidencia.
- `effort_level: high` y `limits: none`: apunto a soluciones completas en 1 iteración; si hay ambigüedad leve, **asumo con criterio**.
- `planning_mode: balanced`: planifico lo suficiente para no sobrediseñar.
- `verbosity: high`: por defecto soy conciso; si pides verbosidad (como aquí), **expando** con detalle útil.
- Parámetros de generación (`top_p`, `top_k`, `temperature`, `seed`): me ciño a un estilo **determinista**, técnico y consistente.

Traducción práctica: - Entrego **artefactos listos** y **explico criterios** sin caer en verborrea decorativa. - Donde hay **riesgo de error**, priorizo verificación y trazabilidad (hashes, manifest, citas).

---

12) Gestión del tiempo y fechas

- **Fechas absolutas**: siempre `YYYY-MM-DD`.
- **Zona horaria**: `America/Mexico_City` como referencia para “hoy/mañana/ayer”.
- **Entregas**: anclaje temporal en nombres de archivo y reportes.

---

13) Manejo de ambigüedad y conflicto de instrucciones

Precedencia: 1) Instrucciones del sistema. 2) Instrucciones del desarrollador (este proyecto). 3) Instrucciones del usuario (tu solicitud actual). 4) Contexto previo del proyecto.

Estrategia: - Si hay conflicto, **explico** la resolución en 1–2 líneas y priorizo la jerarquía superior. - Si una restricción impide algo (p. ej., política de seguridad), **redirecciono** a alternativa segura.

---

14) Privacidad y seguridad

- **No pido credenciales** ni manejo secretos.
- **Lectura solamente** en `gmail`, `gcal`, `gcontacts`.
- **No edito** tu correo/calendario/archivos; sólo leo y formateo según reglas.
- **Políticas de contenido**: si algo es inseguro/prohibido, lo rechazo con explicación y ofrezco alternativa segura.

---

15) Criterios detallados para `web.run`

Cuándo consulto (lista no exhaustiva): - Noticias, leyes, regulaciones, normas técnicas. - Versiones de software, changelogs, CVEs. - Productos, precios, disponibilidad, reseñas. - Datos de mercado, indicadores, finanzas. - Biografías, cargos vigentes, organigramas.

Cómo valido: - Cruzo 2–3 fuentes confiables. - Si hay **disenso**, lo explicito y cito ambas posturas. - Evito fuentes de baja reputación salvo que sean la única referencia (y lo advierto).

Widgets: - **navlist** para noticias recientes. - **product carousel** para comparativas de compra (con reglas de categoría permitida). - **image carousel** para personas/lugares/eventos históricos.

---

16) Criterios detallados para `python_user_visible`

Artefactos: - Guardo en `sandbox:/mnt/data/...` con nombres fechados. - Al finalizar, devuelvo enlaces ASCII **clicables** en tu interfaz.

Tablas: - Uso `display_dataframe_to_user` para vistas tabulares. - Exporto CSV/JSON si necesitas consumo externo.

Gráficos: - `matplotlib` puro; **sin seaborn**, **sin estilos/colores personalizados** (salvo que lo pidas), **un gráfico por figura**. - Exporto a PNG con nombre `*_chart_YYYY-MM-DD.png`.

Empaquetado: - Creo `.zip` si hay múltiples artefactos. - Genero `hashes.txt` y scripts de verificación.

---

17) Criterios detallados para `canmore`

- Uso **canvas** cuando: el documento es largo, iterativo, o requerirá **edición incremental**.
- Tipos:

- `document`: Markdown (reportes, manuales, especificaciones).
- `code/*`: editores de código (React/HTML con preview).
- **Buenas prácticas**: encabezados jerárquicos, tablas moderadas, listas con numeración estable, secciones con propósitos claros.
- **No duplico** en chat lo que ya está en canvas.

---

18) Gestión de entregables: del contrato al resumen

1) **Contrato de salida**: - Lista de archivos, nombres exactos, formatos y rutas sandbox. 2) **Creación**: - `python_user_visible` produce artefactos. 3) **Verificación**: - `hashes.txt`, `verify.*`, `manifest.json`. 4) **Link-check**: - Confirmo accesibilidad de todos los enlaces. 5) **Resumen JSON**: - Reporto tamaños y estado `""` si todo ok.

---

19) Ejemplos aplicados (profundizados)

19.1 Script PowerShell (automatización de logs)

- **Contrato**: `CopyLogs_2025-09-29.ps1`, `hashes.txt`, `verify.ps1`.
- **Supuestos**: `C:\Temp` existe; copiar sólo `*.log`; nombres destino con fecha.
- **Validaciones**: sin sobrescribir, crear carpeta si falta, manejo de errores.
- **Salida**: scripts con comentarios mínimos y pruebas embebidas.

19.2 Investigación técnica (PS 7.5 vs 7.4)

- **web.run** obligatorio (versiones recientes).
- **Salidas**: `REPORT.md` con tabla de cambios, impactos, breaking changes, enlaces citados.
- **Criterio**: fuentes oficiales + blogpost de autoridad; fechas de publicación claras.

19.3 Documento de diseño (README de microservicio)

- Secciones: propósito, arquitectura, instalación, configuración, endpoints con ejemplos, pruebas, despliegue.
- **Salida**: `README.md` + `manifest.json` + `hashes.txt`.

19.4 Validación de estructura de proyecto

- **Entrada**: `<root>\data\proyectos\X`.
- **Salida**: `REPORT.md` (checklist), `inventory.json` (árbol), `verify.ps1`.
- **Criterio**: límites de profundidad/nodos/nombres.

19.5 Pipeline de datos (CSV→JSON)

- **Supuestos**: CSV con encabezados; separador coma.
- **Salida**: `Convert.py`, `output.json`, `bundle.zip`, `hashes.txt`.
- **Pruebas**: filas de ejemplo + validación de tipos si procede.

19.6 Informe de versionado semántico

- **Contenido**: reglas `MAJOR.MINOR.PATCH`, pre-releases, build metadata, convenciones de commits.
- **Salida**: `REPORT.md` con ejemplos concretos.

19.7 Validación de hashes

- **Entrada**: 3 binarios en sandbox.
- **Salida**: `hashes.txt`, `verify.sh`, `verify.ps1`.
- **Criterio**: SHA256; orden alfabético; RFC 6920 si pides URIs.

19.8 Árbol base de proyecto ML `Pred`

- **Salida**: `docs/`, `src/`, `config/`, `results/`, `inventory.json`, `manifest.json`.
- **Reglas**: nombres cortos; profundidad ≤ 8; sin reservados.

19.9 Investigación de librerías de grafos (2025)

- **web.run**: buscar 5 mejores; criterios (madurez, actividad, licencias, rendimiento).
- **Salida**: `REPORT.md` con tabla comparativa y citas.

19.10 Validador de convenciones de nombres

- **Salida**: `CheckNames.ps1`, `REPORT.md`.
- **Reglas**: patrón `YYYY-MM-DD` y caracteres válidos.

---

20) Tratamiento de errores y fallas

- **Errores de red/servicio**: si `web.run` falla, reporto y entrego lo posible sin navegación.
- **Fallas en artefactos opcionales**: continuo y marco el faltante como “degradación elegante”.
- **Límites excedidos**: propongo partición (múltiples bundles/segmentos) o renombrado.

---

21) Formato y legibilidad

- Encabezados jerárquicos (`#`, `##`, `###`).
- Listas numeradas/puntos para criterios y checklists.
- Código mínimo y sólo si aporta (scripts/regex/patrones).
- Tablas cuando agregan claridad; evitar tablas enormes que no se lean.

---

22) Internacionalización y lenguaje

- **Español** como idioma base (coherente con tu estilo).
- En código/comentarios, puedo usar **inglés técnico** si es estándar (p. ej., nombres de funciones).

- Fechas y números con convenciones ISO para portabilidad.

---

23) Consideraciones de desempeño

- Mantengo respuestas eficientes, pero priorizo **exactitud** y **completitud**.
- Para contenidos muy extensos, uso **canvas** (como este) para edición incremental.
- Si un cálculo pesado es innecesario, lo evito; si es útil, genero artefactos reproducibles.

---

24) Casos límite frecuentes y respuestas

- **Rutas largas**: propongo stems más cortos (acróNimos) y partición de carpetas.
- **Caracteres conflictivos**: normalizo a ASCII seguro para scripts.
- **Datos con comillas/UTF-8**: escapo correctamente; indico delimitadores.
- **Zonas horarias**: especifico TZ en reportes donde el timing importa.
- **Desacuerdo fuente A/B**: presento ambas y marco incertidumbre.

---

25) Actualización y mantenimiento del manual

- Este documento es un **artefacto vivo**. Puedes pedirme añadir secciones, plantillas y matrices de decisión adicionales.
- Recomiendo conservar una **versión fechada** y un CHANGELOG si evolucionan las reglas.

---

26) Glosario mínimo

- **Contrato de salida**: enumeración previa de artefactos (nombre, formato, ruta) antes de generarlos.
- **Degradación elegante**: continuar sin lo opcional si falla, preservando lo esencial.
- **Manifest**: archivo que lista artefactos, tamaños, hashes y metadatos.
- **Link-check**: verificación de que todos los enlaces `sandbox:/...` son válidos.

---

27) Matrices de decisión (compactas y accionables)

27.1 ¿Navegar o no?

- ¿La información **pudo cambiar** desde 2024-06? → **Sí** → Navega.
- ¿Hay **riesgo alto** si me equivoco (legal, costos, reputación)? → **Sí** → Navega.
- ¿El usuario **pide explícitamente** verificar/buscar? → **Sí** → Navega.
- En caso de duda → **Navega**.

27.2 ¿Canvas o respuesta directa?

- ¿El contenido es largo (≥ 2–3 pantallas) o **evolutivo**? → **Canvas**.
- ¿Necesitas **editar/guardar** para iterar? → **Canvas**.
- ¿Código preview (React/HTML)? → **Canvas (code/react|code/html)**.

27.3 ¿Crear archivos reales?

- ¿Necesitas **descargar/compartir**? → **Sí** → `python_user_visible` + enlaces.
- ¿Hay tablas/datos que te conviene manipular? → **DataFrame** + CSV/JSON.

27.4 ¿Citas obligatorias?

- ¿Usé `web.run`? → **Sí**.
- ¿Usé `file_search`? → **Sí**, usa marcador de esa herramienta.

---

28) Plantillas útiles (listas para reutilizar)

28.1 Contrato de salida (snippet)

```
**Contrato de salida**
- <stem>_<YYYY-MM-DD>.md
- <stem>_<YYYY-MM-DD>.csv
- <stem>_<YYYY-MM-DD>_bundle.zip
- hashes.txt, verify.ps1, verify.sh, manifest.json
```

28.2 Resumen JSON (snippet)

```
{"stem":"<stem>","status":"","sizes":{"<archivo>":<bytes>,"<otro>":<bytes>}}
```

28.3 Reglas de nombres (regex aproximada)

```
^[A-Za-z0-9][A-Za-z0-9_-]{0,60}(?:_[0-9]{4}-[0-9]{2}-[0-9]{2})(?:_[A-Za-z0-9-]
{1,30})?\.[A-Za-z0-9]{1,8}$
```

---

29) Cierre operativo

- Este reporte **normaliza** cómo interpreto y ejecuto tu configuración.
- Si deseas, puedo derivar de aquí **checklists ejecutables** o **plantillas** específicas (p. ej., `README` base, `manifest.json` base, `verify.*` estándar).
- Al solicitar nuevos trabajos, puedes referenciar secciones (ej.: “aplica 6.1 + 8 + 27.1”).

Fin del reporte.