Plantillas README CANON — Rag Corp (Pro-Senior)

Objetivo: unificar el estilo de documentación del repo.

- **README padre (index/portal):** explica la carpeta, su misión, límites, y contiene el **índice** de “hojas”.
- **README hoja (detalle):** súper detallado. Explica cómo funciona por dentro, contratos, errores, seguridad, performance, extensiones y troubleshooting.

Regla: los **README no llevan TARJETA CRC** (por tu norma), pero sí deben ser **operativos**, con intención y con recorridos rápidos.

---

1) Reglas de consistencia (aplican a todos los README)

1.1 Títulos

- Formato: `# <Área> — <Rol>` o `# <Área>`
- Siempre con un tagline 1 línea en “metáfora operativa”:
- “Como un **taller operativo**…”
- “Como un **router**…”
- “Como un **lector**…”
- “Como un **puente/adaptador**…”

1.2 Lenguaje

- Español claro y técnico.
- Hablar en términos de **responsabilidades**, **límites** e **invariantes**.
- Evitar texto “decorativo”: cada sección debe ayudar a operar o mantener.

1.3 “Rutas rápidas por intención” (obligatorio)

- Al inicio, incluir una lista de accesos rápidos para alguien que llega con una intención concreta.

1.4 Tablas obligatorias

- “Mapa del territorio” siempre presente.
- En README padre: tabla **corta**.
- En README hoja: tabla **completa**.

1.5 Troubleshooting (obligatorio)

- Debe incluir “síntoma → causa probable → dónde mirar → solución”.

1.6 Índice (obligatorio en README padre)

- El padre siempre cierra con “Ver también (Índice de hojas)”.

---

2) Plantilla CANON — README Padre (Index / Portal)

> Ubicación típica: `README.md` en la raíz de un área grande. Ejemplos: `apps/frontend/app/README.md`, `apps/backend/README.md`, `src/features/README.md`.

```
# <Nombre del área> — <Tagline humano>
> Como un **<taller/puerta/enrutador/central>**: <qué hace en humano>.

## 🎯 Misión
Este directorio es la **unidad <rol>** de <sistema>. Desde acá se <operar/definir/organizar> <X>.
El código/implementación de <Y> vive en `<ruta>`, mientras que acá se mantiene <Z>.

- Qué resuelve: <1–3 bullets>
- Qué no resuelve: <1–2 bullets con “y por qué”>

## ⛳ Si venís con una intención concreta (rutas rápidas)
- Arquitectura y capas → `./<subarea>/README.md`
- Punto de entrada principal → `./<archivo>`
- Configuración y envs → `./<doc>`
- Tests → `./<tests>/README.md`
- Troubleshooting → `./<doc>`

##    Qué SÍ hace
- <bullet>
- <bullet>
- <bullet>

## 📢 Qué NO hace (y por qué)
- No hace <X>. Razón: <razón>. Consecuencia: <impacto práctico>.
- No hace <Y>. Razón: <razón>. Consecuencia: <impacto práctico>.

## 🗝️ Mapa del territorio
| Recurso | Tipo | Responsabilidad (en humano) |
| :-- | :-- | :-- |
| `<archivo/carpeta>` | `<tipo>` | `<qué aporta>` |
| `<archivo/carpeta>` | `<tipo>` | `<qué aporta>` |

## 🧩 Arquitectura del área (límites y dependencias)
### Rol arquitectónico
- Rol: <router root / feature module / infra adapter / ui kit / tooling>

### Reglas de límites (imports/dependencias)
- Puede importar: <...>
- No puede importar: <...>
- Regla fuerte: <en 1–2 líneas>
```

````
### Invariantes (cosas que no se rompen)
- Invariante 1: <...>
- Invariante 2: <...>

##    ¿Cómo funciona por dentro?
Input → Proceso → Output (nivel macro, operativo).

- **Input:** <...>
- **Proceso:** <...>
- **Output:** <...>

##   🎷 Guía de uso (Snippets)
```ts
// Por qué: ejemplo mínimo de uso.
````

```
# Por qué: comando operativo típico.
```

🧩 Cómo extender sin romper nada

- Paso 1: <qué tocar>
- Paso 2: <dónde cablear>
- Paso 3: <qué tests agregar>
- Invariantes a respetar: <...>

🎺 Troubleshooting

- **Síntoma:** <...>
- **Causa probable:** <...>
- **Dónde mirar:** <archivo/ruta>
- **Solución:** <pasos>

Ver también (Índice de hojas)

- `./<subarea>/README.md` — <qué explica>
- `./<subarea>/README.md` — <qué explica>
- `./<subarea>/README.md` — <qué explica>

````
---

# 3) Plantilla CANON — README Hoja (Detalle máximo)

> Ubicación típica: `README.md` dentro de un submódulo.
> Ejemplos: `apps/backend/app/infrastructure/parsers/README.md`, `apps/
frontend/app/(app)/README.md`.

```md
# <Nombre del submódulo>
````

```
> Como un **<lector/puente/filtro/orquestador>**: <qué hace en humano>.

## 🎯 Misión
Este módulo implementa <X> a través de <estrategias/puertos/
adaptadores>.
Resuelve: <lista corta>. Mantiene invariantes: <lista corta>.

- Punto de entrada: `<archivo>`
- Contratos/DTOs: `<archivo>`
- Errores tipados: `<archivo>`

## ⛳ Recorridos rápidos por intención
- **Quiero ver el punto de entrada** → `<archivo>`
- **Quiero ver selección/estrategia** → `<archivo>`
- **Quiero ver normalización/límites** → `<archivo>`
- **Quiero ver errores y contratos** → `<archivo>`
- **Quiero ver ejemplos de uso** → sección “Guía de uso”

##    Qué SÍ hace
- <bullet>
- <bullet>
- <bullet>

## 📢 Qué NO hace (y por qué)
- No hace <X>. Razón: <razón>. Impacto: <impacto práctico>.
- No hace <Y>. Razón: <razón>. Impacto: <impacto práctico>.

## 🔎 Mapa del territorio
| Recurso | Tipo | Responsabilidad (en humano) |
| :-- | :-- | :-- |
| `<archivo>` | Archivo | <...> |
| `<archivo>` | Archivo | <...> |
| `<carpeta>` | Carpeta | <...> |

##    ¿Cómo funciona por dentro? (paso a paso)

### 1) Entrada principal: `<punto_de_entrada>`
- **Input:** <...>
- **Proceso:**
  1. <...>
  2. <...>
  3. <...>
- **Output:** <...>

### 2) Estrategias / Registry / Selección (si aplica)
- **Input:** <...>
- **Proceso:** <...>
- **Output:** <...>

### 3) Errores y contratos (tipado)
- Errores:
```

```
  - `<ErrorA>`: <qué significa>
  - `<ErrorB>`: <qué significa>
- Contratos/DTOs:
  - `<ContratoA>`: <qué contiene>

## 🪁 Seguridad y límites defensivos
- Validaciones: <...>
- Límites: <...>
- Mitigaciones: <...>
- No confiar en: <...>

##    Performance
- Timeouts/Abort: <...>
- Estrategias para evitar trabajo innecesario: <...>
- Caching (si aplica): <...>

##    Conexiones y roles
- Rol arquitectónico: <...>
- Recibe órdenes de: <...>
- Llama a: <...>
- Reglas de límites: <...>

##   🎷 Guía de uso (Snippets)
```ts
// Por qué: ejemplo mínimo real.
```

```
# Por qué: comando típico.
```

🧩 Cómo extender sin romper nada

1) Implementar/crear `<nuevo componente/archivo>` 2) Registrar en `<lugar>` 3) Agregar tests en `<path>` 4) Respetar invariantes: <...>

Checklist de calidad (DoD del submódulo)

- [ ] CRC en archivos (excepto README/JSON)
- [ ] Tipos estrictos y APIs pequeñas
- [ ] Errores tipados/normalizados
- [ ] Seguridad revisada
- [ ] Performance revisada
- [ ] Tests agregados/ajustados
- [ ] No rompe boundaries
- [ ] Documentación actualizada

🎺 Troubleshooting

- **Síntoma:** <...>
- **Causa probable:** <...>
- **Dónde mirar:** <...>
```

• **Solución:** <...>

Ver también

• `../README.md` (padre)
• `../../<otra area>/README.md` ```

---

4) Reglas “Padre vs Hoja” (resumen)

README Padre (Index)

• Responde: **¿Qué es esta carpeta y cómo me oriento?**
• Contiene: misión + mapa + límites + guía macro + **índice de hojas**.
• No baja al micro de cada archivo.

README Hoja (Detalle)

• Responde: **¿Cómo funciona esto por dentro y cómo lo extiendo?**
• Contiene: paso a paso + contratos + errores + seguridad + performance + troubleshooting.

---

5) Convención práctica para el árbol (recomendación)

• Cada carpeta “grande” tiene README padre.
• Cada submódulo que tenga lógica propia tiene README hoja.
• Los README padres siempre linkean a las hojas.

Ejemplo (frontend/app): - `apps/frontend/app/README.md` (padre) - `apps/frontend/app/(app)/README.md` (hoja) - `apps/frontend/app/(auth)/README.md` (hoja) - `apps/frontend/app/(app)/admin/README.md` (hoja) - `apps/frontend/app/(app)/workspaces/README.md` (hoja)