Code Smells (Crawler) - A4

Resumo

Categoria (Refactoring Guru)	**Smell**	**Refatoracao aplicada**
Bloaters - Long Parameter List	Pipeline precisava passar 5 argumentos dispersos para obter o curriculo	Introduce Parameter Object com `CurriculumParams` e leitura via `_read_html`
Couplers - Global Data	Funcoes HTTP dependiam da constante global `BASE`	Dependency Injection utilizando `CrawlerSettings` e helper `_url()`
Dispensables - Duplicate Code	Cada chamada salvava HTML montando nome de arquivo manualmente	Extract Method em `utils.io_raw.save_raw` com hash/time para nome unico
Change Preventers - Divergent Change	Ajustar estrategia exigia tocar CLI, pipeline e funcoes HTTP ao mesmo tempo	Introduce Strategy (`AjaxStrategy` / `FullPageStrategy`) e seletor `fetch_with_strategy`

Bloaters - Long Parameter List

- **Onde ocorria:** `crawler/src/crawler_app/collectors/enumerate_pipeline.py:130` (antes do refactor).
- **Sintoma:** A coleta de cada modalidade repassava `curso_id`, `catalogo_id`, `modalidade_id`, `periodo_id`, `cp` separadamente para `fetch_arvore_with_params`, poluindo a assinatura e dificultando evolucao.
- **Risco:** Inclusao de um novo filtro ou estrategia demandava adicionar argumentos em varias funcoes, elevando chance de inconsistencias e bugs silenciosos.
- **Refactoring aplicado:** *Introduce Parameter Object* (`CurriculumParams`) encapsulando o conjunto de parametros e reaproveitado na Strategy e nos parsers.
- **Evidencia "Antes/Depois":**

```
-            html_arvore = fetch_arvore_with_params(
-                session,
-                curso_id=curso_id,
-                catalogo_id=CATALOGO_TARGET,
-                modalidade_id=mid,
-                periodo_id=PERIODO_TARGET,
-                cp=CP_TARGET,
-                raw_dir=arvore_dir,
-            )
+            params = CurriculumParams(
+                curso_id=curso_id,
+                catalogo_id=catalogo_int,
```

```
+                modalidade_id=modalidade_id,
+                periodo_id=str(PERIODO_TARGET),
+                cp=str(CP_TARGET),
+            )
+            raw_path = fetch_with_strategy(session, settings, params, arvore_dir)
+            html_arvore = _read_html(raw_path)
```

- **Resultados:** Parametros passaram a viajar como objeto imutavel, simplificando testes e reduzindo a necessidade de sincronizar funcoes quando o escopo muda.

Couplers - Global Data

- **Onde ocorria:** `crawler/src/crawler_app/collectors/arvore_http.py:1-70` (versao anterior usava `BASE` global).
- **Sintoma:** As funcoes HTTP dependiam da constante `BASE` definida em `collectors/config.py`, acoplando endpoints ao estado global e dificultando apontar para ambientes alternativos.
- **Risco:** Alterar `base_url` exigia editar modulos diferentes ou confiar em mocks complexos, aumentando risco de configuracao incorreta entre dev e prod.
- **Refactoring aplicado:** *Dependency Injection* com `CrawlerSettings` (Refactoring Guru -> Couplers/Global Data) e helper `_url()` para montar caminhos a partir da configuracao.
- **Evidencia "Antes/Depois":**

```
-from .config import BASE
-
-def fetch_arvore_page(session: requests.Session, *, raw_dir: str, label: str,
curso_id: Optional[str] = None) -> str:
-    params = {"curso": curso_id} if curso_id else {}
-    resp = session.get(f"{BASE}/arvore/", params=params, timeout=30)
+from ..config.settings import CrawlerSettings
+
+def fetch_arvore_page(
+    session: requests.Session,
+    settings: CrawlerSettings,
+    *,
+    raw_dir: str,
+    label: str,
+    curso_id: Optional[int] = None,
+) -> str:
+    params = {"curso": curso_id} if curso_id is not None else {}
+    resp = session.get(
+        _url(settings, "/arvore/"),
+        params=params,
+        timeout=settings.timeout_s,
+    )
```

- **Resultados:** Agora basta ajustar `.env` ou passar `--base-url` na CLI para redirecionar a coleta; testes podem injetar URLs fake e timeouts customizados.

Dispensables - Duplicate Code

- **Onde ocorria:** `crawler/src/crawler_app/utils/io_raw.py:8-18` (assinatura antiga de `save_raw`).
- **Sintoma:** Cada chamada precisava montar nomes unicos e garantir `ensure_dir` manualmente, gerando repeticao de logica e risco de sobrescrita de arquivos.
- **Risco:** Em execucoes seguidas, HTMLs eram sobrescritos porque os nomes eram previsiveis; tambem aumentava o ruido ao adicionar novos coletores.
- **Refactoring aplicado:** *Extract Method* com responsabilidade de naming em `save_raw`, usando hash do `name_hint` mais timestamp.
- **Evidencia "Antes/Depois":**

```
-def save_raw(base_dir: str, filename: str, content: str) -> str:
-    ensure_dir(base_dir)
-    fullpath = os.path.join(base_dir, filename)
-    with open(fullpath, "w", encoding="utf-8") as f:
-        f.write(content if content is not None else "")
-    return fullpath
+def save_raw(html: str, raw_dir: str, name_hint: str) -> str:
+    ensure_dir(raw_dir)
+    key = hashlib.md5(name_hint.encode("utf-8")).hexdigest()
+    filename = f"{int(time.time())}_{key}.html"
+    path = os.path.join(raw_dir, filename)
+    with open(path, "w", encoding="utf-8") as f:
+        f.write(html if html is not None else "")
+    return path
```

- **Resultados:** Saidas RAW tornaram-se idempotentes, com colisoes improvaveis e sem duplicar logica de criacao de nome em cada Strategy ou helper.

Change Preventers - Divergent Change

- **Onde ocorria:** `crawler/src/crawler_app/collectors/enumerate_pipeline.py` executava fluxo HTTP direto sem abstrair estrategia.
- **Sintoma:** Alterar a forma de busca (AJAX x pagina cheia) exigia modificar CLI, pipeline e helpers, causando "shotgun surgery" sempre que o GDE mudava comportamento.
- **Risco:** Dificuldade de aplicar hotfix rapido em campo; fallback manual era esquecido e a coleta falhava por completo diante de erros do endpoint AJAX.
- **Refactoring aplicado:** *Introduce Strategy* com `fetch_with_strategy` centralizando a decisao e fallback automatico.
- **Evidencia "Antes/Depois":**

```
-from .arvore_http import (
-    polite_sleep,
-    fetch_arvore_page,
-    fetch_modalidades_fragment,
-    fetch_arvore_with_params,
-)
+from .arvore_http import fetch_arvore_page, fetch_modalidades_fragment,
polite_sleep
+from .strategies import AjaxStrategy, FullPageStrategy
```

```
...
-            polite_sleep()
-            try:
-                html_arvore = fetch_arvore_with_params(
-                    session,
-                    curso_id=curso_id,
-                    catalogo_id=CATALOGO_TARGET,
-                    modalidade_id=mid,
-                    periodo_id=PERIODO_TARGET,
-                    cp=CP_TARGET,
-                    raw_dir=arvore_dir,
-                )
+            polite_sleep(settings)
+            params = CurriculumParams(
+                curso_id=curso_id,
+                catalogo_id=catalogo_int,
+                modalidade_id=modalidade_id,
+                periodo_id=str(PERIODO_TARGET),
+                cp=str(CP_TARGET),
+            )
+
+            try:
+                raw_path = fetch_with_strategy(session, settings, params,
arvore_dir)
+                html_arvore = _read_html(raw_path)
```

- **Resultados:** Qualquer nova estrategia (ex.: headless browser) pode ser adicionada com classe dedicada, sem alterar CLI; fallback garante continuidade mesmo com falhas temporarias.

Mapa de rastreabilidade

- Long Parameter List -> commit `af938da8` ("AV4 - Code Smells") -> arquivos `collectors/enumerate_pipeline.py`, `types.py`.
- Global Data -> commit `af938da8` -> arquivo `collectors/arvore_http.py`.
- Duplicate Code -> commit `af938da8` -> arquivo `utils/io_raw.py`.
- Divergent Change -> commit `af938da8` -> arquivos `collectors/enumerate_pipeline.py`, `collectors/strategies/*.py`, `cli.py`.

Como validar

- Executar `python -m src.crawler_app.cli collect --strategy auto` e verificar logs de fallback.
- Rodar `python -m src.crawler_app.cli curriculum --course-id 34 --year 2022 --strategy full` e checar HTML salvo em `crawler/data/raw`.
- Inspecionar `crawler/data/json` para confirmar deduplicacao de disciplinas.
- Conferir logs para `AjaxStrategy falhou` quando `CRAWLER_STRATEGY=auto` aciona o fallback.

Referencias

- Refactoring Guru - Code Smells: https://refactoring.guru/refactoring/smells
- Refactoring Guru - Bloaters / Long Parameter List

- Refactoring Guru - Couplers / Global Data
- Refactoring Guru - Dispensables / Duplicate Code
- Refactoring Guru - Change Preventers / Divergent Change