Sintesi documentazione

README.md

- Pipeline genera KB Markdown con pre-onboarding, tagging e onboarding completo
- Output per cliente isolato (`raw/`, `book/`, `semantic/`, `config/`, `logs/`)
- Variabili d'ambiente sensibili: `SERVICE_ACCOUNT_FILE`, `DRIVE_ID`, `GITHUB_TOKEN`, `LOG_REDACTION`, `ENV`, `CI`
- Log centralizzati, mascheramento segreti e path-safety (`is_safe_subpath`)
- Exit codes tipizzati (`ConfigError`, `PreviewError`, `PushError`)

docs/architecture.md

- RAW sempre locale; Drive solo nel pre-onboarding
- Separazione ruoli: orchestratori (UX/CLI) vs moduli tecnici senza prompt/exit
- Repository strutturato per moduli riusabili e adapter uniformi
- Invarianti: path-safety, scritture atomiche, log strutturati con `run_id`

docs/developer_guide.md

- Redazione log centralizzata via `compute_redact_flag`
- `ClientContext.load` come entry-point unico, con helper `_load_env` e `compute_redact_flag`
- Orchestratori solo UX; moduli tecnici senza `sys.exit()` o `input()`
- Principi: modularità, idempotenza, scritture sicure, consistenza API

docs/coding_rule.md

- Obbligo di type hints, docstring brevi, PEP 8, import ordinati
- Orchestratori vs moduli: `sys.exit()` solo negli orchestratori
- Path traversal: usare `is_safe_subpath`; scritture atomiche `safe_write_*`
- Config via `yaml.safe_load`, env centralizzate, cache invalidabile
- Comandi esterni tramite `proc_utils.run_cmd` con timeout/retry

docs/policy_push.md

- Push incrementale default; force push solo con `--force-push` + `--force-ack` e allow-list branch

docs/versioning_policy.md

- SemVer: MAJOR incompatibile, MINOR feature, PATCH bugfix; ogni PR che tocca API/CLI aggiorna CHANGELOG e doc corrispondenti

docs/user_guide.md

- Modalità interattiva vs CLI; pre-onboarding, tagging, onboarding con preview/push opzionali
- Log centralizzati, exit codes tipizzati, troubleshooting Docker/push/slug

Decisioni architetturali vincolanti

- Unico punto d'ingresso `ClientContext.load`; orchestratori separati dalla logica
- Redazione log gestita in `env_utils.compute_redact_flag` e propagata nei logger
- Path-safety e scritture atomiche obbligatorie per ogni output
- Push GitHub sicuro by default; force push governato da double-flag e allow-list

---

Mappatura codebase

Modulo/Pacchetto	Responsabilità principali	Dipendenze salienti
`src/pre_onboarding.py`	Setup locale e Drive, gestione UX/CLI, exit codes	`pipeline.context`, `pipeline.drive_utils`, `pipeline.config_utils`
`src/tag_onboarding.py`	Copia/Download PDF, generazione tags, validazione	`pipeline.drive_utils`, `pipeline.path_utils`, `pipeline.file_utils`
`src/onboarding_full.py`	Conversione PDF→MD, enrichment, preview, push	`pipeline.content_utils`, `adapters.preview`, `pipeline.github_utils`
`pipeline/context.py`	Dataclass `ClientContext`, load env/config, path init	`env_utils`, `path_utils`, `logging_utils`
`pipeline/path_utils.py`	Validazione slug, path traversal guard, sanitize	`yaml`, `logging_utils`
`pipeline/file_utils.py`	Scritture atomiche e guardie path (duplicati)	`path_utils` (should), `logging_utils`
`pipeline/env_utils.py`	Lettura variabili d'ambiente, redazione log	`dotenv`, `ConfigError`
`pipeline/content_utils.py`	Conversione e generazione markdown, fingerprint, summary/readme	`logging_utils`, `config_utils`, `path_utils`
`pipeline/github_utils.py`	Push GitHub con incrementale/force governance	`proc_utils`, `env_utils`, `path_utils`

`adapters/preview.py`	Avvio/stop preview Docker	`proc_utils`, `logging_utils`	
`semantic/*`	Estrazione e validazione tag semantici	`yaml`, `logging_utils`	

Doc ↔ Codice Matrix

Regola	Previsto	Osservato	Esito	Note
Path traversal prevention	`is_safe_subpath` o `ensure_within` ovunque si scriva su disco	`content_utils.generate_summary_markdown` verifica con `is_safe_subpath`	Parziale	usa `is_safe_subpath` ma non `ensure_within`
Scritture atomiche per config	`safe_write_*` per file critici	`_ensure_config` usa `shutil.copy`	Non aderente	manca atomicità
Orchestratori isolati da logica tecnica	`sys.exit` solo negli orchestratori	`pre_onboarding.py` usa `sys.exit` correttamente	Aderente	moduli tecnici non chiamano exit
Redazione log centralizzata	`compute_redact_flag` e logger con `redact_logs`	`tag_onboarding_main` calcola `redact_logs` se assente	Aderente	
CSV generati in modo sicuro	usare libreria `csv`	`_emit_tags_csv` scrive CSV manuale senza quoting	Non aderente	rischio separatori

Tabelle di review

A1 – Correttezza tecnica

Principio (fonte)	Esempio nel codice	Raccomandazione

Path-safety e atomicità obbligatorie	`file_utils.ensure_within` duplicato rispetto a `path_utils.ensure_within` *(Da migliorare)*	Consolidare `ensure_within` in `path_utils` ed eliminarne la copia in `file_utils`.
Config template scritto senza atomicità	`_ensure_config` usa `shutil.copy` *(Da migliorare)*	Copiare il template con `safe_write_bytes` e `ensure_within`.
Redazione log deterministica	`compute_redact_flag` implementato correttamente *(OK)*	—

Attività suggeritaUnificare ensure_within e applicare atomicità al config

Inizia l'attività

A2 – Uso corretto di funzioni/interfacce

Principio (fonte)	Esempio nel codice	Raccomandazione
Orchestratori gestiscono solo UX/exit	`pre_onboarding.py` usa `ensure_valid_slug` e `sys.exit` solo nel main *(OK)*	—
Env centralizzate via `env_utils.get_env_var`	`_load_env` usa `get_env_var` e `get_bool` *(OK)*	—
Output moduli batch-safe	`_emit_tags_csv` scrive CSV manuale senza quoting *(Da migliorare)*	Usare `csv.writer` con quoting minimale.

Attività suggeritaRefactor `_emit_tags_csv` con `csv.writer`

Inizia l'attività

A3 – Ottimizzazione dei flussi

Principio (fonte)	Esempio nel codice	Raccomandazione

Import ordinati e al top-level	`import shutil` dentro `_copy_local_pdfs_to_raw` *(Da migliorare)*	Spostare l'import in testa al file.
Path-safety esplicita per output markdown	`generate_summary_markdown` e `generate_readme_markdown` usano solo `is_safe_subpath` *(Parziale)*	Applicare `ensure_within` sui file generati per fail-fast.

Attività suggeritaPulizia import e guardia path nella generazione markdown

Inizia l'attività

B – Eleganza & Best practice

Principio	Metica/Evidenza	Rischio	Impatto atteso	Sforzo
Duplicazione helper (`ensure_within`)	due implementazioni divergenti	Medio	Riduce bug e aumenta coerenza	Quick win
CSV manuale senza quoting	`_emit_tags_csv` stringhe concatenate	Alto	Migliora robustezza I/O	Quick win
Import dentro funzione	`import shutil` interno	Basso	Performance e stile	Quick win
Path guard mancante in generatori MD	solo `is_safe_subpath` senza `ensure_within`	Medio	Fail-fast su path traversal	Medio
Config copy non atomica	`shutil.copy` in `_ensure_config`	Medio	Evita config corrotte	Medio

Executive Summary

Valutazione complessiva: 82/100

Punti di forza

1. Architettura modulare con orchestratori separati dalla logica tecnica.
2. Logging strutturato e redazione centralizzata ben implementati.
3. Test coverage di moduli critici (path_utils, logging_utils, github_utils).

Rischi principali

1. Duplicazione della guardia `ensure_within` può generare incoerenze.
2. Serializzazione CSV manuale suscettibile a errori di parsing.
3. Scrittura `config.yaml` non atomica rischia file parziali.

Quick wins commit-ready

- Centralizzare `ensure_within`.
- Refactor `_emit_tags_csv` con `csv.writer`.
- Spostare import `shutil` e aggiungere `ensure_within` nei generatori markdown.

Prioritizzazione

- Alta: unificazione `ensure_within`, CSV sicuro.
- Media: atomicità `_ensure_config`, guardia path generatori MD.
- Bassa: pulizia import.

Roadmap

1. Refactor sicurezza I/O (Alta) – consolidare path-safety e atomic write.
2. Hardening flussi tagging (Media) – CSV robusto e validazioni aggiuntive.
3. Polish & tests (Bassa) – cleanup import, aumentare test per file_utils/env_utils.

Proposte operative (patch)

1. Deduplicare `ensure_within`

```
--- src/pipeline/file_utils.py +++ src/pipeline/file_utils.py @@ -from
.exceptions import ConfigError -from .logging_utils import
get_structured_logger - -_logger =
get_structured_logger("pipeline.file_utils") - - -def ensure_within(base:
Path, target: Path) -> None: - """Garantisce che `target` risieda sotto
`base` una volta risolti i path.""" - try: - base_r = Path(base).resolve() -
tgt_r = Path(target).resolve() - except Exception as e: - raise
ConfigError(f"Impossibile risolvere i path: {e}", file_path=str(target)) from
e - - if not str(tgt_r).startswith(str(base_r)): - raise ConfigError(f"Path
traversal rilevato: {tgt_r} non è sotto {base_r}", file_path=str(target))
+from .exceptions import ConfigError +from .logging_utils import
get_structured_logger +from .path_utils import ensure_within # reuse
canonical implementation + +_logger =
get_structured_logger("pipeline.file_utils")
```

Riduce duplicazioni e mantiene una sola fonte di verità per la validazione path.

Attività suggeritaConvergere su ensure_within in file_utils

Inizia l'attività

2. Scrittura config atomica

```
--- src/pipeline/context.py +++ src/pipeline/context.py @@ -
config_path.parent.mkdir(parents=True, exist_ok=True) - template_config =
Path("config") / "config.yaml" - if not template_config.exists(): +
config_path.parent.mkdir(parents=True, exist_ok=True) + template_config =
Path("config") / "config.yaml" + if not template_config.exists(): raise
ConfigError( f"Template config.yaml globale non trovato: {template_config}",
slug=slug, file_path=template_config, ) - shutil.copy(template_config,
config_path) + from pipeline.file_utils import safe_write_bytes +
safe_write_bytes(config_path, template_config.read_bytes(), atomic=True)
return config_path
```

Garantisce scrittura atomica del `config.yaml` iniziale.

Attività suggeritaScrittura atomica di config.yaml

Inizia l'attività

3. CSV robusto nel tagging

```
--- src/tag_onboarding.py +++ src/tag_onboarding.py @@ - lines =
["relative_path,suggested_tags"] - for rel, tags in rows: -
lines.append(f"{rel},{tags}") - safe_write_text(csv_path, "\n".join(lines) +
"\n", encoding="utf-8", atomic=True) + import csv + with csv_path.open("w",
encoding="utf-8", newline="") as f: + writer = csv.writer(f) +
writer.writerow(["relative_path", "suggested_tags"]) + for rel, tags in rows:
+ writer.writerow([rel, tags])
```

Evita problemi con virgole nei percorsi.

Attività suggeritaCSV con quoting standard

Inizia l'attività

4. Import al top-level e guardia path MD

```
--- src/tag_onboarding.py +++ src/tag_onboarding.py @@ -from pathlib import
Path -from typing import Optional, List +from pathlib import Path +from
typing import Optional, List +import shutil @@ - import shutil for src in
pdfs: try: rel = src.relative_to(src_dir)

--- src/pipeline/content_utils.py +++ src/pipeline/content_utils.py @@
summary_path = md_dir / "SUMMARY.md" - if not is_safe_subpath(summary_path,
md_dir): - raise PipelineError( - f"Path di output non sicuro per SUMMARY.md:
{summary_path}", - slug=context.slug, - file_path=summary_path, - ) +
ensure_within(md_dir, summary_path) @@ readme_path = md_dir / "README.md" -
if not is_safe_subpath(readme_path, md_dir): - raise PipelineError( - f"Path
```

```
di output non sicuro per README.md: {readme_path}", - slug=context.slug, -
file_path=readme_path, - ) + ensure_within(md_dir, readme_path)
```

Mantiene import ordinati e applica fail-fast sulla generazione markdown.

Attività suggeritaPulizia import e path guard

Inizia l'attività

5. Test file_utils assenti → aggiungerli

```
--- tests/test_file_utils.py (nuovo) +++ tests/test_file_utils.py @@ +from
pathlib import Path +import pytest + +from pipeline.file_utils import
safe_write_text +from pipeline.path_utils import is_safe_subpath + +def
test_safe_write_text_crea_backup(tmp_path: Path): + target = tmp_path /
"f.txt" + safe_write_text(target, "a") + safe_write_text(target, "b") + bak =
target.with_suffix(".txt.bak") + assert bak.exists() + assert
is_safe_subpath(bak, tmp_path)
```

Incrementa la copertura su operazioni critiche di I/O.

Attività suggeritaCopertura minima per file_utils

Inizia l'attività

Checklists finali

Pre-commit

- ✅ `ruff --fix`
- ✅ `black .`
- ✅ `pytest -q`

PR

- Descrizione chiara delle modifiche e motivazioni tecniche
- Log/output allegati per `_emit_tags_csv` e scrittura `config.yaml`
- Casi limite: percorsi con virgole, path traversal, force push non autorizzato
- Piano di rollback: revert commit in caso di regressioni

Prossimi passi consigliati

1. Core – Consolidare helper di path-safety e refactor tagging CSV (responsabile: maintainer pipeline).
2. Infra – Aggiungere test CI per `file_utils` e scenari di scrittura atomica.
3. QA – Integrare test di path traversal e quoting CSV nelle suite automatiche.