**📑 DOCUMENTAZIONE TECNICA - RIFATTORIZZAZIONE PROGETTO PUBLISCRIPT**

**1. 📖 Introduzione al Progetto**

**Publiscript** è un'applicazione progettata per automatizzare l'interazione con sistemi di AI (es. Genspark) allo scopo di generare libri, effettuare analisi di mercato e gestire dati editoriali.

**Scopo della rifattorizzazione:**

* Rendere il codice più modulare e manutenibile.
* Separare chiaramente le responsabilità (UI, AI, Framework, Analisi).
* Consentire in futuro di sostituire o integrare nuove AI senza modificare il core.

**Principi da rispettare:**

* **Non alterare** le logiche interne delle funzioni e classi esistenti.
* **Spostare fisicamente** il codice nei nuovi moduli.
* **Aggiornare correttamente gli import.**

**2. 📂 Struttura finale ad albero**

plaintext

CopiaModifica

publiscript/

│

├── ai\_interfaces/

│ ├── genspark\_driver.py

│ ├── browser\_manager.py

│ ├── file\_text\_utils.py

│ └── interaction\_utils.py

│

├── framework/

│ ├── crisp\_framework.py

│ ├── crisp\_extractors.py

│ ├── crisp\_utils.py

│ └── analysis/

│ ├── market\_analysis.py

│ ├── buyer\_persona.py

│ └── gap\_analysis.py

│

├── ui/

│ ├── app\_launcher.py

│ ├── ai\_connection.py

│ ├── chat\_manager.py

│ ├── cooldown\_manager.py

│ ├── book\_builder.py

│ ├── log\_utils.py

│ └── components.py

│

└── main.py

**3. 📜 Dettaglio file per file**

**ai\_interfaces/**

| **File** | **Descrizione** | **Provenienza** |
| --- | --- | --- |
| genspark\_driver.py | Gestisce login, invio richieste all'AI Genspark | genschat.py |
| browser\_manager.py | Setup e gestione browser Selenium | genschat.py |
| file\_text\_utils.py | Pulizia testo, gestione file | genschat.py |
| interaction\_utils.py | Invio prompt, gestione input box | genschat.py |

**framework/**

| **File** | **Descrizione** | **Provenienza** |
| --- | --- | --- |
| crisp\_framework.py | Struttura progetto libro CRISP | crisp\_framework.py |
| crisp\_extractors.py | Estrazione risposte e dati | crisp\_extractors.py |
| crisp\_utils.py | Utility di supporto | crisp\_utils.py |

**framework/analysis/**

| **File** | **Descrizione** | **Provenienza** |
| --- | --- | --- |
| market\_analysis.py | Estrazione dati mercato e concorrenza | crisp\_extractors.py |
| buyer\_persona.py | Profilazione del buyer persona | crisp\_extractors.py |
| gap\_analysis.py | Analisi recensioni negative e opportunità | crisp\_extractors.py |

**ui/**

| **File** | **Descrizione** | **Provenienza** |
| --- | --- | --- |
| app\_launcher.py | Avvio GUI/CLI Gradio | genschat\_gui.py |
| ai\_connection.py | Connessione browser/AI | genschat\_gui.py |
| chat\_manager.py | Backup, gestione contesto | genschat\_gui.py |
| cooldown\_manager.py | Gestione pause/cooldown richieste | genschat\_gui.py |
| book\_builder.py | Costruzione libro usando CRISP | genschat\_gui.py |
| log\_utils.py | Gestione log | genschat\_gui.py |
| components.py | (Futuro) Componenti GUI Gradio | genschat\_gui.py |

**root/**

| **File** | **Descrizione** | **Provenienza** |
| --- | --- | --- |
| main.py | Entry point: avvia GUI/CLI | Nuovo |

**4. 🔄 Import da correggere**

**Regola generale:**

* Tutti i vecchi import relativi vanno aggiornati in base alla nuova struttura.
* Mai lasciare riferimenti diretti tipo import genschat.

**Alcuni esempi chiave:**

| **Prima** | **Dopo** |
| --- | --- |
| from crisp\_utils import replace\_variables | from publiscript.framework.crisp\_utils import replace\_variables |
| from crisp\_framework import CRISPFramework | from publiscript.framework.crisp\_framework import CRISPFramework |
| from genschat import setup\_browser | from publiscript.ai\_interfaces.browser\_manager import setup\_browser |

**5. 📈 Schema struttura completa**

plaintext

CopiaModifica

publiscript/

│

├── ai\_interfaces/ (gestione AI e browser)

├── framework/ (gestione logica di generazione libri)

│ └── analysis/ (moduli di analisi)

├── ui/ (gestione interfaccia Gradio e sessione)

└── main.py (entry point generale)

**6. 📈 Schema a flusso del funzionamento**

plaintext

CopiaModifica

[ main.py ]

↓

[ app\_launcher.py (avvio interfaccia) ]

↓

[ book\_builder.py (costruzione libro) ]

↓

[ crisp\_framework.py (framework CRISP) ]

↓

[ analysis modules (mercato, buyer, gap) ]

↓

[ ai\_interfaces (interazione AI e browser) ]

↓

[ Browser Genspark ]

**7. ⚙️ Procedura operativa per rifattorizzazione**

1. **Creare la nuova struttura di cartelle** secondo lo schema.
2. **Spostare il codice** nei nuovi file seguendo la mappatura dettagliata.
3. **Correggere tutti gli import** nei file coinvolti.
4. **Verificare** che ogni modulo venga caricato senza errori (ImportError ecc.).
5. **Eseguire un test completo** avviando main.py.
6. **Controllare** che la generazione libro e gestione AI funzioni come prima.

**8. 🚫 Errori comuni da evitare**

* ❗ Modificare le logiche interne delle funzioni.
* ❗ Lasciare riferimenti sbagliati (import genschat).
* ❗ Mischiare codice che appartiene a livelli diversi (es: logica AI dentro UI).
* ❗ Dimenticare i \_\_init\_\_.py dentro ogni cartella.

**9. ✅ Checklist finale**

* Cartelle create correttamente.
* File spostati secondo il piano.
* Import aggiornati.
* Codice funzionante senza errori.
* Test di generazione libro e analisi superato.
* Pronto per aggiungere nuove AI in futuro.

**🏁 FINE DOCUMENTAZIONE TECNICA**