**Rifattorizzazione di book\_builder.py**

book\_builder.py (ORIGINALE)

├─────────────────────────────────────────────────────────────┐

│ │

│ CLASSE AIBookBuilder: │

│ ├── \_\_init\_\_ │

│ ├── metodi UI e gestione interfaccia │

│ │ ├── create\_interface │

│ │ ├── load\_analysis\_results │

│ │ ├── format\_analysis\_results\_html │

│ │ ├── show\_feedback │

│ │ └── update\_analysis\_status │

│ │ │

│ ├── metodi interazione AI │

│ │ ├── send\_to\_genspark │

│ │ ├── get\_last\_response │

│ │ ├── handle\_context\_limit │

│ │ └── reset\_context\_manual │

│ │ │

│ ├── metodi analisi e framework │

│ │ ├── \_analyze\_market\_crisp │

│ │ ├── \_analyze\_market\_legacy │

│ │ ├── execute\_prompt │

│ │ └── execute\_with\_updates │

│ │ │

│ ├── metodi utility file/testo │

│ │ ├── replace\_variables\_advanced │

│ │ ├── check\_unresolved\_placeholders │

│ │ ├── get\_clean\_input\_box │

│ │ └── safe\_text\_input │

│ │ │

│ └── metodi gestione database │

│ ├── load\_projects\_list │

│ ├── recupera\_ultimo\_progetto │

│ ├── ripristina\_ultima\_analisi │

│ └── delete\_project │

└─────────────────────────────────────────────────────────────┘

│

│ RIFATTORIZZAZIONE

▼

┌─────────────────────────┬─────────────────────────┬─────────────────────────┐

│ │ │ │

│ ui/book\_builder.py │ ai\_interfaces/ │ framework/ │

│ ├── classe AIBookBuilder│ ├── genspark\_driver.py │ ├── crisp\_framework.py │

│ │ ├── \_\_init\_\_ │ │ ├── send\_to\_genspark│ │ │

│ │ ├── create\_interface│ │ ├── get\_last\_response│ │ │

│ │ ├── format\_results │ │ └── reset\_context │ │ │

│ │ ├── show\_feedback │ │ │ │ │

│ │ ├── analyze\_market │ ├── browser\_manager.py │ ├── analysis/ │

│ │ └── update\_status │ │ ├── handle\_context │ │ ├── market\_analysis.py│

│ │ │ │ └── setup\_browser │ │ │ ├── \_analyze\_market\_crisp│

│ ├── ui/log\_utils.py │ │ │ │ │ └── \_analyze\_market\_legacy│

│ │ └── log\_history │ ├── interaction\_utils.py │ │ │

│ │ │ │ ├── get\_clean\_input │ │ ├── buyer\_persona.py │

│ └── ui/components.py │ │ ├── safe\_text\_input │ │ └── gap\_analysis.py │

│ │ │ └── click\_send │ │ │

│ │ │ │ └── crisp\_extractors.py│

│ │ └── file\_text\_utils.py │ ├── replace\_variables│

│ │ ├── replace\_variables│ └── extract\_data │

│ │ └── check\_placeholders│ │

│ │ │ │

└─────────────────────────┴─────────────────────────┴─────────────────────────┘

**Suddivisione dettagliata dei metodi:**

**1. ui/book\_builder.py (mantiene la classe AIBookBuilder)**

* \_\_init\_\_
* create\_interface
* format\_analysis\_results\_html
* show\_feedback
* update\_analysis\_status
* load\_analysis\_results
* Versioni semplificate degli altri metodi che ora delegheranno ai moduli appropriati

**2. ai\_interfaces/genspark\_driver.py**

* send\_to\_genspark
* get\_last\_response
* check\_for\_generation\_error

**3. ai\_interfaces/browser\_manager.py**

* setup\_browser
* handle\_context\_limit
* reset\_context\_manual
* take\_debug\_screenshot

**4. ai\_interfaces/interaction\_utils.py**

* get\_clean\_input\_box
* safe\_text\_input
* click\_send\_button
* wait\_for\_stable\_response

**5. ai\_interfaces/file\_text\_utils.py**

* replace\_variables\_advanced
* check\_unresolved\_placeholders
* create\_focused\_context
* send\_essential\_context

**6. framework/analysis/market\_analysis.py**

* \_analyze\_market\_crisp
* \_analyze\_market\_legacy
* \_filter\_legacy\_prompt\_sections

**7. framework/crisp\_framework.py**

(già esiste ma riceve metodi da book\_builder.py)

* execute\_prompt
* execute\_with\_updates
* Funzioni per la gestione delle fasi CRISP

Questo schema mostra chiaramente come distribuire i metodi attuali di book\_builder.py nei nuovi file della struttura rifattorizzata, mantenendo la logica esistente ma migliorando l'organizzazione e la modularità.

**Prossimi passi per la rifattorizzazione**

**1. Creare la struttura delle cartelle finale**

**publiscript/**

**├── ai\_interfaces/**

**├── framework/**

**│ └── analysis/**

**└── ui/**

**2. Iniziare a spostare i file nelle cartelle appropriate**

**Iniziamo dagli elementi più indipendenti per ridurre al minimo le dipendenze circolari. Ecco un piano in sequenza:**

**a) Creare i moduli di ai\_interfaces/**

1. **browser\_manager.py - Contiene funzioni per gestire il browser Selenium:**
   * **setup\_browser**
   * **check\_login**
   * **handle\_context\_limit**
   * **reset\_context\_manual**
2. **interaction\_utils.py - Funzioni per interagire con l'interfaccia Genspark:**
   * **get\_clean\_input\_box**
   * **safe\_text\_input**
   * **click\_send\_button**
   * **wait\_for\_stable\_response**
3. **file\_text\_utils.py - Funzioni per manipolare testo e file:**
   * **replace\_variables\_advanced**
   * **check\_unresolved\_placeholders**
   * **create\_focused\_context**
4. **genspark\_driver.py - Funzioni di alto livello per interagire con Genspark:**
   * **send\_to\_genspark**
   * **get\_last\_response**
   * **check\_for\_generation\_error**

**b) Creare i moduli del framework/**

1. **crisp\_framework.py (esistente) - Il core del framework CRISP**
2. **analysis/market\_analysis.py - Funzioni di analisi di mercato:**
   * **analyze\_market\_crisp**
   * **analyze\_market\_legacy**
   * **\_filter\_legacy\_prompt\_sections**

**c) Aggiornare i moduli UI/**

1. **book\_builder.py - La classe principale modificata per delegare ai moduli specializzati**
2. **app\_launcher.py - Per l'inizializzazione dell'interfaccia Gradio**
3. **chat\_manager.py - Per la gestione delle chat e dei log**

**3. Specificare quale file iniziare a spostare**

**Iniziamo dalla creazione del modulo ai\_interfaces/browser\_manager.py che è relativamente indipendente. Dovremo estrarre le seguenti funzioni dal file originale book\_builder.py:**

**Copy# publiscript/ai\_interfaces/browser\_manager.py**

**import time**

**from selenium import webdriver**

**from selenium.webdriver.common.by import By**

**from selenium.webdriver.support.ui import WebDriverWait**

**from selenium.webdriver.support import expected\_conditions as EC**

**from datetime import datetime**

**from pathlib import Path**

**def setup\_browser():**

**"""Configura e inizializza il browser Selenium."""**

**# Codice estratto dal metodo originale di AIBookBuilder o da altre funzioni correlate**

**# ...**

**def check\_login(driver):**

**"""Verifica se l'utente è loggato in Genspark."""**

**# Codice estratto dal metodo originale**

**# ...**

**def handle\_context\_limit(driver, log\_callback=None):**

**"""**

**Gestisce il limite di contesto in Genspark.**

**Args:**

**driver: Il driver Selenium**

**log\_callback: Funzione di callback per il logging (opzionale)**

**Returns:**

**bool: True se è stato rilevato e gestito un limite di contesto, False altrimenti**

**"""**

**# Codice estratto dal metodo handle\_context\_limit di AIBookBuilder**

**# ...**

**def reset\_context\_manual(driver, log\_callback=None):**

**"""**

**Reset completo del contesto: chiude la chat corrente, apre una nuova sessione,**

**e ricarica il contesto se necessario.**

**Args:**

**driver: Il driver Selenium**

**log\_callback: Funzione di callback per il logging (opzionale)**

**Returns:**

**bool: True se il reset è riuscito, False altrimenti**

**"""**

**# Codice estratto dal metodo reset\_context\_manual di AIBookBuilder**

**# ...**

**def take\_debug\_screenshot(driver, prefix, log\_callback=None):**

**"""Scatta uno screenshot per debugging."""**

**# Codice estratto dal metodo take\_debug\_screenshot di AIBookBuilder**

**# ...**

**Vuoi che procediamo con questo file per primo e ti mostri come dovrebbe essere dopo la rifattorizzazione?**

**1. Crea la struttura framework/**

Controlla se hai già creato questi file:

framework/

├── crisp\_framework.py # Framework CRISP principale

├── crisp\_extractors.py # Estrattori di dati

├── crisp\_utils.py # Utility per CRISP

└── analysis/ # Sottocartella per funzioni di analisi

├── market\_analysis.py

├── buyer\_persona.py

└── gap\_analysis.py

Se non li hai ancora creati, il prossimo passo sarebbe spostare o creare questi file. Il file più importante qui è framework/analysis/market\_analysis.py che dovrebbe contenere il metodo \_analyze\_market\_crisp che abbiamo corretto prima.

**2. Crea la struttura ui/**

Verifica se hai già creato questi file:

ui/

├── app\_launcher.py # Avvio dell'interfaccia

├── ai\_connection.py # Gestione connessione AI

├── chat\_manager.py # Gestione chat e messaggi

├── cooldown\_manager.py # Gestione degli intervalli di richiesta

├── book\_builder.py # Interfaccia principale del builder

└── log\_utils.py # Utilità di logging

Il file ui/book\_builder.py dovrebbe contenere la versione rifattorizzata della classe AIBookBuilder che ora delegherà molte delle sue funzionalità ai moduli specializzati.

Cartelle e file che già ho:  
  
/Framework: "C:\Users\a\Documents\PubliScript\_Refactored\framework\\_\_init\_\_.py" "C:\Users\a\Documents\PubliScript\_Refactored\framework\crisp\_extractors.py" "C:\Users\a\Documents\PubliScript\_Refactored\framework\crisp\_framework.py" "C:\Users\a\Documents\PubliScript\_Refactored\framework\crisp\_utils.py" "C:\Users\a\Documents\PubliScript\_Refactored\framework\\_\_pycache\_\_" "C:\Users\a\Documents\PubliScript\_Refactored\framework\analysis"

/ai\_interfaces "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\\_\_pycache\_\_" "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\\_\_init\_\_.py" "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\browser\_manager.py" "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\file\_text\_utils.py" "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\genspark\_driver.py" "C:\Users\a\Documents\PubliScript\_Refactored\ai\_interfaces\interaction\_utils.py"

/ui: "C:\Users\a\Documents\PubliScript\_Refactored\ui\book\_builder - Copia.py.bak" "C:\Users\a\Documents\PubliScript\_Refactored\ui\book\_builder.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\book\_builder.py.backup" "C:\Users\a\Documents\PubliScript\_Refactored\ui\chat\_manager.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\components.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\cooldown\_manager.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\database\_manager.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\log\_utils.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\\_\_pycache\_\_" "C:\Users\a\Documents\PubliScript\_Refactored\ui\\_\_init\_\_.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\ai\_connection.py" "C:\Users\a\Documents\PubliScript\_Refactored\ui\app\_launcher.py"

**Tutti i def di book\_builder.py**C:\Users\a\Documents\PubliScript\_Refactored>findstr /n "def " ui\book\_builder.py

28: def \_\_init\_\_(self):

78: # Prompt di analisi default

79: self.default\_analysis\_prompt = """1) Analizza la concorrenza su {amazon\_url} per la keyword {keyword} nel mercato {market}: elenca i primi 10 risultati Amazon con titolo, sottotitolo, BSR, prezzo, recensioni, formato, keyword usate nei titoli, pattern visivi delle copertine; aggiungi dati da Google Trends, query emergenti e insight dai social; concludi con una tabella di sintesi e commento su cosa domina e cosa manca; scrivi in {lingua}, titoli e keyword nella lingua del {market}; concludi con la parola FINE.

84:6) Genera 3 idee editoriali differenzianti per un nuovo libro su {keyword} nel mercato {market}: per ciascuna proposta definisci lÔÇÖangolo editoriale, lÔÇÖapproccio (pratico, teorico, visualeÔÇª), il target specifico e una USP sintetica (max 2 righe) che risponda ai bisogni emersi; scrivi in {lingua}, titoli e keyword nella lingua del {market}; concludi con la parola FINE.

90: def recupera\_ultimo\_progetto(self):

94: def ripristina\_ultima\_analisi(self):

98: def ripristina\_analisi\_da\_database(self, selected\_index, start\_from\_phase=None):

143: def resume\_analysis(self, project\_id, selected\_phases=None):

192: def process\_prompt(prompt\_text):

240: def load\_projects\_list(self):

253: def check\_existing\_analysis(self, keyword):

257: def load\_project\_details(self, selected\_index):

261: def diagnose\_and\_fix\_database(self):

265: def export\_project(self, selected\_index):

269: def update\_project\_count(self):

273: def delete\_project(self, project\_display\_name):

282: def create\_test\_project(self):

286: def search\_projects(self, keyword=""):

290: def get\_database\_stats(self):

296: def add\_log(self, message):

300: def log\_prompt\_location(self, prompt\_id, section\_number, action, details=None):

304: def connect\_callback(self):

315: def log\_get\_call(url):

359: def log\_get\_call(url):

480: def add\_url\_change\_monitor(self):

515: def \_analyze\_market\_crisp(self, book\_type, keyword, language, market, selected\_phases=None):

547: def execute\_prompt(self, prompt\_text, step\_id=None, project\_data=None):

1005: # Se la sezione ha fallito definitivamente - decisione se continuare

1006: self.add\_log(f"ÔØî Fallimento definitivo per sezione {i+1}/{len(sections)}")

1007: print(f"DEBUG: Fallimento definitivo per sezione {i+1}/{len(sections)}")

1167: def check\_unresolved\_placeholders(self, text):

1173: def get\_clean\_input\_box(self):

1209: def safe\_text\_input(self, input\_box, text):

1225: def click\_send\_button(self):

1252: def wait\_for\_stable\_response(self, max\_wait\_cycles=45, stability\_threshold=5, cycle\_wait=20):

1329: def save\_response\_to\_project(self, project\_data, prompt\_id, line, response, is\_final):

1400: def process\_combined\_response(self, response\_list, prompt\_id, project\_data):

1425: def get\_fallback\_response(self, prompt\_id, project\_data):

1451: def execute\_with\_updates(self, func, \*args, \*\*kwargs):

1479: def analyze\_market(self, book\_type, keyword, language, market, analysis\_prompt=None, use\_crisp=None):

1489: use\_crisp: Se True, usa il framework CRISP; se None, usa il valore di default

1684: analysis\_prompt = self.default\_analysis\_prompt

1752: def load\_project\_list(self):

1802: def \_analyze\_market\_legacy(self, book\_type, keyword, language, market, analysis\_prompt):

1822: def select\_all\_phases(self, analysis\_type):

1841: def deselect\_all\_phases(self, analysis\_type):

1861: def \_filter\_legacy\_prompt\_sections(self, analysis\_prompt, selected\_phases):

1872: def take\_debug\_screenshot(self, prefix):

1884: def get\_last\_response(self):

1958: def handle\_context\_limit(self):

2026: def reset\_context\_manual(self, driver):

2129: def update\_analysis\_status(self, status\_text, progress\_percentage=None):

2157: def show\_feedback(self, title, message, type="info"):

2202: def create\_focused\_context(self, original\_context\_file, max\_size=8000):

2280: def send\_essential\_context(self, driver, context\_file):

2352: def check\_for\_generation\_error(self, response):

2412: def handle\_consecutive\_errors(self, prompt\_text, max\_retries=3):

2461: def continue\_analysis(self):

2473: def \_continue\_analysis\_crisp(self):

2506: def continue\_executor(prompt\_text):

2541: def complete\_analysis(self):

3487: def \_complete\_analysis\_crisp(self):

3628: def \_complete\_analysis\_legacy(self):

3793: def create\_selection\_dialog(self, titoli\_options, indici\_options, voice\_style):

3799: titoli\_options = [{"id": 1, "titolo": "Il tuo nuovo libro", "sottotitolo": "", "display": "Titolo predefinito"}]

3801: indici\_options = [{"id": 1, "content": "INTRODUZIONE\n\nCAPITOLO 1: Fondamenti\n\nCAPITOLO 2: Metodologia\n\nCAPITOLO 3: Applicazione\n\nCONCLUSIONE", "display": "Indice predefinito"}]

3864: def update\_title\_preview(title\_display):

3870: def update\_index\_preview(index\_display):

3876: def confirm\_selection(title\_display, index\_display, voice\_style):

3911: def cancel\_selection():

3912: # Usa i valori predefiniti

3913: default\_title = titoli\_options[0]["titolo"] + " - " + titoli\_options[0]["sottotitolo"] if titoli\_options else "Il tuo nuovo libro"

3914: default\_index = indici\_options[0]["content"] if indici\_options else "INTRODUZIONE\n\nCAPITOLO 1: Fondamenti\n\nCAPITOLO 2: Metodologia\n\nCAPITOLO 3: Applicazione\n\nCONCLUSIONE"

3918: self.book\_title.update(value=default\_title)

3921: self.book\_index.update(value=default\_index)

3926: self.add\_log("ÔÜá´©Å Selezione annullata, usati valori predefiniti")

3959: def handle\_selection\_result(self, title\_display, index\_display, voice\_style):

3999: def \_generate\_book\_crisp(self, book\_title, book\_language, voice\_style, book\_index):

4229: def \_generate\_book\_legacy(self, book\_title, book\_language, voice\_style, book\_index):

4249: def split\_prompt(self, text, prompt\_id=None, section\_number=None):

4290: def send\_to\_genspark(self, text, prompt\_id=None, section\_number=None):

4320: def update\_ui\_and\_return(response, success=True, message=None):

4858: def \_set\_use\_crisp(self, value):

4869: def log\_history\_string(self):

4873: def generate\_book(self, book\_title, book\_language, voice\_style, book\_index):

4952: def \_parse\_book\_index(self, book\_index):

4980: def \_load\_chapter\_prompt(self, book\_type):

4982: # Cerca prima in un file, altrimenti usa il template predefinito

4988: # Template predefinito

5014: def \_handle\_missing\_placeholders(self, text):

5026: def \_generate\_chapter\_content(self, chapter\_title, prompt):

5058: def \_save\_chapter(self, chapter\_title, chapter\_content, book\_title):

5089: def load\_analysis\_results(self):

5220: def export\_to\_docx(self):

5311: def export\_to\_pdf(self):

5346: def export\_to\_txt(self):

5356: # 2. Ottieni la keyword corrente o usa un valore predefinito

5380: def debug\_check\_components(self):

5391: def load\_saved\_analyses\_list(self):

5407: def load\_saved\_analysis(self, keyword):

5437: def get\_current\_keyword(self):

5465: def create\_interface(self):

5558: value=self.default\_analysis\_prompt,

6137: def clear\_logs():

6218: # Nel metodo create\_interface, dopo aver definito load\_analysis\_btn:

6265: def format\_analysis\_results\_html(self, keyword, market, book\_type, language, context=None):

6414: def process\_text(self, text):

6432: def process\_list\_html(self, content, list\_type):

6491: def process\_patterns\_html(self, content, pattern\_type):

6564: def process\_table\_html(self, content):

6605:def apply\_debug\_patching(builder\_instance):

6616: def debug\_send\_to\_genspark(self, text, prompt\_id=None, section\_number=None):

C:\Users\a\Documents\PubliScript\_Refactored>

**1. Da spostare in framework/book\_generator.py:**

* \_generate\_book\_crisp (riga 3999)
* \_generate\_book\_legacy (riga 4229)
* generate\_book (riga 4873)
* \_parse\_book\_index (riga 4952)
* \_load\_chapter\_prompt (riga 4980)
* \_handle\_missing\_placeholders (riga 5014)
* \_generate\_chapter\_content (riga 5026)
* \_save\_chapter (riga 5058)

**2. Da spostare in framework/formatters.py:**

* format\_analysis\_results\_html (riga 6265)
* process\_text (riga 6414)
* process\_list\_html (riga 6432)
* process\_patterns\_html (riga 6491)
* process\_table\_html (riga 6564)

**3. Da spostare in ai\_interfaces/context\_manager.py:**

* create\_focused\_context (riga 2202)
* send\_essential\_context (riga 2280)
* handle\_context\_limit (riga 1958) - già spostato?
* reset\_context\_manual (riga 2026) - già spostato?

**4. Da spostare in ui/interface\_utils.py:**

* update\_analysis\_status (riga 2129)
* show\_feedback (riga 2157)
* debug\_check\_components (riga 5380)

**5. Funzioni per file utility generici**

* execute\_with\_updates (riga 1451) → framework/utils.py
* check\_unresolved\_placeholders (riga 1167) → framework/utils.py o già spostato?