Università degli Studi di Milano
Facoltà di scienze e tecnologie
Laurea Magistrale in Informatica

STORYGEN AI : GENERAZIONE DI NARRATIVA INTERATTIVA

Relatore: Prof. Laura Anna Ripamonti

Co-relatore: Dott. Marco Ligabue

Autore:
Federico Maglione
Numero matricola: 962587

Anno accademico 2024-2025

Indice

Capitolo 1
Introduzione e riassunto

1.1 Introduzione generale

Negli ultimi anni il settore videoludico ha conosciuto un'evoluzione significativa, non soltanto dal punto di vista tecnico e grafico, ma anche, e soprattutto, nella componente narrativa. I videogiochi contemporanei si configurano sempre più come esperienze interattive complesse, nelle quali la costruzione della trama e lo sviluppo dei personaggi assumono un ruolo centrale nel determinare l'immersione e il coinvolgimento del giocatore [1, 2].

La narrazione interattiva, infatti, è divenuta uno degli elementi distintivi della produzione videoludica moderna, trasformando il videogioco da mero intrattenimento a mezzo espressivo e culturale capace di generare emozioni, riflessioni e partecipazione attiva [2]. In questo contesto, l'applicazione dell'intelligenza artificiale alla generazione automatica di storie videoludiche rappresenta una delle frontiere più innovative della ricerca contemporanea. L'obiettivo non è più soltanto quello di creare mondi realistici o personaggi credibili, ma di generare in modo dinamico narrazioni coerenti, adattive e interattive, in grado di reagire alle scelte del giocatore e di adattarsi ai contesti ludici in cui sono inserite [3].

Il progetto qui presentato nasce con l'intento di sviluppare un sistema per la generazione automatica di storie videoludiche attraverso l'utilizzo di modelli di intelligenza artificiale generativa. Tale sistema si pone come strumento di supporto al processo creativo di *game designer* e *narrative designer*, fornendo un aiuto concreto nelle fasi di brainstorming, strutturazione della trama e definizione dei legami tra personaggi ed eventi.

Il sistema trae ispirazione dai principi e dalla struttura concettuale del progetto tesi *GHOST – A Ghost Story Writer* [4], un sistema che permetteva la generazione semi-automatica di storie interattive mediante la manipolazione di nodi narrativi. Tuttavia, rispetto a GHOST, la soluzione proposta introduce una componente di intelligenza artificiale generativa, capace di analizzare, adattare e creare contenuti narrativi dinamici sulla base di prompt strutturati, documenti di riferimento teorico e database narrativi personalizzati. Questa integrazione consente di superare i limiti di rigidità e ripetitività dei modelli precedenti, offrendo uno strumento più flessibile, creativo e adatto ai contesti produttivi moderni.

A livello metodologico, il progetto si è avvalso di diversi riferimenti teorici e strumenti di supporto alla scrittura narrativa, che ne hanno influenzato la struttura e i principi progettuali:

- ***Plotto: The Master Book of All Plots*** di William Wallace Cook [5], un classico della narrativa che propone un sistema di situazioni drammatiche, utile come base per la definizione di trame e conflitti ricorrenti. Questo approccio ha ispirato la creazione di un archivio strutturato di "basi narrative" da cui l'IA può attingere nella fase di generazione.

- ***Periodic Table of Storytelling***, uno strumento ideato dalla community di TVTropes [6], che analizza e classifica i tropi narrativi più comuni in film, serie TV, libri e videogiochi. Tale modello ha fornito la base per la rappresentazione dei nodi narrativi e per la loro organizzazione in categorie funzionali (es. personaggi, eventi, ambienti, oggetti).

- ***Slay the Dragon: Writing Great Video Games*** di Robert Denton Bryant e Keith Giglio [1], testo di riferimento nel campo del *game narrative design*, utilizzato nella definizione degli elementi narrativi fondamentali del sistema, come gli archi drammatici, le strutture a più atti e la correlazione tra gameplay e storytelling.

Grazie a questi riferimenti, il sistema sviluppato si colloca a metà strada tra la tradizionale costruzione manuale della trama e le più recenti tecniche di generazione narrativa automatica basate su intelligenza artificiale, con l'obiettivo di creare uno strumento pratico, creativo e teoricamente fondato per la progettazione di storie videoludiche.

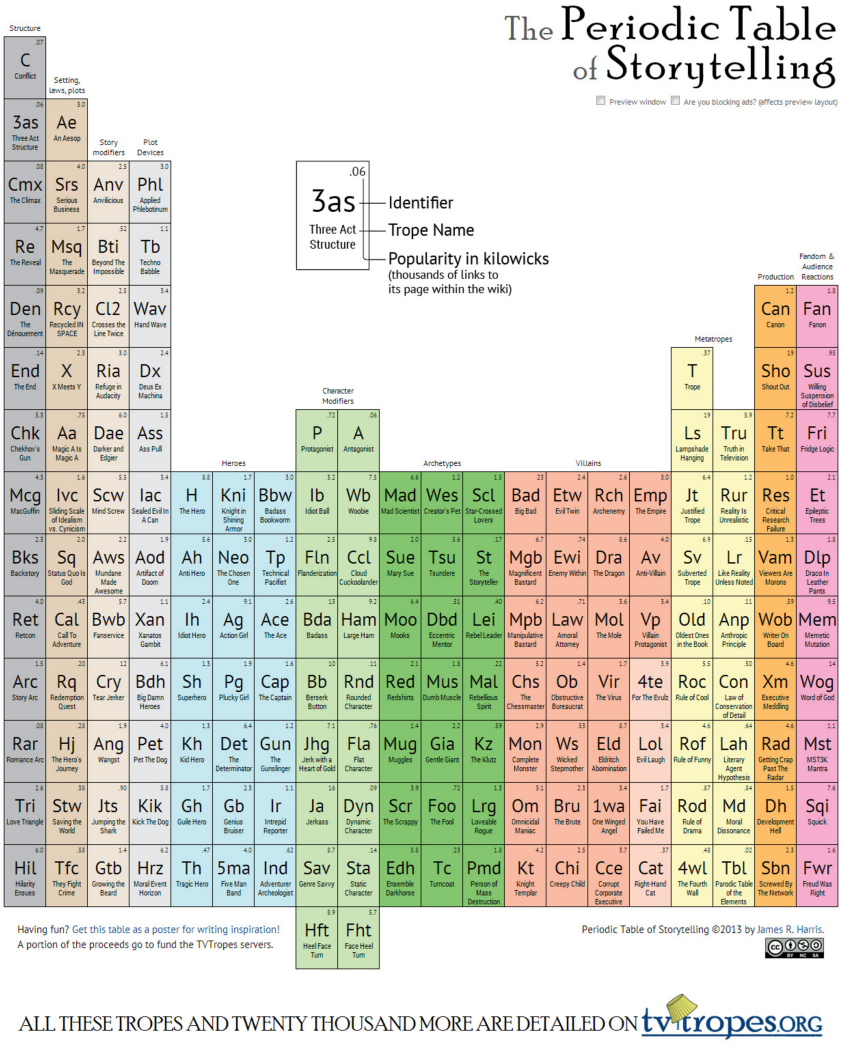

Figura 1: Periodic table of Storytelling

1.2 Stato dell'arte e motivazioni

Negli ultimi anni, lo storytelling nei videogiochi ha subito una profonda trasformazione, passando da semplice elemento accessorio a componente centrale del game design. Le produzioni contemporanee pongono crescente attenzione alla costruzione di trame complesse, personaggi psicologicamente coerenti e universi narrativi condivisi, elementi che contribuiscono in modo determinante alla qualità dell'esperienza di gioco [1, 2]. Esempi come *The Last of Us*, *Disco Elysium* o *Baldur's Gate* dimostrano come la scrittura e la progettazione narrativa possano rappresentare un fattore di distinzione e di valore culturale per il prodotto videoludico.

Parallelamente a questa evoluzione, la ricerca accademica ha iniziato a esplorare strumenti e modelli per l'automazione della generazione narrativa, con l'obiettivo di affiancare il designer nelle fasi più complesse del processo creativo. Progetti come *GHOST – A Ghost Story Writer* hanno introdotto l'idea di rappresentare la narrazione attraverso nodi logici interconnessi, ognuno dei quali descrive un elemento con una funzione narrativa specifica (personaggio, evento, luogo, oggetto, ecc.) [4]. Tale approccio ha dimostrato la possibilità di modellare una storia come rete semantica di relazioni, superando la linearità tipica della narrazione tradizionale.

Altri strumenti e riferimenti teorici hanno fornito ispirazione significativa. Tra questi, *Plotto: The Master Book of All Plots* di William Wallace Cook [5] è considerato uno dei primi tentativi sistematici di classificazione delle trame narrative. Il volume propone una struttura combinata di situazioni drammatiche e relazioni umane, concepita per supportare lo scrittore nella generazione di nuovi intrecci. Nel contesto di questo progetto, Plotto ha ispirato l'idea di un archivio di strutture narrative basilari utilizzabili come base per la generazione automatica delle storie.

Un ulteriore riferimento teorico è la *Periodic Table of Storytelling*, sviluppata dalla community TVTropes [6], che raccoglie e organizza centinaia di tropi narrativi, ossia modelli ricorrenti di situazioni, personaggi o dinamiche drammatiche impiegati nei media contemporanei. Tale struttura è stata utilizzata come base concettuale per la rappresentazione dei nodi narrativi nel sistema, consentendo di associare a ciascun nodo una funzione drammatica, un tema e un archetipo riconoscibile.

Infine, il testo *Slay the Dragon: Writing Great Video Games* di Robert Denton Bryant e Keith Giglio [1] rappresenta il principale riferimento per la definizione degli elementi narrativi videoludici. Il volume affronta la narrazione interattiva dal punto di vista dello *screenwriting* adattato al gameplay, ponendo l'accento su concetti come *player agency*, *emotional arc* e struttura del gioco. In particolare, le nozioni di sequenza narrativa, arco dell'eroe e struttura a più atti sono state riprese e adattate all'interno del sistema sviluppato per garantire coerenza con i principi del *game narrative design* moderno.

Questi contributi teorici e pratici hanno evidenziato la necessità di strumenti capaci di coniugare complessità narrativa e automazione intelligente, consentendo di generare storie non soltanto corrette dal punto di vista sintattico, ma anche

coerenti, significative e contestualmente rilevanti. L'introduzione dei modelli linguistici di grandi dimensioni (*Large Language Models*, LLM), come GPT, Gemini o Claude, ha aperto nuove prospettive in questa direzione, permettendo di combinare apprendimento semantico, ragionamento narrativo e creatività generativa.

Il progetto presentato in questa tesi nasce dunque dalla volontà di unire la rigidità strutturale del modello GHOST alla flessibilità linguistica e semantica dell'intelligenza artificiale generativa, con l'obiettivo di costruire un sistema capace di:

- generare storie videoludiche coerenti e articolate, basate su modelli narrativi consolidati;
- supportare il *game designer* nella fase di ideazione e sviluppo;
- adattarsi a differenti generi, complessità e stili narrativi.

In questa prospettiva, l'uso dell'intelligenza artificiale non sostituisce la creatività umana, ma la amplifica, fornendo un ambiente di lavoro ibrido in cui autore e macchina cooperano per la costruzione di esperienze narrative sempre più ricche, dinamiche e personalizzabili [1].

1.3 Architettura generale del sistema

L'intero sistema è stato progettato secondo un'architettura *client–server*, concepita per garantire modularità, scalabilità e una chiara separazione delle responsabilità tra le componenti [7, 8]. Il server (backend), sviluppato in Python, gestisce l'esecuzione sequenziale dei quattro step fondamentali del processo di generazione narrativa e mette a disposizione un set di servizi RESTful per l'interazione con il client [7]. Il client (frontend), realizzato in Unreal Engine, funge da interfaccia utente e consente di selezionare lo step desiderato, inviare richieste al server, visualizzare i risultati ottenuti e analizzare in modo interattivo le storie generate. L'architettura complessiva si articola dunque in due componenti principali.

Backend – Generazione e validazione della storia: Il backend costituisce la base logica del sistema e si articola in quattro fasi principali, ognuna delle quali è gestita da un modulo Python dedicato. Ogni fase corrisponde a uno step specifico del flusso narrativo e viene eseguita in modo sequenziale o su richiesta, seguendo un modello di pipeline modulare frequentemente adottato nei sistemi di generazione automatica dei contenuti.

1. **Step 1 – Generazione della struttura narrativa:**
   In questa fase, attraverso l'utilizzo di un modello di intelligenza artificiale generativa (Gemini 2.5 Flash), il sistema elabora una struttura narrativa di base, definendo nodi, connessioni e relazioni tra eventi, personaggi e temi principali. Il risultato viene salvato in formato JSON e successivamente visualizzato come grafo narrativo, traducendo la struttura logica in una rappresentazione grafica leggibile e analizzabile. Questo step costituisce il fondamento su cui si basano tutte le fasi successive, in quanto definisce l'ossatura narrativa dell'intera storia.

2. **Step 2 – Generazione dei capitoli narrativi:**
   A partire dalla struttura generata, il sistema suddivide la narrazione in capitoli coerenti, ciascuno rappresentante una sotto-sequenza narrativa autonoma ma interconnessa con le altre, in linea con i principi della segmentazione narrativa videoludica [1]. Ogni capitolo include obiettivi narrativi, spazi e ambienti di riferimento, opposizioni e conflitti, nonché adattamenti di gameplay e scelte dinamiche, fungendo da ponte tra struttura astratta e contenuto giocabile.

   In questa fase, il sistema utilizza una base informativa strutturata per estrarre elementi semantici e contestuali, garantendo coerenza con la lore e le meccaniche narrative definite.

3. **Step 3 – Generazione della storia dettagliata:**
   In questo step, il sistema redige la narrazione testuale completa di ogni capitolo, integrando descrizioni, dialoghi, dinamiche di causa–effetto e transizioni

emotive tra le scene. Si tratta della fase in cui la struttura astratta prende forma concreta, diventando una storia videoludica fruibile e coerente con i principi del *game narrative design* [2]. L'output viene prodotto in formato JSON e può essere direttamente analizzato o modificato dall'utente.

4. **Step 4 – Validazione e correzione automatica:**
   L'ultima fase è dedicata alla verifica e convalida della coerenza narrativa. La storia generata viene analizzata da due moduli distinti:

   - **Label Validator**, che controlla la coerenza semantica e terminologica di personaggi, eventi, temi e capitoli;
   - **Flow Validator**, che verifica la continuità narrativa tra i capitoli, individuando eventuali discontinuità o contraddizioni logiche.

Insieme, questi due moduli garantiscono che la storia finale rispetti i principi di coerenza, progressione e plausibilità narrativa, riducendo errori tipici dei sistemi generativi non guidati. Il backend è orchestrato da un modulo principale che gestisce i job in esecuzione, la comunicazione tra gli step e la serializzazione dei risultati. L'intero processo è stato concepito per essere scalabile e asincrono, consentendo l'esecuzione di più richieste contemporaneamente e la gestione di task complessi in background [8].

Frontend – Interfaccia utente e interazione con il sistema: Il frontend, sviluppato in Unreal Engine, rappresenta la componente visiva e interattiva del sistema. Attraverso un widget dedicato, l'utente può selezionare lo step di generazione o validazione desiderato, visualizzare in tempo reale i risultati prodotti dal server, esplorare graficamente la struttura narrativa in forma di grafo, leggere e analizzare la storia generata per ciascun capitolo e generare una sintesi narrativa automatica della storia complessiva. L'interfaccia è stata progettata con l'obiettivo di mantenere un flusso di lavoro fluido e intuitivo, riducendo al minimo la complessità tecnica per l'utente finale, in linea con le best practice di progettazione degli strumenti di authoring interattivo.

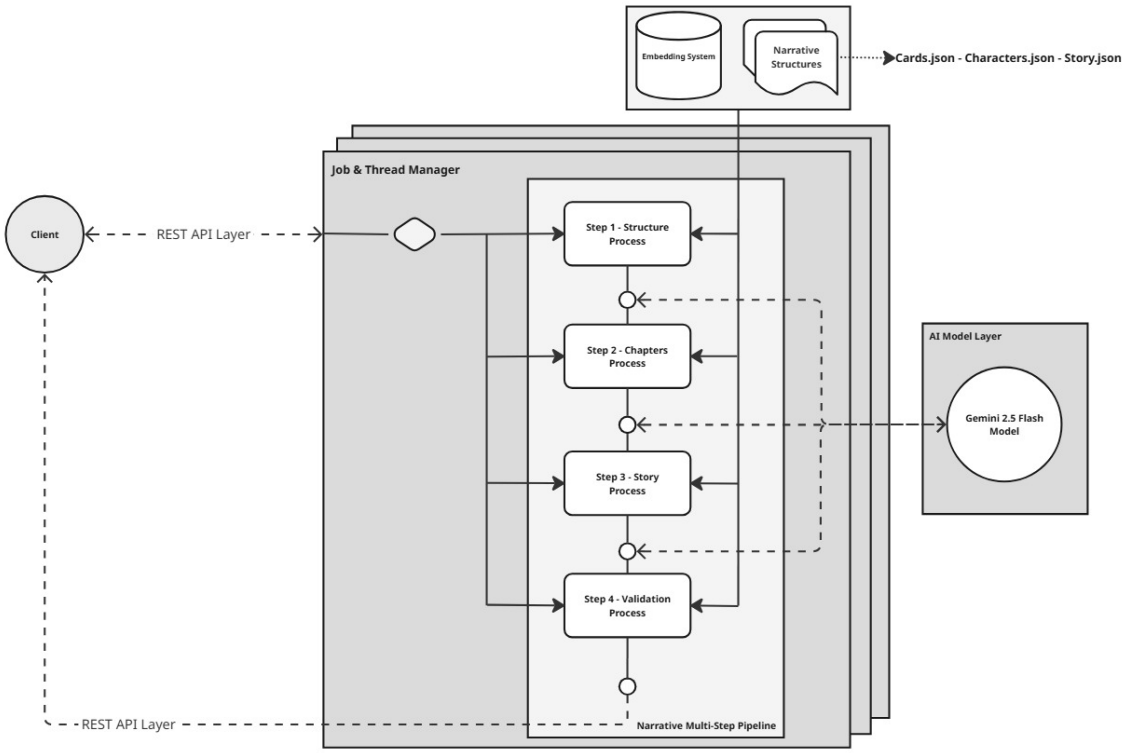

Figura 2: Overview architettura

1.4 Obiettivi del lavoro

L'obiettivo principale di questo lavoro è la progettazione e realizzazione di un sistema modulare e scalabile per la generazione automatica di storie videoludiche, capace di combinare la struttura narrativa tradizionale con le potenzialità dell'intelligenza artificiale generativa. Il progetto mira a creare uno strumento di supporto creativo per il *game designer* e il *narrative designer*, in grado di automatizzare parte del processo di ideazione e sviluppo della trama, mantenendo al tempo stesso un elevato livello di coerenza narrativa, adattabilità e controllo autoriale.

A differenza dei sistemi di generazione narrativa puramente procedurali, basati su regole statiche o schemi predefiniti, il sistema proposto adotta un approccio ibrido, in cui:

- la struttura narrativa di base è definita secondo modelli logici derivati da GHOST e dalla *Periodic Table of Storytelling* [4, 6];
- la creazione dei contenuti testuali è affidata a un modello di intelligenza artificiale generativa, in grado di interpretare e ampliare gli elementi strutturali con creatività, coerenza semantica e sensibilità contestuale [9].

In termini pratici, gli obiettivi specifici del lavoro possono essere suddivisi in tre macro aree:

1.4.1 Obiettivi di ricerca e sperimentazione

- Analizzare e integrare i principali modelli di generazione narrativa automatica, con particolare attenzione all'equilibrio tra controllo strutturale e libertà creativa.
- Verificare l'efficacia dell'uso di modelli linguistici di grandi dimensioni (LLM) come strumento di supporto alla scrittura videoludica, valutando la qualità, la coerenza e la varietà delle storie prodotte.
- Sperimentare la combinazione di framework narrativi classici (come il Viaggio dell'Eroe, la struttura in tre atti e i tropi narrativi) con approcci adattivi basati su AI, in linea con i principi del narrative design interattivo [10, 1].
- Definire un flusso di lavoro standardizzato per la generazione, validazione e correzione automatica di storie videoludiche.

1.4.2 Obiettivi implementativi

- Progettare un'architettura a moduli in grado di suddividere il processo narrativo in quattro step distinti, seguendo principi di separazione delle responsabilità e scalabilità [8].

- Implementare un backend Python con servizi RESTful per l'interazione tra le componenti e la gestione asincrona dei job di generazione [7].
- Integrare modelli linguistici e sistemi di embedding semantico per garantire una comprensione contestuale dei dati narrativi e dei riferimenti teorici.
- Realizzare un frontend in Unreal Engine che consenta la selezione degli step, la visualizzazione dei risultati, l'analisi dei grafi narrativi e la sintesi finale della storia.
- Implementare un sistema di validazione automatica in grado di identificare incoerenze semantiche, logiche o cronologiche nella storia generata.

1.4.3 Obiettivi qualitativi e progettuali

- Garantire la coerenza narrativa interna tra i vari capitoli e atti, assicurando la continuità di personaggi, temi e relazioni.
- Mantenere un elevato grado di adattabilità del sistema, permettendo la generazione di storie appartenenti a generi differenti con un'unica base strutturale.
- Offrire un'interfaccia intuitiva che favorisca l'interazione tra utente e sistema, riducendo le barriere tecniche per il designer non esperto.
- Creare una piattaforma estendibile, capace di evolversi nel tempo con l'aggiunta di nuovi moduli e modelli di intelligenza artificiale.

1.4.4 Sintesi finale degli obiettivi

L'obiettivo finale del progetto è dunque quello di congiungere la logica narrativa classica con la generazione automatica, dimostrando che l'intelligenza artificiale può essere usata per espandere la capacità ideativa umana. Il sistema proposto non intende sostituire la figura del *narrative designer*, ma fornire uno strumento di supporto intelligente, capace di:

- generare strutture narrative coerenti e flessibili;
- adattare i contenuti in base alle esigenze del progetto;
- ridurre i tempi di prototipazione narrativa;
- migliorare la qualità e la varietà delle esperienze interattive prodotte.

In tal modo, il lavoro si inserisce in un più ampio filone di ricerca volto a esplorare le intersezioni tra intelligenza artificiale, design narrativo e creatività umana, ponendo le basi per una nuova generazione di strumenti a supporto della progettazione narrativa nei videogiochi.

1.5 Risultati e feedback

La fase di sperimentazione del sistema ha evidenziato risultati promettenti sia dal punto di vista tecnico sia da quello narrativo. I test condotti su scenari differenti hanno evidenziato come l'approccio multistep adottato struttura → capitoli → storia → validazione consenta di mantenere un buon controllo sulla progressione narrativa, riducendo la propagazione di errori e migliorando la coerenza complessiva della storia. I risultati mostrano che:

- le strutture narrative generate risultano solide e ben formate;
- la suddivisione in capitoli produce sequenze coerenti e funzionali al ritmo narrativo;
- la narrazione completa presenta una qualità generalmente buona, risultando comprensibile e coerente;
- i moduli di validazione migliorano sensibilmente la continuità narrativa identificando molte delle incoerenze interne.

I test condotti con studenti, sviluppatori e appassionati di narrativa interattiva hanno messo in luce sia i punti di forza del sistema sia alcune aree di miglioramento. Tra le principali osservazioni:

Chiarezza del flusso e modularità. La suddivisione in step è stata apprezzata perché consente di comprendere facilmente la logica del processo e di intervenire su ogni fase senza compromettere le altre. Gli utenti hanno sottolineato come il sistema favorisca un approccio "guidato" alla generazione narrativa, ideale per la fase di brainstorming o di pre-produzione di un progetto videoludico.

Coerenza narrativa generale. Le storie generate presentano una buona coerenza logica tra i capitoli, specialmente quando la struttura di partenza è sufficientemente dettagliata. Gli utenti hanno evidenziato che, nella maggior parte dei casi, la storia generata risulta ben definita, coerente e facilmente comprensibile nel suo insieme. Tuttavia, è emersa la necessità di migliorare la gestione delle dipendenze tra eventi e la diversificazione dei finali, che tendono talvolta a seguire pattern ricorrenti.

Dettaglio degli elementi narrativi. Sono emerse situazioni in cui alcuni elementi risultano poco caratterizzati o non descritti a sufficienza, soprattutto nei capitoli intermedi o nelle sezioni meno centrali della trama.

Coerenze locali tra capitoli. Sebbene la struttura narrativa complessiva sia solida, la continuità tra capitoli può talvolta risultare poco chiara o non adeguatamente definita, generando piccole discontinuità o salti logici.

Incoerenze nella descrizione di alcuni elementi. In casi più complessi, alcuni elementi vengono descritti in modo incoerente rispetto alla trama globale definita, principalmente a causa di ambiguità nei prompt o di interpretazioni divergenti da parte del modello.

Limitazioni tecniche dei modelli AI attuali. Parte di queste criticità è imputabile alle limitazioni tecniche dei modelli linguistici utilizzati, che, pur essendo avanzati, possono faticare nella gestione di lunghe catene logiche, dipendenze narrative complesse o descrizioni molto dettagliate. È prevedibile che tali problemi possano essere significativamente ridotti adottando modelli più performanti, incluse versioni evolute dei LLM o modelli a pagamento con maggiore capacità contestuale e potenza inferenziale, capaci di mantenere meglio la coerenza narrativa e la qualità del dettaglio.

Supporto creativo complessivo. Nonostante i limiti locali, gli utenti concordano nel ritenere che la storia generata fornisce un supporto completo e concreto all'utilizzatore, risultando altamente utile per attività di brainstorming, prototipazione rapida e definizione preliminare della narrativa.

Interfaccia e visualizzazione. L'integrazione con Unreal Engine è stata particolarmente apprezzata, soprattutto per la possibilità di visualizzare graficamente il grafo narrativo e analizzare la progressione della storia in modo immediato.

1.5.1 Conclusione

I risultati complessivi confermano la validità dell'approccio ibrido adottato, che unisce la struttura narrativa tradizionale alla generazione adattiva degli LLM. Pur con alcune limitazioni tecniche, il sistema rappresenta uno strumento efficace e versatile, capace di supportare il *game designer* nella progettazione narrativa e di aprire la strada a nuove modalità di creazione di contenuti interattivi. L'evoluzione futura dei modelli linguistici, unitamente all'espansione del sistema, promette di rendere la generazione narrativa automatica uno strumento sempre più affidabile, coerente e creativo.

Capitolo 2
Contesto e stato dell'arte

2.1 Lo storytelling: definizione e ruolo nella comunicazione umana

Lo storytelling può essere descritto come l'arte di costruire e trasmettere storie e rappresenta una delle forme più antiche e fondamentali di comunicazione umana. Fin dalle prime civiltà, il racconto ha costituito uno strumento essenziale per tramandare conoscenze, valori, paure e speranze, assumendo funzioni educative, rituali e culturali. Dai miti delle tragedie greche ai romanzi moderni, la narrazione ha accompagnato l'evoluzione dell'umanità, adattandosi ai media e alle tecnologie di ogni epoca [2].

La sua funzione principale è quella di creare significato, organizzando l'esperienza umana in forme comprensibili e condivisibili. Raccontare una storia significa dare ordine al caos degli eventi, stabilire relazioni di causa-effetto, attribuire ruoli e motivazioni ai personaggi e trasformare l'esperienza individuale in esperienza collettiva [2].

2.1.1 Aristotele e le origini della teoria della narrativa

Le prime teorie sulla costruzione delle storie risalgono ad Aristotele, che nella *Poetica* definiva la struttura degli eventi come l'elemento centrale del dramma, ritenendola più importante dei personaggi e del linguaggio stesso [11]. Secondo Aristotele, una narrazione efficace deve possedere:

- **Unità di azione**: gli eventi devono essere collegati da rapporti causali.
- **Inizio, parte centrale e conclusione**: una struttura che consenta alla vicenda di evolversi secondo una logica interna.
- **Peripezia e riconoscimento**: momenti di svolta e rivelazione che trasformano la comprensione della storia.
- **Catarsi**: una funzione emotiva che coinvolga lo spettatore attraverso empatia, paura e compassione.

Il modello aristotelico rappresenta tuttora un pilastro della narrativa e influenza profondamente anche le forme moderne dello storytelling, inclusa la narrazione videoludica [11].

2.1.2 Da Campbell a Propp: archetipi e strutture ricorrenti

Nei secoli successivi, numerosi studiosi hanno tentato di identificare strutture universali e ricorrenti nelle storie. Joseph Campbell, con *The Hero with a Thousand Faces*, ha proposto il modello del *"Viaggio dell'Eroe"* [10], una struttura narrativa ciclica basata su 17 tappe che descrivono il percorso archetipico del protagonista verso la trasformazione.

Parallelamente, Vladimir Propp ha condotto un'analisi morfologica delle fiabe popolari russe, identificando 31 funzioni narrative ricorrenti e 7 archetipi (eroe, antagonista, aiutante, mandante, donatore, ecc.) [12]. La sua analisi mostra come le storie possano essere scomposte in unità fondamentali che si combinano secondo regole precise, anticipando concetti oggi centrali nello storytelling.

2.2 La narrazione videoludica nella letteratura scientifica

La narrazione nei videogiochi è diventata, nel corso degli ultimi vent'anni, uno dei temi centrali della ricerca accademica. La natura interattiva delle narrazioni ha posto la necessità di ripensare ai modelli narrativi tradizionali, portando alla nascita di una vasta letteratura dedicata allo studio delle forme narrative emergenti nei giochi digitali.

Uno dei primi contributi fondamentali è quello di Janet Murray, che nel suo testo *Hamlet on the Holodeck* (1997) descrive il videogioco come un ambiente narrativo potenzialmente infinito, capace di coniugare immersione e trasformazione [2]. Murray introduce il concetto di *proceduralità narrativa*, sottolineando come la capacità di generare risposte e scenari in tempo reale offra nuove possibilità espressive non presenti nei media lineari [2].

Accanto a Murray, Henry Jenkins ha proposto il concetto di *narrative architecture*, sostenendo che i videogiochi non devono essere intesi come narrazioni lineari, ma come spazi narrativi che il giocatore esplora costruendo la propria esperienza [13]. Per Jenkins, il *level design* non è semplicemente una questione tecnica: esso costituisce la vera architettura del racconto, una forma di *environmental storytelling* che comunica informazioni attraverso ambientazioni, oggetti ed eventi ambientali [13].

La ricerca contemporanea riconosce che la narrazione videoludica non è una semplice trasposizione del linguaggio filmico o letterario, ma un sistema multistrato, in cui convivono molteplici tipologie di narrazione. Autori come Gonzalo Frasca hanno distinto tra *ludology* e *narratology*, sostenendo che i videogiochi vadano analizzati principalmente come sistemi di regole che generano significato attraverso il gioco stesso [14]. Tuttavia, i modelli più recenti integrano entrambe le prospettive, riconoscendo che la narrazione videoludica nasce dal dialogo tra struttura ludica e struttura narrativa.

La complessità della narrazione nei videogiochi è stata ulteriormente approfondita in studi di design narrativo, come *Slay the Dragon: Writing Great Video Games* [1], che distingue tra *gameplay-driven storytelling* e *story-driven gameplay*. Il primo privilegia la meccanica come generatore di significato; il secondo costruisce il gameplay attorno a una struttura narrativa.

Questa ricca tradizione teorica fornisce un quadro solido per comprendere la natura peculiare dello storytelling interattivo e costituisce il fondamento concettuale su cui si basano i metodi e gli strumenti utilizzati nel progetto descritto in questa tesi.

2.3 Evoluzione dello storytelling nei videogiochi

Lo storytelling nei videogiochi ha attraversato una trasformazione radicale negli ultimi quarant'anni, passando da elemento accessorio a componente fondamentale dell'esperienza videoludica. Questa evoluzione non è stata lineare, ma ha seguito un percorso in cui innovazioni tecnologiche, cambiamenti culturali e nuove forme di design che hanno progressivamente ampliato la complessità narrativa dei giochi digitali.

2.3.1 Le origini: narrazione minimale (anni '70–'80)

Nei primi videogiochi arcade e nei sistemi domestici di fine anni '70, la narrazione aveva un ruolo quasi simbolico. Titoli come *Space Invaders* (1978) o *Pac-Man* (1980) si limitavano a una cornice narrativa appena accennata, spesso descritta unicamente nel manuale di istruzioni. La priorità non era raccontare una storia, ma proporre una sfida ludica immediata, basata su punteggi e abilità.

Con l'avvento del computer come mezzo creativo, iniziarono ad emergere i primi tentativi di narrazione strutturata, come nelle *text adventure*: *Colossal Cave Adventure* (1976) e *Zork* (1980) introdussero un modello di storytelling testuale ricco di descrizioni, enigmi e progressione narrativa. Pur nella loro semplicità tecnica, questi giochi posero le basi della narrazione interattiva moderna.

Figura 3: Pac-Man Year 1980

2.3.2 Gli anni '90: l'ingresso della narrativa cinematografica

L'avanzamento delle capacità grafiche con sistemi a 16-bit e, successivamente, l'introduzione del 3D cambiarono radicalmente l'approccio alla narrazione. In questo periodo nacquero titoli che cercavano di imitare la regia, i ritmi e il linguaggio del cinema: *Final Fantasy VI* (1994), *Chrono Trigger* (1995), *Metal Gear Solid* (1998), *Half-Life* (1998) e *Baldur's Gate* (1998) introdussero personaggi complessi, sequenze animate, dialoghi articolati e trame emotivamente dense.

La narrazione, da semplice cornice, iniziò a diventare parte integrante dell'identità del gioco.

Figura 4: Final Fantasy VI Year 1994

2.3.3 La maturità narrativa dei 2000: interattività, scelte e moralità

Con l'ingresso nel nuovo millennio, la narrazione videoludica compì un ulteriore salto verso forme più articolate, interattive e ramificate. È in questo periodo che si afferma il concetto di *player agency*, ossia la capacità del giocatore di modificare la storia attraverso scelte morali o strategiche.

BioWare, con *Star Wars: Knights of the Old Republic* (2003), *Mass Effect* (2007) e *Dragon Age* (2009), introdusse sistemi conversazionali a scelta multipla, gestione delle relazioni e finali alternativi. Parallelamente, *The Elder Scrolls* e *Fallout* di Bethesda sperimentarono narrazioni emergenti basate su mondi aperti, dove le storie non venivano solo raccontate, ma scoperte.

La narrazione smise dunque di essere per lo più lineare, ma divenne ramificata, modulare e personalizzabile.

Figura 5: Dragon Age Year 2009

2.3.4 2010–2020: la fusione tra gameplay e narrazione

Il decennio successivo vide l'affermarsi definitivo del videogioco come mezzo narrativo maturo. Titoli come *The Last of Us* (2013), *The Witcher 3* (2015), *Red Dead Redemption 2* (2018) e *God of War* (2018) introdussero un modello di scrittura altamente cinematografico ma profondamente integrato con il gameplay.

Nel frattempo, il settore indie esplorò forme sperimentali di narrazione, spesso più intime e concettuali: *Journey* (2012), *Inside* (2016), *Celeste* (2018) e *Disco Elysium* (2019) mostrarono come la narrazione potesse emergere dal movimento, dall'ambiente o dalle meccaniche stesse, senza bisogno di cutscene tradizionali.

Figura 6: God of War Year 2018

2.3.5 Oggi: mondi narrativi complessi e sistemi intelligenti

L'odierna generazione di videogiochi presenta strutture narrative sempre più intricate, animate da mondi aperti, IA comportamentali e sistemi dinamici. Contemporaneamente, l'industria è chiamata a confrontarsi con la crescente complessità dei progetti: mantenere coerenza narrativa per decine e decine di ore di gioco, con centinaia di personaggi e linee di dialogo, è diventato un compito monumentale.

È in questo contesto che si fa strada l'idea di utilizzare strumenti basati su intelligenza artificiale per supportare *narrative designer*, scrittori e team di sviluppo nelle fasi di prototipazione, worldbuilding e controllo della coerenza.

La generazione di storie, un tempo dominio esclusivo dell'autore umano, diventa oggi un'area sperimentale in cui strutture narrative formali e modelli linguistici generativi possono collaborare per creare nuove forme di narrazione interattiva.

Figura 7: ARC Raiders Year 2025

Capitolo 3

Metodologie e tecnologie utilizzate

Nell'industria videoludica moderna, la creazione della storia segue processi strutturati. Gli studi *tripla A* adottano un modello di sviluppo che combina molteplici passaggi per realizzare videogiochi di successo [15]. La narrazione viene costruita iterativamente, attraverso prototipazione e playtesting [15]. Nonostante ciò, la complessità crescente delle produzioni moderne rende difficile mantenere coerenza e varietà in storie lunghe decide e decine di ore [1]. Per questo motivo, molte aziende stanno valutando l'introduzione di strumenti intelligenti per assistere gli autori nelle fasi preliminari di brainstorming, worldbuilding e gestione delle dipendenze narrative [1].

3.1 Metodologie attuali per generare storie videoludiche

La creazione di storie videoludiche nei giochi *tripla A* è un processo complesso che coinvolge scrittori, *narrative designer*, *quest designer*, *level designer*, *game director* e team di ricerca [1]. A differenza delle narrazioni lineari presenti in cinema o letteratura, la storia nei videogiochi deve integrarsi con sistemi di gioco, scelte del giocatore, ritmi di gameplay e vincoli tecnologici [1].

3.1.1 Concept narrativo e definizione della visione

Il processo di ideazione di un videogioco a forte componente narrativa inizia con la definizione di una visione condivisa del progetto. In questa fase preliminare vengono individuati i temi centrali, il tono emotivo, l'esperienza desiderata per il giocatore e la cosiddetta *Player Fantasy*, intesa come l'insieme di aspettative, ruoli e azioni che definiscono l'esperienza ludica proposta al giocatore [15].

Questi elementi costituiscono una base concettuale che orienta le successive scelte narrative e di game design, fungendo da riferimento costante durante l'intero ciclo di sviluppo [15].

3.1.2 La writers' room e la costruzione della macro-struttura narrativa

Una volta definita la visione generale, il lavoro si sviluppa spesso all'interno di una *writers' room* [1]. In questa fase il team narrativo discute e formalizza l'identità del progetto, definendo:

- l'ambientazione generale;
- gli archi narrativi principali e secondari;
- i personaggi chiave e le loro motivazioni;
- la struttura globale della storia (ad esempio tre atti, atti multipli).

Questa fase è prevalentemente concettuale e ha l'obiettivo di costruire un *framework* narrativo condiviso, riducendo il rischio di incoerenze tra le diverse componenti del team di sviluppo [1].

3.1.3 Il Narrative Design Document

La visione narrativa viene successivamente formalizzata nel *Narrative Design Document* (NDD), un insieme di documenti che rappresenta il riferimento narrativo principale del progetto [1]. L'NDD descrive:

1. la struttura degli atti e degli snodi narrativi principali;
2. i personaggi, i loro archetipi, background e relazioni;
3. gli eventi chiave e i principali *turning points*;
4. i temi portanti e i conflitti centrali;
5. le modalità di integrazione tra gameplay e narrazione.

Nei progetti contemporanei l'NDD non è un documento statico, ma viene aggiornato iterativamente e mantenuto in stretta relazione con il *game design*, il *level design* e le missioni secondarie, garantendo coerenza tra narrazione ed esperienza ludica [1].

3.1.4 Strutture ramificate e gestione della complessità narrativa

Molti videogiochi narrativi moderni adottano strutture ramificate che prevedono scelte multiple, percorsi alternativi e finali differenziati [1]. La progettazione di tali sistemi richiede l'uso di strumenti dedicati alla gestione dei grafi narrativi e delle condizioni logiche che regolano l'evoluzione della storia [1].

Tra gli strumenti più utilizzati si possono citare piattaforme come *Articy Draft* o editor narrativi proprietari, che consentono di collegare dialoghi, eventi, variabili di stato e conseguenze narrative [16].

La gestione della complessità narrativa deve tenere conto non solo delle scelte del giocatore, ma anche della coerenza temporale, della psicologia dei personaggi, della continuità delle relazioni e dei vincoli produttivi legati agli asset di gioco [1].

3.1.5 Iterazione, playtesting e riscrittura

Nel contesto videoludico la narrazione è fortemente legata al processo di iterazione. Attraverso il playtesting viene valutata l'efficacia della storia in relazione al gameplay, verificando il ritmo narrativo, la significatività delle scelte offerte al giocatore e la coerenza complessiva del mondo di gioco [15].

Questa fase comporta frequenti riscritture, modifiche alle missioni, aggiustamenti degli archi dei personaggi e una continua mediazione tra esigenze narrative e vincoli ludici [1].

3.1.6 Scalabilità narrativa e complessità produttiva

Nei videogiochi di grandi dimensioni la narrazione deve essere progettata per essere scalabile [1]. Titoli contemporanei presentano una quantità elevata di contenuti narrativi, personaggi e linee di dialogo, rendendo la coerenza interna una delle principali sfide produttive [1].

Per affrontare tale complessità sono necessari strumenti di gestione avanzati, pipeline iterative, team interdisciplinari e processi di revisione continui [15]. In questo contesto si colloca il crescente interesse verso metodologie automatizzate e sistemi di supporto basati su intelligenza artificiale, concepiti come strumenti di assistenza al lavoro di scrittori e designer.

3.2 La nuova frontiera: l'intelligenza artificiale nello storytelling videoludico

Negli ultimi anni, l'intelligenza artificiale ha introdotto una trasformazione radicale nel modo in cui vengono progettati, generati e gestiti i contenuti narrativi all'interno dei videogiochi. Se in passato la narrazione videoludica dipendeva interamente dal lavoro umano, spesso accompagnato da un enorme investimento di tempo, risorse e competenze specialistiche, l'avvento dei modelli linguistici di grandi dimensioni (*Large Language Models*, LLM) ha aperto nuove possibilità per il supporto alla fase di design.

L'AI non si limita a generare testo: essa è in grado di interpretare contesto, strutture narrative, ruoli dei personaggi, relazioni, tropi narrativi ricorrenti e dipendenze logiche, grazie a capacità di ragionamento sempre più avanzate. Questa nuova tecnologia consente di progettare strumenti capaci di amplificare la creatività del *game designer*, accelerare la prototipazione e ridurre la complessità gestionale tipica dei grandi progetti *tripla A*.

3.2.1 AI come strumento creativo per autori e *game designer*

Quindi una delle applicazioni più utili di tali modelli riguarda il supporto alla narrazione. L'AI non sostituisce l'autore umano, ma lo affianca in tre compiti fondamentali:

1. **Brainstorming narrativo:**
   L'AI può proporre idee, varianti di trama, *twist* alternativi e nuovi archetipi di personaggio.

2. **Prototipazione rapida della storia:**
   Un sistema AI può generare rapidamente bozze di capitoli, descrizioni di ambienti, dialoghi e *quest*.

3. **Controllo della coerenza interna:**
   I modelli possono essere utilizzati per verificare incongruenze logiche, errori di continuità o problemi legati alla caratterizzazione dei personaggi.

Questo approccio permette agli sviluppatori di concentrarsi sugli elementi più creativi del processo, delegando all'AI compiti ripetitivi, di supporto o preparatori.

3.2.2 Strumenti ibridi: combinare strutture formali e AI generativa

Il sistema descritto in questa tesi utilizza un approccio ibrido con una struttura narrativa formale, arricchita da un motore di generazione testuale *AI-driven* e sostenuta da un sistema di validazione logico-semantico.

Questo approccio consente di superare il limite principale dei LLM, ovvero la difficoltà nel mantenere coerenza su lunghi archi narrativi, offrendo una narrativa organizzata e controllata, in cui l'AI agisce all'interno di vincoli formalizzati.

3.3 Introduzione al sistema sviluppato

Per raggiungere tali obiettivi, il sistema è stato progettato come applicazione *client–server* [7]. Il backend, sviluppato in Python [17], gestisce in maniera completa l'esecuzione dei quattro step principali della pipeline narrativa:

- generazione della struttura;
- creazione dei capitoli;
- produzione della storia dettagliata;
- validazione automatica;
- generazione sintesi storia.

Il frontend, realizzato in Unreal Engine [18], consente all'utente di interagire con ciascuna di queste fasi tramite un'interfaccia intuitiva, permettendo di avviare i processi, visualizzare gli output e monitorare lo stato delle elaborazioni.

L'architettura complessiva si fonda su una pipeline suddivisa in moduli indipendenti, caratteristica che garantisce scalabilità, rendendo possibile l'aggiunta di nuovi step o la modifica di quelli esistenti e al tempo stesso modularità, poiché ogni fase produce output formattati e riutilizzabili. Questa suddivisione consente inoltre di controllare progressivamente la complessità narrativa, riducendo gli errori tipici dei modelli generativi e migliorando la trasparenza e la verificabilità del processo. Un ruolo centrale nel funzionamento del sistema è dato dai meccanismi di *embedding*, grazie ai quali l'AI non opera esclusivamente sui prompt, ma può integrare informazioni provenienti da documenti strutturati, oltre che da testi di riferimento. Ciò consente al modello di lavorare su basi narrative coerenti e controllate, migliorandone la capacità di produrre contenuti rilevanti e semanticamente accurati.

Per garantire un'esecuzione fluida e non bloccante, il backend adotta inoltre un sistema a thread che permette di gestire in maniera efficiente operazioni particolarmente intensive, mantenendo allo stesso tempo un flusso asincrono delle richieste [17].

3.4 Architettura generale del sistema

L'architettura del sistema è stata progettata seguendo un approccio orientato alla modularità e alla separazione delle responsabilità, al fine di integrare modelli AI, risorse narrative strutturate e un'interfaccia interattiva senza generare dipendenze rigide tra le componenti. Il sistema adotta un architettura *client–server* [7], scelta che permette di isolare la logica computazionale dalla gestione dell'interazione con l'utente e di distribuire il carico elaborativo in maniera controllata.

Il backend rappresenta il nucleo logico del sistema e opera come motore della pipeline narrativa. Tuttavia, a livello architetturale non viene percepito come un insieme di step isolati, ma come un unico *workflow*, basato su elaborazioni successive di dati strutturati. Ogni fase della pipeline, dalla generazione della struttura fino alla validazione, applica regole differenti ma opera secondo lo stesso principio architetturale: riceve un input formalizzato, lo elabora tramite modelli AI e restituisce un output verificabile. Questo pattern consente di mantenere una separazione chiara tra contenuto narrativo, logica di elaborazione e controllo della coerenza. La comunicazione tra frontend e backend avviene attraverso servizi REST, che facilitano l'interoperabilità e consentono di mantenere un'interfaccia stabile anche qualora il sistema venisse esteso con nuove funzionalità, nuovi step o differenti modelli AI.

Un elemento fondamentale dell'architettura è rappresentato dall'uso di *embedding* e risorse narrative strutturate come base dell'intero sistema. A differenza di un approccio puramente generativo, l'architettura qui adottata prevede l'integrazione di file JSON, documenti di riferimento e politiche narrative in modo da fornire ai modelli un contesto coerente. In questo modo la generazione non è affidata esclusivamente al modello, ma avviene all'interno di un ambiente controllato. Per assicurare reattività e continuità del servizio, il backend utilizza un sistema multithread, che permette di eseguire operazioni di generazione e validazione in modo asincrono [17]. Con questo approccio la pipeline è progettata per essere composta da processi indipendenti che possono essere monitorati e completati senza interferire con gli altri. Il frontend realizzato in Unreal Engine interagisce con il sistema attraverso una serie di richieste e svolge il ruolo di mediatore tra l'utente e il *workflow* interno. Dal punto di vista architetturale, esso non contiene logica narrativa, ma esclusivamente la gestione dell'interazione, della visualizzazione e della comunicazione con il server. Questa separazione consente al frontend di rimanere leggero, sostituibile e facilmente integrabile in pipeline di sviluppo videoludiche più ampie.

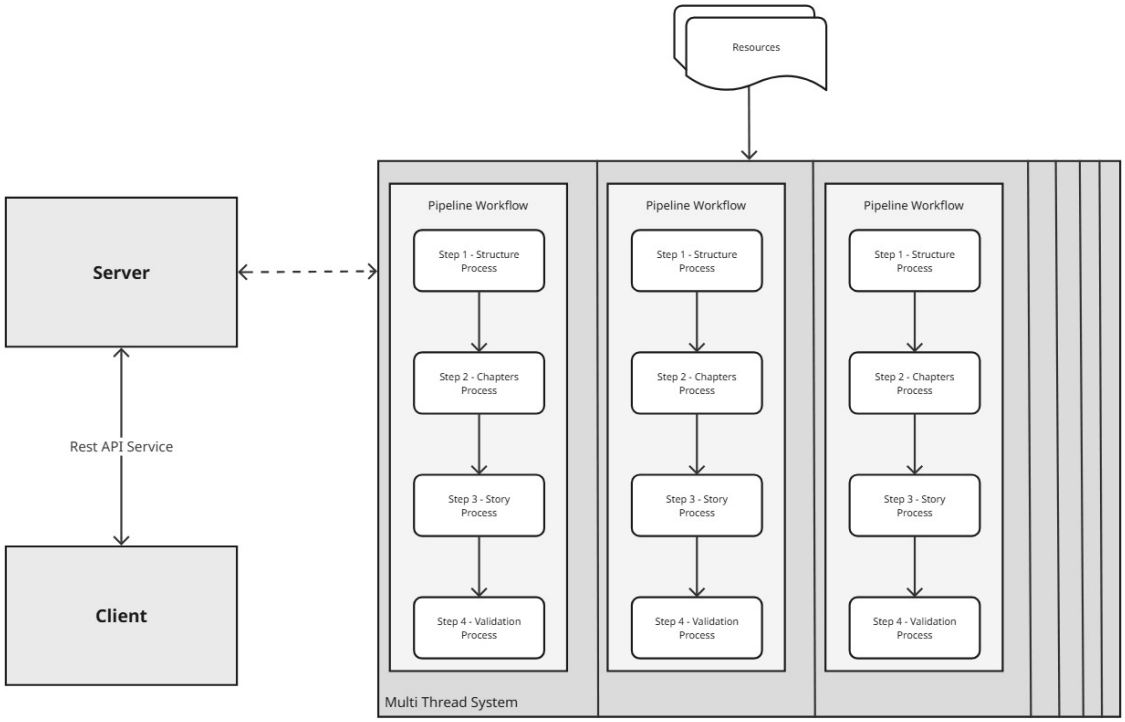

Figura 8: Overview architettura / workflow sistema

3.5 Backend del sistema: tecnologie e metodologia di funzionamento

3.5.1 Tecnologie utilizzate

Il backend è sviluppato interamente in Python [17], linguaggio scelto per la sua versatilità, la ricchezza dell'ecosistema dedicato al *machine learning* e la semplicità di integrazione con sistemi esterni tramite API. Python inoltre rende possibile combinare in modo naturale l'uso di modelli AI, la manipolazione di dati strutturati, la creazione di servizi REST e la gestione di processi concorrenti. Il sistema fa uso di un modello di intelligenza artificiale di ultima generazione (*Gemini 2.5 Flash*) [19], in grado di analizzare prompt complessi, produrre strutture narrative articolate e adattare la generazione al contesto fornito. La scelta di un modello leggero ma performante risponde alla necessità di garantire tempi di risposta ragionevoli, mantenendo al contempo un livello qualitativo sufficiente per una pipeline narrativa *multi-step*.

Accanto ai modelli AI, il backend utilizza librerie per la comunicazione HTTP, la gestione del *threading* e la manipolazione dei dati JSON, fondamentali per strutturare i contenuti narrativi, scambiare informazioni con il frontend e gestire la pipeline in modo fluido.

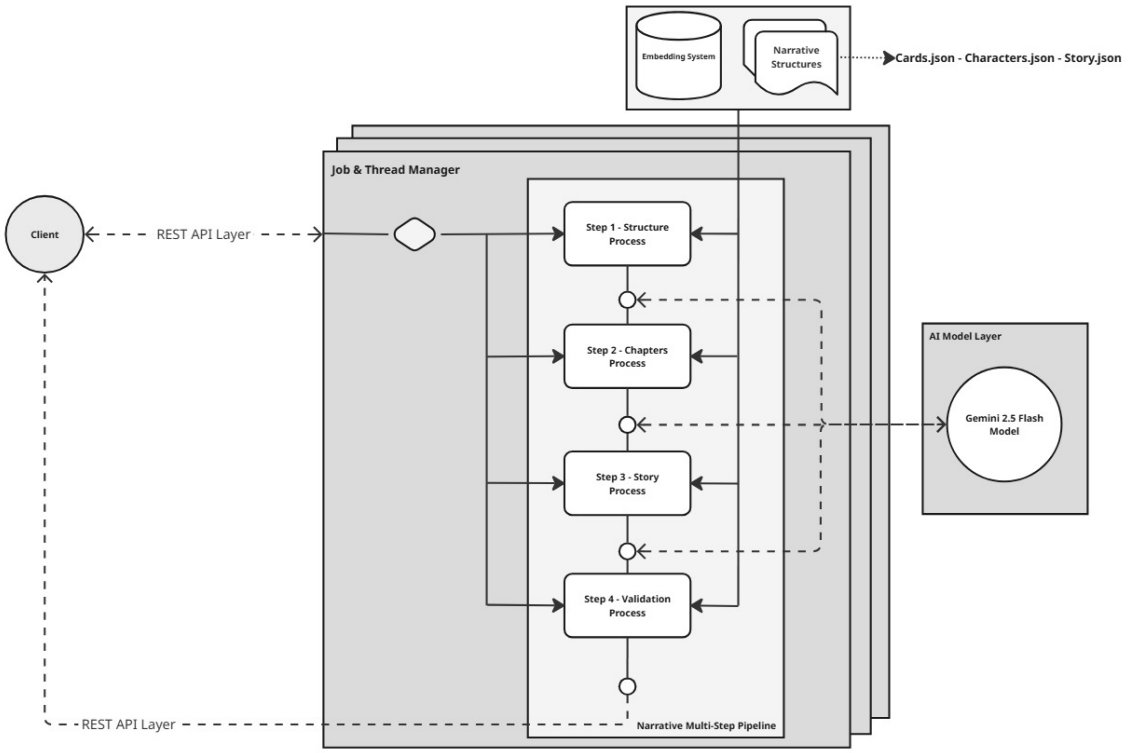

Figura 9: Overview architettura backend

3.5.2 Gestione delle risorse narrative strutturate

Un elemento chiave del backend è la gestione dei documenti narrativi strutturati, che rappresentano la base semantica su cui poggia l'intera generazione. Questi documenti includono file JSON dedicati alla definizione degli elementi fondamentali della storia, come archetipi dei personaggi, tropi narrativi, obiettivi di trama, eventi cardine e vincoli di coerenza.

I principali insiemi di dati utilizzati dal sistema sono:

- `Story.json`, che definisce la struttura narrativa di riferimento e gli elementi generali della trama;
- `Characters.json`, che descrive profili, ruoli, archetipi e relazioni dei personaggi;
- `Cards.json`, che contiene elementi contestuali e moduli narrativi utilizzati nella costruzione della struttura.

Questi file costituiscono una sorta di grammatica narrativa che permette all'AI di operare all'interno di uno spazio semantico controllato, riducendo la generazione

casuale e favorendo la coerenza interna. A supporto di tali risorse, il sistema utilizza inoltre testi di riferimento come *Slay the Dragon* che forniscono un quadro teorico per la definizione delle funzioni narrative e delle strutture utilizzate.

3.5.3 Sistema di *embedding* e *model recognition*

Per consentire al modello AI di lavorare efficacemente con le risorse e i documenti complessi, il backend integra un sistema di *embedding*, che trasforma testi e informazioni strutturate in rappresentazioni vettoriali. Queste rappresentazioni permettono al sistema di analizzare semanticamente i documenti, recuperare informazioni rilevanti e fornire al modello AI un contesto mirato e coerente.

Il sistema di *embedding* supporta un approccio tipico dei sistemi *Retrieval-Augmented Generation* (RAG), nel quale la generazione testuale viene guidata da conoscenze estratte dinamicamente. In questo modo il backend può "riconoscere" elementi pertinenti della storia, allineare le generazioni a vincoli narrativi predefiniti e ridurre errori legati alla perdita di contesto.

Questa metodologia permette di compensare alcuni limiti intrinseci dei modelli linguistici (in particolare la difficoltà nel mantenere coerenza su lunghe sequenze) attraverso il recupero mirato di informazioni semantiche.

3.5.4 Pipeline narrativa *multi-step*

La generazione della storia è strutturata come una pipeline composta da fasi distinte, ognuna delle quali elabora un livello specifico della narrazione. Tale divisione deriva dalla seguente considerazione: una generazione monolitica porterebbe a una maggiore perdita di coerenza e renderebbe difficile controllare in modo dettagliato l'evoluzione della trama.

La pipeline adottata nel backend segue un principio di evoluzione progressiva dei dati: ogni step riceve un input validato, applica regole narrative e produce un output formalizzato che diventa il punto di partenza per la fase successiva. Questo approccio riduce il carico cognitivo del modello, distribuisce la complessità lungo diversi livelli e consente di intervenire in modo mirato su eventuali incoerenze.

Tale struttura favorisce inoltre l'adattabilità del sistema, permettendo ad esempio di sostituire singole fasi, aggiungere nuove componenti o affinare la logica dei moduli senza compromettere l'intera architettura.

3.5.5 Sistema a thread e gestione asincrona dei job

Poiché la generazione narrativa può richiedere tempi di elaborazione significativi, il backend utilizza un sistema multithread per garantire fluidità e reattività. Le richieste provenienti dal frontend vengono gestite come job indipendenti, i quali possono essere eseguiti e monitorati senza bloccare l'intero server.

Questa metodologia assicura che più operazioni possano essere avviate in parallelo, migliorando l'esperienza dell'utente e prevenendo congestioni dovute a processi particolarmente onerosi. Il server mantiene uno stato interno per ciascun job, aggiornando dinamicamente l'avanzamento delle elaborazioni e restituendo notifiche al frontend quando i risultati sono pronti.

3.5.6 Comunicazione tramite REST API

La comunicazione tra backend e frontend avviene attraverso dei servizi REST, che definiscono un protocollo standard per l'invio delle richieste e la ricezione dei risultati. Ogni fase della pipeline è associata a un endpoint dedicato, che riceve i parametri necessari, avvia il job corrispondente e restituisce un identificatore univoco attraverso cui monitorare l'elaborazione.

L'adozione di un'architettura RESTful conferisce al sistema flessibilità e interoperabilità e permette, infatti, non solo l'integrazione con Unreal Engine, ma anche la possibilità futura di collegare strumenti esterni, editor narrativi o applicazioni di prototipazione. La separazione netta tra logica narrativa e interfaccia di presentazione favorisce inoltre la sostituibilità delle componenti e la scalabilità dell'intero sistema.

3.6 Frontend: Unreal Engine e interfaccia utente

Il frontend del sistema è realizzato in Unreal Engine [18], uno dei motori di sviluppo più diffusi e consolidati nell'industria videoludica. La scelta di questa tecnologia risponde alla volontà di integrare il sistema di generazione narrativa all'interno di un ambiente nativo per la produzione di videogiochi, consentendo al *game designer* di sperimentare e valutare la storia generata direttamente in un contesto affine al prodotto finale.

Quindi il frontend agisce come un mediatore, traducendo l'azione dell'utente in richieste formali inviate tramite API e presentando i risultati in maniera organizzata.

3.6.1 Tecnologie utilizzate in Unreal Engine

Il frontend sfrutta principalmente il sistema UMG (*Unreal Motion Graphics*) per la creazione dell'interfaccia utente e la gestione dei widget [20]. Questa tecnologia consente di progettare pannelli, pulsanti e aree di testo attraverso un approccio visuale, mantenendo al contempo la possibilità di estendere le funzionalità tramite Blueprint o codice C++.

L'utilizzo di Unreal Engine permette inoltre di predisporre con facilità future estensioni del progetto, come l'integrazione delle storie generate in prototipi di gameplay, la visualizzazione grafica della struttura narrativa o l'esplorazione interattiva del mondo narrativo prodotto dall'AI. Tale scelta metodologica mantiene aperta la possibilità di trasformare il sistema da un semplice strumento di supporto alla scrittura a un vero tool narrativo *embedded* nel flusso di sviluppo del gioco.

3.6.2 Comunicazione con il backend tramite REST API

Il frontend comunica con il backend attraverso chiamate HTTP REST usando il plugin VaRest disponibile sulla piattaforma [21]. Ogni azione dell'utente, avvio di uno step, richiesta dello stato di un job, recupero del risultato, viene tradotta in una chiamata API corrispondente.

All'interno di Unreal Engine, le richieste REST vengono gestite tramite componenti dedicati al *networking*, che si occupano di serializzare i dati in formato JSON, inviare la richiesta al server e interpretare la risposta. Il frontend aggiorna quindi dinamicamente l'interfaccia in funzione dei risultati ottenuti, mostrando l'avanzamento dei job, eventuali errori e gli output generati.

3.6.3 Interfaccia utente (UI/UX)

L'interfaccia utente è progettata secondo criteri di chiarezza e minimalismo, con l'obiettivo di permettere al *game designer* di concentrarsi sui contenuti narrativi e non sull'interazione con il sistema. La UI si articola in alcune sezioni principali:

- un pannello dedicato alla selezione dello step della pipeline;
- un'area per la configurazione dei parametri (titolo, descrizione, elementi opzionali);
- una sezione per visualizzare i risultati della generazione;
- un'area di sintesi narrativa per facilitare l'analisi del contenuto.

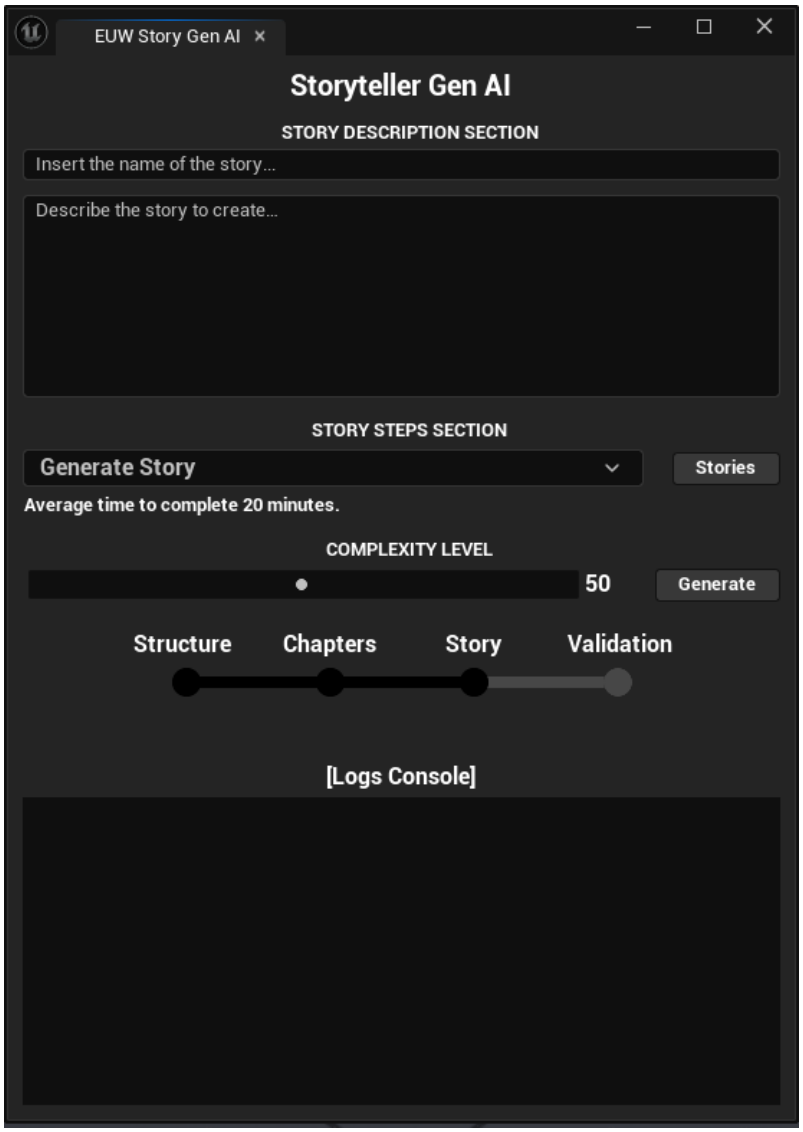

Figura 10: Overview UI/IX FrontEnd

3.6.4 Visualizzazione dei risultati

Una parte essenziale del frontend è dedicata alla presentazione dei contenuti generati. Poiché la storia cresce in complessità a ogni step della pipeline, è necessario un sistema di visualizzazione che possa adattarsi a dimensioni diversi.

Unreal Engine consente di organizzare questi contenuti all'interno di widget dinamici, aggiornabili in tempo reale, che permettono all'utente di scorrere, analizzare e confrontare i vari output. Questo approccio facilita il lavoro del *narrative designer*, che può valutare rapidamente la coerenza del racconto e richiedere nuove generazioni in caso di necessità.

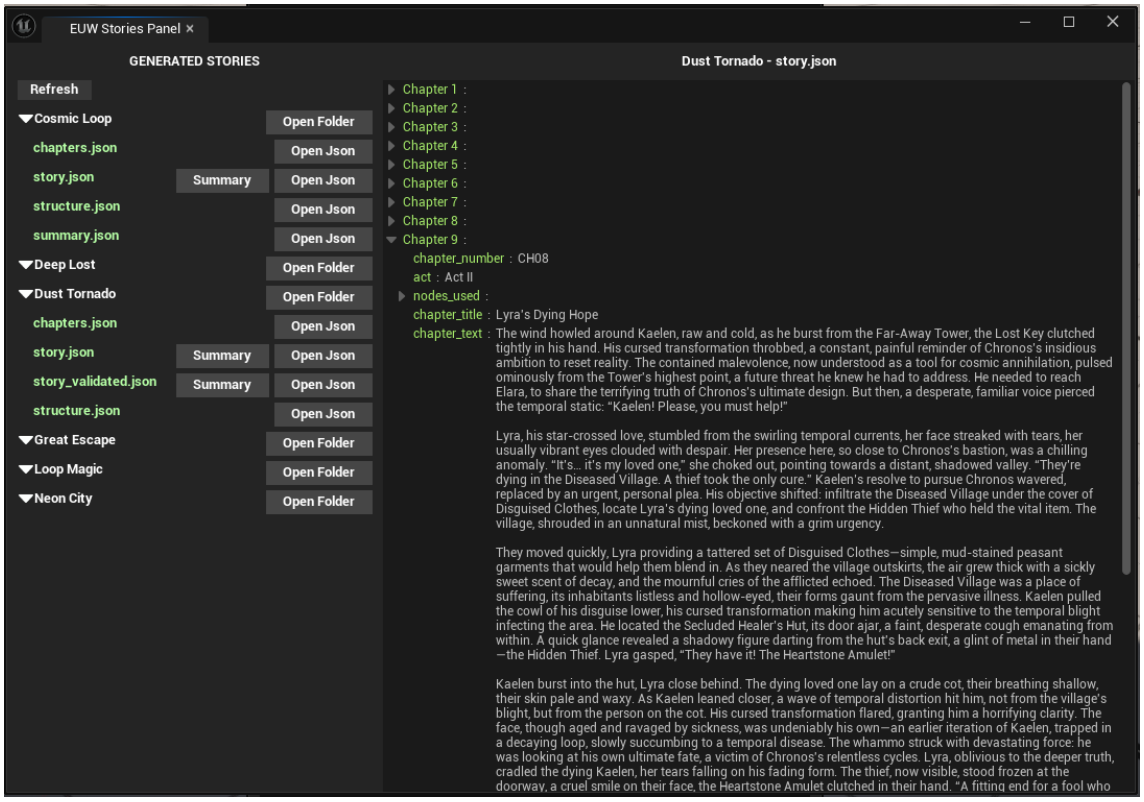

Figura 11: Overview architettura / workflow sistema

3.7 Modelli AI e strategie di *prompting*

L'impiego di modelli di intelligenza artificiale costituisce l'elemento centrale dell'intero sistema di generazione narrativa. Il backend si affida infatti a un modello linguistico di grandi dimensioni (LLM) per elaborare contenuti coerenti, strutturati e semanticamente rilevanti, basandosi sulle informazioni fornite dall'utente e sulle risorse narrative definite nei documenti strutturati.

3.7.1 Modello AI utilizzato

Il sistema utilizza un modello linguistico generativo di ultima generazione, caratterizzato da elevate capacità di comprensione del contesto, generazione testuale e adattamento narrativo. Il modello impiegato (*Gemini 2.5 Flash* [19]) è stato selezionato in base a criteri di efficienza, velocità di risposta e qualità dei testi prodotti, in quanto consente di gestire interazioni complesse senza compromettere i tempi di elaborazione.

Un LLM di questo tipo è in grado di:

- analizzare prompt articolati e riferimenti multipli;
- generare contenuti con struttura coerente;
- comprendere relazioni logiche e temporali;
- adattare lo stile narrativo alle esigenze del progetto;
- operare con informazioni provenienti da *embedding* e documenti esterni.

3.7.2 *Prompt engineering*

Il *prompting* rappresenta una delle componenti più delicate e importanti del sistema. Ogni fase della pipeline narrativa utilizza prompt specifici, progettati per guidare il modello in modo rigoroso, evitando generazioni eccessivamente libere o incoerenti.

I prompt sono strutturati secondo i seguenti criteri:

- contestualizzazione iniziale, che definisce ruolo, stile e vincoli del modello;
- descrizione degli obiettivi, che chiarisce ciò che deve essere generato nello specifico step;
- definizione di regole di contesto, struttura e relazioni;
- integrazione delle risorse narrative, come elementi dei file JSON, archetipi dei personaggi, tropi o regole strutturali;

- vincoli formali sull'output, che impongono un formato preciso e facilmente validabile;
- strategie di decomposizione, che suddividono richieste complesse in sottoproblemi più gestibili.

Questo approccio permette di superare uno dei limiti principali dei modelli generativi: la tendenza a produrre contenuti inconsistenti o eccessivamente generici. Il prompt funge così da "regia" del processo creativo, assicurando che il modello operi all'interno di confini ben definiti.

3.7.3 Controllo del contesto e gestione della coerenza

Una delle difficoltà maggiori nella generazione narrativa riguarda il mantenimento della coerenza tra parti distinte della storia. Per affrontare questo problema, il sistema adotta due strategie complementari.

Suddivisione del processo in step progressivi

La pipeline *multi-step* riduce la complessità delle generazioni, permettendo al modello di concentrarsi su un livello narrativo alla volta. In questo modo:

- Step 1 si focalizza esclusivamente sulla struttura;
- Step 2 sulla suddivisione in capitoli;
- Step 3 sulla narrazione dettagliata;
- Step 4 sulla verifica e correzione.

Questa metodologia riduce il rischio di errori cumulativi e migliora la qualità complessiva della storia.

Integrazione con *embedding* e documenti esterni

Attraverso il sistema di *embedding*, il modello riceve un contesto ricco e selezionato, evitando la perdita di informazioni chiave tra le varie fasi. Le risorse narrative (`Story.json`, `Characters.json`, `Cards.json`) diventano quindi parte attiva del processo generativo, garantendo continuità semantica e consistenza logica.

3.7.4 Ruolo del modello AI nel sistema

Nel complesso, il modello AI non è utilizzato come un semplice generatore di testo, ma come una componente centrale dell'architettura narrativa: un interprete delle regole, un trasformatore di strutture e un produttore di contenuti sotto vincoli formali.

L'adozione del *prompting* strutturato, l'integrazione con *embedding* e il controllo progressivo della pipeline permettono di sfruttare il modello in modo più affidabile e prevedibile, trasformandolo da strumento generico a motore specializzato per la costruzione di storie videoludiche.

Capitolo 4

Analisi del problema e del design del sistema

4.1 Introduzione al problema

La generazione automatica di storie videoludiche rappresenta una delle sfide più complesse nel campo dell'intelligenza artificiale applicata ai media interattivi. A differenza della narrativa tradizionale, la storia di un videogioco deve essere coerente, dinamica, adattabile alle interazioni del giocatore e capace di integrare elementi ludici, emozionali e strutturali all'interno di un unico flusso narrativo. Questa complessità rende particolarmente difficile affidarsi a un processo generativo lineare, soprattutto quando il sistema deve garantire stabilità semantica, continuità logica e varietà creativa su più livelli di granularità.

Il progetto sviluppato in questa tesi nasce proprio dall'osservazione delle difficoltà che incontrano i modelli AI generativi quando vengono applicati a problemi di narrativa estesa. Le prime sperimentazioni hanno evidenziato limiti significativi: perdita di coerenza tra sezioni differenti della storia, difficoltà nel mantenere la continuità dei personaggi, confusione tra elementi del mondo narrativo e una generale instabilità nella gestione della complessità quando la trama cresce in numero di eventi e ramificazioni.

La necessità di comprendere a fondo tali problematiche ha guidato la progettazione del sistema, Un'architettura modulare, basata su una pipeline a step progressivi, su risorse narrative strutturate e su meccanismi di embedding, è risultata essenziale per affrontare tali problematiche. Allo stesso tempo, sono emerse nuove sfide legate al prompting, ai limiti dei modelli linguistici, alla gestione del flusso asincrono delle operazioni, alla differenziazione dei contenuti generati e alla scalabilità dell'intero sistema.

In questo capitolo si analizzeranno in dettaglio le problematiche individuate durante la progettazione e la sperimentazione del sistema, evidenziando le motivazioni

che hanno portato alla suddivisione della generazione in step distinti, alla definizione di un grafo narrativo strutturato e all'introduzione di tecniche di validazione automatica. Tale analisi costituisce le fondamenta su cui si basa l'implementazione fornita nei capitoli successivi.

4.2 Complessità narrativa e struttura del grafo

La generazione automatica di storie videoludiche richiede un modello capace di rappresentare in modo formale gli elementi fondamentali della narrazione. Nel sistema sviluppato, questa rappresentazione prende la forma di un grafo narrativo, composto da nodi che descrivono eventi o stati della trama e da connessioni che esprimono relazioni di progressione, causa-effetto o alternative narrative. Di seguito un esempio di grafo minimo per la rappresentazione della struttura narrativa di base.

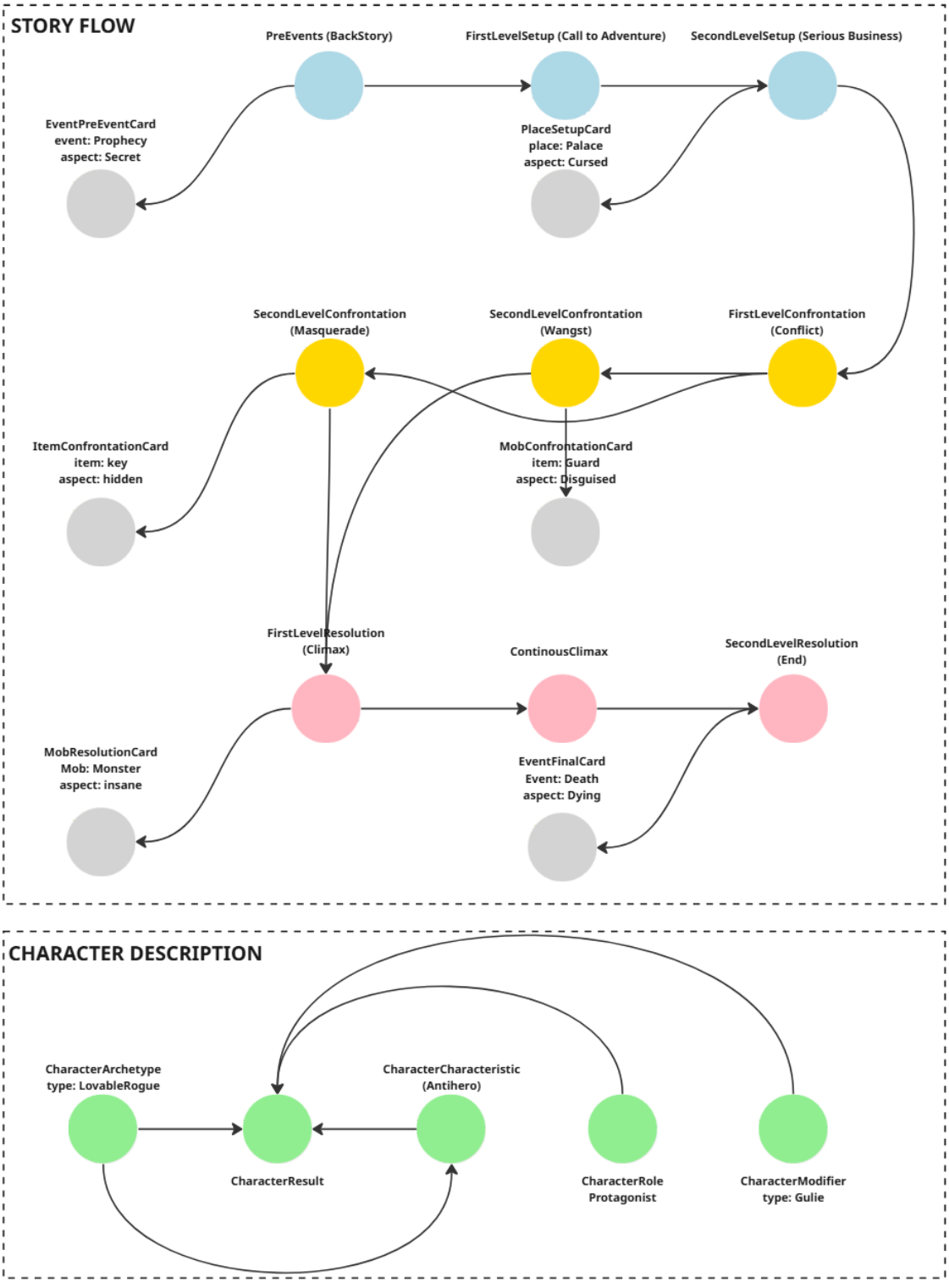

Figura 12: Minimal Example Graph Structure

Il grafo fornisce un supporto strutturale essenziale: da un lato permette di progettare una linea narrativa coerente, dall'altro introduce la possibilità di includere ramificazioni e percorsi alternativi tipici delle produzioni videoludiche moderne. Tuttavia, il livello di complessità del grafo influenza in maniera decisiva la qualità della narrazione generata dal modello, e determina il margine di variabilità che può essere effettivamente gestito.

4.2.1 Grafo semplice vs grafo complesso

Durante la sperimentazione del sistema sono emerse differenze significative legate alla struttura del grafo narrativo.

Grafo semplice: maggiore coerenza, minore profondità Un grafo semplice presenta poche ramificazioni, un numero ridotto di eventi chiave e una progressione narrativa prevalentemente lineare. Quando l'AI opera su una struttura di questo tipo, tende a garantire una buona coerenza interna: i personaggi mantengono ruoli e comportamenti più stabili, gli eventi seguono una logica prevedibile e il rischio di contraddizioni diminuisce in modo significativo. Tuttavia, la semplicità della struttura si traduce spesso in storie meno articolate, con limitata variabilità e scarsa adattabilità a generi narrativi che richiedono scelte più articolati, percorsi multipli o evoluzioni complesse del protagonista. La narrazione appare quindi più controllata ma meno ricca.

Grafo complesso: maggiore profondità, minore stabilità Un grafo narrativo complesso si caratterizza per un numero elevato di ramificazioni, percorsi alternativi e intrecci tra eventi e personaggi. Questa struttura consente all'AI di generare storie molto più ricche, articolate e diversificate, con una profondità che si avvicina maggiormente alle produzioni videoludiche moderne. Tuttavia, la complessità introduce anche un carico cognitivo che i modelli generativi riescono a gestire solo in parte. Quando la trama si ramifica ampiamente:

- il modello può perdere riferimenti stabiliti in capitoli precedenti;
- la coerenza dei personaggi può diventare instabile;
- alcuni eventi possono non allinearsi alla trama principale;
- aumentano le probabilità di incoerenze e contraddizioni tra diverse sezioni della storia.

In altre parole, maggiore è la profondità narrativa, minore è la stabilità che il modello riesce a mantenere lungo l'intero sviluppo della trama. Da qui emerge l'esigenza di un design che permetta di bilanciare varietà e coerenza, adottando

strategie come step generativi progressivi, vincoli strutturali più rigidi e strumenti di validazione dedicati.

4.2.2 Impatto della complessità sulla coerenza narrativa

L'aumento del numero di nodi e connessioni comporta una crescita delle informazioni che l'AI deve gestire simultaneamente. Quando la struttura si espande, si osservano difficoltà specifiche che dipendono dalla capacità del modello di mantenere un quadro narrativo globale. Uno dei problemi più evidenti riguarda la continuità logica: il modello può perdere eventi precedenti o reinterpretare in modo errato relazioni di causa-effetto, soprattutto quando queste dipendenze sono distanti tra loro. Anche i personaggi risultano più vulnerabili: cambiamenti improvvisi nelle motivazioni, variazioni di tono o comportamenti ingiustificati appaiono più spesso quando la narrazione si ramifica.

La coerenza tra i capitoli è un altro aspetto influenzato direttamente dalla complessità. Con l'aumento delle diramazioni emergono differenze indesiderate nella descrizione di luoghi, oggetti o relazioni, oppure generazioni ridondanti che duplicano eventi già accaduti. Le derive narrative, percorsi che si allontanano progressivamente dai temi centrali, diventano più frequenti.

Questi fenomeni hanno guidato la progettazione del sistema verso l'uso di step generativi sequenziali, validatori automatici e embedding che fungano da "ancoraggi semantici". Inoltre, la progettazione del grafo utilizza ruoli narrativi espliciti, che permettono al modello di muoversi all'interno di un contesto più definito evitando deviazioni non controllate.

4.3 Difficoltà nella generazione di storie complesse e logicamente coerenti

La produzione di storie videoludiche tramite modelli generativi incontra limiti che diventano particolarmente evidenti quando la trama richiede una continuità estesa nel tempo o una gestione articolata delle relazioni tra personaggi ed eventi. Anche in presenza di una buona struttura narrativa, il modello può faticare a mantenere una visione unitaria della storia, poiché non dispone di una memoria narrativa intrinseca ma opera esclusivamente sul contesto fornito di volta in volta.

La causa principale è che i modelli generativi non possiedono una memoria narrativa globale: ragionano soltanto sulla base del contesto immediatamente disponibile nel prompt. Di conseguenza, quando la storia cresce e il contesto non può più essere interamente fornito, la qualità logica della narrazione si degrada.

4.3.1 Caratterizzazione e stabilità dei personaggi

La costruzione di personaggi coerenti nel tempo è un altro elemento delicato. Nei videogiochi story-driven, i personaggi devono mantenere una psicologia riconoscibile, un'evoluzione credibile e un insieme di motivazioni stabili. Tuttavia, quando la narrazione si sviluppa in molti capitoli o in percorsi ramificati, il modello può attribuire al personaggio comportamenti inattesi, modificare relazioni consolidate o introdurre stati emotivi non coerenti con quanto stabilito nelle fasi precedenti.

Queste incoerenze risultano particolarmente problematiche quando la storia si biforca o si sviluppa attraverso molte alternative: il modello tende a "resettare" parte della caratterizzazione, trattando talvolta i percorsi alternativi come storie indipendenti.

4.3.2 Scalabilità della narrazione

L'aumento della lunghezza e della ramificazione richiede prompt sempre più ricchi e dettagliati. Quando questi superano la capacità effettiva del modello di integrarli, emergono ripetizioni, omissioni e semplificazioni non intenzionali. Oltre alle incoerenze logiche, si può osservare un degrado stilistico: descrizioni meno variegate, dialoghi più generici e una perdita graduale di profondità tematica.

4.3.3 Sensibilità del modello al prompting

La qualità dell'output dipende fortemente dalla formulazione del prompt. Piccole variazioni nell'ordine, nella chiarezza o nella quantità delle informazioni possono generare differenze significative nei contenuti prodotti. Questo rende necessario progettare prompt altamente controllati per ogni fase della pipeline, evitando ambiguità che potrebbero introdurre incoerenze.

4.3.4 Differenziazione delle storie generate

Un limite ricorrente riguarda la tendenza dei modelli a produrre storie simili tra loro. Anche modificando input e condizioni narrative, il modello converge spesso su temi ricorrenti, archetipi comuni e strutture narrative già note. Per ridurre questa omogeneità, il sistema integra risorse basate su embedding e database narrativi più ampi, che forniscono un contesto più ricco e stimolano il modello verso soluzioni meno prevedibili. Nonostante ciò, la varietà narrativa rimane influenzata dai limiti dei modelli attuali.

4.4 Limitazioni tecnologiche dei modelli AI generativi

Le difficoltà osservate nella generazione narrativa non dipendono esclusivamente dalla complessità intrinseca delle storie videoludiche, ma sono anche il risultato diretto delle limitazioni tecnologiche dei modelli di intelligenza artificiale attualmente disponibili. Sebbene i Large Language Models (LLM) abbiano compiuto enormi progressi negli ultimi anni, essi presentano ancora restrizioni strutturali che influenzano in modo significativo la qualità, la coerenza e l'affidabilità delle narrazioni prodotte. In questa sezione vengono analizzate le principali limitazioni riscontrate durante lo sviluppo del sistema e il modo in cui tali limiti hanno influenzato il design della pipeline multi-step.

4.4.1 Limiti dei LLM nel ragionamento a lungo termine

LLM utilizzato non possedendo un meccanismo interno che gli consenta di mantenere una rappresentazione consolidata della storia nel tempo. La sua capacità di ragionamento dipende principalmente dal contenuto fornito nel prompt. Questo vincolo diventa evidente quando la narrazione richiede di gestire elementi distanti tra loro, come l'evoluzione di un personaggio o la progressione di un arco narrativo complesso. Il modello non può richiamare autonomamente informazioni passate se non vengono esplicitamente reinserite nell'input. Da ciò deriva l'impossibilità, per LLM, di funzionare come un "narratore globale" senza un supporto esterno.

4.4.2 Lentezza e costi computazionali

La generazione narrativa richiede spesso prompt voluminosi, numerose richieste sequenziali e operazioni di validazione che aumentano sensibilmente il tempo di elaborazione. A ciò si aggiunge il costo computazionale del modello stesso: modelli più avanzati producono risultati qualitativi migliori, ma introducono tempi di risposta più elevati e un maggiore consumo di risorse.

Questa lentezza non rappresenta solo un ostacolo all'usabilità, ma influisce anche sulle scelte nella gestione della pipeline. Un approccio monolitico non sarebbe realisticamente utilizzabile, e persino la suddivisione in step richiede un attento bilanciamento tra qualità dell'output e tempi di elaborazione accettabili per l'utente.

4.4.3 Necessità del sistema a step

Le limitazioni descritte rendono impossibile affidare l'intera creazione della storia a un unico passaggio. Una narrazione complessa, con struttura, capitoli, evoluzione dei personaggi e coerenza globale, richiede un controllo progressivo, suddiviso in fasi specializzate, ognuna progettata per ridurre il carico del modello e delimiatare il

contesto operativo. La suddivisione del processo in più step risponde alle seguenti esigenze:

- ridurre il carico del modello distribuendo la complessità;
- fornire contesti delimitati e specifici per ogni fase;
- permettere verifiche intermedie della coerenza;
- introdurre controlli automatici che intercettano errori prima che si propaghino allo step successivo.

Ogni fase della pipeline ha quindi una funzione specifica: la struttura narrativa imposta la radice narrativa, i capitoli definiscono la progressione, la storia dettagliata sviluppa i contenuti e la validazione corregge gli errori generati in precedenza. Questo approccio è stato scelto come diretta conseguenza dei limiti operativi dei modelli AI.

4.4.4 Dipendenza dalle tecnologie disponibili

Le capacità effettive di generazione dipendono fortemente dal modello utilizzato. Esistono differenze significative tra modelli gratuiti, modelli commerciali e modelli specializzati, sia in termini di qualità dell'output sia nella capacità di sostenere contesti estesi.

Durante lo sviluppo del sistema è stato necessario operare entro i limiti dei modelli gratuiti disponibili, adottando soluzioni progettuali che compensassero le loro debolezze. È ragionevole prevedere che modelli futuri, più performanti o dedicati allo storytelling, possano ridurre significativamente molti dei limiti attuali. Tuttavia, al momento della realizzazione, tali vincoli hanno rappresentato una componente fondamentale del design.

4.5 Problematiche del prompting

Il prompting costituisce uno degli aspetti più critici nella progettazione di sistemi basati su modelli generativi. La qualità dell'output dipende infatti in modo diretto dalla precisione, dalla struttura e dalla coerenza semantica del prompt, che agisce come ponte tra l'intenzionalità dell'utente e l'interpretazione del modello. Durante lo sviluppo del sistema, il prompting si è rivelato un elemento centrale, capace di influenzare profondamente l'intera pipeline.

4.5.1 Sensibilità dei modelli al prompt

I modelli linguistici generativi mostrano una forte dipendenza dalla forma e dalla struttura delle istruzioni ricevute. Variando l'ordine delle informazioni, la granularità dei vincoli o il livello di dettaglio, il modello può produrre risultati molto diversi pur in presenza di contenuti simili. Tale sensibilità si manifesta soprattutto quando il prompt deve integrare molteplici elementi narrativi: personaggi, obiettivi, tono emotivo, vincoli strutturali e collegamenti con capitoli precedenti.

Un prompt troppo generico può portare il modello a interpretare liberamente la storia, introducendo elementi non desiderati. Al contrario, un prompt eccessivamente dettagliato può ridurre la varietà e la naturalezza della narrazione. Trovare un equilibrio tra guida e libertà espressiva diventa dunque essenziale.

Inoltre la lunghezza del prompt influenza la qualità della storia generata. Quando la quantità di informazioni necessarie supera una certa soglia, il modello può ignorare parti del contesto o enfatizzare dettagli minori in modo imprevedibile. La progettazione dei prompt deve quindi essere attenta, calibrata e adattata al livello narrativo trattato. E quindi necessario enfatizzare la necessità di utilizzare una struttura multi step, suddividendo il prompt per funzionalità.

4.5.2 Differenziazione degli aspetti narrativi

Un ulteriore problema riguarda la capacità del prompt di orientare il modello verso storie realmente diverse tra loro. Sebbene un LLM possa generare infinite variazioni superficiali, tende spesso a riutilizzare strutture narrative ricorrenti, archetipi comuni e schemi probabilistici consolidati. Il risultato è una variabilità limitata, soprattutto quando la storia richiede originalità tematica o una forte personalizzazione dei personaggi.

Senza un prompting mirato, le narrazioni risultano spesso troppo simili, sia nella progressione degli eventi sia nella natura dei conflitti o delle motivazioni dei personaggi. Per contrastare questa tendenza, il sistema utilizza basi dati strutturate, embedding narrativi e documenti di riferimento che ampliano i possibili spunti narrativi a disposizione del modello.

4.5.3 Impatto sul design dell'intero sistema

La complessità del prompting ha avuto un impatto diretto sul design complessivo del sistema. La necessità di fornire al modello istruzioni precise ma non limitanti ha portato a:

- suddividere il processo generativo in step con prompt dedicati;
- definire un formato di output uniforme e facilmente validabile;
- integrare vincoli narrativi esplicitati nei documenti strutturati;
- introdurre validatori che intercettano deviazioni o fraintendimenti del prompt;
- costruire un ambiente controllato in cui il modello opera con un contesto definito.

Le problematiche del prompting mostrano che la generazione narrativa non può essere lasciata alla sola creatività del modello, ma richiede una regia precisa e consapevole. La qualità dell'intera pipeline dipende dalla capacità di strutturare prompt che siano chiari, completi e calibrati per il livello narrativo trattato.

4.6 Problemi di coerenza interna

Nonostante la pipeline multi-step e l'uso di strutture narrative controllate, la generazione automatica di storie presenta inevitabilmente alcune incoerenze interne. Queste non derivano da errori del sistema in sé, ma dai limiti strutturali dei modelli linguistici.

In particolare, durante la sperimentazione sono emersi fenomeni ricorrenti come incoerenze temporali, variazioni non giustificate nella caratterizzazione dei personaggi, contraddizioni tra capitoli generati separatamente e discontinuità stilistiche. Si tratta di comportamenti tipici dei modelli generativi quando operano su narrazioni estese o articolate, e non possono essere completamente evitati attraverso il prompting o l'uso di documenti strutturati.

4.6.1 Il ruolo dei validatori nella gestione della coerenza

Per affrontare i problemi descritti è stato necessario introdurre un sistema di controllo e correzione basato su due validatori:

- Il *Label Validator* interviene sugli elementi semantici, verificando che personaggi, oggetti narrativi, luoghi e relazioni siano utilizzati in modo coerente e uniforme.
- Il *Flow Validator* si concentra invece sulla continuità tra i capitoli, individuando conflitti temporali, incoerenze negli eventi e derive narrative.

Questi strumenti non eliminano completamente le incoerenze, ma riducono sensibilmente la loro frequenza e impattano positivamente sulla leggibilità e stabilità della storia finale.

Essi rappresentano una risposta necessaria ai limiti dei modelli AI, integrando un livello di controllo "human-like" che l'AI non possiede nativamente.

4.7 Limiti di utilizzo del sistema

Nonostante la progettazione modulare, il sistema sviluppato presenta alcuni limiti legati sia alla natura dei modelli generativi, sia alle scelte adottate per garantire coerenza e controllabilità. Questi limiti ne definiscono con chiarezza il campo di applicazione, indicando quali contesti risultano più adatti e quali potrebbero richiedere estensioni o adattamenti futuri.

4.7.1 Focalizzazione sui videogiochi story-driven

Il sistema è stato progettato principalmente per supportare videogiochi con una forte componente narrativa. Le pipeline multi-step, la struttura del grafo, gli strumenti di validazione e i meccanismi di embedding rispondono alle esigenze tipiche degli story-driven games, nei quali:

- la trama costituisce il nucleo dell'esperienza;
- i personaggi hanno ruoli e archi narrativi definiti;
- gli eventi si sviluppano secondo un ordine coerente.

Questa focalizzazione implica che il sistema non è attualmente ottimizzato per generare contenuti per generi videoludici dove la narrativa non svolge un ruolo centrale. Pur essendo tecnicamente possibile utilizzare il sistema per generare storie di qualunque tipo, la sua efficacia diminuisce in ambienti in cui la narrazione non segue quella definita nel sistema.

4.7.2 Dipendenza dalla qualità dei documenti di supporto

La qualità delle storie generate è fortemente influenzata dai contenuti presenti nei documenti di riferimento. Se la base di dati è troppo limitata, non copre adeguatamente un genere narrativo o contiene archetipi troppo generici allora anche la generazione dell'AI risulterà meno ricca e meno diversificata.

Questo limite evidenzia il ruolo centrale dell'embedding semantico e del contesto strutturato: il sistema opera come un modello guidato da un repertorio di informazioni esplicite. In assenza di una base dati adeguatamente ampia, la varietà narrativa rimane limitata.

4.8 Complessità e prestazioni del sistema

La generazione automatica di storie videoludiche richiede un sistema capace di gestire una considerevole quantità di elaborazioni interne. La complessità narrativa e quella computazionale avanzano infatti parallelamente: più articolata è la storia, maggiore è lo sforzo richiesto al sistema per elaborarla, verificarla e mantenerla coerente. Per questo motivo, durante la fase di progettazione è stato necessario analizzare attentamente l'impatto della complessità sulla qualità della generazione e sulle prestazioni complessive della pipeline.

4.8.1 Complessità computazionale delle pipeline multi-step

La pipeline multi-step adottata nel sistema, che prevede la generazione della struttura, dei capitoli, del testo dettagliato e infine la validazione, è stata pensata per distribuire la complessità del compito. Ogni step lavora su un insieme specifico di informazioni e garantisce maggiore controllo sulla coerenza. Tuttavia, questa architettura comporta un carico computazionale superiore rispetto a una generazione monolitica: ogni passaggio richiede tempo, produce dati intermedi da analizzare e può richiedere correzioni o rigenerazioni parziali.

Anche se questa soluzione è più lenta, rappresenta un approccio realistico per mantenere un livello accettabile di qualità narrativa e di stabilità complessiva del racconto generato.

4.8.2 Responsività del sistema e percezione dell'utente

Dal punto di vista dell'utente, non è soltanto la velocità effettiva a determinare la qualità dell'esperienza, ma anche la percezione di reattività del sistema. Per questo motivo il backend utilizza un sistema multithread che evita il blocco dell'interfaccia e consente di monitorare l'avanzamento degli step.

Nonostante ciò, storie molto lunghe o dotate di molte ramificazioni possono richiedere un tempo di elaborazione significativo. La sensazione di "lentezza" può dunque emergere quando il modello deve processare una grande quantità di contesto, quando si utilizzano modelli particolarmente pesanti o quando i validatori rilevano errori che richiedono una rigenerazione del contenuto.

La complessità narrativa e quella computazionale sono due aspetti inseparabili del sistema. La generazione di storie articolate richiede un investimento computazionale elevato, determinato tanto dai limiti intrinseci dei modelli AI quanto dalla necessità di controllare in modo progressivo e rigoroso la coerenza narrativa. La pipeline multi-step, i validatori e la struttura del grafo aggiungono lavoro extra, ma rappresentano strumenti indispensabili per ottenere storie solide, coerenti e utilizzabili in un contesto videoludico.

4.9 Necessità della suddivisione in step

La suddivisione in step rappresenta uno degli elementi centrali del sistema sviluppato. La generazione di una storia completa, articolata, coerente e strutturata richiede infatti un processo progressivo, suddiviso in livelli di astrazione differenti, ciascuno con obiettivi specifici. Senza questa suddivisione, la qualità del racconto prodotto decadrebbe rapidamente, rendendo impossibile controllare la coerenza logica tra i vari elementi della storia.

4.9.1 Complessità crescente e impossibilità della generazione monolitica

Una generazione monolitica sarebbe, almeno in teoria, la soluzione più semplice: un prompt completo e una singola risposta che produca l'intera storia. Tuttavia, questo approccio è impraticabile per due motivi fondamentali. Da un lato, il modello non può gestire simultaneamente tutte le informazioni richieste per una narrazione complessa; dall'altro, un prompt eccessivamente lungo porta il modello a ignorare parte del contesto o a reinterpretarlo in modo errato.

L'assenza di controlli intermedi farebbe inoltre sì che eventuali incoerenze introdotte nelle prime fasi si propagherebbero fino al risultato finale. In altre parole, un errore iniziale comprometterebbe l'intera narrazione, costringendo a rigenerare tutto il contenuto.

4.9.2 Controllo della narrazione a livelli diversi

Ogni fase della pipeline affronta un livello distinto della narrazione e richiede un tipo di ragionamento diverso. La struttura narrativa fornisce la base, definendo eventi principali e turning point; i capitoli ne articolano il ritmo; la storia dettagliata arricchisce il tutto con dialoghi, emozioni e descrizioni; i validatori, infine, verificano che l'insieme sia coerente.

Ridurre questi livelli a un unico passaggio significherebbe sacrificare il controllo e la qualità del risultato, oltre a ridurre la granularità con cui è possibile gestire e correggere eventuali incoerenze.

4.9.3 Riduzione degli errori tramite generazione progressiva

La suddivisione in step permette di intercettare errori e deviazioni prima che diventino problematiche. Una struttura narrativa incoerente può essere corretta prima di generare i capitoli; un capitolo problematico può essere sistemato senza dover rigenerare l'intera storia; un evento fuori posto può essere ridefinito senza comprometterne altri già verificati. Questo approccio progressivo consente di individuare precocemente le incoerenze e impedire che si propaghino agli step successivi.

4.9.4 Maggiore flessibilità e adattabilità

Un ulteriore vantaggio della suddivisione in step è la maggiore flessibilità progettuale. Ogni fase può essere modificata, estesa o migliorata in maniera indipendente. Ad esempio, è possibile:

- sostituire il modello AI utilizzato in uno specifico step;
- introdurre nuovi moduli di generazione (come side quest o dialoghi);
- modificare il livello di dettaglio dei capitoli senza toccare la struttura narrativa.

La modularità non è dunque un semplice beneficio aggiuntivo, ma una componente essenziale per garantire scalabilità e adattabilità a futuri sviluppi.

4.9.5 Riduzione del carico cognitivo del modello AI

Un modello generativo opera sempre in base al contesto fornito nel prompt. Se questo contesto è troppo ampio o caotico, il modello perde parte delle informazioni, interpreta elementi in modo errato o produce risposte incongruenti. La suddivisione in step consente invece di fornire all'AI un contesto ridotto, mirato e specifico per la fase considerata.

In questo modo il modello lavora in condizioni più favorevoli, riducendo la probabilità di incoerenze e migliorando la qualità del testo generato.

Capitolo 5
Implementazione del sistema

5.1 Introduzione

A partire dal modello concettuale della pipeline narrativa multi–step descritta nei capitoli precedenti, è stato sviluppato un prototipo funzionante che integra: il modello di intelligenza artificiale generativa, strutture dati narrative, fonti teoriche, servizi REST e un'interfaccia utente realizzata in Unreal Engine.

La fase implementativa è stata progettata in stretta continuità con le considerazioni emerse durante l'analisi del problema: la suddivisione del processo in più step, l'utilizzo di strutture narrative specifiche, l'impiego di embedding e di modelli AI esterni solo elementi dettati dalle varie considerazioni fatte.

Dal punto di vista tecnologico, il sistema si basa su un'architettura client–server. Il *backend*, sviluppato in Python, funge da motore narrativo e gestisce l'intera pipeline di generazione attraverso una serie di moduli specializzati: creazione della struttura narrativa (Step 1), generazione dei capitoli (Step 2), stesura della storia dettagliata (Step 3) e validazione automatica (Step 4). Ogni modulo dialoga con il modello di intelligenza artificiale esterno (Gemini 2.5 Flash) tramite prompt strutturati e produce output in formato JSON, pensati per essere riutilizzati negli step successivi.

Il *frontend*, realizzato in Unreal Engine, espone tali funzionalità all'utente attraverso una serie di widget dedicati, permettendo di configurare i parametri principali (titolo, genere, complessità narrativa), avviare i diversi step di generazione, visualizzare i risultati testuali e ispezionare la struttura complessiva della storia. La comunicazione tra frontend e backend avviene mediante servizi REST , che fungono da intermediari tra le richieste dell'utente e l'esecuzione dei processi di generazione e validazione.

In questo capitolo verranno descritte nel dettaglio le scelte implementative adottate, partendo da una panoramica generale del sistema e del suo diagramma architetturale per poi concentrarsi sui singoli step della pipeline.

5.2 Descrizione generale del sistema

5.2.1 Backend

Il backend rappresenta il core computazionale dell'intero progetto ed è responsabile dell'elaborazione narrativa, della comunicazione con il modello AI e della gestione della pipeline multi–step. È stato sviluppato in Python per sfruttare la sua flessibilità, la disponibilità di librerie per il trattamento del linguaggio naturale e la facilità di integrazione con servizi REST.

Il backend esegue quattro funzioni principali, corrispondenti agli step della pipeline:

1. **Generazione della struttura narrativa** (Step 1): mediante il modulo `step1_structure.py`;
2. **Generazione dei capitoli narrativi** (Step 2): tramite `step2_chapters.py`;
3. **Creazione della storia dettagliata** (Step 3): implementato in `step3_story.py`;
4. **Validazione logica e semantica** (Step 4): suddivisa in due moduli: `step4_label_validator.py` e `step4_flow_validator.py`.
5. **Generazione sintesi storia** (Step 5): tramite il modulo `step5_summary.py`.

Ogni modulo della pipeline è progettato per operare in modo autonomo e produrre un output strutturato in formato JSON, così da poter essere riutilizzato dallo step successivo. Questo design riduce il rischio che errori locali compromettano l'intero processo e permette di intervenire su una singola fase senza modificare le altre.

La comunicazione tra backend e frontend avviene tramite servizi REST implementati in `server_jobs_main.py`. Il server espone endpoint dedicati per ciascuno step e utilizza un sistema multithread per gestire richieste concorrenti, evitando che l'elaborazione di una storia complessa blocchi il flusso di lavoro dell'utente. È inoltre presente un sistema di logging che consente di monitorare lo stato di avanzamento dei processi e di identificare eventuali anomalie durante la generazione e validazione.

5.2.2 Frontend

Il frontend del sistema è realizzato in Unreal Engine. L'interfaccia è composta da widget che consentono all'utente di:

- configurare i parametri iniziali (titolo, genere, complessità, descrizione della storia);
- avviare ciascuno dei cinque step della pipeline;

- visualizzare i risultati generati, sia in forma testuale sia grafica;

Il frontend non contiene logica funzionale: la sua funzione è esclusivamente quella di gestire l'interazione con l'utente e inviare le richieste al backend tramite chiamate REST. Tale separazione garantisce leggerezza, indipendenza dal modello AI utilizzato e facilità di integrazione con eventuali strumenti futuri. Inoltre, la scelta di Unreal permette di immaginare, in un'eventuale estensione del progetto, un'integrazione diretta tra la storia generata e il mondo di gioco, facilitando test rapidi o prototipazioni narrative.

5.3 Diagramma dell'architettura del sistema

Per rappresentare in modo chiaro l'organizzazione interna del sistema e le interazioni tra le sue componenti principali, è stato realizzato un diagramma architetturale che illustra il flusso dei dati, la suddivisione tra frontend e backend e il ruolo dei diversi moduli che compongono la pipeline narrativa.

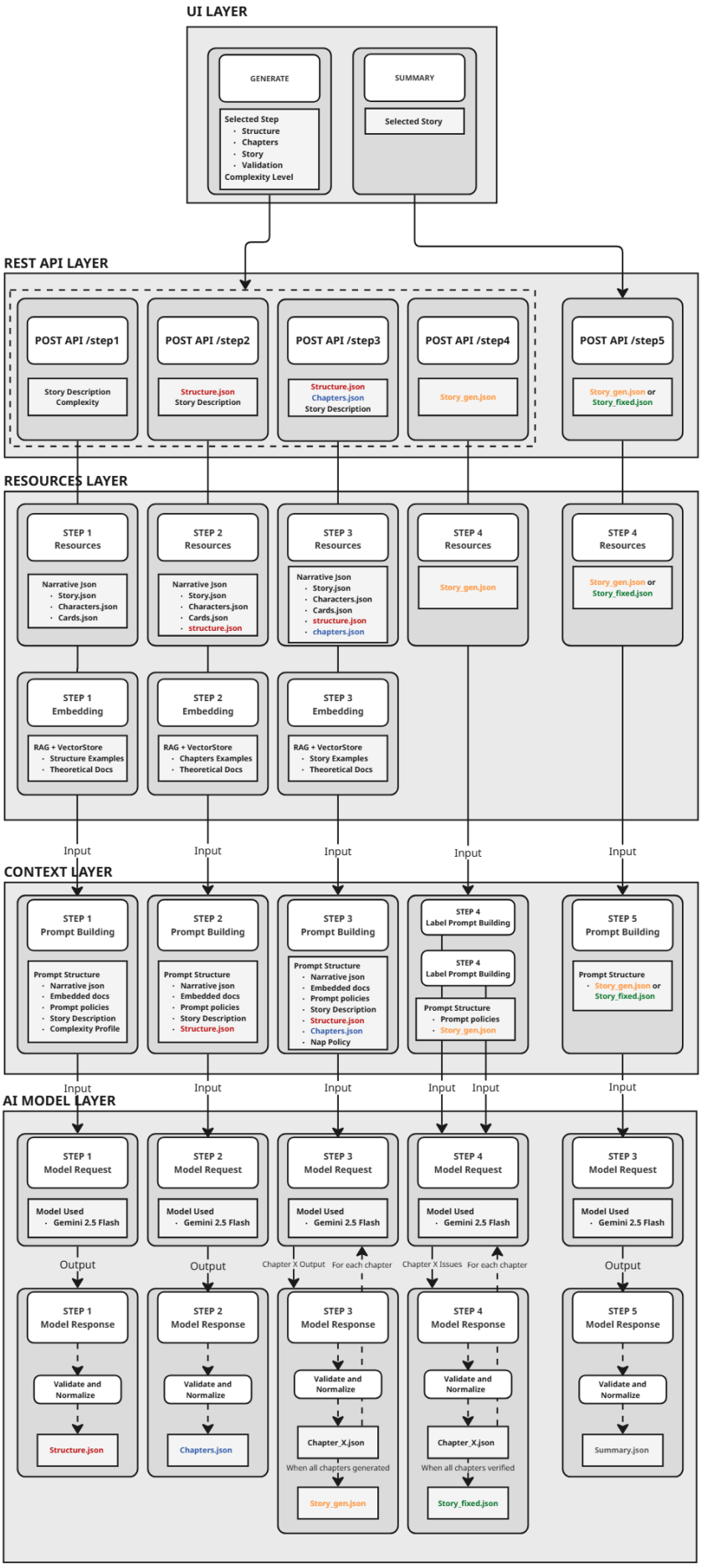

Figura 13: StoryGEN AI Architecture

Il diagramma evidenzia una struttura basata su due livelli distinti:

5.3.1 Livello Backend

Il backend, gestisce tutte le operazioni computazionali legate alla generazione e alla validazione della storia. Comprende:

- **Server REST**, definito nel modulo `server_jobs_main.py`, che espone endpoint specifici per ciascun step della pipeline;
- **Moduli di generazione** (`step1_structure.py`, `step2_chapters.py`, `step3_story.py`), responsabili delle tre fasi della produzione narrativa;
- **Moduli di validazione** (`step4_label_validator.py`, `step4_flow_validator.py`), dedicati al controllo semantico e logico della storia generata;
- **Moduli di sintesi** (`step5_summary.py` dedicato alla generazione di una sintesi della storia generata includendo elementi di gameplay utili per il designer;
- **Database narrativi JSON** (`Story.json`, `Characters.json`, `Cards.json`) che rappresentano il contesto strutturato utilizzato dall'AI;
- **Sistema di embedding**, integrato per consentire al modello AI di ancorarsi a risorse semantiche esterne oltre il semplice prompt;
- **Thread Manager**, incaricato di avviare le elaborazioni in processi indipendenti, evitando blocchi dell'intero server durante le chiamate ai modelli AI.

5.3.2 Livello Frontend

Il frontend, rappresenta lo strato di interazione del sistema.Il diagramma mette in evidenza:

- **Widget di controllo**, che permettono all'utente di selezionare la complessità narrativa, specificare titolo e descrizione della storia, scegliere lo step da eseguire e consultare gli output;
- **Modulo di comunicazione REST**, responsabile dell'invio delle richieste al server Python;
- **Modulo di visualizzazione**, che mostra la struttura narrativa generata (come il grafo prodotto dallo Step 1), i capitoli e la storia dettagliata.

5.3.3 Flusso dei dati

Il diagramma descrive anche il flusso operativo:

1. L'utente interagisce con il frontend e invia una richiesta al server.
2. Il backend elabora la richiesta applicando lo step corrispondente della pipeline narrativa.
3. Il modello AI viene interrogato tramite prompt specializzati.
4. Il backend elabora la risposta, la valida e genera eventuali output grafici (come la visualizzazione del grafo).
5. Il risultato viene restituito al frontend e mostrato all'utente.

5.4 Implementazione dello Step 1: Generazione della struttura narrativa

5.4.1 Come viene generata una struttura narrativa

Lo Step 1 rappresenta la fase fondamentale della pipeline, poiché definisce la struttura base della storia attraverso un insieme di nodi narrativi e relazioni. L'obiettivo dello step è generare una rappresentazione astratta della trama che sia formalmente strutturata, coerente con il contesto narrativo fornito dall'utente e adatta agli step successivi della pipeline. La generazione della struttura avviene tramite il modello di intelligenza artificiale (Gemini 2.5 Flash) e segue un processo guidato da tre elementi principali:

1. **Il prompt template**, costruito dinamicamente sulla base dei file narrativi (Cards.json, Characters.json, Story.json).
2. **Il livello di complessità scelto dall'utente**, che determina il numero di nodi da generare.
3. **Le regole strutturali fornite al modello**, come la suddivisione in tre atti e la presenza di specifiche categorie di nodi.

Il risultato finale è una struttura narrativa espressa in formato JSON, contenente una lista di nodi e una lista di relazioni tra tali nodi. Ogni nodo rappresenta un elemento narrativo funzionale (intro, conflitto, turning point, climax, ecc.), mentre le relazioni descrivono la progressione della trama e le eventuali ramificazioni.

5.4.2 Logica implementativa e flusso operativo

Tutto il processo è implementato nel modulo `step1_structure.py` e segue una sequenza di operazioni ben definite che trasformano i dati forniti dall'utente e i file narrativi del sistema in una struttura formale utilizzabile negli step successivi.

Il flusso operativo può essere descritto in modo lineare, poiché ogni fase prepara i dati necessari per quella successiva. Il sistema inizia con la raccolta del contesto narrativo, includendo il titolo della storia, la descrizione iniziale, e le informazioni narrative provenienti dai file `Story.json`, `Characters.json` e `Cards.json`. Questi dati forniscono al modello AI un insieme di vincoli, archetipi e risorse semantiche che guidano la generazione.

Una volta estratto il contesto, il sistema costruisce dinamicamente il prompt, integrando:

1. titolo, descrizione iniziale fornita dall'utente,
2. le informazioni narrative strutturate,
3. il profilo di complessità determinato dallo *slider* selezionato dall'utente,
4. le regole formali che definiscono la struttura a tre atti e i tipi di nodi che devono essere generati.

Il prompt così composto viene inviato al modello AI per richiedere la generazione della struttura narrativa completa. Il backend riceve quindi l'output del modello e procede al parsing e alla normalizzazione, convertendo il risultato in un file JSON valido. Una volta validato l'output, il sistema elabora la versione grafica della struttura tramite il modulo `GraphVisualizer.py`. Questa rappresentazione permette di visualizzare nodi, connessioni e distribuzione degli atti, tramite un sistema visivo.

Utilizzo dei file JSON narrativi

La generazione della struttura narrativa nello Step 1 è guidata da tre principali fonti di informazione strutturata: `Story.json`, `Characters.json` e `Cards.json`. Questi file contengono rispettivamente:

- **Story.json**: informazioni sui temi narrativi, archetipi, tropi e strutture comuni della storia.
- **Characters.json**: descrizioni dei personaggi, ruoli narrativi, tratti psicologici, obiettivi, relazioni.
- **Cards.json**: elementi contestuali (luoghi, oggetti, eventi ricorrenti, atmosfere narrative, minacce, risorse).

La loro funzione principale è fornire al modello AI un contesto coerente e ricco di informazioni, riducendo la variabilità eccessiva del modello e migliorando la coerenza semantica.

Questi file non sono stati creati arbitrariamente, ma derivano da un processo di estrazione e rielaborazione della tesi GHOST, un sistema narrativo che utilizzava "nodi concettuali" per descrivere elementi ricorrenti della narrazione. Durante l'esecuzione dello step, questi file vengono caricati e integrati nel prompt come blocchi di conoscenza semantica, così da ancorare la generazione ai vincoli e ai principi narrativi definiti dal sistema.

5.4.3 Struttura dei file narrativi: Story, Characters e Cards

Story.json: modello della struttura narrativa

Il file `Story.json` definisce lo schema astratto della storia in termini di fasi e tipi di nodi narrativi. Alla radice è presente il nodo `"Story"`, che contiene:

- una descrizione generale della storia (`Story_description`);
- tre macro–sezioni corrispondenti agli atti classici: `Setup`, `Confrontation` e `Resolution`.

Ciascuna di queste sezioni descrive una parte specifica della struttura a tre atti e contiene a sua volta insiemi di nodi parametrizzati. Ad esempio, la sezione `Setup` è organizzata come segue:

```
"Story": {
  "Story_description": "...",
  "Setup": {
    "Setup_description": "...",
    "PreEvents_{id}": { ... },
    "FirstLevelSetup_{id}": { ... },
    "SecondLevelSetup_{id}": { ... }
  },
  ...
}
```

Ogni sotto–blocco (ad esempio `PreEvents_{id}` o `FirstLevelSetup_{id}`) rappresenta una *famiglia di nodi* avente una funzione narrativa specifica. Nel caso di `PreEvents_{id}`, ad esempio, troviamo:

```
"PreEvents_{id}": {
  "PreEvents_description": "Describes events prior to the narrative...",
```

```
  "type": [
    "Backstory",
    "Prequel"
  ],
  "type_description": {
    "Backstory": "Background events or context from before the story begins...",
    "Prequel": "Past events specifically related to characters..."
  },
  "card": {
    "type": [
      "eventCard",
      "itemCard",
      "mobCard",
      "placeCard"
    ],
    "type_description": {
      "eventCard": "Significant events that influence the start...",
      "itemCard": "Objects that influence / describe the pre-narrative events",
      "mobCard": "Entities present before the main story begins.",
      "placeCard": "Places relevant to the initial setting or pre-events."
    },
    "connections_out": [...],
    "routes": [...]
  },
  "connection_out": [...],
  "routes": [...]
}
```

Gli elementi più rilevanti sono:

- `type`: l'insieme dei tipi di nodo possibili per quella famiglia (es. *Backstory*, *Prequel*);
- `type_description`: una descrizione semantica di ciascun tipo, utilizzata dal modello AI come guida;
- `card`: il legame con le card contestuali (eventi, oggetti, mob, luoghi) definite in `Cards.json`;
- `connection_out` e `routes`: indicazioni di connettività, usate per definire come quel tipo di nodo può collegarsi ad altri nella struttura complessiva.

Strutture simili sono presenti in `Confrontation` e `Resolution`, con blocchi quali:

- `FirstLevelConfrontation_{id}`, `SecondLevelConfrontation_{id}` per la fase di conflitto;
- `FirstLevelResolution_{id}`, `SecondLevelResolution_{id}`, `ContinuousClimax_{id}`, `ClimaxEvent_{id}` per la fase di risoluzione.

Nel complesso, `Story.json` fornisce un *meta–modello* della struttura narrativa: non descrive una singola storia, ma una famiglia di possibili storie, specificando quali tipi di nodi possono esistere in ciascun atto, come sono caratterizzati e come possono collegarsi tra loro. Lo Step 1 utilizza queste informazioni nel prompt per istruire il modello su:

- quali tipi di nodi scegliere per ciascun atto;
- come differenziare i nodi di setup, conflitto e risoluzione;
- come generare una struttura che rispetti la progressione logica definita dal modello di base.

Characters.json: modello dei personaggi

Il file `Characters.json` definisce un modello astratto per la costruzione dei personaggi. Alla radice troviamo il blocco `"Character_{id}"`, che non rappresenta un singolo personaggio, ma un *prototipo* di come un personaggio può essere descritto combinando:

- archetipi (`CharacterArchetype_{id}`);
- allineamenti e caratteristiche (`CharacterCharacteristic_{id}`);
- ruolo narrativo (`CharacterRole_{id}`);
- modificatori (`CharacterModifier_{id}`);
- risultato finale (`CharacterResults_{id}`).

La struttura complessiva è del tipo:

```
"Character_{id}": {
  "Character_description": "...",
  "CharacterArchetype_{id}": { ... },
  "CharacterCharacteristic_{id}": { ... },
  "CharacterRole_{id}": { ... },
  "CharacterModifier_{id}": { ... },
  "CharacterResults_{id}": { ... }
}
```

Ogni sotto–blocco rappresenta una porzione di identità del personaggio. Ad esempio, `CharacterArchetype_{id}` contiene:

```
"CharacterArchetype_{id}": {
  "CharacterArchetype_description": "...",
  "type": [
    "MadScientist",
    "StarCrossedLovers",
    "Tsundere",
    "Storyteller",
    "Mooks",
    "EccentricMentor",
    ...
  ],
  "type_description": {
    "MadScientist": "An obsessive and brilliant character driven by...",
    "StarCrossedLovers": "...",
    ...
  },
  "connection_out": [...],
  "routes": [...]
}
```

In modo analogo, `CharacterCharacteristic_{id}` organizza gli allineamenti in categorie come `HeroesAlignment` e `VillainsAlignment`, ciascuna con un elenco di possibili valori (es. *Hero*, *ChosenOne*, *Antihero* per gli eroi):

```
"CharacterCharacteristic_{id}": {
  "HeroesAlignment": [
    "Hero",
    "KnightInShineArmor",
    "ChosenOne",
    "Determinator",
    ...
  ],
  "HeroesAlignment_description": {
    "Hero": "The central character who undertakes the adventure...",
    ...
  },
  "VillainsAlignment": [...],
  "VillainsAlignment_description": {...},
  "alignment": "...",
```

```
  "connection_out": [...],
  "connection_in": [...],
  "routes": [...]
}
```

La sezione `CharacterRole_{id}` definisce invece il ruolo drammatico:

```
"CharacterRole_{id}": {
  "CharacterRole_description": "...",
  "type": [
    "Protagonist",
    "Antagonist",
    "TheWoobie",
    "CloudCuckoolander",
    ...
  ],
  "type_description": {
    "Protagonist": "The main character driving the story...",
    "Antagonist": "The opposing force or character challenging the protagonist...",
    ...
  },
  "connection_out": [...],
  "routes": [...]
}
```

Infine, `CharacterModifier_{id}` introduce modificatori (es. *Kid*, *Tragic*, *SuperHero*) che raffinano ulteriormente il profilo del personaggio, mentre `CharacterResults_{id}` descrive la sintesi finale:

```
"CharacterResults_{id}": {
  "CharacterResults_description": "...",
  "connection_in": [
    "Archetype",
    "Alignment",
    "Characteristics",
    "Role",
    "Modifier"
  ],
  "name": "",
  "description": ""
}
```

In termini funzionali, `Characters.json` non contiene "personaggi pronti", ma un insieme di categorie e combinazioni possibili. Lo Step 1 utilizza queste informazioni per:

- definire nodi narrativi centrati su personaggi (es. nodi in cui il focus è l'arco del protagonista o il ruolo dell'antagonista);
- arricchire la struttura con riferimenti a ruoli, archetipi e dinamiche di personaggi coerenti;

Cards.json: contesto, eventi, oggetti, luoghi e side quest

Il file `Cards.json` definisce gli elementi modulari che caratterizzano il contesto della storia: eventi, oggetti, creature, luoghi e possibili side quest. È organizzato principalmente in due blocchi:

- `"Cards"`: contiene le card principali suddivise in quattro categorie:
  - `EventCards`,
  - `ItemCards`,
  - `MobCards`,
  - `PlaceCards`;
- `"SideQuest_{id}"`: definisce informazioni strutturali relative alle quest secondarie.

Ogni famiglia di card è ulteriormente suddivisa per fase della storia (pre–events, setup, confrontation, resolution, quest). Ad esempio, per gli eventi:

```
"EventCards": {
  "EventPreEventsCard_{id}": { ... },
  "EventSetupCard_{id}": { ... },
  "EventConfrontationCard_{id}": { ... },
  "EventResolutionCard_{id}": { ... },
  "EventFinalCard_{id}": { ... },
  "EventQuestCard_{id}": { ... }
}
```

Ogni card contiene tipicamente due liste principali:

- una lista di elementi (eventi, oggetti, mob o luoghi);
- una lista di aspetti narrativi (aspetti tematici o descrittivi applicabili).

Ad esempio, una card di setup per gli eventi è strutturata come:

```
"EventSetupCard_{id}": {
  "events": [
    "HardTimes",
    "Contest",
    "Storm",
    "Dream",
    "Escape",
    "Fight",
    "Meet",
    "Hurt",
    "Chase",
    "FallInLove",
    "Death",
    "ObjectBreak",
    "Transformation",
    "Rescue",
    "Fire",
    "Prophecy",
    "Pursuit",
    "Imprison"
  ],
  "aspects": [
    "Cursed",
    "Disguised",
    "Evil",
    "Hidden",
    "Insane",
    "Invisible",
    "Lost",
    ...
  ],
  "connections_in": [...]
}
```

In modo analogo, le card degli oggetti, mob, luoghi contengono liste di elementi associati a tali tipi.

Infine, il blocco `SideQuest_{id}` definisce pattern per le quest secondarie, legando un personaggio a una serie di possibili percorsi (`routes`) e connessioni in uscita:

```
"SideQuest_{id}": {
```

```
  "characterName": "",
  "connection_out": [...],
  "routes": [...]
}
```

Dal punto di vista funzionale, `Cards.json` fornisce allo Step 1 un insieme di elementi concreti (eventi, oggetti, nemici, luoghi, aspetti) che possono essere associati ai nodi della struttura narrativa permettendo di:

- arricchire i nodi astratti definiti in `Story.json` con contenuti più specifici;
- mantenere coerenza tra i tipi di card utilizzati nelle diverse fasi della storia;
- predisporre fin dallo Step 1 il terreno per la generazione di quest e side quest durante gli step successivi.

5.4.4 Costruzione del prompt

La costruzione del prompt è uno degli elementi più delicati dello Step 1. L'intero comportamento del modello generativo dipende dalla chiarezza, dalla struttura e dai vincoli presenti nelle istruzioni. Il prompt impiegato nello Step 1 è organizzato in blocchi funzionali, ciascuno con un ruolo specifico nel controllo della generazione:

1. Definizione del ruolo del modello La prima parte del prompt chiarisce al modello quale compito deve svolgere e quale identità operativa deve assumere. In questa fase l'AI viene definita come un "architetto narrativo", responsabile della costruzione di grafi narrativi coerenti. Questo orientamento funzionale migliora la qualità delle scelte strutturali che il modello effettua durante la generazione.

2. Disponibilità delle risorse narrative Il prompt incorpora le informazioni estratte dai file `Story.json`, `Characters.json` e `Cards.json`. Queste risorse definiscono il vocabolario strutturale consentito: tipologie di nodi, archetipi, temi, ruoli dei personaggi, categorie di card e regole di composizione. In questo modo l'AI non inventa liberamente nuovi concetti, ma seleziona elementi all'interno di un inventario definito e controllato, garantendo maggiore coerenza semantica.

3. Regole generali sul formato dell'output Una sezione del prompt specifica in maniera rigorosa il formato dell'output. Il modello deve restituire un JSON valido, senza spiegazioni aggiuntive. Sono definite anche le strutture attese dei nodi, le chiavi obbligatorie e l'uso dei suffissi numerici per distinguere le istanze. Queste direttive servono a produrre un output formalmente corretto e compatibile con gli step successivi.

4. Vincoli strutturali sulla narrazione Una parte significativa del prompt riguarda le regole strutturali:

- come generare e connettere i nodi nei tre atti,
- come utilizzare correttamente le relazioni *1-to-1* e *1-to-N*,
- come istanziare sidequest, card narrative e archetipi dei personaggi,
- quali nodi possono includere card e quali no.

Queste regole evitano deviazioni indesiderate, garantendo che la struttura generata rispetti l'architettura prevista dal sistema e sia utilizzabile negli step successivi.

5. Regole semantiche e coerenza narrativa Il prompt contiene istruzioni dedicate alla coerenza tematica e stilistica. L'AI viene guidata a operare in un contesto coerente con il genere, il tono e l'ambientazione forniti dall'utente, evitando combinazioni incompatibili. Sono inoltre richieste descrizioni semantiche per ogni tipo, tema o allineamento utilizzato, così da rafforzare la base logica dei nodi generati.

6. Integrazione degli embedding Una sezione del prompt specifica che il modello può fare riferimento a informazioni provenienti da documenti indicizzati tramite embedding. Questo meccanismo permette all'AI di acquisire ulteriori dettagli semantici (ad esempio definizioni di archetipi o regole narrative) senza inserire manualmente lunghi testi nel prompt e migliorandone la coerenza narrativa.

7. Profilo di complessità Infine, il prompt integra i parametri calcolati dalla funzione `complexity_profile`, che ne stabiliscono la complessità generale della strutura da generare. Questi vincoli permettono al modello di adattare la struttura alla complessità desiderata dall'utente.

5.4.5 Il ruolo del livello di complessità

Il livello di complessità è un parametro centrale dello Step 1 e viene impostato dall'utente tramite uno *slider* con valori compresi tra 1 e 100. Questo valore non si traduce semplicemente in un numero fisso di nodi, ma viene interpretato dalla funzione `complexity_profile(slider)` come un vero e proprio profilo di complessità narrativa.

In primo luogo, il valore selezionato viene mappato in tre fasce concettuali (`zone`): *low*, *medium* e *high*, corrispondenti rispettivamente a strutture semplici, intermedie e complesse. All'interno di ciascuna zona, il valore numerico dello *slider* viene utilizzato per interpolare una serie di parametri che controllano diversi aspetti della struttura, tra cui:

- il numero massimo di nodi generabili (`max_nodes`);
- il numero minimo di personaggi principali richiesti (`min_characters`);
- il numero medio di diramazioni per arco narrativo (`branches_per_arc`);
- la proporzione tra collegamenti lineari e collegamenti "uno a molti" (`one_to_n_ratio`);
- la profondità massima di ricorsione delle ramificazioni (`recursion_depth`);
- il numero di collegamenti tra atti diversi (`cross_act_links`);
- la percentuale di percorsi opzionali rispetto alla linea principale della storia (`optional_paths_ratio`);
- il numero di side quest per personaggio principale (`per_major_character`).

Dal punto di vista quantitativo, la funzione `complexity_profile` definisce per ciascuna zona intervalli distinti:

- per una complessità **bassa** (`zone = "low"`), il numero massimo di nodi varia indicativamente da 3 a 25, con poche ramificazioni per arco, bassa profondità ricorsiva e un numero contenuto di collegamenti tra atti e percorsi opzionali; la struttura tende a essere più lineare e facilmente controllabile;
- per una complessità **media** (`zone = "medium"`), il numero massimo di nodi cresce (circa 26–40), aumentano le diramazioni per arco, la quota di collegamenti uno-a-molti e il numero di side quest per personaggio; la storia assume una struttura più articolata, con maggiore spazio per trame parallele e sviluppo dei personaggi;
- per una complessità **alta** (`zone = "high"`), il sistema può arrivare a gestire fino a circa 41–50 nodi, con molte ramificazioni per arco, forte presenza di percorsi opzionali, collegamenti incrociati tra atti e un numero significativo di side quest per ciascun personaggio principale; la struttura risultante è ricca, densa e fortemente interconnessa.

Il risultato della chiamata a `complexity_profile` è un dizionario che viene salvato e poi iniettato nel prompt dello Step 1. In questo modo il modello AI non riceve soltanto un generico "livello di complessità", ma un insieme di vincoli numerici e proporzionali che ne guidano le scelte durante la generazione della struttura: quante diramazioni introdurre, quanto rendere opzionali alcuni percorsi, quanto intrecciare gli atti tra loro e quanto spazio lasciare alle side quest.

5.4.6 Struttura a tre atti e tipologie di nodi narrativi

La generazione della struttura narrativa nello Step 1 si basa su un modello a tre atti (*Setup*, *Confrontation*, *Resolution*). Nel sistema sviluppato, questa suddivisione non è soltanto una scelta stilistica, ma un vincolo formale imposto al modello AI per garantire una progressione narrativa coerente. Ogni atto contiene famiglie di nodi con una funzione narrativa specifica, definite direttamente all'interno del file `Story.json`, che funge da blueprint dell'intera struttura.

Nel seguito vengono descritte le tre sezioni principali e le relative famiglie di nodi così come sono modellate in `Story.json`.

Atto I: Setup

L'atto di *Setup* introduce il mondo narrativo, stabilisce il tono della storia, presenta i personaggi principali e prepara gli eventi che condurranno al conflitto. In `Story.json`, questa fase è articolata in tre famiglie di nodi principali: `PreEvents_{id}`, `FirstLevelSetup_{id}` e `SecondLevelSetup_{id}`.

PreEvents_{id} I nodi `PreEvents_{id}` descrivono eventi anteriori all'inizio della storia, che contribuiscono a definire il passato del protagonista o del mondo di gioco.

In `Story.json` questa famiglia è caratterizzata dal campo `type` con due valori possibili:

- *Backstory*
- *Prequel*

Questi nodi rappresentano dunque la backstory o eventi "prequel" che fungono da radici della narrazione. Inoltre, `PreEvents_{id}` può includere un blocco `card` con diverse tipologie di card (*eventCard*, *itemCard*, *mobCard*, *placeCard*), collegate alle rispettive famiglie in `Cards.json`. In questo modo, gli eventi pregressi possono essere arricchiti da oggetti, luoghi o entità rilevanti per la storia.

FirstLevelSetup_{id} I nodi `FirstLevelSetup_{id}` espandono il setup iniziale introducendo motivazioni, obiettivi e prime decisioni del protagonista. Il campo `type` elenca una serie di archetipi narrativi di alto livello:

- *Call to Adventure*
- *The Hero Journey*
- *The Redemption Quest*

- *Saving The World*
- *They Fight the Crime*
- *Status Quo is God*
- *RetCon*

Queste etichette descrivono il tipo di configurazione narrativa che caratterizza l'ingresso del protagonista nella storia . La connessione in uscita `connection_out` e `routes` definiscono i collegamenti verso `SecondLevelSetup_{id}`.

SecondLevelSetup_{id} I nodi `SecondLevelSetup_{id}` rappresentano configurazioni tematiche che guidano lo sviluppo iniziale della storia sulla base degli eventi precedenti. A differenza dei `FirstLevelSetup_{id}`, qui il campo principale non è `type` ma `themes`, che contiene:

- *Serious Business*
- *Masquerade*
- *X Meets Y*
- *Wangst*
- *Moral Event Horizon*
- *Sealed Evil In A Can*
- *Applied Phlebotinum*

Questi temi descrivono il tono e le dinamiche narrative che caratterizzano il prosieguo dell'atto I. Anche `SecondLevelSetup_{id}` può includere un blocco `card` che collega eventi, oggetti, nemici e luoghi alle sezioni corrispondenti in `Cards.json`, e definisce i collegamenti verso il primo livello di Confrontation.

Atto II: Confrontation

L'atto di *Confrontation* rappresenta la fase centrale della storia, in cui le tensioni crescono, il protagonista affronta ostacoli e il conflitto principale si intensifica. In `Story.json`, questa sezione è composta da due famiglie di nodi: `FirstLevelConfrontation_{id}` e `SecondLevelConfrontation_{id}`.

FirstLevelConfrontation_{id} I nodi `FirstLevelConfrontation_{id}` costituiscono il primo livello del conflitto. Qui il campo `type` prevede i seguenti valori:

- *Conflict*
- *MacGuffin*
- *Chekhov's Gun*
- *Love Triangle*

Questi tipi rappresentano configurazioni narrative tipiche della fase centrale: conflitti veri e propri, oggetti o obiettivi che muovono la trama (MacGuffin), elementi introdotti perché avranno un ruolo chiave (Chekhov's Gun), o dinamiche relazionali complesse (Love Triangle). Le connessioni e le `routes` definiscono il collegamento verso i nodi di secondo livello del Confrontation.

SecondLevelConfrontation_{id} I nodi `SecondLevelConfrontation_{id}` modellano i temi che emergono durante il conflitto. Come per `SecondLevelSetup_{id}`, anche qui il campo principale è `themes`, con lo stesso insieme di possibili valori:

- *Serious Business*
- *Masquerade*
- *X Meets Y*
- *Wangst*
- *Moral Event Horizon*
- *Sealed Evil In A Can*
- *Applied Phlebotinum*

Questi temi vengono collocati nella fase di Confrontation per definire il tono e le implicazioni morali e simboliche del conflitto. Anche in questo caso è presente un blocco `card` che permette di collegare eventi, oggetti, mob e luoghi specifici alle card di Confrontation in `Cards.json`, e le `routes` collegano questa famiglia di nodi alla prima parte della Resolution.

Atto III: Resolution

L'atto di *Resolution* conclude la storia, risolve i conflitti principali e definisce lo stato finale del mondo di gioco e dei personaggi. In `Story.json`, questa fase è articolata in quattro famiglie di nodi: `FirstLevelResolution_{id}`, `ContinuousClimax_{id}`, `SecondLevelResolution_{id}` e `ClimaxEvent_{id}`.

FirstLevelResolution_{id} I nodi `FirstLevelResolution_{id}` rappresentano la parte iniziale della risoluzione, comprendendo rivelazioni chiave e momenti di climax. Il campo `type` contiene:

- *The Reveal*
- *Climax*

Questi valori identificano rispettivamente il momento della rivelazione (quando informazioni nascoste vengono finalmente svelate) e il culmine della storia, in cui la tensione raggiunge il suo apice. Questa famiglia include anche un flag `BigDamnHero` (*Yes/No*), che indica se il protagonista compie un'azione eroica decisiva, e può includere card legate alla fase di climax e risoluzione.

ContinuousClimax_{id} I nodi `ContinuousClimax_{id}` rappresentano la continuazione del climax e la transizione verso la parte finale della risoluzione. Non definiscono tipi o temi aggiuntivi, ma modellano il flusso tra il culmine della storia e gli esiti conclusivi, tramite campi di connessione in ingresso e in uscita (`connection_in`, `connection_out`) e relative `routes`.

SecondLevelResolution_{id} I nodi `SecondLevelResolution_{id}` descrivono la chiusura vera e propria della storia e la sua valenza tematica finale. Qui il campo `type` prevede:

- *End*
- *Aesop*

End rappresenta la conclusione degli eventi e la chiusura dei fili narrativi, mentre *Aesop* indica una conclusione con esplicita lezione morale. Anche questa famiglia può usare un blocco `card` per collegare eventi, luoghi, oggetti e mob alle card finali definite in `Cards.json`.

ClimaxEvent_{id} I nodi `ClimaxEvent_{id}` modellano un evento drammatico specifico collegato alla condizione di `BigDamnHero`. Sono attivati quando il protagonista compie un'azione eroica decisiva nel contesto del climax. Questi nodi non definiscono ulteriori tipi, ma fungono da punto focale per rappresentare l'apice dell'eroismo del protagonista e le sue conseguenze narrative.

Relazioni e regole di collegamento tra nodi

Ogni famiglia di nodi definisce campi `connection_in`, `connection_out` e `routes` che specificano come i nodi possono essere collegati tra loro. Queste informazioni vengono utilizzate dal modello per:

- guidare il collegamento logico tra le diverse fasi della storia,
- evitare salti narrativi incoerenti,
- mantenere una progressione tra Setup, Confrontation e Resolution

Ruolo delle tipologie di nodi nella generazione

Durante la generazione della struttura, lo Step 1 seleziona i tipi di nodo sulla base dell'atto di appartenenza, del profilo di complessità e delle regole definite in `Story.json`. A ciascun nodo vengono poi associati:

- elementi di contesto (eventi, luoghi, oggetti, entità) derivati da `Cards.json`;
- riferimenti a personaggi, archetipi e ruoli definiti in `Characters.json`;
- connessioni in ingresso e in uscita coerenti con i campi `connection_in`, `connection_out` e `routes`.

Il risultato è una struttura narrativa coerente, modulare e semanticamente ancorata al modello di base, pronta per essere sviluppata in capitoli nello Step 2.

5.4.7 Generazione della struttura narrativa: funzionamento, logica e output

La generazione della struttura narrativa nello Step 1 è il risultato dell'interazione tra tre componenti principali: il prompt strutturato, i file JSON contenenti la base di conoscenza narrativa e il modello AI. L'obiettivo di questa fase è generare un grafo narrativo completo, costituito da un numero definito di nodi distribuiti nei tre atti e collegati da relazioni logiche. Tale grafo rappresenta l'ossatura della storia e costituisce l'input principale per le fasi successive della pipeline.

La generazione avviene attraverso una sequenza ordinata di operazioni che combinano vincoli strutturali, contenuti semantici e scelte autonome dell'AI. Nel seguito viene illustrato il funzionamento interno del processo.

Selezione dei nodi e livello di complessità

Il processo inizia dal parametro di complessità scelto dall'utente, che determina il numero totale di nodi da generare. Tipicamente:

- un livello di complessità basso genera pochi nodi, privilegiando linearità e coerenza;
- un livello medio produce una struttura articolata con più ramificazioni;
- un livello elevato genera numerosi nodi e percorsi alternativi, aumentando la profondità narrativa ma anche la possibilità di incoerenze.

Il modello, guidato dal prompt, suddivide automaticamente il totale dei nodi tra i tre atti, rispettando proporzioni drammaturgiche standard (più nodi nel Confrontation, meno nel Resolution). Per ciascun nodo, il modello seleziona:

1. la famiglia di appartenenza (es. PreEvents, FirstLevelConfrontation, ClimaxEvent);
2. il tipo specifico all'interno della famiglia (es. Backstory);
3. eventuali card associate (eventi, oggetti, luoghi, mob);
4. eventuali riferimenti a personaggi o archetipi.

Questa scelta combina regole del modello (vincoli narrativi) con "ispirazioni" tratte dai file JSON.

Costruzione del contenuto dei nodi

Una volta determinato il tipo di nodo, il modello genera una breve descrizione che ne sintetizza la funzione narrativa. La descrizione viene prodotta sulla base di:

- la definizione del tipo in `Story.json`;
- il contesto tematico e ambientale fornito da `Cards.json`;
- gli archetipi o ruoli dei personaggi estratti da `Characters.json`;
- la sintesi descrittiva fornita dall'utente (titolo, pitch narrativo o descrizione generale).

Il risultato è una rappresentazione ad alto livello di un evento o stato narrativo. Esempio (semplificato):

```
{
"id": "N3",
"act": 1,
"type": "BackStory",
"description": "Il protagonista assiste a un evento
                misterioso nella Foresta di Lume,
che mette in moto la catena degli eventi principali.",
"cards": ["Event: Prophecy", "Place: Forest"]
}
```

Generazione delle connessioni tra nodi

Una volta generati i nodi, il modello si occupa di definire le connessioni tra di essi. Le connessioni rappresentano:

- relazioni ,
- transizioni narrative,
- sviluppi alternativi,
- dipendenze funzionali tra eventi.

`Story.json` fornisce indicazioni sulle possibili connessioni (connection out) e percorsi (routes) per ciascuna famiglia di nodi. Il modello utilizza tali indicazioni per:

1. evitare collegamenti incoerenti (es. un nodo di climax che precede un nodo di setup);
2. mantenere una progressione corretta tra i tre atti;
3. costruire ramificazioni sensate quando richieste da una complessità elevata.

Esempio di connessione generata:

```
"connections": [
{ "from": "Story.Prequel", "to": "Story.FirstLevelSetup" },
{ "from": "Story.FirstLevelSetup", "to": "Story.SecondLevelSetup" },
{ "from": "Story.SecondLevelSetup", "to": "Story.FirstLevelConfrontation" }
]
```

Formattazione dell'output

Il risultato finale dello Step 1 è un JSON strutturato contenente:

- lista dei nodi, ciascuno con:
  - id univoco,
  - atto di appartenenza,
  - tipo,
  - descrizione sintetica,
  - card associate.
- lista delle connessioni,
- metadati (complessità scelta, numero dei nodi, timestamp di generazione).

5.5 Implementazione dello Step 2: Generazione dei capitoli narrativi

Lo Step 2 estende la struttura generata nello Step 1 e la trasforma in una sequenza di capitoli narrativi organizzati nei tre atti della storia. Mentre nel primo step i nodi rappresentano elementi astratti del grafo, in questa fase ciascun nodo viene reinterpretato come un capitolo dotato di obiettivi narrativi, elementi scenici e primi spunti di gameplay. Il risultato è un JSON strutturato che funge da base per la generazione della storia testuale nello Step 3.

Dal punto di vista implementativo, lo Step 2 opera combinando tre insiemi di informazioni:

- la struttura narrativa (nodi e relazioni) prodotta nello Step 1;
- le basi semantiche contenute in `Story.json`, `Characters.json` e `Cards.json`;
- la `NAP Policy`, che definisce come gli aspetti narrativi si traducano in elementi ludici (encounters, interazioni, rischi, ostacoli, varianti).

5.5.1 Logica implementativa

La generazione dei capitoli segue un processo ricorrente che, per ciascun nodo della struttura narrativa, combina analisi strutturale e interpretazione semantica. In primo luogo il sistema esamina il nodo, identificandone l'atto di appartenenza, il tipo o il tema narrativo, le connessioni previste nel grafo e il ruolo drammatico che esso

ricopre. Questa lettura permette di collocare il nodo nella progressione della storia e di definirne la funzione all'interno dell'arco narrativo dell'atto.

Una volta interpretato il nodo sul piano strutturale, il sistema recupera tutto il contesto semantico necessario alla costruzione del capitolo. Vengono quindi analizzati i personaggi coinvolti insieme alle card pertinenti associate al nodo. A questo si aggiungono le descrizioni semantiche di tipi, temi e altri elementi narrativi definiti nei file JSON. L'integrazione di queste informazioni consente di mantenere continuità con la struttura generata nello Step 1, garantendo che ogni capitolo rifletta coerentemente gli elementi introdotti in precedenza e contribuisca alla progressione complessiva della storia.

5.5.2 Costruzione del prompt

La costruzione del prompt nello Step 2 è organizzata in modo modulare, analogamente a quanto avviene nello Step 1 è permette di guidare il modello nella generazione del chapter plan, uno schema di tutti i capitoli della storia, includendo componenti sia narrative che ludiche.

1. Definizione del ruolo del modello La sezione iniziale stabilisce che l'AI deve operare come "narrative video game writer", incaricato di trasformare la struttura a nodi generata nello Step 1 in una sequenza coerente di capitoli. Il modello deve quindi progettare capitoli dotati di una chiara funzione narrativa e ludica. Fin da subito vengono esplicitati i tre elementi obbligatori di ogni capitolo: la narrativa, gli *Encounter Plans* e il relativo *Chapter Adapt*.

2. Accesso alle risorse narrative Il prompt rende disponibili al modello tutti gli elementi JSON necessari: la struttura narrativa prodotta in precedenza, le definizioni dettagliate di nodi, personaggi e card, e la descrizione iniziale della storia definita dall'utente. Questo insieme costituisce il contesto semantico entro il quale l'AI deve operare.

3. Regole formali per la composizione dei capitoli Viene inoltre ribadita la divisione obbligatoria della storia nei tre atti classici e la proporzione dei capitoli assegnata a ciascuna sezione. Altre indicazioni riguardano la necessità di una progressione logica e cronologica, la continuità tematica e la coerenza tra eventi, che richiede che ogni capitolo rifletta gli eventi accumulati lungo i nodi precedenti del grafo narrativo.

4. Trasformazione dei nodi in segmenti narrativi Il prompt chiarisce che un capitolo non equivale a un singolo nodo. Il modello deve invece ragionare per sequenze, fondendo più nodi quando compongono insieme una fase narrativa coerente

o, al contrario, suddividendoli in più capitoli quando il contenuto è denso o comporta sviluppi multipli. Questa sezione definisce anche come gestire rami paralleli: essi non devono produrre percorsi alternativi, ma diverse sfaccettature di una stessa linea narrativa che il modello deve integrare in un'unica sequenza.

5. Uso corretto di personaggi, card e sidequest Un'altra parte del prompt regola l'integrazione degli elementi narrativi. Il protagonista deve essere presente in ogni capitolo, e i personaggi secondari devono influenzare attivamente gli eventi. Le card devono essere incorporate come elementi che modellano gli ambienti, gli ostacoli e i segnali narrativi. Le sidequest, devono essere collocate con criterio all'interno della progressione dei capitoli, avere un chiaro aggancio alla storia principale e contribuire in maniera verificabile al suo sviluppo.

6. Costruzione del ChapterAdapt tramite NAP Policy Una delle componenti più tecniche del prompt riguarda l'uso della NAP Policy, che definisce le caratteristiche di giocabilità del capitolo. Il modello deve analizzare i nodi utilizzati, mapparli alle categorie della policy fornita e generare un unico *chapter_adapt* che ne stabilisce ritmo, interattività, intensità , modalità diegetica e quantità minima e massima di encounter. Questo meccanismo assicura che ogni capitolo sia calibrato non solo narrativamente, ma anche in termini di esperienza di gioco.

8. Produzione degli Encounter Plans Gli Encounter Plans rappresentano il cuore ludico del capitolo. Ogni encounter descrive obiettivi, spazi, ostacoli, segnali di feedback, percorsi di successo o fallimento e possibili variazioni dello stato narrativo. Il modello è tenuto a conformarsi ai vincoli del ChapterAdapt.

9. Regole di continuità Per evitare incoerenze nella progressione, il prompt definisce infine le regole che governano la transizione tra capitoli. Ogni capitolo deve aprire reagendo in modo naturale alla chiusura del precedente, introducendo personaggi o informazioni solo quando motivati e rispettando sempre la catena causa–effetto. Ogni cambiamento di luogo richiede una transizione esplicita e nessun elemento può apparire senza essere stato acquisito o introdotto in precedenza.

10. Specifica del formato dell'output Il prompt si conclude imponendo un formato di output rigidamente definito: un singolo oggetto JSON contenente l'elenco dei capitoli, ciascuno composto da titolo, nodi utilizzati, encounter plans e chapter adapt. Questa struttura garantisce la validità dell'output e la sua immediata integrazione nello Step 3.

5.5.3 Utilizzo della NAP Policy: Encounter Plan e Chapter Adapt

All'interno dello Step 2, la generazione dei capitoli non si limita a strutturare la narrazione: deve trasformare ogni nodo del grafo in un segmento giocabile, dotato di ritmo, interattività e obiettivi concreti. Per ottenere questo risultato il sistema utilizza due elementi complementari, Chapter Adapt ed Encounter Plan, guidati dalla NAP Policy (Node-Adaptive Playability). Questi tre componenti lavorano insieme per tradurre la natura narrativa di ogni nodo in una precisa configurazione di gameplay.

Il ruolo della NAP Policy

La NAP Policy può essere definita come la grammatica di giocabilità dello Step 2: definisce quali livelli di interattività, opposizione, ritmo e quantità di encounter sono appropriati per ciascun tipo di nodo e per ciascun atto. Essa fornisce al sistema una mappatura esplicita che collega:

- il tipo del nodo (es. Prequel, Setup, Confrontation, Resolution),
- la fase dell'arco narrativo in cui il nodo si trova,
- e le caratteristiche di giocabilità necessarie (beat profile, whammo, interactivity level, opposition, encounter count).

Questo garantisce coerenza nel ritmo tra capitoli dello stesso atto e variabilità progressiva man mano che la storia evolve. Di seguito un estratto della NAP Policy applicata:

```
{
  "nap_version": "1.0",
  "beats_profiles": {
    "soft3": ["Hook&Goal","Complication","StateDelta+Hook"],
    "soft5": ["Hook&Goal","Approach","Complication",
              "Push/Tradeoff","StateDelta+Hook"],
    "full7": ["Hook&Goal","Approach","Complication","MidGoal+Whammo",
              "Reversal","Push/Tradeoff","StateDelta+Hook"]
  },
  "acts_caps": {
    "I":   { "max_opposition_intensity": 2, "whammo_severity": "soft→medium" },
    "II":  { "max_opposition_intensity": 3, "whammo_severity": "medium→hard" },
    "III": { "max_opposition_intensity": 3, "whammo_severity": "hard" }
  },
  "nodes": {
    "PreEvents.Backstory": {
      "narrative_scope": "Contextualize the events preceding the
                          story (not playable or very light)",
      "beats_required": "soft3",
      "interactivity_level": 0,
      "whammo_required": false,
      "opposition_intensity": 0,
      "encounters_minmax": [0,1],
      "default_diegesis_mode": "narrated_past",
      "allowed_diegesis_modes": ["narrated_past","present_time"]
    },
    ...
    "SecondLevelConfrontation": {
    "narrative_scope": "Escalation + tough choices",
    "beats_required": "full7",
    "interactivity_level": 3,
    "whammo_required": true,
    "opposition_intensity": 3,
    "encounters_minmax": [2,3]
    },
  }
}
```

Chapter Adapt: policy per giocabilità del capitolo

Una volta determinati i nodi da cui il capitolo deriva, lo Step 2 applica la NAP Policy per costruire il Chapter Adapt, cioè l'oggetto che definisce i limiti e le regole entro cui il capitolo deve essere giocato. Il Chapter Adapt stabilisce:

- il beat profile che definisce il ritmo interno del capitolo (soft3, soft5, full7);
- il livello di interattività previsto;
- l'intensità massima dell'opposizione consentita;
- se il capitolo richiede un whammo (twist ludico-narrativo);
- il numero minimo e massimo di encounter da generare;
- il diegesis mode, cioè il modo in cui gli eventi devono essere percepiti (presente, flashback giocabile, narrato, ecc.);
- lo scope narrativo, che riassume il ruolo funzionale del capitolo.

Il Chapter Adapt è quindi il "contratto" del capitolo: qualunque contenuto generato successivamente deve rispettarlo. Se l'Encounter Plan proposto risultasse troppo complesso, troppo semplice o in conflitto con questi vincoli, verrebbe automaticamente corretto.


```
{
  "Chapter 1": {
      "Chapter_act": "Act I",
      "Chapter_name": "The Fading Echoes of the Past",
      "Description": "..."
      "chapter_adapt": {
        "narrative_scope": "Contextualize the events preceding
                           the story (not playable or very light)",
        "beats_required": "soft3",
        "interactivity_level": 0,
        "whammo_required": false,
        "opposition_intensity": 0,
        "encounters_minmax": [
          0,
          1
        ],
        "diegesis_mode": "narrated_past"
      }
    }
```

```
    ... (altri capitoli)
  "Chapter 5": {
      "Chapter_act": "Act II",
      "Chapter_name": "The First Echo of Conflict",
      "chapter_adapt": {
        "narrative_scope": "First significant clash",
        "beats_required": "full7",
        "interactivity_level": 2,
        "whammo_required": true,
        "opposition_intensity": 2,
        "encounters_minmax": [
          1,
          3
        ],
        "diegesis_mode": "present_time"
      }
    }
}
```

Encounter Plan: gameplay del capitolo

Se il Chapter Adapt rappresenta la regola, l'Encounter Plan rappresenta il contenuto. Per ogni capitolo, l'AI deve produrre uno o più encounter, ovvero micro-strutture di gameplay che definiscono ciò che il protagonista fa, vede e affronta momento per momento. Ogni Encounter Plan include:

- obiettivi chiari e verificabili (goals);
- gli spazi o le configurazioni topologiche in cui ha luogo l'azione;
- la tipologia di opposizione o pressione (mobs, trappole, distorsioni);
- un eventuale whammo, se richiesto dal Chapter Adapt;
- i state deltas, cioè gli effetti permanenti sullo stato della storia o dei personaggi.

Gli encounter sono elementi base di gameplay che lo Step 3 espanderà in prosa giocabile. Lo Step 2 non scrive ancora il testo narrativo: genera solo la struttura funzionale dei momenti giocati. Di seguito un esempio di encounter plan

```
{
  "Chapter 5": {
      "Chapter_act": "Act II",
      "Chapter_name": "The First Echo of Conflict",
```

```
      "encounter_plans": [
        {
          "goals": [
            "Defend against the shadowy forces attempting
             to control Kaelen's cursed form",
            "Utilize the Invisible Mirror to deflect or reveal attacks",
            "Escape the immediate conflict zone before succumbing to the curse"
          ],
          "core_verbs": [
            "defend",
            "utilize",
            "escape"
          ],
          "whammo": "A powerful surge of the curse temporarily
                     grants Kaelen immense, uncontrolled power, THEREFORE he must
                     choose between risking innocent lives or suppressing
                     it at great personal cost.",
          "candidate_spaces": [
            "Collapsing Alleyways, Whispering Nexus"
          ],
          "opposition": [
            {
              "mob": "Shadow Weavers",
              "pattern": "corruption/control",
              "hint": "Attempts to amplify Kaelen's curse,
                       vulnerable to reflected energy"
            }
          ],
          "state_deltas": [
            "Kaelen_curse_intensified",
            "Shadow_Weavers_repelled",
            "Kaelen_moral_dilemma_introduced"
          ]
        }
      ],
}
```

Relazione tra NAP, Chapter Adapt e Encounter Plan

L'intero processo segue una catena ben definita:

- Nodo narrativo → Regola NAP: Il sistema identifica a quale regola della NAP il nodo corrisponde e ne eredita i vincoli (ritmo, interattività, whammo, ecc.).

- Nodo + NAP → Chapter Adapt: Se un capitolo deriva da più nodi, le regole vengono aggregate secondo gli aggregation rules della NAP (massimi, minimi, merging, clamping per atto).
- Chapter Adapt → Encounter Plan: Gli encounter vengono generati o corretti in modo da rispettare il numero minimo/massimo previsto, includere o meno il whammo secondo policy, non superare il livello di opposizione ammesso e coprire il beat profile richiesto.
- Encounter Plan → base gameplay per Step 3: Step 3 convertirà encounter e chapter adapt in prosa interattiva, ma senza modificarne la struttura logica.

L'uso combinato di NAP + Chapter Adapt + Encounter Plan permette allo Step 2 di essere un sistema di progettazione narrativa per videogiochi, in cui ogni segmento della storia è calibrato per essere sia coerente con la trama sia adatto alla trasformazione in gameplay.

5.6 Implementazione dello Step 3: Generazione della storia dettagliata

Lo Step 3 costituisce la fase in cui la narrazione assume piena forma: partendo dalla struttura generata nello Step 1 e dal piano dei capitoli prodotto nello Step 2, il sistema elabora la versione estesa della storia, trasformando ogni capitolo in una sequenza coerente di scene, azioni, dialoghi e stati narrativi. A differenza delle fasi precedenti, prevalentemente strutturali, questo step opera a livello testuale e semantico, traducendo ogni *chapter plan* in prosa narrativa giocabile.

5.6.1 Logica implementativa

L'implementazione dello Step 3 è basata su un processo fortemente incrementale. La generazione non avviene in blocco, ma capitolo dopo capitolo: per ogni sezione della storia, il modello riceve il contesto completo prodotto fino a quel momento, così da garantire continuità tematica, coerenza nei personaggi, stabilità del tono e progressione naturale dei conflitti. Ogni iterazione di generazione utilizza un prompt dedicato che integra:

- l'intera struttura narrativa dello Step 1 (tipi di nodi, relazioni, ruoli drammatici);
- il capitolo corrente come definito dallo Step 2 (Description, ChapterAdapt, EncounterPlans);
- la storia completa prodotta nei capitoli precedenti;

- la lista dei capitoli ancora da sviluppare (per mantenere coerenza prospettica);
- i dati semantici provenienti da `Story.json`, `Characters.json` e `Cards.json`.

Lo Step 3 traduce i vincoli strutturali e ludici degli Encounter Plan in scene concrete, facendo emergere azioni verificabili, motivazioni, ostacoli, progressioni di stato e segnali di transizione tra capitoli integrando allo stesso tempo la struttura narrativa definita nello step 1. Dopo la generazione, ogni output viene validato, convertito in JSON e integrato nella storia globale, diventando parte del contesto per la generazione del capitolo successivo.

Generazione incrementale della storia

L'intero flusso di generazione si basa su un ciclo iterativo che consente al modello di operare in modo informato e progressivo. Per ogni capitolo:

1. il sistema prepara il contesto includendo storia generata finora, struttura narrativa complessiva e definizioni semantiche;
2. costruisce un prompt che combina i vincoli ludico-narrativi dello Step 2 con gli elementi strutturati dallo Step 1;
3. invia al modello solo il capitolo corrente da espandere, ma corredato dal contesto necessario per mantenere coerenza (capitoli precedentemente generati e lista di capitoli da generare usati come contesto);
4. integra l'output validato nella storia complessiva, aggiornando lo stato narrativo globale.

Questo approccio incrementale permette al modello di "ricordare" ciò che è già accaduto e di generare coerentemente il capitolo corrente. La generazione procede quindi mantenendo una continuità naturale tra causa ed effetto, in cui ogni evento deriva logicamente dai precedenti; assicura inoltre una coerenza psicologica e comportamentale nei personaggi, evitando cambiamenti improvvisi o non motivati. Allo stesso tempo, il sistema rispetta i vincoli ludici definiti dal *ChapterAdapt*, che influiscono sul ritmo, sull'interattività e sull'intensità degli ostacoli, e garantisce transizioni fluide da un capitolo al successivo, senza salti logici o cambi improvvisi di ambientazione.

5.6.2 Costruzione del prompt

Il prompt dello Step 3 è progettato per trasformare la pianificazione astratta dei capitoli in una narrazione giocabile, mantenendo al tempo stesso un allineamento stretto con la struttura complessiva della storia e con i vincoli ludici definiti negli Step 1 e 2. A differenza dei prompt precedenti, qui il modello deve generare testo narrativo e un set coerente di *encounters* per il singolo capitolo corrente.

1. Ruolo del modello La sezione introduttiva del prompt definisce chiaramente l'identità operativa del modello come *narrative video game writer AI* e gli assegna due compiti distinti ma strettamente collegati: da un lato produrre la prosa narrativa dettagliata e giocabile del capitolo; dall'altro generare un insieme completo di *encounters* che coprano ciò che accade in quel capitolo a livello di gameplay. Il modello è istruito a trattare gli *encounter plans* come basi di gameplay da espandere.

2. Risorse contestuali e uso congiunto delle strutture Come negli step precedenti, il prompt fornisce al modello un insieme articolato di risorse JSON, ma in questo caso l'enfasi è sul loro uso combinato:

- la *narrative seed phrase*, descrizione data in input dall'utente, che ancora il capitolo al tema di fondo della storia;
- le definizioni di base della storia, personaggi e carte, usate per interpretare tipi, temi, archetipi, allineamenti e elementi di worldbuilding;
- la *full narrative story structure*, cioè il grafo completo generato nello Step 1, con nodi, collegamenti e logica di ramificazione;
- il *current chapter* (attuale capitolo da espandere), contenente titolo, nodi utilizzati, breve descrizione, *encounter_plans* e *chapter_adapt*;
- i *previous chapters* (capitoli precedentemente espansi), che fungono da contesto per gli eventi già accaduti;
- i *future chapters* (capitoli non espansi), che servono per assicurare coerenza direzionale.

Il prompt dichiara in modo esplicito che tutte queste fonti devono essere utilizzate insieme.

3. Regole sulla struttura narrativa e integrazione nella storia a tre atti Il prompt richiama la suddivisione della storia nei tre atti (Setup, Confrontation, Resolution) e le regole già definite per i nodi di ciascun atto. Questa parte ricorda al modello che il capitolo corrente deve:

1. rispettare il ruolo dei nodi coinvolti (ad esempio, Setup come introduzione, Confrontation come escalation, Resolution come chiusura);
2. mantenere la *chain of influence*, cioè costruire gli eventi del capitolo sulla base di tutti i nodi precedenti nella catena, non solo sull'ultimo;
3. integrare in modo naturale le carte narrative (luoghi, oggetti, mob, eventi) collegate alla sezione attuale della storia.

4. Continuità capitolo per capitolo e gestione del flusso Una parte corposa del prompt è dedicata alla *chapters continuity*. Qui vengono specificati i vincoli che regolano il passaggio tra capitoli: ogni nuovo capitolo deve "consumare" la chiusura del capitolo precedente, mostrando le conseguenze immediate dell'ultimo evento. Il modello è guidato a:

- evitare salti temporali o informativi non giustificati,
- introdurre nuovi personaggi solo in modo motivato,
- giustificare ogni cambio di luogo con una transizione,
- rispettare il principio di *acquisition-before-use*: un oggetto, una conoscenza o un alleato possono essere utilizzati solo se introdotti in capitoli precedenti o acquisiti nel capitolo corrente.

Il prompt introduce inoltre la nozione di *Carry-In Ledger*: tutti gli elementi narrativi già comparsi (chiavi, ferite, relazioni, condizioni del mondo) devono essere ripresi, consumati o evoluti, evitando oggetti o eventi "dimenticati" e situazioni ridondanti.

5. Uso di ChapterAdapt e Encounter Plans Un blocco centrale del prompt è dedicato al rapporto tra *chapter_adapt* e *encounter_plans*. Il *ChapterAdapt*, calcolato in Step 2 a partire dalla NAP Policy, definisce:

- il profilo di beat richiesti (`soft3`, `soft5`, `full7`),
- la presenza o meno del *whammo* (il colpo di scena di metà capitolo),
- il livello di interattività e l'intensità dell'opposizione,
- il numero minimo e massimo di encounter consentiti,
- la *diegesis_mode* (passato narrato, flashback giocabile, presente).

6. Struttura degli encounters e allineamento con la prosa Il prompt definisce in modo dettagliato la struttura JSON attesa per ciascun encounter (id, obiettivo, spazio, opposizioni, regole, feedback, condizioni di successo/fallimento, *state_delta*, eventuale *whammo*, beat coperti, durata stimata). Viene anche specificato il vincolo di *lockstep* tra narrazione e encounters:

- ogni conseguenza rilevante mostrata nella prosa deve apparire in almeno uno *state_delta* di encounter;
- ogni goal, spazio o opposizione definito negli encounters deve essere reso visibile nella narrazione, evitando che gli encounters sembrino disallineati rispetto al testo del capitolo.

8. Formato di output e integrazione nella pipeline Infine, il prompt specifica in modo rigoroso il formato di output: un singolo oggetto JSON che contiene identificativo del capitolo, atto, nodi utilizzati, titolo, testo narrativo e lista degli encounters.

5.6.3 Beat narrativi e ruolo degli *encounter*

Nel modello implementato, ogni capitolo non è soltanto un blocco di testo, ma una sequenza ritmica di momenti chiave (*beats*) che ne determinano il passo, la tensione e la giocabilità. Questi beat rappresentano le micro–unità che scandiscono l'andamento del capitolo: dall'apertura con un obiettivo chiaro, al primo avanzamento nello spazio di gioco, fino agli ostacoli, al colpo di scena e alla chiusura con una conseguenza leggibile. Il sistema adotta un set canonico di sette beat:

- *Hook&Goal*,
- *Approach*,
- *Complication*,
- *MidGoal+Whammo*,
- *Reversal*,
- *Push/Tradeoff*,
- *StateDelta+Hook*

I beats vengono combinati in tre profili di complessità (`soft3`, `soft5`, `full7`). La NAP Policy associa a ciascun tipo di nodo (PreEvents, Setup, Confrontation, Resolution) un profilo di beat e un livello di interattività, intensità dell'opposizione e numero di encounter attesi; nello Step 2 questi parametri vengono aggregati, generando per ogni capitolo un oggetto `chapter_adapt` con il campo `beats_required`. Lo Step 3 utilizza tale informazione per strutturare sia la prosa sia gli encounter effettivi: il capitolo deve coprire esattamente i beat richiesti dal profilo, senza introdurne di ulteriori.

Ogni *encounter_plan* viene espanso in un encounter completo e in una sezione corrispondente di prosa: i beat richiesti vengono distribuiti tra gli encounter del capitolo (tramite il campo `beats_covered`) in modo che l'unione degli stessi copra esattamente il profilo definito da `beats_required`. In pratica, i beat forniscono la griglia temporale su cui vengono "agganciati" gli encounter: un encounter può aprirsi con Hook&Goal e Approach, un altro può sviluppare Complication e Push/Tradeoff, mentre il beat finale StateDelta+Hook garantisce che il capitolo si chiuda con una trasformazione di stato chiara e un gancio verso il capitolo successivo.

Gli encounter generati nello Step 3 hanno quindi una duplice funzione. Da un lato, costituiscono la specifica ludica del capitolo: ogni encounter è descritto da un identificatore, un obiettivo testabile, uno spazio diegetico (con eventuale topologia e interazioni), un set di verbi d'azione, un modello di opposizione (mob, trappole, pressioni ambientali), regole locali, segnali di feedback, condizioni di fallimento e successo, modifiche allo stato del mondo o dei personaggi, possibili ricompense ed eventuale whammo. Dall'altro lato, questi encounter fungono da vincolo per la prosa: quanto viene descritto nel testo del capitolo deve riflettere le azioni, gli spazi, i rischi e le conseguenze codificati negli encounter. In questo modo il sistema mantiene un allineamento stretto tra livello narrativo e livello ludico. Di seguito un esempio di encounters:


```
"encounters": [
    {
        "encounter_id": "CH04-E1",
        "source_seed_index": 0,
        "goal": "Investigate the source of Kaelen's cursed transformation,
        locate the Invisible Mirror to understand its connection to the time loops,
        and evade the disguised agents of Chronos monitoring Kaelen's movements.",
        "space": {
            "type": "Alchemist's Hidden Study",
            "topology": "cluttered workbench, bookshelves, archways",
            "interactive": [
                "arcane lights",
                "scrolls and ancient texts",
                "cluttered workbench"
            ]
        },
        "verbs": [
            "investigate",
            "locate",
            "evade",
            "perceive",
            "channel"
        ],
        "opposition": [
            {
                "type": "Wizard",
                "pattern": "disguised patrol/surveillance",
                "ai_hint": "Uses illusion magic, avoids direct confrontation,
                observes from shadows."
            }
```

```
],
"rules": [
    "The Invisible Mirror is initially hidden by illusion magic,
    requiring temporal perception to reveal.",
    "Disguised agents of Chronos become visible when Kaelen's
    cursed transformation flares or is channeled."
],
"feedback": [
    "A low hum emanates from the distortion, growing stronger
    as Kaelen nears the Invisible Mirror.",
    "Kaelen's skin prickles and his cursed transformation flares,
    indicating the presence of Chronos's agents.",
    "The disguised agent's illusion shimmers and
    breaks when hit by channeled cursed energy."
],
"fail_states": [
    "Kaelen fails to fully reveal the Invisible Mirror,
    gaining only fragmented insights.",
    "Kaelen is detected by the disguised agent,
    increasing Chronos's awareness of his actions."
],
"success_conditions": [
    "Kaelen fully reveals the Invisible Mirror.",
    "Kaelen understands the true connection between
    his transformation and Chronos's manipulations.",
    "Kaelen successfully evades or disables the disguised agent."
],
"state_delta": [
    "Kaelen_transformation_progress_known",
    "Invisible_Mirror_activated",
    "Chronos_presence_confirmed",
    "Chronos_agents_visibility_increased"
],
"reward": [],
"whammo": "The Invisible Mirror reveals not Kaelen's
reflection, but a distorted image of Chronos manipulating
the cursed clock, THEREFORE Kaelen realizes the architect
of his suffering is closer than he thought.",
"therefore_or_but": "BUT: The Invisible Mirror reveals
a terrifying truth: Chronos is not a distant threat but
the direct architect of Kaelen's cursed transformation,
forcing Kaelen into a direct confrontation with a disguised agent.",
```

```
        "estimated_minutes": 10,
        "beats_covered": [
            "Hook&Goal",
            "Approach",
            "Complication",
            "MidGoal+Whammo",
            "Reversal",
            "Push/Tradeoff",
            "StateDelta+Hook"
        ]
    },
```

5.7 Implementazione dello Step 4: validazione e correzione automatica

5.7.1 Logica implementativa

Lo Step 4 è l'ultima fase della pipeline ed ha il compito di verificare, correggere e convalidare la storia generata negli step precedenti. A differenza delle fasi di generazione, che sono orientate alla produzione di nuovi contenuti narrativi, lo Step 4 è interamente dedicato al controllo di qualità. Il suo obiettivo è ridurre il più possibile incoerenze, rotture di continuità e problemi nell'uso degli elementi narrativi (personaggi, oggetti, luoghi, eventi), intervenendo sui testi già prodotti.

Operativamente, lo Step 4 lavora sulla storia generata dallo Step 3, capitolo per capitolo, e applica due tipi distinti di validazione:

- una validazione di tipo **"label–based"**, che controlla l'uso coerente di personaggi, item, mob, luoghi, temi, etichette e termini narrativi;
- una validazione di tipo **"flow–based"**, che verifica la coerenza del flusso narrativo, delle transizioni tra capitoli, della cronologia e dei collegamenti causa–effetto

Entrambi i moduli seguono uno schema simile: per ogni capitolo analizzano il testo dell'intera storia, individuano problemi significativi, producono un report strutturato in JSON e, quando necessario, generano patch narrative che vengono applicate automaticamente al testo. Il risultato è una versione "corretta" della storia, accompagnata da file di report che documentano gli interventi effettuati.

5.7.2 Label Validator

`step4_label_validator.py` è il modulo incaricato di controllare la coerenza semantica interna di ciascun capitolo. Il suo scopo principale è verificare che gli

elementi narrativi siano utilizzati in modo coerente rispetto a quanto stabilito negli step precedenti lungo il corso della storia.

A livello logico, il Label Validator riceve in input l'intera storia (in formato JSON). Per ogni capitolo:

1. estrae il testo del capitolo corrente e il contesto globale della storia;
2. costruisce un prompt di validazione che include la storia completa, il capitolo da esaminare e un elenco di regole narrative da rispettare;
3. invoca il modello AI con un template che richiede un'analisi strutturata e un output rigorosamente in formato JSON;
4. interpreta il risultato, salvando un report degli eventuali problemi rilevati;
5. se il capitolo è marcato come "needs_fixes", costruisce un secondo prompt che chiede al modello di proporre patch concrete da applicare al testo;
6. applica le correzioni suggerite, aggiornando il JSON della storia.

Il prompt di validazione è progettato per far lavorare il modello su un capitolo alla volta ma con la possibilità di consultare l'intera storia per evitare contraddizioni. Tra le linee guida incluse nel prompt vi sono, ad esempio: evitare che compaiano personaggi non introdotti, impedire l'uso di etichette meta–narrative ("itemCard", "mobCard", nomi di categorie interne) nel testo , assicurarsi che oggetti o elementi introdotti in un capitolo abbiano un effetto o una conseguenza nei capitoli seguenti, controllare che i ruoli dei personaggi non cambino senza motivazione narrativa.

L'output del modello per la fase di validazione segue uno schema JSON del tipo:

```
"chapter_analysis": {
  "chapter_number": "3",
  "chapter_title": "...",
  "status": "pass" | "needs_fixes",
  "label_issues": [
    {
      "issue": "...",
      "evidence": "...",
      "suggested_fix": "..."
    }
  ]
}
```

Se lo stato è “pass”, il capitolo viene considerato valido e il sistema passa al capitolo successivo. Se è “needs_fixes”, il Label Validator costruisce un secondo prompt, che contiene il capitolo, il report dei problemi e istruzioni per generare un nuovo testo corretto. Il modello restituisce un JSON con le patch da applicare, che il modulo inserisce nel capitolo aggiornando il file della storia. In questo modo, lo Step 4 introduce un primo livello di pulizia semantica, riducendo la presenza di etichette improprie o riferimenti incoerenti.

5.7.3 Flow Validator

`step4_flow_validator.py` affronta un problema complementare: la coerenza del flusso narrativo. Il Flow Validator valuta se il capitolo si inserisce correttamente nella linea temporale della storia e nella progressione complessiva. Anche in questo caso, l’input è costituito dalla storia completa e dal capitolo corrente. Il modulo:

- costruisce un prompt che fornisce al modello l’intera storia come contesto e il capitolo da analizzare come focus;
- chiede una valutazione sulla coerenza rispetto ai capitoli precedenti e successivi, sulle relazioni causa–effetto, sulla continuità delle informazioni e sul rispetto della cronologia;
- richiede un output JSON contenente un giudizio complessivo e, in caso di problemi, una lista di “major issues” e relative proposte di correzione;

se il capitolo è marcato come “needs_fixes”, costruisce un secondo prompt che chiede al modello di generare frammenti narrativi integrativi (frasi, brevi paragrafi, transizioni) da inserire in punti specifici del capitolo e poi applica automaticamente le modifiche suggerite, aggiornando il testo del capitolo. Il template del prompt di validazione per il flusso narrativo contiene sezioni dedicate a:

- apertura del capitolo (verifica del “bridge” con il capitolo precedente);
- sviluppo centrale (continuità di obiettivi, conflitti, informazioni);
- chiusura (preparazione del capitolo successivo);
- identificazione di salti temporali non motivati, riaperture di conflitti già chiusi, riutilizzo di eventi senza escalation, decisioni non basate su eventi precedenti.

L’output JSON del Flow Validator è organizzato in modo simile a quello del Label Validator, ma con campi specifici per il flusso logico:

```
{
  "chapter_analysis": {
    "chapter_number": "3",
    "chapter_title": "...",
    "status": "pass" | "needs_fixes",
    "coherence": "...",
    "logical_thread": "...",
    "major_narrative_issues": [
      "..."
    ],
    "fixes_todo": [
      {
        "issue": "...",
        "evidence": "...",
        "where_to_insert": "...",
        "patch_to_apply": "..."
      }
    ]
  }
}
```

Le patch suggerite dal modello vengono poi applicate al testo del capitolo nelle posizioni indicate (apertura, metà, chiusura, dopo un paragrafo specifico), ricostruendo una versione del capitolo che rispetta meglio la progressione narrativa complessiva.

5.7.4 File di report e risultato finale

L'esecuzione congiunta di Label Validator e Flow Validator produce una serie di file di output che documentano lo stato della storia e gli interventi effettuati. Dal punto di vista funzionale, lo Step 4 non garantisce una perfezione assoluta, i limiti dei modelli generativi restano presenti, ma introduce un livello di controllo aggiuntivo che migliora in modo significativo la leggibilità e la stabilità della storia finale.

5.8 Implementazione dello step 5: Generazione sintesi

Una volta completata la generazione della storia (Step 3), il sistema mette a disposizione una funzionalità di sintesi accessibile dal *Pannello Stories*: l'utente seleziona una storia già prodotta e avvia la generazione automatica del *summary*. L'obiettivo

è ricostruire e organizzare in modo analitico ciò che è già presente nel testo finale, trasformandolo in una rappresentazione compatta e strutturata, utile sia per la lettura rapida sia per attività come revisione, pitch, game design.

5.8.1 Logica implementativa e funzionamento

La generazione della sintesi opera nella fase di *post-processing* della storia completa. A differenza degli step precedenti, in cui l'AI costruisce contenuto nuovo rispettando vincoli strutturali e di playability, qui il modello assume un ruolo di analista: riceve in input esclusivamente la storia finale e produce una descrizione gerarchica che ne evidenzia arco narrativo, atti, temi e implicazioni ludiche.

A livello operativo, il flusso è lineare: il sistema recupera la storia selezionata (testo e metadati eventualmente presenti), la inserisce in un prompt dedicato e invoca il modello generativo richiedendo un output in JSON. L'attenzione principale è posta su due aspetti: (i) fedeltà assoluta al testo sorgente, e (ii) completezza, cioè la capacità di non omettere snodi rilevanti (eventi, rivelazioni, climax, chiusura). Il risultato è pensato come una "mappa" della storia.

5.8.2 Costruzione del prompt

Nella parte iniziale il modello viene istruito ad analizzare la storia fornita e a produrre un riassunto seguendo *esattamente* una struttura prefissata, organizzata in cinque macro-sezioni: una sintesi completa dell'intera trama, una ricostruzione in tre atti con riferimento ai capitoli, l'identificazione dei temi principali, la ricostruzione della struttura videoludica implicita e, infine, un'analisi degli elementi di gameplay e design derivabili da storia ed encounter.

Un elemento chiave è la presenza di vincoli espliciti: il modello viene vincolato a non inventare nulla e a considerare la storia come unica fonte di verità. Inoltre, la richiesta di mantenere un tono professionale da *narrative analysis + game design* orienta l'output verso una forma descrittiva e tecnica, più adatta a documentazione e progettazione. Infine, il prompt impone che l'intero risultato sia restituito come file JSON, così da renderlo direttamente utilizzabile a livello applicativo (visualizzazione nel pannello, esportazione o versionamento).

5.8.3 Descrizione output

L'output della generazione della sintesi consiste in un singolo oggetto JSON strutturato, che riflette fedelmente l'organizzazione richiesta dal prompt e separa in modo netto i diversi livelli di analisi narrativa e ludica. Ogni sezione svolge una funzione specifica e può essere utilizzata indipendentemente dalle altre, sia a fini di consultazione sia come supporto al game design.

1. Full Story Summary La prima sezione fornisce una sintesi completa ed esaustiva dell'intera storia. In questa parte vengono condensati i personaggi principali, i loro obiettivi, i conflitti centrali, gli antagonisti, gli eventi chiave, le rivelazioni, il climax e l'epilogo. L'obiettivo è offrire una visione d'insieme che permetta di comprendere l'intero arco narrativo senza dover leggere il testo completo.

```
"full_story_summary": {
  "overview": "Kael, a temporal thief trapped in
              a looping reality, attempts to steal the Godseed
              while confronting the Temporal Warden Xylar...",
  "main_characters": ["Kael", "Jara", "Rix", "Xylar"],
  "ending": "The loop is broken, the world is saved,
            but Kael loses his memories."
}
```

2. Three-Act Summary with Chapters La seconda sezione riorganizza la narrazione secondo la struttura in tre atti. Ogni atto contiene un riassunto complessivo e un elenco dei capitoli che lo compongono, identificati tramite i titoli effettivamente presenti nella storia. Questo consente di mantenere un collegamento diretto con la suddivisione a capitoli prodotta negli step precedenti.

```
"three_act_summary": {
  "act_I": {
    "summary": "The world and the temporal conflict are introduced...",
    "chapters": [
      {
        "title": "The Cursed Clock's Echoes",
        "summary": "Kael relives a fragmented memory tied to a
                    forbidden prophecy."
      }
    ]
  }
}
```

3. Main Themes of the Story La terza sezione identifica e descrive i temi portanti della storia, come dilemmi morali, trasformazioni identitarie o concetti metafisici. Ogni tema è esplicitamente collegato agli eventi narrativi che lo supportano, evitando interpretazioni arbitrarie.

```
"main_themes": [
  {
    "theme": "Sacrifice and Identity",
```

```
    "description": "The protagonist saves the world at the
                    cost of his own memories."
  },
  {
    "theme": "Determinism vs Free Will",
    "description": "The temporal loop challenges the characters'
                    ability to change fate."
  }
]
```

4. General Videogame Structure La quarta sezione traduce la narrazione in una struttura videoludica ad alto livello. Vengono estratti il flusso del gioco, la suddivisione in atti e capitoli, il pacing e il ruolo dei personaggi in termini di gameplay.

```
"videogame_structure": {
  "acts": 3,
  "chapter_based_progression": true,
  "core_pacing": ["exploration", "stealth", "combat"],
  "side_quests": "Integrated within the main acts"
}
```

5. Gameplay & Design Elements L'ultima sezione analizza in dettaglio tutti gli elementi di gameplay derivabili dalla storia e dagli encounter. La struttura è suddivisa in sottosezioni dedicate, come core gameplay loop, combattimento, stati mentali, boss fight e scelte del giocatore.

```
"gameplay_design_elements": {
  "core_gameplay_loop": "Infiltrate, manipulate time,
                         adapt to loop consequences",
  "boss_fights": [
    {
      "name": "Xylar, the Temporal Warden",
      "narrative_role": "Guardian of causality",
      "mechanics": "Temporal distortions and phase-based combat"
    }
  ],
  "fail_states": ["Time expiration", "Mental collapse"]
}
```

5.9 Implementazione UI/UX Unreal Engine

Il front-end dell'applicativo è stato progettato e implementato utilizzando Unreal Engine come ambiente di sviluppo principale, con l'obiettivo di fornire un'interfaccia avanzata per la gestione dell'intero processo di generazione narrativa basato su intelligenza artificiale. A differenza di un'interfaccia web o desktop tradizionale, il sistema è stato concepito come un ambiente editoriale interattivo, capace di avviare la generazione delle storie, supportarne l'analisi, la validazione e l'esplorazione strutturale.

La scelta di Unreal Engine è motivata dalla necessità di gestire interfacce complesse, strutture dati gerarchiche e flussi di lavoro articolati, tipici della progettazione narrativa videoludica. L'intero front-end è stato sviluppato come plugin editor-only, evitando qualsiasi dipendenza dal runtime di gioco e garantendo un'integrazione nativa all'interno dell'editor.

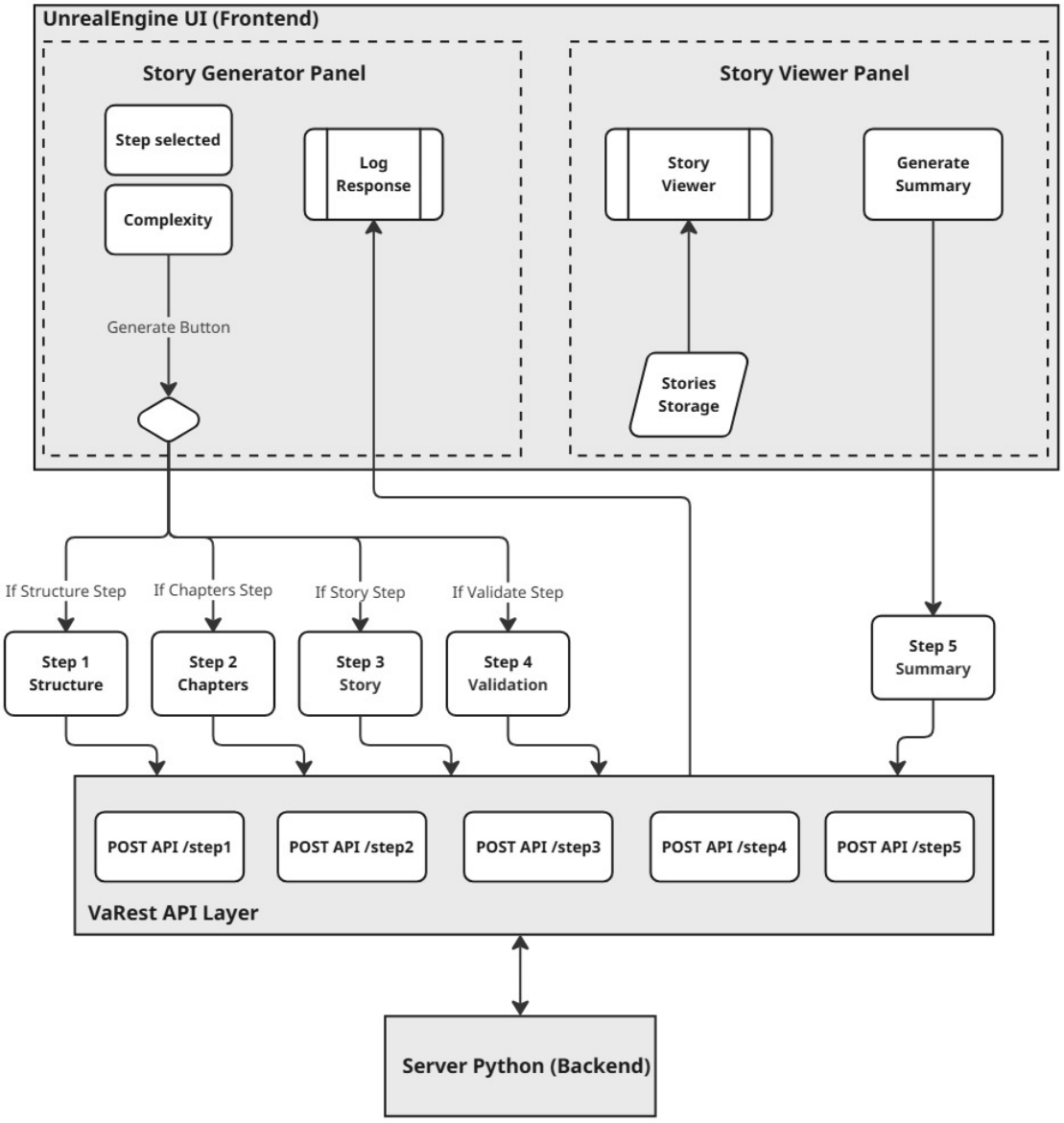

Figura 14: UI/UX architecture

5.9.1 Struttura generale del front-end

L'interfaccia è stata progettata seguendo un approccio modulare, separando chiaramente:

- la **logica di gestione** (gestione degli step, chiamate al server, stati del sistema);
- la **presentazione grafica** (widget, layout, interazioni);
- le **funzionalità di supporto** (lettura/scrittura file, parsing JSON).

Per raggiungere questo obiettivo sono stati utilizzati diversi strumenti messi a disposizione da Unreal Engine:

- **Editor Utility Widgets (EUW)**, per creare finestre personalizzate utilizzabili direttamente all'interno dell'editor;
- **Widget Blueprint (WBP)**, per definire componenti UI riutilizzabili;
- **classi C++**, per estendere le funzionalità native di Unreal e colmare alcune limitazioni dei Blueprint;
- un **plugin editor-only**, che incapsula l'intero front-end rendendolo portabile e facilmente distribuibile.

5.9.2 Editor Utility Widget principale

Il cuore del front-end è rappresentato da un **Editor Utility Widget principale**, che funge da punto di accesso a tutte le funzionalità del tool. Questo widget è stato progettato come una finestra persistente dell'editor e viene aperto tramite una voce dedicata nel menu di Unreal.

Dal punto di vista implementativo, l'Editor Utility Widget svolge principalmente il ruolo di **controller**:

- raccoglie gli input dell'utente;
- gestisce lo stato globale dell'applicazione;
- coordina i vari widget secondari;
- invia richieste HTTP al backend per avviare o monitorare gli step di generazione.

5.9.3 Interfaccia di generazione della storia

All'interno dell'interfaccia principale è presente una sezione dedicata alla **configurazione della generazione narrativa**. Questa sezione è stata costruita interamente tramite **Widget Blueprint**, come mettono a disposizione diversi input per l'utente. L'utente può inserire una descrizione testuale che rappresenta l'idea narrativa iniziale. Questo input viene utilizzato come seed narrativo, ovvero come punto di partenza semantico per la generazione della struttura della storia.

Accanto al seed narrativo, l'utente può selezionare un valore di complessità, che influisce direttamente sulla profondità e sul livello di dettaglio della storia generata. Dal punto di vista implementativo, questo parametro viene semplicemente raccolto dall'interfaccia e inviato al backend, dove viene utilizzato per modulare il comportamento del modello di intelligenza artificiale.

È inoltre possibile selezionare lo step di generazione, scegliendo se generare solamente la struttura, capitoli, la storia o includere anche la fase di validazione. Di seguito nel dettaglio i vari input:

- nome storia;
- descrizione storia (seed narrativo);
- menu a tendina per la scelta dello step di generazione;
- slider per livello di complessità.

Dal punto di vista logico, i Blueprint raccolgono i parametri selezionati dall'utente per poi costruire la richiesta HTTP strutturata ed inviarla al backend. Il Frontend avvia un sistema di pooling per verificare lo stato di avanzamento delle richieste (job lato backend):

La comunicazione tra il front-end e il backend avviene tramite richieste HTTP asincrone. Quando l'utente avvia un processo di generazione, il sistema invia una richiesta iniziale che restituisce un identificativo univoco del job. A partire da questo momento, il front-end entra in una fase di monitoraggio, interrogando periodicamente il backend per verificarne lo stato di avanzamento.

Questo meccanismo è stato progettato per gestire processi potenzialmente molto lunghi, evitando il blocco dell'interfaccia e fornendo all'utente un feedback continuo (onde evitare timeout). Particolare attenzione è stata dedicata alla gestione del ciclo di vita del widget, in modo da interrompere correttamente il polling e notificare il backend in caso di chiusura dell'interfaccia.

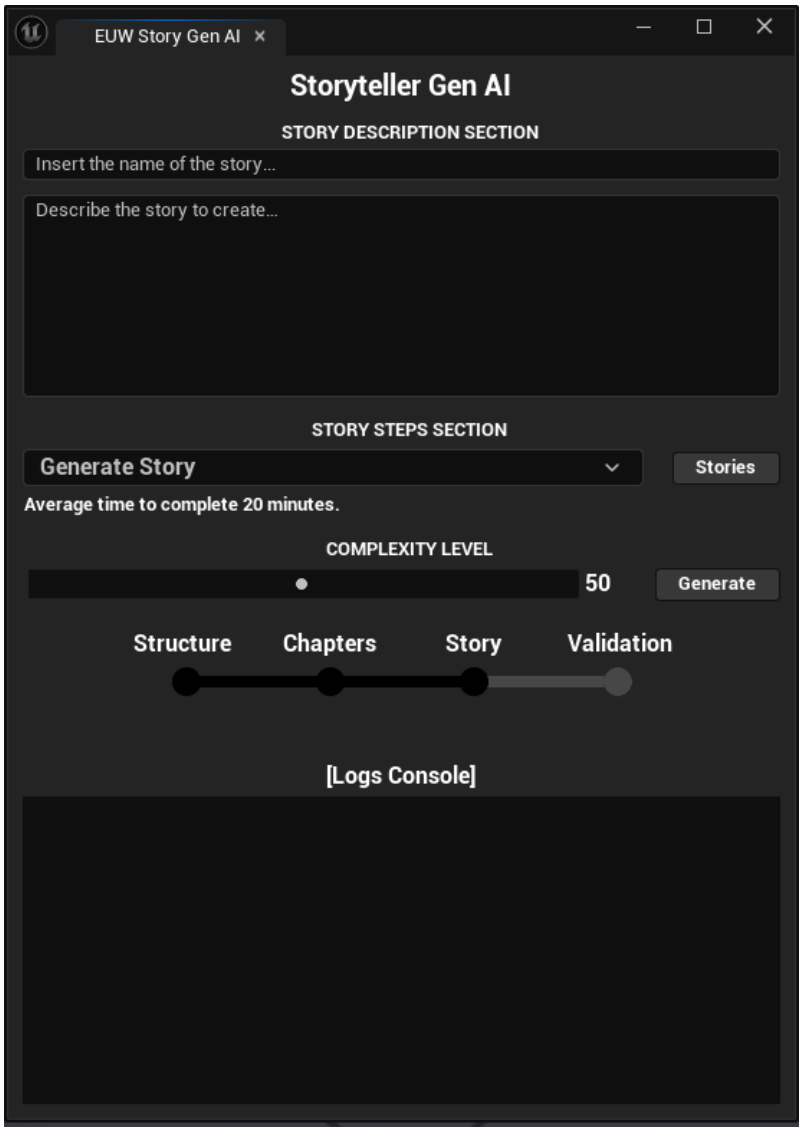

Figura 15: Story Gen UI

5.9.4 Pannello *Stories*

Il pannello Stories rappresenta l'area dell'interfaccia dedicata all'esplorazione dei risultati generati. In questo pannello vengono visualizzate tutte le storie prodotte dal sistema, organizzate in una struttura gerarchica che riflette l'organizzazione dei file sul filesystem.

Ogni storia può essere espansa per visualizzarne i contenuti interni, come capitoli, summary e file di validazione. Questa rappresentazione gerarchica consente all'utente di comprendere immediatamente le relazioni tra i diversi elementi narrativi e di accedere rapidamente alle informazioni di interesse.

A livello implementativo il pannello è stato progettato come di seguito:

- un **Widget Blueprint principale**, responsabile della struttura generale;
- una serie di **widget entry**, utilizzati per rappresentare i singoli elementi storie;
- logica C++ di supporto per la gestione dei file.

Ogni storia generata viene salvata su disco in una cartella dedicata, contenente i diversi file JSON prodotti durante gli step (struttura, capitoli, summary, validazioni). Il pannello Stories esegue una scansione di queste cartelle e costruisce una **vista gerarchica**, permettendo all'utente di distinguere le singole storie , espandere o comprimere le varie cartelle e selezionare i vari file specifici di ogni storia per l'analisi

5.9.5 Widget di riga e interazione

Ogni elemento visualizzato nel pannello Stories è rappresentato da un Widget Blueprint dedicato, responsabile della gestione dell'interazione utente. Il widget di riga controlla lo stato di espansione e compressione dei nodi, aggiornando dinamicamente l'interfaccia e le icone associate. La logica di visualizzazione è basata su uno stato interno che determina la visibilità dei figli e il tipo di icona mostrata.

Le operazioni più complesse, come l'apertura di cartelle nel filesystem o la lettura dei file JSON, non vengono eseguite direttamente all'interno del widget Blueprint, ma delegate a funzioni implementate in C++ ed esposte ai Blueprint. Questa scelta permette di mantenere i widget leggeri e focalizzati esclusivamente sull'interazione.

5.9.6 Visualizzazione dei JSON

Per l'analisi dei contenuti generati dal backend è stato integrato un visualizzatore JSON. I file prodotti presentano strutture annidate e complesse, che rendono inefficace una semplice visualizzazione testuale. Il visualizzatore consente invece di esplorare il JSON in maniera strutturata, evidenziando chiavi, valori e tipologie di dato attraverso una codifica cromatica.

Dal punto di vista tecnico, il JSON viene letto da file tramite C++, convertito in una struttura gerarchica e visualizzato mediante widget Slate personalizzati, tramite apposito script C++. Questa soluzione consente un'analisi approfondita dei capitoli, delle connessioni narrative e dei summary generati.

5.9.7 Gestione del ciclo di vita e delle operazioni asincrone

Poiché alcuni step di generazione possono richiedere diversi minuti (o ore), il frontend è stato progettato per inviare richieste non bloccanti e interrompere correttamente le richieste nel momento in cui il widget viene chiuso.

Quando l'utente chiude la finestra del tool o preme sull'apposito pulsante di stop, vengono:

- fermati i timer di polling;
- inviati eventuali segnali di cancellazione al backend;
- rilasciate le risorse del widget.

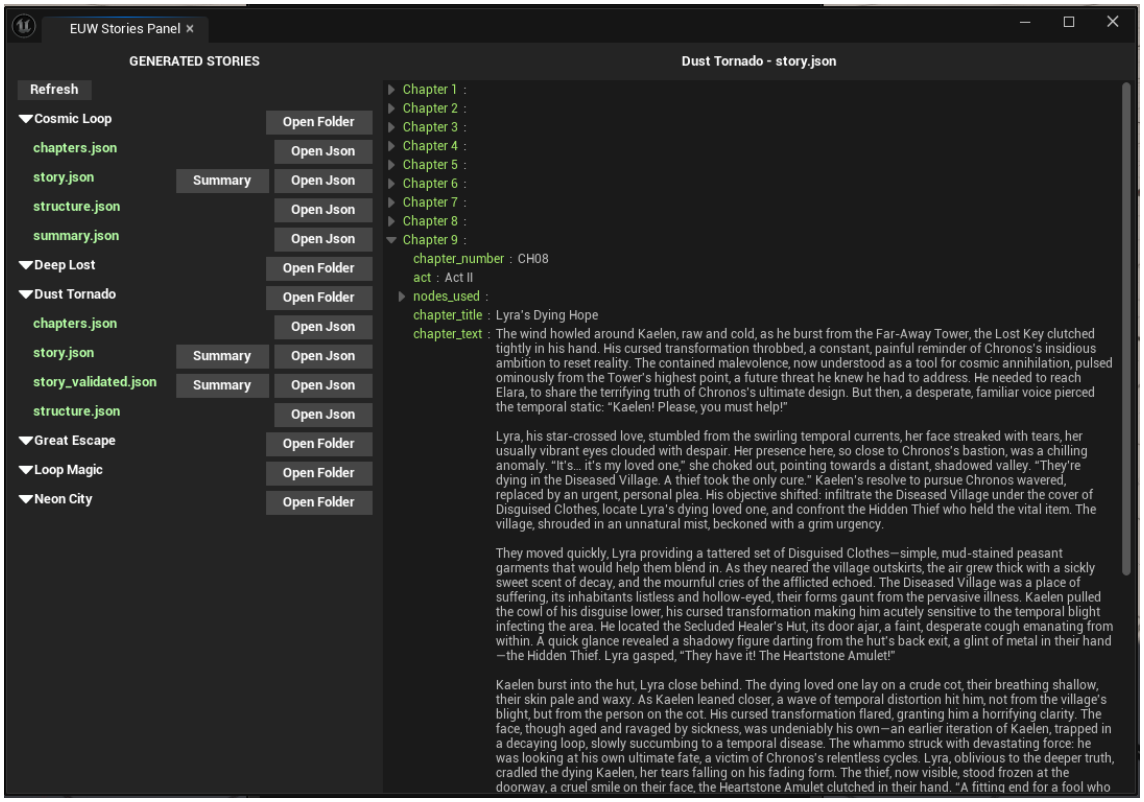

Figura 16: Story Manage UI

Capitolo 6

Analisi dei feedback e valutazione sperimentale del sistema

6.1 Introduzione alla fase di valutazione

Per valutare l'efficacia del sistema di generazione automatica di storie videoludiche è stata condotta una fase di testing qualitativo e quantitativo, coinvolgendo utenti con differenti livelli di esperienza nel settore videoludico e nella scrittura narrativa. L'obiettivo principale di questa fase non era misurare esclusivamente la qualità letteraria delle storie generate, ma comprendere il valore reale dello strumento come *supporto al processo creativo*, in relazione agli obiettivi descritti nei capitoli precedenti.

I feedback sono stati raccolti tramite un questionario strutturato, composto da sezioni dedicate a:

- profilo ed esperienza degli utenti;
- usabilità dell'interfaccia;
- qualità percepita delle storie generate;
- affidabilità dello strumento come supporto creativo;
- integrazione del sistema in una pipeline reale di sviluppo.

I risultati presentati in questo capitolo costituiscono una validazione empirica delle scelte architetturali, metodologiche e progettuali adottate nel sistema.

6.2 Profilo dei partecipanti

Il campione coinvolto nella sperimentazione presenta una composizione eterogenea, comprendente:

- game developer e aspiranti sviluppatori;
- utenti con esperienza nell'uso di strumenti di intelligenza artificiale per la scrittura;
- gamer con interesse per la narrazione interattiva;
- utenti con livelli diversi di esperienza nella scrittura di storie per videogiochi.

Questa varietà ha permesso di raccogliere feedback da prospettive differenti, sia tecniche sia creative, rendendo l'analisi più rappresentativa degli scenari di utilizzo reali del sistema. I dati mostrano una prevalenza di utenti con esperienza medio-bassa nella scrittura narrativa videoludica, aspetto particolarmente rilevante in quanto il sistema è stato progettato anche come strumento di supporto per utenti non esperti.

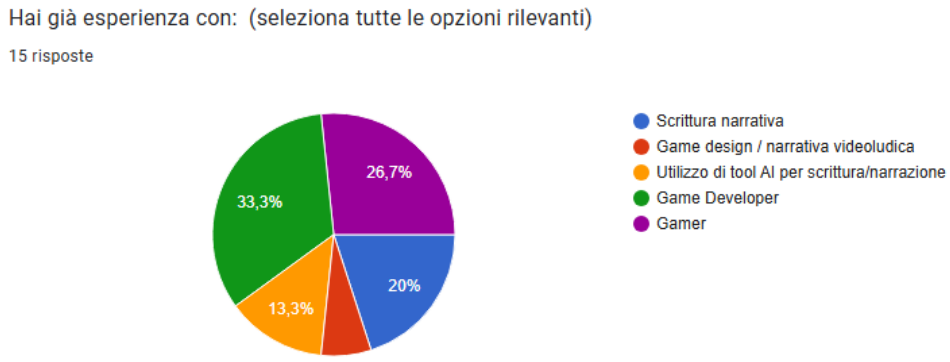

Figura 17: Personal data for testing

6.3 Usabilità dell'interfaccia e flusso di utilizzo

Una prima area di analisi riguarda la facilità di utilizzo del sistema e la chiarezza dell'interfaccia sviluppata in Unreal Engine. I feedback raccolti indicano una valutazione complessivamente positiva su aspetti quali:

- semplicità di configurazione dei prompt e dei parametri di generazione;
- comprensibilità della suddivisione in step della pipeline narrativa;
- chiarezza dei feedback visivi forniti durante l'esecuzione;
- facilità di navigazione tra storie, capitoli e file di output.

Gli utenti hanno apprezzato in particolare la distinzione esplicita tra i diversi step del processo (struttura, capitoli, storia, validazione), che riflette fedelmente la logica descritta nei capitoli di implementazione. Questa separazione ha contribuito a rendere il sistema percepito come controllabile e trasparente, riducendo l'effetto di "scatola nera" tipico di molti strumenti basati su AI.

Alcuni feedback suggeriscono possibili miglioramenti, principalmente legati a:

- una maggiore evidenziazione dello stato dei job in esecuzione;
- un'organizzazione ancora più gerarchica dei contenuti narrativi;
- strumenti aggiuntivi di confronto tra versioni della storia.

Tali osservazioni risultano coerenti con la complessità crescente dei contenuti generati e rappresentano indicazioni utili per sviluppi futuri del front-end.

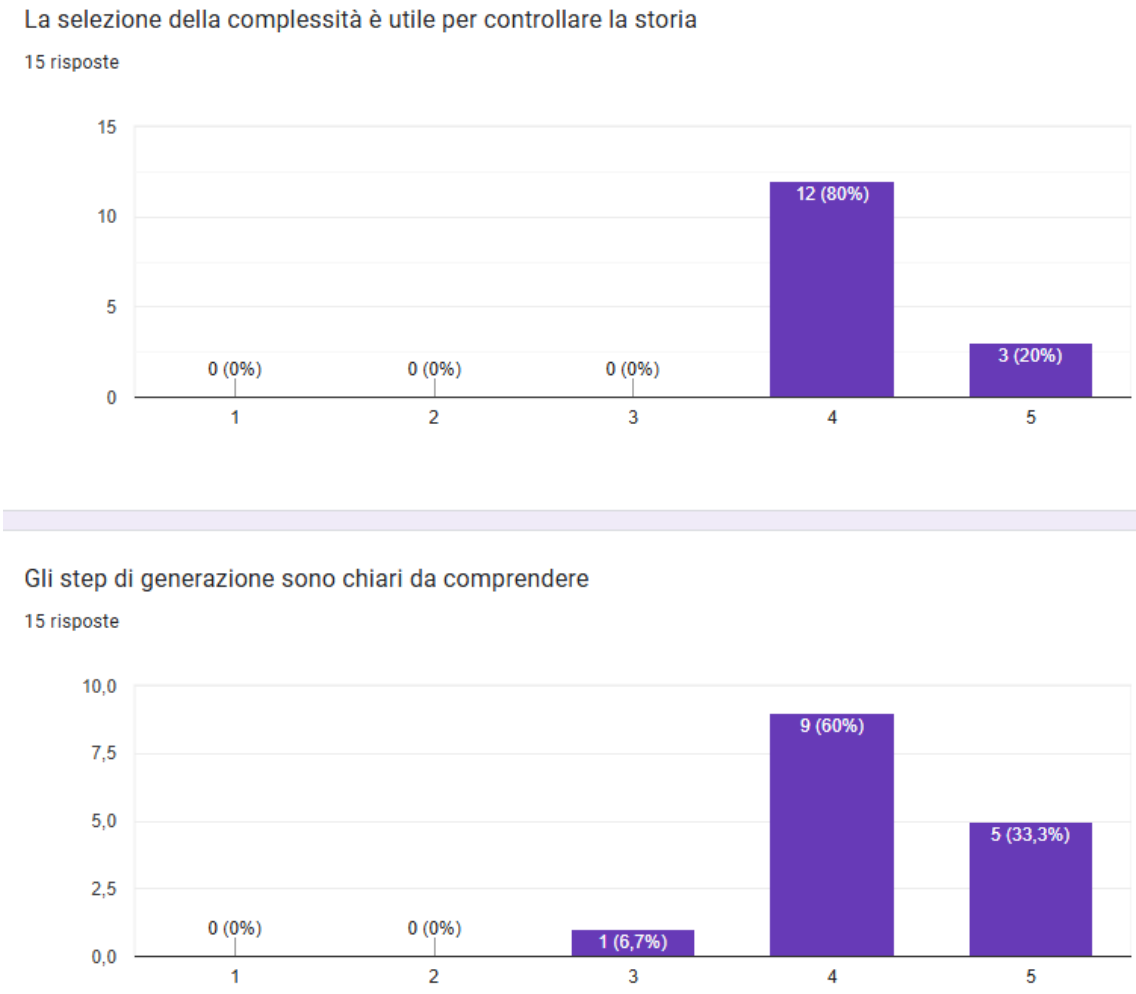

Figura 18: Interface usability

6.4 Qualità percepita delle storie generate

Un aspetto centrale dell'analisi riguarda la qualità narrativa delle storie prodotte dal sistema. I partecipanti hanno valutato positivamente la capacità del sistema di:

- generare strutture narrative solide;
- mantenere una progressione logica tra capitoli;
- costruire archi narrativi comprensibili;
- fornire una base narrativa coerente su cui lavorare.

È emerso chiaramente che la storia generata non viene percepita come un prodotto finale pronto per la pubblicazione, ma come una *bozza narrativa avanzata*, utile soprattutto nelle fasi di brainstorming, pre-produzione e prototipazione. Questo risultato è in linea con gli obiettivi dichiarati del sistema, che non mira a sostituire il lavoro dell'autore umano, ma ad accelerarne e supportarne il processo creativo.

Alcuni utenti hanno segnalato una tendenza a pattern narrativi ricorrenti, in particolare nei capitoli intermedi o nei finali, evidenziando un limite intrinseco dei modelli linguistici utilizzati. Tuttavia, tali criticità sono state spesso mitigate quando la struttura iniziale risultava sufficientemente dettagliata, confermando l'importanza della pipeline multi-step adottata.

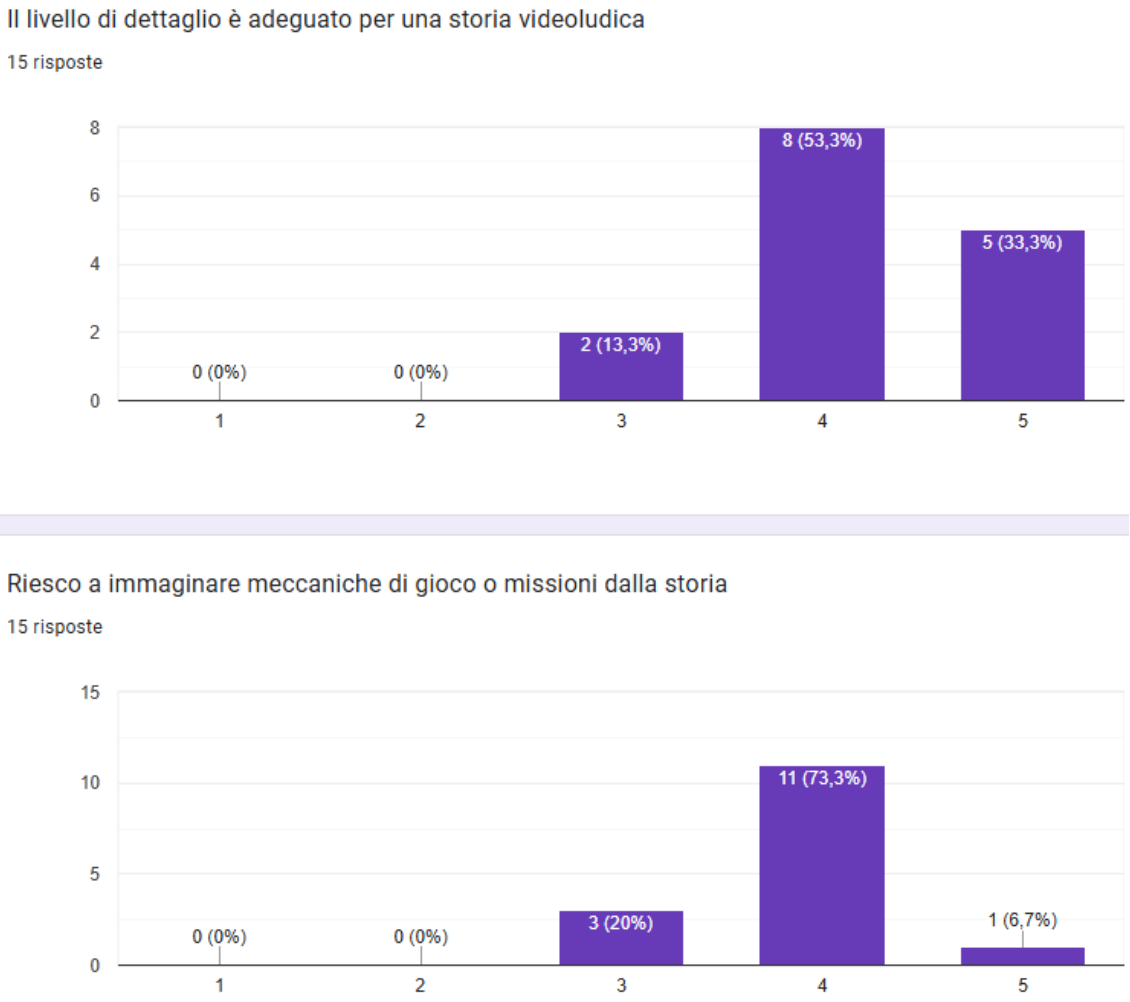

Figura 19: Story quality

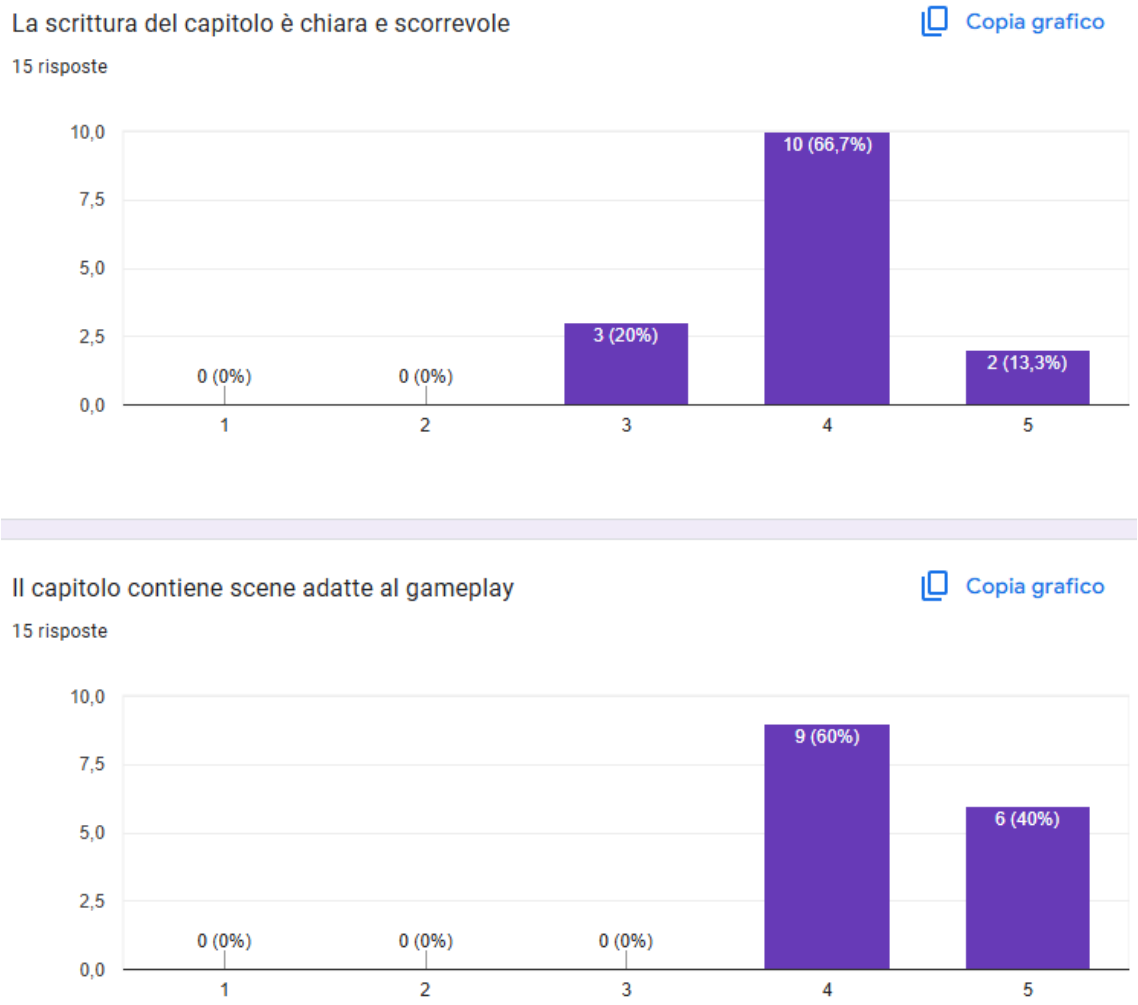

Figura 20: Story gameplay flow

6.5 Affidabilità dello strumento come supporto creativo

Un indicatore particolarmente significativo riguarda il livello di fiducia degli utenti nel sistema come generatore di bozze narrative. La maggior parte dei partecipanti ha espresso una fiducia medio-alta nell'utilizzo dello strumento per:

- esplorare idee narrative alternative;
- strutturare rapidamente una trama;
- individuare potenziali sviluppi narrativi;
- supportare la scrittura collaborativa.

La fiducia risulta invece più cauta nel considerare il sistema come unica fonte narrativa, sottolineando ancora una volta il ruolo dell'AI come *co-autore* e non come autore autonomo. Questo dato conferma la validità dell'approccio ibrido adottato, che combina struttura formale, validazione automatica e generazione linguistica controllata.

Figura 21: Creative support usability

6.6 Integrazione in una pipeline reale di sviluppo

Un'ulteriore sezione del questionario ha identificato la possibilità di utilizzo del sistema in una pipeline reale di sviluppo videoludico. Molti utenti hanno dichiarato che utilizzerebbero il sistema in contesti reali, in particolare:

- nella fase di ideazione iniziale;
- durante la definizione della struttura narrativa;
- come strumento di supporto per narrative designer in fasi iniziali;
- per la verifica preliminare della coerenza di una storia complessa.

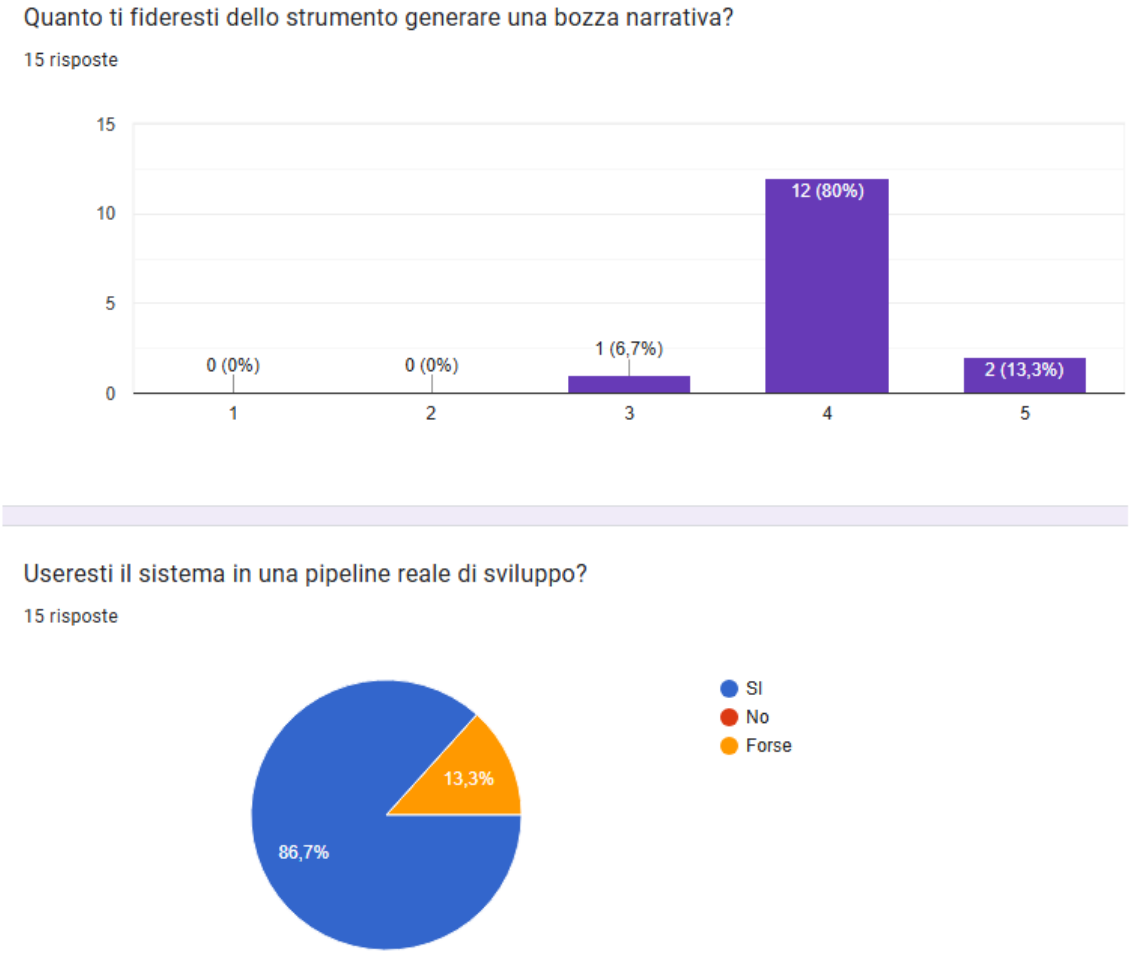

Figura 22: Usability in working environment

Coloro che hanno espresso incertezza o riserve hanno indicato come possibili miglioramenti:

- una maggiore personalizzazione dello stile narrativo;
- un controllo più granulare sui vincoli di generazione;
- strumenti avanzati di editing e revisione.

Questi feedback rafforzano l'idea che il sistema possa trovare una collocazione concreta nel workflow produttivo, soprattutto se integrato con strumenti già esistenti e arricchito da funzionalità di editing avanzato.

6.7 Criticità emerse e limiti del sistema

I partecipanti hanno segnalato alcune criticità, riconducibili principalmente a:

- ambiguità nei prompt iniziali;
- difficoltà del modello nel mantenere coerenza su catene narrative molto lunghe;
- descrizione talvolta superficiale di elementi secondari;
- limiti computazionali dei modelli AI utilizzati.

È importante sottolineare come molte di queste criticità siano coerenti con i limiti attualmente noti dei modelli linguistici di grandi dimensioni. Diversi utenti hanno osservato che l'introduzione dei moduli di validazione migliora sensibilmente la qualità complessiva, suggerendo che l'approccio multi-step con controllo automatico rappresenti una direzione promettente.

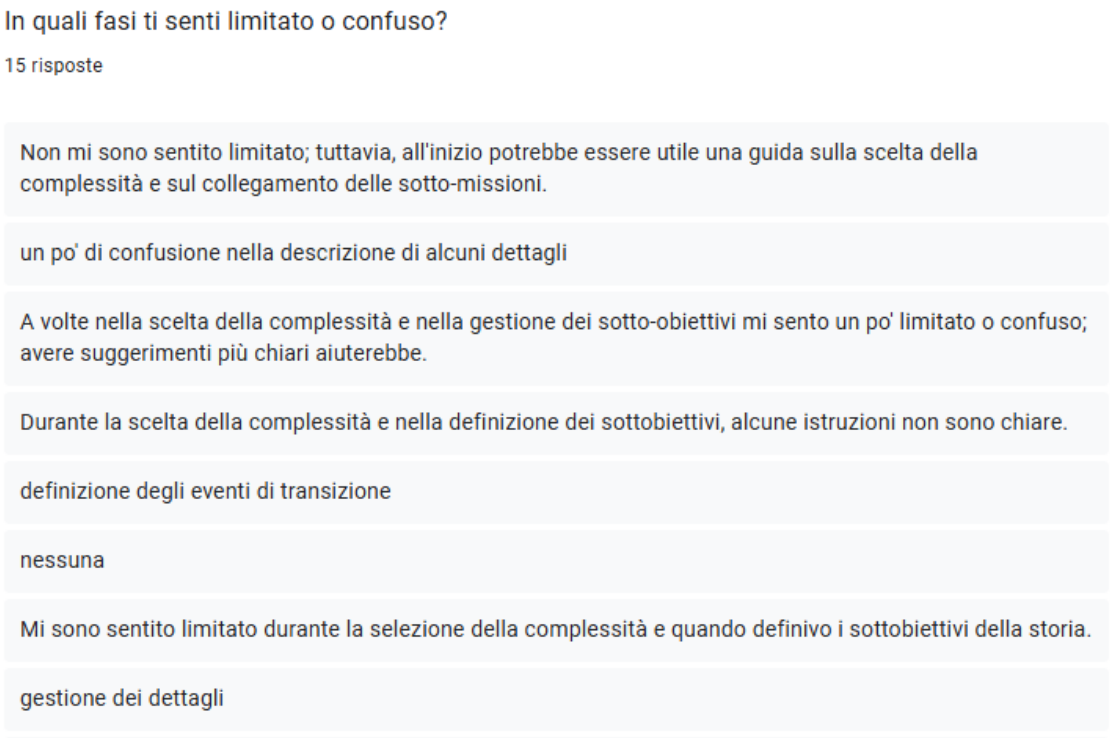

Figura 23: Feedback users

6.8 Sintesi dei risultati

Nel complesso, i feedback raccolti confermano che il sistema:

- fornisce un supporto concreto al processo creativo;
- migliora la gestione della complessità narrativa;
- favorisce la prototipazione rapida di storie videoludiche;
- risulta comprensibile e utilizzabile anche da utenti non esperti.

La sperimentazione dimostra che l'approccio adottato, basato su una pipeline narrativa strutturata e sull'uso controllato dell'intelligenza artificiale generativa, rappresenta una soluzione efficace e praticabile per la progettazione narrativa nei videogiochi moderni.

6.9 Discussione critica dei risultati

L'analisi dei feedback raccolti consente di trarre una serie di considerazioni critiche sull'efficacia del sistema, sui suoi limiti strutturali e sulle potenzialità future. I risultati ottenuti mostrano una sostanziale coerenza tra gli obiettivi progettuali iniziali e l'esperienza d'uso percepita dagli utenti, confermando la validità dell'approccio metodologico adottato.

Un primo elemento di rilievo riguarda la scelta di strutturare il processo di generazione narrativa come una pipeline multi-step. I feedback indicano chiaramente che la suddivisione in fasi distinte (struttura, capitoli, storia dettagliata, validazione) non solo migliora la qualità complessiva della narrazione, ma contribuisce anche a rendere il sistema più comprensibile e controllabile. Questo aspetto risulta particolarmente importante in un contesto in cui i modelli di intelligenza artificiale tendono a essere percepiti come "black box". La possibilità di intervenire su ogni livello della generazione riduce tale opacità e rafforza la fiducia dell'utente nel sistema.

Dal punto di vista narrativo, i risultati confermano che i modelli AI, se guidati da strutture formali e vincoli espliciti, sono in grado di produrre contenuti coerenti e funzionali allo scopo. Tuttavia, emerge con chiarezza che la qualità della storia dipende fortemente dalla solidità della struttura iniziale. Le storie che partono da una struttura ben definita risultano più consistenti, mentre strutture vaghe o eccessivamente generiche tendono a produrre narrazioni più ripetitive o meno caratterizzate. Questo dato rafforza l'idea che il valore del sistema risieda nell'integrazione tra controllo umano e generazione automatica, piuttosto che in una delega totale alla macchina.

Un altro aspetto critico riguarda il ruolo del sistema come strumento di supporto creativo. I feedback indicano che il sistema viene percepito come particolarmente

utile nelle fasi di brainstorming, prototipazione e pre-produzione, mentre viene considerato meno adatto come generatore di contenuti finali pronti per l'uso diretto. Questa distinzione è significativa, poiché conferma che l'AI non sostituisce il lavoro del narrative designer, ma ne amplia le capacità, riducendo il tempo necessario per esplorare alternative narrative e strutturare una storia complessa.

Dal punto di vista dell'interfaccia e dell'esperienza utente, l'integrazione con Unreal Engine si è rivelata una scelta efficace. La possibilità di visualizzare storie, capitoli e strutture narrative direttamente all'interno di un ambiente familiare per il game designer migliora l'accessibilità dello strumento e ne favorisce l'adozione in contesti reali. Tuttavia, alcuni feedback suggeriscono che l'aumento della complessità narrativa richiede strumenti di visualizzazione e navigazione ancora più avanzati, evidenziando una naturale tensione tra semplicità dell'interfaccia e ricchezza dei contenuti.

Per quanto riguarda i limiti del sistema, le criticità emerse sono in gran parte riconducibili alle limitazioni attuali dei modelli linguistici di grandi dimensioni. In particolare, la difficoltà nel mantenere coerenza su archi narrativi molto lunghi e la tendenza a riutilizzare schemi narrativi ricorrenti rappresentano problemi noti nella letteratura sull'AI generativa. L'introduzione dei moduli di validazione (Label Validator e Flow Validator) ha dimostrato di poter mitigare parzialmente tali problemi, suggerendo che un'ulteriore evoluzione del sistema dovrebbe puntare su meccanismi di controllo e revisione sempre più sofisticati.

In sintesi, la fase di valutazione conferma che il sistema sviluppato raggiunge gli obiettivi prefissati, offrendo uno strumento concreto, flessibile e utilizzabile per la progettazione narrativa videoludica. I feedback raccolti forniscono anche indicazioni preziose per future evoluzioni, evidenziando come la combinazione tra strutture narrative formali e intelligenza artificiale generativa rappresenti una direzione promettente per il futuro dello storytelling interattivo.

Capitolo 7
Conclusioni finali e sviluppi futuri

7.1 Conclusioni finali

Il lavoro presentato in questa tesi ha affrontato il tema della generazione automatica di storie videoludiche attraverso l'utilizzo di modelli di intelligenza artificiale generativa, con l'obiettivo di progettare e realizzare un sistema capace di supportare il lavoro di game designer e narrative designer nelle fasi di ideazione, strutturazione e analisi della narrativa.

A partire da un'analisi dello stato dell'arte sullo storytelling interattivo e sulle metodologie narrative adottate nell'industria videoludica, il progetto ha messo in evidenza una problematica centrale: la crescente complessità delle storie moderne rende sempre più difficile mantenere coerenza, controllo e varietà narrativa lungo archi di gioco estesi. In questo contesto, l'intelligenza artificiale non è stata concepita come un sostituto dell'autore umano, ma come uno strumento di supporto capace di amplificarne le capacità creative e di ridurre il carico operativo nelle fasi più ripetitive o strutturali del processo.

Il sistema sviluppato si fonda su un approccio ibrido che combina strutture narrative formali con la flessibilità dei modelli linguistici di grandi dimensioni. La scelta di suddividere la generazione in una pipeline multi-step si è rivelata determinante per garantire controllo, trasparenza e qualità del risultato finale. Ogni fase della pipeline affronta un livello differente della narrazione, permettendo di gestire separatamente struttura, capitoli, testo narrativo e validazione, riducendo la propagazione degli errori e migliorando la coerenza complessiva.

Dal punto di vista implementativo, l'adozione di un'architettura client–server modulare ha consentito di separare chiaramente la logica computazionale dalla visualizzazione e dall'interazione con l'utente. L'integrazione con Unreal Engine ha dimostrato come il sistema possa essere inserito efficacemente in un contesto di sviluppo videoludico reale, fornendo strumenti di analisi e ispezione direttamente all'interno di un ambiente familiare per i designer.

I feedback raccolti durante la fase di sperimentazione confermano la validità dell'approccio adottato. Gli utenti hanno riconosciuto nel sistema un valido strumento di supporto creativo, particolarmente efficace nelle fasi di brainstorming, prototipazione e definizione preliminare della narrativa. Allo stesso tempo, sono emersi limiti legati alla gestione di archi narrativi molto complessi e alla tendenza dei modelli generativi a riproporre pattern ricorrenti, criticità che riflettono limiti noti delle tecnologie attuali.

Nel complesso, il progetto dimostra che la combinazione tra strutture narrative controllate e intelligenza artificiale generativa rappresenta una direzione concreta e promettente per lo sviluppo di strumenti a supporto dello storytelling videoludico. Il sistema non intende automatizzare completamente la scrittura, ma offrire un ambiente di lavoro ibrido in cui autore e macchina cooperano nella costruzione di esperienze narrative più ricche, flessibili e adattabili.

7.2 Sviluppi futuri

Sebbene il sistema sviluppato raggiunga gli obiettivi prefissati, numerose possibilità di estensione e miglioramento emergono naturalmente dall'analisi dei risultati e dei feedback raccolti.

Un primo ambito di sviluppo riguarda il potenziamento dei meccanismi di validazione. L'introduzione di moduli più avanzati per il controllo della coerenza temporale, delle relazioni e dell'evoluzione dei personaggi potrebbe migliorare ulteriormente l'affidabilità delle storie generate, soprattutto in contesti narrativi molto estesi o fortemente ramificati.

Un secondo possibile sviluppo concerne l'integrazione di modelli linguistici più avanzati o specializzati, in grado di mantenere una memoria narrativa più profonda e di gestire archi di lunga durata con maggiore precisione. L'adozione di modelli futuri con capacità contestuali estese potrebbe ridurre significativamente alcune delle criticità riscontrate durante la sperimentazione.

Dal punto di vista funzionale, il sistema potrebbe essere esteso per supportare ulteriori forme di contenuto narrativo, come dialoghi ramificati, sistemi di scelte morali più complesse, eventi dinamici o strutture procedurali adattive basate sul comportamento del giocatore. Queste estensioni permetterebbero di avvicinare ulteriormente il sistema alle esigenze di produzioni videoludiche su larga scala.

Un ulteriore sviluppo interessante riguarda l'integrazione diretta con prototipi di gameplay. Collegare la generazione narrativa a sistemi ludici simulati o a prototipi interattivi consentirebbe di valutare non solo la coerenza della storia, ma anche il suo impatto sull'esperienza di gioco, rafforzando il legame tra narrativa e gameplay.

Infine, il sistema potrebbe evolversi in una piattaforma collaborativa, in cui più utenti possano lavorare simultaneamente sulla stessa storia, confrontando varianti narrative, annotando modifiche e iterando collettivamente sul contenuto generato.

In conclusione, questo lavoro rappresenta un primo passo verso la costruzione di strumenti intelligenti per lo storytelling videoludico, capaci di coniugare rigore strutturale e creatività generativa. L'evoluzione delle tecnologie di intelligenza artificiale, unita a una progettazione narrativa consapevole, apre prospettive significative per il futuro del game design e della narrazione interattiva.

Capitolo 8
Appendice

Bibliografia

[1] Robert Denton Bryant and Keith Giglio. *Slay the Dragon: Writing Great Video Games*. St. Martin's Press, 2015.

[2] Janet H. Murray. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. MIT Press, 1997.

[3] Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference*, 2003.

[4] Marco Guarneri. Ghost: A ghost story writer. Master's thesis, Università degli Studi di Milano, 2016.

[5] William Wallace Cook. *Plotto: The Master Book of All Plots*. Ellis Publishing Company, 1928.

[6] TV Tropes. Periodic table of storytelling. `https://tvtropes.org`, 9999.

[7] Roy T. Fielding. Architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California, Irvine*, 2000.

[8] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 3rd edition, 2012.

[9] Tom B. et al. Brown. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[10] Joseph Campbell. *The Hero with a Thousand Faces*. Pantheon Books, New York, 1949.

[11] Aristotle. Poetics (translated by s. h. butcher). Online text, 1895.

[12] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, Austin, 2 edition, 1968. Translated by Laurence Scott; revised and edited by Louis A. Wagner.

[13] Henry Jenkins. Game design as narrative architecture. In Noah Wardrip-Fruin and Pat Harrigan, editors, *First Person: New Media as Story, Performance, and Game*. MIT Press, Cambridge, MA, 2004.

[14] Gonzalo Frasca. Ludologists love stories, too: Notes from a debate that never took place. In Mark J. P. Wolf and Bernard Perron, editors, *The Video Game Theory Reader*. Routledge, New York, 2003.

[15] Jesse Schell. *The Art of Game Design: A Book of Lenses*. A K Peters/CRC Press, Boca Raton, FL, 3 edition, 2019.

[16] Nevigo. articy:draft documentation / product overview. Online documentation, 2025.

[17] Python Software Foundation. Python 3 documentation. Online documentation, 2025.

[18] Epic Games. Unreal engine documentation. Online documentation, 2025.

[19] Google. Gemini model documentation. Online documentation, 2025.

[20] Epic Games. Unreal motion graphics (umg) ui designer documentation. Online documentation, 2025.

[21] ufna. Varest plugin for unreal engine. Project page / repository, 2025.