

# QuestMaster AI - Relazione di Progetto

Umberto Domenico Ciccia, Salvatore Chiricosta, Giovanni Ferraro

Corso: Intelligenza Artificiale  
Anno Accademico: 2024/2025

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
<b>2</b>	<b>Obiettivi del Progetto</b>	<b>4</b>
2.1	Obiettivi Primari . . . . .	4
2.2	Obiettivi Secondari . . . . .	4
<b>3</b>	<b>Architettura del Sistema</b>	<b>5</b>
3.1	Panoramica Architetturale . . . . .	5
3.2	Componenti Principali . . . . .	5
3.2.1	<b>Core Components</b> . . . . .	5
3.2.2	<b>Services Layer</b> . . . . .	6
<b>4</b>	<b>Tecnologie Utilizzate</b>	<b>6</b>
4.1	Linguaggi e Framework Core . . . . .	6
4.2	AI e Machine Learning . . . . .	6
4.3	Containerizzazione e Deployment . . . . .	6
4.4	Development Tools . . . . .	7
4.5	Librerie di Supporto . . . . .	7
<b>5</b>	<b>Agenti AI Implementati</b>	<b>7</b>
5.1	Story Generator Agent . . . . .	7
5.2	PDDL Generator Agent . . . . .	8
5.3	Reflection Agent . . . . .	8
5.4	Frontend Generator Agent . . . . .	9
<b>6</b>	<b>Flusso di Funzionamento</b>	<b>9</b>
6.1	Phase 1: Story Generation e PDDL Validation . . . . .	9
6.2	Phase 2: Interactive Story Game . . . . .	9
6.3	Ciclo di Raffinamento . . . . .	10
<b>7</b>	<b>Struttura del Progetto</b>	<b>10</b>
7.1	Moduli Chiave . . . . .	10
7.1.1	<b>app.py</b> - Applicazione Principale . . . . .	10
7.1.2	<b>cli.py</b> - Interfaccia Command Line . . . . .	10
<b>8</b>	<b>Installazione e Utilizzo</b>	<b>11</b>
8.1	Prerequisiti . . . . .	11
8.2	Installazione con Docker (Raccomandato) . . . . .	11
8.3	Installazione Locale . . . . .	11
8.4	Utilizzo CLI . . . . .	12

8.4.1	<b>Esecuzione Completa</b>	12
8.4.2	<b>Esecuzione per Fasi</b>	12
8.4.3	<b>Opzioni Avanzate</b>	12
8.5	<b>Configurazione Ambiente</b>	13
8.5.1	<b>Variabili Ambiente</b>	13
8.5.2	<b>File di Configurazione</b>	13
<b>9</b>	<b>Esempi Pratici</b>	<b>13</b>
9.1	<b>PDDL Generato</b>	13
9.1.1	<b>Domain File</b>	13
9.1.2	<b>Problem File</b>	14
9.2	<b>Output del Sistema</b>	14
9.2.1	<b>Phase 1 Output</b>	14
9.3	<b>Interfaccia Web Generata</b>	14
<b>10</b>	<b>Conclusioni e Sviluppi Futuri</b>	<b>15</b>
10.1	<b>Risultati Raggiunti</b>	15
10.1.1	<b>Obiettivi Tecnici</b>	15
10.1.2	<b>Obiettivi di Qualità</b>	15
10.2	<b>Innovazioni Introdotte</b>	15
10.2.1	<b>Integrazione AI Generativa + Classical Planning</b>	15
10.2.2	<b>Sistema Multi-Agente Specializzato</b>	16
10.2.3	<b>Validation-Driven Development</b>	16
10.3	<b>Limitazioni Attuali</b>	16
10.3.1	<b>Limitazioni Tecniche</b>	16
10.3.2	<b>Limitazioni Funzionali</b>	16
10.4	<b>Sviluppi Futuri</b>	17
10.4.1	<b>Miglioramenti a Breve Termine</b>	17
10.4.2	<b>Miglioramenti a Medio Termine</b>	17
10.4.3	<b>Visione a Lungo Termine</b>	18
10.5	<b>Impatto e Valore del Progetto</b>	18
10.5.1	<b>Valore Accademico</b>	18
10.5.2	<b>Valore Pratico</b>	18
<b>11</b>	<b>Bibliografia e Riferimenti</b>	<b>19</b>
<b>12</b>	<b>Repository e Risorse</b>	<b>19</b>

# 1 Introduzione

**QuestMaster AI** è un sistema innovativo che combina l'intelligenza artificiale generativa con tecniche di pianificazione classica per creare esperienze narrative interattive. Il progetto rappresenta un'implementazione avanzata di un sistema multi-agente che utilizza PDDL (Planning Domain Definition Language) e Large Language Models (LLM) per generare automaticamente quest narrative logicamente consistenti e coinvolgenti.

Il sistema è stato progettato come un'applicazione a due fasi che assiste gli autori nella creazione di avventure interattive attraverso un processo automatizzato ma controllabile, garantendo sia la creatività narrativa che la coerenza logica dell'esperienza di gioco.

## 2 Obiettivi del Progetto

### 2.1 Obiettivi Primari

- **Generazione Automatica di Narrative:** Creare storie coinvolgenti e coerenti utilizzando AI generativa.
- **Validazione Logica:** Garantire che le quest generate siano logicamente risolvibili attraverso PDDL.
- **Interfaccia Interattiva:** Fornire un'esperienza utente intuitiva e immersiva.
- **Architettura Modulare:** Implementare un sistema facilmente estensibile e manutenibile.

### 2.2 Obiettivi Secondari

- **Containerizzazione:** Deployment semplificato attraverso Docker.
- **CLI Completa:** Strumenti da riga di comando per sviluppatori.
- **Logging Avanzato:** Sistema di monitoraggio e debugging completo.
- **Validazione Continua:** Ciclo di raffinamento automatico per migliorare la qualità.

## 3 Architettura del Sistema

### 3.1 Panoramica Architetture

QuestMaster AI segue un'architettura multi-layered basata su agenti specializzati:

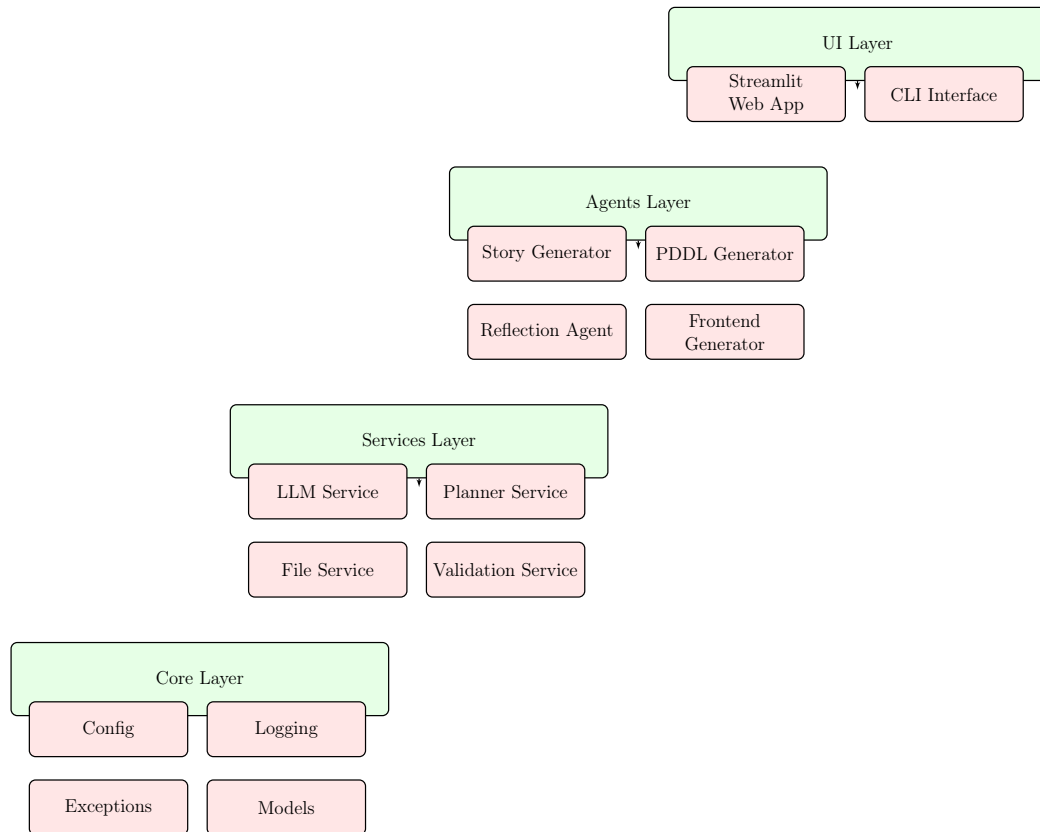


Figura 1: Architettura del Sistema QuestMaster AI

### 3.2 Componenti Principali

#### 3.2.1 Core Components

- **Configuration Management:** Gestione centralizzata delle configurazioni.
- **Logging System:** Sistema di logging strutturato con Rich e Structlog.
- **Exception Handling:** Gestione degli errori specializzata per dominio.

- **Data Models:** Modelli Pydantic per validazione e serializzazione.

### 3.2.2 Services Layer

- **LLM Service:** Interfaccia con OpenAI GPT per generazione di contenuti.
- **Planner Service:** Integrazione con Fast Downward planner.
- **File Service:** Gestione I/O di file e persistenza.
- **Validation Service:** Validazione di PDDL e narrative.

## 4 Tecnologie Utilizzate

### 4.1 Linguaggi e Framework Core

- **Python 3.9+:** Linguaggio principale del progetto.
- **Pydantic 2.0+:** Validazione e serializzazione dati.
- **Streamlit 1.28+:** Framework per interfaccia web.
- **Click 8.0+:** Framework per CLI.

### 4.2 AI e Machine Learning

- **OpenAI API 1.0+:** Large Language Models (GPT-4o-mini).
- **Fast Downward:** Classical planner per PDDL.
- **PDDL:** Planning Domain Definition Language.

### 4.3 Containerizzazione e Deployment

- **Docker:** Containerizzazione dell'applicazione.
- **Docker Compose:** Orchestrazione multi-container.
- **Shell Scripting:** Automazione setup e deployment.

## 4.4 Development Tools

- **pytest**: Testing framework.
- **Black**: Code formatting.
- **Ruff**: Linting veloce.
- **mypy**: Type checking.
- **Rich**: Output colorato e formattato.
- **Tenacity**: Retry logic per API calls.

## 4.5 Librerie di Supporto

- **structlog**: Logging strutturato.
- **pathvalidate**: Validazione percorsi file.
- **pytest-cov**: Coverage testing.
- **pytest-asyncio**: Testing asincrono.

# 5 Agenti AI Implementati

## 5.1 Story Generator Agent

**Responsabilità:** Generazione narrativa principale.

**Caratteristiche:**

- Creazione di lore dettagliato con personaggi, ambientazioni e trama.
- Integrazione con template JSON per struttura consistente.
- Generazione di elementi narrativi ricchi e coinvolgenti.
- Adattamento a vincoli di branching factor e profondità.

**Implementazione:**

```
1 class StoryGeneratorAgent:
2     def generate_story(self, lore_input: str) -> Lore:
3         # Utilizza LLM per generare narrative strutturate
4         # Valida la coerenza narrativa
5         # Restituisce oggetto Lore validato
```

Listing 1: StoryGeneratorAgent Code Snippet

## 5.2 PDDL Generator Agent

**Responsabilità:** Conversione narrative in modelli di pianificazione.

**Caratteristiche:**

- Traduzione di storie in domini e problemi PDDL.
- Generazione di azioni, predicati e condizioni goal.
- Commenti esplicativi per ogni elemento PDDL.
- Ottimizzazione per solvibilità.

**Elementi PDDL Generati:**

- **Domini:** Definizione azioni, predicati, tipi.
- **Problemi:** Stato iniziale, oggetti, goal.
- **Azioni:** `move`, `collect_item`, `overcome_obstacle`, `complete_quest`.

## 5.3 Reflection Agent

**Responsabilità:** Validazione e raffinamento iterativo.

**Caratteristiche:**

- Analisi di inconsistenze logiche in PDDL.
- Suggerimenti di miglioramento automatici.
- Ciclo di raffinamento iterativo.
- Interazione con utente per approvazione modifiche.

**Processo di Riflessione:**

1. Analisi del PDDL generato.
2. Identificazione problemi di solvibilità.
3. Generazione suggerimenti specifici.
4. Implementazione modifiche approvate.



## 5.4 Frontend Generator Agent

**Responsabilità:** Creazione interfacce interattive.

**Caratteristiche:**

- Generazione di interfacce Streamlit dinamiche.
- Adattamento UI ai requisiti della storia.
- Integrazione con sistema di navigazione.
- Componenti interattivi personalizzati.

## 6 Flusso di Funzionamento

### 6.1 Phase 1: Story Generation e PDDL Validation

**Dettaglio del Processo:**

1. **Input Processing:** Lettura del file lore iniziale.
2. **Story Generation:** StoryGeneratorAgent crea narrative dettagliate.
3. **PDDL Translation:** PDDLGeneratorAgent converte in formato pianificazione.
4. **Validation:** Fast Downward verifica solvibilità.
5. **Refinement Loop:** ReflectionAgent migliora iterativamente se necessario.
6. **Output:** PDDL validato e storia finalizzata.

### 6.2 Phase 2: Interactive Story Game

**Componenti Generati:**

- Interfaccia web Streamlit personalizzata.
- Sistema di navigazione basato su stati.
- Gestione delle scelte utente.
- Visualizzazione progressiva della storia.

## 6.3 Ciclo di Raffinamento

Il sistema implementa un ciclo di raffinamento automatico per garantire qualità:

1. **Detection:** Identificazione automatica di problemi.
2. **Analysis:** Analisi approfondita delle cause.
3. **Suggestion:** Generazione di soluzioni specifiche.
4. **Validation:** Test delle modifiche proposte.
5. **Integration:** Applicazione delle modifiche approvate.

## 7 Struttura del Progetto

### 7.1 Moduli Chiave

#### 7.1.1 app.py - Applicazione Principale

```
1 class QuestMasterApp:
2     def __init__(self, api_key: Optional[str] = None):
3         # Inizializzazione servizi e agenti
4
5     def run_phase1(self, lore_path: Optional[str] = None)
6     -> ValidationResult:
7         # Esecuzione Phase 1: Story Generation
8
9     def run_phase2(self) -> None:
10        # Esecuzione Phase 2: Interactive Game
```

Listing 2: app.py Code Snippet

#### 7.1.2 cli.py - Interfaccia Command Line

- Comando phase1: Esecuzione generazione storia.
- Comando phase2: Avvio gioco interattivo.
- Comando run: Esecuzione completa pipeline.
- Opzioni configurazione: log-level, debug, api-key.

## 8 Installazione e Utilizzo

### 8.1 Prerequisiti

- Python 3.9+
- Docker (opzionale ma raccomandato)
- OpenAI API Key
- Sistema Unix-like (macOS/Linux)

### 8.2 Installazione con Docker (Raccomandato)

```
1 # Clone del repository
2 git clone https://github.com/umbertocicciaa/QuestMasterAI
3   .git
4
5 # Setup automatico con Docker
6 ./docker-setup.sh
7
8 # Oppure manualmente
9 docker-compose up --build
```

Listing 3: Installazione con Docker

Accesso all'applicazione:

- Web Interface: <http://localhost:8501>
- Logs: `docker-compose logs -f questmaster`

### 8.3 Installazione Locale

```
1 # Setup ambiente virtuale
2 python -m venv venv
3 source venv/bin/activate # su Windows: venv\Scripts\
4   activate
5
6 # Installazione dipendenze
7 pip install -e .
8
9 # Setup Fast Downward
```

```
9 ./start.sh
10
11 # Configurazione API Key
12 export OPENAI_API_KEY="your-api-key-here"
```

Listing 4: Installazione Locale

## 8.4 Utilizzo CLI

### 8.4.1 Esecuzione Completa

```
1 python -m questmaster.cli run
```

Listing 5: Esecuzione Completa

### 8.4.2 Esecuzione per Fasi

```
1 # Phase 1: Story Generation
2 python -m questmaster.cli phase1 --lore-path data/lore.
   json
3
4 # Phase 2: Interactive Frontend
5 python -m questmaster.cli phase2
```

Listing 6: Esecuzione per Fasi

### 8.4.3 Opzioni Avanzate

```
1 # Debug mode
2 python -m questmaster.cli --debug --log-level DEBUG
   phase1
3
4 # Custom API key
5 python -m questmaster.cli --api-key "sk-custom-key" run
6
7 # Help
8 python -m questmaster.cli --help
```

Listing 7: Opzioni Avanzate CLI

## 8.5 Configurazione Ambiente

### 8.5.1 Variabili Ambiente

```
1 export OPENAI_API_KEY="sk-your-openai-key"
2 export CHATGPT_MODEL="gpt-4o-mini-2024-07-18"
3 export LOG_LEVEL="INFO"
4 export DEBUG="False"
5 export FAST_DOWNWARD_TIMEOUT="300"
```

Listing 8: Variabili Ambiente

### 8.5.2 File di Configurazione

Le configurazioni sono gestite tramite `pyproject.toml` e il sistema di settings di Pydantic.

## 9 Esempi Pratici

### 9.1 PDDL Generato

#### 9.1.1 Domain File

```
1 (define (domain legacy-quest)
2   (:requirements :strips :typing)
3   (:types character location item obstacle)
4
5   (:predicates
6     (at ?c - character ?loc - location) ; Character is at
7     location
8     (has_item ?c - character ?item - item) ; Character
9     has item
10    (quest_completed) ; Quest completion state
11  )
12
13  (:action move
14    :parameters (?c - character ?from - location ?to -
15    location)
16    :precondition (at ?c ?from)
17    :effect (and (not (at ?c ?from)) (at ?c ?to))
18  )
19 )
```

Listing 9: PDDL Domain File

### 9.1.2 Problem File

```
1 (define (problem legacy-quest-scenario)
2   (:domain legacy-quest)
3
4   (:objects
5     hero - character
6     village forest cave - location
7     sword shield quest_item - item
8     dragon - obstacle
9   )
10
11   (:init
12     (at hero village)
13   )
14
15   (:goal (quest_completed))
16 )
```

Listing 10: PDDL Problem File

## 9.2 Output del Sistema

### 9.2.1 Phase 1 Output

```
1 Starting Phase 1: Story Generation
2 Phase 1 completed successfully!
3 Execution time: 45.23s
4 Plan length: 8 steps
```

Listing 11: Output Fase 1

## 9.3 Interfaccia Web Generata

Il sistema genera automaticamente un'interfaccia Streamlit che include:

- **Navigation Panel:** Controlli per muoversi nella storia.
- **Story Display:** Visualizzazione testuale dell'avventura.
- **Action Buttons:** Bottoni per le scelte disponibili.
- **State Tracking:** Monitoraggio del progresso.
- **Rich Formatting:** Stile e formattazione accattivanti.

## 10 Conclusioni e Sviluppi Futuri

### 10.1 Risultati Raggiunti

Il progetto QuestMasterAI ha raggiunto con successo gli obiettivi prefissati:

#### 10.1.1 Obiettivi Tecnici

- ✓ **Architettura Multi-Agente:** Sistema modulare e estensibile.
- ✓ **Integrazione PDDL:** Validazione logica attraverso classical planning.
- ✓ **Pipeline Automatizzata:** Processo end-to-end completamente automatico.
- ✓ **Interfaccia Moderna:** UI web intuitiva e responsive.
- ✓ **Containerizzazione:** Deployment semplificato con Docker.

#### 10.1.2 Obiettivi di Qualità

- ✓ **Robustezza:** Gestione errori e recovery automatico.
- ✓ **Scalabilità:** Architettura predisposta per estensioni.
- ✓ **Manutenibilità:** Codice ben strutturato e documentato.
- ✓ **Testing:** Suite di test automatizzati.
- ✓ **Logging:** Sistema di monitoraggio completo.

### 10.2 Innovazioni Introdotte

#### 10.2.1 Integrazione AI Generativa + Classical Planning

L'aspetto più innovativo del progetto è l'integrazione seamless tra:

- **LLM per creatività:** Generazione di narrative coinvolgenti.
- **PDDL per logica:** Garanzia di coerenza e solvibilità.
- **Ciclo di raffinamento:** Miglioramento iterativo automatico.

### 10.2.2 Sistema Multi-Agente Specializzato

Ogni agente ha responsabilità specifiche:

- **Separazione delle responsabilità:** Maggiore manutenibilità.
- **Specializzazione:** Ottimizzazione per task specifici.
- **Coordinazione:** Collaborazione efficace tra agenti.

### 10.2.3 Validation-Driven Development

- **Feedback immediato:** Validazione continua della qualità.
- **Auto-correzione:** Capacità di auto-miglioramento.
- **Garanzie formali:** Uso di classical planning per verifica.

## 10.3 Limitazioni Attuali

### 10.3.1 Limitazioni Tecniche

- **Dipendenza da API esterne:** Richiede connessione internet e API key.
- **Tempo di elaborazione:** Generazione può richiedere diversi minuti.
- **Complessità PDDL:** Limitato a domini relativamente semplici.
- **Lingua:** Attualmente ottimizzato principalmente per italiano.

### 10.3.2 Limitazioni Funzionali

- **Personalizzazione limitata:** Template predefiniti per la struttura.
- **Interazione utente:** Interfaccia principalmente read-only in Phase 2.
- **Multimedia:** Supporto limitato per immagini e audio.
- **Persistenza:** Stato di gioco non persiste tra sessioni.



## 10.4 Sviluppi Futuri

### 10.4.1 Miglioramenti a Breve Termine

#### Enhanced User Interaction

```
1 # Implementazione di feedback loop utente
2 class InteractiveRefinement:
3     def get_user_feedback(self, suggestions: List[str])
4     -> UserFeedback:
5         # Chat interface per approvazione modifiche
6
7     def apply_user_preferences(self, preferences:
8     UserPreferences):
9         # Personalizzazione basata su preferenze utente
```

Listing 12: Enhanced User Interaction

- **Multi-language Support**

- Estensione a inglese, francese, spagnolo.
- Template localizzati per diverse culture narrative.
- Agenti specializzati per stili narrativi regionali.

- **Advanced PDDL Features**

- Supporto per temporal planning.
- Conditional effects e derived predicates.
- Optimization objectives per qualità narrativa.

### 10.4.2 Miglioramenti a Medio Termine

- **Visual Story Generation**

- Integrazione con DALL-E per generazione immagini.
- Interfaccia grafica avanzata con Streamlit.
- Supporto per mappe interattive.

- **Persistent Game State**

- Database per salvataggio progressi.
- Sistema di account utente.
- Multiplayer collaborative stories.

- **Advanced AI Agents**

- Character development agent per personaggi dinamici.
- Dialogue generation agent per conversazioni.
- Emotion tracking agent per coinvolgimento emotivo.

#### 10.4.3 Visione a Lungo Termine

- **Educational Platform**

- Integrazione in ambienti educativi.
- Strumenti per insegnanti.
- Analytics per apprendimento.

- **Commercial Applications**

- Sistema di authoring per game designer.
- Template marketplace.
- API per integrazione in giochi esistenti.

- **Research Contributions**

- Pubblicazioni scientifiche su AI + Planning.
- Benchmark per narrative generation.
- Open source community development.

### 10.5 Impatto e Valore del Progetto

#### 10.5.1 Valore Accademico

- **Ricerca Innovativa:** Combinazione unica di tecnologie.
- **Contributo Scientifico:** Metodologia riproducibile.
- **Formazione:** Esempio pratico di sistemi AI complessi.

#### 10.5.2 Valore Pratico

- **Strumento Reale:** Utilizzabile per creazione contenuti.
- **Framework Estensibile:** Base per progetti futuri. **Best Practices:** Esempio di sviluppo software moderno.

## 11 Bibliografia e Riferimenti

- **OpenAI API Documentation:** Utilizzo di Large Language Models.
- **PDDL Documentation:** Planning Domain Definition Language.
- **Fast Downward:** Classical Planning System.
- **Streamlit Documentation:** Framework per interfacce web.
- **Pydantic:** Data validation e settings management.
- **Docker:** Containerizzazione e deployment.

## 12 Repository e Risorse

- **GitHub Repository:** QuestMasterAI.
- **Documentation:** Documentazione completa nel README.md.
- **Issues:** Sistema di tracking per bug e feature requests.
- **Examples:** Esempi e template nella cartella `resources/`.

*Questa relazione documenta lo sviluppo e l'implementazione di QuestMasterAI, un sistema innovativo che dimostra le potenzialità dell'integrazione tra AI generativa e classical planning per la creazione di esperienze narrative interattive di alta qualità.*