

INTER ACTIVE STORY

06.12.2024

/ Angeles, Jacob
1114812

/ Bondi, Francesco
0989247

/ Casali, Edoardo
1030816

/ Alma Mater Studiorum
Università di Bologna

/ Laurea in Informatica
per il Management

/ Ingegneria del Software
aa 2023/24

PROJECT OVERVIEW

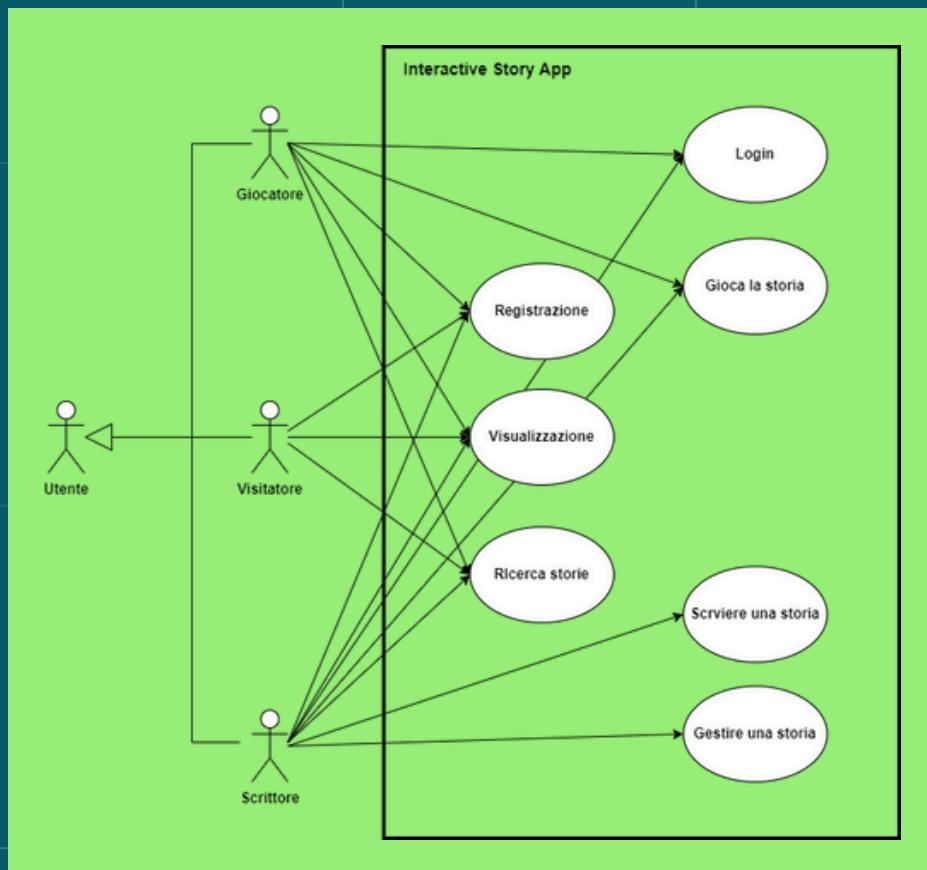
L'obiettivo del progetto è sviluppare un'applicazione interattiva che consenta agli utenti di creare, gestire e giocare storie interattive. La piattaforma permette un'esperienza **narrativa ramificata**, dove ogni decisione del giocatore influenza lo sviluppo della storia. Le funzionalità principali includono la possibilità di giocare ad una **storia interattiva** a scelta, la risoluzione di indovinelli e la possibilità di salvare lo stato del gioco in corso.



Il sistema consente agli utenti registrati di scrivere nuove storie. Ogni storia deve avere un inizio, diverse diramazioni e uno o più finali. Durante la creazione, gli autori possono definire scenari che presentano scelte multiple, indovinelli (testuali o numerici) e oggetti raccoglibili. Gli oggetti raccolti vengono aggiunti all'inventario del protagonista e possono essere richiesti per accedere a determinate diramazioni. Gli scenari esistenti possono essere modificati nel loro testo descrittivo, ma le diramazioni predefinite non possono essere alterate.

INCEPTION PHASE

Durante la fase di inception, abbiamo svolto un'analisi approfondita dei requisiti del progetto, definendo i **casi d'uso** (use case) e progettando il **modello di dominio** in modo chiaro e strutturato.



Un Use Case (caso d'uso) è una descrizione strutturata di come un sistema interagisce con uno o più attori (utenti o sistemi esterni) per raggiungere uno specifico obiettivo. Si tratta di un elemento chiave nell'ingegneria del software per documentare i requisiti funzionali di un sistema, evidenziando i flussi di interazione tra l'attore e il sistema stesso.

Ogni Use Case include un flusso principale di eventi (o scenario principale) e, se necessario, uno o più flussi alternativi, per coprire tutte le possibili varianti o eccezioni.

INCEPTION PHASE

INTERACTIVE STORY.

Use Case: Registrazione**ID:** UC1**Attore:** Utente**Pre-condizioni:**

- Nessuna.

Sequenza Principale:

1. L'utente inserisce un nuovo username e una password.
2. Il sistema verifica se lo username esiste già.
3. Se lo username non esiste, il sistema registra il nuovo utente.

Sequenza Alternativa (Username già esistente):

1. L'utente inserisce uno username già in uso.
2. Il sistema verifica che lo username esiste già.
3. Il sistema informa l'utente e richiede l'inserimento di uno username diverso.
4. L'utente fornisce un nuovo username e il flusso ritorna al passo 2 della Sequenza Principale.

Post-condizioni:

- L'utente viene registrato come giocatore e/o scrittore.
- L'utente è abilitato ad effettuare il login.

Use Case: Login**ID:** UC2**Attore:** Giocatore / Scrittore**Pre-condizioni:**

- L'utente deve essere già registrato nel sistema.

Sequenza Principale:

1. L'utente inserisce il proprio username e la password.
2. Il sistema verifica i dati forniti.
3. Se la verifica ha esito positivo, il sistema consente l'accesso all'utente.

Sequenza Alternativa (Credenziali non valide):

1. L'utente inserisce il proprio username e la password.
2. Il sistema verifica i dati forniti.
3. Se la verifica ha esito negativo, il sistema informa l'utente che i dati inseriti non sono validi.
4. Il sistema richiede all'utente di reinserire i dati corretti.

Post-condizioni:

- Per giocatore: l'utente può accedere e giocare una storia.
- Per scrittore: l'utente può gestire o scrivere una storia.

Use Case: Ricerca**ID:** UC3**Attore:** Utente**Pre-condizioni:**

- Nessuna.

Sequenza Principale:

1. L'utente visualizza l'elenco completo delle storie disponibili.

Sequenza Alternativa (Credenziali non valide):

1. L'utente applica uno o più filtri per restringere l'elenco delle storie (es. genere, autore, valutazione, ecc.).
2. Il sistema aggiorna l'elenco mostrando solo le storie che soddisfano i criteri selezionati.

Post-condizioni:

- L'utente può selezionare una storia dall'elenco per visualizzarne i dettagli o iniziarla.

Use Case: Visualizzazione**ID:** UC4**Attore:** Utente**Pre-condizioni:**

- L'utente deve aver effettuato una ricerca storia (UC3).

Sequenza Principale:

1. L'utente seleziona una storia dall'elenco delle storie disponibili.
2. Il sistema mostra il primo scenario della storia selezionata.

Sequenza Alternativa (Username già esistente):

- Nessuna specificata.

Post-condizioni:

- L'utente può decidere di effettuare il login per continuare la storia.

Use Case: Scrivere una storia**ID:** UC5**Attore:** Scrittore**Pre-condizioni:**

- L'utente deve essere registrato e autenticato.

Sequenza Principale:

1. Lo scrittore crea una nuova storia.
2. Lo scrittore aggiunge uno o più scenari alla storia.
3. Per ciascuno scenario, lo scrittore crea un indovinello che può essere risolto utilizzando degli oggetti o facendo una scelta.

Sequenza Alternativa:

- Non specificata.

Post-condizioni:

- La storia diventa disponibile per essere giocata dagli utenti.

Use Case: Gestire una storia**ID:** UC6**Attore:** Scrittore**Pre-condizioni:**

- L'utente deve essere registrato e autenticato.

Sequenza Principale:

1. Lo scrittore seleziona una delle storie che ha creato.
2. Lo scrittore modifica uno o più scenari della storia.
3. Per ciascuno scenario, lo scrittore modifica un indovinello che può essere risolto utilizzando degli oggetti o facendo una scelta.

Sequenza Alternativa:

- Non specificata.

Post-condizioni:

- Le modifiche apportate diventano visibili e aggiornano la versione giocabile della storia.

Use Case: Giocare una storia**ID:** UC7**Attore:** Giocatore**Pre-condizioni:**

- L'utente deve essere registrato e autenticato.

Sequenza Principale:

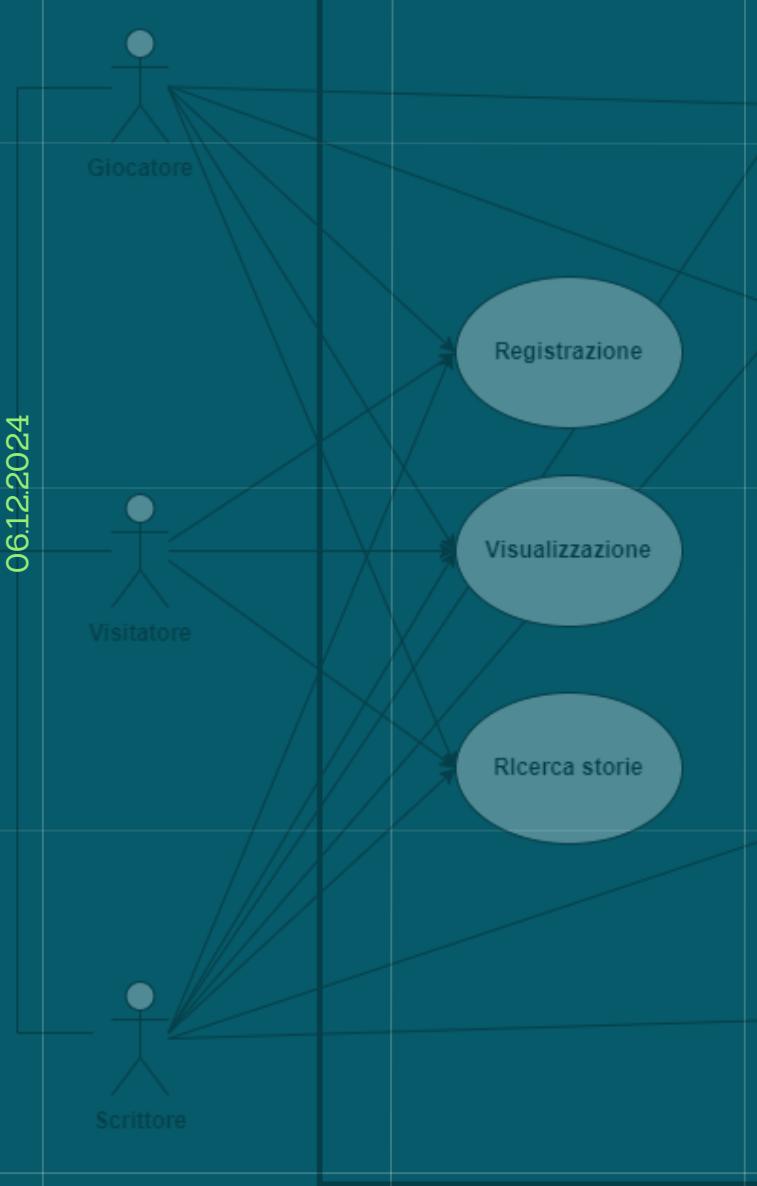
1. Il giocatore seleziona una storia disponibile.
2. Il giocatore inizia a giocare dal primo scenario e prosegue attraverso le varie scelte/opzioni disponibili.
3. Per ogni risposta corretta o corretto utilizzo degli strumenti disponibili, il giocatore accede allo scenario successivo.
4. Il giocatore raggiunge eventualmente il finale della storia.

Sequenza Alternativa (Errore o blocco):

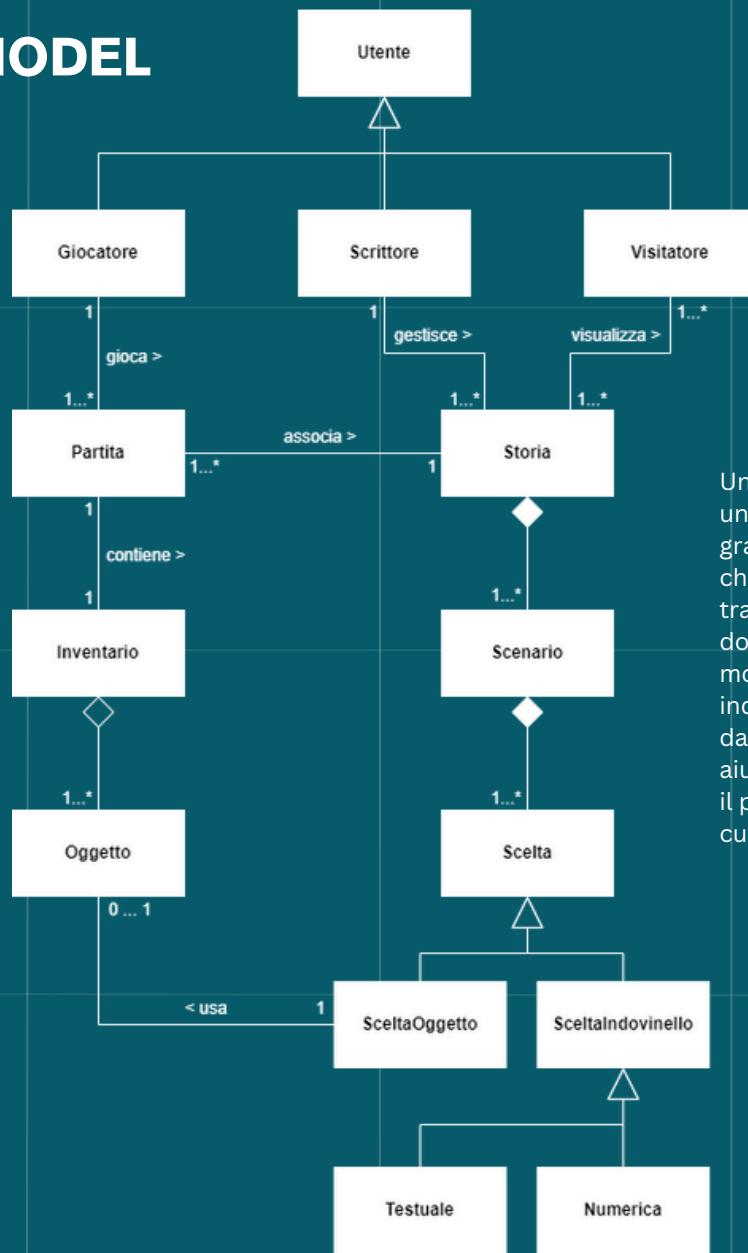
1. Se il giocatore fornisce una risposta sbagliata, il sistema visualizza un messaggio di errore e il giocatore rimane nello scenario attuale.
2. Il giocatore può decidere di tornare allo scenario precedente, se consentito.

Post-condizioni:

- Il giocatore termina la storia o può riprendere il gioco in base al proprio progresso.



DOMAIN MODEL



Un *diagramma di dominio* è una rappresentazione grafica che illustra i concetti chiave (entità) e le relazioni tra essi in un determinato dominio applicativo. È un modello concettuale, indipendente dall'implementazione, che aiuta a comprendere meglio il problema e il contesto in cui opera il sistema.

06.12.2024

BURN DOWN CHART



PROJECT DEVELOPMENT

Il progetto è stato sviluppato seguendo un *modello di processo ibrido strutturato-agile* che combina pratiche formali di modellazione con l'approccio iterativo e incrementale di Scrum. Il modello si articola in due fasi principali, ciascuna con obiettivi specifici e attività correlate.

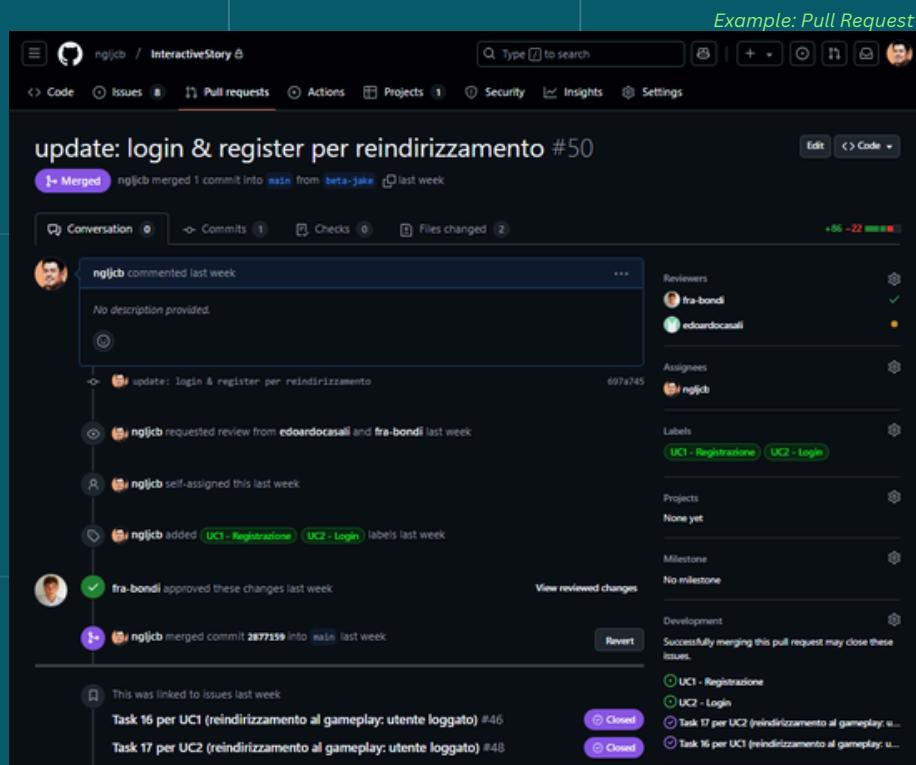
Durante la fase di **Inception**, il team si è concentrato sull'analisi iniziale e sulla creazione degli artefatti fondamentali per la definizione del sistema. Sono stati prodotti artefatti chiave come la modellazione dei *casi d'uso*, che ha descritto in dettaglio le interazioni previste tra gli utenti e il sistema, il *modello di dominio*, che rappresenta le entità principali e le loro relazioni all'interno del contesto applicativo. Questa fase ha permesso di stabilire una base solida per il design e lo sviluppo successivo, allineando il team sugli obiettivi e sui requisiti.

La seconda fase, denominata **Construction**, ha seguito le pratiche di gestione di processo proprie del framework **Scrum**, suddividendo il lavoro in *sprint incrementali*. All'interno del team, i membri hanno ricoperto a rotazione ruoli chiave per favorire la collaborazione e la comprensione reciproca. Il development team era composto da due membri responsabili dello sviluppo tecnico, mentre il ruolo di **Scrum Master** veniva ricoperto da un membro che, a seconda della dimensione del gruppo, poteva svolgere anche attività tecniche. Il ruolo di **Product Owner**, essenziale per la gestione delle priorità e dei requisiti, era assegnato a un singolo membro per sprint. Questo approccio dinamico nella distribuzione dei ruoli ha permesso al team di sviluppare una comprensione olistica del progetto, migliorando la qualità complessiva del prodotto.

Il team ha rigorosamente seguito gli eventi del framework **Scrum**, organizzando lo **Sprint Planning** all'inizio di ogni iterazione per pianificare le attività e definire gli obiettivi. I **Daily Scrum**, brevi incontri quotidiani via incontri su **Teams**, hanno facilitato il monitoraggio del progresso e la risoluzione di eventuali blocchi. Al termine di ogni sprint, lo **Sprint Review** ha fornito un'occasione per mostrare il lavoro completato e ricevere feedback, mentre lo **Sprint Retrospective** ha permesso al team di riflettere sulle prestazioni e identificare miglioramenti per le iterazioni successive. Questi eventi hanno assicurato un ciclo di sviluppo altamente iterativo e focalizzato sul miglioramento continuo.

Durante la fase di sviluppo, ogni modifica veniva sottoposta a revisione tramite **pull request** su **GitHub**, garantendo così un controllo della qualità rigoroso e un processo decisionale collaborativo. Questo approccio ha facilitato l'integrazione continua e ha ridotto il rischio di conflitti nel codice, migliorando la qualità e la stabilità del prodotto finale.

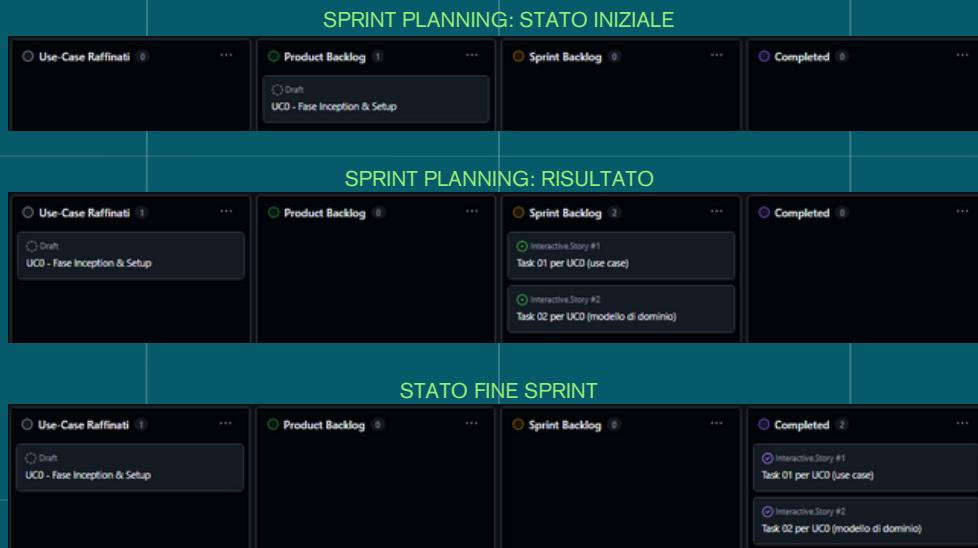
Il **diario del progetto** documenta in dettaglio le attività svolte, le decisioni prese e i risultati raggiunti durante ogni sprint. Questo documento riflette non solo il progresso tecnico, ma anche la crescita del team in termini di collaborazione e adozione delle best practices di sviluppo software.



DIARIO DEL PROGETTO

Sprint 0

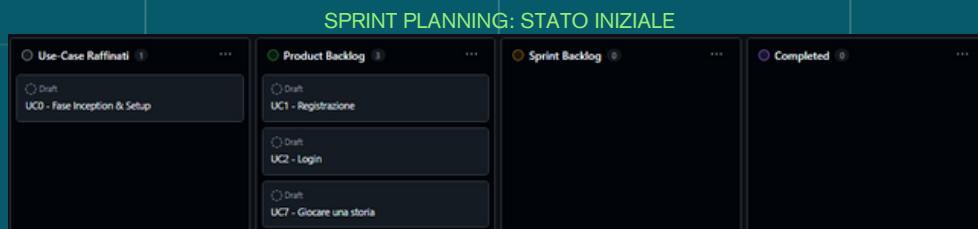
Lo sviluppo dell'applicativo è stato preceduto da uno sprint preliminare, interamente dedicato alla fase preparatoria. Durante questa fase, il team ha analizzato i requisiti del progetto, individuato i casi d'uso (Use Case) e progettato il modello di dominio, creando così una chiara visione della struttura e degli obiettivi del sistema. Sono state inoltre scaricate e configurate tutte le tecnologie necessarie per lo sviluppo, accompagnate da una fase di studio e preparazione per garantire una piena padronanza degli strumenti scelti. Infine, è stato selezionato lo stack tecnologico, realizzati i modelli di design e configurata la repository su GitHub. Questo approccio metodico ha permesso di gettare solide fondamenta per il progetto e di affrontare con maggiore efficienza la pianificazione delle attività e la suddivisione degli sprint successivi.

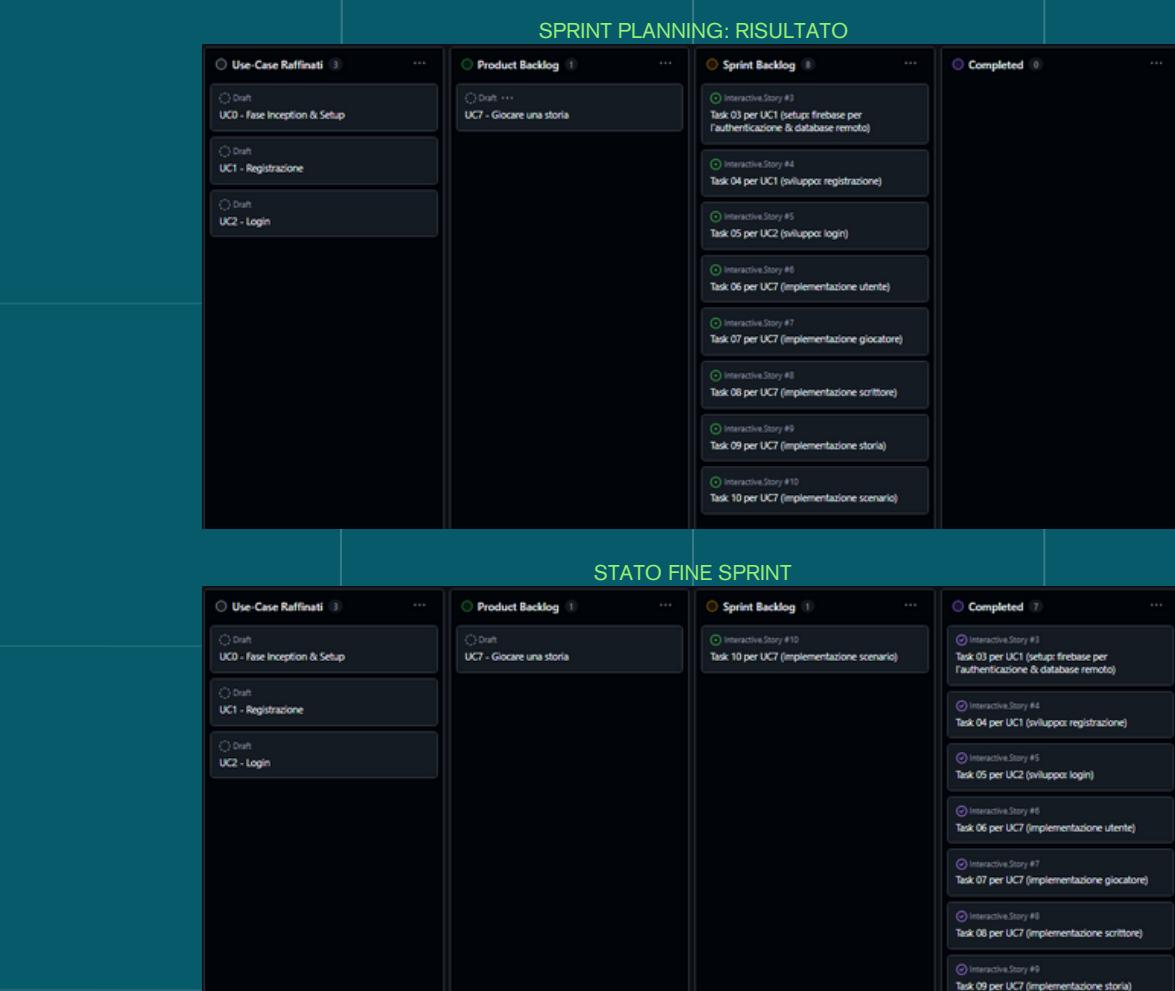


Sprint 1

Nella fase iniziale dello sviluppo, il team si è concentrato sulla realizzazione delle funzionalità di base dell'applicazione, con particolare attenzione alle funzioni di autenticazione, login e registrazione degli utenti. Parallelamente, è stato avviato lo sviluppo del gioco, inizialmente separato dal vincolo dell'autenticazione, per consentire una progressione più fluida delle funzionalità principali. Le attività sono state suddivise tra la business logic e l'interfaccia utente, permettendo di organizzare il lavoro in modo strutturato ed efficiente.

Assegnazione ruoli:	Descrizione	Operatore
	Product Owner	Jacob Angeles
	Scrum Master	Francesco Bondi
	Team Member	Edoardo Casali





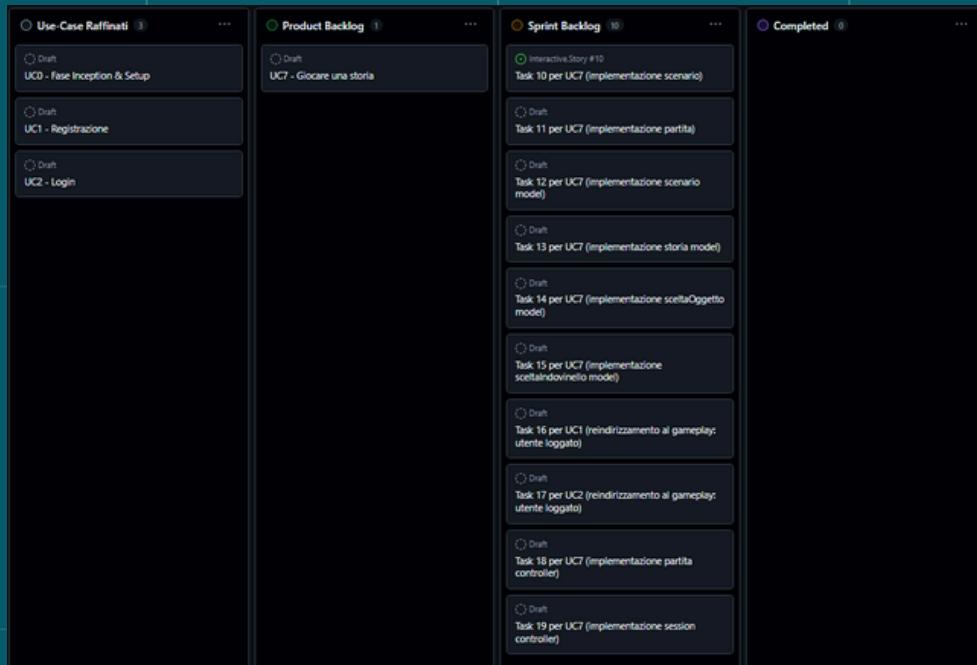
Sprint 2

Nel secondo sprint, il team ha proseguito lo sviluppo del gioco, concentrandosi sulla definizione e implementazione delle entità principali. Le attività si sono focalizzate sulla creazione dello scenario e delle scelte interattive, garantendo una struttura chiara e funzionale. Parallelamente, è iniziata la gestione del PartitaController e del SessionController, elementi fondamentali per orchestrare il flusso delle partite e la gestione delle sessioni utente. Inoltre, è stato completato il collegamento delle entità alla sessione di gioco, permettendo una gestione coerente e dinamica dell'esperienza utente. Questa fase ha rappresentato un passo cruciale per l'integrazione e il consolidamento delle funzionalità core del sistema.

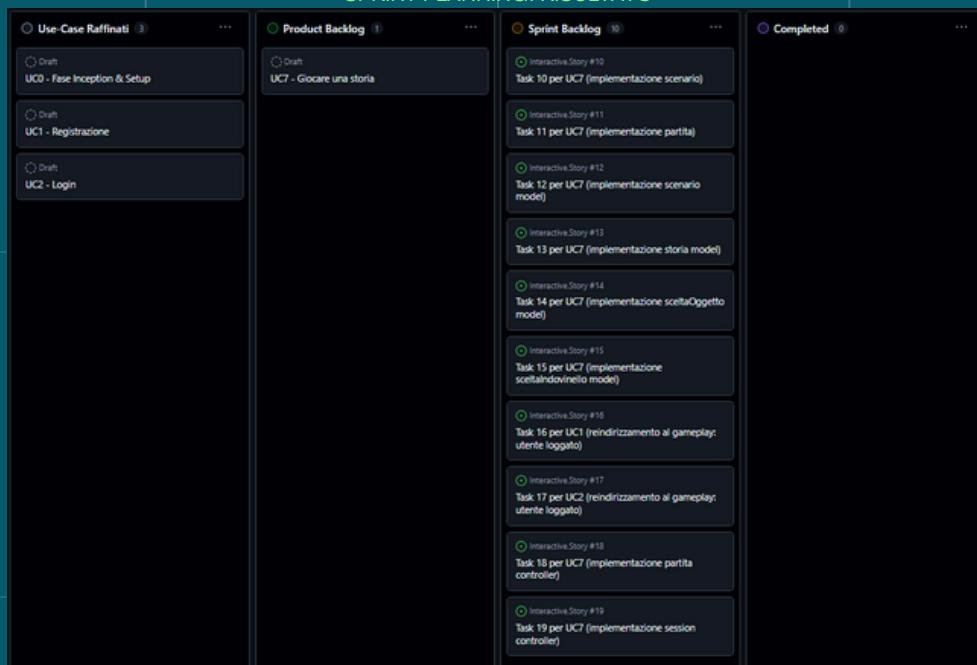
06.12.2024

Assegnazione ruoli:	Descrizione	Operatore
	Product Owner	Francesco Bondi
	Scrum Master	Edoardo Casali
	Team Member	Jacob Angeles

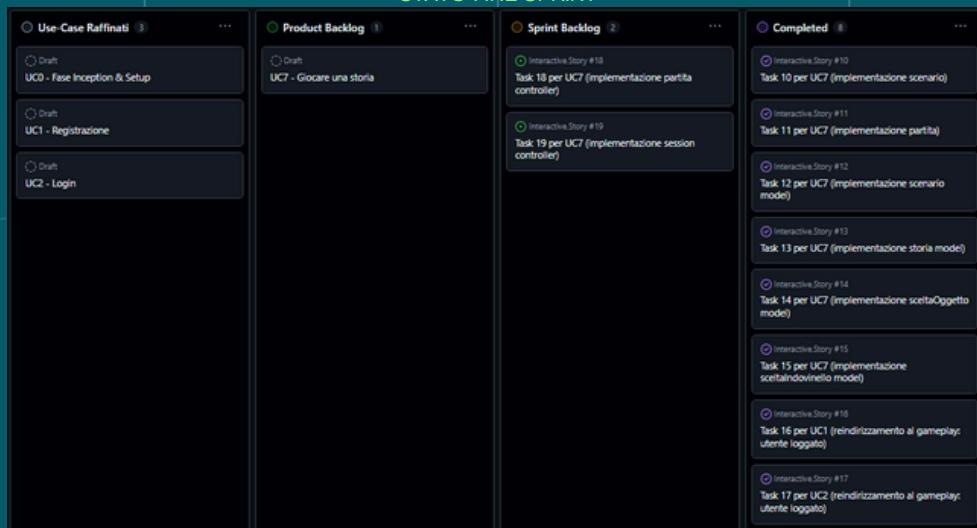
SPRINT PLANNING: STATO INIZIALE



SPRINT PLANNING: RISULTATO



STATO FINE SPRINT



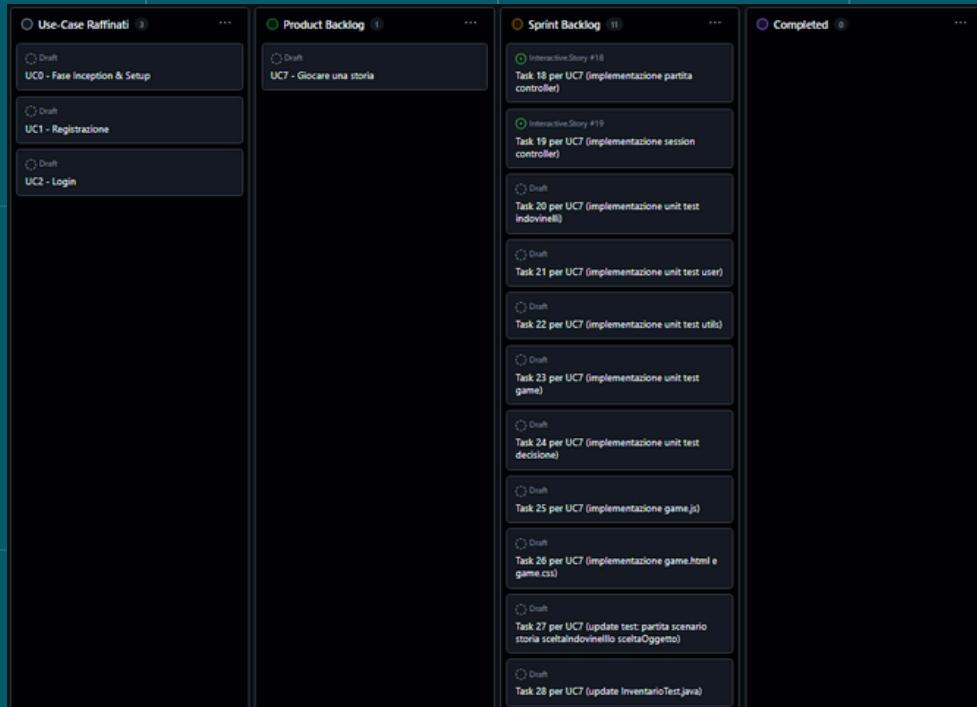
06.12.2024

Sprint 3

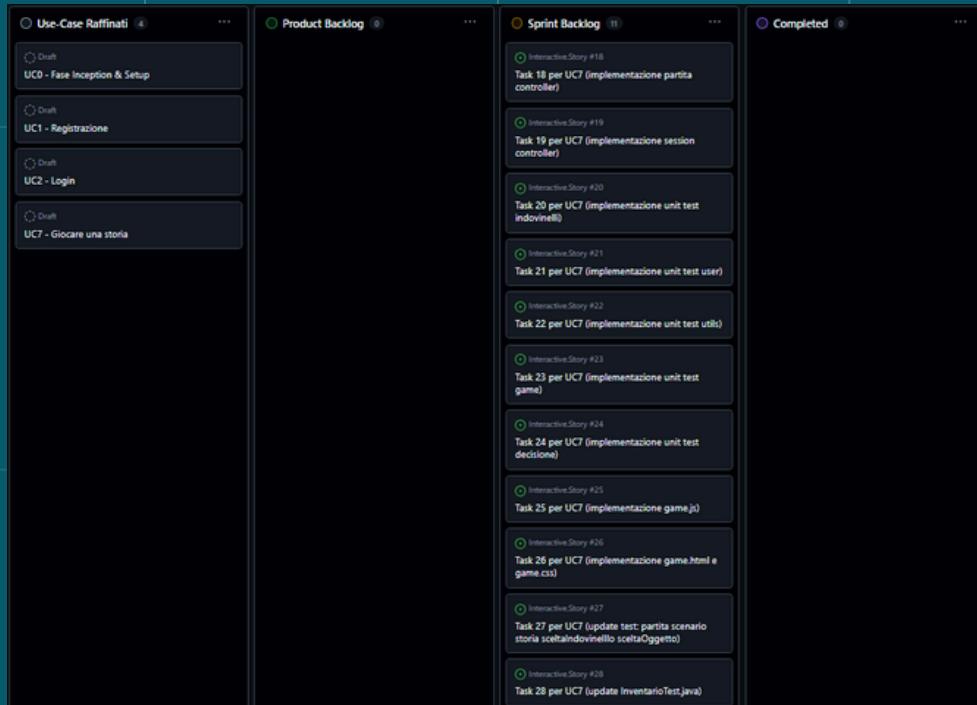
Al terzo sprint, il team ha completato lo sviluppo del gioco, portando a termine tutte le componenti backend rimaste. Contestualmente, è stato implementato il frontend, integrando le funzionalità sviluppate con un'interfaccia utente intuitiva e funzionale. Infine, sono stati condotti i test necessari per garantire la stabilità, la correttezza e la coerenza dell'applicazione, consolidando il lavoro svolto nei precedenti sprint.

Assegnazione ruoli:	Descrizione	Operatore
	Product Owner	Edoardo Casali
	Scrum Master	Jacob Angeles
	Team Member	Francesco Bondi

SPRINT PLANNING: STATO INIZIALE



SPRINT PLANNING: RISULTATO

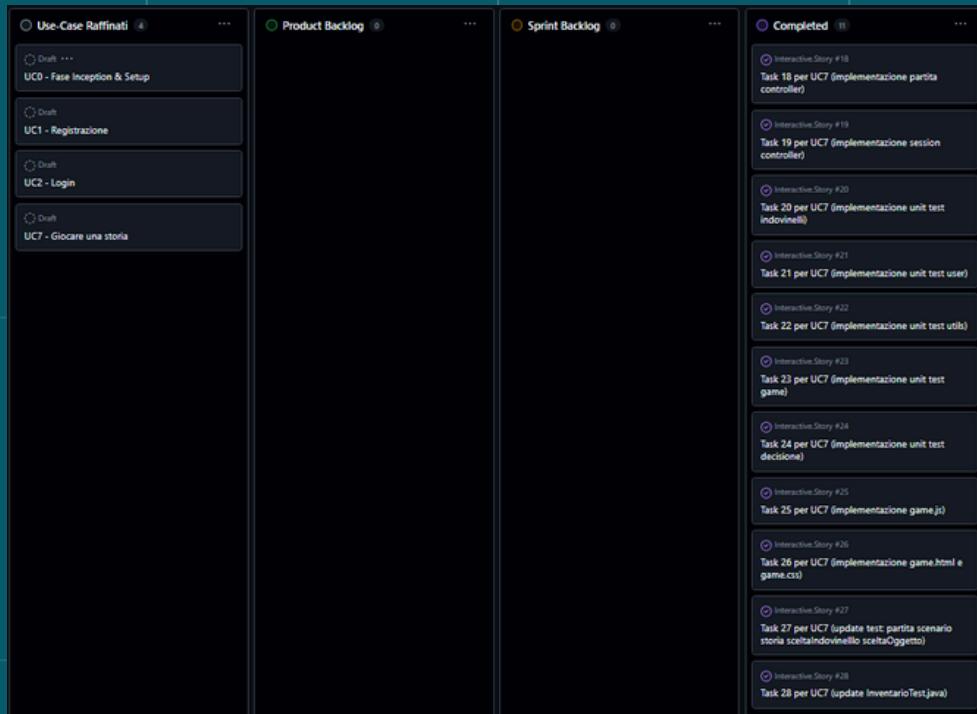


06.12.2024

PROJECT DIARY

INTERACTIVE STORY.

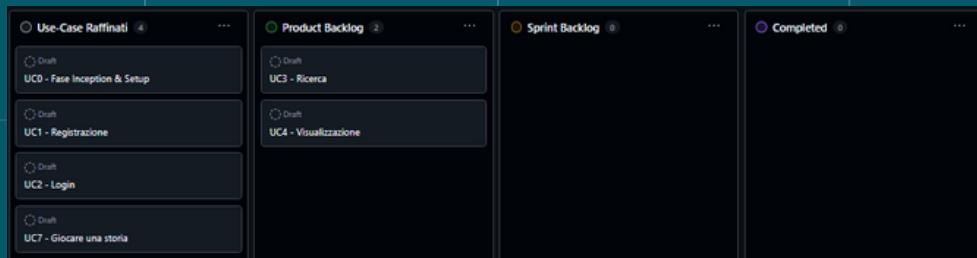
STATO FINE SPRINT

**Sprint 4**

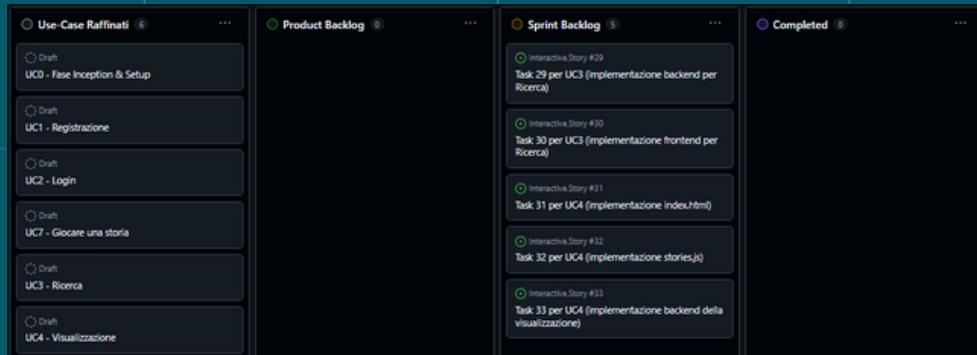
Durante il quarto sprint, il team si è dedicato all'implementazione delle funzionalità di ricerca e visualizzazione delle storie. Questa fase ha coinvolto sia lo sviluppo backend, per garantire una gestione efficiente e dinamica delle storie, sia il frontend, per offrire agli utenti un'interfaccia intuitiva e reattiva.

Assegnazione ruoli:	Descrizione	Operatore
	Product Owner	Francesco Bondi
	Scrum Master	Edoardo Casali
	Team Member	Jacob Angeles

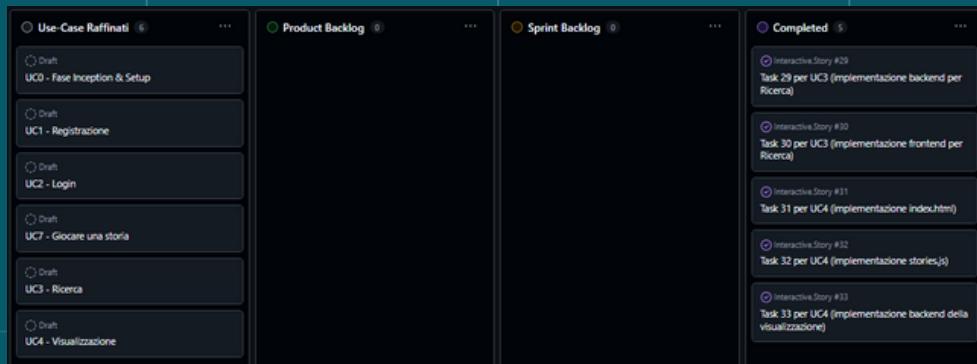
SPRINT PLANNING: STATO INIZIALE



SPRINT PLANNING: RISULTATO



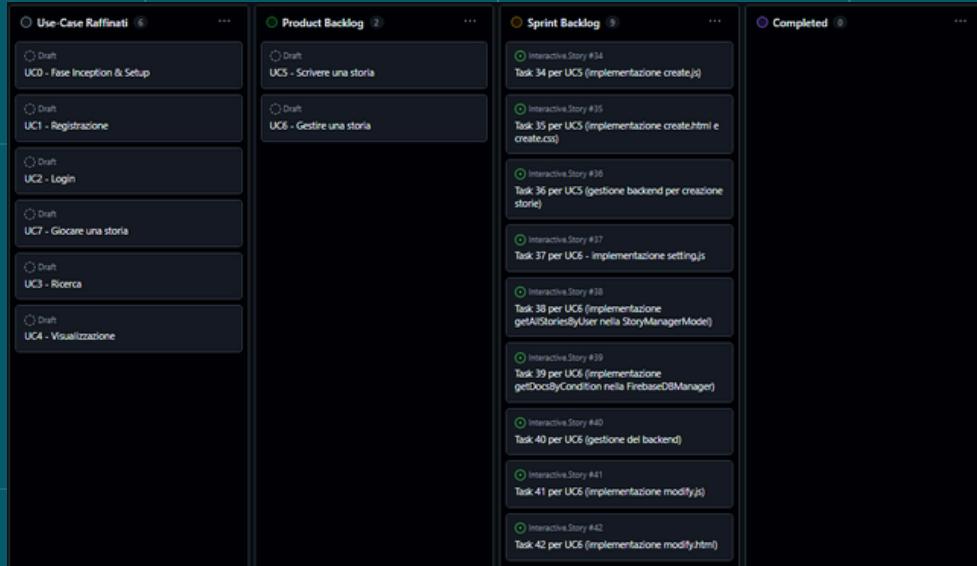
STATO FINE SPRINT

**Sprint 5**

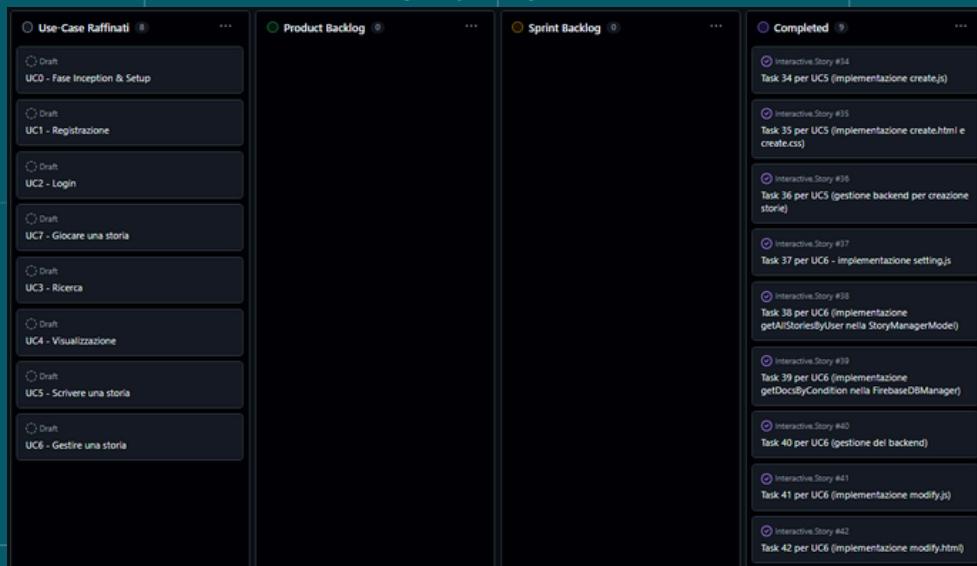
Nell'ultimo sprint, il team si è concentrato sull'implementazione delle funzionalità di creazione e modifica delle storie. Sono state sviluppate le logiche backend necessarie per gestire le storie in modo efficace e realizzato il frontend per fornire un'interfaccia semplice e funzionale agli utenti. Questa fase ha completato le principali funzionalità dell'applicazione, preparando il progetto per il rilascio finale.

Assegnazione ruoli:	Descrizione	Operatore
	Product Owner	Jacob Angeles
	Scrum Master	Francesco Bondi
	Team Member	Edoardo Casali

SPRINT PLANNING



STATO FINE SPRINT



INSTALLAZIONE E CONFIGURAZIONE

Prerequisiti

Per avviare e configurare correttamente il progetto, sono necessari i seguenti strumenti e risorse:

- **Java JDK 21+:** È fondamentale avere installata una versione di Java 21 o superiore per supportare tutte le funzionalità del backend sviluppato con Spring Boot.
- **Maven:** Utilizzato per la gestione delle dipendenze e per la compilazione del progetto backend. Assicurati che Maven sia configurato correttamente nel tuo ambiente.
- **Firebase:** Il progetto utilizza Firebase Firestore come database. È necessario configurare un progetto Firebase e ottenere le credenziali per l'accesso, rappresentate dal file google-services.json. ([Per la consegna del progetto, abbiamo incluso il file di configurazione.](#))

Passaggi per l'Installazione

- **Clona il Repository:** Il primo passo è ottenere una copia del progetto dal repository GitHub. Usa il comando seguente per clonare il repository e accedere alla directory principale del progetto:

```
git clone https://github.com/ngljcb/InteractiveStory.git  
cd InteractiveStory
```

- **Configurazione Iniziale (to skip):** Una volta clonato il progetto, è necessario configurare il file delle credenziali per Firebase, che autentica l'applicazione con il progetto Firebase. Procedi come segue:
 - Naviga nella directory resources dove risiede la configurazione del progetto:

```
cd /src/main/resources/
```

- Configura il file **google-services.json** fornendo i dettagli del tuo progetto Firebase ([config file già incluso del progetto per la consegna](#)). Un esempio del contenuto del file è il seguente:

```
{  
    "type": "xxxxxxxx",  
    "project_id": "xxxxxxxx",  
    "private_key_id": "xxxxxxxx",  
    "private_key": "xxxxxxxx",  
    "client_email": "xxxxxxxx",  
    "client_id": "xxxxxxxx",  
    "auth_uri": "xxxxxxxx",  
    "token_uri": "xxxxxxxx",  
    "auth_provider_x509_cert_url": "xxxxxxxx",  
    "client_x509_cert_url": "xxxxxxxx",  
    "universe_domain": "xxxxxxxx"  
}
```

- **Compilazione ed Esecuzione:** Dopo aver configurato Firebase, puoi procedere con la compilazione e l'esecuzione del progetto:

- Installa le dipendenze e compila il progetto utilizzando Maven. Questo comando scaricherà tutte le librerie necessarie e preparerà il progetto per l'esecuzione:

```
mvn clean install
```

- Una volta completata la compilazione, esegui il progetto utilizzando il comando Maven seguente:

```
mvn spring-boot:run
```

- **Accedi al Gioco:** Una volta che il server è avviato correttamente, puoi accedere all'interfaccia del gioco tramite il browser. Apri un browser e naviga all'indirizzo:

```
http://localhost:8080
```

MANUALE DELL'UTENTE

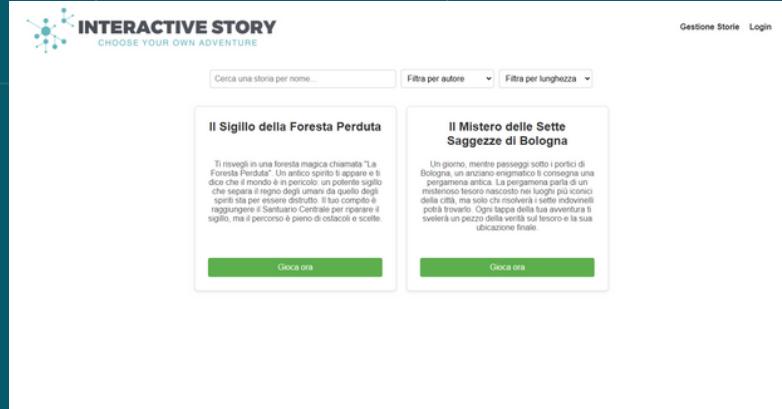
Benvienuto al manuale dell'utente per Interactive Story, un gioco interattivo in cui il giocatore esplora storie avvincenti prendendo decisioni che influenzano il corso della narrazione. Questo manuale ti guiderà attraverso i passaggi per accedere, giocare e completare il gioco.

Accesso al Gioco

Accedi al gioco tramite il browser all'indirizzo <http://localhost:8080>. La pagina principale mostra un elenco di storie disponibili, ognuna con un **titolo**, il **primo scenario** e il pulsante "**Gioca ora**". Puoi utilizzare la **barra di ricerca** al centro per filtrare le storie per nome.

Nel menu in alto a destra trovi le opzioni per **Gestione Storie** e **Login/Logout**, utili per amministrare le storie o accedere al sistema.

Il login è necessario per iniziare o continuare una partita. La navigazione è semplice e immediata, pensata per offrirti un'esperienza fluida.



Registrazione

Inserisci **email** e **password**, il sistema userà la parte iniziale dell'email come username. Clicca su "**Register**" per completare. Se hai già un account, usa il link "[Login here](#)".

Register

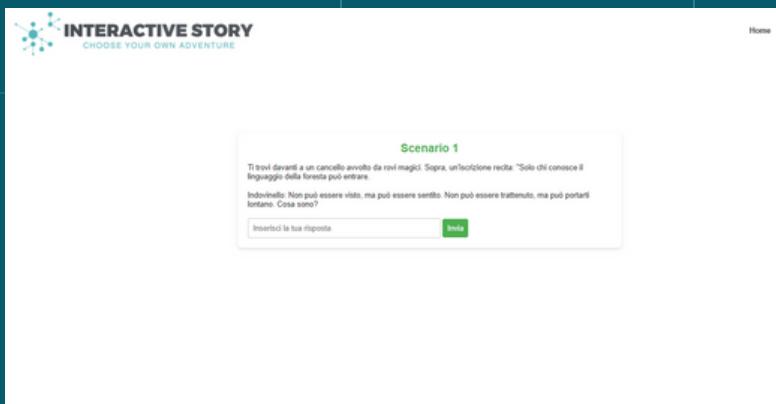
Already have an account? [Login here](#).

Login

Don't have an account? [Register here](#).

Login

Inserisci **email** e **password** per accedere al tuo account. Se non hai un account, usa il link "[Register here](#)" per registrarti.



Giocare una storia

Selezionando una storia, accedi a questa schermata che visualizza lo scenario attuale. Leggi attentamente la descrizione e l'indovinello, inserisci la risposta nel campo di testo e clicca su "**Invia**" per avanzare nel gioco. A seconda della risposta fornita, accederai allo scenario successivo specifico. Se desideri interrompere il gioco, puoi cliccare su "**Home**" nel menu in alto a destra.

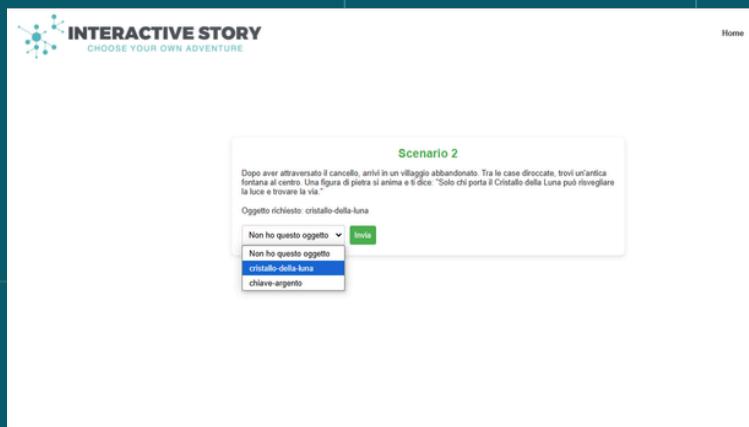
Oggetti trovati

Durante il tuo percorso, potrai trovare oggetti utili per proseguire nella storia. Quando ottieni un oggetto, apparirà un popup che ti avviserà del nuovo oggetto acquisito. Questo verrà automaticamente aggiunto e salvato nel tuo inventario, pronto per essere utilizzato al momento opportuno.

localhost:8080 says

Complimenti! Hai trovato "chiave-argento" durante il percorso. È stato aggiunto al tuo inventario!

OK

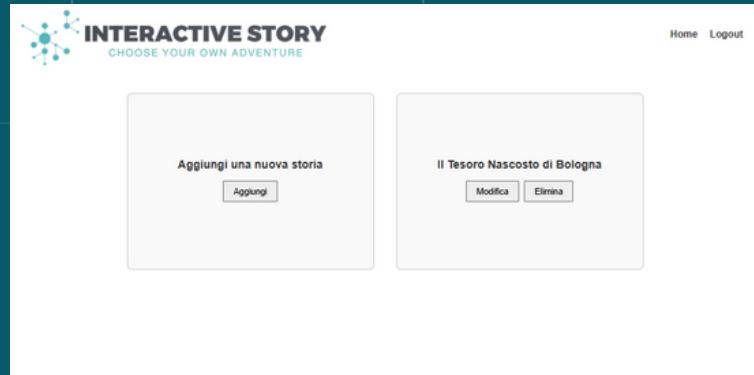


Giocare una storia

In alcuni scenari ti verrà richiesto un oggetto specifico per proseguire. Potrai facilmente selezionarlo dal tuo inventario. A seconda della situazione, se possiedi l'oggetto richiesto o meno, verrai indirizzato a uno scenario specifico. Il gioco continua seguendo questa logica fino a raggiungere la conclusione della storia.

Gestione storie

Dalla home page, cliccando sul menu "*Gestione Storie*" accedi a questa schermata. Qui puoi scegliere di modificare una delle storie che hai creato in precedenza oppure iniziare a creare una nuova storia da zero.



Creazione nuova storia

Cliccando su "*Aggiungi*", accedi a questa schermata per creare una nuova storia. Qui puoi inserire il titolo e la descrizione della storia. Successivamente, per ogni scenario, devi specificare l'ordine, la descrizione, il tipo di scelta e, se presente, l'oggetto trovato. Puoi aggiungere nuovi scenari cliccando su "*Aggiungi Scenario*" e, al termine, salvare la storia cliccando su "*Crea Storia*".

Modifica storia

Cliccando su "*Modifica*", accedi a questa schermata dove puoi aggiornare i dettagli di una storia esistente. Puoi modificare il titolo, la descrizione e aggiungere o aggiornare i dettagli degli scenari, come l'ordine, la descrizione, il tipo di scelta e l'oggetto trovato nello scenario. Al termine delle modifiche, puoi salvare i cambiamenti.

STRUMENTI & TECNOLOGIE



Nel progetto sono state utilizzate le seguenti tecnologie: **Spring Boot** è stato utilizzato per la gestione del progetto e lo sviluppo del backend, supportato dal linguaggio **Java** che ha garantito affidabilità e performance. Per il frontend sono stati adottati **JavaScript**, **HTML** e **CSS**, che hanno permesso di realizzare un'interfaccia utente interattiva. Il testing del codice è stato eseguito tramite l'utilizzo di **JUnit** per la creazione e l'esecuzione di test automatici. **Maven** per la gestione delle dipendenze e il processo di build del progetto, semplificando l'integrazione di librerie e strumenti esterni.

La gestione del database è stata effettuata con **Firebase Firestore**, database **NoSQL** per memorizzare storie, scenari e progressi degli utenti. Mentre per l'autenticazione degli utenti è stato usato Firebase, una soluzione sicura e facile da integrare. Per il versionamento del codice è stato utilizzato **GitHub** per facilitare il lavoro collaborativo, mentre **GitHub Projects** è stato impiegato per la gestione dei task e l'organizzazione delle attività del team.

MVC

Nel progetto è stato adottato il design pattern architetturale **MVC**(Model-View-Controller)

Il pattern architetturale **MVC** (*Model-View-Controller*) separa le responsabilità del software in tre componenti principali: **Model**, che gestisce la logica di business e i dati; **View**, che si occupa della presentazione e dell'interfaccia utente; **Controller**, che funge da intermediario, gestendo le interazioni tra Model e View

Le cartelle sono organizzate nel seguente modo:

- La cartella **Config** contiene tutte le configurazioni necessarie, inclusi servizi, routing e sicurezza, fornendo un punto centrale per la gestione degli aspetti configurabili del sistema.
- La cartella **Controller** ospita gli oggetti controller, che si occupano di gestire le richieste degli utenti e di orchestrare la logica tra la vista e il modello, in linea con il pattern MVC.
- Nella cartella **Model** sono presenti gli oggetti model, responsabili della rappresentazione della logica di business e dell'interazione con il livello di persistenza dei dati.
- La cartella **Servizi** include gli oggetti che gestiscono servizi specifici, supportando funzionalità trasversali e modulari all'interno del progetto.
- La cartella **Entity** contiene le entità principali del progetto, definite nella fase di inception, che rappresentano gli elementi fondamentali del dominio applicativo.
- La cartella **Resources** raccoglie gli oggetti di tipo view, utilizzati per gestire la rappresentazione visiva dell'applicazione, in linea con il pattern MVC.



SVILUPPO: DETTAGLI

Il progetto si basa su una struttura ben organizzata e modulare, progettata per supportare un sistema di storie interattive con una gestione completa di backend, frontend e database.

Il backend gestisce tutte le funzionalità critiche, tra cui la gestione delle storie e degli scenari, la persistenza dei dati, lo stato della sessione di gioco e l'esposizione di **REST API** per il collegamento con il frontend. Il sistema consente di creare, recuperare, modificare ed eliminare storie interattive, ognuna delle quali è composta da una serie di scenari interconnessi che definiscono il flusso narrativo. La persistenza dei dati è gestita utilizzando **Firebase Firestore**, un *database NoSQL* che garantisce un accesso rapido e scalabile ai dati relativi a utenti, storie, scenari e progressi del gioco.

Uno degli aspetti fondamentali del backend è la gestione della sessione e dello stato della partita. Grazie all'annotazione `@SessionAttributes` di Spring, il sistema può mantenere lo stato del gioco per l'intera durata della sessione utente, garantendo così un'esperienza di gioco fluida e coerente. Il backend espone diversi **endpoint REST** per consentire al frontend di interagire con il sistema. Questi includono funzionalità come la configurazione di una nuova partita (`/setup`), il salvataggio del progresso dell'utente (`/save-progress`), il recupero dei dettagli dello scenario corrente (`/current-scenario`), e la terminazione della partita (`/end`). L'architettura del backend è quindi progettata per essere altamente flessibile, consentendo l'integrazione di nuove funzionalità senza compromettere la stabilità del sistema.

Il progetto si sviluppa attorno a due classi chiave, **PartitaController** e **StoryManager**, che rappresentano il cuore della gestione del gameplay e della configurazione delle storie interattive. Queste classi, basate su Spring Boot, seguono un'architettura modulare che separa chiaramente le responsabilità tra la gestione della sessione di gioco e il controllo dei dati delle storie.

PartitaController

La classe PartitaController è responsabile della gestione dinamica delle sessioni di gioco. Utilizza Spring Session per mantenere lo stato della sessione tramite annotazioni come `@SessionAttributes`, che salvano istanze come **Partita** e **Giocatore** nel contesto della sessione. Questo consente di mantenere il progresso del giocatore tra le richieste HTTP, garantendo una navigazione coerente attraverso gli scenari della storia. L'inizializzazione di una partita avviene tramite il metodo `/setup`, che crea dinamicamente un'istanza di Partita, associa il giocatore attivo e carica gli scenari della storia dalla base dati utilizzando il modello **StoriaModel**. Il metodo verifica anche se esiste un progresso salvato, recuperandolo e ripristinando il gioco dallo stato precedente. Durante il gioco, il metodo `/play` permette di avanzare nello scenario basandosi sulle scelte dell'utente, utilizzando la logica encapsulata nella classe Partita, che interagisce con le entità **Scenario** e **Scelta** per determinare lo scenario successivo. Inoltre, il controller supporta il salvataggio dello stato corrente con il metodo `/save-progress`, che salva su Firebase informazioni come scenario corrente e inventario, utilizzando il modello StoriaModel.

StoryManager

La classe StoryManager si concentra invece sulla gestione statica delle storie e dei loro scenari. È progettata per fornire **funzionalità CRUD** alle storie, consentendo agli amministratori di creare, leggere, aggiornare e cancellare storie e scenari. Il metodo `/api/stories` fornisce un endpoint per ottenere una lista di tutte le storie, mentre `/api/stories/{id}` consente di filtrare per utente. Quando una nuova storia viene aggiunta, il metodo `/api/stories` utilizza **StoryManagerModel** per salvare i dettagli della storia e dei relativi scenari su Firebase. La logica è altamente modulare, con metodi dedicati alla gestione di dettagli specifici degli scenari, come *indovinelli* o *oggetti*, attraverso l'uso di modelli separati (**SceltaIndovinelloModel**, **SceltaOggettoModel**). La classe supporta anche la modifica di storie esistenti con il metodo `/api/stories/{id}/modify`, che aggiorna sia i metadati della storia sia i dettagli degli scenari collegati. L'approccio è scalabile e si basa su mappe di dati per consentire un'agevole manipolazione dei dettagli delle storie.

Entrambe le classi sfruttano appieno le capacità di **Spring Boot** per la gestione delle **richieste HTTP** e la modularità del progetto. Mentre **PartitaController** è focalizzata sulla logica runtime del gameplay, **StoryManager** gestisce l'aspetto amministrativo delle storie, mantenendo separati gli ambiti di responsabilità e rendendo il sistema flessibile e facilmente estensibile. L'integrazione con Firebase tramite i modelli aggiunge persistenza, garantendo che i dati delle storie e i progressi dei giocatori siano sempre sincronizzati con il backend.

Le entità del progetto sono progettate per rappresentare i concetti chiave del diagramma di dominio fornito e tradurli in classi Java strettamente correlate tra loro per realizzare la logica del gioco interattivo. Ogni classe rappresenta una parte significativa del dominio e interagisce con le altre tramite associazioni e metodi ben definiti.

Partita

La classe Partita rappresenta una sessione di gioco associata a un giocatore e a una storia. Include uno stato (**stato**) che indica se la partita è attiva o terminata, e un **inventario** che contiene gli oggetti trovati dal giocatore durante il gioco. L'oggetto **Scenario** rappresenta lo scenario corrente, che viene inizializzato tramite il metodo **setup**. Questo metodo recupera la storia e gli scenari da **StoriaModel**, configurando dinamicamente gli **indovinelli** o gli **oggetti** specifici per ciascuno scenario.

Il metodo **play** consente al giocatore di avanzare nello scenario rispondendo a un indovinello o interagendo con un oggetto. Utilizzando il **tipo di scelta** (Scelta), viene determinato il prossimo scenario. Inoltre, Partita gestisce il salvataggio dello stato tramite il metodo **saveStory**, che registra le informazioni su Firebase tramite **StoriaModel**.

Storia

La classe Storia rappresenta una raccolta di scenari che compongono una storia. Ogni istanza di Storia contiene dettagli come **titolo**, **descrizione**, e una **lista di scenari**. La configurazione degli scenari avviene attraverso il metodo **setScenari**, che utilizza ScenarioModel per recuperare i dettagli di ogni scenario da un backend remoto (Firebase). Ogni scenario è un'istanza della classe **Scenario**, che viene associata alla storia. La classe include metodi per accedere a uno scenario specifico tramite il suo ID (**getScenarioById**) o per recuperare il primo scenario della storia (**getPrimoScenario**). Questa struttura consente una navigazione sequenziale o dinamica tra gli scenari.

Scenario

Scenario rappresenta una tappa nella storia, con proprietà come **descrizione**, **ordine**, e **tipo di scelta** (tipoScelta). Questa classe gestisce due tipi principali di scelte, **SceltaIndovinello** e **SceltaOggetto**, che sono configurate dinamicamente tramite i metodi **getIndovinello** e **getOggetto**. La configurazione delle scelte avviene recuperando i dati da **SceltaIndovinelloModel** o **SceltaOggettoModel** e associandoli all'attributo scelte. Lo scenario può includere un oggetto che viene aggiunto all'inventario del giocatore quando raggiunto. La classe fornisce metodi per verificare se uno scenario ha scelte o oggetti e per recuperare queste informazioni.

Scelta

L'interfaccia Scelta definisce un **contratto comune** per tutte le tipologie di scelte che possono verificarsi all'interno del gioco. Essa include metodi come **esegui**, che implementa la logica per determinare quale scenario seguire sulla base della risposta dell'utente, e **testoDaMostrare**, che fornisce il contenuto da visualizzare all'utente. Inoltre, offre metodi per recuperare gli scenari corretti o errati, consentendo di gestire le diramazioni della storia.

SceltaIndovinello

La classe SceltaIndovinello implementa l'interfaccia Scelta ed è progettata per rappresentare una decisione basata sulla risoluzione di un enigma. Questa classe include un oggetto di tipo Indovinello, che contiene il testo e la risposta corretta dell'enigma, e due attributi che identificano lo scenario successivo in caso di risposta corretta o errata. Il metodo esegui confronta la risposta fornita dall'utente con quella corretta e restituisce l'ID dello scenario appropriato. Questa classe rappresenta quindi una logica deterministica basata sull'input dell'utente.

SceltaOggetto

D'altra parte, la classe SceltaOggetto rappresenta una decisione che si basa sul possesso di un determinato oggetto nell'inventario del giocatore. Gli attributi principali includono il nome dell'oggetto richiesto, lo scenario successivo se l'oggetto è presente e quello da seguire se l'oggetto è assente. Il metodo esegui verifica la presenza dell'oggetto richiesto nell'inventario del giocatore e restituisce l'ID dello scenario successivo basandosi sul risultato di tale controllo. Questa classe introduce una logica condizionale legata allo stato del giocatore.

Oggetto

La classe Oggetto rappresenta un'entità semplice ma fondamentale: un oggetto che il giocatore può ottenere nel corso della storia. Ogni oggetto è caratterizzato da un nome e un ID univoco generato tramite UUID. Questi oggetti sono utilizzati principalmente per determinare il percorso narrativo o per sbloccare specifiche scelte nel gioco.

Inventario

La classe Inventario funge da contenitore per gli oggetti raccolti dal giocatore. Essa è implementata come una lista di oggetti e fornisce metodi per aggiungere, rimuovere e verificare la presenza di un determinato oggetto. Inoltre, è possibile recuperare tutti i nomi degli oggetti presenti tramite il metodo getNomiOggetti. Questa classe rappresenta lo stato dinamico del giocatore e svolge un ruolo cruciale nella logica di molte scelte, in particolare quelle basate su SceltaOggetto. L'inventario è progettato per essere facilmente estendibile, permettendo una gestione efficiente degli oggetti nel contesto del gioco.

Sul lato frontend, il sistema utilizza una combinazione di HTML, CSS e JavaScript per fornire un'interfaccia utente interattiva. Il frontend comunica con il backend attraverso chiamate REST API, permettendo all'utente di interagire con il gioco in tempo reale. L'interfaccia presenta lo scenario corrente, incluse le descrizioni, le scelte disponibili e i feedback in base alle azioni dell'utente. Il file game.js gestisce la logica lato client, occupandosi del recupero dello scenario corrente dall'API /current-scenario, dell'invio delle risposte tramite /play, e del salvataggio del progresso dell'utente con /save-progress. Inoltre, il frontend permette la navigazione verso il menu principale e gestisce eventi come il clic su pulsanti e l'invio di input.

The background of the slide features a dark, abstract design composed of several large, semi-transparent, iridescent bubbles of various sizes. These bubbles have a metallic, liquid-like appearance with visible internal reflections and highlights. They are scattered across the frame, some overlapping each other. The overall effect is organic and fluid.

Progetto Ingegneria del Software

aa 2023-2024

06.12.2024

INTERACTIVE STORY.