# Descrizione del progetto

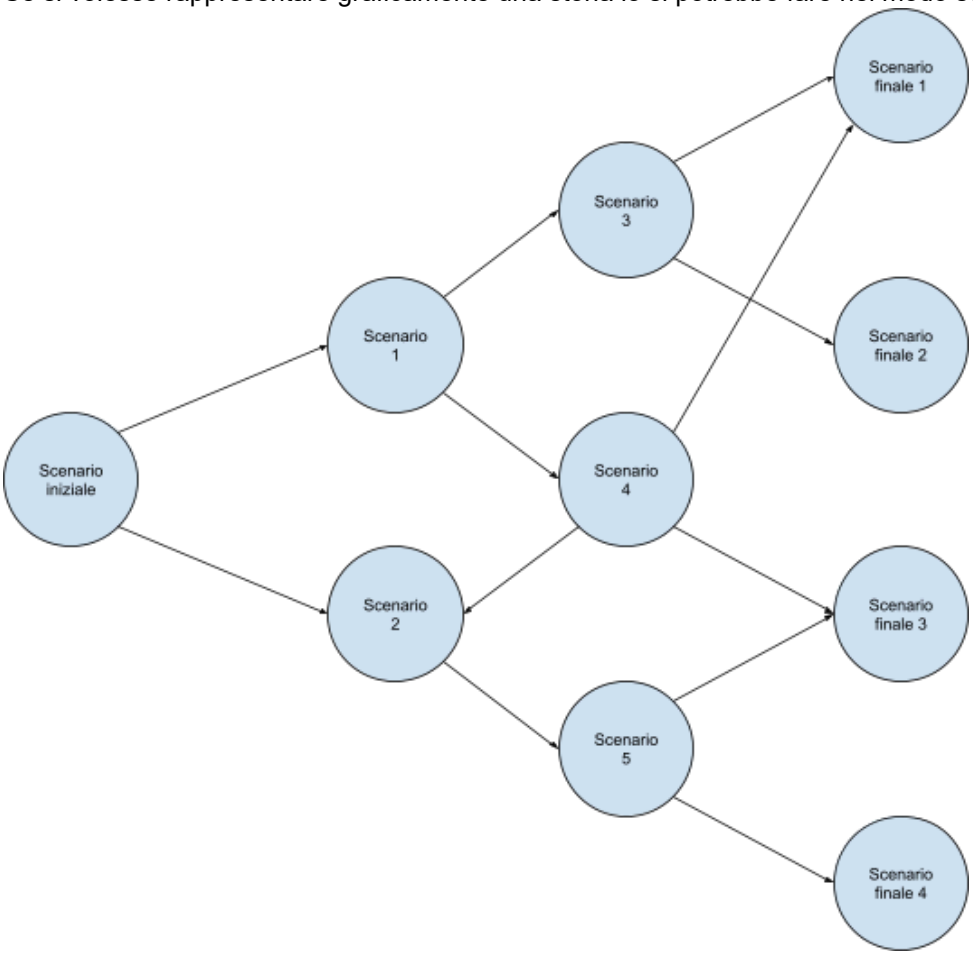
Versione 1.1 del 23/01/2024

**EDIT del 23/01/2024**: Aggiunta data di discussione per il mese di Aprile 2024

Il progetto per l'anno accademico 2023/24 consiste nella realizzazione di una applicazione web che permette agli utenti di scrivere e giocare a storie interattive. È un progetto da realizzare in gruppo. Ogni gruppo deve essere formato da 3/4 persone.

## Cosa si intende per storia interattiva

Una storia interattiva è una narrazione in cui il giocatore può prendere decisioni che influenzano lo sviluppo della storia. Il giocatore si trova di fronte a una sequenza di scenari, ognuno dei quali presenta una o più scelte. La scelta effettuata determina il passaggio a un nuovo scenario, che può essere diverso a seconda della scelta effettuata.
Ogni storia interattiva ha un inizio, un numero indefinito di possibili diramazioni e uno o più finali.
La storia è scritta da uno scrittore, che crea gli scenari e le regole che governano le scelte del giocatore.
Se si volesse rappresentare graficamente una storia lo si potrebbe fare nel modo seguente:




## Specifiche

Nello specifico, in questo progetto si prendono in considerazione i seguenti scenari per individuare le funzionalità che vengono offerte agli utenti:

**Scrittura storie**: il sistema deve permettere ad un utente di creare nuove storie potendo specificare 2 o più alternative su dove/come il personaggio giocante possa procedere oppure chiedere di risolvere un indovinello testuale o numerico. Ogni storia deve avere un inizio, delle diramazioni e uno o più finali. Inoltre, durante alcuni scenari deve essere possibile raccogliere degli oggetti che vengono aggiunti all'inventario del protagonista e che possono essere richiesti per accedere ad alcune diramazioni.

**Giocare a una storia**: il sistema deve permettere ad un utente di giocare ad una storia. Giocare ad una storia vuol dire leggere lo scenario e, in base a questo, prendere una decisione (alcune delle quali possono essere

precluse in caso non si abbia uno specifico oggetto nell'inventario) o rispondere ad un indovinello (testuale o numerico). In alcuni casi sarà possibile scegliere di raccogliere degli strumenti che verranno aggiunti ad un inventario. Dipendentemente dalla scelta presa o dalla risposta data viene mostrato un altro scenario. Il ciclo continua finché non si raggiunge il finale della storia.

**Gestione delle storie scritte**: il sistema deve permettere di modificare il testo che descrive lo scenario di una storia. La varie diramazioni non possono essere modificate.

**Gestione delle storie giocate**: il sistema deve permettere di interrompere e salvare lo stato di una storia che si sta giocando e cancellare una storia in corso.

**Ricerca/visualizzazione storie**: il sistema deve permettere ad un visitatore di ricercare le storie e visualizzare il primo scenario senza però poter prendere alcuna scelta.

**Registrazione**: il sistema deve permettere ad un visitatore di registrarsi. Una volta registrato un utente può scrivere delle storie.

[Opzionale per i gruppi da 3, obbligatorio per quelli da 4] Per poter anche giocare alle storie, in fase di registrazione (o successivamente) deve essere eseguito il pagamento una tantum utilizzando un servizio esterno (vedere appendice).

**Persistenza dei dati**: il sistema deve permettere di salvare i dati degli utenti e di recuperarli anche dopo essere stato spento e riavviato

Nel dettaglio, l'applicazione deve permettere le seguenti operazioni:

Registrazione, login e logout di un utente. Un utente deve avere almeno le seguenti proprietà:

- Username
- Password

[Opzionale per i gruppi da 3, obbligatorio per quelli da 4] Ci sono due tipi di utente registrato: con la semplice registrazione un utente può solamente scrivere delle storie. Per poter giocare alle storie deve essere fatto un pagamento una tantum utilizzando un servizio esterno (vedere appendice). Il pagamento potrà andare a buon fine oppure no. In caso di fallimento del pagamento deve essere possibile riprovare.

Ricerca/browsing di storie dal catalogo anche applicando dei filtri (username dello scrittore, lunghezza della storia, …).

Scrittura storie: questa funzionalità permette ad un utente di scrivere una nuova storia che verrà aggiunta al catalogo. Una storia è formata da più scenari. In ogni scenario deve essere possibile raggiungerne almeno un altro (fatta eccezione per gli scenari finali). Il collegamento da uno scenario a un altro può essere fatto con:

- Semplice scelta del giocatore
- Indovinello con soluzione testuale/numerica
- Scelta del giocatore dipendente dall'avere o meno un oggetto nell'inventario. Deve quindi essere possibile raccogliere oggetti

La modifica di una storia è permessa solamente per la parte testuale. La struttura delle scelte non può essere modificata.

Giocare una storia: un giocatore può scegliere quale storia giocare dal catalogo. Una volta scelta una storia questa viene presentata mostrando il testo che descrive il primo scenario e le opzioni per proseguire. All'input dell'utente ci sarà un cambio di scenario. Il loop continua finché il giocatore non raggiunge uno scenario finale. A quel punto la partita è conclusa. Deve essere possibile interrompere una giocata e ricominciarla in un secondo momento. Si possono giocare più storie in contemporanea.

È anche possibile interrompere una storia eliminando la partita.

# Tecnologie

La seguente lista specifica quali sono le tecnologie che è obbligatorio utilizzare per svolgere il progetto.

Git (https://git-scm.com/) → per il versionamento dei file

GitHub (https://github.com/) → come repository remoto, issue tracking system, project management board

[Opzionale per i gruppi da 3, obbligatorio per quelli da 4] GitHub Actions→ CI/CD pipeline. Alcune idee per cosa utilizzarla sono:

- Automazione dell'esecuzione dei test di unità
- Controllo di formattazione e regole decise del gruppo riguardo al codice sorgente
- Creazione automatica di un eseguibile

Strumenti alternativi possono essere adottati previa richiesta e autorizzazione in quanto quelli specificati vengono usati per la consegna e la valutazione.

Il gruppo può scegliere quali linguaggi, framework, e librerie utilizzare per la realizzazione del progetto.

Uno possibile stack tecnologico, e per il quale sarà fornita assistenza specifica, è il seguente:

Java

Maven (https://maven.apache.org/) → per la gestione delle dipendenze e del processo di build del progetto

GWT (https://www.gwtproject.org/) → per la realizzazione delle pagine web

MapDB (https://mapdb.org/) → persistenza dei dati

Gson (https://github.com/google/gson) → interazione con JSON

JUnit (https://junit.org/junit5/) → per la realizzazione di unit testing

Nell'appendice ci sono alcuni esempi che si possono seguire utilizzando questo stack tecnologico

# Modello di processo

Il modello di processo suggerito è un ibrido strutturato-agile composto da due fasi:

- **Inception**, in cui vengono prodotti almeno i seguenti artefatti:
  - Modellazione dei casi d'uso
  - Modello di dominio
  - Glossario
- **Construction**, in cui si eseguono le pratiche di gestione di processo di Scrum.
  I componenti del gruppo si immedesimano a turno in (tra parentesi quadre il numero di membri che devono immedesimarsi nel ruolo):
  - Development team [2]
  - Scrum master [1 se il gruppo è da 4, 1 che farà anche parte anche del development team se il gruppo è da 3]
  - Product owner [1]

  I ruoli si cambiano al termine di ogni sprint.
  Devono essere rispettati gli eventi del framework Scrum:
  - Sprint planning
  - Daily scrum
  - Sprint review
  - Sprint retrospective

  E prodotti gli artefatti standard:
  - Product backlog
  - Sprint backlog
  - Burn down chart

Durante la fase di construction, il team di sviluppo dovrà adottare il workflow feature branch facendo uso delle pull request di GitHub.

# Product backlog e Sprint backlog

Per la gestione del lavoro da svolgere si useranno le board messe a disposizione da GitHub (i Projects).
È quindi necessario creare un nuovo project e collegarlo al repository. La board sarà successivamente popolata da quelli che GitHub chiama "item".
Il product backlog (colonna Product Backlog della board) sarà popolato dagli Use Cases individuati durante la fase di inspection. Questi non devono essere trasformati in issue vere e proprie. Quando, durante lo Sprint Planning, si decide che si vuole lavorare ad uno Use Case, questo viene raffinato in task veri e propri (che in questo caso corrispondono alla realizzazione delle funzionalità a supporto dei diversi scenari relativi al caso d'uso, i cosiddetti "use case slice") e le varie issue devono essere create.

Esempio stato iniziale della board

| UC raffinati | Product Backlog | Sprint Backlog | … |
|---|---|---|---|
| | UC-1 | | |
| | UC-2 | | |

Esempio stato della board dopo lo sprint planning

| UC raffinati | Product Backlog | Sprint Backlog | … |
|---|---|---|---|
| UC-1 | UC-2 | Task 1 per UC-1 | |
| | | Task 2 per UC-1 | |

Dove la colonna "UC raffinati" serve solo per mantenere uno storico dei casi d'uso processati durante l'avanzamento del progetto.

# Prodotti da realizzare

Dalle fasi di inception e production si devono consegnare i seguenti artefatti:

- **Modello dei casi d'uso** (con la **descrizione** di ognuno di essi)
- **Modello di dominio**
- **Burn down chart**

Sono inoltre richiesti i seguenti documenti:

- **Manuale utente:** questo documento deve contenere una guida accessibile ad un utente inesperto che spieghi come utilizzare il prodotto realizzato.

**Manuale dello sviluppatore:** questo documento deve contenere le istruzioni su come installare e lanciare su un nuovo computer il software sviluppato a partire dall'ottenimento del codice sorgente. Inoltre, deve contenere una panoramica che documenti come sono state realizzate le funzionalità richieste; in particolare se sono stati individuati e utilizzati dei design pattern, specificando quali vantaggi ha portato la loro adozione.
**Diario del progetto**: questo documento deve riportare in modo sintetico come il progetto è avanzato nel tempo e come il gruppo si è organizzato. In particolare deve essere riportato:

- Inizio e fine delle iterazioni e degli sprint
- Chi ha fatto quale ruolo durante lo sprint
- Stato della board a inizio e fine sprint. Questo permette di capire la vostra pianificazione e il vostro riuscire a seguire quanto pianificato

# Testing dell'applicazione

È richiesto che vengano implementati i test di unità per l'applicazione.
Potete adottare l'approccio TDD (Test Driven Development).

Non è richiesto di implementare i test per il codice che si occupa della gestione dell'interfaccia grafica.

# Modalità e tempi di consegna

Orientativamente le date (pubblicate almeno un mese prima) per la discussione saranno nei mesi di:

- Febbraio 2024
- Aprile 2024
- Giugno 2024
- Luglio 2024
- Settembre 2024
- Novembre 2024
- Dicembre 2024

Nel caso di necessità particolari potranno essere concordate date aggiuntive.
Dopo Dicembre 2024 le specifiche del progetto cambieranno.

La data ultima per la consegne è una settimana prima della data della discussione e deve avvenire:

- Creando una "Release" su GitHub nella quale, oltre ad essere presente il codice sorgente, devono essere inseriti anche tutti i documenti redatti ed esportati in formato PDF. Fare riferimento all'appendice se non si sa come creare una release;
- Inviando una mail a marco.ferrati2@unibo.it (assicurandosi di mettere in CC tutti i componenti del gruppo) in cui è incluso il link al progetto;
- Invitando al repository di GitHub l'utente "jjocram".

Successivamente, ogni gruppo verrà convocato per discutere il progetto nelle date che verranno rese note di volta in volta su Virtuale.

Durante la discussione verrà richiesta una demo del prodotto realizzato.

## Valutazione

La valutazione del progetto avverrà tenendo conto dei seguenti punti:

- La realizzazione delle specifiche funzionali
- L'adeguatezza dello stack tecnologico scelto
- La semplicità di "deploy" dell'applicazione, inteso come numero di step necessari per avviare l'applicazione su una macchina che non ha mai eseguito il progetto
- L'organizzazione, la leggibilità e la qualità del codice
- L'adozione di una metodologia di lavoro e l'utilizzo appropriato degli strumenti di testing, versioning, bug tracking, code reviewing e project tracking
- L'adeguatezza della documentazione allegata al progetto e in particolare della descrizione del processo di sviluppo adottato
- La discussione del progetto

# Consigli vari ed eventuali

- In caso di dubbi o perplessità utilizzate il forum di Virtuale o mandate una mail a marco.ferrati2@unibo.it
- Per la stesura dei documenti potete utilizzare Google Doc, Word, Markdown, Asciidoc, LaTeX, …
- Per la realizzazione dei diagrammi potete utilizzare un software specifico (plantUML, starUML, …) oppure software generico per la creazione di diagrammi (draw.io, Miro, …)
- Fate richiesta di accesso al GitHub Student Developer Pack per avere accesso a tutte le feature di GitHub (https://education.github.com/pack#offers)

Se volete condividere risorse online assicuratevi che siano raggiungibili da un utente che non è registrato al servizio che state usando (fatta eccezione per GitHub e OneDrive dell'università)

# Appendice

## Stack tecnologico proposto

Come punto di partenza per l'implementazione si può prendere spunto da questo repository: https://github.com/jjocram/gwt-stockwatch-maven

- Non è per forza necessario usare l'archetipo usato nel progetto presente nel repository d'esempio
- Oltre all'archetipo utilizzato nel repository d'esempio potete utilizzare il webAppCreator di GWT assicurandovi di creare un progetto Maven: https://www.gwtproject.org/doc/latest/RefCommandLineTools.html#webAppCreator

Se si vuole eseguire un'operazione all'avvio del server Jetty, è necessario creare una nuova classe che implementi l'interfaccia ServletContextListener e aggiungere alla configurazione del web server il riferimento all'implementazione del listener.

- Esempio implementazione listener
- Esempio configurazione

L'interazione con il sistema di pagamento può essere fatta senza l'uso di librerie esterne (nel caso vogliate usarne siete liberi di farlo). Un esempio in puro Java lo si può trovare al seguente link: https://gist.github.com/jjocram/d6cedb62cc8b49a46fb61b47e59d3aac

## Sistema di pagamento

Il sistema di pagamento è un servizio esterno con il quale è richiesta l'interazione. Viene fornito un file JAR eseguibile (https://gitlab.com/jjocram/naive-payment-provider//uploads/5db8e9d22c3ff1fa274318cdc11e8431/NaivePaymentProvider-1.1.0.jar) con il quale è possibile avviare tale servizio e che sarà eseguito in locale sulle vostre macchine. Eseguitelo da terminale (`java -jar NaivePaymentProvider-1.1.0.jar`) altrimenti per terminarlo dovrete utilizzare un gestore di processi e non potete averne più di uno attivo alla volta. Il web-service sarà raggiungibile all'indirizzo http://localhost:6789/

Il sistema di pagamento è un web-service scritto in Java e in ascolto sulla porta `6789` della vostra macchina. L' end-point da usare è `/pay` e accetta in input, come corpo della richiesta, un JSON e ne restituisce un altro con l'esito della transazione (50% successo, 50% fallimento).
La struttura dei JSON con i quali interagire, istruzioni su come avviare il servizio, e output nel caso di errori sono meglio descritti nel repository al seguente link: https://gitlab.com/jjocram/naive-payment-provider
Nota: non è come realmente funzionano i payment provider. Questa è una semplificazione a solo scopo didattico.

## Release

Per creare una release su GitHub:

- Aprire la pagina principale del repository
- Nella colonna di destra individuare la sezione "Releases"
- Cliccare su "Create new release"
- Compilare i vari campi testuali richiesti
- Allegare i file PDF utilizzando l'apposita area (quella con una freccia che punta verso il basso)
- Cliccare su "Publish release"