Report fase di Pre-Gaming

Gruppo: 7
Membri del gruppo: Giorgia Postiglione, Giuseppe Ruggiero, Andrea Tudino, Andrea Vitolo
Trello Board: <https://trello.com/b/dAKRhXb6/ingsoftware>
GitHub: <https://github.com/Tyusibo/IngSoftGruppo07IZ>

Introduzione

Abbiamo supposto che Andrea Tudino ricopra per questa fase il ruolo di product owner; infatti, hai esposto al team le prime tre User Epic. Successivamente sia il team di sviluppo che il product owner hanno scomposto le user Epic in User Stories con più fasi di affinamento di queste ultime. In seguito, abbiamo proseguito con una fase di stima di delle User Stories, in particolare abbiamo scelto di stimare innanzitutto i value points di ogni user story, valutando in particolare il valore che ognuna avrebbe portato all'utente finale ed utilizzando la metrica basata sulla serie di fibonacci, in modo da rendere evidente le differenze tra le varie User Stories. Successivamente il team di sviluppo ha stimato gli story points utilizzando la stessa metrica sopra definita. Infine, il team ha calcolato il BFTB (Bang For The Buck) per dare una misura del valore in relazione allo sforzo (rapporto tra value points e story points).

Architettura

Il nostro sistema è suddiviso in tre componenti fondamentali, ciascuna con un ruolo ben definito:

- **Modelli:** I modelli costituiscono il cuore dell'applicazione, assumendo la responsabilità di eseguire azioni impartite dal controllore. Ricevono input dal controllore, elaborano i dati in modo appropriato e gli restituiscono i risultati.
- **Vista:** La vista rappresenta l'interfaccia utente, permettendo all'utente di inserire input. Viene aggiornata dal controllare. La sua responsabilità è presentare i dati in un formato comprensibile e garantire un'esperienza utente intuitiva.
- **Controllore:** Il controllore svolge un ruolo centrale, facilitando la comunicazione tra tutte le componenti del sistema. Coordina gli input utente, gestisce le richieste dei modelli e orchestra l'interazione tra le diverse parti del sistema. Si occupa di interpretare gli input dell'utente, aggiornare i modelli di conseguenza e comunicare con la vista per riflettere i cambiamenti.

Dall'analisi delle componenti emerge quindi chiaramente che l'architettura più adatta alle esigenze del sistema da implementare è del tipo MVC (Model-Control-View).

User stories prodotte in questa fase

Al seguente link nel foglio *User Stories Pre-gaming* è allegata la tabella contenente tutte le User Stories estrapolate durante questa fase, ovvero quelle derivanti dalle prime tre User Epics.

https://docs.google.com/spreadsheets/d/1jYiJdclKfv4hlbiJVQQ3ZSFj0A611gO-KJ7naodZIP0/edit?usp=sharing

Definition of Done

- Coding convention definite nel pre-gaming rispettate.
- Il codice deve essere commentato seguendo la convenzione javadoc.
- La documentazione deve essere aggiornata.
- I test unitari devono essere eseguiti con esito positivo.
- I test di integrazione devono essere eseguiti con esito positivo.
- I test di regressione devono essere eseguiti con esito positivo.
- Tutti i criteri di accettazione sono soddisfatti.

Language/Development Environment

- **Linguaggio:** Java
- **UI Framework:** JavaFX (con FXML)
- **UI Design Tool:** Scene Builder
- **IDE:** NetBeans
- **Project Management Tool:** Trello
- **Version Control System:** Git (tramite GitHub)

Coding convention

Durante questa fase, sono state anche definite le coding convention necessarie a garantire uniformità, chiarezza e manutenibilità del codice all'interno del progetto. In particolare, il team ha deciso di:

- **Utilizzare il camelCase** per la scrittura di nomi di variabili e metodi, adottando la convenzione.

- **Scrivere il codice in lingua italiana**, inclusi nomi di classi, metodi, variabili e costanti, per migliorare la comprensione da parte di tutto il team.

- **Adottare la documentazione con Javadoc**, così da garantire una descrizione formale e strutturata e facilitando la manutenzione a lungo termine.

In più, verranno applicate le buone pratiche di scrittura del codice quali nomi significativi per variabili, classi e metodi e organizzazione ordinata del codice, raggruppando attributi, costruttori e metodi in sezioni distinte e mantenendo una corretta indentazione.