Testing: Sala Giochi Virtuale

Progetto: Sala Giochi Virtuale
Data: 12 Gennaio 2026
Team di Progetto: Team UNI
Corso: Ingegneria del Software – Università degli Studi di Bergamo

Contenuto:

- Modelling
- Software Architecture
- Software Design Pattern

1. Software Testing Strategy (Piano di Test)

1.1 Obiettivi

L'attività di testing ha l'obiettivo di verificare la conformità del software rispetto ai requisiti funzionali definiti e di garantire la robustezza del codice prima del rilascio finale. Inoltre questa fase risulta essenziale durante l'aggiunta di nuove funzionalità, per evitare di introdurre nuovi bug in un sistema funzionate.

1.2 Strategia Ibrida

Data la natura del progetto (Applicazione Desktop *JavaFX* con *database embedded*), è stata adottata una strategia di testing ibrida:

- **Automated Unit Testing (White Box):** Utilizzato per il controllo dell'autenticazione e della gestione del database sull'assegnamento dei punteggi.
    - *Framework:* **JUnit**
    - *Target:* Package `.db` e `.videogame` (logica).
- **Manual Functional Testing (Black Box):** Utilizzato per verificare il corretto funzionamento dell'interfaccia grafica (messaggi di errore, pop-up e comportamento dei pulsanti), il controllo della logica dei giochi e la corretta comunicazione tra GUI, logica e database.
    - *Target:* Package `.gui` e `.videogame` (grafica).

2. Unit Testing (Casi di Test Implementati)

2.1 Descrizione dei Test Case

- *LoginTest:* Test case che confronta l'ugualianza tra il valore "oracolo" preso dal DB e quello restituito dai metodi della classe *User,* sia per il campo *Password* che per il *NickName.*
- *scoreTest:* Test case per controllare che il valore del punteggio assegnato ad un gioco corrisponda con il valore restituito dall'istanza dello stesso.

2.2 Implementazione (Codice JUnit)

Test Suite: UserTest

```
private User u = new User("CiaoSonoAndre", "Andrea", "xxx");

@Test
void correctPasswordLogin() {          // Test case: Login Riuscito
        assertEquals("xxx", u.getPassword() );
}
```

Test Suite: GameLogicTest.java

```
private Battleship b = new Battleship("BattleShip", new Category("Strategia"));

@Test
void scoreTest() {                     // Test case: score del punteggio non corretto
        assertEquals(100, b.getScore());
}
```

3. Manual Testing (GUI & Integrazione)

Per coprire le parti non testate automaticamente (Package GUI), sono stati eseguiti dei test manuali con l'obiettivo di valutare il comportamento dell'applicazione in diversi scenari.

Scenario MT-01: Flusso di Registrazione e Login

1. **Azione:** Avviare l'app e procedere alla registrazione.
2. **Input:** Inserire i dati di Nickname, Password e conferma Password.
3. **Verifica:** Il sistema chiude il popup e mostra messaggio successo.
4. **Azione Successiva:** Login con le stesse credenziali.
5. **Risultato Atteso:** Login effettuato.
6. **Esito:** PASS

Scenario MT-02: Gestione Partita e Classifica

1. **Azione:** Avviare una partita di gioco e vincerla.
2. **Verifica:** Appare popup "Hai Vinto!".
3. **Azione Successiva:** Andare nella stanza "Tabellone" a controllare l'aumento del punteggio associato al profilo con il quale l'utente sta giocando.
4. **Risultato Atteso:** Comparsa del pop-up di vittoria e nella tabella del punteggio compare il punteggio aggiornato sommato con i punti del gioco vinto
5. **Esito:** PASS

- ***Precisazione***: è stato svolto il medesimo scenario con l'obiettivo di perdere una partita al fine di verificare che in caso di sconfitta non vengano modificati i punti. Queste verifiche sono riferite a tutte le tipologie di giochi esclusa la Roulette.

Scenario MT-03: Gestione Partita Roulette caso punti insufficienti

1. **Azione:** Avviare una partita al gioco Roulette.
2. **Verifica:** Uscita dal gioco dopo la comparsa del pop-up di errore.
3. **Azione Successiva:** Convertire i punti in gettoni per poter iniziare il gioco.
4. **Risultato Atteso:** Uscita dal gioco preceduta dal messaggio di errore.
5. **Esito:** PASS

Scenario MT-04: Gestione Partita Roulette e controllo punteggio

1. **Azione:** Avviare una partita al gioco Roulette e perderla.
2. **Verifica:** Corretta conversione dei punti in gettoni e decremento dei punti del giocatore.
3. **Azione Successiva:** Dopo aver effettuato la puntata di gioco controllate il valore del punteggio nel “Tabellone”.
4. **Risultato Atteso:** Visto che non è stata vinta la partita non ci devono essere stati aumenti di punteggio ma il punteggio deve risultare inferiore a causa del costo di conversione del gettone utilizzato durante la partita (e perso)
5. **Esito:** PASS

- ***Precisazione***: è stato svolto il medesimo scenario con l’obiettivo di vincere la partita al fine di verificare la corretta variazione dei punti (inizialmente decrementati per l’acquisto dei gettoni e successivamente incrementati dalla vittoria)