



UNIVERSITÀ
DEGLI STUDI
FIRENZE

Advanced Programming Techniques

SVILUPPO DI UN SEMPLICE SISTEMA DI
SCOMMESSE CALCISTICHE

MANUEL DRAGO

Academic Year 2022-2023

INDICE

1	Applicazione realizzata	5
1.1	Scelte implementative e di design	5
2	Tecniche e framework utilizzati	7
3	Descrizione dello sviluppo e del testing	9
3.1	Problemi riscontrati	11
4	Istruzioni d'uso	13

ELENCO DELLE FIGURE

Figura 1	GUI	6
Figura 2	Add abilitato	10
Figura 3	Delete abilitato	10
Figura 4	Change Odds abilitato	11

APPLICAZIONE REALIZZATA

L'applicazione sviluppata riguarda la gestione di un semplice sistema di scommesse calcistiche per la creazione di un **palinsesto di eventi sportivi**. Ogni evento è caratterizzato da una quadrupla: [*squadra di casa, squadra di trasferta, esito, quota*].

Tramite l'interfaccia grafica dell'applicazione (fig. 1) si potranno aggiungere eventi al palinsesto, eliminarli e modificarli andando ad agire sulla sola variabile *quota*, un parametro in continuo cambiamento nell'ambito delle scommesse.

1.1 SCELTE IMPLEMENTATIVE E DI DESIGN

L'applicazione è stata progettata sulla base del pattern architetturale **Model-View-Presenter**, una variante del **Model-View-Controller**.

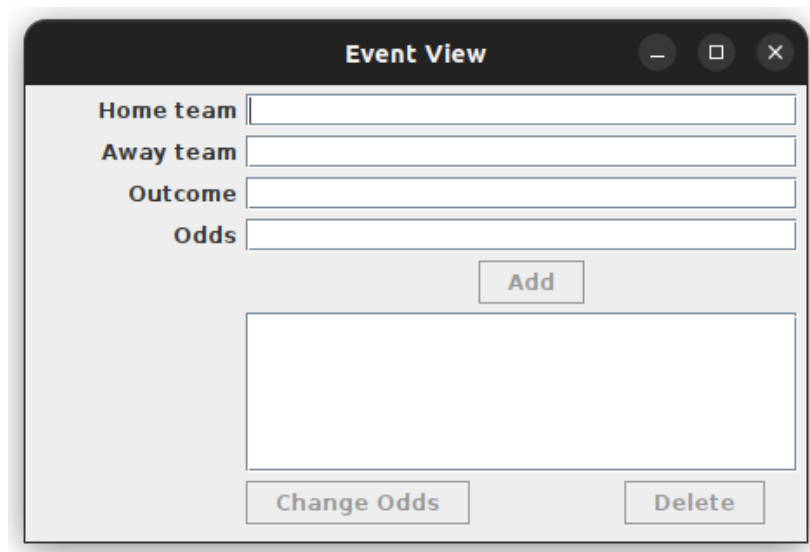
Nel mio contesto, il **domain model** è rappresentato dalla sola classe *Evento*. Gli eventi verranno collezionati all'interno del palinsesto, ovvero la **repository**, la quale potrà essere modificata dal *controller* che agisce da intermediario tra essa e la *view*.

Come accennato poc'anzi, tramite la *view* (l'interfaccia utente) è possibile compiere tre azioni:

1. aggiunta di un evento al palinsesto;
2. rimozione di un evento dal palinsesto;
3. modifica della quota di un evento nel palinsesto.

L'aggiunzione di un evento è possibile se e solo se la quota è superiore ad 1.0 e nel palinsesto non esiste già un evento con la stessa tripla [*squadra di casa, squadra di trasferta, esito*] (è invece possibile immettere una stessa partita associandole esiti diversi).

Similmente, la modifica della quota di un evento è possibile se e solo se la nuova quota è superiore ad 1.0.



The image shows a window titled "Event View" with standard window controls (minimize, maximize, close) in the top right corner. The window contains a form with four input fields labeled "Home team", "Away team", "Outcome", and "Odds". Below these fields is an "Add" button. At the bottom of the window, there are two buttons: "Change Odds" on the left and "Delete" on the right. A large, empty rectangular box is positioned between the "Add" button and the bottom buttons.

Figura 1: GUI

Per **scelta implementativa**, si decide di non notificare l'utente circa il successo o il fallimento delle azioni effettuate: semplicemente, se l'utente esegue un'azione "legittima" essa viene effettuata, altrimenti non accade nulla.

TECNICHE E FRAMEWORK UTILIZZATI

L'applicazione è stata realizzata utilizzando l'approccio **TDD** appreso durante il corso, seguendone strettamente le tre leggi e rispettando la piramide dei test. Nell'ultima parte dello sviluppo è stato anche integrato l'approccio **BDD**, per la scrittura degli *e2e* test.

Di seguito viene presentato un elenco di tutti gli strumenti (framework, piattaforme, librerie, ecc.) utilizzati nel progetto.

- JUnit 4
- JaCoCo
- Coveralls
- PIT
- Maven
- Mockito
- Git
- GitHub
- GitHub Actions
- Docker
- MongoDB
- Testcontainers
- AssertJ Swing
- Cucumber
- SonarCloud

DESCRIZIONE DELLO SVILUPPO E DEL TESTING

Gli aspetti più interessanti del testing riguardano la *repository* e la *view*.

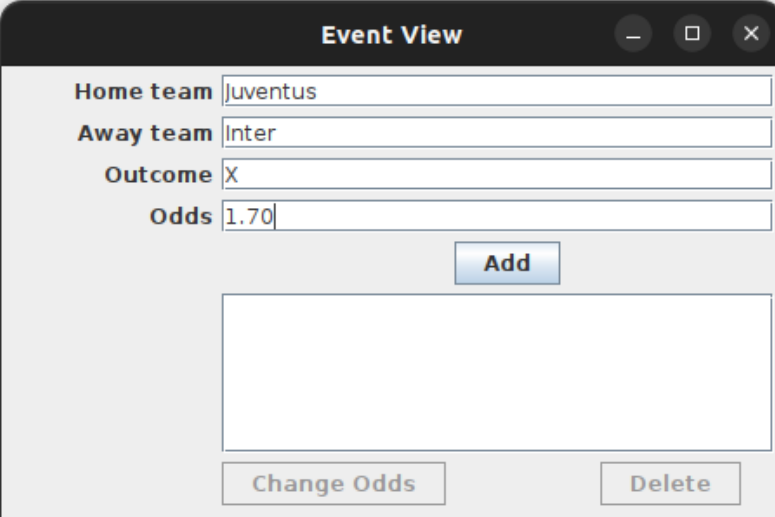
Per quanto concerne la prima, sono stati implementati degli **unit** test per verificare la corretta manipolazione del palinsesto tramite le azioni specificate. A tal proposito, è stato utilizzato un database in-memory come **fake** per non interagire con un server reale, consentendo dunque di testare le funzionalità implementate in isolamento.

Per quanto riguarda la seconda, sono stati scritti degli **unit** test per verificare il corretto funzionamento dell'interfaccia grafica e la delegazione di tutte le azioni al *controller*. I vincoli imposti all'interfaccia grafica sono i seguenti:

1. il bottone *Add* deve attivarsi solo quando tutte le caselle testuali contengono qualcosa che non sia solo spazi vuoti (fig. 2);
2. il bottone *Delete* deve attivarsi solo quando viene selezionato un evento presente nella lista (fig. 3);
3. il bottone *Change Odds* deve attivarsi solo quando viene selezionato un evento presente nella lista e la casella testuale relativa alla quota contenga qualcosa che non sia solo spazi vuoti (fig. 4).

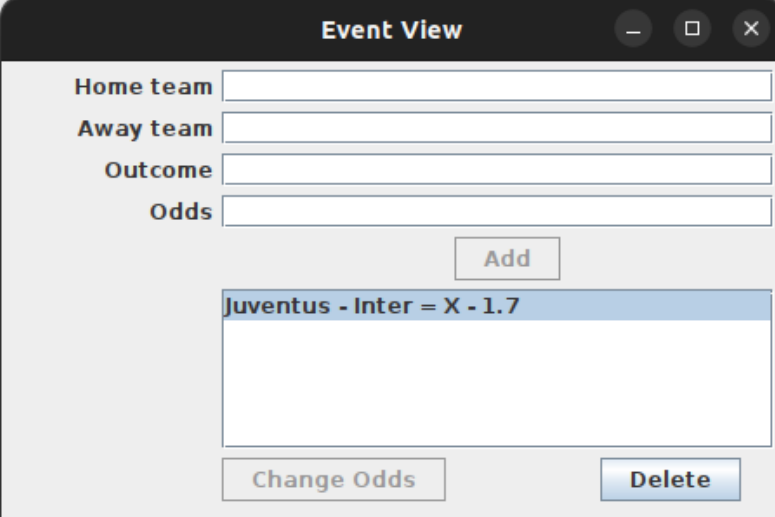
Un altro aspetto interessante riguarda l'utilizzo della libreria *Testcontainers* negli **integration** test che coinvolgono il *controller* e il database reale. Grazie ad essa è possibile scrivere dei test *self-contained*, ovvero che non dipendono da risorse esterne.

Infine, la parte più importante riguarda la scrittura degli **e2e** test, implementati tramite l'utilizzo di *Cucumber*. Gli scenari testati mirano a verificare l'effettiva aggiunta, rimozione e modifica della quota di un evento facendo asserzioni sulla sola interfaccia utente, senza basarsi sui dettagli implementativi interni (**black box testing**).



The 'Event View' window features a dark header with the title 'Event View' and standard window controls. Below the header, there are four input fields: 'Home team' (containing 'Juventus'), 'Away team' (containing 'Inter'), 'Outcome' (containing 'X'), and 'Odds' (containing '1.70'). An 'Add' button is positioned to the right of the 'Odds' field. Below these fields is a large, empty rectangular box. At the bottom of the window, there are two buttons: 'Change Odds' and 'Delete'.

Figura 2: Add abilitato



The 'Event View' window is shown in a state where the 'Delete' button is enabled. The input fields for 'Home team', 'Away team', 'Outcome', and 'Odds' are now empty. The 'Add' button is disabled. The large rectangular box below the input fields now contains the text 'Juventus - Inter = X - 1.7'. The 'Change Odds' and 'Delete' buttons remain at the bottom.

Figura 3: Delete abilitato

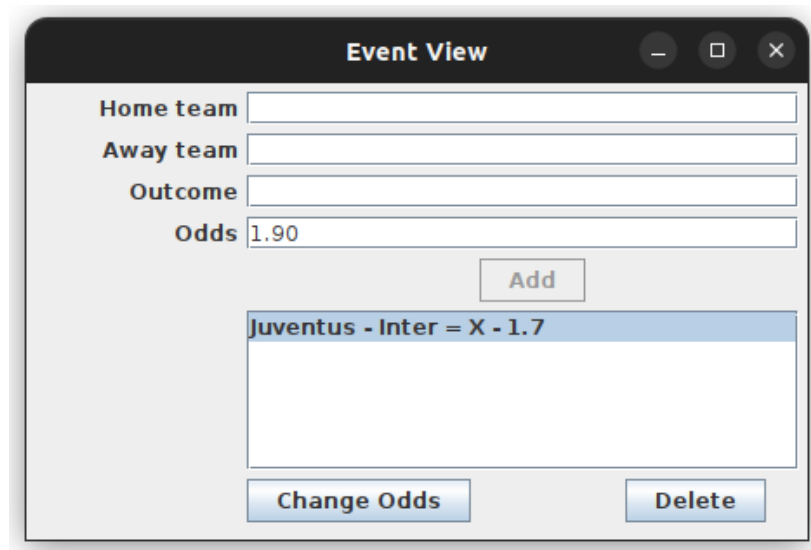


Figura 4: Change Odds abilitato

3.1 PROBLEMI RISCONTRATI

Le principali difficoltà riscontrate riguardano il vano tentativo di utilizzare la libreria *Testcontainers* anche negli *e2e* test. Come spiegato nel libro di testo, configurare tale libreria quando il codice dei test è in un file separato rispetto a quello che esegue i test, è piuttosto complesso. Non essendo riuscito a trovare una soluzione al problema, per eseguire gli *e2e* test è necessario prima avviare un container con *MongoDB* (**in Eclipse va fatto manualmente**¹, in Maven viene fatto automaticamente grazie all'utilizzo del plugin per Docker).

¹ Eseguire il comando: `docker run -p 27017:27017 -rm mongo:4.4.3`

ISTRUZIONI D'USO

Per la build completa del progetto e l'esecuzione di tutti i test, posizionarsi nella directory in cui si trova il file *pom.xml* e digitare da terminale il seguente comando.

```
mvn verify -Pjacoco,pit
```

Specificando i profili *jacoco* e *pit*, verranno anche eseguiti il **code coverage** e il **mutation testing** (modalità *stronger*) sulle sole classi di interesse (escludendo dunque il modello ed il main).