DRAFT

Alex Mazzotti Martino Narducci Davide Braccini

June 10, 2023

Contents

1 Analisi

1.1 Requisiti

Il software proposto consiste nella realizzazione di un videogioco, di nome “DRAFT: Draft Relentlessly A Football Team”, che tratta la simulazione di un torneo di calcio, dove l'utente deve creare la propria squadra, attraverso un draft, con la quale partecipare al torneo.

Requisiti funzionali:

- Possibilità per l'utente di selezionare con quale stagione calcistica desidera giocare.
- Creazione della squadra attraverso un "draft". Il draft consiste, inizialmente nello scegliere un modulo, dopodiché per ogni posizione del modulo si dovrà scegliere un calciatore
- Simulazione della partita con la squadra dell'utente. Durante la simulazione, l'utente deve avere la possibilità di effettuare sostituzioni nella propria squadra

Requisiti non funzionali:

- Grafica intuitiva e user-friendly
- Portabilità su Windows, Mac OS e UNIX
- Possibilità per l'utente di personalizzare la propria esperienza scegliendo un nome per la propria squadra e selezionando un logo tra quelli proposti
- Visualizzare una cronologia delle squadre precedentemente create
- Possibilità di simulare i risultati delle partite con le squadre rimanenti nel torneo, anche se la squadra dell'utente dovesse risultare sconfitta

1.2 Analisi e modello del dominio

Il gioco dovrà offrire all'utente la possibilità di creare una squadra, scegliendo in primis il modulo di gioco tra quelli possibili, dopodiché per ogni posizione dovrà scegliere un giocatore tra quelli proposti casualmente dal sistema. Dovrà essere possibile simulare le partite che dovranno sempre ritornare un vincitore (non ammesso il pareggio). Durante il corso della partita l'utente dovrà avere la possibilità di effettuare sostituzioni di giocatori, che determineranno il variare dell'andamento della partita. (Figura 1.1)

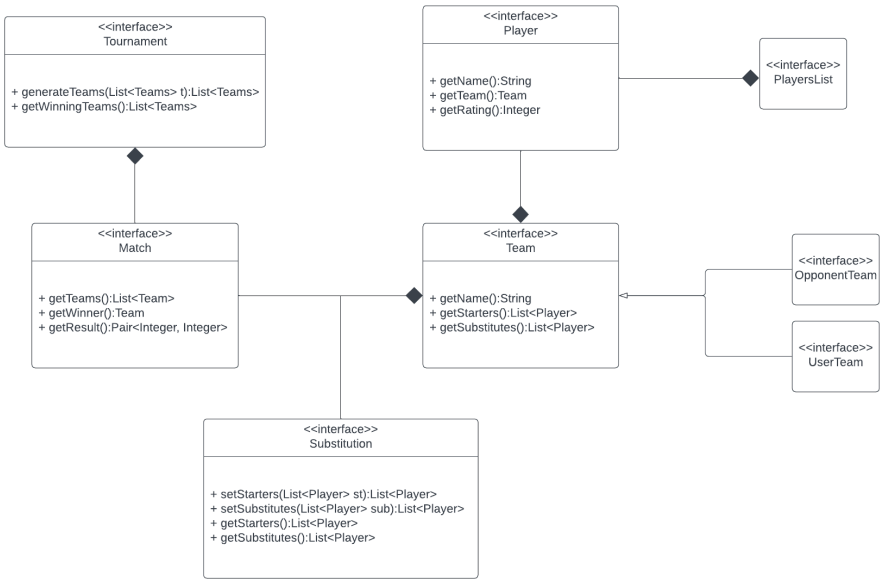
<<interface>>
Tournament
+ generateTeams(List<Teams> t):List<Teams>
+ getWinningTeams():List<Teams>
<<interface>>
Player
+ getName():String
+ getTeam():Team
+ getRating():Integer
<<interface>>
PlayersList
<<interface>>
Match
+ getTeams():List<Team>
+ getWinner():Team
+ getResult():Pair<Integer, Integer>
<<interface>>
Team
+ getName():String
+ getStarters():List<Player>
+ getSubstitutes():List<Player>
<<interface>>
OpponentTeam
<<interface>>
UserTeam
<<interface>>
Substitution
+ setStarters(List<Player> st):List<Player>
+ setSubstitutes(List<Player> sub):List<Player>
+ getStarters():List<Player>
+ getSubstitutes():List<Player>

Figura 1.1

2 Design

2.1 Architettura

L'architettura del sistema segue il pattern MVC (Model-View-Controller) con una variante significativa nell'utilizzo delle View e dei Controller, ovvero sono presenti più View e più Controller mantenendo unico il Model. Più nello specifico:

- Il Model rappresenta la parte del sistema responsabile della gestione dei dati. Comprende tutte le classi e le strutture che mantengono e gestiscono i dati e le operazioni su di essi. La caratteristica distintiva di questa implementazione è che il Model è condiviso tra le diverse View e i Controller. In altre parole, tutte le parti dell'applicazione utilizzano lo stesso Model per accedere ai dati e interagire con la logica di business.

- La View è la parte dell'applicazione che si occupa dell'interfaccia utente. Tuttavia, a differenza dell'approccio tradizionale con una singola View, in questo caso sono presenti più View indipendenti. Ogni View corrisponde a una schermata specifica dell'applicazione e contiene gli elementi dell'interfaccia utente necessari per visualizzare i dati e interagire con l'utente. Ogni View ha il corrispondente Controller associato.

- Infine, il Controller collega il Model e la View. Nell'implementazione attuale, sono presenti più Controller, ciascuno associato a una specifica View. Ogni Controller gestisce le interazioni tra la sua View corrispondente e il Model condiviso. Quando si verificano eventi generati dalla View (per esempio come un clic su un pulsante), il Controller li cattura e li elabora, interagendo con il Model per recuperare e/o modificare i dati necessari. Successivamente, il Controller aggiorna la sua View e restituisce a egli quello che ha richiesto.

Questa modifica all'architettura offre diversi vantaggi. Innanzitutto, consente una migliore separazione delle responsabilità tra le diverse View, rendendo più chiara la gestione delle interfacce utente specifiche. Inoltre, mantenendo un Model condiviso, si garantisce la coerenza dei dati e della logica in tutto il sistema. Infine, l'utilizzo di Controller specifici per ogni View contribuisce a mantenere il codice più modulare e facilmente mantenibile.
Uno svantaggio di utilizzare questa strategia, è una maggiore complessità generale del sistema. Per esempio una cattiva programmazione delle iterazioni tra le diverse View + Controller può portare a complicazioni ed eventuali errori. Nel nostro caso visualizziamo una singola view alla volta, ma più in generale, se ci dovrebbe essere un uso contemporaneo (parallelo) delle View potrebbe portare a problemi di inconsistenza sui dati quindi a problemi di concorrenza.

In sintesi, l'architettura del sistema segue il pattern MVC, ma con la differenza che sono presenti più View e più Controller che condividono lo stesso Model (Figura 2.1). Questa modifica consente una maggiore modularità e separazione, mantenendo però la coerenza dei dati.

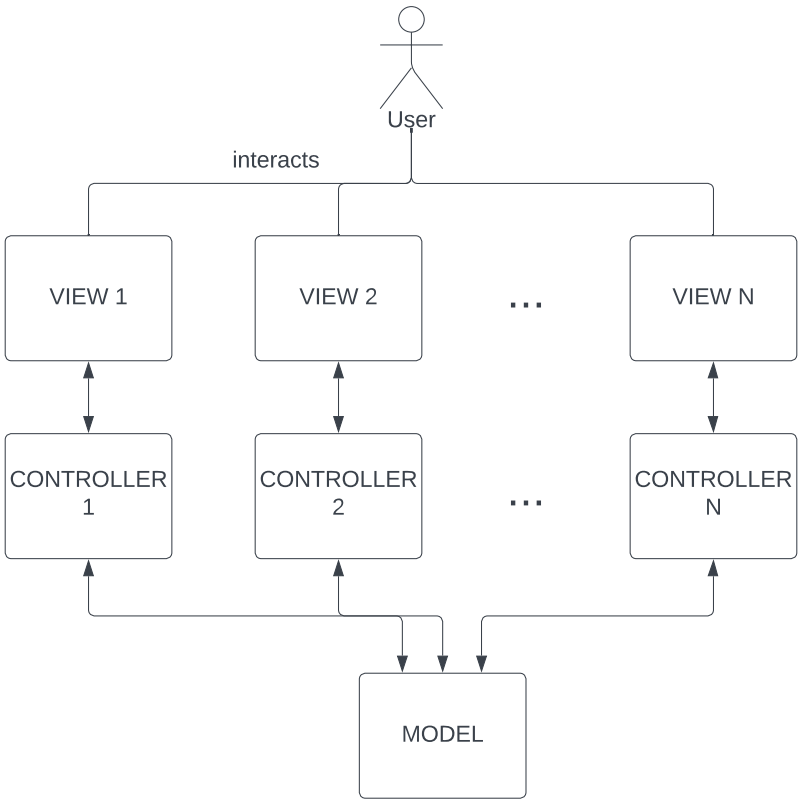

Figura 2.1

Una nota importante riguardo all'implementazione dell'architettura è che le diverse visualizzazioni hanno una sequenza temporale. In altre parole, ci sono dei pulsanti che consentono di navigare tra le diverse interfacce. È di fondamentale importanza garantire una corretta programmazione per coordinare e gestire lo scambio delle diverse interfacce grafiche.
Le iterazioni tra le diverse View possoono essere visualizzate nel seguenze diagramma (Figura 2.2):

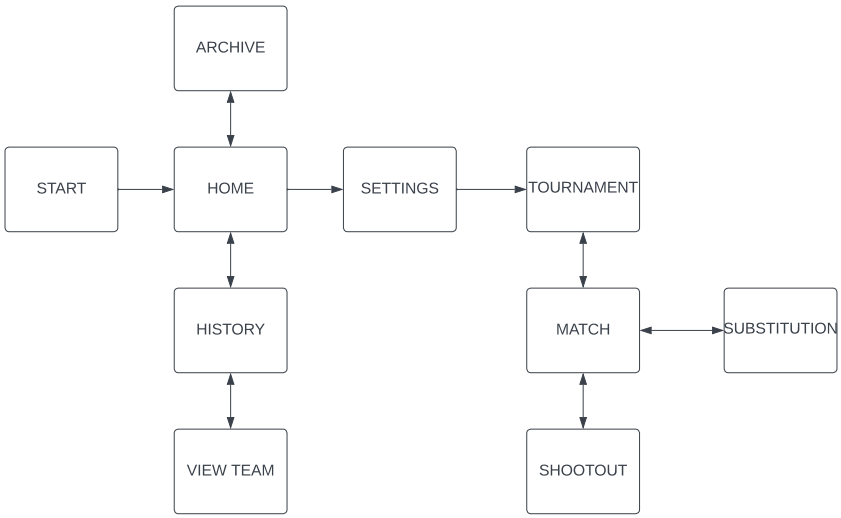

Figura 2.2

2.2 Design dettagliato

Alex Mazzotti
Il mio compito principale era analizzare un database specifico di calciatori online, estrarre i dati da esso e fornire ai miei colleghi una lista di calciatori valida per il loro compito.
Dopo una ricerca e uno studio dei possibili database disponibili sul web, ho scelto il database di www.kickest.it . Ho optato per questa scelta perché è l'unico che fornisce in dettaglio tutte le statistiche e le informazioni sui calciatori della prima divisione italiana, e viene aggiornato regolarmente, includendo anche dati delle stagioni meno recenti.

- Scraping dei dati

  L'estrazione automatizzata di dati da siti web è chiamata "web scraping". Dopo aver analizzato le fonti online e le diverse librerie disponibili in Java, ho individuato due possibili tecniche:

  - La prima utilizza la libreria Jsoup, che consente di accedere e manipolare il codice HTML di un determinato URL.
  - La seconda tecnica utilizza un web driver, in cui non si analizza direttamente l'HTML, ma ci consente di simulare la navigazione di un sito web, quindi di ottenere le informazioni da esso.

Il metodo 1 è più veloce ed efficace in termini di web scraping, ma ha una limitazione quando i siti web caricano i loro dati tramite script, come ad esempio script JavaScript. Infatti, Jsoup non attende il caricamento degli script e prende solo il codice HTML.
Nel caso del dominio che abbiamo preso in considerazione, le tabelle vengono caricate tramite script, quindi la scelta obbligata è utilizzare un web driver.
Un'altra caratteristica da gestire è il fatto che nella tabella caricata non è presente la totalità dei calciatori, ma sarà necessario scorrere la tabella cliccando nell'apposito button.
In breve, il funzionamento dello scraping con web driver nella nostra applicazione funziona cosi:

- Apertura della pagina web
- Selezionare la stagione da scaricare
- Attendere il caricamento della tabella
- Lettura della 1° tabella
- Click del button per passare alla successiva pagina della tabella
- Lettura 2° tabella
- Click del button per passare alla successiva pagina della tabella
- Cosi fino all'ultima tabella...
- Chiusura della pagina web

Questo procedimento viene eseguito in background, ovvero l'utente non vedrà pagine aprirsi e caricare le tabelle, ma vedrà semplicemente se lo scraping sarà eseguito correttamente o no.
Nella seguente figura (Figura 2.3) è mostrato il diagramma dell'architettura dello scarping dei dati

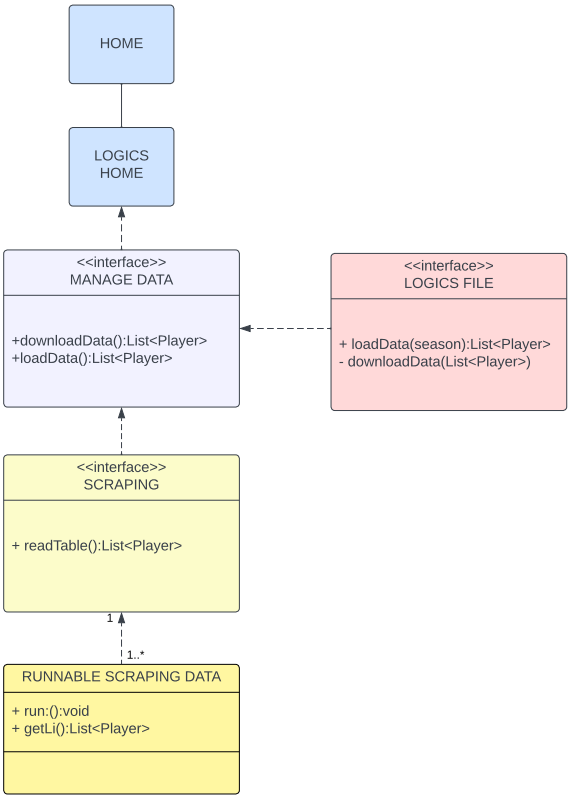

Figura 2.3

- Thread
  Un problema che ho riscontrato al termine della realizzazione dello scraping, è stata la lentezza nell'estrazione dei dati, quindi una efficienza molto bassa dell'algoritmo.
  Per migliorare l'efficienza quindi ho proseguito implementando l'algoritmo usando tecniche di programmazione parallela. Quindi ho suddiviso la quantità di lavoro tra i diversi thread. Il risultato è stata una diminuzione nel tempo totale nello scraping circa del 66%.
  Sulla macchina personale, l'esecuzione in seriale è di 46 secondi, mentre l'esecuzione in parallelo con 7 thread è di circa 16 secondi. Ovviamente i dati cambiano in base alla potenza di calcolo dell'elaboratore.
  Il concetto è reso più intuibile con l'immagine seguente (Figura 2.4)

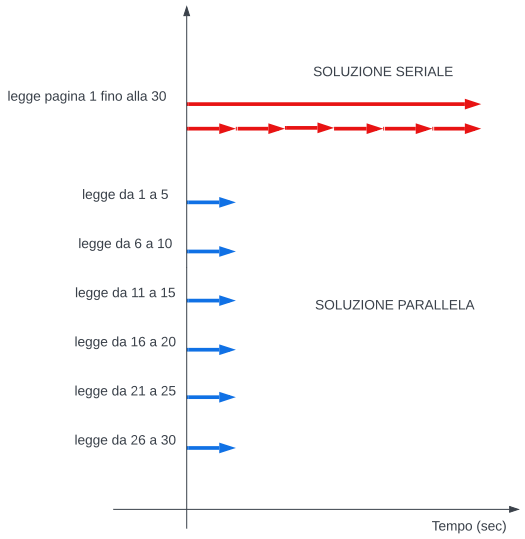

Figura 2.4

- Salvataggio dati in file
  Una volta eseguita l'estrazione dei dati, questi saranno salvati in locale in un file corrispondente alla stagione scaricata. Questo a scopo di non dover scaricare tutte le volte dal sito web (operazione onerosa), e poter utilizzare l'applicazione con un uso offline. (Figura 2.5)

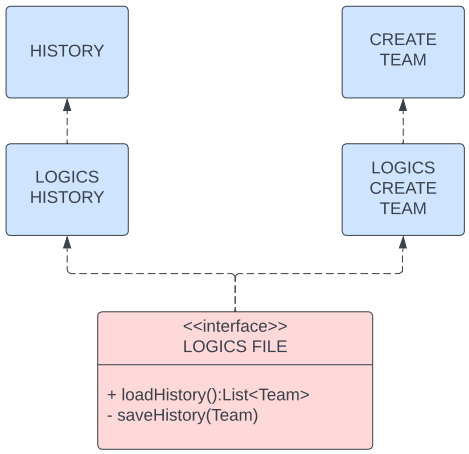

Figura 2.5

Dalla figura si noti come ManageData aggiunga un livello di astrazione: se vogliamo fare scraping nella classe ManageData è presente il metodo downloadData, mentre se vogliamo caricare la tabella salvata in locale usiamo il metodo loadData.
DownloadData utilizza le interfacce Scraping per estrarre e LogicsFile per salvare i file in locale.

- Scraping stagione
  Oltre a estrarre le liste dei calciatori, ogni volta che si esegue l'applicazione e si è online, l'applicazione è in grado di aggiornare in modo automatico la lista delle stagioni che si possono estrarre. Questo a scopo di non avere problemi di consistenza dei dati e sincronizzazione.
  Per esempio se viene aggiunta nel database online una nuova stagione "2023-2024", il sistema sarà in grado di rilevare la nuova tabella e la aggiungerà come opzione tra le stagioni selezionabili dall'utente.
  La particolarità di questa funzionalità è che viene eseguita come thread indipendente (quindi non è bloccante) all'avvio dell'applicazione. L'architettura delle classi è simile a quella dello scraping delle tabelle. (Figura 2.6)

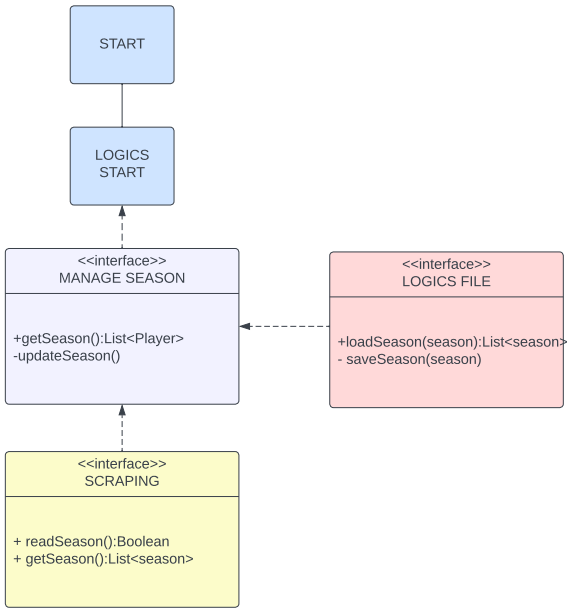

Figura 2.6

- Complicanze e limitazione dell'uso di Web Driver
  Per utilizzare un WebDriver, è fondamentale avere installato sul computer

che esegue l'applicazione il browser compatibile con il WebDriver scelto. Inoltre è necessario aver aggiornati sia le libreire in java, che i driver specifici per i browser (non utilizzare quindi versioni meno recenti dei browser). Per la nostra applicazione quindi ho optato per di fare scraping attraverso o Google-Chrome o Firefox. Ho fatto questa scelta poiché sono i due più diffusi browser, inoltre Firefox è il browser di default in quasi tutti i sistemi Unix-like.
La limitazione prinicipale di questo sistema è quindi che se non sono installati i browser citati non sarà possibile effettuare lo scaricamento online dei calciatori (quindi un aggiormento dei calciatori), sarà comunque possibile utilizzare l'applicazione per fare il draft e simulare il torneo.

- Calcolo rating calciatori
  Un altro compito che mi è stato assegnato consisteva nel calcolare i rating dei calciatori in base alle loro statistiche. Per fare ciò, ho analizzato le informazioni relative ai calciatori, le ho studiate e ho restituito come risultato un rating compreso tra 0 e 100. Ho calcolato quattro tipi di rating per ogni calciatore: il rating generale, il rating per l'attacco, il rating per il centrocampo e il rating per la difesa. Per ottenere una soluzione valida, ho utilizzato concetti di algebra lineare per calcolare i rating. Inizialmente, ho individuato quale calciatore aveva il valore più alto per una particolare caratteristica. Successivamente, ho creato delle funzioni lineari, logaritmiche ed esponenziali, che, dati un valore e il valore massimo tra tutti i calciatori, restituiscono il rating corrispondente compreso tra un valore minimo e un valore massimo.

  Per esempio, se un calciatore ha segnato 21 gol, che è il massimo tra tutti i giocatori, otterrà il rating massimo, che in questo caso è 100. Al contrario, se un calciatore non ha segnato alcun gol, otterrà il rating minimo, che in questo caso è 60.
  Un esempio intermedio: se un calciatore ha segnato 7 gol e il massimo tra tutti i giocatori è 21, la funzione restituirà un rating di 90 per i gol. Considerando i dati: numero massimo di gol = 21 e intervallo del rating [60, 100], possiamo avere i seguenti esempi:
  1) $0/21 = 0\% \rightarrow f(0) = 0$
  2) $7/21 = 33\% \rightarrow f(33) = 90$
  3) $21/21 = 100\% \rightarrow f(100) = 100$
  Gestione della storia delle squadre Funzionalità opzionale era il salvataggio delle squadre create in un file.
  Al termine della creazione di ciascuna squadra questa è aggiunta al file storico.txt. Sarà possibile visualizzare la lista delle squadre nella View History.

Davide Braccini

- Gestione del torneo

  Problema: Immagazzinare e mostrare il risultato di ogni singolo match disputato per ciascun turno del torneo (ottavi di finale, quarti di finale, semifinali e finale), sia per i match che riguardano la squadra utente nel caso non sia stata eliminata, sia per i match in cui si affrontano due squadre avversarie.

  Soluzione: ho creato un'interfaccia LogicsTournament che ha un metodo simulaMatch, il quale si occupa di chiamare i metodi necessari a simulare le partite e ottenere i risultati di ogni singolo match. Dopodiché dalla classe Tournament (cioè la view) ottengo i dati che mi servono grazie ai vari getter presenti nell'interfaccia e aggiorno le label che si vedono nel tabellone.
  Ho fatto in modo che tutti i meccanismi descritti sopra si attivino al premere del bottone "simula" grazie ad un ActionListener, così da permettere all'utente di interagire con la GUI e visualizzare l'andamento del Torneo turno dopo turno. (Figura 2.7)

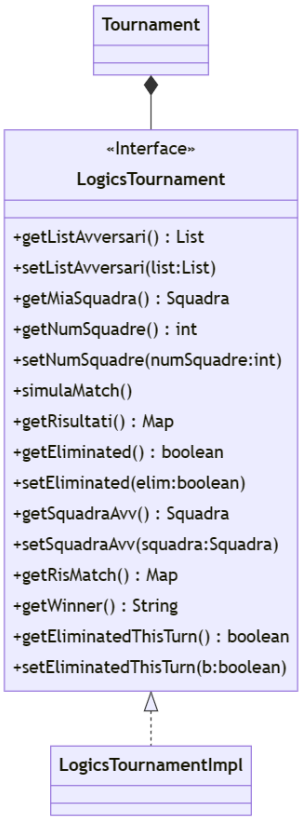

Figura 2.7

- Gestione della creazione squadra

  Problema: Fare in modo che l'utente possa interagire con la GUI scegliendo i giocatori che desidera e utilizzare i dati immessi dall'utente per creare la squadra.

  Soluzione: Per la view ho utilizzato la classe CreateTeam (la quale estende Base, che a sua volta estende JFrame) che si occupa di creare i componenti grafici necessari all'utente per scegliere gli 11 calciatori della sua squadra, cioè il JPanel del frame principale con la formazione, più 2 frame aggiuntivi che servono a scegliere il modulo che si desidera e i calciatori da fare entrare nella formazione.
  Invece per quanto riguarda la logica sottostante ho utilizzato la composizione, tenendo un campo LogicsCreateTeam dentro la classe CreateTeam.
  Questo mi ha permesso di implementare diverse funzionalità come per esempio la selezione dei calciatori da mettere a disposizione dell'utente, che dovevano essere casuali e non ripetersi, e l'aggiunta di quest'ultimi alla squadra titolare con il metodo addPlayerInTeam. Le riserve invece sono 7, vengono aggiunte automaticamente dall'applicazione e non possono essere uguali a nessuno degli 11 iniziali. (Figura 2.8)

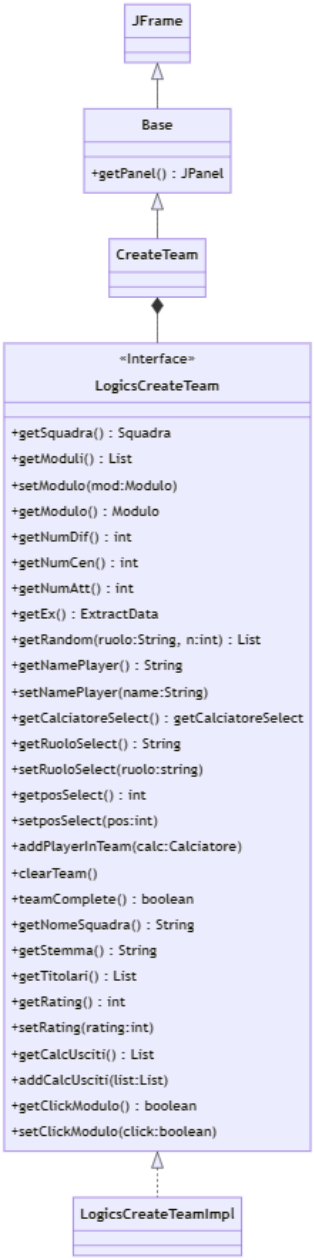

Figura 2.8

Martino Narducci
Il mio lavoro si è concentrato sulla creazione delle squadre avversarie e soprattutto sulla simulazione delle partite.

- Simulazione partita
  L'obiettivo principale della simulazione di una partita era sviluppare un algoritmo che utilizzasse non solo il rating dei calciatori, ma anche le loro statistiche personali. Per creare questo algoritmo, mi sono ispirato alle istruzioni di calcolo del risultato di un gioco chiamato Calcio virtuale
  Il calcolo del risultato finale viene ottenuto mediante l'implementazione di metodi presenti in due diverse interfacce. L'interfaccia SimulatingFunction (utilizzando anche un metodo dell'interfaccia ExtractData) definisce le funzioni utilizzate dall'interfaccia SimulatingMatch, che a sua volta restituisce il risultato finale attraverso un metodo Questo metodo è stato reso riutilizzabile aggiungendo un parametro che specifica il minuto di gioco, in modo da calcolare il risultato sia all'inizio della partita, che ad ogni sostituzione effettuata e in caso di tempi supplementari.
  Successivamente, basandosi sul risultato ottenuto, vengono generati il minutaggio e il marcatore per ogni gol attraverso gli algoritmi presenti nei metodi dell'interfaccia LogicsMatch. Una volta ottenuti il minutaggio e il marcatore, vengono passati alla parte di visualizzazione (View) rappresentata dalla classe Match. (Figura 2.9)

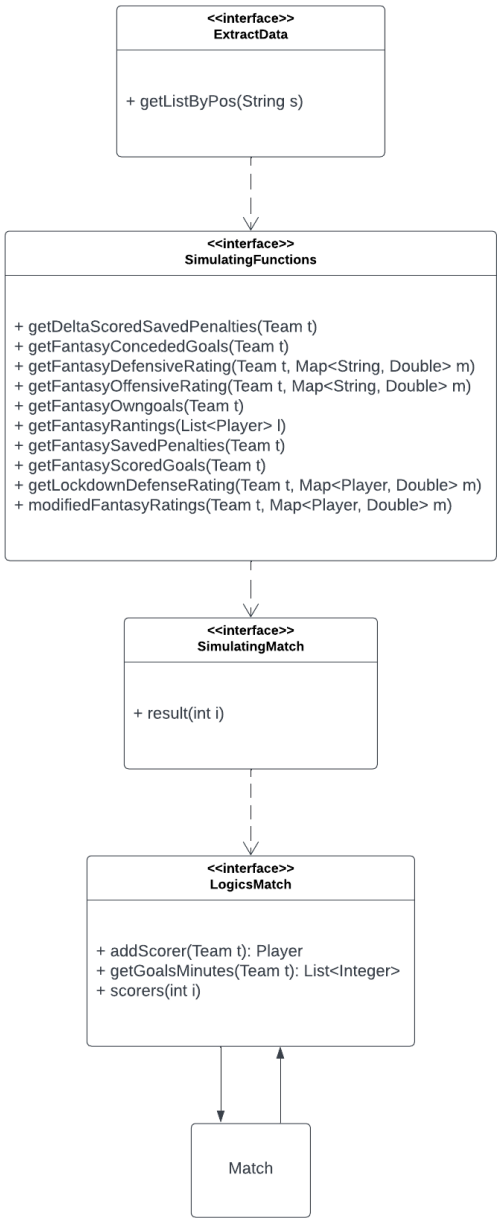

Figura 2.9

- Creazione squadre avversarie
  Mi sono dedicato anche alla creazione delle squadre avversarie da inserire nel tabellone del torneo. Ho optato per la generazione casuale delle squadre utilizzando il metodo fornito dall'interfaccia "CreateOpponent-

Teams", quindi non all'interno di "LogicsTournament". Questo è utile per poter riutilizzare tale funzionalità anche in possibili altri contesti. Alla creazione di un oggetto che implementa interfaccia, deve essere passato il numero di squadre avversarie che si desidera ottenere. (Figura 2.10)

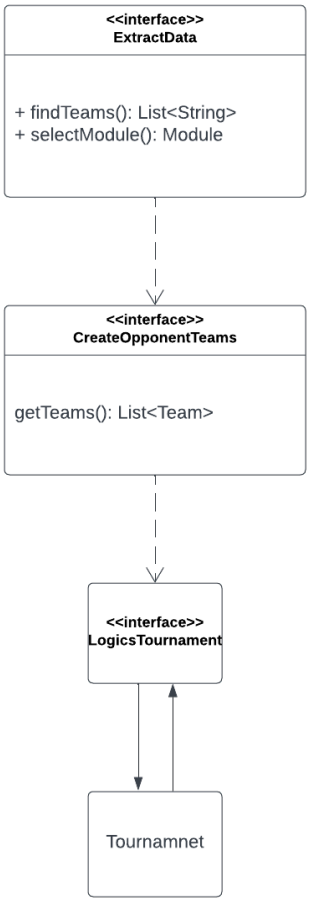

Figura 2.10

- Sostituzione giocatori
  Come funzionalità opzionale, ho implementato la possibilità per l'utente di effettuare sostituzioni all'interno della sua squadra. Per fare ciò, ho creato un'istanza di un oggetto di tipo "Substitution", che è una vista dedicata alla gestione delle sostituzioni. Questo è utile per mantenere una struttura modulare e una separazione delle responsabilità tra questa GUI e quella di "Match". Una volta effettuata la sostituzione desiderata, l'oggetto "Substitution" restituirà la nuova lista aggiornata di calciatori

titolari e riserve. (Figura 2.11)

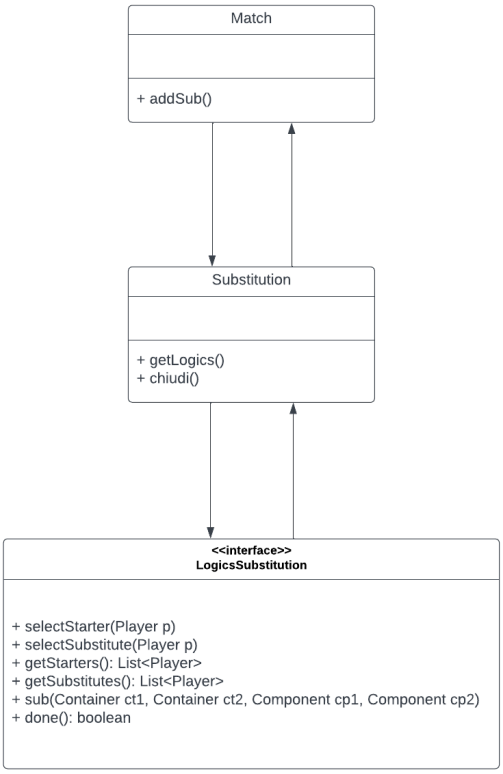

Figura 2.11

- Tiri di rigore
  Siccome la partita deve restituire sempre un vincitore, non può finire in pareggio. Se capita che i tempi regolamentari e quelli supplementari finiscano in parità serve ricorre ai tiri di rigore. Anche in questo caso, in Match, ho creato un'istanza di un oggetto di tipo vista, chiamato "Shootout" che grazie all'interfaccia "LogicsShootout" gestisce i tiri di rigori. La creazione di una nuova GUI è utile per rendere il codice modulare e per permettere la riusabilità. (Figura 2.12)

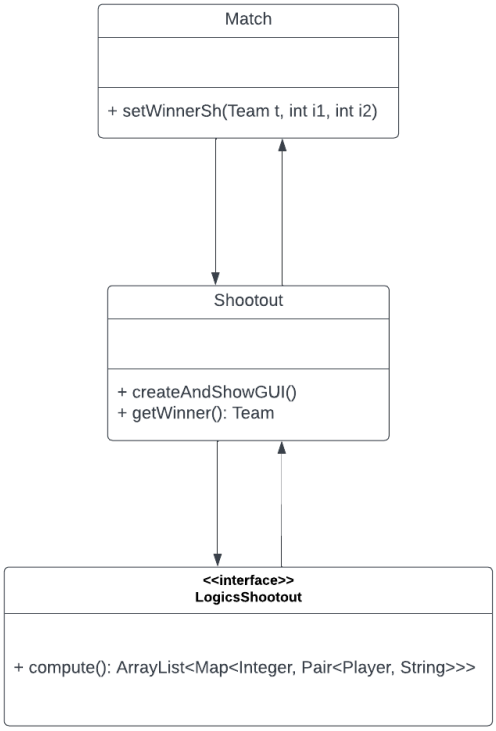
Match
+ setWinnerSh(Team t, int i1, int i2)
Shootout
+ createAndShowGUI()
+ getWinner(): Team
<<interface>>
LogicsShootout
+ compute(): ArrayList<Map<Integer, Pair<Player, String>>>

Figura 2.12

3 Sviluppo

3.1 Testing automatizzato

Per testare la nostra applicazione sono stati effettuati sia test manuali che test automatici, in particolare quelli automatici sono stati eseguiti grazie alla libreria JUnit vista a lezione e hanno riguardato:

- creazione della squadra utente
- creazione delle squadre avversarie
- rating dei calciatori
- estrazione dei dati

3.2 Metodologia di lavoro

Abbiamo suddiviso il carico di lavoro come deciso inizialmente in fase di analisi, per quanto riguarda invece l'integrazione delle parti sviluppate individualmente ci siamo confrontati per trovare punti di incontro e idee che ci aiutassero a collegare le varie parti del gioco.
Riguardo al DVCS abbiamo utilizzato Git, utilizzando un solo branch (master) nel quale apportavamo i vari aggiornamenti giorno dopo giorno.
Di seguito sono indicate le attività svolte singolarmente da ciascun membro del gruppo:

Alex Mazzotti
I compiti da me sviluppati sono stati:

- estrazione delle tabelle online in base alla stagione
- estrazione delle stagioni disponibili
- salvataggio delle tabelle in un file per ciascuna stagione
- salvataggio delle stagioni in un file
- parallelizzazione dell'estrazione delle tabelle (migliorare l'efficienza)
- visualizzazione delle liste dei calciatori in delle tabelle ordinate (archivio)
- personalizzazione dello stemma e scelta nome della squadra
- salvataggio delle squadre create in un file (storico)

Sviluppate le interfacce grafiche e i controller a loro associati (nel nostro sistema hanno il prefisso Logics):

- Start
- Home

- Archive
- History
- ViewTeam
- Settings

inoltre sviluppata una superclasse Base, da cui estendono tutte le altre classi per l'interfaccia, e una classe UtilsGui con metodi statici.

Davide Braccini
Il mio ruolo all'interno del gruppo è stato quello di:

- Occuparmi della parte di progetto che riguarda la creazione della squadra attraverso draft, cioè scelta di un modulo e costruzione del team tramite selezione di un giocatore su cinque disponibili (random) per singolo ruolo, quindi l'implementazione dell'interfaccia LogicsCreateTeam e delle classi LogicsCreateTeamImpl e CreateTeam.

- Occuparmi della parte di progetto che riguarda il torneo, cioè creare gli accoppiamenti tra squadre, gestire i risultati prodotti dalle simulazioni delle partite per decretare la squadra vincitrice e produrre la componente grafica che mostri l'andamento del torneo turno dopo turno; cioè la creazione dell'interfaccia LogicsTournament e delle classi LogicsTournamentImpl e Tournament.
  Qui c'è stato un lavoro di integrazione tra la mia parte e quella implementata invece dai miei colleghi, in particolare nella classe Torneo.java e LogicsTorneoImpl.java dove avevo il compito di svolgere le simulazioni delle partite riguardanti la squadra utente, questo ha postato alla creazione di nuovi metodi da parte dei miei compagni (Match.getWinner, Match.getGolS1, Match.getGolS2...) che ho poi utilizzato per gestire e costruire le varie fasi del torneo.

Martino Narducci
I miei compiti sono stati volti alla simulazione delle partite, e alla creazione squadre avversarie.

- Per quanto riguarda la prima parte: la parte di model è stata sviluppata in SimulatingFunction, SimulatingMatch e in parte in ExtractData, la parte di controller in LogicsMatch, LogicsSubstitution e LogicsShootout, e la parte di view in Match, Substitution e Shootout.

- Per quel che riguarda la seconda parte: la parte di model è stata sviluppata in Module, CreateOpponentTeams, OpponentTeam (dalla quale poi è stata creata UserTeam e l'interfaccia che implementano, Team) e in parte in ExtractData.

3.3 Note di sviluppo

Alex Mazzotti
Dove possibile mi sono impegnato a utilizzare aspetti di linguaggio più avanzati. Dei quali ci sono:

- uso di lamda expression per ridurre e rendere più leggibile il codice
- uso di stream per ridurre e rendere più leggibile il codice, ed eseguire operazioni sulle liste dei calciatori (vedi la classe ExtractImpl)
- uso di thread con runnable per estrarre le tabelle in modo parallelo, e le stagioni in modo non bloccante
- uso di Optional (vedi Extract)
- uso di metodi statici (vedi classe UtilsGui)
- gestione di un web driver per la navigazione automatica di siti web

Davide Braccini
Ho riadattato alle mie esigenze questo pezzo di codice per fare dinamicamente il resize del frame nel momento in cui creo Torneo: ResizeTournament
Source: mbraccini/The-Gravitational-N-Body-Problem

Martino Narducci
Ho usato le lamba e gli stream solo in ExtractData. Per i problemi riscontrati ho cercato su internet, per lo più su i forum specializzati.

4 Commenti finali

4.1 Autovalutazione e lavori futuri

Alex Mazzotti
Sono molto soddisfatto del lavoro portato a termine, sia da me che dei miei colleghi. Essendo uno dei primi progetti in cui si è sviluppato da zero e da una nostra idea lo ho trovato, personalmente, stimolante.
Per quanto riguarda la mia parte mi ritengo molto contento, perché oltre ad aver rispettato i requisiti obbligatori prefissati sono riuscito a portare a termine anche i requiti opzionali.
Inizialmente ci sono state parecchie difficoltà nel capire come lavorare e integrare il contributo di tutti in maniera equa, in parte ci siamo riusciti e in parte no. Per esempio sapendo che si trattava di un progetto lungo io ho iniziato la settimana stessa dalla richiesta dell'elaborato, mentre i miei colleghi hanno iniziato il lavoro qualche settimana dopo.
Pensando a progetti fututi, sicuramente si può integrare il sistema analizzando più dati, e migliorare aspetti come l'interfaccia grafica.
Un' idea potrebbe essere quella di creare un sistema più complesso come un gioco manageriale di calcio, dove puoi allenare e gestire tutti gli aspetti di una squadra calcistica.

Davide Braccini
Sono consapevole che il mio codice potrebbe non essere perfetto e che potrebbe migliorare sotto diversi aspetti, tuttavia ho lavorato per implementare le funzionalità richieste e sono abbastanza soddisfatto di ciò che ho ottenuto.
Inoltre lavorare a questo progetto di gruppo mi ha dato la possibilità di utilizzare in modo consistente il DVCS Git, questo mia ha fatto imparare e/o approfondire le sue svariate funzionalità, già in parte viste durante il corso, e a comprenderne la sua efficacia.
In futuro, durante la scrittura del codice, cercherò di soffermarmi di più per cercare di utilizzare, quando possibile, le funzionalità avanzate del linguaggio e più in generale di suddividere meglio il lavoro che ho da svolgere.

Martino Narducci
Sono abbastanza soddisfatto del mio lavoro, specialmente considerando il mio limitato livello di competenza nella programmazione.
Non ritengo ci siano punti di forza, mentre di sicuro una mia debolezza è il non avere una ampia visione per valutare le scelte da fare; infatti, più volte mi è capitato di dover risolvere bug dovuti alla scarsa qualità del codice che avevo scritto. Mi sarebbe piaciuto dedicare più tempo per rendere il codice migliore e in futuro mi piacerebbe ampliare il gioco con nuove funzionalità, come sfida personale.
All'inizio del progetto, non avevo grandi aspettative sulle mie capacità, ma la pratica fatta per dover sviluppare questo progetto, certamente, mi ha aiutato ad estendere le mie capacità ed ora mi sento appagato dai risultati che ho ottenuto,

anche se modesti.

4.2 Difficoltà incontrate e commenti per docenti

Tutti abbiamo seguito il corso di oop dell'anno precedente (2021-22) e abbiamo fatto il possibile per adeguarci al nuovo regolamento.
Il progetto è stato interamente realizzato in Eclipse.

A Guida utente

All'avvio dell'applicazione si presenta la seguente schermata: (Figura A.1)

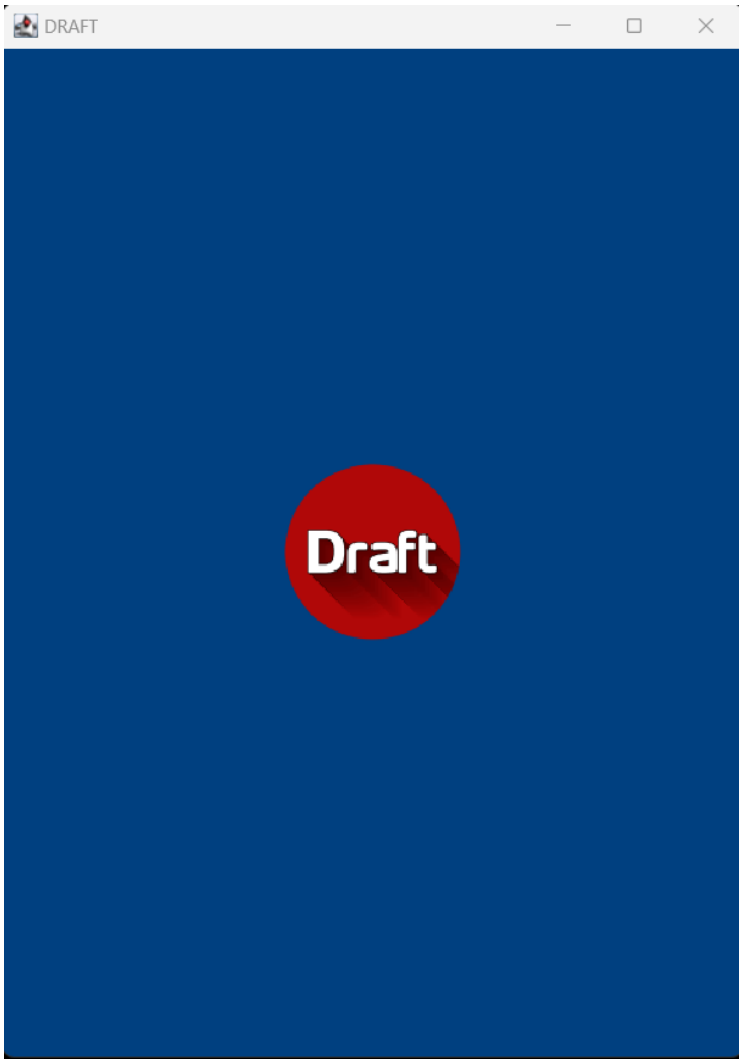

Figura A.1

Premendo sul bollino rosso con su scritto "Draft" si arriverà alla pagina successiva. (Figura A.2)

Figura A.2

Prima di iniziare a giocare possiamo soffermarci su alcuni aspetti:

- Con il bottone "Choose a season" possiamo scegliere la stagione sportiva che preferiamo tra le quattro proposte, i giocatori e le squadre presenti dopo dipenderanno da questa.
- Il bottone "Download season" ci permette di scaricare la stagione selezionata.
- Con il bottone "Archive" possiamo visualizzare l'elenco dei giocatori presenti (variano a seconda della stagione selezionata)
- Con il bottone "History" invece possiamo visualizzare le eventuali squadre precedentemente utilizzate per giocare a FootballDraft, cioè il loro nome,

lo stemma, i giocatori titolari, le riserve e una valutazione.

Dopo aver settato le impostazioni che si desiderano si può avanzare premendo START.
A questo punto inizia la costruzione della squadra con l'inserimento del nome e dello stemma. (Figura A.3)

Figura A.3

Ora vediamo che l'applicazione ha creato 2 frame aggiuntivi, uno per il modulo (Figura A.4) e uno per i calciatori (Figura A.5); per prima cosa viene richiesto di selezionare un modulo, dopodiché passiamo all'inserimento dei calciatori titolari uno ad uno in base alle nostre preferenze (Figura A.6), le riserve verranno aggiunte in automatico dall'applicazione in modo casuale.

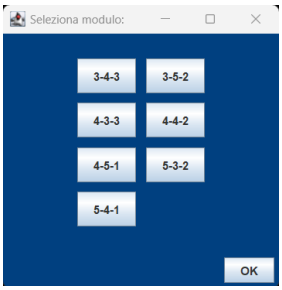

Figura A.4

Figura A.5

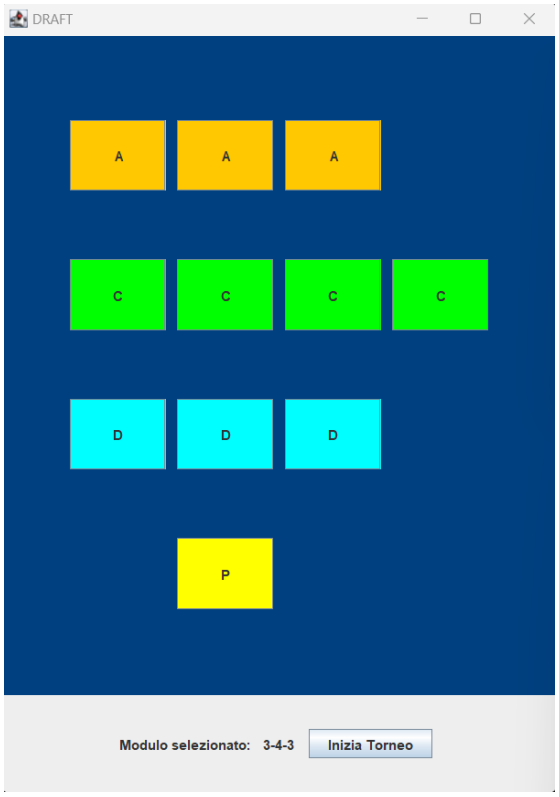

Figura A.6

Una volta completata la formazione si può premere il bottone “Inizia Torneo”

Figura A.7

Qui (Figura A.7) viene visualizzato il tabellone del torneo con le squadre partecipanti a partire dagli ottavi di finale.
Premendo il bottone “Simula” potrebbe succedere:

- Se la squadra utente è ancora in gioco viene visualizzata la simulazione della partita che la riguarda (Figura A.8), vengono simulate le altre partite e al termine di tutto ciò viene creato e visualizzato il turno successivo del Torneo.

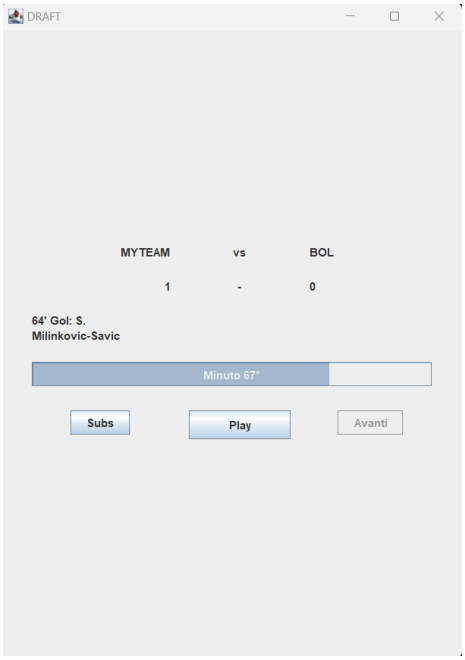

Figura A.8

- Se la squadra utente è stata eliminata vengono simulate le partite che riguardano le altre squadre rimanenti e viene creato e visualizzato il turno successivo del Torneo.

Dopo qualche turno si arriverà alla squadra vincitrice!