



FANTACALCIO

GROUP PROJECT

26/09/2017

DIEGO ERCOLI MAT.090306

LEONARDO MAZZARACCHIO MAT.090210

OBIETTIVI

- Mettere in pratica le conoscenze e le competenze teoriche/pratiche apprese nei corsi di base di informatica (Basi di Dati, Programmazione, Ingegneria del Software, Algoritmi e Strutture Dati ecc...);
- Progettare ed implementare da zero un sito in grado di organizzare ed elaborare in modo efficiente una grande quantità di dati, e fornire interfacce atte a favorire una buona **user-experience**.

COS'È IL FANTACALCIO?



- Ciascun partecipante all'interno di un gruppo, attraverso un'asta, acquista dei calciatori con lo scopo di formare una **squadra virtuale**, chiamata rosa.
- Per ogni giornata del campionato di serieA, ogni partecipante schiera una **formazione virtuale** scegliendo i calciatori dalla propria rosa.
- Al termine di ogni giornata, ad ogni formazione schierata verrà calcolato un punteggio sulla base dei **voti** assegnati ai calciatori scesi in campo.
- Il **punteggio** di ciascuna formazione verrà utilizzato per elaborare la **classifica**.

DESCRIZIONE GENERALE DEL SISTEMA (PROSPETTIVA UTENTE)

L'utente completa con successo la registrazione al sito.

Entra a far parte di un **gruppo** privato, o ne crea uno nuovo invitando membri e scegliendo il tipo di **competizione**: scontri tutti contro tutti o scontri diretti.

L'amministratore del gruppo inserisce i **calciatori** acquistati dai **partecipanti** durante un'asta al di fuori del sito, formando per ciascuno la **rosa**.

Al termine di ogni giornata l'amministratore del sito, caricherà le **prestazioni** dei calciatori di Serie A. In seguito verranno calcolati i **punteggi** di tutte le formazioni schierate, aggiornando classifiche.

Prima dell'inizio di ogni **giornata**, l'utente inserisce una **formazione** scegliendo quali calciatori schierare tra quelli acquistati.

Accedendo alla pagina del gruppo, l'utente controlla la propria rosa, ed ha accesso alle varie informazioni disponibili.

TECNOLOGIE UTILIZZATE

Lato client

HTML5

CSS (Bootstrap)

Javascript

JQuery

JQuery UI

Ajax

Lato server

JSP (JavaServer Pages)

Java Servlet

Hibernate

DBMS

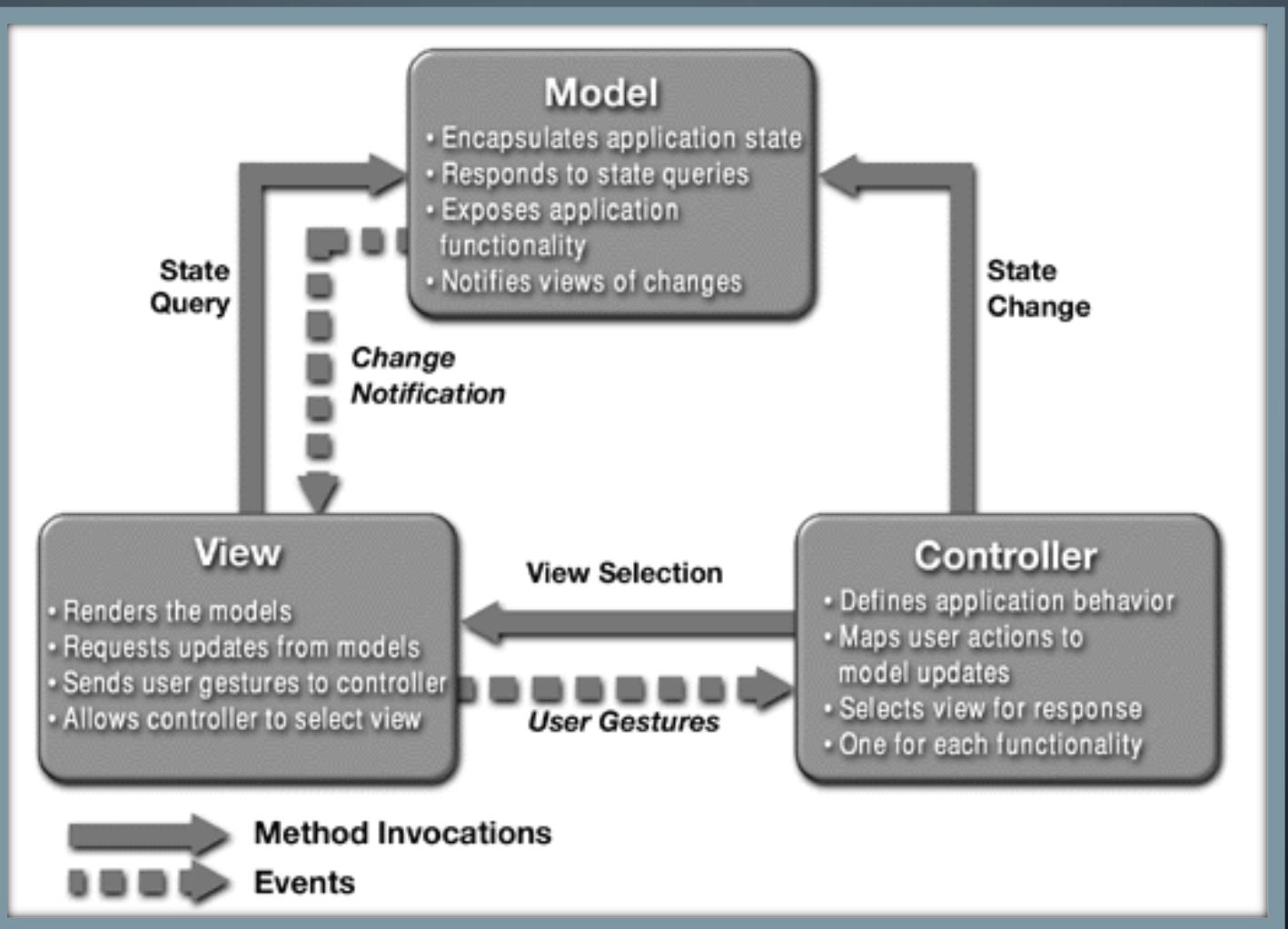
MySQL

MVC

MODEL: incapsula lo stato dell'applicazione, definendo gli oggetti che la compongono e le proprietà di cui quest'ultimi sono costituiti.

VIEW: gestisce la logica di presentazione dei dati. Ciò implica che essa deve gestire la costruzione dell'interfaccia grafica (GUI), attraverso cui gli utenti interagiranno con il sistema.

CONTROLLER: ha la responsabilità di trasformare le richieste che l'utente invia interfacciandosi con la View, in eventuali azioni che possono modificare lo stato del Model.



MVC INTEGRAZIONE DELLE TECNOLOGIE: MODEL

Il model viene definito attraverso le classi java, che rappresentano il **modello di dominio** del sistema.

Caratteristiche classe model:

- i dati sono contenuti negli attributi che devono essere privati e accessibili utilizzando appositi metodi di accesso get/set.
- Per lavorare con le Collections, è necessario redifinire il metodo **equals** (scegliendo uno o più attributi)
- Implementando l'interfaccia **Comparable<E>**, si può definire il concetto di ordinamento.

MVC INTEGRAZIONE DELLE TECNOLOGIE: CONTROLLER

Il ruolo del controller è ricoperto dalla **servlet**, la quale gestisce le richieste inviate dall'utente, in particolare:

- Effettua i dovuti controlli;
- Legge eventuali parametri;
- Esegue la richiesta dell'utente (recupero di dati, aggiornamento, lettura file, ...);
- Riferimenti ai risultati (oggetti del **model**) dell'elaborazione, vengono memorizzati temporaneamente come attributi dell'oggetto request (oggetto HttpServletRequest che rappresenta la richiesta);
- Inoltra la richiesta alla vista.

MVC INTEGRAZIONE DELLE TECNOLOGIE: VIEW

Il ruolo della View viene ricoperto da JSP. La pagina JSP non deve creare oggetti appartenenti a classi Java, o modificare lo stato degli oggetti ricevuti.

Infatti è solamente responsabile della **presentazione dei risultati**.

Con l'utilizzo della tecnologia JSP si cerca di ridurre il più possibile la presenza di <% codice Java %> utilizzando i tag e le librerie che essa fornisce.

RIEPILOGO

Vediamo uno scenario che aiuti a chiarire le interrelazioni dei tre componenti architettonici. Nel seguente caso, un utente vuole recuperare la password associata al suo account, per autenticarsi.

```
<form autocomplete="off" name="form_pass" action="recupero_psw" method="POST">
  <div class="container-fluid">
    <div class="row">
      <div class="panel col-md-8 col-md-offset-2 col-sm-8 col-sm-offset-2 border">

        <h1>
          <span class="cursive">Recupero password</span>
        </h1>
        <br>
        <p class="imp"> Invieremo le credenziali all'indirizzo email specificato. </p>

        <div class="row">
          <div class="form-group col-md-8 col-md-offset-2 col-sm-8 col-sm-offset-2">
            <label for="email">Indirizzo e-mail</label>
            <div class="input-group">
              <span class="input-group-addon input-group-blue"><i class="glyphicon glyphicon-envelope"></i></span>
              <input type="email" name="email" id="email" class="input_group input_blue form-control text-center" size="30" autofocus required />
            </div>
          </div>
        </div>

        <div class="form-group">
          <button type="submit" name="submit" class="btn btn-md">INVIA</button>
        </div>
      </div>
    </div>
  </form>
```

All'interno delle servlet le richieste HTTP POST vengono gestite tramite il metodo
doPost(HttpServletRequest request, HttpServletResponse response)

```
public void doPost(HttpServletRequest request,HttpServletResponse response) {  
    //prendo l'email inviata come parametro  
    String email = request.getParameter("email");  
    //recupero, se esistono, user e password associati all'email  
    Utente utente = gU.recuperoCredenziali(email);  
    //contiene la pagina a cui fare il forward  
    String indirizzo=null;  
    //se esistono user e password  
    boolean invio = false;  
    if(utente!=null)  
    {  
        //imposto un riferimento all'oggetto utente  
        request.setAttribute("utente", utente);  
        //creo il messaggio dell'email  
        String messaggio = "<p> Le tue credenziali di accesso sono:<br><br> User: " + utente.getUser() + "<br><br> Password: " + utente.getPassword() + "<br> </p>";  
        //invio l'email  
        invio = gMail.send(email,messaggio);  
    }  
    request.setAttribute("invio", invio);  
    indirizzo = "/WEB-INF/recupero_psw.jsp";  
  
    RequestDispatcher dispatcher = request.getRequestDispatcher(indirizzo);  
    try  
    {  
        dispatcher.forward(request, response);  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Nella View verso cui è stato effettuato il forward, la pagina JSP accede ai dati ricevuti dal Controller e genera il documento html.

```
<c:if test="${not empty invio}">
    <div class="panel col-md-6 col-md-offset-3 col-sm-6 col-sm-offset-3 border first_riq riquadro_g">
        <c:choose>
            <c:when test="${empty utente}">
                <p class="p-md grey"> Email non riconosciuta! </p>
            </c:when>
            <c:otherwise>
                <c:choose>
                    <c:when test="${not invio}">
                        <p class="p-md grey"> Si è verificato un errore durante l'invio dell'email! </p>
                    </c:when>
                    <c:otherwise>
                        <p class="p-md grey">
                            Utente <strong>${utente.user}</strong>, <br>
                            al tuo indirizzo ${utente.email}
                            è stato invitato un messaggio con le tue credenziali.
                        </p>
                    </c:otherwise>
                </c:choose>
            </c:otherwise>
        </c:choose>
    </div>
</c:if>
```

RACCOLTA DEI REQUISITI

Si vuole realizzare una piattaforma web in grado di gestire tutte le fasi del gioco fantacalcio, incentrato sul campionato di calcio "Serie A". Alla base del campionato ci sono i **calciatori**, identificati univocamente da un id, ciascuno dei quali è caratterizzato da un nome, cognome ed un **ruolo** che gli viene assegnato. Il ruolo può essere: portiere, difensore, centrocampista o attaccante. Lo stesso ruolo può ovviamente essere assegnato a più calciatori. In un dato momento, un **calciatore attivo** è un calciatore che ha sottoscritto un contratto con un club della Serie A, il quale ha un nome che lo identifica.

Un campionato di Serie A, si suddivide in 38 **giornate**, in genere della durata di un weekend. Per ciascuna di esse interessa sapere l'anticipo, ovvero la data e ora in cui la giornata inizia, se la giornata è terminata o è aperta, e l'id univoco associato alla giornata. All'interno di una giornata si svolgono massimo 10 **partite** di campionato, ognuna delle quali si identifica dall'id della giornata in cui si svolge, e da un attributo che indica il numero della partita all'interno della giornata. Ogni partita è composta da un club che gioca in casa, uno che gioca in trasferta e dal risultato della partita, determinato dai gol fatti da ciascuna squadra. Ciascun club gioca fino a 19 partite in casa e altrettante in trasferta. Ad ogni partita il calciatore che ha giocato ottiene una **prestazione**, a cui viene associato un codice univoco. Una prestazione contiene le seguenti informazioni riguardo al calciatore a cui è associata: gol fatto, gol subito, assist, autogol, ammonizione, espulsione, rigore sbagliato, rigore parato, voto (assegnato dalla Gazzetta dello sport), e il fantavoto calcolato partendo dal voto ottenuto, che viene poi alterato dai bonus/malus registrati (gol, assist, ecc...).

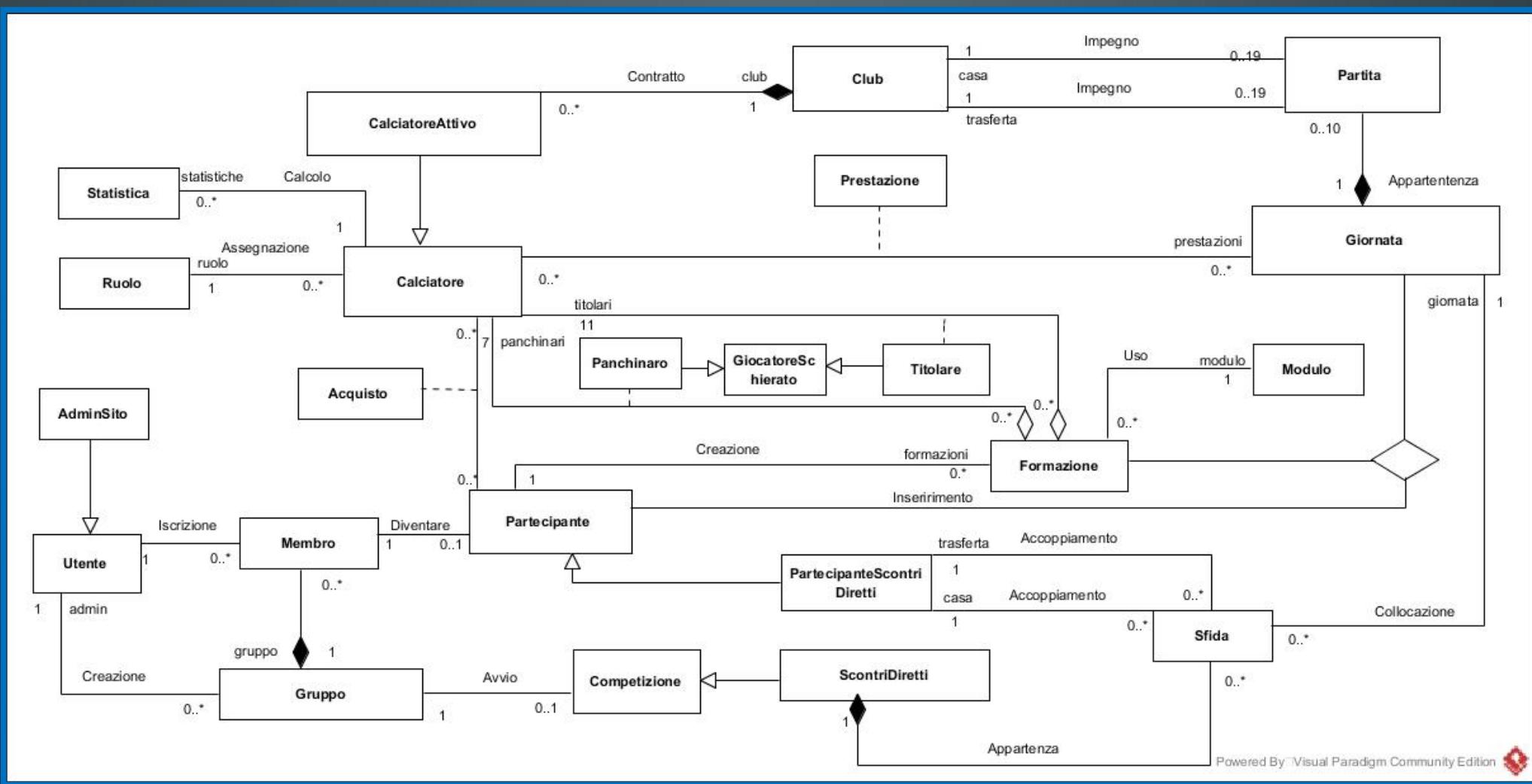
All'interno della piattaforma web un **utente** per prima cosa si registra, inserendo i propri dati anagrafici (nome, cognome, ecc...). Oltre a questi dati inserisce un username, che lo contraddistingue, una password e un indirizzo email in suo possesso, utile per attivare il proprio account. L'**amministratore del sito** è un utente che gestisce tutti i dati appartenenti alla serie A.

Una volta creato con successo un account, l'utente può creare un proprio **gruppo** o unirsi ad uno già esistente tramite un invito. Un gruppo è composto da un nome, che lo rappresenta, un amministratore che corrisponde all'utente che l'ha creato, e da un insieme di uno o più utenti che diventeranno **membri**: una volta registrato il gruppo, amministratore compreso. In seguito alla registrazione, verrà inviata una richiesta di partecipazione ai membri. I membri che fanno parte del gruppo saranno rintracciabili tramite un codice, e di essi sarà utile sapere lo stato della risposta alla richiesta. L'amministratore del gruppo, verificata la risposta dei membri, può scegliere se avviare una **competizione** a scontri tutti contro tutti o

una a scontri diretti. Una volta avviata la competizione, i membri che hanno accettato la richiesta diventeranno **partecipanti**, con un budget a disposizione per acquistare i calciatori, e un punteggio che determinerà la classifica all'interno del gruppo. Ogni competizione sarà quindi associata ad uno ed un solo gruppo, e conterrà informazioni quali: la data di avvio della competizione, il budget che i partecipanti avranno a disposizione, e il numero di calciatori che un partecipante può acquistare per ciascun ruolo definito. Una **competizione a scontri diretti** è una competizione che, oltre alle informazioni di una normale competizione, è caratterizzata anche da una soglia gol, ed un intervallo di punteggio, informazioni utili per calcolare gli esiti degli scontri. All'interno di una competizione di questo tipo, viene generato un calendario con delle giornate che corrispondono alle giornate di campionato. Ad ogni giornata, ciascun partecipante viene associato ad un altro, in una **sifida** rappresentata quindi da un partecipante che gioca in casa, uno che gioca in trasferta, dai gol fatti da ciascun partecipante (determinati usando la soglia gol, e l'intervallo di punteggio), e in codice che lo identifica. Un **partecipante di una competizione a scontri diretti** è un partecipante di cui, oltre al budget disponibile è utile conoscere il numero di gol fatti in sfide in cui giocava in casa, e quelli fatti in sfide in cui giocava in trasferta, nonché il numero di vittorie, sconfitte e pareggi conseguiti, e i punti (ottenuti dalla somma dei punti derivanti dagli esiti delle sfide).

All'interno di una competizione, l'amministratore del gruppo crea per ciascun partecipante una rosa di calciatori, derivante dall'asta svolta tra amici dove ciascuno di essi ha acquistato dei calciatori. L'**acquisto** di un calciatore è individuabile tramite un id d'acquisto, e caratterizzato da un costo che rappresenta quanto il partecipante ha pagato per quel calciatore, il quale verrà sottratto dal budget a sua disposizione. Ad ogni giornata il partecipante inserisce una **formazione** a cui viene associato un **modulo**, contraddistinto dagli altri dal suo nome, che determina i calciatori che possono essere posizionati all'interno della formazione per ciascun ruolo. Nella formazione il partecipante sceglie quindi quali calciatori schierare tra quelli che ha acquistato. I **calciatori schierati** sono rintracciabili attraverso un apposito id, e possono essere **titolari** (fino a 11) o **panchinari** (fino a 7). Al termine di ogni giornata l'amministratore del sito, provvederà a caricare le prestazioni dei calciatori di Serie A attraverso un opportuno file. In seguito all'importazione verranno calcolati i punteggi di tutte le formazioni schierate dai partecipanti, in base alle prestazioni svolte dai calciatori inseriti, aggiornando le classifiche.

PROGETTAZIONE MODEL



PROBLEMA: OBJECT-RELATIONAL IMPEDANCE MISMATCH

- Il nostro sistema aderisce al paradigma della programmazione orientata agli oggetti, in cui i dati sono encapsulati nelle variabili interne di ciascun oggetto.
- Una database relazione rappresenta i dati in formato tabellare (insieme di tuple).
- Disallineamento tra i due paradigmi:
 - -Ereditarietà(intrinsecamente OOP)
 - -Identità (chiave primaria in database)
 - -Associazioni (foreign key in database, riferimento unidirezionale in OOP)

SOLUZIONE: ORM (OBJECT-RELATIONAL MAPPING)

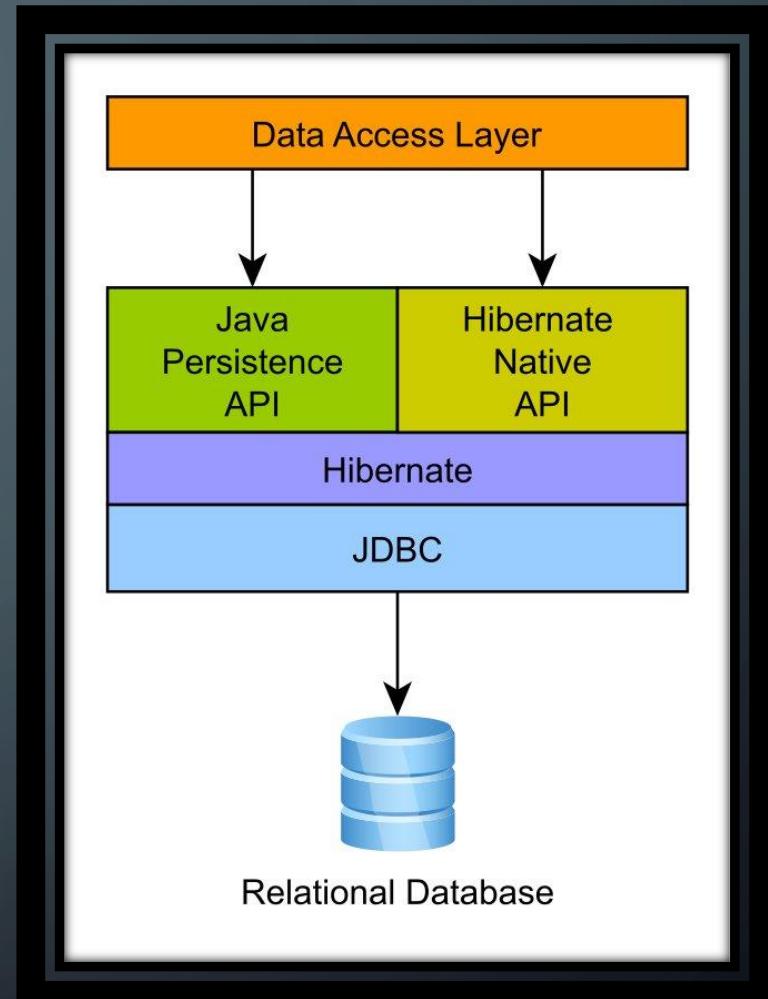
- Superamento (più o meno completo) dell'incompatibilità di fondo tra il progetto orientato agli oggetti ed il modello relazionale sul quale è basata la maggior parte degli attuali RDBMS utilizzati (mapping tramite annotazioni).
- Elevata portabilità rispetto alla tecnologia DBMS utilizzata (cambiando database, occorre riscrivere il file di configurazione)

```
<properties>
    <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
    <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/fantacalcio" />
    <property name="javax.persistence.jdbc.user" value="root" />
    <property name="javax.persistence.jdbc.password" value="" />
    <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
    <property name="hibernate.show_sql" value="true" />
    <!--<property name="hibernate.hbm2ddl.auto" value="update" /> -->
    <property name="hibernate.connection.pool_size" value="300"/>
</properties>
```
- Drastica riduzione della quantità di codice sorgente da scrivere, sfruttando le api del framework.

JPA: SPECIFICHE JAVA HIBERNATE: FRAMEWORK

Una volta configurato in modo appropriato, Hibernate si posiziona tra il Data Access Layer dell'applicazione Java e il database relazionale. L'applicazione Java utilizzerà le API di JPA per operare sui dati di dominio. Il framework Hibernate si occuperà di generare automaticamente le opportune query SQL.

- Creazione di un nuovo oggetto → Query di inserimento.
- Recupero di un grafo di oggetti → Query di selezione
- Aggiornamento stato oggetto → Query di aggiornamento



PROGETTAZIONE CONTROLLER

Un utente richiede l'esecuzione di un particolare task, inviando la richiesta ad una specifica servlet responsabile della sua gestione.

La servlet innanzitutto deve accertarsi che l'utente disponga dei dovuti **requisiti**.

id	definizione requisiti	Nome package	elenco servlet	idea implementativa
a	-l'utente non è stato ancora autenticato	iscrizione	Registrazione, Login, RecuperoPsw,...	Verificare che non sia stata impostata la variabile di sessione 'user'.
b	-l'utente è stato autenticato	home	Logout, ProfiloUtente, CreazioneGruppo,..	Variabile di sessione 'user' impostata
c	-requisiti di b -l'utente è un amministratore	admin	LetturaDatiSerieA, ElaboraGiornata, Reset ...	Ereditare la definizione di c. Dato un username, verificare se l'utente è un amministratore
d	-requisiti di b - l'utente è un membro del gruppo "corrente"	gruppo	ProfiloGruppo, StatoGruppo..	Ereditare la definizione di b. Dato la variabile di sessione 'gruppo', verificare se l'utente è un membro.
e	-requisiti di d -il gruppo "corrente" ha avviato la competizione	gioco	InserimentoFormazione, Classifica, Rose, ...	Ereditare la definizione di d. Dato la variabile di sessione 'gruppo', verificare la presenza di una competizione.
f	-requisiti di e -l'utente è l'amministratore del gruppo.	giocoAdmin	InserimentoRose, ModificaProfilo...	Ereditare la definizione di e. Dato un username ed un nome di gruppo, verificare che l'utente ne è l'admin.

- Le servlet (classi concrete) possono essere raggruppate in insiemi (chiamati **package**), in base alla condivisione dei requisiti d'accesso richiesti;
- In ogni package, vi è una classe astratta che definisce questi requisiti;
- Questa classe astratta ne può estendere un'altra, ereditando una definizione già messa a punto.

In tal caso errà invocata la versione del metodo che è stata riscritta usando la parola riservata **super**.

Requisito b

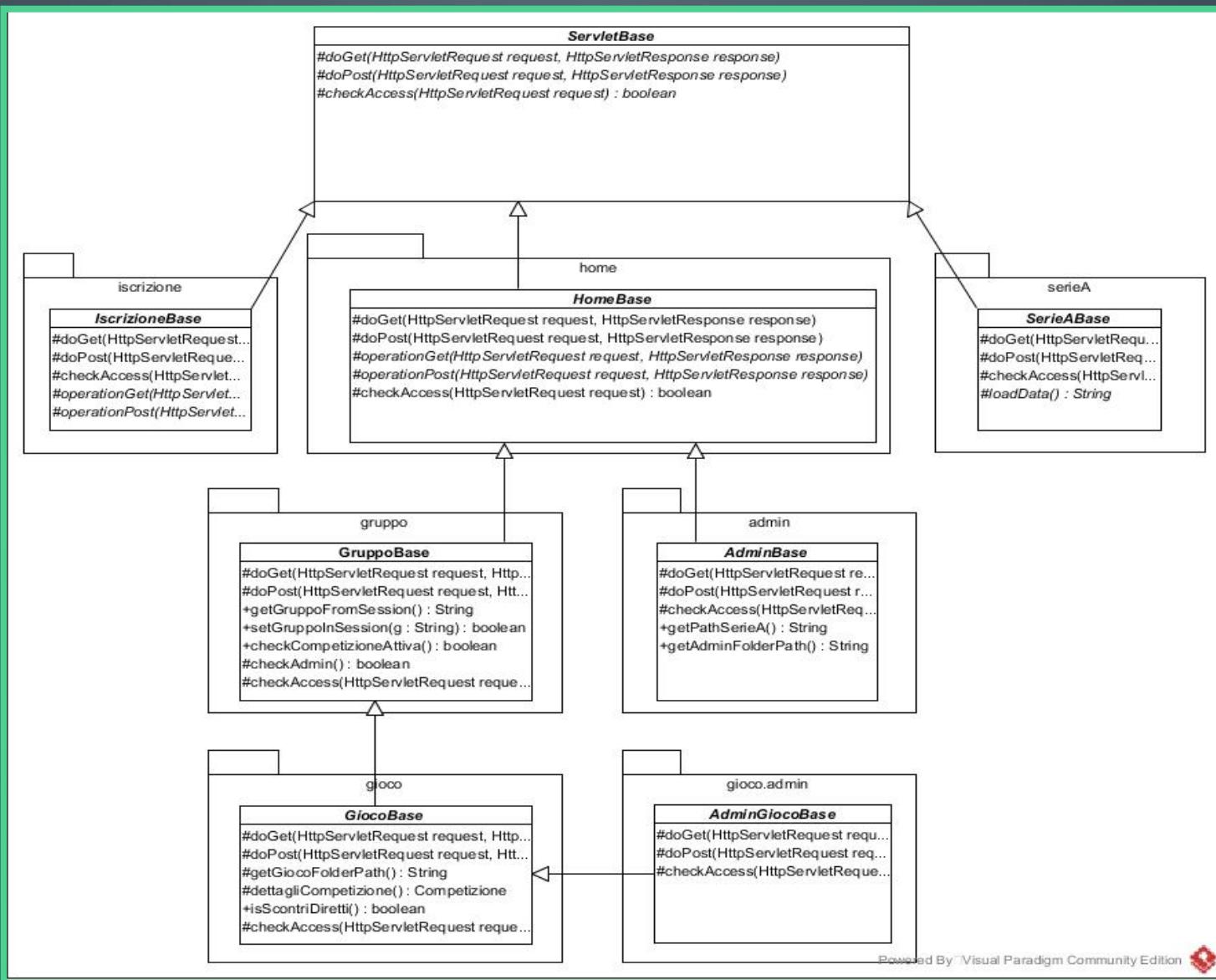
Requisito a

- Requisiti di a

- L'utente è un amministratore del sito
- L'utente deve essere autenticato/loggato

```
/**  
 * L'utente deve essere loggato ed allo stesso tempo  
 * essere un amministratore del sito.  
 */  
@Override  
protected boolean checkAccess(HttpServletRequest request) {  
    GestoreUtente gu = new GestoreUtente();  
    //se l'utente è loggato  
    if(super.checkAccess(request))  
    {  
        String user = this.getUserFromSession(request);  
        //se l'utente è l'admin del sito  
        if(gu.checkAdmin(user))  
            return true;  
    }  
    return false;  
}
```

GENERALIZZAZIONE CLASSI ASTRATTE



PROGETTAZIONE VIEW

Le View sono state progettate con l'intento di favorire la manutenzione e il riutilizzo del codice, implementando parti comuni di codice in View separate che vengono poi incluse in tutte quelle View che necessitano di quella parte.

Es. Menu principale.

SVILUPPO SOFTWARE

Per lo sviluppo del software è stato utilizzato uno stile iterativo, dove si suddivide il progetto in più iterazioni, all'interno del quale viene svolto un ciclo di sviluppo software composto da:

1. Analisi dei requisiti;
2. Progettazione;
3. Implementazione;
4. Testing.

ZONA SERIE A

Zona fondamentale per il funzionamento della piattaforma, poiché contiene quelle funzionalità che permettono all'amministratore del sito di caricare ed aggiornare i dati sui quali la piattaforma si basa.

È possibile scaricare i file in formato "[.excel](#)" tramite delle fonti esterne. Per agevolarne la lettura ciascun file viene convertito nel formato "[.csv](#)", e caricato all'interno del sito. A seconda del contenuto, è stato implementato un apposito lettore (lettore calciatori, lettore voti, lettore calendario).

MODALITA' DI LETTURA

I file sono in un formato in cui ciascun record viene rappresentato da una riga di testo, al cui interno i valori dei vari campi sono stringhe divise dal carattere separatore “;”. Come vengono individuate le informazioni in ciascuna riga?

- Ogni lettore ha una lista di nomi di colonne d'interesse, che si aspetta di trovare nella riga d'intestazione del file;
- Se nell'intestazione trova questi nomi di colonne allora si crea una tabella hash, che associa ad ogni nome di colonna di interesse la relativa posizione;
- Quando andrà ad esaminare ciascuna riga, utilizzerà la tabella hash creata precedentemente per prelevare le informazioni desiderate.

LETTORE CALCIATORI

Aggiorna club e rispettivi calciatori che ne fanno parte.

Scrive in un file di log, i nuovi calciatori che non hanno un'immagine propria, a cui è stata associata un'immagine di default.

LETTORE VOTI

Acquisisce da ciascuna riga la prestazione che un calciatore ha svolto in una partita di campionato. La prestazione comprende i gol e gli autogol che il calciatore ha fatto, i quali vengono utilizzati per determinare i risultati della partita in cui ha giocato.

```
Voti Ufficiosi;;;;;;;;
Giornata : 3;;;;;;
Leggenda Visualizzabile nel sito;;;;;;
ID;Giocatore;Ruolo;Ruolo2;Squadra;Min.Gioc.;G;GF;GS;Aut;Ass;CS;GF;GS;Aut;Ass;TS;GF;GS;Aut;Ass;M2;M3;Amm;Esp;Gdv;Gdp;RigS;RigP;Rt;Rs;T;VG;VC;VTS
154;SPORTIELLO M.;P;P;ATALANTA;;6,01;0;1;0;0;7;0;1;0;0;6;0;1;0;0;6,5;6,33;0;0;0;0;0;0;0;0;1;5;6;5
320;MASIELLO A.;D;D;ATALANTA;;7,01;1;0;0;0;7;1;0;0;0;7;1;0;0;0;7;7;0;0;0;0;0;0;0;0;1;10;10;10
350;RAIMONDI C.;D;D;ATALANTA;;6,01;0;0;0;0;6;0;0;0;0;6;0;0;0;0;6;6;0;0;0;0;0;0;0;0;6;6;6
379;TOLOI R.;D;D;ATALANTA;;6,01;0;0;0;0;6;5;0;0;0;6;5;0;0;0;0;6,25;6,33;1;0;0;0;0;0;0;1;5;5;6;6
397;ZUKANOVIC E.;D;D;ATALANTA;;6,02;0;0;0;0;6;0;0;0;0;6;5;0;0;0;0;6;6,17;0;0;0;0;0;0;0;1;6;6;6
416;KONKO A.;D;D;ATALANTA;;6,02;0;0;0;0;6;0;0;0;0;6;0;0;0;0;6;6;0;0;0;0;0;0;0;1;6;6;6
547;D'ALESSANDRO M.;C;T;ATALANTA;;6,02;0;0;0;0;7;0;0;0;0;7;0;0;0;0;6;5;6,67;0;0;0;0;0;0;0;1;6;7;7
```

LETTORE VOTI

Leggendo le prestazioni dei calciatori,
vengono generati i risultati delle partite.

GIORNATA			
n°	23	Stato	aperta
Atalanta	2	0	Cagliari
Bologna	1	7	Napoli
Chievo	0	0	Udinese
Empoli	1	1	Torino
Genoa	0	1	Sassuolo
Juventus	1	0	Inter
Milan	0	1	Sampdoria
Palermo	1	0	Crotone
Pescara	2	6	Lazio
Roma	4	0	Fiorentina

La prestazione acquisita viene utilizzata anche per aggiornare la statistica di quel calciatore. La statistica racchiude quindi un riepilogo di tutte le prestazioni che man mano un calciatore svolge durante il campionato, le quali sono utili a determinare un quadro completo del calciatore, come per esempio il suo rendimento generale, e quello nelle partite che gioca in casa o trasferta.

idcalciatore	stagione	presenze	gol	assist	golsubiti	rigoriparati	media	fantamedia	mediacasa	fantamediacasa	mediatrasf	fantamediatrasf	medianobonus	devstd	gialli	rossi
1	2016/2017	19	0	2	0	0	5,79	5,84	5,75	5,80	5,83	5,89	5,71	0,57	2	0
3	2016/2017	22	3	0	0	0	6,18	6,55	6,23	6,18	6,14	6,91	6,08	0,57	2	0
4	2016/2017	1	0	0	0	0	6,00	6,00	6,00	6,00	(NULL)	(NULL)	6,00	0,00	0	0
5	2016/2017	6	0	0	0	0	5,67	5,50	6,17	6,17	5,17	4,83	5,67	0,80	0	1
6	2016/2017	3	1	0	0	0	5,67	6,33	5,00	5,00	6,00	7,00	5,00	0,94	2	0
7	2016/2017	8	0	0	0	0	5,38	5,38	5,17	5,17	5,50	5,50	5,38	0,70	0	0

ZONA UTENTE

La zona utente racchiude quelle funzionalità utili per poter accedere alla piattaforma, gestire i propri dati e iniziare a giocare con gli amici.

Registrazione

l'utente deve inserire una email in suo possesso per ricevere una mail di conferma con cui attivare l'account.

Login/recupero password

tramite un apposita checkbox
l'utente può decidere di rimanere sempre connesso, creando una sessione persistente.

Profilo utente

è possibile modificare alcune informazioni del proprio profilo, come email o password, e aggiungere un'immagine di profilo.

Registrazione gruppo

l'utente che crea il gruppo ne diventa amministratore, e inserisce gli altri membri

Richieste

I membri che sono stati inseriti nel gruppo ricevono un richiesta di partecipazione, che possono accettare o rifiutare.

Avvio competizione

Dopo aver creato con successo un gruppo e scelto il tipo di competizione, l'amministratore di tale gruppo può avviare la competizione con i membri che hanno accettato.

REGISTRAZIONE GRUPPO

- All'amministratore del gruppo vengono forniti dei suggerimenti sui membri che si possono inserire, tramite l'autocomplete.
- L'autocomplete sfrutta una combinazione delle tecnologie Ajax e JQuery, per effettuare ricerche sugl'utenti il cui username inizia con i caratteri digitati dall'amministratore;
- Una volta che viene selezionato un utente dalla lista restituita dall'autocomplete, esso viene inserito in una lista di utenti da escludere nelle ricerche successive

REGISTRAZIONE GRUPPO

Membri

NOTA: scegliere un numero che sia compreso tra 1 e 11

3

CONFERMA

INFO!
Le caselle sono dotate di autocomplete. Se si stanno inserendo utenti registrati al sito, compariranno dei suggerimenti, altrimenti gli utente che si stanno inserendo non sono registrati al sito.

Membro n° 1
 diego.94

Membro n° 2
 El Pipita

Membro n° 3
 El
El Pipitanew
CREA NUOVO GRUPPO

Elements Console Sources Network »

top Filter Default levels ▾ 1 item hidden by filters

```
Ho aggiunto alla lista di utenti-> diego.94
▶ ["diego.94"]
Ho aggiunto alla lista di utenti-> mattia.biasco
▶ (2) ["diego.94", "mattia.biasco"]
Ho rimosso alla lista di utenti-> mattia.biasco
▶ ["diego.94"]
Ho aggiunto alla lista di utenti-> El Pipita
▶ (2) ["diego.94", "El Pipita"]
```

ZONA GIOCO



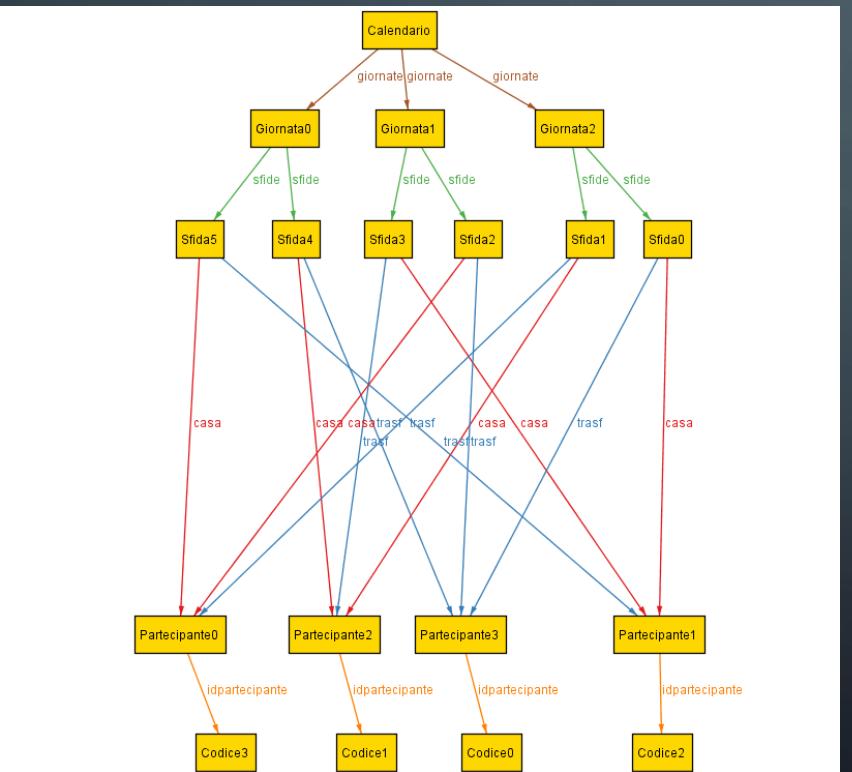
Generazione Calendario Requisiti

In una competizione a scontri diretti, deve essere generato un opportuno calendario di sfide tra i partecipanti, per le varie giornate di serie A rimanenti.

Dati quindi n partecipanti in una competizione e $n-1$ giornate, occorre generare una sfida per ogni possibile coppia distinta di partecipanti.

Ciascuna sfida deve essere associata in maniera casuale ad una giornata, in modo tale che entrambi i partecipanti coinvolti risultino “impegnati” in una sola sfida per quella particolare giornata.

Questi requisiti possono essere espressi in maniera più rigorosa, attraverso il linguaggio di specifiche **Alloy**. Il problema è stato tradotto utilizzando le *signature(model)* ed i *fact* (vincoli).



```

sig Partecipante
{
    idparticpane: one Codice,
}
sig Codice{}
//genera tanti codici quanti sono i partecipanti
{ #(Codice) = #(Partecipante) }

sig Giornata
{
    // numero: one Int./
    calendario: one Calendario,
}

sig Sfida
{
    casa: one Partecipante,
    trasf: one Partecipante,
    giornata: one Giornata,
}

one sig Calendario
{}

//Ogni partecipante possiede un codice univoco
fact CodiceUnivoco
{
    all p1,p2: Partecipante | (p1.idparticpane = p2.idparticpane) implies (p1 = p2)
}

fact disposizionePartecipanti
{
    all disj p1,p2: Partecipante | p2 in ((casa,p1).trasf) iff p1 not in ((casa,p2).trasf)
}

/*Le sfide si differenziano tra loro per la presenza di almeno un partecipante differente,
a prescindere dal fatto che giochi in casa o in trasferta.*/
fact combinazionePartecipanti
{
    all s1,s2: Sfida | ((s1.casa = s2.casa and s1.trasf = s2.trasf) or (s1.casa = s2.trasf and s1.trasf = s2.casa)) implies s1=s2
}

/*Una sfida coinvolge 2 partecipanti distinti.
*/
fact Sfide
{
    no s: Sfida | s.casa = s.trasf
}

//un partecipante in una determinata giornata è coinvolto in una sola sfida.
fact impegnoPartecipantePerGiornata
{
    all g: Giornata| all p: Partecipante| one s: (giornata,g)| s.casa = p iff not s.trasf = p
}

pred Genera
{
    #Partecipante = 6
}

run Genera for 15

```

Generazione Calendario Processo Risolutivo

L'algoritmo sviluppato genera casualmente una matrice M , di dimensione $[n, n-1]$ contenente le disposizioni di n elementi in classe 2. Vediamo un esempio in cui $n=4$.

Lista partecipanti			
	0	1	2
0			
1			
2			
3			

0 ->	03 , A	02 , R	01 , A
1 ->	02 , A	03 , R	00 , R
2 ->	01 , R	00 , A	03 , A
3 ->	00 , R	01 , A	02 , R

- L'indice di riga i ($0 \leq i \leq n$), rappresenta l'*i-esimo partecipante* che gioca in casa;
- L'indice di colonna J ($0 \leq J \leq n-1$) rappresenta la *J-esima giornata* nell'intervallo di giornate;
- $M(i,J)$ identifica l'oggetto sfida, con attributi:
 - Indice avversario;
 - Tipo: ANDATA/RITORNO

Matrice generata casualmente
 $D(n,2)$

0->	03,A	02,R	01,A
1->	02,A	03,R	00,R
2->	01,R	00,A	03,A
3->	00,R	01,A	02,R

SFIDE
ANDATA

0->	03	01
1->	02	
2->		00 03
3->		01



SFIDE
RITORNO

0->		02	
1->		03	00
2->	01		
3->	00		02

Giornate Serie A									...
ANDATA			RITORNO			ANDATA			...
0	1	2	3	4	5	6	7	8	...

GENERAZIONE DI 4 CALENDARI DIVERSI, PER UN GRUPPO DI 8 PARTECIPANTI.

06,R 07,A 01,R 03,A 04,R 02,A	00 05 06 07 01 03
04,R 06,A 00,A 07,A 03,R 05,A	01 02 04 06 00 07
03,R 04,A 07,R 06,A 05,R 00,R	02 01 03 04 07 06
02,A 05,A 04,R 00,R 01,A 06,A	03 07 02 05 04 00
01,A 02,R 03,A 05,A 00,A 07,A	04 06 01 02 03 05
07,R 03,R 06,R 04,R 02,A 01,R	05 00 07 03 06 04
00,A 01,R 05,A 02,R 07,R 03,R	06 04 00 01 05 02
05,A 00,R 02,A 01,R 06,A 04,R	07 03 05 00 02 01

05,R 01,A 04,R 03,A 07,R 02,A	00 06 05 01 04 03
04,R 00,R 07,R 02,A 06,R 03,A	01 05 04 00 07 02
07,R 06,A 03,R 01,R 05,R 00,R	02 04 07 06 03 01
06,R 05,A 02,A 00,R 04,R 01,R	03 07 06 05 02 00
01,A 07,A 00,A 05,A 03,A 06,A	04 02 01 07 00 05
00,A 03,R 06,R 04,R 02,A 07,A	05 01 00 03 06 04
03,A 02,R 05,A 07,A 01,A 04,R	06 00 03 02 05 07
02,A 04,R 01,A 06,R 00,A 05,R	07 03 02 04 01 06

02,R 03,A 05,R 07,A 01,R 04,A	00 06 02 03 05 07
07,R 04,A 06,R 02,A 00,A 03,A	01 05 07 04 06 02
00,A 05,A 03,R 01,R 07,R 06,A	02 04 00 05 03 01
05,R 00,R 02,A 04,A 06,R 01,R	03 07 05 00 02 04
06,R 01,R 07,R 03,R 05,R 00,R	04 02 06 01 07 03
03,A 02,R 00,A 06,A 04,A 07,A	05 01 03 02 00 06
04,A 07,A 01,A 05,R 03,A 02,R	06 00 04 07 01 05
01,A 06,R 04,A 00,R 02,A 05,R	07 03 01 06 04 00

06,R 04,A 02,R 05,A 07,R 01,A	00 03 06 04 02 05
04,R 03,A 07,R 06,A 05,R 00,R	01 02 04 03 07 06
03,R 05,A 00,A 04,A 06,R 07,A	02 01 03 05 00 04
02,A 01,R 05,R 07,A 04,R 06,A	03 00 02 01 05 07
01,A 00,R 06,R 02,R 03,A 05,A	04 07 01 00 06 02
07,R 02,R 03,A 00,R 01,A 04,R	05 06 07 02 03 00
00,A 07,A 04,A 01,R 02,A 03,R	06 05 00 07 04 01
05,A 06,R 01,A 03,R 00,A 02,R	07 04 05 06 01 03

INSERIMENTO ROSE

- L'amministratore del gruppo scegli il partecipante di cui creare la rosa;
- Sceglie il ruolo dei calciatori da inserire nella rosa, premendo un apposito pulsante;
- Seleziona dalla lista svincolati generata, il calciatore da aggiungere;
- Il calciatore viene tolto dalla lista svincolati e inserito nella rosa;
- Per ogni calciatore aggiunto inserisce il costo determinato durante l'asta tra amici, controllando il budget a disposizione del partecipante.

LISTA SVINCOLATI

ROSA PARTECIPANTI

Lista:

CALCIATORE	SQUADRA	GAZZETTA
Abate	Milan	10
Acerbi	Sassuolo	18
Adjapong	Sassuolo	3
Ajeti	Torino	4
Alex sandro	Juventus	17
Andelkovic	Palermo	4
Ansaldi	Inter	7
Anteи	Sassuolo	4
Antonelli	Milan	4
Astori	Fiorentina	11
Avelar	Torino	4
Barba	Empoli	4
Barzaglio	Juventus	11
Basta	Lazio	8
Bonucci	Juventus	13
Bovo	Pescara	4
Brivio	Genoa	4
Bubnjic	Udinese	2
Burdizzo	Genoa	9
Cacciatore	Chievo	7

RUOLO	CALCIATORE	SQUADRA	COSTO
P	 Audero	Juventus	1
P	 Neto	Juventus	1
P	 Buffon	Juventus	85
D	 Strinic	Napoli	1
D			

INSERIMENTO FORMAZIONE

L'inserimento di un calciatore nella formazione avviene usando il drag & drop.

All'interno di ciascuna riga della tabella rosa, alla cella che contiene un calciatore viene associato l'evento draggable. Inoltre gli viene associato un selettori id utile quando si effettuerà il drop.

RUOLO	CALCIATORE	GIOCA IN
P	Rafael	FC - N

A green arrow points from the table row to the corresponding HTML code block.

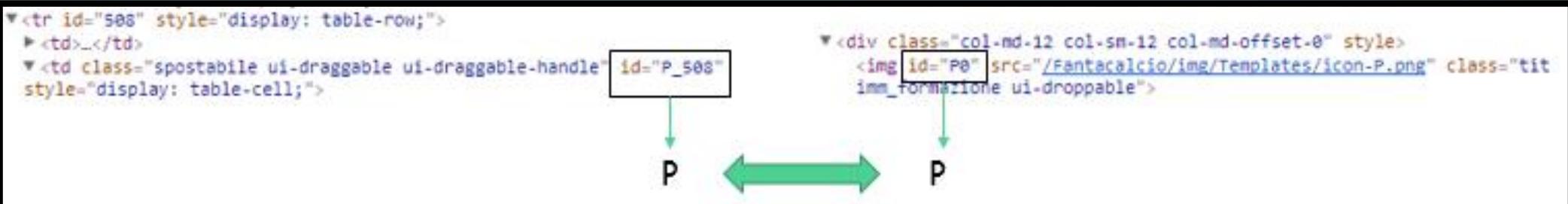
```
<tr id="508" style="display: table-row;">  |  | Rafael |  | |
```

Ciascun riquadro che contiene un calciatore titolare o panchinaro all'interno della formazione, avrà al suo interno un elemento a cui è associato l'evento droppable. Inoltre gli viene assegnato un selettore id che viene poi confrontato con quello assegnato al calciatore che si sta spostando.

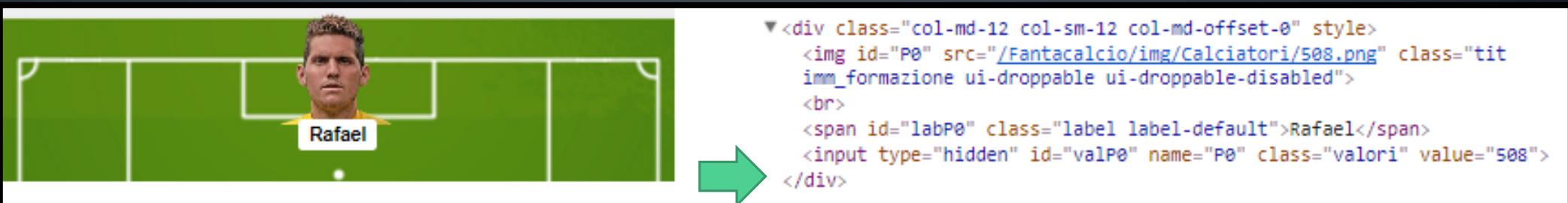


```
▼<div class="col-md-12 col-sm-12 col-md-offset-0" style="border: 1px solid black; padding: 5px">
  
  <br>
  <span id="labP0" class="label label-default">P</span>
  <input type="hidden" id="valP0" name="P0" class="valori" value="P">
</div>
```

Nel momento in cui si rilascia un calciatore in uno dei precedenti riquadri, viene chiamata una funzione Javascript che per prima cosa acquisisce e confronta il primo carattere dei due selettori id.



Se sono uguali allora il calciatore e i suoi dati vengono posizionati in quel riquadro.



Il partecipante deve inserire la formazione prima che le partite che si svolgono in quella giornata inizino ufficialmente. Nella pagina dedicata all'inserimento della formazione viene mostrato un countdown, per ricordare al partecipante quanto tempo ha a disposizione.

Countdown 23 giornata:

000 days **23:59:27** hours minutes seconds

INSERIMENTO FORMAZIONE THREAD DEMONE

- È stato definito un processo che viene mandato in esecuzione quando l'applicazione viene avviata, e effettua controlli ciclici a determinati intervalli.
- Nel momento in cui scade il tempo utile per inserire la formazione di una giornata, invoca un metodo che prende tutti quei partecipanti che non hanno inserito una formazione entro la scadenza, ma che ne avevano inserita una nella giornata precedente.
- Per questi partecipanti carica la formazione precedente anche per la giornata ora in corso di svolgimento.

ELABORA GIORNATA

Quando una giornata viene chiusa, si effettua un calcolo completo sui punteggi ottenuti da tutte le formazioni schierate dai partecipanti, in quella giornata. Se un gruppo svolge la competizione a sfide, questi punteggi vengono usati per determinare l'esiti delle sfide.

Infine viene aggiornata per ogni gruppo la rispettiva classifica.

CLASSIFICA

- La classifica varia a seconda del tipo di competizione che il gruppo ha scelto;
- Data una lista di partecipanti il suo ordinamento viene effettuato usando il metodo polimorfo `sort`, appartenente alla classe `java.util.Collections`;
- La classe Java interessata implementa l'interfaccia Java `Comparable<E>`, in particolare `compareTo(E obj)`;
- Partecipanti e Partecipanti scontri diretti hanno una definizione del metodo `compareTo` differente.

CONCLUSIONI

La piattaforma permette agli utenti di giocare a fantacalcio, tuttavia ci sono tante altre funzionalità che potrebbero essere implementate in futuro per migliorare l'esperienza dell'utente, come per esempio:

- Grafici sugli andamenti dei calciatori;
- Utilizzare un protocollo crittografico SSL per le comunicazioni Client-Server;
- Dati per ciascun calciatore i parametri costo e fantamedia, risolvere un [problema di programmazione lineare binaria](#) che suggerisce quali calciatori acquistare(variabili decisionali binarie), avendo a disposizione un certo budget e dovendo comprare un certo numero di calciatori per ruolo. L'obiettivo è massimizzare la qualità della rosa (fantamedia complessiva). I vincoli sono imposti dal budget e dal numero di calciatori acquistare. La risoluzione viene effettuata implementando un apposito algoritmo della ricerca operativa.