UNIVERSITÀ DEGLI STUDI DI PADOVA

Corso di Laurea Specialistica in Ingegneria Informatica

Tesi di Laurea

PROGETTAZIONE E SVILUPPO DI UN BROWSERGAME DI TCHOUKBALL

Relatore:
Prof. Sergio Congiu

Laureando:
Puppo Manuele

Anno Accademico 2012-2013

Indice:

Alla fine di questo mio percorso di studi universitari vorrei ringraziare tutte le persone che mi hanno accompagnato ed incoraggiato negli ultimi anni
Un ringraziamento particolare va alla mia famiglia per avermi supportato nei momenti difficili comprendendo le mie necessità

1 - Introduzione:

I Browsergame (videogiochi per browser) sono una tipologia di videogioco utilizzabile attraverso internet, tramite un qualsiasi browser, connettendosi al rispettivo sito web. Questi giochi nella maggior parte dei casi non necessitano di alcuna installazione o download di applicazioni aggiuntive, solo eventualmente di alcuni plugin. L'utilizzo di componenti lato server permette di interagire con altri giocatori iscritti al medesimo gioco. La mancanza di installazione permette agli utenti di controllare e impostare il loro stato di gioco da una qualsiasi postazione internet, pur non essendo davanti al proprio computer.

Il Tchoukball (letto *ciukbol*) è uno sport di squadra nato in Svizzera negli anni sessanta. Si gioca in squadre avversarie di 7 elementi su un terreno di gioco delle stesse dimensioni di un campo da basket. Sono necessari 2 speciali pannelli delle dimensioni di 1 metro per 1 metro collocati alle estremità del campo e una palla da pallamano di taglia 2. Il campionato italiano vene giocato esclusivamente all'interno di palestre ma il tchoukball può essere praticato su ogni superficie comprese erba e sabbia. In tutte le varianti l'area antistante al pannello è un semicerchio di raggio 3 metri.
Il gioco inizia con la rimessa da fondo campo, a fianco di uno dei pannelli, e la squadra in possesso di palla ha a disposizione tre passaggi per costruire un'azione prima di attaccare lanciandola contro il pannello che, dotato di una rete elastica, la fa rimbalzare. Lo scopo è fare in modo che questo rimbalzo tocchi il terreno di gioco all'esterno dell'area realizzando così un punto. Se gli avversari riescono ad intercettare ed a bloccare saldamente il rimbalzo, il gioco riprende immediatamente e la squadra che ha difeso passa istantaneamente all'attacco. La particolarità di questo gioco è che si può attaccare, cioè tirare, indifferentemente su entrambi i pannelli che sono a disposizione di entrambe le squadre, per un massimo di tre attacchi consecutivi. Durante i passaggi la palla non può essere intercettata dagli avversari e non si possono in nessun modo ostacolare i giocatori nei loro movimenti.
In questo modo quindi viene creato un sistema dove viene premiato il gioco di squadra attraverso cambi veloci di pannello dove si svolge l'azione o finte, l'attaccante è libero di esprimere le sue capacità tecniche senza nessun ostacolo imposto da un gioco di contatto con gli avversari.

L'obbiettivo del lavoro di progettazione e sviluppo è quello di costruire un Browsergame di Tchoukball manageriale dove è possibile creare la propria squadra, allenare i singoli giocatori, impostare tattiche e cambi in occasione di particolari eventi durante la partita che viene simulata automaticamente ogni domenica. Ogni partita poi a differenza degli altri browsergame manageriali classici potrà essere visualizzata e seguita minuto per minuto.

2 - Strumenti utilizzati:

Gli strumenti utilizzati per la realizzazione della piattaforma si possono dividere in due categorie: gli strumenti diretti e gli strumenti indiretti.
Gli *strumenti diretti* sono tutte le tecnologie che permangono nella struttura del progetto dalle sue prime fasi di sviluppo alla pubblicazione finale nella memoria del web server che fornisce il servizio di hosting e il dominio di riferimento al sito web del browsergame. Tali posso essere classificati i linguaggi di programmazione, script, strutture dati e strumenti per la gestione che devono servire costantemente il progetto.
Gli *strumenti indiretti* invece sono tutti i software e simulatori che ne hanno permesso esclusivamente la produzione, ma che non saranno più necessari se non in caso di manutenzione o modifica della piattaforma.

Strumenti diretti:

HTML (HyperText Markup Language) - Strumento base per la programmazione ipertestuale la cui sintassi stabilita dal World Wide Web Consortium (W3C) è pubblica. Non è un linguaggio di programmazione in quanto non prevede l'uso di qualsiasi funzione o strutture dati, ma solo un linguaggio di markup che descrive le modalità di impaginazione, formattazione o visualizzazione grafica dei contenuti di una pagina web utilizzando tag di formattazione. Tuttavia supporta l'inserimento di script ed oggetti esterni.

CSS (Cascading Style Sheets) - Strumento di supporto al semplice HTML che permette di definire la formattazione di documenti HTML. Permette una programmazione più chiara e facile da utilizzare sia per gli autori che producono i documenti, che per gli utenti che possono modificarne la visualizzazione.

PHP (PHP HyperText Preprocessor) - Come indicato dall'acronimo ricorsivo il PHP invece è un linguaggio di scripting di livello superiore all'HTML. Originariamente concepito per la produzione di pagine web dinamiche, è utilizzato principalmente per sviluppare applicazioni web lato server. L'elaborazione di codice PHP da parte del server produce codice HTML da inviare al browser dell'utente che ne fa richiesta, oppure può essere utilizzato per l'elaborazione di richieste SQL da inviare al database per operazioni di gestione o manutenzione anche temporizzate (con il supporto di software come Crontab).

SQL (Structured Query Language) - E' un linguaggio di interrogazione per database relazionali. Progettato per leggere, modificare e gestire informazioni memorizzate in un sistema di gestione dati, può essere incluso nella programmazione PHP per interagire in modo dinamico sulla pagina web in modo trasparente rispetto all'utilizzatore del sistema.

Crontab - Comando nei sistemi operativi Unix-like che consente la pianificazione di esecuzioni, ossia consente di registrare una cronologia di istruzioni (in questo caso script PHP) presso il sistema per essere poi mandati in esecuzione periodicamente. Il comando fa riferimento a un demone chiamato *crond*, esso non è altro che un software costantemente in esecuzione in background (non necessariamente a controllo diretto dell'utente), che segue, in questo caso, una cronologia di istruzioni da processare. E’ molto utile per quanto riguarda ogni operazione temporizzata di manutenzione o di gioco prevista dalla piattaforma.

Actionscript 3.0 - E' il linguaggio di scripting di Adobe Flash molto simile a Javascript. Grazie alla diffusione del suo plugin utilizzato per la realizzazione di applicazioni dinamiche, siti web e animazioni. E’ un linguaggio di alto livello non compilato ma interpretato, la sua esecuzione è affidata al compilatore nativo di Flash Player. Nel progetto viene utilizzato per ogni sorta di animazione e data la sua grande collezione di librerie si presta all'utilizzo web, lato server infatti può richiamare da output l'esecuzione di script PHP attraverso URL che possono includere quindi anche variabili.

AJAX (Asynchronous Javascript and XML) - E' una tecnica di sviluppo di applicazioni web interattive. Questa tecnologia asincrona e si basa su uno scambio di richieste in background tra browser e server che coinvolge dei dati extra caricati in modo completamente trasparente che causano l'aggiornamento dinamico di pagine web (o di sue porzioni) in modo indipendente da esplicite richieste di aggiornamento dell'utente. Nel caso della piattaforma in esame AJAX è prevalentemente utilizzato per la validazione in tempo reale dei dati inseriti nei campi di testo o per l'aggiornamento istantaneo di porzioni di codice.

SMTP (Simple Mail Transfer Protocol) – Protocollo standard per la trasmissione via internet di email. Solitamente utilizzato dalle caselle di posta elettronica, è possibile utilizzarlo attraverso PHP per l’invio, come in questo caso, di email a utenti che hanno completato la procedura di registrazione nuovo account.

Strumenti indiretti:

Apache HTTP Server - Piattaforma server web che realizza le funzioni di trasporto delle informazioni, di internetwork e di collegamento. Utilizzato in locale dà la possibilità di testare interamente il sistema progettato, utilizzando nell'insieme ogni aspetto dello sviluppo citato come potrà accedervi l'utente finale una volta completato e pubblicato.

Oracle MySQL Server - Servizio che mette a disposizione un database relazionale gestibile grazie all'utilizzo della sintassi SQL. Il suo utilizzo è largamente diffuso nei sistemi LAMP (Linux Apache MySQL PHP).

Adobe Flash Professional CS5 - Software per uso prevalentemente grafico che consente di creare animazioni vettoriali principalmente per il web. Viene utilizzato inoltre per creare giochi o interi siti web. Arricchito poi dalla possibilità di utilizzo dell'Actionscript nelle sue animazioni. Per il progetto è essenziale nella creazione delle animazioni dinamiche grazie alla possibilità di esportarle in formato *swf*. Questi oggetti sono includibili nel codice il quale permette la loro esecuzione con passaggio di parametri e restituzione di output.

3 - Struttura del sito che permette l'esecuzione del Browsergame:

Un Browsergame è un videogioco creato per essere utilizzato attraverso la rete internet e la sua visualizzazione avviene quindi attraverso pagine web. I documenti html pertanto sono la struttura stessa del gioco, che utilizzando le tecnologie e gli strumenti indicati, rendono le pagine dinamiche ed i contenuti interattivi.
Il webserver impone che alla richiesta via URL del sito internet esso risponda di default inviando il codice HTML prodotto dalle istruzioni del documento *index.php*. Questo documento oltre che essere una presentazione grafica, gestisce il riconoscimento della sessione e la procedura di login. Se non si possiede un account è necessario procedere alla registrazione e all'attivazione dello stesso, in questo modo viene creato fisicamente nel database l'utente, la relativa squadra e vengono generati i primi 12 giocatori.
Una volta effettuato il login viene creata una sessione che mantiene attivo il riconoscimento dell'utente ad ogni passaggio della navigazione nei documenti che compongono la piattaforma, essa può scadere dopo un tempo determinato di inattività oppure attraverso il comando "*logout*" che distrugge la sessione e riconduce alla visualizzazione di *index.php*.
A sessione attiva è possibile da ogni pagina visualizzata accedere ad un menù sempre attivo che permette di spostarsi da un documento all'altro e visualizzare quindi i propri giocatori, le tattiche, il campionato, le partite da disputare in calendario, l'allenamento, il mercato ed il bilancio settimanale.
Dopo aver introdotto rapidamente la struttura della piattaforma segue una spiegazione completa di ogni sua parte per definire le dinamiche complesse nascoste all'utente utilizzatore finale.

3.1 - Login e Registrazione Utente:

Entrambe le procedure di login e registrazione utente sono legate alla pagina iniziale del sito internet che compone il Browsergame, *index.php*. Da essa si possono digitare le proprie credenziali di Username e Password accedendo così al proprio account oppure procedere alla registrazione tramite la procedura definita dal codice nel documento *form.php.*

index.php è strutturato in modo da determinare immediatamente se esiste o meno una sessione attiva. Al fine di attivare il parsing di linguaggio PHP si utilizza il tag di apertura "*<?php*" e quello di chiusura "*?>*" in questo modo tutto ciò che è contenuto tra i due tag verrà interpretato come codice PHP all'interno del documento che normalmente viene trattato come interamente composto da HTML. L'estensione stessa del file "*.php*" che si differenzia dal più comune "*.html*" (o "*.htm*" indica che il documento ha la possibilità di contenere entrambe le tipologie di codice.
Tramite l'istruzione predefinita di PHP "*session_start()*" vengono inizializzati i dati di una sessione o si riprende quella corrente basata sull'id o un cookie. In mancanza di questa istruzione la variabile di sistema "*$_SESSION[]*" di fatto non esiste.
Attraverso l'istruzione if successivamente il codice verifica se una sessione è di fatto impostata, "*isset()*" restituisce true se a "*auth*" nell'array di sistema "*$_SESSION[]*" è stato assegnato un valore in fase di creazione di una sessione precedentemente, se il valore restituito invece è false significa che non è stata creata una sessione e quindi l'utente non si è mai autenticato oppure la sessione stessa è scaduta:

```
                          ----- index.php -----
<?php

session_start();

if(isset($_SESSION['auth'])){
                           //la sessione esiste
} else {
                           //la sessione non esiste, procedi al
login/registrazione
}

?>
                             ----- end -----
```

Nel caso venga riconosciuta attiva la sessione, viene visualizzata la home page. Oltre che a un riepilogo dei dati utente, vi è il collegamento alla pagina di visualizzazione del proprio campionato ed un riassunto delle ultime partite disputate. In ogni documento della navigazione, dopo essersi autenticati, è possibile avviare la procedura di logout tramite il link alla pagina di *logout.php*:

```
<a href='logout.php'>Logout</a>
```

Questo script non contiene nessun elemento grafico pur essendo un documento in formato “*.php*“ *logout.php* si compone di solo tre istruzioni (oltre ai necessari tag) che permettono di fare scadere la sessione. La prima funzione già esaminata in precedenza (`session_start();`) permette di utilizzare i dati di sessione attivandola relativamente a questa pagina, la seconda invece, “`session_destroy();`” è una funzione predefinita che distrugge tutti i dati associati alla sessione corrente. La terza e ultima istruzione “`header( 'Location: index.php' );`”. si tratta di una redirezione istantanea a *index.php* che una volta eseguita non rileverà più nessuna variabile inizializzata in “*$_SESSION[]*” riconducendo in questo modo l'utente alla procedura di login o registrazione nuovo utente.

----- *logout.php* -----

```
<?php

session_start();
session_destroy();
header( 'Location: index.php' );

?>
```

----- *end* -----

Se la sessione invece non viene riconosciuta come attiva, la pagina presenta il form nel quale inserire le proprie credenziali al fine di effettuare il login ed il link alla pagina di registrazione nuovo utente.
Questa parte di codice è in puro HTML. La struttura contenuta nei tag `<form>` e `</form>` contiene i campi testo da riempire con le proprie credenziali specificati dal tag `<input type='text'>` ed il pulsante per inviare i dati da `<input type='submit'>`. Il campo di testo contrassegnato invece da `<input type='password'>` è in grado di nascondere il testo in via di inserimento, mascherando ogni carattere con un punto o un asterisco al suo posto, ma il contenuto resta ben definito al momento della trasmissione dei dati.
Una volta inserite le credenziali di Username e Password, si può procedere attraverso il bottone submit. I valori inseriti vengono passati come parametri al documento *login.php* attraverso il metodo *_POST*.
L'ultima istruzione è il collegamento alla pagina di registrazione nuovo utente, procedura descritta dal documento *form.php*.

----- index*.php (login e registrazione)* -----

```
<form action='login.php' method='POST'>"
Username:<br>
<input type='text' name='username' maxlength='20'><br><br>
Password:<br>
<input type='password' name='password' maxlength='20'><br><br>

<input type='submit' value='Login'>
</form>
<a href='form.php'>Click here</a> for create your account
```

----- *end* -----

Form di login

Form di login con tasto di invio dati, il link in basso porta alla procedura di registrazione nuovo utente

Il metodo *_POST* per lo scambio di variabili tra documenti permette di inoltrare richieste a script contenuti in altri documenti PHP, è possibile utilizzare anche il metodo *_GET*. La differenza tra i due metodi in questione verrà affrontata nei prossimi capitoli.

In questa prima parte del codice contenuto in *login.php* viene utilizzata la procedura di connessione al servizio del server MySQL.
La prima istruzione permette di specificare il charset utilizzato dalla pagina in modo da poter trattare con ogni tipo di carattere non standard (lettere accentate o caratteri stranieri). In questo modo ogni carattere speciale viene compreso ed utilizzato correttamente invece di rischiare di sbagliare codifica e salvarlo in modo errato.
Successivamente vengono inizializzate le variabili necessarie alla connessione al server MySQL, username e password per l'autenticazione, l'URL al server (che in questo caso localhost o 127.0.0.1) ed il database di riferimento che contiene le tabelle ed i record riguardanti utenti, relativi dati di gioco ed impostazioni standard.
La funzione `mysql_connect($hostname, $username, $password);` prende come parametri le credenziali e l'hostname per stabilire un'effettiva connessione con il database. E' chiaro che le istruzioni in PHP quindi possono connettersi a qualsiasi URL specificato, anche non in locale.
Successivamente la funzione `mysql_select_db($database);` seleziona il database all'interno del server MySQL con il quale si vuole interagire.
E' buona abitudine specificare nuovamente il charset da utilizzare per evitare problemi di compatibilità, lo si fa utilizzando l'istruzione `mysql_query("SET NAMES utf8");`.

Una volta connessi al server è possibile interagire con le vere e proprie tabelle e record contenute nel database specificato. E' possibile quindi utilizzare ogni tipo di comando espresso rigorosamente in sintassi SQL sempre nei limiti dei permessi assegnati all'utente del quale sono state precedentemente analizzate le credenziali.

Il vantaggio di poter utilizzare stringhe di comandi SQL all'interno del codice PHP lo si può notare immediatamente alla prima query del codice:

```
"SELECT * FROM users WHERE username='".$_POST['username']."'"
```

Questa stringa descrive in sintassi SQL la query che verrà utilizzata sul database "*tchoukball*" essa seleziona ogni record, grazie al carattere "*" dalla tabella *users* nel quale il campo "*username*" è uguale a "*$_POST['username'] "*.
In PHP ogni variabile ha il prefisso "*$*". Si intuisce che questa variabile è il parametro passato da *index.php* tramite il form per il login.
Grazie alle proprietà di concatenazione stringhe di PHP il valore passato può essere introdotto in una query per ricavarne altri dati utili dal database. Essendo impossibile al momento della registrazione utilizzare un username che è già presente nel database, il record restituito sarà solo uno, quello con l'username corrispondente.
Attraverso il comando `$dlin = mysql_query();` , fornendo come parametro la stringa stessa, la query viene inoltrata al server MySQL ed il risultato, in forma di struttura simile a un array viene memorizzato in "*$dlin*".
`$ilin = mysql_fetch_array($dlin);` poi permette di trasferirne il contenuto in "*$ilin*" che diventa un vero e proprio array con indici testuali, uno corrispondente a ogni campo del record in output dal database. E' quindi possibile gestire i dati specificandone l'indice testuale in questo modo `$ilin['password'];`

La password poi passata allo stesso modo di username viene per ragioni di sicurezza convertita in MD5 e salvata inizializzando la variabile `$mdp5 = md5($_POST['password']);`
La funzione `md5();` verrà approfondita successivamente.

----- *login.php* (1) -----

```
header('Content-type: text/html; charset=utf-8');

$username = "user";
$password = "pass";
$hostname = "localhost";
$database = "tchoukball"

$dbh = mysql_connect($hostname, $username, $password)
     or die("Unable to connect to MySQL");

@mysql_select_db($database) or die( "Unable to select database");

mysql_query("SET NAMES utf8");

$dlin = mysql_query("SELECT * FROM users WHERE
username='".$_POST['username']."'")or die(mysql_error());

$ilin = mysql_fetch_array($dlin);

$md5p = md5($_POST['password']); //conversione md5 della password in
input
```

----- *end* -----

In questa parte del codice contenuto in *login.php* è possibile analizzare l'algoritmo che autentica l'utente e crea la sessione.
Innanzitutto con la prima istruzione "*if*" viene verificato che il campo password salvato nell'array "*$ilin*" corrisponda alla password inserita dal form di autenticazione di *index.php* convertita in MD5, se il controllo fallisce la pagina restituisce in output HTML un messaggio di errore ed il link a *index.php* per ripetere la procedura. Al contrario se il controllo dà esito positivo viene verificato attraverso "`If($ilin['ver'] == 1)`", che l'account sia stato attivato attraverso l'URL fornito via email (campo richiesto alla registrazione). Se il controllo fallisce, un secondo messaggio di errore informa l'utente che l'attivazione non è avvenuta.
Nel caso in cui entrambi i controlli diano esito positivo significa che l'account è attivo e la password è corretta, autenticato quindi l'utente viene attivata la sessione e la relativa variabile "*$_SESSION[]*". In essa vengono inizializzati gli indici "*auth*" che conferma l'avvenuta autenticazione e la variabile "*user_id*" dell'utente che ha completato la procedura.

```
                        ----- login.php (2) -----

If ($ilin['password'] == $md5p){

     If($ilin['ver'] == 1){

          session_start();
          $_SESSION['auth'] = 1;
          $_SESSION['user_id'] = $ilin['username'];
          header( 'Location: index.php' );

     }else{

          Print"Welcome ".$_POST['username']."., your account have to be
activate, check your email from tbgame.com and follow the instructions";

     }

}else{
     Print "Login Failed, check your username and password -> <a
href='index.php'>Login</a>";
}

?>
                             ----- end -----
```

Nel caso invece che l'utilizzatore del sistema non possieda le credenziali per potervi accedere, è necessario che le crei attraverso la procedura di registrazione utente descritta dal documento "*form.php*".
La visualizzazione di questo file si presenta come un elenco di campi di testo dove inserire i propri dati e listbox dove selezionare un valore.
Il codice presenta il metodo di connessione al database "*tchoukball*" nel server MySQL già descritto in *login.php* e successivamente si ripresenta la struttura del form di login contenuto in *index.php*, in questo caso però è composto da elementi che lo rendono ben più complesso. La pagina infatti prevede l'ausilio di una serie di funzioni *javascript* che permettono l'utilizzo della tecnologia AJAX per la validazione dei dati inseriti nei campi di testo in tempo reale ed il controllo dinamico delle listbox per la selezione dello stato e relativa regione di provenienza dell'utente.

Ogni campo di testo è impostato in modo che dinamicamente, ad ogni nuovo carattere inserito, faccia apparire alla sua destra un simbolo che indica la validità (una V verde) o meno (una X rossa) dei dati inseriti. Nel caso dei dati come “Nome” o “Cognome” viene verificato che il campo non sia nullo, in campi invece come “Nome Squadra” o “Username” il controllo è ben più complesso e approfondito.

Per quanto riguarda le listbox invece relative a “Stato” e “Regione”, i valori sono estratti dal database attraverso una query e l’utilizzo di AJAX in è progettato solo per cambiare dinamicamente il contenuto della listbox “Regione” a seconda di quale stato viene selezionato nella prima listbox.

----- *form.php (form)*-----

```
<form action="newuser.php" method="POST">

Nome:<br>
<input type="text" name="name" maxlength="15" onkeyup="ckNulln(this.value)"><a
id="checkNn"></a><br><br>

Cognome:<br>
<input type="text" name="surname" maxlength="15"
onkeyup="ckNulls(this.value)"><a id="checkNs"></a><br><br>

Nome Squadra:<br>
<input type="text" name="team" maxlength="25" onkeyup="ckTeam(this.value)"><a
id="checkT"></a><br><br>

Stato:<br>
<...>

Regione:<br>
<...>

Username:<br>
<input type="text" name="username" maxlength="20"
onkeyup="ckUser(this.value)"><a id="checkU"></a><br><br>

Email:<br>
<input type="text" name="email" maxlength="60" onkeyup="ckEmail(this.value)"><a
id="checkE"></a><br><br>

Password:<br>
<input type="password" name="password" maxlength="20"
onkeyup="ckP(this.value)"><a id="checkP"></a><br><br>

Ripeti Password:<br>
<input type="password" name="password2" maxlength="20"
onkeyup="ckP2(this.value)"><a id="checkP2"></a><br><br>
<input type="submit" value="Submit">

</form>
</body>
</html>
```

----- *end* -----

Esaminiamo ora come esempio per tutti i campi di testo “Nome Squadra”.
Oltre a `<input type="text">` già esaminato in precedenza insieme a tutte le altre impostazioni è importante porre l’attenzione `onkeyup="ckTeam(this.value)"`. Questo parametro indica che la funzione *ckTea*m deve essere eseguita fornendo come parametro il contenuto stesso del campo di testo, la funzione si attiva quando l’evento *onkeyup* viene rilevato, ossia quando l’utente aggiorna il campo inserendo al suo interno qualsiasi carattere da tastiera. Teniamo presente inoltre che alla fine del campo è presente un elemento `<a id="checkT"></a>`. Questo semplice elemento ora è privo di contenuti e quindi a livello HTML non viene visualizzato, ma ad esso è assegnato l’id *checkT* , molto importante per i passaggi successivi.

```
Nome Squadra:<br>
<input type="text" name="team" maxlength="25" onkeyup="ckTeam(this.value)"><a
id="checkT"></a><br><br>
```

La funzione *ckTea*m è definita in javascript nella head di *form.php* all’interno del tag
`<script language="javascript" type="text/javascript"></script>`
All’interno di questo tag in effetti vengono definite tutte le funzioni relative a ogni riferimento di questo tipo. Qui di seguito la funzione *getXMLHTTP* che verifica il tipo di richiesta per questioni di compatibilità ed è necessaria all’esecuzione di tutte le singole funzioni relative a ogni campo di testo. Successivamente la funzione *ckTeam* permette al file *ckteam.php*, tramite lo script contenuto al suo interno, di eseguire le dovute verifiche sui contenuti del campo di testo che viene passato come parametro attraverso la costruzione della variabile URL: `var URL="ckteam.php?s="+str;`
L’output dell’esecuzione di *ckteam.php* verrà utilizzato attraverso il paragrafo vuoto indicato prima marcato dall’id *checkT* attraverso l’istruzione *document.getElementById(‘checkT’).*

----- form.php (script 1) -----

```
<html>
<head>

<title>Form registrazione nuovo utente</title>
<script language="javascript" type="text/javascript">

function getXMLHTTP() {
      var xmlhttp=false;
      try{
            xmlhttp=new XMLHttpRequest();
      }
      catch(e)    {
            try{
                  xmlhttp= new ActiveXObject("Microsoft.XMLHTTP");
            }
            catch(e){
                  try{
                        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
                  }
                  catch(e1){
                        xmlhttp=false;
                  }
            }
      }
      return xmlhttp;
}
```

```
function ckTeam(str) {
      var URL="ckteam.php?s="+str;
      var reqt = getXMLHTTP();
      if (reqt) {
            reqt.onreadystatechange = function() {
                  if (reqt.readyState == 4) {
                        // only if "OK"
                        if (reqt.status == 200) {
                  document.getElementById('checkT').innerHTML=reqt.responseText;
                        } else {
                              alert("reqt.status = \n" + reqt.status);
      alert("There was a problem while using XMLHTTP:\n" + reqt.statusText);
                        }
                  }
            }
            reqt.open("GET", URL, true);
            reqt.send(null);
      }
}

</script>
</head>
<body>
...
```

----- *end* -----

form.php attraverso la procedura in javascript invia una richiesta quindi a *ckteam.php* corredata di parametro (contenuto del campo di testo). Lo script di *ckteam.php* accetta il parametro e lo salva come contenuto della variabile "*$s*" calcolando immediatamente il suo numero di caratteri e salvandolo in "*$st*". Successivamente dopo la solita procedura di connessione al server, seleziona tutti i record della tabella *teams*. Impostate le variabili di controllo di lunghezza minima e non nulla a 2 (valore neutro tra 0 e 1 ossia *true* o *false*), viene impostata a *true* il valore di non esistenza "*$bEx*" (risparmiando una eventuale assegnazione successiva). "*$bT*" invece indica il successo di tutti i controlli parziali.

Utilizzando l'istruzione while è possibile scorrere tutto l'array di record preso dal database e confrontare quindi se esiste il testo digitato tra i nomi delle squadre presenti in memoria. Se non si rileva corrispondenza la variabile relativa resta *true* (= 1) ed il controllo è passato, in caso contrario viene settata a 0, quindi *false*. I successivi controlli in serie verificano le lunghezze minime (>= a 3 caratteri) e che il parametro non sia nullo.
Il controllo finale tramite `$bT = $bEx * $bZ * $bM;` garantisce che ogni verifica parziale sia *true* (quindi = 1) per poter assegnare a "*$bT*" il valore *true*.
Il risultato finale è grafico, ossia una stringa HTML visualizzabile. La stringa sarà vuota in caso di campo nullo e sarà un'immagine negli altri casi, X rossa in caso di controllo non passato o una V verde in caso di esito positivo.

Questo risultato grafico in output verrà sostituito al paragrafo `<a id="checkT"></a>` all'interno del codice di *form.php*. Paragrafo che si trova accanto al campo di testo "Nome Squadra" che indicherà in questo modo la validità o meno del testo scritto al suo interno in tempo reale.

----- *ckteam.php* -----

```
<?php

<...>

$s=$_GET["s"];
$st = strlen($s);

<...>

$data = mysql_query("SELECT * FROM teams")or die(mysql_error());

$bEx = 1; //bool true if doesn't exist
$bZ = 2;    //bool true if lenght not = 0
$bM = 2; //bool true if lenght > 3

$bT = 0; //bool final

$gna = "";

while($info = mysql_fetch_array( $data )) //team exist in db
{
If ($info['team'] == $s) {
            $bEx = 0;   //false if team exist
            $gna = $info['team'];
      } else {
//          $bEx = 1;   //true if team doesn't exist
      }
}



If ($st < 3) {
      $bM = 0;    //team name too short = false
      If ($st == 0) {
            $bZ = 0;    //false if team name = NULL
      } else {
            $bZ = 1; //true if team name not NULL
      }
} else {
      $bM = 1; //true if team name > 2
      $bZ = 1; //true if team name not NULL
}

$bT = $bEx * $bZ * $bM; //final check
If ($bT == 0) {
      If ($bZ == 0) {
      Print "";

      } else {
            Print "<img src='img/cross.png' height='15' width='15' align='top'
            hspace='10' vspace='5' title='almost 2 characters'/>";
      }
} else {
      Print "<img src='img/tick.png' height='15' width='15' align='top'
      hspace='10' vspace='5' title='ok'/>";
}
mysql_close($dbh);
?>
```

----- *end* -----

Tale procedura si applica a tutti i campi di testo nella pagina di registrazione nuovo utente a seconda delle esigenze di controllo relative a che tipo di testo esso dovrà contenere. Degno di nota il caso del campo email, governato dall'output del file *ckemail.php*, che oltre alle verifiche generiche sul numero di caratteri e sull'esistenza nel database, effettua anche un controllo sulla composizione utilizzando una *RegExp* ossia una Espressione Regolare.
Una *RegExp* è una sequenza di simboli che identifica un insieme di stringhe. Ossia definisce una funzione che prende in ingresso una stringa e restituisce un valore booleano (*true*/*false*) a seconda che la stringa segua o meno un certo pattern. Nel caso l'Espressione Regolare utilizzata è:

```
$pattern = "^([a-zA-Z0-9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(@)
([a-zA-Z0-9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(\.[a-z]{2,4})$";
```

le stringhe del tipo *a-z* indicano l'intervallo di lettere dalla a alla z, lo stesso per *A-Z* e per *0-9*. Ciò che è contenuto tra *[* e *]* invece è un semplice elenco di simboli. I raggruppamenti vengono effettuati tra parentesi tonde ed gli * indicano che quei simboli possono essere ripetuti da 0 a infinite volte. Il simbolo + impone una concatenazione e espressioni come *{2,4}* indicano la lunghezza minima e massima della stringa. Questa complicata Espressione Regolare descrive una stringa del tipo indirizzo email, quindi che sia composta da un certo numero di caratteri alfanumerici o simboli seguiti da una @, seguita da altri caratteri alfanumerici o simboli seguiti da un punto e ancora da soli 2 o 4 caratteri. Per esempio *indirizzo@email.it* è una stringa accettata dal pattern definito e quindi contenuta in esso. E' comunque nell'interesse dell'utente inserire un valido indirizzo dato che è solo tramite questo che l'account può essere attivato.

Il secondo caso in cui viene utilizzato AJAX riguarda la dinamicità di aggiornamento della listbox "Regione" relativamente al valore selezionato per la listbox "Stato".
I meccanismi di scambio informazioni tra le pagine è identico al precedente, in questo caso però non è un paragrafo esterno ad essere aggiornato, ma un vero e proprio elemento complesso del form, in particolare i suoi contenuti.
La listbox relativa allo stato "sfoglia" il risultato di una query alla tabella "*country*" e tramite la sintassi `<OPTION value=".$info['ID'].">".$info['country']."</OPTION>` ordina al suo interno i valori del campo "*country*" relativa ai record estratti, impostando come opzione selezionata la stringa "*Select Country*". La listbox dipendente "Regione" invece contiene solo il valore selezionato "*Choose Country First*". In questo modo il sistema è inizializzato.
L'unica opzione della listbox "Regione" è identificata dall'id sull'elemento chiamata "*statediv*" e la funzione di attivazione dello script di AJAX è `onchange ="showReg(this.value)"`.
Funzione "*showReg*" che verrà richiamata ad ogni cambio di selezione della listbox passando come parametri il valore "*id*" del record selezionato.

----- form.php (listbox) -----

```
Stato:<br>
<SELECT name="country" onchange="showReg(this.value)">
<option selected="selected">Select Country</option>
<?php
while($info = mysql_fetch_array( $data ))
{
Print "<OPTION value=".$info['ID'].">".$info['country']."</OPTION>";
}
Print "</SELECT><br>";
?>
<br>
Regione:<br>
<SELECT id="statediv" name="region"><option>Choose Country
First</option></SELECT>
<br><br>
```

----- end -----

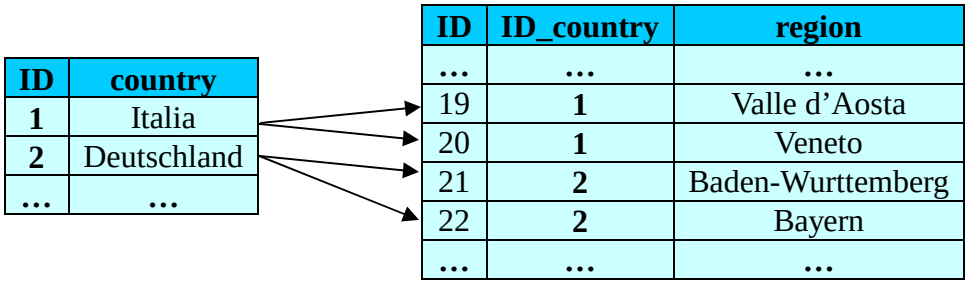
Di seguito lo script della funzione "*showReg*" definita dal codice di *form.php*. Essa viene eseguita sempre attraverso la funzione "*getXMLHTTP*" definita precedentemente in *form.php (script 1)* e definisce la richiesta, arricchita da passaggio di parametri al documento *getreg.php*.
I dati restituiti agiranno sul tag HTML identificabile dal suo "*id*" (*statediv*).

----- *form.php (script 2)* -----

```
function showReg(countryId) {
      var strURL="getreg.php?q="+countryId;
      var req = getXMLHTTP();
      if (req) {
            req.onreadystatechange = function() {
                  if (req.readyState == 4) {
                        // only if "OK"
                        if (req.status == 200) {

      document.getElementById('statediv').innerHTML=req.responseText;

                        } else {
                              alert("req.status = \n" + req.status);
      alert("There was a problem while using XMLHTTP:\n" + req.statusText);
                        }
                  }
            }
      req.open("GET", strURL, true);
      req.send(null);
      }
}
```

----- *end* -----

Come già descritto, i documenti correlati ad azioni di questo tipo, sotto il controllo di javascript e della tecnologia AJAX, restituiscono un output grafico che andrà a sostituire dinamicamente la visualizzazione di alcuni elementi marcati con un "*id*" definito, che in questo caso prende il nome di "*statediv*" in *form.php*, all'interno delle opzioni della listbox "Regione".
Successivamente al salvataggio del parametro "*id*" relativo allo stato passato dalla funzione javascript, si procede controllando se la variabile della prima listbox è stata selezionata. Se il valore quindi della listbox "Stato" è ancora a "*Select Country*", il campo "Regione" resta inizializzato a "*Choose Country First*".
In caso contrario invece la procedura avvia la connessione al server SQL eseguendo una query alla tabella "Region"; essa contiene i record di tutte le regioni differenziate per gruppi a seconda della variabile "*ID_country*". Questo id (oltre a quello di chiave primaria, perciò autoincrementale e univoco), segnala a che stato ogni regione appartiene, stabilendo così una relazione tra le tabelle.



ID	country
1	Italia
2	Deutschland
…	…

ID	ID_country	region
…	…	…
19	1	Valle d'Aosta
20	1	Veneto
21	2	Baden-Wurttemberg
22	2	Bayern
…	…	…

Questo schema riassume la struttura delle tabelle "country" e "region". La relazione tra le due si basa sulla chiave primaria della prima in relazione al valore del campo "ID_country" della seconda.

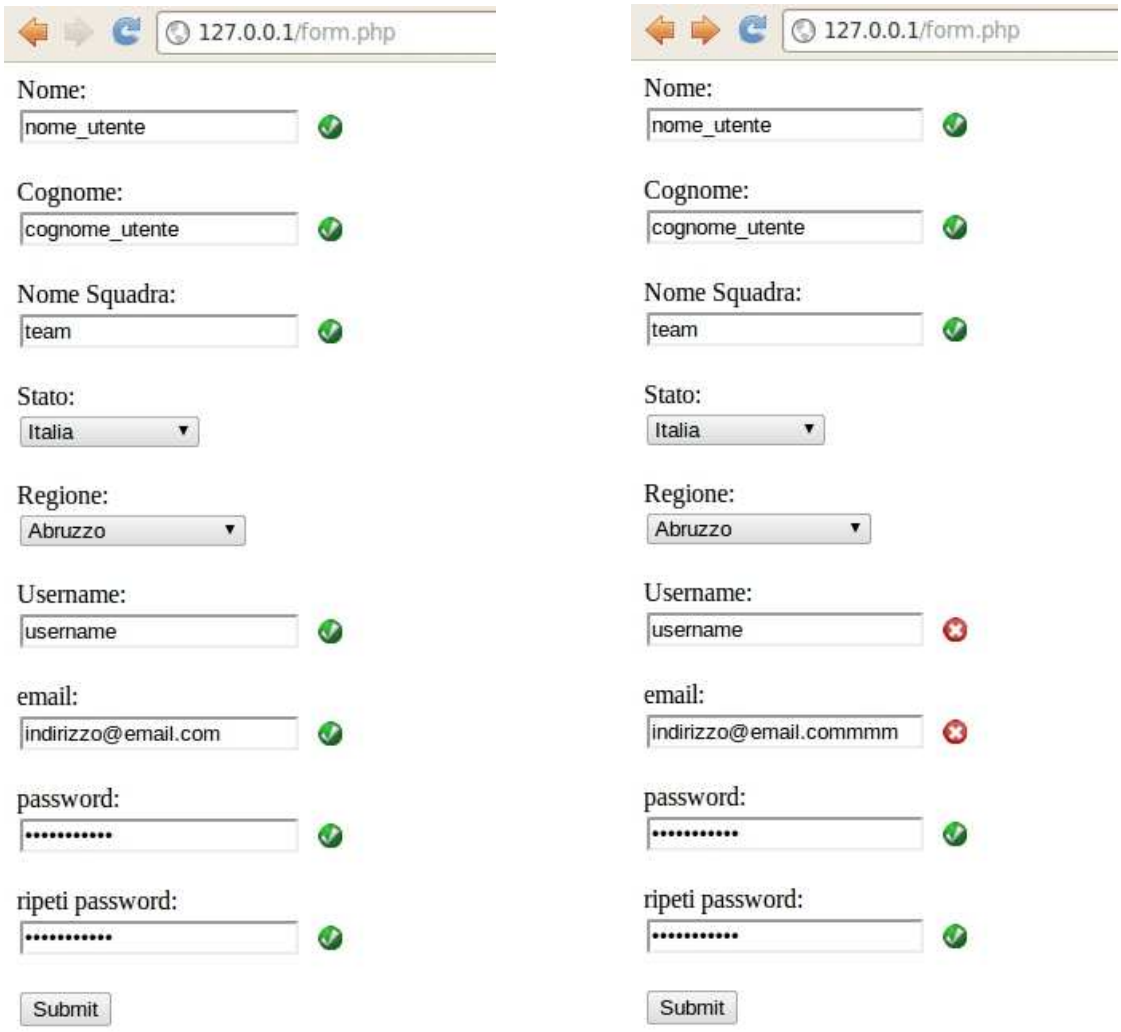
Selezionati quindi tutti i record che abbiano come “*ID_country*” quello passato via parametro, viene composta una nuova listbox e quindi utilizzata come output in *form.php* in sostituzione al paragrafo HTML marcato con id “*statediv*”.

----- *getreg.php* -----

```
<?php

<...>

$q=$_GET["q"];

If ($q == "Select Country") {

Print "<SELECT name='region'>";
Print "<option selected>Choose Country First</option>";
Print "</SELECT>";

} else {

<...>

$data = mysql_query("SELECT * FROM region WHERE ID_country='".$q."'")or
die(mysql_error());

Print "<SELECT name='region'>";

$val=1;
while($info = mysql_fetch_array( $data ))
{
Print "<OPTION value=".$val.">".$info['region']."</OPTION>";
$val=$val+1;
}

Print "</SELECT>";

}

<...>

?>
```

----- *end* -----

Ritorniamo quindi alla visualizzazione del form composto dal codice di *form.php*.
Una volta validati graficamente è possibile procedere.



Form di registrazione utente: a sinistra un esempio di compilazione corretta, ogni campo viene graficamente validato. A destra invece un esempio di compilazione scorretta, in questo caso "username" è già presente nel database e l'email invece è chiaramente composta in modo non conforme al pattern di controllo.

Alla pressione del tasto"*Submit*" i dati collezionati nel form vengono inviati con metodo "*_POST*" al documento *newuser.php*.
Questo documento molto complesso effettua nuovamente il controllo su ogni campo e se i dati non sono validati è in grado di descrivere via codice HTML la lista degli errori rilevati nei dati con i quali si desidera effettuare la registrazione. Data la sua lunghezza e complessità viene quindi affrontato per parti.
La prima in esame si occupa delle variabili fondamentali al controllo dell'input.
Successivamente al salvataggio in variabili di ogni dato passato al codice dal form di provenienza, viene eseguita la procedura di connessione al database esaminata in precedenza.
Ogni input poi viene utilizzato come parametro alla funzione `strlen();` essa provvede a determinare da quanti caratteri è composta una stringa. Il valore viene utilizzato per inizializzare un secondo elenco di variabili di una quantità corrispondente a quello precedente relativo ai valori veri e propri.

Per ovvi motivi, i valori delle listbox non vengono trattati nello stesso modo essendo dei semplici valori numerici di id che identificano direttamente nel database “Stato” e ”Regione”.
Questi sono gli unici che non vengono “trattati” nello stesso modo degli altri da questa porzione di codice. La loro validazione è stata precedentemente forzata a causa di una scelta limitata a dei valori preimpostati. Resta solo da verificare che un valore sia stato effettivamente scelto.

L’ultima sequenza di istruzioni inizializza un grande numero di variabili. Ognuna di esse riferita ad un singolo campo del form di immissione. Per semplicità l’ordine di ognuna di loro è relativo all’ordine di collezione iniziale e di utilizzo nelle successive parti di codice, la loro descrizione presente anche nel documento php sottoforma di commento (`//commento ininfluente all’esecuzione`) rende la procedura di più semplice comprensione.

----- newuser.php (controllo 1) -----

```
<?php

<...>

$n=$_POST["name"];
$s=$_POST["surname"];
$t=$_POST["team"];
$c=$_POST["country"];
$r=$_POST["region"];
$us=$_POST["username"];
$e=$_POST["email"];
$p1=$_POST["password"];
$p2=$_POST["password2"];

<...>       //procedura connessione al database tchoukball

//determino lunghezza in caratteri di tutti gli input tranne le listbox
$nl = strlen($_POST["name"]);
$sl = strlen($_POST["surname"]);
$tl = strlen($_POST["team"]);
//$cl = strlen($_POST["country"]); essendo valori predefiniti non ha senso
//$rl = strlen($_POST["region"]); essendo valori predefiniti non ha senso
$usl = strlen($_POST["username"]);
$el = strlen($_POST["email"]);
$p1l = strlen($_POST["password"]);
$p2l = strlen($_POST["password2"]);

//variabili di controllo -> se tutto è = 1 si può procedere
$ckn = 0; //true se name rispetta parametri (da 2 a 15 caratteri)
$aln = ""; //avviso di errore su campo name

$cks = 0; //true se surname rispetta parametri (da 2 a 15 caratteri)
$als = ""; //avviso di errore su campo surname

$ckt = 1; //true se team rispetta parametri (da 3 a 25 caratteri) e di
univocità (diventa 0 se trova corrispondenza)
$alt = ""; //avviso di errore su campo team

$ckcou = 0; //true se valore country è valido
$alcou = ""; //avviso di errore su campo country

$ckreg = 0; //true se valore region è valido
$alreg = ""; //avviso di errore su campo region

$ckus = 1; //true se username rispetta parametri (da 3 a 20 caratteri) e di
univocità(diventa 0 se trova corrispondenza)
$alus = ""; //avviso di errore su campo username
```

```
$cke = 1; //true se team rispetta parametri di univocità  e conformità  al
pattern (diventa 0 se trova corrispondenza)
$ale = ""; //avviso di errore su campo email

$ckp1 = 0; //true se password rispetta parametri (da 9 a 20 caratteri)
$alp1 = ""; //avviso di errore su campo password

$ckp2 = 0; //true se password2 rispetta parametri (da 9 a 20 caratteri)
$alp2 = ""; //avviso di errore su campo password2

$ckpwd = 0; //true se i campi password inseriti sono uguali
$alpwd = ""; //avviso di errore su confronto campi password

                                ----- end -----
```

La seguente sezione di codice prende in esame tre particolari tipologie di controllo sull'input.

La prima tipologia di controllo si effettua sui campi "Nome" e "Cognome" che tra i due è identico, esso valida la stringa solo se la sua lunghezza in caratteri è compresa tra i 2 ed i 15 caratteri. Da notare che l'input è già limitato in *form.php* durante la fase di inserimento dati, ma per questioni di sicurezza che verranno approfondite in seguito, il controllo viene ripetuto.
In caso di successo la variabile di controllo inizializzata precedentemente "*$ckn*" viene impostata = 1 (*true*), in caso contrario resta impostata a *false* e vengono invece assegnate alle rispettive variabili le cause dell'errore attraverso "*$aln*".

Si rivela ben più complesso il controllo sul campo "Nome Squadra" il quale, oltre a validare la lunghezza in caratteri del parametro (tra i 3 ed i 25 caratteri), deve anche verificare che il nome scelto non esista già tra i record della tabella *teams*.
Le istruzioni procedono quindi eseguendo una query che permette di prelevare dal server SQL ogni record relativo alla tabella "`SELECT * FROM teams`". Successivamente il ciclo *while* permette di confrontare il campo trasmesso con ogni valore del campo "*team*" dei record estratti.
Nel caso di questo dato la variabile di controllo è stata inizializzata a *true*, quindi se si verifica una corrispondenza, essa viene cambiata in *false* (= 0) e viene impostato il messaggio d'errore come nel caso precedente per procedere successivamente con gli altri controlli descritti.
E' d'obbligo precisare che questo medesimo controllo è effettuato attraverso la tecnologia AJAX in *form.php*.

Il terzo tipo di controllo è riferito ai campi "Stato" e "Regione". La validazione si limita a ispezionare l'id passato. Se esso non è numerico significa che la listbox non è stata utilizzata. Da qui gli errori vengono impostati evidenziando il problema. In caso di successo del controllo invece la relativa variabile viene impostata a *true*.

Come ultima parte di codice la validazione del campo “Username”. La procedura è identica a quella già descritta relativa al campo “Nome Squadra”, l’unica differenza è che i caratteri inseriti hanno limite superiore pari a 20.

----- *newuser.php (controllo 2)* -----

```
//controllo su campo name
If ($nl > 1) {
      If ($nl < 16) {
            $ckn = 1;
      } else {
      $aln = "Your name must have a lenght between 2 and 15 characters";
      }
} else {
      $aln = "Your name must have a lenght between 2 and 15 characters";
}

<...>       //controllo su campo surname
//controllo su campo team
If ($tl > 2) {
      If ($tl < 26) {
            $dteam = mysql_query("SELECT * FROM teams")or die(mysql_error());

            while($iteam = mysql_fetch_array( $dteam ))
            {

                  If ($iteam['team'] == $_POST["team"]) {
                        $alt = "Your team already exists";
                        $ckt = 0;
                  }

            }
      } else {
            $ckt = 0;
            $alt = "Your team name must have a lenght between 3 and 25
characters";
            }
} else {
      $ckt = 0;
      $alt = "Your team name must have a lenght between 3 and 25 characters";
}

//controllo su campo country
If (is_numeric($_POST["country"])){
      $ckcou = 1;
} else {
      $alcou = "You must choose a country";
}

<...>       //controllo su campo region

<...>       //controllo su campo username
```

----- *end* -----

Questa terza porzione di codice riguardante la validazione dati in *newuser.php* inizia occupandosi del campo “Email” e dei due campi “Password”.

I dati derivanti dal campo “Email” vengono analizzati nello stesso modo descritto per “Nome Squadra” e “Username” verificando la presenza di oggetti identici nella tabella *users*.
Si noti però che come accade attraverso la tecnologia AJAX in *form.php* viene eseguito un controllo anche dell’appartenenza al pattern specificato attraverso espressione regolare. PHP prevede questo tipo di controlli fornendo allo sviluppatore la funzione `ereg($pattern,$_POST["email"]);` che mette a confronto i parametri in argomento e restituisce *true* se il valore passato rientra nel pattern.
L’errore conseguente a esito negativo indica all’utente il formato che deve avere il dato inserito (xx@yy.zz).

Entrambi i campi “Password” vengono analizzati allo stesso modo controllando solo il numero di caratteri inseriti che deve essere compreso tra 9 e 20.
Più interessante il controllo che li mette in relazione, quello digitato all’interno dei due campi di testo infatti deve corrispondere dato che il secondo contiene la ripetizione del primo.
Per questioni di sicurezza le password devono essere trattate sempre convertendole in MD5, quindi per prima cosa, lo script salva in “*$hp1*” e “*$hp2*” le conversioni che ora possono essere trattate.
Se le variabili coincidono la relativa variabile di controllo viene impostata a *true*, in caso contrario il messaggio d’errore avviserà l’utente che ha digitato la password erroneamente.

----- *newuser.php (controllo 3)* -----

```
//controllo su campo email

$pattern = "^([a-zA-Z0-9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(@)([a-zA-Z0-
9])+([a-zA-Z0-9]+[-_\.]?)*([a-zA-Z0-9])+(\.[a-z]{2,4})$";

$cksp = ereg($pattern,$_POST["email"]); //true se email è contenuto nel pattern

If ($cksp == 1) {
            $demail = mysql_query("SELECT * FROM users")or die(mysql_error());
            while($iemail = mysql_fetch_array( $demail ))
            {

            <...>       //controllo if su database

            }
} else {
      $cke = 0;
      $ale = "Your email must have this structure: xx@yy.zz";
}

<...> //controllo su campo password

<...> //controllo su campo password2

//controllo di uguaglianza tra password
$hp1 = md5($_POST["password"]);     //conversione hash MD5 password
$hp2 = md5($_POST["password2"]);    //conversione hash MD5 password2
If ($hp1 == $hp2) {
      $ckpwd = 1;
} else {
      $alpwd = "Your password is different from repeated password";
}
```

----- *end* -----

Il seguente codice sfrutta le validazioni e le informazioni collezionate fino ad ora al fine di eseguire un controllo globale che permetterà la vera e propria registrazione utente.
Al fine di verificare che ogni indice sia impostato a *true*, quindi = 1, la variabile di controllo globale "*$ckq*" viene inizializzata effettuando una semplice moltiplicazione tra ogni indice descritto nella prima parte del file *newuser.php*. Se anch'esso risulta uguale a 1 (quindi *true*), significa che tutti i controlli su ogni singolo campo hanno dato esito positivo.
Per comodità poi viene inizializzato l'oggetto "*$alq*" come un array di tutte le variabili nelle quali sono stati precedentemente impostati eventuali stringhe contenenti messaggi d'errore, ossia le cause del fallito controllo.

A questo punto non resta che utilizzare concretamente ogni informazione rilevata. Viene quindi utilizzata l'istruzione if per verificare che l'assegnazione di "*$ckq*" abbia dato risultato *true*.
Il risultato positivo di questo controllo porta all'esecuzione di una serie di istruzioni che concludono il processo creando nel database effettivamente un nuovo utente aggiornando la collezione di record nella tabella *users* e avviando una serie di altre procedure più complesse che riguardano a fondo la struttura dei dati salvati.
La mancata positività al controllo esprimibile invece attraverso un'istruzione `else` provocherebbe solo la visualizzazione della lista di errori che hanno portato a questo risultato. Questa casistica sarà successivamente analizzata.

----- newuser.php (controllo finale) -----

```
//controllo finale validità di tutte le restrizioni dei campi
$ckq = $ckn * $cks * $ckt * $ckcou * $ckreg * $ckus * $cke * $ckp1 * $ckp2 *
$ckpwd;
$alq = array($aln, $als, $alt, $alcou, $alreg, $alus, $ale, $alp1, $alp2,
$alpwd); //array contenente tutti i messaggi di errore

If ($ckq == 1){
```

----- end -----

La successiva parte di codice provvede ad aggiornare il database vero e proprio utilizzando ogni elemento elaborato fino ad ora.
Innanzitutto attraverso la seguente sintassi SQL la tabella *teams* viene aggiornata memorizzando quindi il nome squadra scelto:

```
"INSERT INTO teams (ID, team) VALUES ('','".$_POST['team']."')"
```

ID	team
1	Nome Squadra 1
2	Nome Squadra 2
3	…

Struttura della tabella "teams" che contiene l'elenco delle squadre di tutto il gioco

La struttura della tabella è semplicissima, ogni nome squadra viene salvato nel campo "*team*", come è facilmente intuibile dalla relativa query SQL. Si noti che nella stessa query al campo "*team*" viene assegnato il valore della variabile *"$_POST['team']"*, mentre al campo "*ID*" non viene assegnato nessun valore. Questo perché negli attributi del campo stesso del record è indicato che il campo non

può essere NULL (quindi senza valore) ed essendo la *chiave primaria*, che deve essere univoca, è autoincrementata. Non è necessario quindi specificare un valore, il server SQL assegna automaticamente al campo un valore univoco ricavato dall'ultimo record creato.
La procedura è talmente precisa e importante che nel caso ci siano *n* valori e, per qualche ragione l'*n-esimo* venisse eliminato, in caso di una successiva aggiunta di record, l'incremento automatico riserverebbe l'*n-esimo* valore incrementando direttamente l'"*ID*" a *n+1*.

Le istruzioni successive eseguono una query "*SELECT*" sulla stessa tabella *teams* al fine di identificare il valore del campo "*ID*" della squadra appena creata utilizzando la corrispondenza tra il campo "*team*" e "*$_POST['team']*". Questo dato verrà successivamente utilizzato per creare la corrispondenza *utente – squadra*.

----- *newuser.php (teams)* -----

```
      $sql="INSERT INTO teams (ID, team) VALUES ('','".$_POST['team']."')";
      mysql_query($sql,$dbh);

      $dqt = mysql_query("SELECT * FROM teams")or die(mysql_error());
      while($iqt = mysql_fetch_array( $dqt ))  //team presenti nel database
      {
            If ($iqt['team'] == $_POST['team']) {    //trova ID nuovo team
inserito
                  $idteam = $iqt['ID'];
            }
      }
```

----- *end* -----

La seguente porzione di codice è fondamentale riguardo la sicurezza. La registrazione nuovo utente infatti non si conclude al termine dell'esecuzione del documento *newuser.php* che stiamo analizzando. Al termine di questa fase infatti l'utente, per poter accedere al suo nuovo account appena creato, dovrà attivarlo attraverso una procedura di validazione dei dati inseriti. Uno specifico URL fornito via email completerà la registrazione, ma analizzeremo in seguito le meccaniche del codice coinvolto.
Al fine di produrre un metodo che renda univoca la sua iscrizione al gioco è necessario generare una "key" di 40 caratteri casuali alfanumerici. "*$char*" contiene l'elenco di tutti i caratteri permessi e "*$dim*" indica la lunghezza della stringa che deve essere prodotta.
Il ciclo for infine "riempie" carattere per carattere dallo 0 al 39 (indici per le posizioni dei caratteri) la variabile "*$key*" con un carattere scelto in posizione casuale rispetto alla variabile "*$char*". La variabile "*$key*" inizializzata a stringa vuota viene di ciclo in ciclo riempita concatenando ad essa un carattere alla volta.

La variabile "*$ver*" identifica, se è inizializzata a 0, l'account non attivo, ovviamente ora non può essere impostata a 1 essendo l'account in via di creazione. Per questioni descrittive, la variabile è citata, ma non è necessario utilizzarla nella query di inserimento finale essendo il suo campo nel database impostato in modo che il valore predefinito sia appunto 0, successivamente modificabile.

Infine viene generato il “Timestamp” della registrazione, ossia l’identificazione in secondi del momento in cui è stata avviata questa procedura, per poter tracciare il momento esatto in cui l’utente ha voluto registrarsi al sistema.
Utile non solo a fine estetico o informativo rispetto all’utente, ma anche per poter tracciare eventuali inattività, statistiche oppure per poter effettuare al meglio le operazioni di manutenzione temporizzata sui dati del server.
Grazie alla funzione “*time();*” fornita da PHP è possibile attraverso un’unica funzione stabilire il tempo corrente nel formato numero di secondi trascorsi dal 1 Gennaio 1970 (Unix Epoch).
Il valore sarà quindi semplicemente, seppur notevolmente grande, un tipo dato intero.

PHP mette anche a disposizione delle funzioni per la visualizzazione di tale dato, come specificato nel commento del codice, è possibile utilizzando “*date();*” visualizzare questa data in modo ottimale specificando ogni frazione di tempo, dagli anni fino ai secondi.

----- *newuser.php (key)* -----

```
//generazione valore key_value;

      $char = "aAbBcCdDeEfFgGhHiIlLjJkKmMnNoOpPqQrRsStTuUvVwWxXyYzZ0123456789";
      $dim = 40;
      srand((double)microtime()*1000000);
      $key = '' ;
      for($inc=0; $inc<$dim; $inc++){
            $rand = rand(0, strlen($char)-1);
            $scar = substr($char, $rand, 1);
            $key = $key . $scar;
      }


//verifica account per il momento = 0, verrà verificato via email più tardi

      $ver = 0; // predefinito alla creazione di un nuovo record


//timestamp della registrazione
      $treg = time();

//    $timev = date('d M Y - H:i:s', $treg); VISUALIZZAZIONE FORMATTATA
```

----- *end* -----

Giunti alla conclusione di ogni tipo di validazione dei dati inseriti dall’utente e della determinazione di eventuali altri dati, è finalmente possibile passare alla vera e propria scrittura dei dati in memoria. Questa operazione si può effettuare attraverso una query di inserimento SQL. La struttura della query già vista ma mai esaminata accuratamente si esprime secondo la seguente sintassi:

```
INSERT INTO users (id, username, ...) VALUES('','".$_POST['username']."', ...)
```

“*INSERT*” specifica il tipo di query, che in particolare è di inserimento in tabella, la sintassi “*INTO users*” indica che la tabella di destinazione è *users*. La complessa stringa definita nel codice è in realtà di semplice comprensione, basta infatti semplificarla per capire che la struttura è molto semplice. Alla serie di campi indicati all’interno della prima parentesi è sufficiente assegnare i valori nello stesso ordine presenti nella seconda.

Struttura dati per quel che riguarda la tabella users, ogni record è formato dall'unione delle due tabelle sottostanti

id	username	email	password	key_control	ver
1	User1	utente1@email.it	500bd40ft6…	0	1
2	User2	utente2@mail.com	482c811tf5…	4ghYtFr6NBeh…	0
3	…	…	…	…	

La stringa nel campo password per questioni di sicurezza è salvata in conversione MD5. key_control è un campo che contiene una stringa lunga 40 caratteri solo se ver è impostato a 0 e quindi l'account relativo al record non è ancora stato attivato. Se invece ver = 1 significa che l'account è attivo e utilizzabile.

name	surname	team	country	region	time
Marco	Bianchi	1	1	8	1344698578
Mario	Rossi	2	1	16	1345284265
…	…	…	…	…	…

I valori del campo team si riferiscono alla chiave primaria delle squadre memorizzate nella tabella teams, come country e region che si riferiscono alle relative tabelle.

I campi come "*id*" oppure "*ver*" che il database imposta autonomamente non necessitano di un valore particolare da assegnare, ma devono essere comunque "citati" nella sintassi per poter mantenere l'ordine della successione dei campi da elaborare. Utilizzando quindi '' ossia NULL, i valori da assegnare vengono calcolati ed utilizzati dal server MySQL in accordo alle regole prestabilite nel sistema.

```
----- newuser.php (creazione record) -----

//inserimento in tabella users

      $insq="INSERT INTO users (id, username, email, password, key_control,
name, surname, team, country, region, time) VALUES ('','".$_POST['username']."',
'".$_POST['email']."', '".$hp1."', '".$key."', '".$_POST['name']."',
'".$_POST['surname']."', ".$idteam.", ".$_POST['country'].",
".$_POST['region'].", ".$treg.")";
      mysql_query($insq,$dbh) or die(mysql_error());

----- end -----
```

Se la procedura di inserimento dati ha successo è necessario a questo punto impostare le basi per la successiva attivazione dell'account via email. A questo punto l'ipotetico utente che ha inserito correttamente le proprie credenziali, per poter utilizzare la piattaforma, deve come ultimo passo della registrazione nuovo utente attivare il proprio account.
Per fare ciò esso dovrà accedere alla casella di posta elettronica relativa all'indirizzo email fornito al server e accedere all'URL indicato nel messaggio di posta appena ricevuto. In seguito saranno analizzate le meccaniche della procedura.

Il codice quindi una volta avvenuta la prima registrazione dovrà provvedere ad inoltrare un'email utilizzando il servizio SMTP messo a disposizione dal web server che offre l'hosting.
Al fine di sfruttare tale servizio PHP mette a disposizione una semplice funzione, l'istruzione "*mail();*" Inizializzando infatti poche variabili come destinatario "*$dest*", oggetto dell'email "*$oggetto*", e corpo stesso del messaggio "*$body*" come semplici stringhe, la funzione "*mail();*" potrà sfruttale al fine di inviare un semplicissimo messaggio di posta elettronica. Ispezionando il codice è facile notare come il PHP, grazie all'utilizzo di variabili, renda immediata l'intera

procedura. E' possibile inoltre arricchire l'output utilizzando dei parametri aggiuntivi come mittente, allegati o header.
L'invio dell'email è seguito dalla visualizzazione via HTML di un avviso di completamento della procedura che invita il nuovo utente ad attivare l'account.

----- *newuser.php (invio email)* -----

```
//PROCEDURA INVIO EMAIL (SERVER SMTP)

      $dest = $_POST['email'];
      $oggetto = "www.tbgame.com - Account activation";
      $body = "Dear ".$_POST['username']." welcome in www.tbgame.com\n\nActivate
your account -> www.tbgame.com/auth_key.php?key=".$key." and enjoy Tchoukball";

      mail($dest, $oggetto, $body);

      Print "Thank you ".$_POST['name']."<br>, an email has been sent to your email
adress<br>Follow the instruction to activate your new account";
```

----- *end* -----

Nel caso in cui la validazione dei valori inseriti nel form di *form.php* non abbia successo, la procedura descritta in queste ultime pagine non avviene, il controllo quindi effettuato dall'istruzione `If ($ckq == 1)` fallisce e ne consegue che viene eseguito ciò che il codice descrive dopo l'istruzione `else` nel seguente codice.
Come descritto precedentemente, tutti gli eventuali messaggi d'errore inizializzati in fase di validazione vengono ordinati all'interno dell'array "*$alq*". Ne consegue che per poterli visualizzare è necessario creare un ciclo for, che permette di stamparli in HTML in modo ordinato.
Il controllo interno al ciclo stabilisce che, nel caso la stringa contenuta in ogni indice dell'array `$alq[]` sia vuota, non accada nulla. In caso contrario il codice manda in output la stringa che descrive l'errore ed attraverso il tag HTML `<br>` punti ad una nuova riga preparando così la visualizzazione ordinata di un eventuale altro messaggio d'errore che spieghi all'utente quale dato è stato inserito in modo errato.
Terminato il ciclo la pagina visualizzata fornisce un link per poter retrocedere nella cronologia del browser di una sola pagina. Utilizzando la funzione `onclick='history.back();` infatti è possible farlo in modo che i dati inseriti non siano persi, come accadrebbe invece facendo la stessa operazione sfruttando un link che invii una nuova richiesta al server di *form.php*.

----- *newuser.php (else)* -----

```
} else {

      for ($i = 0; $i <= 9; ++$i) {
            If ($alq[$i] == ""){
            } else {
            Print $alq[$i]."<br>";
            }
      }

Print "<input type='button' value='Back to Form' onclick='history.back();'>";

}

?>

</body>
</html>
```

----- *end* -----

Terminata l'analisi di questo documento, in accordo con i passaggi che deve affrontare l'utente per poter utilizzare la piattaforma, segue l'attivazione dell'account appena creato. Non è possibile ancora infatti utilizzare il sistema, come descritto nell'analisi relativa a *login.php*, dove risultava chiaro che per poter accedere al sito internet autenticandosi con le proprie credenziali, era necessario che il campo "*ver*" del record relativo all'utente che tenta di portare a termine la procedura di login fosse impostato a 1. Fintanto che l'attivazione non è avvenuta esso sarà impostato a 0.
L'utente quindi accedendo alla casella di posta elettronica riferita all'indirizzo email inserito nella pagina *form.php*, dovrà visualizzare il messaggio inviato automaticamente dal server che si presenterà nella seguente forma:

> ***Mittente:*** www.tbgame.com
> ***Oggetto:*** www.tbgame.com - Account activation
>
> Dear **User1** welcome in www.tbgame.com
>
> Activate your account ->
> www.tbgame.com/auth_key.php?key=gDh7GF45yFdjn4hsnG5f567FQfE3HgfR1ax4G2h5
> and enjoy Tchoukball

Analizzando il messaggio è chiara la presenza di un URL composto da più parti:

www.tbgame.com/auth_key.php?key=gDh7GF45yFdjn4hsnG5f567FQfE3HgfR1ax4G2h5

si può individuare "www.tbgame.com" che da solo punterebbe a *index.php*, quindi alla home del sito, ma in questo caso è seguito da "/auth_key.php". Questo fa in modo che il web server invii al browser dell'utente esattamente *auth_key.php*, quindi uno specifico documento contenuto nella stessa directory (lato server) di *index.php*. Successivamente la sintassi "?key" descrive un effettivo passaggio di parametri al documento richiesto, un passaggio di parametri che opera esattamente come il metodo "*_GET*" utilizzato solitamente nei form di invio dati. Esso infatti costruisce un URL di questo tipo al momento dell'esecuzione delle istruzioni di invio.
"key" definisce il nome dell'attributo nell'array di sistema relativo al metodo "*_GET*", e permette quindi di essere estratto da esso attraverso la sintassi "*$_GET['key']*", già vista in precedenza ma riguardante il contrapposto metodo "*_POST*".
Il codice di 40 caratteri alfanumerici finale preceduto da "=", è il valore dell'attributo "key".

L'URL in esame quindi, effettua una richiesta di esecuzione di *auth_key.php* al web server, fornendo come parametro una variabile "key" al quale viene attribuito il valore stringa espresso nella sintassi.

Qui di seguito il codice del documento *auth_key.php*. Il suo scopo principale è, come abbiamo visto, quello di attivare gli account creati precedentemente attraverso *newuser.php*, ma questa è solo la prima delle sue funzionalità. Da essa ne derivano altre, le quali non sono altro che la conseguenza del completamento della procedura di registrazione utente.
Infatti oltre alla creazione dei record relativi nelle tabelle *user*s e *teams*, è necessario, solo una volta portata a termine l'attivazione, inizializzare ogni singolo aspetto del gioco, come i giocatori, le tattiche di base e molte altre che analizzeremo in seguito.

Il codice é composto dalle usuali procedure iniziali, la cattura del parametro trasmesso con la richiesta e successivamente la procedura di connessione al database SQL.
La chiave di 40 caratteri alfanumerici viene misurata ed utilizzata per inizializzare la variabile "*$kl*" che indica la lunghezza della stringa.
La successiva istruzione if, verifica che l'attributo catturato sia effettivamente composto da 40 caratteri, se il controllo dà esito positivo, una query estrae dalla tabella *users* solo il record il cui campo "*key_value*" corrisponde al parametro "*$_GET['key']*" e lo rende elaborabile attraverso l'assegnazione alla variabile array "*$ik*".
Il confronto successivo determina la validità della stringa e quindi, se la validazione ha successo, imposta tramite una query di "*UPDATE*" i valori "*ver*" = 1 e "*key_contro*l" = 0, identificando il record tramite l'"*id*" ottenuto tramite la precedente operazione di estrazione del dato memorizzato in "*$ik['id']*".

La procedura azzera il campo "key_control" e lo marchia come verificato impostando "ver" = 1

----- auth_key.php (attivazione) -----

```
<...>

<?php

<...>

$kl=strlen($_GET["key"]);

<...>       //connessione al database

If ($kl == 40){

$dk = mysql_query("SELECT * FROM users WHERE key_control='".$_GET['key']."'")or
die(mysql_error());

      $ik = mysql_fetch_array($dk);

      If($ik['key_control'] == $_GET['key']){

$up = "UPDATE users SET ver='1',key_control='0' WHERE id='".$ik['id']."'";
mysql_query($up,$dbh);
```

----- end -----

L'attivazione sembra ora completata, ma in realtà successivamente, all'interno dello stesso spazio di istruzioni creato dal passaggio di entrambi i controlli effettuati dalle istruzioni if, vengono eseguite un gran numero di operazioni conseguenti. Queste operazioni inizializzano il sistema in modo da prepararlo adeguatamente ad accogliere i nuovi utenti iscritti.

La prima delle operazioni è costituita dalla generazione dei 12 giocatori di base assegnati all'account dell'utente.
E opportuno chiarire come i giocatori generati vengono memorizzati nei record del database.
Ogni giocatore ha delle caratteristiche, oltre al nome esso avrà un "club" di appartenenza e delle caratteristiche, ossia delle statistiche relative alle sue capacità. Al fine di comprendere al meglio il tipo di dato che stiamo per affrontare analizziamo un record standard relativo ad un giocatore chiaramente univoco.

ID	name	cou	special	user_id	Resi	Vel	Pass	Tec	Rif
16	…	…	…	…	…	…	…	…	…
17	Felicetto Quarto	1	0	2	0	0	5	1	5
18	…	…	…	..	…	…	…	…	…

Reg	Dif	Att	_Tal	DT	Exp	Form	Anni	LR
…	…	…	…	…	…	…	…	…
4	7	4	1	0	2	5	15	D
…	…	…	…	…	…	…	…	…

Queste due tabelle costituiscono insieme un unico record relativo ai giocatori e alla loro memorizzazione su database

La tabella *players* contiene ogni giocatore che verrà generato in tutto il gioco. E' immediato quindi che essa dovrà immagazzinare una mole di record sostanziosa. Ogni nuovo ingresso quindi dovrà essere contraddistinto in modo univoco, si rivela necessaria l'importanza del campo "*ID*", chiave primaria, autoincrementale di valore non NULL.
Il secondo campo "*name*" assume il valore del nome completo del giocatore. Esso viene stabilito secondo due diverse procedure che affronteremo tra poco.
"*cou*" descrive la provenienza del giocatore e si riferisce alla chiave primaria della tabella *country* vista in precedenza.
"*special*" ha a che fare con la generazione del giocatore e si riferisce a una di esse.
"*user_id*" è un riferimento all'utente proprietario del giocatore, in questo modo si specifica a quale squadra esso appartiene.
I seguenti valori partendo da "*Resi*" fino a "*Att*" descrivono la misura delle abilità base del giocatore che sono rispettivamente *Resistenza*, *Velocità*, *Passaggi*, *Tecnica*, *Riflessi*, *Regia*, *Difesa* e *Attacco*. Questi definiscono, tramite valori che vanno da 0 a 9, il grado di abilità dei giocatori nei relativi ambiti.
"*_Tal*" esprime il talento come parametro nascosto, il valore governa il grado di apprendimento durante l'allenamento. Non è incrementabile in nessun modo.
"*DT*" o Disciplina Tattica indica la capacità di un giocatore di posizionarsi correttamente secondo le tattiche prefissate all'interno del terreno di gioco, maggiore disciplina tattica diminuisce la possibilità che il giocatore sia fuori posizione. Cresce di pari passo all'esperienza.
"*Exp*" è l'esperienza ed aumenta a seconda dei minuti di gioco collezionati durante le partite.
"Form" descrive la forma fisica, un giudizio generale il cui miglioramento o peggioramento è totalmente casuale ma non repentino, se basso influisce il rendimento in campo.
"Anni" semplicemente l'età del giocatore, parametro che influisce allenamento oppure il semplice abbassamento del livello delle prestazioni del giocatore.

“LR” singolo carattere che definisce se il giocatore è mancino o destro. Le prestazioni del ruolo di ala in campo vengono influenzate da questo parametro.

Definito a grandi linee il significato dei campi del record possiamo procedere all’analisi della procedura di generazione dei 12 giocatori di base che avviene all’attivazione del proprio account. Questa porzione di codice infatti viene eseguita solo se l’attivazione ha avuto buon fine.

Precedentemente è stato accennato che per generare il nome di un giocatore è possibile utilizzare due diverse metodologie. La prima delle due è quella standard, essa crea la stringa sfruttando dei database ausiliari, quello dei nomi e quello dei cognomi. Per ogni stato di appartenenza quindi ne esisterà una coppia in modo da poter rendere il nome del giocatore creato attendibile a seconda della nazione di provenienza.
L’intero gioco è infatti progettato per poter ospitare campionati diversi a seconda della nazionalità dell’utente. La versione attuale prevede la possibilità di creare giocatori italiani e tedeschi a seconda dei dati inseriti dall’utente in fase di registrazione.
La seconda metodologia invece fornisce la possibilità di avere nella propria squadra un giocatore “famoso” ossia realmente esistente. Tali giocatori non hanno una distinzione di caratteristiche fisiche che li rende diversi da quelli generati in modo normale.
La struttura del loro nome composto da tre parti, *Nome (Soprannome) Cognome*, li rende facilmente individuabili. Questa particolarità può avvenire con una probabilità del 3% ad ogni generazione.

Il seguente codice descrive la prima parte di questa fase di generazione. Il codice attraverso la query precedentemente eseguita, ricava la nazionalità dell’utente che ha appena attivato l’account utilizzando `$ik['country'];`.
In modo semplice poi identifica quali database deve utilizzare e semplicemente, se il valore catturato poco fa è 1, significa che l’utente è italiano quindi verranno utilizzati i database *nameITA* per i nomi e *surnameITA* per i cognomi. Nel caso sia 2, verranno utilizzati quelli tedeschi.

Questi database ausiliari sono semplici, ogni record contiene una chiave primaria ed il campo che identifica la stringa nome o cognome. La caratteristica è che ognuno di essi contiene circa 10000 record ed i nomi non hanno distinzione di sesso, essendo il tchoukball uno sport misto.

Successivamente attraverso l’istruzione `$dbspec = "specials";` viene inizializzato il nome della tabella appunto *specials* relativo ai giocatori realmente esistenti.

Esso è costruito in modo interessante. “*ID*” rappresenta la consueta chiave primaria. “*name*” relativo al giocatore realmente iscritto ai campionati. “*cou*” indica la nazionalità del giocatore, ha poco senso infatti che questa caratteristica non sia decritta, il giocatore rischierebbe di non essere effettivamente conosciuto.
Gli ultimi due campi permettono l’esclusività del giocatore. Esso infatti non può essere in possesso di più utenti. Un giocatore inutilizzato ha valore “*used*” impostato a 0, come per il campo “*user_id*” non esistendo utenti che abbiano “*id*” di valore 0.

Una volta che un record di questa tabella viene prelevato per essere incluso tra le fila di una squadra, esso rimarrà esclusivo, quindi “*used*” impostato a 1 e “*user_id*” va ad indicare a quale utente esso è stato assegnato.
Descriveremo in seguito come, attraverso le operazioni temporizzate esso può ritornare disponibile, ma può accadere solo tramite vendita, licenziamento o ritiro.

ID	name	cou	used	user_id
…	…	…	…	…
5	Nome (Soprannome) Cognome	1	0	0
6	Manuele (Manu) Puppo	1	1	3
…	…	…	…	…

Esempio di giocatori realmente esistenti memorizzati nella tabella specials del database

Il codice poi prosegue eseguendo le tre query alle tabelle determinate precedentemente e memorizzando i risultati in "*$dname*", "*$dsname*" e "*$dpsp*", i quali vengono poi analizzati per estrarre il numero di record estratti. In questo modo si hanno a disposizione i dati ed il relativo conteggio riguardo le tabelle di nomi, cognomi e giocatori speciali.
Infine vengono inizializzate la variabile "*$isSpe*" che controlla l'utilizzo eventuale dei record di *specials*, e la funzione "*ckmax*" di controllo relativo al massimo valore raggiunto per un dato parametro. L'esatto scopo della funzione verrà descritto in seguito.

----- auth_key.php (giocatori) -----

```
$ucou = $ik['country'];      // country dell'utente appena validato

If($ucou == 1){   //selezione database nomi e cognomi italiani
      $dbname = "nameITA";
      $dbsurname = "surnameITA";
} else {
      If($ucou == 2){   //selezione database nomi e cognomi tedeschi
            $dbname = "nameGER";
            $dbsurname = "surnameGER";
      }
}

$dbspec = "specials";   //database nomi speciali

$dn = "SELECT * FROM ".$dbname;     //query al db indicato dei nomi
$dname = mysql_query($dn,$dbh);     //esegui query

$ds = "SELECT * FROM ".$dbsurname;  //query al db indicato dei cognomi
$dsname = mysql_query($ds,$dbh);    //esegui query

$dsp = "SELECT * FROM ".$dbspec;    //query al db specials
$dpsp = mysql_query($dsp,$dbh);     //esegui query

$nname = mysql_num_rows($dname);    //conta numero nomi
$nsname = mysql_num_rows($dsname);  //conta numero cognomi
$nspecials = mysql_num_rows($dpsp);//conta numero nomi spacials

$isSpe = 0; // 1 se è stato selezionato un giocatore da database "special"

//Funzione di controllo parametro al massimo
function ckmax($ckv){
      If($ckv < 9){
            $ckr = 1;
      } else {
            $ckr = 0;
      }
      return $ckr;
}
```

----- end -----

La successiva parte di codice non è altro che un ciclo, che effettua le stesse operazioni 12 volte, tante quante sono i giocatori. L'intera procedura è alquanto complessa e, dato che molte delle istruzioni eseguite sono già state analizzate adeguatamente nei precedenti passaggi, l'analisi si fermerà alle logiche di creazione giocatori, ossia, dividendo la successione di istruzioni in diverse fasi, la descrizione delle funzioni si fermerà sull'intento di esse senza spiegare per estero il significato di variabili e sintassi.

Viene quindi generato un numero casuale tra 1 e 100, il risultato inizializza la variabile "*$spck*", la condizione successiva verifica la possibilità di utilizzare un giocatore prelevato dal database *specials*. La condizione `$spck > 3` infatti stabilisce che se il numero estratto è maggiore di 3, quindi rientrante nel range 4 – 100, nome e cognome verranno estratti dalle due tabelle specificate precedentemente tramite un "*ID*" generato casualmente (range 1 – numero record) relativamente agli ID della tabella selezionata. Nel caso in cui invece il valore di "*$spck*" sia compreso nel range 1 – 3, quindi al 3%, i nomi saranno estratti direttamente dalla tabella dei nomi realmente esistenti, sempre in modo casuale.

Questa seconda casistica di estrazione include un ulteriore controllo. Il campo "*name*" del record prelevato dalla tabella *specials*, non può essere ripetuto nel gioco. Allo scopo di valutare questa eventualità viene svolto un controllo sul campo "*used*" relativo al record in elaborazione, se esso è impostato a 0, il nome selezionato viene prelevato e l'esecuzione viene liberata dal ciclo while impostando "*$ckus* = 0". Se invece "*used*" è impostato a 1, viene effettuata una nuova ricerca casuale fino alla selezione di un dato adeguato.
I restanti campi, relativi a quello utilizzato, sono aggiornati al termine del ciclo for.

----- *auth_key.php (ciclo - nomi)* -----

```
for ($y = 0; $y < 12; ++$y) { //Ciclo creazione 12 giocatori
      $ckus = 1; // verifica che l'eventuale special estratto sia libero
      $spck = rand (1, 100); // al 3% scarica un giocatore special
      If ($spck > 3){
            $qnam = rand(1, $nname);
            $qsur = rand(1, $nsname);
            $fn = "SELECT * FROM ".$dbname." WHERE ID='".$qnam."'";
            $fs = "SELECT * FROM ".$dbsurname." WHERE ID='".$qsur."'";
            $nk = mysql_query($fn,$dbh);
            $ng = mysql_fetch_array($nk);
            $sk = mysql_query($fs,$dbh);
            $sg = mysql_fetch_array($sk);
            $pname = $ng['Name']." ".$sg['Surname'];
            $isSpe = 0;
      } else {
            while ($ckus == 1){
                  $qspec = rand(1, $nspecials);
                  $fn = "SELECT * FROM specials WHERE ID='".$qspec."'";
                  $nk = mysql_query($fn,$dbh);
                  $ng = mysql_fetch_array($nk);

                  If ($ng['used'] == 0){
                        $pname = $ng['name'];
                        $isSpe = 1;
                        $ckus = 0;  //libera l'esecuzione dal ciclo
                  }     //end if

            }     //end while
      }
```

----- *end* -----

Attraverso la seguente porzione di codice vengono stabiliti i valori per quanto riguarda le statistiche di ogni giocatore. La primissima parte di istruzioni inizializza esattamente ogni variabile descritta precedentemente che conterrà un valore intero. Numero che esprime da 0 a 9 un giudizio delle capacità di ogni singolo carattere. La regola generale vuole che tutte le caratteristiche inizializzate a 0 abbiano a disposizione 25 incrementi che verranno utilizzati in modo casuale, ma controllato dalla *function ckmax()*. Essa verifica il numero di incrementi già avvenuti e blocca l'esecuzione se questo è minore o uguale a 9 (il massimo), in questo caso l'incremento non va perso ma riassegnato.

Descriviamo quindi la routine contenuta nel ciclo for. Ciclo che verrà eseguito 25 volte, tante quante sono gli incrementi a disposizione.
Ad ogni esecuzione corrisponde un altro ciclo while interno. Esso permette di non disperdere l'incremento eventualmente non assegnato. Viene quindi generato un numero casualmente nel range 1 – 8, ossia il numero di caratteristiche da elaborare. Esso, inizializza la variabile "*$sel*" che identifica uno degli 8 aspetti. Una serie di istruzioni if poi permette di utilizzare l'incremento sulla giusta variabile per la quale viene innanzitutto svolto il controllo attraverso la *function ckmax()* analizzata precedentemente. Se il controllo, verificato attraverso la variabile "*$ck*" da esito positivo, l'aspetto selezionato viene incrementato ed è possibile uscire dal ciclo while per procedere con una nuova assegnazione. Se invece il controllo fallisce, perché l'attributo ha già raggiunto il valore 9, si procede con un successivo ciclo while per eleggere una nuova statistica da incrementare.

----- auth_key.php (ciclo - statistiche) -----

```
//Dichiarazione valori

$nresi = 0; //Resistenza
$ntec = 0;  //Tecnica
$nvel = 0;  //Velocità
$natt = 0;  //Attacco
$ndif = 0;  //Difesa
$nreg = 0;  //Regia
$npass = 0; //Passaggi
$nrifl = 0; //Riflessi

$nanni = 0; //Anni
$ndt = 0;   //Disciplina Tattica sempre a 0 alla creazione
$nexp = 0;  //Esperienza 2 - 5 alla creazione
$nform = 0; //Forma
$ntal = 0;  //Talento
$nhand = "";      //Mancino o destro (M - D)

for ($z = 0; $z <= 25; ++$z) {

      $ver = 1;
      while ($ver == 1){
            $sel = rand(1, 8); //seleziona uno dei valori

            If($sel == 1){                //resistenza
                  $ck = ckmax($nresi);
                  If($ck == 1){
                        $nresi = $nresi + 1;
                        $ver = 0;
                  }
            }

            <...> //tecnica

            <...> //velocità

            <...> //attacco
```

```
            <...> //difesa

            <...> //regia

            <...> //passaggi

            If($sel == 8){                //riflessi
                  $ck = ckmax($nrifl);
                  If($ck == 1){
                        $nrifl = $nrifl + 1;
                        $ver = 0;
                  }
            }
      }     //end while
}     //end for
```

----- *end* -----

La parte successiva determina età, esperienza, forma e talento.

Il valore assegnato alla variabile età può andare da un minimo di 14 anni ad un massimo di 26. L'elaborazione di questo dato si articola in fasce ed incrementi. Innanzitutto viene stabilita un'età di base, un valore casuale tra 14 e 18. Successivamente viene estratto un numero casuale tra 1 e 100 che inizializza la variabile "*$anpc*". All'età di base stabilita ne conseguono degli eventuali incrementi. Per semplicità lo schema successivo descrive la meccanica della routine:

$nanni = età di base = casuale (14 - 18);
$anpc = casuale (1 – 100);
11% -> ETA' = *$nanni* + casuale(6 - 8);
20% -> ETA' = *$nanni* + casuale(4 - 6);
15% -> ETA' = *$nanni* + casuale(1 - 3);
54% -> ETA' = *$nanni*;

In questo modo è meno probabile che ci siano dei giocatori di età massima, molto più probabile invece che ci siano giocatori nella fascia minima.

L'esperienza inizialmente è impostata casualmente nel range 3 – 5, essendo giocatori che non hanno mai partecipato ad un campionato questi valori sono verosimili.

Il parametro forma è totalmente casuale da 0 a 9, esso muta in maniera casuale di settimana in settimana attraverso le operazioni temporizzate settimanali.

La routine che genera il talento è più articolata, come nella realtà questa caratteristica è importante ma non essenziale. Può variare in un range da 1 a 3, ma ottenere un giocatore di massimo talento è possibile solo al 20%. Al 40% è possibile ottenere un giocatore di talento = 2. Nel 40% di possibilità rimanente si ottiene un giocatore di talento base, quindi di valore 1.

----- auth_key.php (ciclo – anni/exp) -----

```
//Determinazione Anni (in questo modo è meno probabile che capitino al massimo
26enni in squadra)
$nanni = rand (14, 18);
$anpc = rand (1, 100);
      If ($anpc >= 90){
            $anpl = rand (6, 8);
            $nanni = $nanni + $anpl;
      } else {
            If ($anpc >= 70){
                  $anpl = rand (4, 6);
                  $nanni = $nanni + $anpl;
            } else {
                  If ($anpc >= 55){
                  $anpl = rand (1, 3);
                  $nanni = $nanni + $anpl;
            }
      }
}

//Esperienza casuale 3 - 5
$nexp = rand (3, 5);

//Forma casuale 0 - 9 e ogni settimana incremento o decremento casuale
$nform = rand (0, 9);

//Determinazione Talento, parametro nascosto (più talento = più raro)
$talpc = rand (0, 9);
If ($talpc >= 8){
      $ntal = 3;
} else {
      If ($talpc >= 4){
            $ntal = 2;
      } else {
            $ntal = 1;
      }
}
```

----- end -----

Prima dell'inserimento dei valori elaborati nell'opportuna tabella del database, come ultimo valore viene stabilito se il giocatore è mancino o destro.
In accordo con le statistiche i mancini sono l'8,5% della popolazione mondiale, ma avendo a che fare con numeri interi il valore viene arrotondato per eccesso al 9%. Come già descritto in precedenza quindi viene generato un numero casuale da 1 a 100, se esso è minore uguale a 9, il giocatore sarà mancino, in caso contrario destro.

Segue la query di inserimento finale che crea all'interno della tabella *players* un nuovo record che descrive sotto ogni aspetto il giocatore creato. La meccanica di questa query è del tutto identica a quelle descritte in precedenza, nella forma quindi "*INSERT INTO players (…) VALUES (…)*".

La fase successiva aggiorna i campi “*used*” e “*user_id*” dell’eventuale giocatore selezionato dalla tabella *specials*. “*user*” viene impostato a 1, “*user_id*” invece, come è facilmente intuibile, viene aggiornato inserendo il valore “id” dell’utente relativamente al quale è stata appena eseguita la procedura di attivazione account.

----- auth_key.php (ciclo – hand/inserimento) -----

```
//Mancino o Destro? 8,5% secondo le statistiche sono i mancini nel mondo,
arrotondando a 9%
$handpc = rand (1, 100);
If ($handpc <= 9){
      $nhand = “M”;
} else {
      $nhand = “D”;
}

$nplq = “INSERT INTO players (ID, name, cou, special, user_id, Resi, Vel, Pass,
Tec, Rif, Reg, Dif, Att, _Tal, DT, Exp, Form, Anni, LR) VALUES (‘’,’”.$pname.”’,
‘”.$ucou.”’, “.$isSpe.”, “.$ik[‘id’].”, “.$nresi.”, “.$nvel.”, “.$npass.”,
“.$ntec.”, “.$nrifl.”, “.$nreg.”, “.$ndif.”, “.$natt.”, “.$ntal.”, “.$ndt.”,
“.$nexp.”, “.$nform.”, “.$nanni.”, ‘”.$nhand.”’)”;
 mysql_query($nplq,$dbh) or die(mysql_error());

If ($isSpe == 1){
      $speUp = “UPDATE specials SET used=’1’,user_id=’”.$ik[‘id’].”’ WHERE
ID=’”.$qspec.”’”; //aggiorna campi ‘used’ e ‘user_id’
      mysql_query($speUp,$dbh);
}

} //fine for per i 12 giocatori
```

----- end -----

Concluso il ciclo di generazione giocatori, è necessario impostare le tattiche di base relative all’utente. Esse sono codificate nella forma di stringhe composte da 132 caratteri esclusivamente numerici, sono relative ad ogni giocatore in campo, quindi per 7 giocatori, e si differenziano per attacco e difesa. Il significato della codifica verrà analizzato nei successivi capitoli.

Le tabelle del database coinvolte in questa procedura sono *tacts* e *tacts_d*. Le due tabelle sono identiche, ma a causa delle caratteristiche del gioco, suddiviso nettamente in fasi di attacco e difesa, è necessario creare diverse strutture dati per una migliore gestione.
Oltre al necessario campo “*ID*”, ogni record è contraddistinto da un campo “*name_t*” che descrive il nome associato alla tattica, il campo “*user*” relativo all’id utente proprietario, la codifica distinta per ogni giocatore dai campi “*P1*” a “*P7*”, ed il campo “*las*t” che a specifica se impostato a 1, la tattica selezionata. Solo una tattica tra quelle di uno stesso utente può avere campo “*last*” impostato a 1, per tutte le altre, è impostato a 0.

ID	name_t	user	P1	P2	P3	P4	P5	P6	P7	last
3	…	…	…	…	…	…	…	…	…	…
4	7vs0	2	0402…	1211…	0672…	0971…	0341…	0760…	0980…	1
5	…	…	…	…	…	…	…	…	…	…

Struttura delle tabelle tacts e tacts_d.

I valori codificati sono il risultato di una necessità grafica e derivano da un'impostazione standardizzata della tattiche di gioco base che ogni giocatore conosce. E' chiaramente possibile modificarle a proprio piacimento, nei capitoli successivi verranno analizzate accuratamente.

----- auth_key.php (tattiche) -----

```
            //creazione tattica iniziale base del 7vs0 attacco
            $tacq = "INSERT INTO tacts (ID, name_t, user, P1, P2, P3, P4, P5,
P6, P7, last) VALUES ('','7vs0', '".$ik['id']."',
'039030041032040030040032042031044032038036043031040037040036040034042051042049043055043059043063043054041061038065038060042066042066',
'048058168054046085110078180079116078118081118078101085120081123080114127114128116127114127115126116126114133114143114138111158113154',
'177030180031180031182032180032180032180033181032179035179034178033185051183051183005818305818205618205918306618506118806518706483053180060183058182056182059183066185061188065187064',
'119166117166115166114151114163041128117133183133047184112185180183043244108244181244041296114294180299097282114267107281117281113280',
'045367043367045367044369045369043369035371044369039371035371039369039389042389040390038390038391039391038395041387038389042392044391',
'117286115283119284118284118285118285113284118285117280112285114285103322108322129322097349116344135334043349111348181348044367180372',
'186366185368184367182367183367185369189376184367187375188376188369187393187395185394190395188395182394188395186394185396184393186394', 1)";
            mysql_query($tacq,$dbh) or die(mysql_error());

            //creazione tattica iniziale base del 7vs0 difesa
            $tacdq = "INSERT INTO tacts_d (ID, name_t, user, P1, P2, P3, P4, P5,
P6, P7, last) VALUES ('','7vs0', '".$ik['id']."',
'122051081034120056084049077036104078073075060052048081048080051081045117039103032112064126030130030148111181059159030169118217024224',
'136045091045143058115058090060145075113078086073115108116104114108127149115126103144136183119180073183115248113211104252138336085332',
'147035104052153036144048113058171051156078137063176080178079175080197121187099174105197139194128126125200171179158115183205221113217',
'164071061067166092116115068094163143115152061142118185114184108183118245114241104244135260118264094259152311116284076311159360061360',
'130206024200026235074265117234073292050290027296041316034327038319053346041351039351067342044353042357102359070374067381120377081395',
'145084083082133119116164093113119207117213114207115254114264114254119308114302108304139316120323094320137358127346090361136382093384',
'204196100206117236197263200234200293193293133287187317189321189318188357183345176343186358190356149339161380158370126359150393106378', 1)";
            mysql_query($tacdq,$dbh) or die(mysql_error());
```

----- end -----

Al termine di tutta la procedura di inizializzazione viene generata la visualizzazione di un messaggio HTML di conferma attivazione avvenuta. Un collegamento a *index.php* porta quindi l'utente alla home invitandolo a procedere effettuando il login ed a interagire con la piattaforma.

Nell'eventualità che qualche aspetto nella procedura non sia stato eseguito in maniera corretta, il sistema avverte l'utente nel primo caso che il valore assegnato al parametro "*key*" passato ad *auth_key.php* non è valido, mentre nel secondo caso che il valore stesso non è corretto.
Le cause sono rispettivamente che la stringa assegnata a "*key*" non è effettivamente composta da 40 caratteri, e che la stringa rilevata non coincide con nessuna altra stringa di attivazione presente tra i record della tabella *user* del database.

Queste problematiche possono essere errori di trasmissione, o semplici errori da parte dell'utente che non ha affrontato la procedura descritta come segnalato, ma possono anche essere tentativi di intrusione. Sono quindi necessari tutti i tipi di validazione possibile sui dati sotto il controllo dell'utilizzatore che potrebbe avere nel peggiore dei casi intenzioni malevole nei confronti della piattaforma.

----- auth_key.php (visualizzazione/errori) -----

```
            Print "Dear ".$ik['name']." (".$ik['username']."), your account has
been activated, proceed to your first login<br><br>";
            Print "Click here -> <a href='index.php'>Login</a>";
            mysql_close($dbh);
      }else{
            Print "Key value isn't valid, please check your activation email on
your mailbox or repeat your registration";
      }
} else {
      Print "Key value isn't correct, please check your activation email on your
mailbox";
}

?>
```

----- end -----

I documenti appena analizzati sono corretti e funzionanti, necessiteranno però di modifiche che aumenteranno di pari passo al completamento dello sviluppo del progetto. L'aggiunta di nuove funzionalità infatti obbliga lo sviluppatore ad includere strutture o a completare quelle già esistenti, correggendo quindi il codice creato fino ad ora al fine di inizializzare nuovi dati all'attivazione dell'account di gioco.

3.2 - Lista giocatori:

Una volta affrontate le procedure di registrazione nuovo utente e successivamente di attivazione account, l'utente possiede le credenziali ed è in grado di effettuare l'autenticazione e quindi di poter utilizzare la piattaforma.
Come già visto nell'analisi di *login.php*, una volta che le credenziali sono state verificate, il sistema crea una sessione. Questa deve essere mantenuta durante tutta la navigazione a meno che essa non scada o l'utente non voglia distruggerla attraverso le istruzioni definite in *logout.php*.

La scadenza della sessione è definita dal sistema attraverso le istruzioni contenute nel file *php.ini*. *php.ini* è il file di configurazione relativo al linguaggio, letto generalmente all'avvio dall'interprete. Il valore di scadenza della sessione è impostato inizialmente a 1440 secondi (24 minuti) di inattività da parte dell'utente e può essere semplicemente modificato agendo su *php.ini* stesso.

L'attivazione della sessione permette l'accesso al menù di navigazione che porta ad ogni pagina web del gioco. Ogni documento quindi che descrive i documenti web visualizzati deve essere provvisto di una particolare porzione di codice.

La seguente struttura, come già accennato durante l'analisi della procedura di autenticazione, deve essere contenuta da *index.php* e da tutti gli altri documenti successivamente visualizzabili.
Esso permette di mantenere la sessione e quindi ogni altra istruzione deve assolutamente essere inserita all'interno del primo if, nel caso in cui la sessione per qualsiasi motivo scompaia, verrà visualizzato solo l'invito ad effettuare nuovamente il login presente dopo l'istruzione else contrapposta alla condizione specificata da if.

```
                          ----- index.php -----
<?php

session_start();

if(isset($_SESSION['auth'])){
                           //la sessione esiste
} else {
                           //la sessione non esiste, procedi al
login/registrazione
}

?>
                             ----- end -----
```

Ad accesso avvenuto *index.php*, riconoscendo la sessione attiva, presenta una panoramica sui dati utente, campionato e presenta sempre attivo sulla sinistra un menù di navigazione. Esso permette di navigare tra i documenti web del browsergame.

Semplice visualizzazione del menù della piattaforma in esame
La sua visualizzazione è possibile solo a login effettuato

Seguendo l'elenco definito dal menù, che inizia con "Home" rappresentato da *index.php*, ci si imbatte in "Lista Giocatori". Questo documento permette la visualizzazione dei giocatori associati all'account dell'utente relativo alla sessione creata.
Per quanto analizzato fino ad ora la lista dei giocatori permette di visualizzare la lista di nomi della dozzina di giocatori di base generati attraverso la routine di attivazione.
Il codice, benché sia stato interamente progettato, è ancora in fase di sviluppo. Il suo scopo finale è quello di creare una struttura ordinata che permetta di visualizzare in modo completo ogni giocatore evidenziando oltre che al nome, ogni altro parametro specificato nel singolo record associato al giocatore presente nella tabella *players* del database.

La seguente porzione di codice si riferisce alla tecnologia CSS già descritta.
Viene applicata impostando tra i tag `<head>` del documento un riferimento al relativo foglio di stile *style.css*. Grazie ad esso l'interprete del linguaggio include nel corrente documento le impostazioni grafiche definite in *style.css* ed applica ad ogni elemento un differente stile a seconda dell'"id" dell'elemento.

----- *playerlist.php* (css) -----

```
<head>
<title>Lista Giocatori</title>

<link rel="stylesheet" type="tect/css" href="style.css">

</head>
```

----- *end* -----

Prendiamo in esame alcuni degli stili definiti dal documento. Essi si occupano esclusivamente di tutto ciò che ha a che fare con la visualizzazione grafica degli oggetti e mai dei contenuti. Tali oggetti possono essere delle stringhe oppure dei semplici spazi che contribuiscono a formare il design.

La seguente sintassi indica un pratico esempio di documento css. La prima istruzione definisce il charset, come già visto relativamente agli header html ed alle query SQL, il charset scelto è *utf-8*.

Dal primo paragrafo *body* che si limita a descrivere uno stile generico che vale per ogni parte del documento, se non ne viene specificata una diversa, passiamo a *#left*.
Esso definisce in modo semplice il riquadro allineato a sinistra, rispettivamente viene specificato il bordo nero e largo 1 pixel, la larghezza di 150 pixel, l'altezza di 800 pixel ed infine lo sfondo che viene prelevato da file, ossia un file immagine *bg1.jpg*.
Sono presenti ulteriori simili definizioni ma le più interessanti sono *#menu* e *.buttons*.

#menu agisce tramite il riferimento id nella pagina HTML, specifica solo un bordo simile a quello di *#left*. Gli elementi che agiscono tramite id possono essere utilizzati una sola volta per documento.

.buttons invece applica le sue proprietà tramite il riferimento classe, essi possono essere richiamati su uno o su un sottoinsieme di elementi con caratteristiche e funzionalità comuni in una pagina HTML.

----- *style.css* -----

```
@charset "utf-8";
/* CSS Document */

body{
	font: 13px/2em Arial, Helvetica, sans-serif;
}


<...>


#left{
	float:left;
	border:1px solid black;
	background:;
	width:150px;
	height:800px;
	background:url(img/bg1.jpg) no-repeat center center;
}

<...>

#menu{
	border:1px solid black;
}

.buttons {
	border:1px solid black;
	background:white;
}
```

----- *end* -----

Notiamo che tra le istruzioni del documento è presente la procedura per il riconoscimento della sessione descritto attraverso *index.php*. All'interno della condizione di riconoscimento della sessione è presente inoltre il collegamento a *logout.php*.

E' immediatamente definita la visualizzazione del menù a sinistra presente ad ogni fase della navigazione composto graficamente dalle impostazioni definite in *style.css*. Vediamo quindi le definizioni per l'id di *#menu* e per le classi *.buttons* applicate ad ogni voce del menù di navigazione.

Successivamente come di consueto il codice avvia la procedura di connessione al database ed una serie di query atte a determinare l'id utente e, tramite esso, la selezione dei giocatori tra i record della tabella *players* che abbiano attributo "*user_id*" uguale a quello appena ricavato.

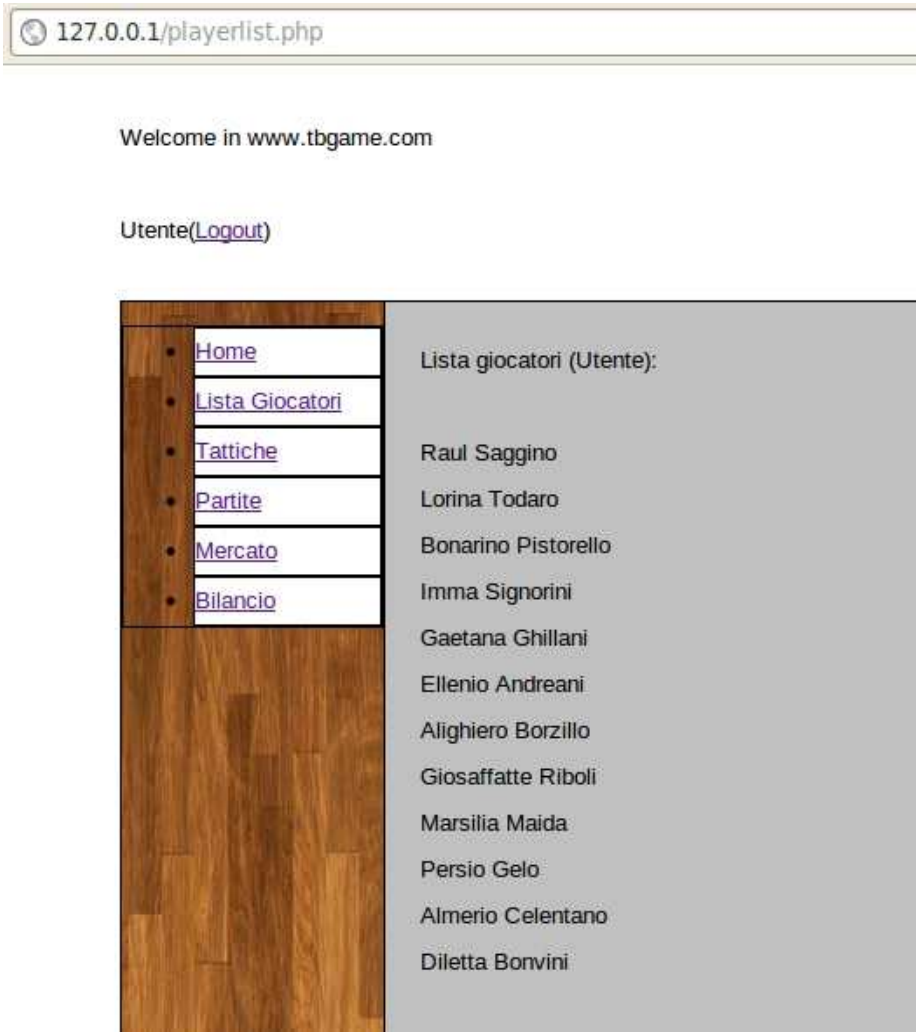
Infine un semplice ciclo while, scorre ogni risultato ottenuto e lo visualizza a schermo sottoforma di elenco. `Print $ipl['name']."<br>";` infatti permette di andare a capo ad ogni esecuzione formando un semplice elenco di nomi grazie all'istruzione HTML `<br>`.

----- *playerlist.php* -----

```
session_start();        //permette l'utilizzo dell'array di sessione

if(isset($_SESSION['auth'])){

?>
      <div> Welcome in www.tbgame.com
      <?php Print "<br><br>" .$_SESSION['user_id']. "(<a
href='logout.php'>Logout</a>)<br><br>";
      ?>

      <div id="left">
      <ul id="menu"> <!-- MENU -->
      <li class="buttons"><a href='index.php'> Home </a></li>
      <li class="buttons"><a href='playerlist.php'> Lista Giocatori </a></li>
      <li class="buttons"><a href='tacts.php'> Tattiche </a></li>
      <li class="buttons"><a href='matchlist.php'> Partite </a></li>
      <li class="buttons"><a href='mercato.php'> Mercato </a></li>
      <li class="buttons"><a href='bilancio.php'> Bilancio </a></li>
      </ul>

      <?php

      <...>       //connessione al database

      Print "Lista giocatori (".$_SESSION['user_id']."):<br><br>";

      $usid = mysql_query("SELECT * FROM users WHERE
username='".$_SESSION['user_id']."'")or die(mysql_error());
      $iui = mysql_fetch_array($usid);
      $plist = mysql_query("SELECT * FROM players WHERE
user_id='".$iui['id']."'")or die(mysql_error());

      while ($ipl = mysql_fetch_array($plist)){
            Print $ipl['name']."<br>";
      }

} else {
      Print "Session have been lost, please <a href='index.php'>Login</a>
again";
}

?>
```

----- *end* -----

Di seguito la visualizzazione finale della pagina *playerlist.php*. Il menù a sinistra è sempre presente ed è richiamato l'username dell'utente che, grazie al collegamento può avviare la procedura di logout in qualsiasi momento.



playerlist.php che permette la visualizzazione dei giocatori in possesso dell'utente corrente accessibile attraverso il collegamento "Lista Giocatori".

Come già accennato, questo non è il risultato definitivo. Il progetto prevede che al posto di ogni nome, sia sviluppato un riquadro contenente tutte le informazioni salvate nella tabella *players*. Queste informazioni contengono anche i valori delle statistiche relative alle abilità di ogni singolo giocatore in formato numerico da 0 a 9.
Una visualizzazione di questo tipo però non sarebbe "*User Friendly*". Allo scopo di rendere il tutto più comprensibile quindi il progetto prevede lo sviluppo di una routine che converta ogni valore numerico in un giudizio (Esempio: Scarso, Buono, Ottimo).

Routine facilmente realizzabile attraverso una serie di condizioni if che impostino un output standard in formato stringa (Sufficiente, Medio, Incredibile), associato al relativo valore.

E’ previsto inoltre che ogni giocatore abbia un valore in termini finanziari calcolato ovviamente in base alle relative capacità e caratteristiche, che possa determinare anche le spese settimanali che incidono sul bilancio del club. In questo modo non è sufficiente spendere molto per avere un forte giocatore tra le proprie fila, ma è anche opportuno che la squadra finanziariamente forte se lo possa permettere. Il livello quindi delle squadre tra i diversi campionati viene regolarizzato da un limite finanziario, disponibilità che aumenta a seconda delle partite vinte e delle eventuali promozioni a campionati di livello superiore.

La dozzina di giocatori quindi non è statica, il progetto infatti specifica la produzione di una struttura mercato che permetta di comprare e vendere giocatori. E’ possibile inoltre licenziarli nel momento in cui essi diventano inutili o invendibili. L’argomento sarà approfondito nei capitoli successivi.

3.3 - Tattiche:

Il paragrafo si occupa dell'analisi del documento *tacts.php*. Attraverso il suo codice è possibile creare, modificare o eliminare le tattiche di gioco, esse rappresentano sostanzialmente le posizioni che ogni giocatore deve tenere in campo.
Si dividono in fase di attacco e difesa. Il tchoukball infatti prevede strategie di gioco diverse durante le due fasi. Non essendoci intercettazione della palla e soprattutto una sola parte del campo dove poter segnare, devono essere affrontate in modo separato.

Il codice prevede l'utilizzo della tecnologia AJAX che verrà toccata successivamente, ma la sua funzione è identica a quella già descritta nel caso di *form.php*.

Successivamente alla procedura di riconoscimento della sessione viene eseguita una query che dai dati della stessa individua l'id dell'utente corrente dalla tabella *users*.

----- *tacts.php* -----

```
<html>
<head>

<title>Tattiche Attacco/Difesa</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="javascript" type="text/javascript">

<...>       //funzione getXMLHTTP();

<...>       //funzione showTacA();

<...>       //funzione showTacD();

</script>
</head>
<body>

<?php

<...>       //connessione al database

session_start();  //permette l'utilizzo dell'array di sessione

If ($_SESSION['auth'] == 1){

$usid = mysql_query("SELECT * FROM users WHERE
username='".$_SESSION['user_id']."'")or die(mysql_error());
$iui = mysql_fetch_array($usid);
```

----- *end* -----

Le funzioni che compongo il codice di *tacts.php*, prevedono che la pagina stessa possa automaticamente effettuare richieste composte a se stessa. Nel caso quindi questo avvenga essa prevede, come *index.php* un controllo iniziale sugli eventuali parametri passati.
Il documento infatti contiene delle listbox dinamiche che elencano tutte le tattiche di attacco e difesa divise in due diversi elementi di questo tipo.

Quando viene selezionata una tattica è possibile eliminarla o rinominarla. L'avvio di questa procedura porta ad un aggiornamento dell'intera pagina, oltre che di alcuni campi delle tabelle *tacts* o *tacts_d*. La selezione di un valore delle listbox invece tramite la tecnologia AJAX permette di aggiornare una sola porzione di codice HTML, ossia l'inclusione dell'oggetto che permette di visualizzare graficamente le tattiche ed un relativo form.

Il seguente codice rappresenta il controllo dei valori dei parametri eventualmente passati con metodo "*_POST*" durante l'aggiornamento.
Il controllo è necessario a causa della presenza di tattiche prestabilite di base create al momento dell'attivazione dell'account utente. Queste tattiche che hanno nome "7vs0" non possono essere eliminate o rinominate. Nel caso invece in cui altre nuove tattiche inserite debbano essere rinominate, il controllo ha effetto solo sulla validazione dei dati sui caratteri minimi e massimi.
L'istruzione `if (isset($_POST["tNameA"])` verifica che il parametro sia impostato e quindi utilizzabile. Se esso è uguale a "", quindi vuoto, viene generato un alert in javascript che spiega l'errore.

Alert relativo al form di inserimento: richiesta a campo vuoto

In caso contrario invece la verifica che ne consegue controlla il numero massimo di caratteri che non può superare i 15. Il messaggio eventuale d'errore si presenta nelle stesse modalità del precedente.

Se nessuno dei due errori si presenta, significa che il numero di caratteri rientra nel range 1 – 15 ma è necessario effettuare un terzo controllo. Nel caso in cui la tattica venga rinominata con la stringa "7vs0" il messaggio d'errore specifica l'impossibilità dell'azione. "7vs0" infatti indica una tattica "*read-only*" di sistema che non può essere in alcun modo modificata e quindi non possono esistere omonimi del dato.

Alert relativo al form di inserimento: richiesta non permessa

Infine, passati tutti i test, viene eseguita la query di update valori all'interno della tabella *tacts*. La sintassi sfrutta l'id della tattica d'attacco passato anch'esso via metodo *"_POST"* per individuare il record corretto e aggiornare il campo *"name_t"* utilizzando il valore del parametro in esame.

La stessa validazione viene successivamente eseguita relativamente al secondo form, quello relativo alle tattiche di difesa rappresentate nella tabella *tacts_d*.

----- *tacts.php* (rinomina)-----

```
if (isset($_POST["tNameA"])) {
      if ($_POST["tNameA"] == "") {
            echo "<script type='text/javascript'>alert('Errore: Inserisci almeno
1 carattere');</script>";
      } else {
            if (strlen($_POST["tNameA"]) > 15) {
                  echo "<script type='text/javascript'>alert('Errore: Inserisci
fino a 15 caratteri');</script>";
            } else {
                  if ($_POST["tNameA"] == "7vs0") {
                        echo "<script type='text/javascript'>alert('Errore:
Impossibile rinominare in \"7vs0\"');</script>";
                  } else {

                        $upRA = "UPDATE tacts SET name_t='".$_POST['tNameA']."'
WHERE ID='".$_POST['idAt']."'";
                        mysql_query($upRA,$dbh);
                  }
            }
      }
}

<...>       //controllo rinomina su tacts_d
```

----- *end* -----

Il seguente controllo, molto simile al precedente, opera sull'eliminazione del record relativo ad una tattica. Esso opera attraverso l'id della tattica da eliminare, passato via parametro. Una query che sfrutta `$_POST['TacAtt']` identifica il record per il quale è stata avanzata una richiesta di eliminazione, segnalata attraverso la listbox.
`if ($idA[name_t] == "7vs0")` verifica che il record selezionato non abbia come valore del campo *name_t* la stringa "7vs0". Come descritto in precedenza infatti questa tattica è unica e non modificabile o eliminabile. Se il controllo da esito positivo il solito alert avvisa l'utente che l'opzione di eliminare questa determinata tattica non è tollerata dal sistema.

Alert eliminazione: richiesta non permessa

Tra le tattiche di un utente deve sempre esisterne una selezionata, quindi il cui il campo "*last*" sia impostato a 1 (tutti gli altri devono essere impostati a 0). Viene quindi effettuato un controllo sul campo "*last*" del record da eliminare, se esso è impostato a 1, viene eletta come nuova selezionata la tattica di base "7vs0" tramite una query di UPDATE.
Segue l'effettiva eliminazione del record attraverso la query DELETE seguente:

```
"DELETE FROM tacts WHERE ID='".$_POST['TacAtt']."'"
```

Di semplice comprensione essa elimina un record determinabile attraverso il campo "*ID*" precedentemente analizzato.

----- *tacts.php* (elimina)-----

```
if (isset($_POST['TacAtt'])) {           //funzione di rimozione tattica di
attacco selezionata
      $ckDA = mysql_query("SELECT * FROM tacts WHERE
ID='".$_POST['TacAtt']."'")or die(mysql_error());
      $idA = mysql_fetch_array($ckDA);
      if ($idA[name_t] == "7vs0") {
            echo "<script type='text/javascript'>alert('Non puoi eliminare
7vs0');</script>";
      } else {
            if ($idA['last'] == 1) {
                  $upA = "UPDATE tacts SET last='1' WHERE name_t='7vs0' AND
user='".$iui['id']."'";
                  mysql_query($upA,$dbh);
            }
            $delA = mysql_query("DELETE FROM tacts WHERE
ID='".$_POST['TacAtt']."'")or die(mysql_error());
      }
}

<...>       //controllo elimina su tacts_d
```

----- *end* -----

Le tabelle *tacts* e *tacts_d* memorizzano i dati secondo una struttura di questo tipo già visto in precedenza:

ID	name_t	user	P1	P2	P3	P4	P5	P6	P7	last
3	…	…	…	…	…	…	…	…	…	…
4	7vs0	2	0402…	1211…	0672…	0971…	0341…	0760…	0980…	1
5	…	…	…	…	…	…	…	…	…	…

Struttura delle tabelle tacts e tacts_d.

Per poter proseguire nell'analisi ora è necessario analizzare le funzionalità della pagina prodotta dal documento *tacts.php*.
Oltre al menù sulla sinistra ed al collegamento alla procedura di logout presenti ad ogni fase della visualizzazione, essa è composta da due listbox che collezionano i nomi di tutti i record presenti nella tabella *tacts* (attacco) e *tacts_d* (difesa) relativi all'utente corrente. Il valore selezionato in uno di esse produce come risultato la visualizzazione dell'effettiva tattica nei riquadri sottostanti, la listbox relativa all'attacco condiziona l'oggetto di sinistra, mentre quella di difesa quello di destra.

Grazie ala tecnologia AJAX alla selezione di un valore corrisponde l'aggiornamento dinamico dell'oggetto relativo. Allo stesso modo sono aggiornati i form sottostanti che permettono di rinominare la tattica.
Gli oggetti sono il risultato dell'esportazione in formato *.swf* di animazioni creati in Actionscript 3.0 attraverso lo strumento Adobe Flash CS5.
Attraverso gli oggetti.swf, chiamati rispettivamente *tacts.swf* e *tacts_d.swf* è possibile visualizzare in modo dinamico le tattiche. Trascinando utilizzando il cursore del mouse l'oggetto palla rossa all'interno del riquadro i giocatori rappresentati dei quadrati gialli si muovono proporzionalmente al fine di prendere la posizione in campo basata sulla posizione della palla.
Le meccaniche ed il codice Actionscript relativi a questi file verrà analizzato successivamente.

Al fine di comprendere al meglio il codice di *tacts.php* è però necessario conoscere alcuni dettagli di questi oggetti. Attraverso essi, sfruttando i pulsanti NEW e MOD, è possibile generare nuove tattiche. Una volta stabilite le nuove impostazioni, l'oggetto le salva, effettuando una richiesta con parametri a *tacts.php* stesso, il quale a seconda del caso modificherà o creerà un nuovo record nella tabella *tacts* o *tacts_d*.

Visualizzazione riguadro di tacts.php

Possiamo ora comprendere al meglio la seguente parte di codice. I dati prelevati infatti in questo caso derivano proprio da una richiesta a *tacts.php* effettuata da uno degli oggetti.swf inclusi nel codice.
Attraverso `if (isset($_POST['nameTac']))` il documento comprende che la richiesta giunge da *tacts.swf* oppure da *tacts_d.swf*, colleziona quindi ogni dato proveniente dalla richiesta. Le variabili da "*$posPL1*" a "*$posPL7*" rappresentano la codifica delle posizioni dei giocatori espresse in stringhe numeriche da 132 caratteri di cui abbiamo parlato in precedenza ma che analizzeremo successivamente. "*$namet*" esprime il nome della tattica eventualmente memorizzato nel campo "*name_t*" dei record. "*$dbTact*" invece indica di quale tabella si tratta e quindi indirettamente da quale oggetto è stata inoltrata la richiesta.
Le successive due query impostano il campo "*last*" di tutte i record relativi all'utente corrente a 0 e prelevano ognuno di questi record inizializzando la variabile "*$rQu*". I record vengono poi contati inizializzando così la variabile "*$n*".

----- *tacts.php* (richieste oggetti 1)-----

```
if (isset($_POST['nameTac'])) {

      $posPL1=$_POST['PL1'];
      $posPL2=$_POST['PL2'];
      $posPL3=$_POST['PL3'];
      $posPL4=$_POST['PL4'];
      $posPL5=$_POST['PL5'];
      $posPL6=$_POST['PL6'];
      $posPL7=$_POST['PL7'];
      $namet=$_POST['nameTac'];
      $dbTact=$_POST['dbT'];

      $speUp = "UPDATE ".$_POST['dbT']." SET last='0' WHERE
user='".$iui['id']."'";
      mysql_query($speUp,$dbh);

      $rQu = mysql_query("SELECT * FROM ".$_POST['dbT']." WHERE
user='".$iui['id']."'")or die(mysql_error());
      $n = mysql_num_rows($rQu);
```

----- *end* -----

Utilizzando la variabile "*$namet*" gli oggetti possono specificare se la richiesta giunge dalla creazione di una nuova tattica oppure dalla modifica di una esistente.
La differenziazione è facilmente creabile utilizzando nella richiesta dell'oggetto una particolare stringa che non è possibile ritrovare tra i record delle tabelle *tacts* e *tacts_d*.
"" infatti contiene 0 caratteri, caratteristica che non è ritrovabile tra i record già memorizzati dovendo essi essere composti da almeno 1 carattere ed essendo le tattiche preimpostate fornite di un nome ("7vs0") di 4 caratteri.
Grazie a `if($namet == "")` il codice comprende che i dati collezionati devono essere utilizzati per creare una nuova tattica, segue quindi la query di inserimento che utilizza "*$dbTact*" per determinare in quale tabella va inserito il nuovo record, l'id utente catturato dai dati di sessione ed imposta come uovo "*name_t*" una stringa formata dal prefisso "NewTact" al quale si concatena il valore di "*$n*" che indica il numero di record già presenti associati all'utente corrente.
Vengono poi utilizzati i valori da "*$posPL1*" a "*$posPL7*" ed il campo "last" viene impostato a 1, indicando così qual'è l'ultima tattica creata/modificata dall'utente.

Nel caso in cui "*$namet*" sia invece diverso da "", significa che la procedura è di modifica tattica. Se però la modifica è fatta alla tattica "7vs0" viene aggiornato solo il relativo campo "*last*"

impostandolo a 1. Questa operazione è necessaria dato che a inizio proceduta ogni campo "*last*" relativo alle tattiche dell'utente corrente era stato resettato a 0.
Viene successivamente eseguita la visualizzazione dell'alert in javascript che indica all'utente che l'operazione non è possibile perché "7vs0" non è modificabile.
Nel caso in cui invece "*$namet*" non rientri nelle precedenti casistiche, quindi validato, è sicuramente identico a uno dei campi "*name_t*" di uno dei record già presenti in tabella specificata da "$dbTact".
Il record corrispondente a id utente corrente e "*$namet*", subisce l'aggiornamento. L'UPDATE inciderà solo sui campi relativi ad ogni giocatore e sul campo "last" come sempre impostato a 1.

----- *tacts.php* (richieste oggetti 2)-----

```
      if($namet == ""){

            $insT = mysql_query("INSERT INTO ".$dbTact." (ID, name_t, user, P1,
P2, P3, P4, P5, P6, P7, last) VALUES ('','NewTact".$n."', '".$iui['id']."',
'".$posPL1."', '".$posPL2."', '".$posPL3."', '".$posPL4."', '".$posPL5."',
'".$posPL6."', '".$posPL7."', 1)");

      } else {

            if ($namet == "7vs0") {

                  $upM = "UPDATE ".$dbTact." SET last='1' WHERE name_t='7vs0'
AND user='".$iui['id']."'";
                  mysql_query($upM,$dbh);
                  echo "<script type='text/javascript'>alert('Non puoi
modificare 7vs0');</script>";

            } else {

                  $upT = "UPDATE ".$dbTact." SET
P1='".$posPL1."',P2='".$posPL2."',P3='".$posPL3."',P4='".$posPL4."',P5='".$posPL
5."',P6='".$posPL6."',P7='".$posPL7."',last='1' WHERE name_t='".$namet."'" AND
user='".$iui['id']."'";
                  mysql_query($upT,$dbh);
            }
      }
}
```

----- *end* -----

Partendo da questa porzione di codice inizia la visualizzazione vera e proprie della pagina web.
L'intera procedura seguente permette di prelevare l'elenco delle tattiche relative all'utente corrente da entrambe le tabelle *tacts* e *tacts_d*. Successivamente attraverso i due cicli while, viene individuato il record che ha valore del campo "*last*" impostato a 1, e di esso vengono prelevate tutte le informazioni relative al nome e giocatori.

----- *tacts.php (tattica last)*-----

```
$allT = mysql_query("SELECT * FROM tacts WHERE user='".$iui['id']."'")or
die(mysql_error());
$allTD = mysql_query("SELECT * FROM tacts_d WHERE user='".$iui['id']."'")or
die(mysql_error());     //prendo le tattiche relativa all'user_id per attacco e
difesa

while ($cAT = mysql_fetch_array($allT)){
      if ($cAT['last'] == 1)
```

```
	{
	$VAname=$cAT['name_t'];
		$VAPL1=$cAT['P1'];
		$VAPL2=$cAT['P2'];
		$VAPL3=$cAT['P3'];
		$VAPL4=$cAT['P4'];
		$VAPL5=$cAT['P5'];
		$VAPL6=$cAT['P6'];
		$VAPL7=$cAT['P7'];
		$IdA=$cAT['ID'];
	}
}

<...>	//while relativo alle tttiche di tacts_d
```

----- *end* -----

Analizzando la prossima porzione di codice si parla delle interazioni tra PHP, SQL e AJAX. E’ necessario quindi suddividere la descrizione del codice che da ora inizia a concentrarsi su determinate meccaniche piuttosto che sul solo aspetto grafico del progetto.

3.3.1 - Interazioni PHP/Ajax/SQL:

La seguente porzione di codice del documento *tacts.php* si occupa di concetti grafici che sfruttano la tecnologia AJAX. E' necessario quindi evidenziare gli aspetti che governano le dinamiche di aggiornamento in tempo reale di oggetti complessi.

Le prime istruzioni selezionano dal database ogni record che descrive delle tattiche relative all'utente corrente, successivamente viene costruito un form. La procedura è già stata analizzata durante la descrizione di *form.php* ma il codice in esame contiene una particolarità. La voce della listbox che viene costruita con attributo `selected`, corrisponde al record il quale campo "*last*" è impostato a 1. La voce della listbox a cui è stato assegnato questo attributo sarà selezionata direttamente all'inizializzazione della pagina web.
Nel ciclo while di costruzione è presente una istruzione di condizione if. Conoscendo le regole di composizione dei record della tabella sappiamo che tra tutti esiste un solo record che abbia campo "*last*" impostato a 1, i rimanenti saranno impostati a 0.
Il codice quindi analizza un record alla volta cercando questo unico record. Se il controllo fallisce il record viene inserito nella listbox come un normale elemento non selezionato secondo la sintassi `"<OPTION value=".$cATM['ID']."></OPTION>".$cATM['name_t']."</OPTION>"`.
Nel caso in cui invece il controllo verifichi l'unicità, esso viene inserito nella listbox secondo la sintassi `"<OPTION value=".$cATM['ID']."selected>".$cATM['name_t']."</OPTION>"`.
Un'identica procedura viene seguita relativamente alla listbox delle tattiche di difesa.
Le listbox possiedono un riferimento alla funzione javascript `onchange='showTacA(this.value)'` e `onchange='showTacD(this.value)'` che permettono di sfruttare le due relative funzioni descritte all'inizio dell'analisi di *tacts.php*. Esse hanno la stessa funzione di quelle descritte accuratamente durante l'analisi di *form.php* e permettono un aggiornamento dinamico di alcune parti del codice HTML. Il codice a cui si fa riferimento sarà utilizzato per aggiornare elementi complessi come i file *.swf* inclusi nel codice che rappresentano le tattiche graficamente.

----- tacts.php (listbox)-----

```
$allTM = mysql_query("SELECT * FROM tacts WHERE user='".$iui['id']."'")or
die(mysql_error());
$allTMD = mysql_query("SELECT * FROM tacts_d WHERE user='".$iui['id']."'")or
die(mysql_error());

Print "<form action='tacts.php' method='POST'>";
Print"Tattiche (Attacco):<br>";
Print"<SELECT name='TacAtt' onchange='showTacA(this.value)'>";

while($cATM = mysql_fetch_array($allTM))
{
      if ($cATM['last'] == 1) {
            Print "<OPTION value=".$cATM['ID']."
selected>".$cATM['name_t']."</OPTION>";
      } else {
            Print "<OPTION value=".$cATM['ID'].">".$cATM['name_t']."</OPTION>";
      }
}
Print "</SELECT>";
Print "<input type='submit' value='X'>";
Print "</form>";

<...>       //listox tattiche difesa
```

----- end -----

L'ultima parte del codice di *tacts.php* genera gli elementi fondamentali di questa pagina web.
La prima condizione in cui ci si imbatte stabilisce le proprietà del campo di inserimento testo e pulsante di invio relativi al form creato per rinominare una tattica selezionata.
Precedentemente abbiamo chiarito l'esclusività della tattica di base "7vs0", quindi se essa viene selezionata, la proprietà *type* degli elementi del form vengono impostati a "*hidden*". In questo modo gli elementi non sono visualizzabili ed è quindi limitato il loro utilizzo.
Le successive istruzioni prevedono la stessa procedura relativamente alle tattiche di difesa.

Il codice successivo permette di includere un oggetto definito "*application/x-shockwave-flash*". Le caratteristiche dell'elemento vengono analizzate nel prossimo paragrafo, per ora è sufficiente chiarire che l'oggetto è un file con estensione *.swf*.
Il file in esame permette di avere un riscontro grafico sulle tattiche impostate. La sua inclusione inoltre permette di specificare attraverso il parametro "*flashvars*" i parametri che assistono la sua esecuzione.
Riguardo al progetto infatti *flashvars* viene utilizzato al fine di passare a *tacts.swf* i valori codificati relativi ai giocatori ricavati dal record selezionato proveniente dalla tabella *tacts* del database ed il nome della tattica.
Si noti che il paragrafo HTML `div` che contiene l'oggetto, ha un attributo `id='ATTtac'`, tale particolare permette come già analizzato in precedenza, alle funzioni javascript relative alla tecnologia AJAX di elaborare in tempo reale un aggiornamento isolato al codice HTML.

All'interno dello stesso `div` è presente anche la struttura che costituisce un form. In questo caso si tratta del form che permette di rinominare la tattica selezionata. Come descritto precedentemente, gli attributi *type* di determinati elementi di esso sono definiti dalle variabili "*$tYA*" e "*$tYAb*". Variabili inizializzate durante l'esecuzione della prima condizione analizzata in questa porzione di codice che permette di nasconderli nel caso in cui la tattica selezionata sia "7vs0".

Lo stesso codice viene proposto per quanto riguarda le tattiche difensive, le poche differenze si possono riassumere nell'utilizzo di un file swf (*tacts_d.swf*) e di id per il paragrafo `div` (`'DIFtac'`) diversi. Indirettamente anche i valori prelevati dal database saranno reperiti da una tabella differente, *tacts_d*.

Nelle ultime righe di codice è presente la parte conclusiva dell'ormai nota procedura di riconoscimento della sessione utilizzata in ogni pagina web della piattaforma.

----- tacts.php (inclusione swf)-----

```
if ($VAname == "7vs0") {     //proprietà del form per rinominare
      $tYA = "hidden";
      $tYAb = "hidden";
} else {
      $tYA = "text";
      $tYAb = "submit";
}

Print"<div id='ATTtac' style='position : absolute; top : 20px; left : 220px'>";

Print"<object type='application/x-shockwave-flash'";
Print"data='swfs/tacts.swf'";
Print"width='240' height='440'>";
Print"<param name='movie' value='tacts.swf' />";
Print"<param name='quality' value='high'/>";
Print"<param name='flashvars'
value='NAMET=".$VAname."&P1=".$VAPL1."&P2=".$VAPL2."&P3=".$VAPL3."&P4=".$VAPL4."
&P5=".$VAPL5."&P6=".$VAPL6."&P7=".$VAPL7."'/>";
```

```
Print"</object>";

Print "<form action='tacts.php' method='POST'>";
Print "<input type='".$tYA."' name='tNameA' value='".$VAname."'>";
Print "<input type='hidden' name='idAt' value='".$IdA."'>";
Print "<input type='".$tYAb."' value='Rinomina'>";
Print "</form>";
Print"</div>";

<...>       //prorietà form rinomina tattica difesa

<...>       //inclusione tacts_d.swf e form

Print"<br>";
Print"<a href='index.php' style='position : absolute; top : 480px'>Home</a>";

} else {
      Print "Session have been lost, please <a href='index.php'>Login</a>
again";
}

?>
```

----- *end* -----

Le funzioni javascript fanno riferimento a dei file PHP esterni al fine di aggiornare le parti di HTML attraverso la tecnologia AJAX.
Riguardo il codice relativo alle tattiche d'attacco, il documento riceve dalla funzione javascript in *tacts.php* solo l'id della tattica selezionata. Tramite una query di selezione molto semplice il codice ricava dalla tabella *tacts* tutti i dati del record indicato da "*$id*".

Le successive istruzioni agiscono in modo identico a quelle simili in *tacts.php*. E' evidente però che i dati passati tramite *flashvars* sono differenti, il risultato in output quindi è lo stesso oggetto inizializzato però in modo completamente diverso. Lo stesso accade per il form sottostante incluso nello stesso paragrafo HTML da aggiornare.

----- *getAtac.php* -----

```
<?php

<...>

$id=$_GET["id"];

<...>       //connessione al database

$newt = mysql_query("SELECT * FROM tacts WHERE ID='".$id."'")or
die(mysql_error());

$vals = mysql_fetch_array( $newt );

if ($vals['name_t'] == "7vs0") {
      $tYA = "hidden";
      $tYAb = "hidden";
} else {
      $tYA = "text";
      $tYAb = "submit";
}

Print"<div id='ATTtac'>";
Print"<object type='application/x-shockwave-flash'";
```

```
Print"data='swfs/tacts.swf'";
Print"width='240' height='440'>";
Print"<param name='movie' value='tacts.swf' />";
Print"<param name='quality' value='high'/>";
Print"<param name='flashvars'
value='NAMET=".$vals['name_t']."&P1=".$vals['P1']."&P2=".$vals['P2']."&P3=".$val
s['P3']."&P4=".$vals['P4']."&P5=".$vals['P5']."&P6=".$vals['P6']."&P7=".$vals['P
7']."'/>";
Print"</object>";
Print "<form action='tacts.php' method='POST'>";

Print "<input type='".$tYA."' name='tNameA' value='".$vals['name_t']."'>";
Print "<input type='hidden' name='idAt' value='".$id."'>";
Print "<input type='".$tYAb."' value='Rinomina'>";
Print "</form>";
Print"</div>";

mysql_close($dbh);
?>
```

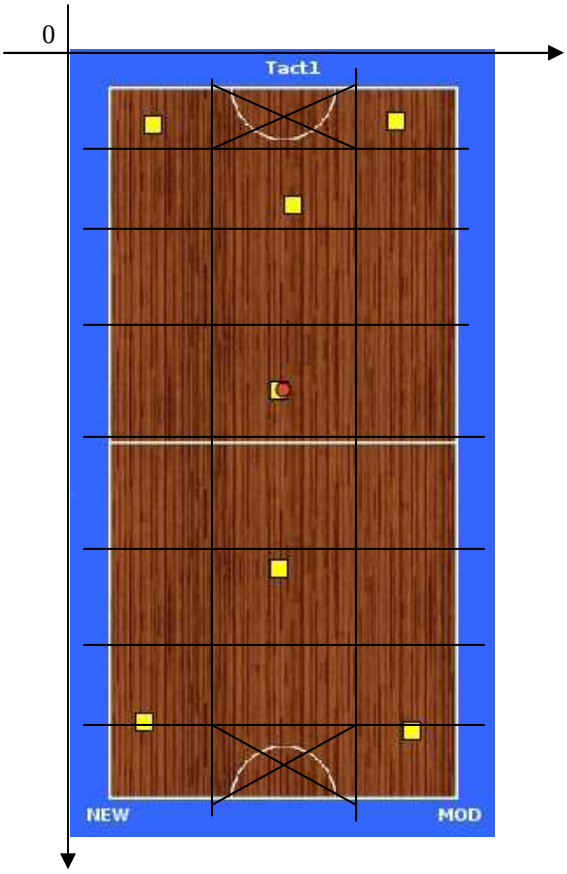
----- *end* -----

Per quanto concerne la parte delle tattiche difensive la procedura si rivela identica ad eccezione della tabella *tacts_d* coinvolta nel codice al posto di *tacts*. Il file esterno PHP che governa l'aggiornamento è *getDtac.php*.

3.3.2 - File tacts.swf e tacts_d.swf in Actionscript 3.0:

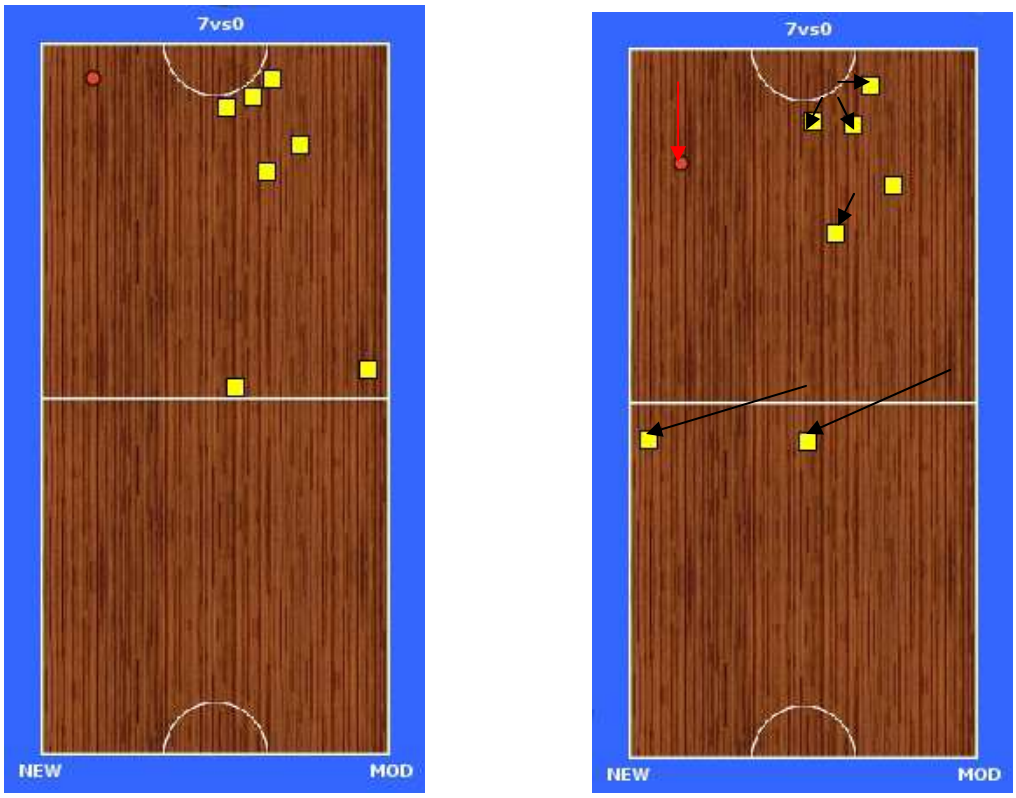
Passiamo ora all'analisi degli oggetti coinvolti nel codice.
Tali oggetti sono i file *tacts.swf* e *tacts_d.swf* riferiti rispettivamente alle tattiche di attacco e di difesa. Essi sono identici in tutto, tranne che per minimi particolari, ossia riferimenti alle tabelle che sfruttano (*tacts* e *tacts_d*). Questo che può sembrare un dettaglio, si rivela invece importante dato che essi possono effettuare delle richieste complete di parametri a documenti PHP che hanno, come abbiamo visto, facile accesso alle tabelle del database.

L'oggetto permette di visualizzare le tattiche selezionate in modo semplice. Il concetto di tattica è basato sulla posizione in campo dei giocatori dipendente dalla posizione della palla.
Il terreno di gioco è suddiviso in 22 settori, relativamente ad ognuno di essi i giocatori prenderanno una posizione definita.
I settori sono numerati da sinistra verso destra partendo dall'alto e procedendo di riga in riga verso il basso. L'oggetto è largo 240 pixel e alto 440. Il bordo è spesso 20 pixel lungo tutto il contorno, si evince quindi che le dimensioni del solo campo sono di 200 X 400 pixel e la posizione dei giocatori è descritta da coordinate cartesiane X e Y con origine posta in alto a sinistra.
I giocatori possono quindi assumere coordinate X da 20 a 220 pixel e Y da 20 a 420 pixel.
Il cerchio rotondo rappresenta la palla, mentre i quadrati gialli rappresentano i giocatori in formazione.



Riferimenti cartesiani per la grafica del file tacts.swf e divisione in settori.
I settori contrassegnati sulle aree non sono validi perchè zona di gioco proibita.

La particolarità dell'oggetto è data dal fatto che i giocatori, che non sono selezionabili, si possono muovere proporzionalmente alla palla.
Poniamo che la palla sia nel settore 1 e volessimo "trascinarla" utilizzando il cursore del mouse fino al settore 3 sottostante. Ai giocatori per i due settori sono associate delle coordinate e quindi delle posizioni diverse. Mentre la palla subisce lo spostamento verso il settore 3, essi cambiano posizione proporzionalmente alla distanza compiuta e da compiere della palla per giungere il settore di destinazione verso le loro nuove posizioni.
Questo effetto grafico ha il compito di conferire fluidità alle azioni in modo da poter comprendere al meglio gli spostamenti che successivamente i giocatori attueranno durante la partita.



Le due rappresentazioni permettono di comprendere come, al movimento della palla dal settore 1 al settore 3, reagiscono i giocatori. Il loro movimento è proporzionale al "trascinamento" via cursore.

Questo processo è di vitale importanza al fine dell'obbiettivo del progetto. Le partite infatti vengono autogenerate con la stessa tecnologia Flash ed è quindi possibile all'utente seguire la vera e propria partita che determinerà il risultato finale dello scontro.

I file *.swf* sono il risultato dell'esportazione di documenti Flash. Questi documenti sono sviluppati attraverso l'utilizzo del linguaggio Actionscript 3.0 già descritto. Attraverso le istruzioni è possibile includere immagini come nel caso dello sfondo di *tacts.swf* che ricorda il parquet utilizzato nei campi sportivi di basket, oppure altri file *.swf*, come vedremo nel caso della partita autogenerata.
I documenti relativi alle tattiche ora in esame sono composti da circa 1400 righe di codice, verranno quindi analizzate solo le caratteristiche e le porzioni di codice fondamentali.

La seguente porzione di codice specifica come è possibile creare gli oggetti che identificano palla e giocatori. Per permettere l'interattività con essi è necessario inizializzarli come *Movie Clip*. Lasciamo ai commenti nel codice spiegare come essi interagiscono nella procedura.

----- tacts.swf (1)-----

```
var ball:MovieClip = new MovieClip(); //variabile per la palla
      ball.name = "ballIns";  //cambio nome istanza
      ball.graphics.beginFill(0xe24a2c, 1); //colore della palla
      ball.graphics.lineStyle(1 , 0x000000); //colore contorno palla
      ball.graphics.drawCircle(0,0,4); //dimensioni della palla
      ball.graphics.endFill();

var pl1:MovieClip = new MovieClip();  //variabile giocatore 1/7
      pl1.name = "play1";
      pl1.graphics.beginFill(0xffff00);  //colore giocatore
      pl1.graphics.lineStyle(1, 0x000000);  //colore contorno giocatore
      pl1.graphics.moveTo(0, 0);    //coordinate di inizio disegno figura
      pl1.graphics.lineTo(10, 0);    //coordinate fino a dove tracciare la forma
      pl1.graphics.lineTo(10, 10);
      pl1.graphics.lineTo(0, 10);
      pl1.graphics.endFill();
```

----- end -----

Essi hanno delle proprietà che permettono di interagire con essi in modo più specifico.
`addChild(pl1);` per esempio permette di aggiungere l'oggetto alla scena di base.
`pl1.buttonMode = true;` permette di rendere l'oggetto selezionabile e quindi di interagire con esso attraverso l'utilizzo del cursore del mouse.

Il codice è predisposto per inizializzare l'animazione impostando la palla al centro della figura, ossia nel decimo settore, come è possibile notare nella prima figura analizzata relativa al capitolo. Di conseguenza i sette elementi quadrati gialli giocatore creati si posizionano alle relative coordinate specificate.
Come descritto durante l'analisi di *tacts.php*, le tattiche vengono passate tramite la collezione di parametri nota come *flashvars*. I valori utilizzati sono le stringhe numeriche di 132 caratteri memorizzate nei record delle tabelle *tacts* e *tacts_d* relativamente ai 7 giocatori che compaiono nella visualizzazione grafica della tattica.

La codifica delle stringhe è molto semplice.
Ogni giocatore è posizionato nel campo a seconda delle coordinate cartesiane X e Y. Sappiamo che il campo è largo 200 pixel, quindi il valore delle ascisse può andare da 20 a 220 pixel. Per quanto riguarda le ordinate il range è 20 – 420.
Per forza di cose i giocatori si muovono all'interno degli stessi limiti. La palla che è anch'essa sottoposta alle stesse restrizioni è però vincolata all'interno di 22 settori ben definiti. Dividendo il campo in 3 colonne e 8 righe i settori sarebbero 24, ma le aree ai lati del campo dove i giocatori possono attaccare sono considerate zona proibita. Il gioco che si svolge all'interno di esse è particolareggiato e le azioni si svolgono in modo ben definito e costretto dalle meccaniche del tchoukball.

La seguente variabile contiene a coppie le coordinate centrali di ogni settore:

```
var secCoo:Array = new Array(50, 40, 190, 40, 50, 85, 120, 85, 190, 85, 50, 135,
120, 135, 190, 135, 50, 190, 120, 190, 190, 190 ,50, 250, 120, 250, 190, 250
,50, 305, 120, 305, 190, 305, 50, 355 ,120, 355, 190, 355, 50, 400, 190, 400);
```

La palla mossa, per essere trascinata dal cursore, ha uno spostamento lineare, e grazie all'utilizzo dell'*EventListener*, è possibile intercettare il primo click del mouse che permette di trascinare poi l'oggetto nel campo.

```
ball.addEventListener(MouseEvent.MOUSE_DOWN, startMov);
```

Questo evento come è intuibile dalla sintassi avvia la funzione *startMov*.
Essa prende l'oggetto in questione come parametro, ne salva le posizioni X e Y iniziali ed avvia altri *EventListener*. Quello relativo al movimento che avvia la funzione *moving* e quello relativo al rilascio dell'oggetto attraverso la funzione *endMov*. Attiva infine la possibilità di trascinare l'oggetto.

----- tacts.swf (1) -----

```
function startMov(e:MouseEvent):void
{
      ogg = MovieClip(e.target);  //riferito all'oggetto cliccato (MOUSE_DOWN)

      <...>       //utilità ausiliari

      //salvataggio posizioni iniziali
      posAX = ogg.x;
      posAY = ogg.y;

      ogg.addEventListener(MouseEvent.MOUSE_MOVE, moving)
      ogg.addEventListener(MouseEvent.MOUSE_UP, endMov)
      ogg.startDrag();
}
```

----- end -----

Le proprietà dell'oggetto *ball.x* e *ball.y*, indicano l'esatta posizione dell'elemento in forma di coordinate cartesiane in qualsiasi istante dell'elaborazione. Utilizzato ball quindi come parametro ed associato ad *ogg*, *ogg.x* e *ogg.y* avranno la stessa funzione.

La complicata funzione *moving* compie un controllo appunto sulla posizione dell'oggetto ed in background monitorizza le coordinate della palla attraverso l'assegnazione del valore *secBall*.
Esso è un semplice valore intero che, come variabile globale, tiene conto costantemente del settore nel quale si stanno operando gli spostamenti.
Un controllo infatti sui limiti dei settori stessi individua inizialmente la colonna nella quale l'oggetto è stato spostato conoscendo i valori di ascisse X entro i quali sono state definite le colonne. Una seconda verifica infine sottopone l'oggetto al confronto con i limiti delle righe, attraverso quindi i valori di ordinate Y.
Il risultato permette al sistema di identificare il settore ed impostare *secBall* con un range di valori da 1 a 22 ossia il numero dei settori.

Infine, una volta che il cursore rilascia l'oggetto palla, la funzione *endMov* tramite l'utilizzo della costantemente aggiornata variabile *secBall*, accede all'array di valori *secCoo* e, attraverso la seguente routine, imposta la palla direttamente nel punto centrale del settore corrente.

```
indice = secBall - 1 + (1 * (secBall - 1));
ball.x = secCoo[indice];
ball.y = secCoo[indice + 1];
```

La funzione si conclude rimuovendo *EventListener* di movimento e fine movimento relativi alle funzioni *moving* e *endMov*. Come descritto precedentemente, il movimento dei giocatori è associato e proporzionale al movimento della palla. La codifica delle coordinare relative ad ognuno di esso viene prelevata dai parametri passati e decodificata per essere memorizzata in delle variabili array che permettono una migliore gestione. Possiamo notare che gli array collezionano le coordinate in diverse componenti X e Y.

```
var pl1x:Array = new Array;
var pl1y:Array = new Array;
var pl2x:Array = new Array;
var pl2y:Array = new Array;
<...>
```

Le stringhe numeriche di 132 caratteri vengono analizzate 3 caratteri alla volta. Essi rappresentano a coppie il valore di X e il valore di Y per ogni settore. Quindi i primi 6 valori indicano le coordinate del giocatore numero 1 quando la palla è nel settore 1, i caratteri dal settimo al dodicesimo indicano le coordinate dello stesso giocatore ma relative al settore 2.
E' evidente quindi che esistono 6 caratteri per ogni settore e 6 * 22 = 132.
La routine di conversione preleva 6 caratteri alla volta suddividendo ciclicamente i parametri memorizzati da `plp1` a `plp7`, sfruttando la funzione *substring()* e l'incremento ciclico dei valori ausiliari *benS* e *endS*, assegna in modo ordinato tutti i valori.

----- tacts.swf (2) -----

```
var zf:int; //ausiliare for
var benS:int = 0; //ausiliare inizio stringa
var endS:int = 3; //ausiliare fine stringa

for (zf=0; zf<22; zf++) {

      pl1x[zf] = plp1.substring(benS, endS);
      pl1y[zf] = plp1.substring(benS+3, endS+3);
      pl2x[zf] = plp2.substring(benS, endS);
      pl2y[zf] = plp2.substring(benS+3, endS+3);
      pl3x[zf] = plp3.substring(benS, endS);
      pl3y[zf] = plp3.substring(benS+3, endS+3);
      pl4x[zf] = plp4.substring(benS, endS);
      pl4y[zf] = plp4.substring(benS+3, endS+3);
      pl5x[zf] = plp5.substring(benS, endS);
      pl5y[zf] = plp5.substring(benS+3, endS+3);
      pl6x[zf] = plp6.substring(benS, endS);
      pl6y[zf] = plp6.substring(benS+3, endS+3);
      pl7x[zf] = plp7.substring(benS, endS);
      pl7y[zf] = plp7.substring(benS+3, endS+3);

      benS = benS + 6;
      endS = endS + 6;
}
```

----- end -----

I giocatori che come descritto, devono muoversi proporzionalmente alla palla nelle posizioni definite dalla procedura appena analizzata, sfruttano parte delle diverse funzioni di movimento *moving* e *endMov*.

L'interprete ovviamente non può sapere dove la palla appena mossa terminerà il proprio spostamento. All'avvio quindi della funzione *startMov* è solo possibile esprimere una previsione basata sui settori vicini a quello dove ora l'oggetto è contenuto.
La procedure molto articolata e complessa calcola la distanza dell'oggetto dal centro del proprio settore e dal centro di tutti i settori limitrofi. Stabilisce quindi una percentuale di avanzamento ipotetica da ogni altro centro facilmente rintracciabile da *secCoo* utilizzando la funzione per il calcolo della distanza tra due punti:

```
dislX = Math.abs(ogg.x - secCoo[indTemp]);
dislY = Math.abs(ogg.y - secCoo[indTemp + 1]);
dist = Math.sqrt(Math.pow(dislX, 2) + Math.pow(dislY,2));
```

Le formule di `dislX` e `dislY` computano il valore assoluto delle differenze delle componenti X e Y dei punti. Infine la formula di `dist` calcola la radice quadrata della somma dei quadrati dei valori calcolati precedentemente.
Applicando questa procedura alle coppie di punti è possibile valutarne la distanza. Su di essa poi confrontando quanto già percorso da quanto manca se ne determinano delle percentuali di avanzamento.
Le percentuali vengono tenute in considerazione poi per i possibili movimenti dei giocatori. Essi varieranno la loro posizione relativa al settore di partenza a seconda di quale settore è più vicino, ma influenzati proporzionalmente dalla collezione di possibilità ricavate dal movimento della palla. Tenendo in considerazione le loro posizioni relative quindi, i giocatori si avvicineranno sempre di più al punto di arrivo tanto quanto la palla sarà trascinata vicino al punto centrale del settore di destinazione. Se il cursore rilascia la palla, essa viene automaticamente impostata al centro del suo settore, anche i giocatori verranno impostati nella posizione definita.

In questo modo la tattica viene visualizzata in modo fluido, così da dare la possibilità all'utente di determinare dei percorsi particolari a seconda dei movimenti della sfera durante l'effettiva visualizzazione della partita.

I file *tacts.swf* e *tacts_d.swf* quindi risultano dei contenitori vuoti di informazioni ma perfettamente calibrati per gestire ogni parametro acquisito da *flashvars*. Come però l'oggetto incluso nella pagina HTML può essere eseguito tramite il passaggio di valori, esso può anche generare un output. Attraverso i pulsanti "*NEW*" e "*MOD*", chiaramente visibili nella parte inferiore di *tacts.swf*, sulla base dei primi valori passati (generalmente la tattica "7vs0") il codice permette di creare una nuova tattica oppure di modificare una esistente che non sia però quella di sistema "7vs0".

La gestione di pulsanti è sempre affidata agli *EventListener*. *tm1* e *tm2* sono gli oggetti relativi ai semplici campi di testo che permettono di visualizzare "*NEW*" e "*MOD*". Alla selezione tramite cursore, i listener intercettano l'evento e avviano le procedure *newTa* e *modTa*.
Accedendo alle funzionalità attivate da questi pulsanti è possibile muovere l'oggetto palla come nella precedente modalità ed i giocatori continuano a comportarsi esattamente nello stesso modo seguendo i movimenti governati dagli stessi parameri sfruttati dall'ultima tattica selezionata dalla listbox.

```
tm1.addEventListener(MouseEvent.MOUSE_DOWN, newTa);
tm2.addEventListener(MouseEvent.MOUSE_DOWN, modTa);
```

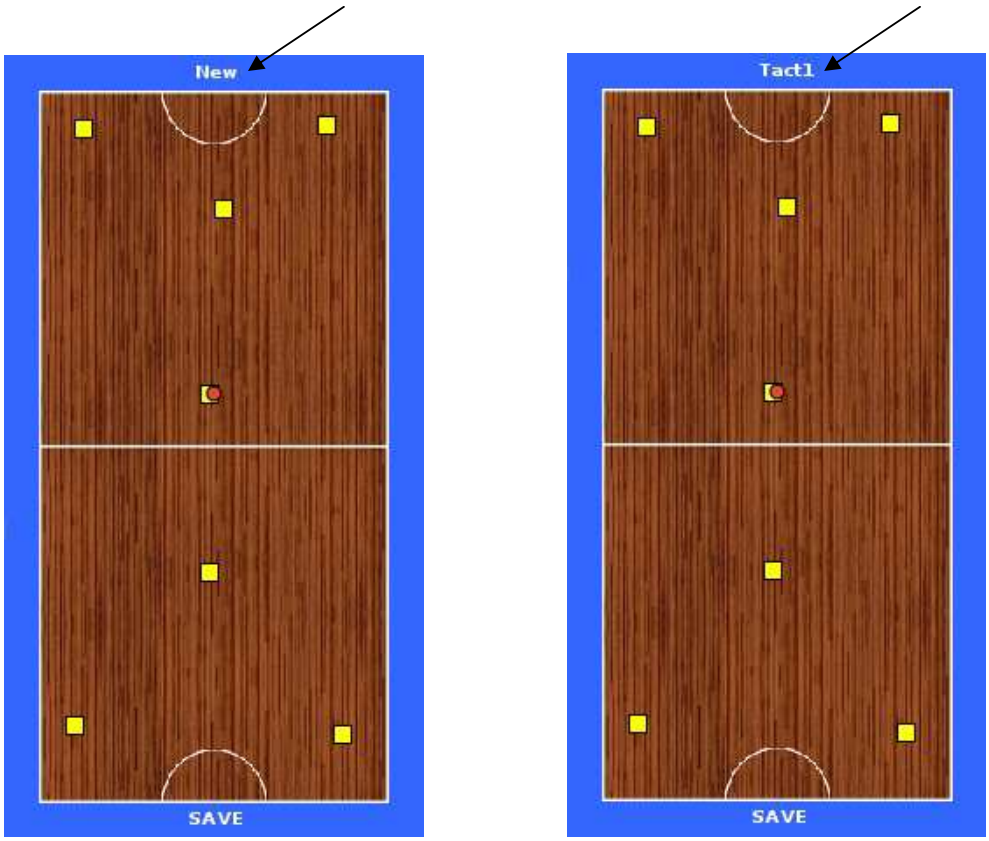
Le funzioni agiscono praticamente allo stesso modo, eseguendo una serie di operazioni fondamentali che trasformano lo stage e lo rendono completamente modificabile come descritto.
La prima permette che i giocatori diventino selezionabili impostando a *true* il parametro `pl1.buttonMode = true;` relativo ad ogni elemento quadrato giallo. Possono quindi, esattamente come l'oggetto palla, essere trascinati e prendere nuove posizioni.
Le nuove coordinate cartesiane ricavate dai parametri già descritti `pl1.x` e `pl1.y` vengono memorizzate negli stessi array descritti in precedenza, differenziati a seconda di ascisse e ordinate. Gli array hanno chiaramente 22 elementi ciascuno, indicizzati da 0 a 21 che riflettono la dipendenza diretta dal settore relativo alla posizione corrente dell'oggetto palla.
La funzione già descritta *startMov* include una sezione dedicata proprio a questo scopo, che riconosce l'elemento in gestione a seconda dell'attributo *ogg.name*.

```
pl1.buttonMode = true;
pl1.addEventListener(MouseEvent.MOUSE_DOWN, startMov)
<...>       //stesse istruzioni per ognuno dei 7 giocatori
```

Successivamente vengono rimossi dalla scena i pulsanti *tm1* e *tm2* e viene aggiunto il pulsante di salvataggio "SAVE" *tm3*. Ad esso viene associato un *EventListener* che avvia le procedure descritte dal codice della funzione *senDa*.

```
removeChild(tm1);
removeChild(tm2);
addChild(tm3);
tm3.addEventListener(MouseEvent.MOUSE_DOWN, senDa)
```

La procedura di generazione di una nuova tattica è sempre relativa al settore. Muovendo quindi la palla da un settore a un altro, si definiscono le nuovissime posizioni dei giocatori in campo muovendo i quadrati gialli che li rappresentano.



Attraverso NEW e MOD si accede a questa modalità che permette di muovere i giocatori e generare o modificare le nuove tattiche. Le diverse procedure sono riconoscibili dal nome della tattica che mantiene lo stesso se modificata.

Una volta che la nuova tattica è stata creata o modificata è necessario rendere definitivo il processo avviando la procedura di salvataggio rappresentata dal tasto “*SAVE*” e descritta dalla funzione *senDa*.

La funzione *senDa* compie due compiti fondamentali: la validazione delle coordinate e l’invio dei dati a *tacts.php*.
Al fine di inviare a *tacts.php* la collezione dei dati nel corretto formato è necessario che ogni elemento contenuto negli array relativi alle coordinate dei giocatori in campo venga nuovamente uniformato alla codifica della stringa di 132 caratteri numerici al fine di poter memorizzare il tutto nella tabella *tacts* o *tacts_d* del database.
La chiave della procedura sta nel fatto che gli elementi passati tramite stringhe che indicano numeri minori di 100 sono composti da 2 e non da 3 caratteri.
Una coordinata per esempio derivante da parametro di *flashvars*, viene estratta sostanzialmente da una stringa di caratteri numerici e non da un numero vero e proprio. Estrapolando quindi una coordinata del tipo 038, viene convertita in numero ed assume il valore di 38 pixel.
Un numero invece di valore 38, convertito in stringa resta un dato stringa composto da 2 caratteri che sono “3” e “8”.
I giocatori come sappiamo possono assumere coordinate da un valore minimo di 20 a un valore massimo di 420, si rende quindi necessario il controllo da parte della funzione.

La validazione di *senDa*, consiste nel prelevare dagli array pl1x[] e pl1y[], i valori numerici e comporre una stringa dichiarata come vuota di nome “*P1P*”.
La procedura seleziona attraverso un ciclo for, tutti e 22 gli elementi di ogni array e concatena il valore numerico individuato alla relativa stringa. Se il valore convertito in stringa ha meno di 3 caratteri, ad esso viene aggiunto come prefisso il carattere “0”. Alternando l’array per le ascisse e l’array per le ordinate, mantenendo valida la regola dei tre caratteri, alla fine del ciclo vengono ottenute in output 7 stringhe da 132 caratteri numerici totalmente conformi a quelle passate da *flashvars* all’inizio della procedura e scomposte inizialmente negli array originali.

----- tacts.swf (senDa - validazione) -----

```
for (k=0; k<22; k++) {

            if (pl1x[k].toString().length < 3) {
                  P1P = P1P + "0" + pl1x[k];
            } else {
                  P1P = P1P + pl1x[k];
            }

            if (pl1y[k].toString().length < 3) {
                  P1P = P1P + "0" + pl1y[k];
            } else {
                  P1P = P1P + pl1y[k];
            }

            <...> //validazione per ogni stringa fino a P7P
      }
```

----- end -----

Il secondo compito di *senDa* è quello di inviare i dati validati a *tacts.php*.
La porzione di codice descrive come preparare i parametri da inserire nella richiesta in modo semplice ed immediato. Il nome della tattica *namet* è lo stesso di quella selezionata dalla listbox di *tacts.php* al momento dell'avvio della procedura di modifica. Il campo invece corrisponde a "*new0000000000000*" nel caso in cui la tattica sia nuova. La ragione dell'utilizzo di questa particolare stringa è stata descritta nell'analisi di *tacts.php*. Notiamo che il riferimento a "*dbT*" indica esplicitamente la tabella *tacts*. Questa è l'unica vera differenza dall'oggetto *tacts_d.swf* che si occupa invece delle tattiche di difesa facendo quindi riferimento alla tabella *tacts_d*.
L'ultima serie di istruzioni costruisce la vera e propria richiesta specificando il metodo "*_POST*". Tramite l'attributo "*_self*" la risposta del web server evita di aprire una nuova pagina del browser, aggiornando semplicemente la stessa da cui la richiesta è partita.

```
                    ----- tacts.swf (senDa - invio) -----

var urlVariables:URLVariables = new URLVariables();
            urlVariables.PL1 = P1P;
            urlVariables.PL2 = P2P;
            urlVariables.PL3 = P3P;
            urlVariables.PL4 = P4P;
            urlVariables.PL5 = P5P;
            urlVariables.PL6 = P6P;
            urlVariables.PL7 = P7P;
            urlVariables.nameTac = namet;
            urlVariables.dbT = "tacts";        //"tacts_d" per tacts_d.swf

      var urlRequest:URLRequest = new URLRequest("tacts.php");
            urlRequest.method = URLRequestMethod.POST;
            urlRequest.data = urlVariables;
            navigateToURL(urlRequest, "_self");

                                ----- end -----
```

Come è stato descritto nella precedente analisi di *tacts.php*, il documento web viene inizializzato nuovamente captando ogni valore qui specificato ed aggiornando quindi il database secondo le istruzioni già analizzate.

3.4 - Campionato e calendario partite:

Il campionato è la struttura dati che permette di collezionare in gruppi le squadre, ossia gli utenti registrati. E' una struttura di tipo piramidale, come ogni campionato infatti esistono diverse serie. Gli elementi che compongono la struttura non sono ancora stati sviluppati, sono invece stati chiaramente progettati ed il lavoro di analisi della piattaforma prosegue da ora (tranne che per alcune eccezioni) descrivendo come essi verranno prodotti.

La progettazione si può suddividere in due diverse fasi. La fase di inizializzazione e la fase stabilizzata.
La fase di inizializzazione tratta della prima pubblicazione della piattaforma in rete, caso unico nel quale i campionati e le tabelle utenti saranno completamente vuote. Non potendo prevedere il numero esatto di utenti che si iscriverà e che quindi giocherà nei campionati creati è necessaria una routine che adatti la strutture agli utenti in modo da ottimizzare le potenzialità di gioco.

Le meccaniche di campionato, approfondite successivamente nel relativo capitolo attraverso la descrizioni delle *operazioni temporizzate*, permettono agli utenti di disputare due partite a settimana. Il giovedì potranno organizzare una partita amichevole con altri utenti, mentre la domenica verranno disputate le partite ufficiali relative al proprio campionato.

La progettazione prevede che sia creata nel database una tabella *campionati*, essa sottoforma di elenco di record collezionerà la lista dei campionati nazionali.

ID	serie	id_serie	cou	T1	T2	T3	T4	T5	T6	T7	T8	inattivi
1	A	1	1	1	2	3	18	11	17	19	20	0
2	B	1	1	5	13	12	4	10	16	15	9	0
3	B	2	1	6	7	14	21	8	0	0	0	3
…	…	…	…	…	…	…	…	…	…	…	…	…
23	A	1	2	26	37	35	27	29	28	38	39	0
…	…	…	…	…	…	…	…	…	…	…	…	…

Tabella "campionati", governata dalle operazioni temporizzate organizza i campionati nazionali.

Il campo "*ID*", indica univocamente i record della tabella. Il campo "*cou*" invece differenzia la nazionalità del campionato, creando infatti dei campionati nazionali separati è possibile generare in seguito delle squadre di giocatori selezionati ed organizzare eventi come i campionati mondiali o europei. Questa eventualità verrà descritta nel capitolo riguardante gli *sviluppi futuri*.
L'attributo "*serie*" descrive il livello del campionato, serie A, B o successive. Questo stratagemma permette di organizzare delle selezioni di promozione/retrocessione attraverso degli spareggi giocati nelle settimane di pausa di fine campionato. Le ultime due squadre classificate di una serie, devono giocare gli spareggi con le migliori due del gruppo di serie sottostanti.
E' previsto infatti che scendendo di un livello, le serie raddoppino, entra quindi in gioco il campo "*id_serie*", che diversifica i campionati minori a partire dalla serie B. Mentre la serie A è unica, le serie B sono due, le serie C sono due per ogni serie B (quindi 4 in tutto) scendendo quindi di ogni livello si ha una progressione di incremento delle serie del tipo 2^x , dove x indica il livello: serie A livello $2^0 = 1$, serie B livello $2^1 = 2$, serie D livello $2^3 = 8$.

Risulta quindi che ogni primo classificato, a fine campionato, della propria serie ha la possibilità di giocarsi uno spareggio con l'ultima o la penultima della serie soprastante, valevole per la promozione. Prendendo come esempio serie A e serie B, l'ultima classificata della seria A, giocherà lo spareggio con la migliore tra le squadre che si sono classificate al primo posto nella serie B.1 e serie B.2. La penultima invece con la peggiore delle due. La decisione relativa a quale delle due è effettivamente la migliore viene presa in base a un *ranking*.
La nozione di *ranking*, verrà approfondita nel capitolo relativo alle *operazioni temporizzate*, esso rappresenterà una classificazione univoca di tutte le squadre iscritte ad un campionato nazionale, giudizio basato in gran parte sulla serie a cui si partecipa, sulla posizione in classifica e sulla differenza punti fatti – punti subiti. Aggiornato automaticamente con cadenza settimanale comporrà in modo univoco una classifica della qualità di tutte le squadre e quindi delle capacità degli utenti che le gestiscono.
I campi da "*T1*" a "*T8*" indicano tramite la chiave primaria della tabella *teams*, le squadre assegnate ad ogni campionato.

La fase di inizializzazione prevede che la tabella *campionati* parta da una situazione di vuoto, in essa quindi, nel momento della pubblicazione del sistema non saranno presenti record. Alla prima registrazione utente è necessario che tra le istruzioni del documento *auth_key.php* si aggiunga una routine che effettui un controllo proprio sulla tabella in esame.
Il controllo dovrà verificare innanzitutto se esistono record riguardanti il campionato relativo alla nazione di provenienza dell'utente attraverso una query di selezione SQL.
Da questa ricerca possono emergere due diversi risultati:

Nell'eventualità che non esistano dei campionati, il codice tramite una query di inserimento dovrà inizializzare un record per la serie A, impostare nel campo "*T1*" l'attributo "*ID*" del team appena attivato e segnalare attraverso il campo "inattivi" che ci sono 7 altri posti da assegnare, impostando ogni altro campo da "*T2*" a "*T8*" a 0.

Se invece esistono già dei record nella tabella *campionati* relativi alla nazione di provenienza dell'utente, tramite una query SQL è necessario selezionarli tutti ed analizzare in ordine di "*serie*" e "*id_serie*" se il campo "*inattivi*" è diverso da 0. Quando il controllo da esito positivo, significa che ci sono in quel record dei campi da "*T1*" a "*T8*" impostati a 0. E' necessario quindi che controllandoli in ordine, l'account utente appena attivato vada ad occupare con il suo "*ID*" in *teams*, il primo campo individuato che abbia per valore 0. Un'istruzione SQL di UPDATE si occuperà di aggiornare i dati, decrementando inoltre il valore del campo "*inattivi*" relativo al record di 1.

E' importante non tenere conto del numero di squadre iscritte, ma effettuare lo stesso controllo ogni volta. Gli utenti infatti dopo una stagione di inattività vengono, attraverso le *operazioni temporizzate*, dichiarati inattivi. Essi continueranno ad esistere senza che l'utente possa effettuare però l'accesso fino a che un nuovo utente non ne prenda il posto. A quel punto attraverso una serie di istruzioni in sintassi SQL di DELETE, ogni traccia dell'utente inattivo viene definitivamente eliminata dal sistema.

La fase di inizializzazione durerà circa 2 settimane. Durante questo periodo gli utenti iscritti potranno disputare delle amichevoli tra di loro il giovedì, ma non ancora partite di campionato.
La fase stabilizzata agisce nello stesso modo. L'unica differenza sarà che i campionati si giocheranno ogni domenica ed il riempimento con sostituzione dei giocatori inattivi avverrà soltanto nella serie di livello più basso esistente. Le squadre di utenti inattivi nelle serie superiori verranno progressivamente retrocessi grazie all'aumento di *ranking* da parte degli utenti attivi e delle partite di spareggio.

E’importante inoltre sottolineare che quando una prima attivazione genera una serie del tutto nuova, i posti delle altre 7 squadre relative al campionato devono essere riempite per dare la possibilità all’utente di avere una normale interazione con la piattaforma. E’ prevista quindi l’esecuzione di una routine al momento dell’attivazione che assegni dei BOT.
I BOT sono delle nuove squadre di sistema, salvate in un’apposita struttura ausiliaria del tutto simile a quelle già analizzate. Generate una sola volta vengono assegnate quando serve ma mai eliminate, solo sostituite da nuovi utenti. Su di essi hanno la precedenza le squadre di utenti inattivi che devono essere smaltiti nelle serie minori.

Una volta che le *operazioni temporizzate* stagionali hanno generato il calendario delle partite per la stagione successiva, dopo che gli spareggi di promozione/retrocessione vengono giocati, è possibile accedere ad ogni partita in calendario. Una nuova tabella *matches* collezionerà l’elenco regolato attraverso la data in cui la partita verrà disputata, di tutte le partite in programma, comprese le amichevoli.

La progettazione prevede una struttura SQL del seguente tipo:

ID	id_camp	data	home	guest	Hp	Gp	AH	DH	AG	DG	plH1	…	plG12	…
1	0	1344698578	3	12	0	0	5	3	32	23	76	…	87	…
2	1	1344697625	3	4	38	22	1	5	55	43	76	…	12	…
…	…	…	…	…							…	…	…	…

Tabella matches, essa colleziona il calendario partite relative a tutte le squadre del sistema

“*id_camp*” indica il campionato di riferimento attraverso l’utilizzo della chiave primaria della tabella *campionati*. Il campo “*data*” esprime il giorno in cui la partita sarà giocata, ovviamente per quanto riguarda il campionato questi valori avranno una semplice cadenza settimanale e sono espressi nel valore in secondi già descritto ed utilizzato nella tabella *users*.
“*home*” e “*guest*” indicano le squadre (“*ID*” in *teams*) che disputeranno il match, ovviamente calcolate in modo equilibrato tra partite di andata e ritorno.
“*Hp*” e “*Gp*” contengono il punteggio se la partita è stata disputata,altrimenti i valori saranno 0.
“*AH*”, “*DH*”, “*AG*” e “*DG*” descrivono le tattiche utilizzate attraverso l’attributo “*ID*” chiave primaria delle tabelle *tacts* e *tacts_d*.
I campi da “*plH1*” a “*plG12*” fanno riferimento alla chiave primaria della tabella *players* ed indicano i 24 (12 + 12) giocatori impostati dall’utente secondo la formazione e la tattica scelta per affrontare la partita.
Esiste la concreta eventualità che siano necessari altri parametri e quindi altri campi per indicare l’impostazione scelta dall’utente al fine di produrre la migliore strategia possibile per affrontare la partita. Basti semplicemente pensare ai ruoli dei giocatori, che codificati come carattere di indice 0 nella stringa relativa ad ogni “*ID*” giocatore ed estratti successivamente dal codice PHP, permetterebbero di risparmiare sulla creazione di nuovi elementi del record.
Condizioni di cambi basati su infortuni o resistenza fisica dei giocatori coinvolti, per esempio creano la necessità di ampliare le possibilità del database.
Tutto questo però fa parte degli eventuali sviluppi futuri, in eccesso per ora alla funzionalità minima della piattaforma ed ipotizzate nei prossimi capitoli.

La regolamentazione della gestione di ogni match è affidato alla pagina *matchlist.php*. Essa permetterà di entrare nel dettaglio di una singola partita ed impostare tutti i valori appena analizzati riguardo ai record della tabella *matches*. Questo è possibile utilizzando elementi HTML come le

listbox per dare un ruolo ed un ordine ai giocatori relativamente alle tattiche scelte, per selezionare eventuali cambi e per selezionare le tattiche stesse da utilizzare. Questi semplici elementi limitano e restringono le opzioni di scelta permettendo così di omettere in molti casi la validazione.
Un pulsante *submit* permetterà infine di salvare le scelte effettuate inviando i dati collezionati al documento *matchlist.php* stesso, che come abbiamo visto in *tacts.php* può riconoscere un'operazione differente dall'inizializzazione semplicemente controllando la presenza di eventuali valori passati tramite metodo "*_POST*".

Nel caso in cui un utente ometta di impostare i dati relativi ad una singola partita attraverso *matchlist.php*, verranno utilizzate esattamente le stesse impostazioni dell'ultima partita giocata nella stessa categoria (campionato o amichevole). Se invece le impostazioni non sono mai state specificate, verranno utilizzate le tattiche di sistema "7vs0" per attacco e difesa. I giocatori selezionati in semplice ordine alfabetico ed i ruoli scelti in modo casuale ma sempre conforme alla posizione in campo. Ovviamente non in base alle capacità.

3.5 - Allenamento:

Ogni giocatore presente nella tabella *players* possiede degli attributi, caratteristiche che ne determinano la capacità ed il rendimento sul terreno di gioco.
Questi parametri però non sono statici, possono infatti aumentare o diminuire a seconda dell'allenamento o di fattori determinati dalle caratteristiche stesse.

La possibilità di gestirli è affidata alla pagina *training.php* che interagirà con le tabelle *players* e *teams* analizzate precedentemente.
Al fine di creare le procedure relative all'allenamento è necessario però ampliare le tabelle già analizzate inserendo nuovi campi nei record.
Al fine di migliorare le statistiche dei giocatori è necessrio allenarli, si rivela quindi necessaria la presenza in squadra di allenatori. Essi possono essere ingaggiati ed a seconda del loro livello (compreso come per le statistiche dei giocatori nel range 0 – 9), il loro ingaggio ed il loro stipendio settimanale aumenterà.
Una listbox consente di scegliere il livello dell'allenatore desiderato ed attraverso la tecnologia AJAX un paragrafo *div* HTML permette di visualizzare, aggiornato alla selezione, il suo costo complessivo. Questi valori standardizzati sono parte integrante del codice HTML.
L'utente è quindi costretto a calcolare tramite *bilancio.php* (che sarà descritta nei prossimi capitoli), se può permettersi o meno un investimento.
Effettuata la scelta, attraverso il tasto *submit* "*Acquista*", l'allenatore verrà aggiunto alla lista di 3 allenatori che ogni squadra può possedere. Il più "bravo" di essi sarà l'allenatore principale, gli altri assistenti miglioreranno la capacità complessiva di allenamento.

La lista di allenatori posseduti dalla squadra di un utente, viene memorizzata in sei nuovi campi della tabella *teams*. Tramite una semplice query di UPDATE i campi precedentemente inizializzati a 0 tramite l'apposita procedura creata in *auth_key.php*, vengono aggiornati aggiungendo i nomi generati attraverso una porzione di codice simile a quella di *newuser.php* per la determinazione dei nomi dei giocatori dell'utente che ha appena portato a termine la registrazione nuovo utente.

Come è facile intuire i campi "*nameA**" memorizzeranno i nomi generati ed i campi "*livA**" i loro valori di abilità. Non è necessario salvare i costi di ognuno di loro dato che sono standardizzati ed associati direttamente ai valori del range 0 – 9. L'ammontare dei costi può essere ipotizzato, ma sarà determinato solo successivamente ad una sessione di test preventiva alla definitiva pubblicazione del browsergame.

…	team	nameA1	nameA2	nameA3	livA1	livA2	livA3
…	Nome Squadra 1	Mario Rossi	Pietro Bianchi	Marco Verdi	7	4	3
…	Nome Squadra 2	Giulio Neri	0	0	9	0	0
…	…	…	…	…	…	…	…

Espansione della tabella originale teams utilizzando 6 nuovi campi per record.

La visualizzazione quindi di *training.php*, ometterà automaticamente gli allenatori il cui nome corrisponderà alla stringa "0".

Il gioco dà all'utente la possibilità di allenare un solo parametro a settimana che deve essere scelto prima del giovedì. Tale caratteristica deve essere selezionata da una semplice listbox con pulsante di invio dati che contiene gli 8 valori fondamentali di Resistenza, Velocità, Attacco, Difesa, Passaggi, Regia, Riflessi e Tecnica.
L'incremento/decremento degli altri parametri saranno analizzati successivamente.
L'interazione da parte dell'utente si conclude in questo modo e gli eventuali cambiamenti posso essere visualizzati giovedì mattina attraverso la pagina già descritta *playerlist.php*.
Ogni settimana quindi nella notte tra il mercoledì ed il giovedì vengono elaborate ed aggiornate le statistiche dei giocatori attraverso le *operazioni temporizzate*.

L'allenamento dei giocatori è quindi definito da più variabili. Alcune di esse sono semplicemente individuabili come il numero e livello degli allenatori assunti ed il parametro scelto da allenare. Gli altri indici sono definiti dal livello di talento dei singoli giocatori, parametro nascosto che influisce la capacità d'apprendimento di ogni singolo elemento. L'età, che diminuisce progressivamente di un piccolo contributo l'incremento delle statistiche dipendentemente a quanto avanza e, fondamentale la percentuale di tempo trascorso in campo durante l'ultima partita di campionato della domenica. Anche le amichevole influiscono leggermente.

Di settimana in settimana è necessario mantenere una traccia della quantità di minuti giocati. Entra in gioco quindi la tabella *players* già descritta, alla quale viene aggiunto il campo "*%*". Essa indicherà settimanalmente una percentuale che descrive la somma tra la percentuale dei minuti giocati in campionato (0% - 100%) ed un aumento proporzionale definito dai minuti giocati in un eventuale amichevole che, valutati in scala percentuale, sono poi diminuiti a un decimo.
A un giocatore per esempio che gioca sempre ma solo in campionato verrà impostato un valore del 100%. Uno che invece che gioca solo uno dei tre tempi di un'amichevole, il valore impostato sarà del 33%, ma per quanto vale un amichevole sarà solo del 3,3% arrotondato per difetto al 3%.
Il giocatore che gioca sempre arriverà ad un tetto massimo di 100% più un decimo di 100%, in tutto quindi a 110%.

…	Form	Anni	LR	%
…	…	…	…	…
…	5	15	D	100
…	…	…	…	

Tabella players con l'aggiunta del campo %

Le formule esatte devono essere prodotte anche dipendentemente al tipo di caratteristica, la Resistenza infatti è molto più facile da allenare rispetto alla Tecnica.

L'effettiva computazione dei miglioramenti relativi ad ogni singolo giocatore sono stabiliti da documenti analizzati successivamente nel capitolo riguardante le *operazioni temporizzate*, ma è possibile già anticipare che il livello degli allenatori congiunti e la scelta della caratteristica da allenare, stabilisce un indice di incremento.
Il valore stabilito poi viene moltiplicato per un indice definito dal valore di talento del singolo giocatore e successivamente moltiplicato ancora per la percentuale memorizzata in *players*. Il valore risultante è l'incremento applicato alla statistica esistente.

Ci sono valori che possono anche essere decrementati. Tutte le statistiche se non allenate subiscono un degrado costante determinato soprattutto dall'età del giocatore.
Il parametro età infatti aumenta in modo costante ad ogni fine stagione di campionato.
Il talento è sempre identico, non può variare, ma la sua funzione è contrastata dagli effetti dell'età.
Il valore di forma aumenta e diminuisce in modo totalmente casuale ma limitato a piccoli sbalzi settimanali.
Il parametro esperienza può solo aumentare e lo fa progressivamente rispetto a quanto il giocatore viene utilizzato settimanalmente a che livello di serie esso gioca.
La disciplina tattica è molto legato all'esperienza ma cala drasticamente se il giocatore non viene utilizzato. Una volta che esso partecipa ad importanti competizioni come in campionato esso risale velocemente in poche settimane per ristabilizzarsi.

3.6 - Mercato:

Il mercato permette di vendere ed acquistare giocatori. Ad esso si può accedere dal menù di sistema attraverso il documento *mercato.php*.
La gestione di questa parte della piattaforma si basa sulla tabella *mercato* del database. La sua funzione però non è esattamente quella di un classico mercato dove i giocatori possono essere acquistati liberamente come avviene per gli allenatori. Il sistema si basa sulle aste, vince quindi l'utente che fa l'offerta più alta.
I giocatori possono essere inseriti nelle aste in due modi: attraverso la generazione da parte del sistema di giocatori compatibili in tutto e per tutto con i campi della tabella *players* (*operazioni temporizzate*), oppure offerti dagli utenti in possesso di giocatori che non vogliono più.

In entrambi i casi i giocatori saranno introdotti o già presenti nella tabella *players* solo che quelli generati automaticamente sono associati alle squadre BOT di sistema descritte precedentemente. Per distinguerli però dai giocatori normalmente in possesso degli utenti, il loro campo "*cou*" sarà impostato a 0 (valore non tollerato), ed il campo "*user_id*" rifletterà la chiave primaria della tabella costruita appositamente per collezionare le squadre BOT.
Nel codice già creato quindi andrà introdotto un nuovo controllo sul campo "*cou*", che farà comprendere al sistema di avere a che fare con giocatori non standard.

I record presenti nella tabella *mercato* quindi non dovranno assumere tutti i valori delle statistiche dei giocatori, ma solamente il loro "*ID*" riferito alla tabella *players* tramite l'attributo "*pl_id*".
Il controllo incrociato poi con la tabella *players*, permetterà di individuare ogni dettaglio relativo al giocatore.
Di seguito la tabella *mercato*, che identifica l'asta attraverso la chiave primaria univoca "*ID*", il giocatore in vendita con l'attributo "*pl_id*", la traccia dell'offerta massima "*off_max*" e l'utente che si sta aggiudicando l'asta attraverso "*winner*". Il campo "*scadenza*" indica quando l'asta si concluderà.
Il tempo previsto di durata dell'asta, è di 7 giorni esatti dal momento in cui viene immesso il giocatore sul mercato.

ID	pl_id	off_max	winner	scadenza
…	…	…	…	…
345	24	45350	0	1349291095
346	78	4000	21	1344767422
…	…	…	…	…

Struttura dati della tabella mercato.

Il tempo è però indicativo, è previsto infatti che ad ogni offerta effettuata negli ultimi 2 minuti, il tempo si prolunghi di 120 secondi per dare il tempo ad eventuali contendenti di di rilanciare.

mercato.php, si presenterà in forma di semplice lista. Accanto ad ogni giocatore in vendita il progetto prevede la presenza di un tasto "Offri" attraverso il quale si accede ad una procedura e quindi ad un nuovo documento PHP (*offerta.php*). La pagina web genererà una visualizzazione specifica sulla singola vendita dotata di un form comprensivo di campo di testo per l'inserimento della cifra da offrire. Alla conferma dell'acquisto, lo specifico campo "*off_max*" viene aggiornato e, nel campo "*winner*" viene impostato l'id dell'utente che si sta aggiudicando il giocatore.

Se il campo “winner” impostato a 0, significa che nessuno ha ancora fatto un’offerta nei confronti del giocatore.

Il progetto prevede inoltre la possibilità di cercare nella tabella *mercato* un elemento dotato di determinate caratteristiche attraverso delle query di selezione dinamiche. E’ sufficiente che *mercato.php*, sia strutturata nello stesso modo di *tacts.php*. Che abbia quindi la possibilità quindi di richiamare se stessa passando dei parametri, specificati in un particolare form presente in *mercato.php*. Per semplicità di utilizzo è possibile sfruttare un nuovo documento PHP ausiliario che abbia l’unico scopo di accogliere attraverso un articolato form questo tipo di ricerche che, *merceto.php*, sfrutti al fine di costruire le giuste query attraverso il PHP, collezionare i risultati, ed elaborare una lista con i dati ricavati, sempre ordinata per scadenza.

3.7 - Bilancio:

Questa pagina web ha l'unica funzione di tenere conto del bilancio settimanale relativo alla propria squadra. Gli aspetti descritti da questo documento sono solo a scopo informativo, permettono di tenere traccia delle spese e dei guadagni fornendo infine il saldo settimanale relativamente anche al saldo della settimana precedente. La visualizzazione della pagina è affidata al documento *bilancio.php*.

Il bilancio si può dividere in due diverse tipologie, quello settimanale ed il bilancio straordinario. Il bilancio settimanale è definito dalle spese e dai guadagni fissi. Ogni settimana infatti ai giocatori ed agli allenatori deve essere pagato lo stipendio, ci sono inoltre delle spese di manutenzione della propria palestra dove vengono svolti allenamenti e partite.
Gli introiti invece sono costituiti dagli sponsor e dalla vendita del merchandise.

Il bilancio straordinario invece è costituito dalle scelte dell'utente, comprare un giocatore, oppure ingaggiare un allenatore, rappresentano delle spese segnate in bilancio quando i pagamenti vengono effettuati. Allo stesso modo vendere un giocatore è un introito straordinario.
Inoltre giocare una partita in casa porterà un guadagno dalla vendita dei biglietti ed un aumento dell'ammontare dei ricavati dalla vendita del merchandise, sempre rispettivamente al concetto di *ranking* espresso precedentemente.
Ad ogni finale di stagione inoltre gli sponsor rinnoveranno i contratti, insieme agli allenatori però, il loro ingaggio quindi dovrà essere pagato di nuovo. Inoltre ci saranno dei tifosi abbonati che verseranno nuovamente al club le quote di iscrizione.

Bilancio settimanale:	**Bilancio straordinario:**
Uscite: - Stipendi giocatori/allenatori - Manutenzione palestra *Entrate:* - Sponsor - Vendita merchandise	*Uscite:* - Acquisto allenatori - Rinnovo allenatori - Acquisto giocatori *Entrate:* - Vendita giocatori - Vendita biglietti - Rinnovo sponsor - Rinnovo abbonamenti - Aumento vendita merchandise

Schematizzazione della struttura del bilancio settimanale e straordinario.

La visualizzazione della pagina quindi sarà basata sul bilancio settimanale descritto al quale di volta in volta vengono aggiunti gli eventuali punti del bilancio straordinario se essi avranno un valore che effettivamente incide sul bilancio totale, quindi se diversi da 0.
E' giusto sottolineare che quando i contratti per gli allenatori vengono rinnovati, il loro stipendio non viene pagato.

Il codice quindi sviluppato in *bilancio.php* è di per se molto semplice strutturalmente, alcuni controlli governati da una serie di condizioni if aggiungono alla visualizzazione HTML della pagina web le eventuali voci del bilancio straordinario in aggiunta a quelle standard del bilancio settimanale.
I calcoli per ognuno di essi sono effettuati dalla pagina stessa estraendo però i dati dalle varie tabelle del database attraverso le consuete query di selezione.
I valori base del bilancio memorizzati nella relativa tabella *bilancio*, sono modificati periodicamente attraverso le *operazioni temporizzate*, ed in casi particolari come acquisti di giocatori/allenatori o vendite di giocatori.

Secondo la progettazione la tabella *bilancio* sarà strutturata come la tabella *teams*. La sua chiave primaria infatti fa da riferimento diretto alla chiave primaria della tabella *users*. In questo modo esse rappresentano una sua estensione, visto che ogni utente avrà solo un bilancio e solo una squadra, ma per semplicità progettuale sono divise permettendo così di avere a che fare con moli di dati limitate.
"*ID*" quindi rappresenta la chiave primaria nonché il riferimento all'utente, "*bil_old*" e "*bil_new*" rappresentano rispettivamente il bilancio della settimana precedente ed il bilancio attuale, le *operazioni temporizzate* si occuperanno di aggiornare il primo campo con il valore del secondo settimanalmente.
L'esistenza di questa tabella implica l'ampliamento del documento *auth_key.php*, aggiungendo in esso una nuova procedura dedicata all'inizializzazione della tabella *bilancio*. Per comodità di esecuzione la procedura memorizzerà la serie di appartenenza nel campo "*serie*" della tabella.
L'omissione di questa operazione implicherebbe una complicata ricerca attraverso delle complesse query da eseguire relativamente a *campionati*.
Il campo "*abbonati*" tiene traccia del numero di tifosi che pagano regolarmente la quota di iscrizione al club e ne determinano quindi gli introiti stagionali e straordinari attraverso i biglietti.
Gli ultimi tre campi tengono traccia di eventi straordinari che sono rispettivamente l'acquisto di giocatori, la loro vendita e l'acquisto di nuovi allenatori.

ID	bil_old	bil_new	serie	abbonati	acq_pl	ven_pl	acq_all
…	…	…	…	…	…	…	…
8	126150	180300	A1	45	0	0	13000
9	30530	23370	B2	92	15780	0	0
…	…	…	…	…	…	…	…

Struttura della tabella bilancio

Le voci del *bilancio* verranno talvolta gestite in modo asincrono.
Gli stipendi dei giocatori devono essere selezionati dai record della tabella *players*, è necessario quindi aggiungere ad essa un nuovo campo "*stipendio*", nel quale verrà prima dell'inizializzazione e successivamente ad ogni stagione (*operazioni temporizzate*) aggiornato il valore dello stipendio in base ai miglioramenti ottenuti tramite l'allenamento.
Gli stipendi degli allenatori invece, sono standardizzati in base al loro valore di abilità determinabile dalla tabella *teams*.
Il costo della manutenzione della palestra è proporzionale al numero degli abbonati, aggiornati settimanalmente in base ai risultati in campionato.

Gli introiti invece ricavati dagli sponsor sono direttamente proporzionali alla classifica di *ranking* e la vendita del merchandise al numero di spettatori che partecipano alle giornate di campionato anche se in trasferta.

Questo valore può essere notevolmente incrementato se la partita è giocata in casa.
Ogni spesa o ricavo straordinari, non sono invece gestiti dalle operazioni temporizzate. Eventi come la vendita o l'acquisto di giocatori o allenatori incide immediatamente sul bilancio. Essi quindi necessitano di particolari campi nella tabella che ne fanno un riassunto.
Il rinnovo dei contratti degli allenatori viene valutato all'inizio della nuova stagione sulla base dei record della tabella *teams*. Lo stesso avviene per il rinnovo degli sponsor che sarà però incrementato del 200% solo a inizio stagione sulla base del *ranking*. Con la stessa cadenza viene accreditato il rinnovo degli abbonamenti valutato proporzionalmente al numero degli abbonati.
Il merchandise inoltre procurerà solamente nel caso di partite di campionato giocate in casa un aumento del 50% sul valore di vendita del merchandise base.

Ogni valore finanziario legato al bilancio è fino ad ora solamente ipotizzabile. Solo dopo lo studio dei risultati di una sessione di test della piattaforma, completa e pubblicata, sarà possibile stabilire correttamente e proporzionalmente ogni singolo attributo finanziario al fine di generare un ambiente di gioco realistico.

4 - Operazioni temporizzate:

Le *operazioni temporizzate* sono state citate moltissime volte nell'analisi della piattaforma descritta fino ad ora. Rappresentano la vera e propria anima del gioco, costituiscono la base di ogni interazione con la piattaforma. Esse effettuano operazioni attraverso automatismi a tempo e possono interagire utilizzando il codice in linguaggio PHP con il database.

L'unico modo per avviare l'esecuzione di uno script PHP da un web server è la richiesta della pagina da parte di un utente attraverso il proprio browser. Il linguaggio non gestisce le operazioni a tempo per il semplice motivo che non è possibile con esso generare e mantenere attiva una routine in background che si occupi di far eseguire al web server uno script PHP in un determinato momento oppure ciclicamente.
Al fine di risolvere questo problema, è quindi necessario utilizzare uno strumento caratteristico dei sistemi derivanti da UNIX chiamato Crond. Esso è un demone costantemente in esecuzione in background che permette, attraverso l'impostazione dei suoi file di configurazione, di pianificare ciclicamente l'esecuzione di qualsiasi file di sistema. Nel caso in esame il demone Crond attraverso il comando *crontab* esegue determinati script PHP consentendo quindi l'utilizzo delle *operazioni temporizzate* relative al sistema.

Questo tipo di operazioni si possono suddividere principalmente in due diverse categorie:
- le operazioni riguardanti il gioco, che si occupano di regolare, aggiornare o generare i contenuti del database al fine di rispecchiare un progresso temporale che renda il sistema più realistico possibile intervenendo per esempio sul bilancio o sulle caratteristiche dei giocatori.
- le operazioni di manutenzione, che hanno la funzione di gestire i dati in modo del tutto trasparente all'utente, eliminando per esempio dati obsoleti dal database oppure avviando la generazione delle partite visualizzabili che decretano il risultato di una partita.

E' necessario chiarire com'è articolato il ciclo temporale di campionato. Ogni serie è composta da 8 squadre, pertanto è necessario che ogni squadra giochi 7 partite relative al girone di andata ed altre 7 di ritorno. Questo impone che il campionato duri 14 settimane. A queste vengono aggiunte 2 settimane di pausa tra un campionato e l'altro durante il quale vengono giocate le eventuali partite di promozione/retrocessione ed una settimana di pausa per tutti. Durante ogni settimana vengono creati degli eventi ciclici, come le amichevoli del giovedì (generate il mercoledì) e l'aggiornamento dei valori allenati il giovedì mattina.

1° a	2° a	3° a	4° a	5° a	6° a	7° a	1° r	2° r	3° r	4° r	5° r	6° r	7° r	1° p	2° p
1°	**2°**	**3°**	**4°**	**5°**	**6°**	**7°**	**8°**	**9°**	**10°**	**11°**	**12°**	**13°**	**14°**	**15°**	**16°**

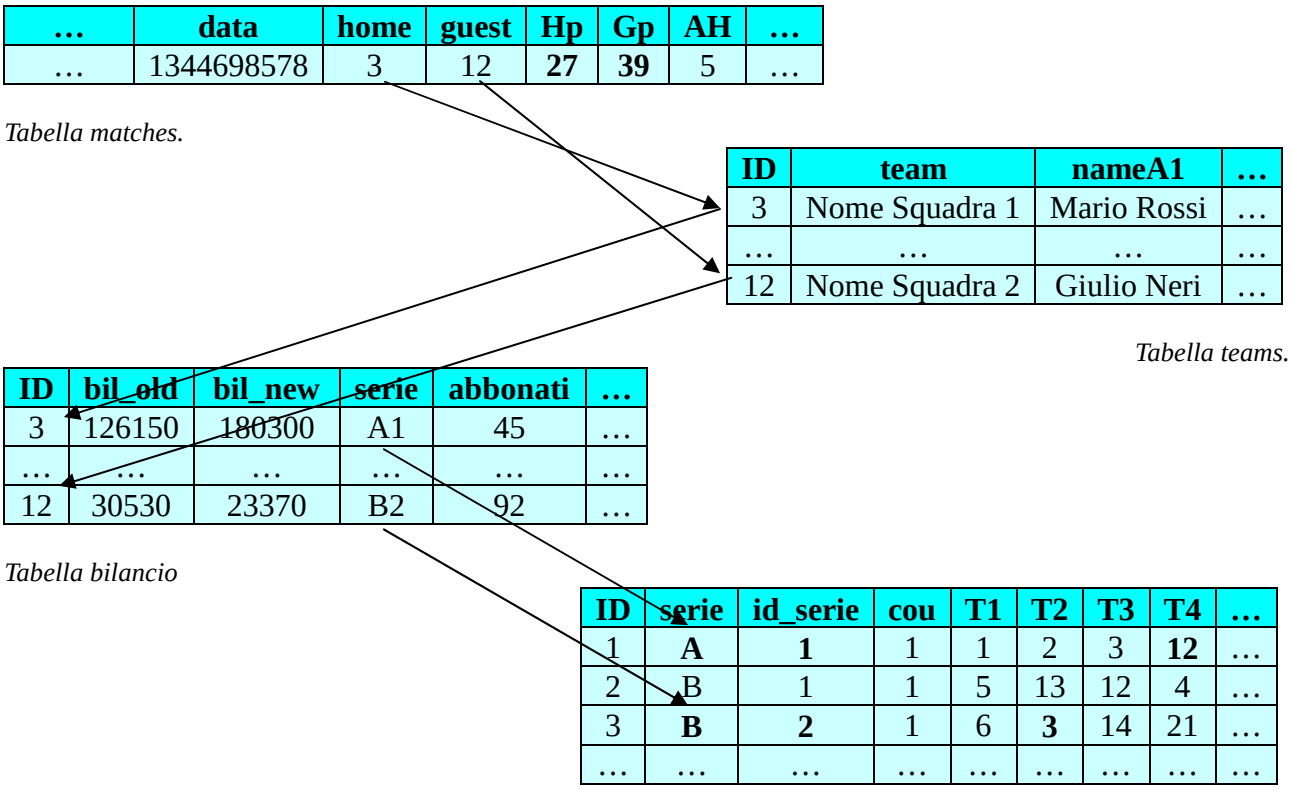
Rappresentazione grafica del ciclo di campionato.

- Operazioni di Gioco:

Le operazioni riguardanti la **riorganizzazione campionati** vengono eseguite quindi ogni 16 settimane. Relativamente al ciclo questa operazione viene effettuata alla fine della prima settimana di pausa, quindi subito aver ottenuto i risultati delle eventuali partite di spareggio promozione/retrocessione. Considerando lo schema essa si colloca al termine della 15° settimana. Lo script composto unicamente da codice PHP, si occupa solo di gestire i dati nel database omettendo completamente qualsiasi output grafico.

L'operazione selezionerà ogni record della tabella *matches* verificando quali di esse si sono svolte nella data coincidente alla domenica della 15° settimana.
Isolati i record vengono esaminati i risultati dei campi "*Hp*" e "*Gp*", a seconda del maggiore dei due valori associati rispettivamente alle squadre "*home*" e "*guest*", viene determinato il vincitore. A questo punto tramite il loro "*ID*" in *teams*, vengono individuate le squadre. Da una ricerca attraverso la tabella *bilancio* vengono calcolate le loro serie di appartenenza, tramite la funzione *substr();* di PHP che isola il primo carattere dai restanti dal campo "*serie*" permettendo così di fissare le basi per effettuare le ricerche nella tabella *campionati*.

Risalendo in questo modo ai campi che indicano le squadre relativamente alle serie di appartenenza, viene evidenziato quale delle due partecipa ad una serie di maggiore livello. Se la squadra che ha vinto compete nella serie di livello maggiore, i record non vengono modificati, in caso contrario i campi "*T**" tra i record della tabella *campionati*, subiscono uno scambio di valore, permettendo così la promozione/retrocessione.



…	data	home	guest	Hp	Gp	AH	…
…	1344698578	3	12	**27**	**39**	5	…

Tabella matches.

ID	team	nameA1	…
3	Nome Squadra 1	Mario Rossi	…
…	…	…	…
12	Nome Squadra 2	Giulio Neri	…

Tabella teams.

ID	bil_old	bil_new	serie	abbonati	…
3	126150	180300	A1	45	…
…	…	…	…	…	…
12	30530	23370	B2	92	…

Tabella bilancio

ID	serie	id_serie	cou	T1	T2	T3	T4	…
1	**A**	**1**	1	1	2	3	**12**	…
2	B	1	1	5	13	12	4	…
3	**B**	**2**	1	6	**3**	14	21	…
…	…	…	…	…	…	…	…	…

Tabella campionati

La routine di **generazione del calendario** viene eseguita al termine della 16° settimana, che coincide con il lunedì (ossia l'inizio) della 1° settimana.
Essa opera relativamente ad ogni record presente nella tabella *campionati*, che colleziona gli "*ID*" delle 8 squadre partecipanti ad esso in una struttura array.
Il seguente codice è costruito sulla logica dell'Algoritmo di Berger, studiato appositamente per comporre gli abbinamenti delle partite di un campionato che preveda i due gironi di andata e ritorno, applicabile ad un numero pari di squadre.

In sostanza esso prevede di dividere in 2 array il numero totale di squadre per poi procedere alla generazione delle partite. Ad ogni ciclo del primo for vengono prodotti gli accoppiamenti per le partite della prima giornata di andata e di ritorno. Al termine del ciclo, gli array subiscono una trasformazione. Ponendo, come nel codice, che gli array vengano inizializzati sulla variabili "*$casa[]*" e "*$trasferta[]*", il primo conterrà i riferimenti alle prime 4 squadre (1 - 2 - 3 - 4), il secondo invece alle ultime 4 (5 - 6 - 7 - 8).
Al fine di garantire una equilibrata e corretta distribuzione dei confronti, le ultime istruzioni eseguite all'interno del ciclo for principale modificano questi array secondo una precisa sequenza:

1) Il valore "*$casa[0]*", inizializza la variabile "*$pivot*";
2) Il contenuto di "*$trasferta[]*" viene spostato a destra di un indice, "*$trasferta[4]*" quindi, che prima non esisteva, acquisisce il valore di "*$trasferta[3]*" ed a "*$trasferta[0]*", che perde il suo valore originario, viene assegnato il valore di "*$casa[1]*";
3) "*$trasferta[4]*" viene troncato da "*$trasferta[]*" per inizializzare la variabile "*$riporto*";
4) Gi elementi di "*$casa[]*" vengono spostati a sinistra di un elemento,perdendo così il valore salvato in "*$casa[0]*" salvato precedentemente in "*$pivot*";
5) Ai valori di "*$casa[]*" viene aggiunto a destra il valore di "*$riporto*" che assumerà quindi la posizione "*$casa[3]*";
6) Infine a "*$casa[0]*" viene sovrascritto il contenuto della variabile "*$pivot*". "*$casa[0]*" prima dello shift era "*$casa[1]*", che è stato trasferito precedentemente in "*$trasferta[0]*".

Ognuno dei sei passi ha un riscontro nel codice seguente.
Conservando così tutti i valori, ma cambiandoli ciclicamente di posizione e array, la generazione standard iniziale del ciclo for principale produce le esatte combinazioni programmando gli accoppiamenti riguardanti le partite da giocare per tutto il campionato.

----- Algoritmo di Berger -----

```
<?php

function AlgoritmoDiBerger($arrSquadre)
 {

    $numero_squadre = count($arrSquadre);
    $giornate = $numero_squadre - 1;

    /* crea gli array per le due liste in casa e fuori */
    for ($i = 0; $i < $numero_squadre /2; $i++)
    {
        $casa[$i] = $arrSquadre[$i];
        $trasferta[$i] = $arrSquadre[$numero_squadre - 1 - $i];
    }

    for ($i = 0; $i < $giornate; $i++)
    {
        /* stampa le partite di questa giornata */
```

```
        echo '<BR>'.($i+1).'a Giornata<BR>';

        /* alterna le partite in casa e fuori */
        if (($i % 2) == 0)
        {
            for ($j = 0; $j < $numero_squadre /2 ; $j++)
            {
                 echo ' '.$trasferta[$j].' - '.$casa[$j].'<BR>';
            }
        }
        else
        {
            for ($j = 0; $j < $numero_squadre /2 ; $j++)
            {
                 echo ' '.$casa[$j].' - '.$trasferta[$j].'<BR>';
            }

        }

        $pivot = $casa[0]; // (1)

        array_unshift($trasferta, $casa[1]); // (2)

        $riporto = array_pop($trasferta); // (3)


        // sposta a sinistra gli elementi di "casa" inserendo all'ultimo
        // posto l'elemento "riporto"

        array_shift($casa); // (4)
        array_push($casa, $riporto); // (5)

        // ripristina l'elemento fisso

        $casa[0] = $pivot ; // (6)
    }
}
?>
```

----- *end* -----

Chiaramente, gli accoppiamenti risultanti da questa porzione di codice, non vengono assolutamente visualizzati come il codice suggerirebbe. Essi, vengono memorizzati in strutture dati e variabili per poi essere inseriti con cadenza settimanale all'interno del database, nella tabella *matches*, attraverso delle query di inserimento. Si occuperà successivamente l'utente ad impostare una ad una le partite a seconda delle sua preferenze attraverso il documento *matchlist.php*.

Le procedure di **riorganizzazione del ranking** si occupano di generare una classifica in base ad un punteggio. Come accennato precedentemente questo è possibile solo aggiungendo un nuovo campo alla tabella *users*. Questo perché esiste una relazione diretta tra l'utente ed il suo punteggio in classifica, ma questa relazione esiste anche tra i record della tabella *users* ed i record delle tabelle *teams* e *bilancio*. Per semplicità quindi, essendo un dato necessario specialmente alla valutazione degli aspetti finanziari il campo relativo verrà aggiunto ai campi della tabella *bilancio*.

Il *ranking* è un semplice valore che esprime una posizione in classifica relativa al numero di utenti presenti nella tabella *users* che viene aggiornato una volta a settimana. Precisamente le operazioni relative a questa procedura vengono eseguite la notte tra la domenica ed il lunedì mattina. Disponibili quindi all'inizio della nuova settimana.

Le formule per il calcolo di questa graduatoria non sono ancora state progettate, è necessario costruirle successivamente ad una sessione di test del sistema completo al fine di regolare i modificatori che contribuiranno ad assegnare i punteggi alle squadre in modo realistico.

Esiste tuttavia un'ipotesi di progetto. Essa prevede che ad ogni squadra sia assegnato di settimana in settimana un punteggio base. Tale punteggio è strettamente dipendente alla serie nella quale la squadra milita. I dati rilevabili dal campo "*serie*" della tabella *bilancio*, forniranno a tutti i club che giocano nella serie maggiore un determinato punteggio base, che cala scendendo di livello, ma si mantiene identico per tutte le squadre della stessa categoria.

A questo punto intervengono i modificatori. Ovviamente quello più incidente è la posizione in classifica nel proprio campionato, ricavabile dall'analisi dei record nella tabella *matchlist*. E in secondo luogo la differenza punti fatti – punti subiti calcolabile nello stesso modo.

Tale livello di precisione dovrebbe essere più che sufficiente a stabilire dei punteggi per ogni utente i cui dati sono presenti nei record della piattaforma, tuttavia ulteriori modificatori possono essere introdotti al fine di rendere la valutazione ancora più accurata.

L'obbiettivo è quello di produrre una classifica omogenea contemplando anche dei casi limite nei quali una squadra di serie inferiore molto promettente, attraverso i modificatori riesca ad eguagliare se non a superare un'altra squadra che milita in una serie superiore, magari sopravvivendo da molte stagioni ad una retrocessione.

Il documento PHP che si occupa di queste operazioni stabilisce di settimana in settimana una graduatoria influenzata del risultato della settimana precedente, memorizzando ogni dato in una struttura array. Questa struttura in seguito ordinata compone la classifica ed i valori ordinali vengono nuovamente memorizzati nel campo "*ranking*" relativo nella tabella *bilancio*.

Settimanalmente, ogni mercoledì notte invece vengono elaborati ed aggiornati i **valori dei progressi relativi all'allenamento**. Essi, come già spigato nel relativo capitolo, sono dipendenti da molte variabili, come il grado di abilità degli allenatori, la percentuale di minuti giocati durante le partite della settimana precedente, il talento e l'età di ogni singolo giocatore, il parametro scelto da allenare settimanalmente ed il tipo di parametro stesso.

Anche in questo caso, è necessaria un'analisi dei risultati di una sessione di test della piattaforma per poter bilanciare al meglio realisticamente la misura degli incrementi da effettuare. Tuttavia il progetto prevede anche che non sia troppo difficile allenare i giocatori, al fine di mantenere l'interesse degli utenti. L'idea è quella di bilanciare le formule in modo da garantire principalmente a seconda del talento del singolo giocatore, un incremento della statistica allenata in un tempo che va da un minimo di 3 settimane ad un massimo di 6.

Risulta difficile a questo punto dimensionare delle formule senza degli effettivi dati alla mano, resta chiaro che i parametri verranno sempre incrementati o decrementati di valori decimali. In questo modo è possibile tenere in considerazione ogni minima variazione in modo trasparente all'utente.
I valori infatti delle caratteristiche memorizzate nei campi dei record della tabella *players*, vengono si generati in formato di numero intero, ma sono poi modificati da valori decimali con la tolleranza del centesimo di incremento/decremento (due cifre dopo la virgola). La visualizzazione invece da documento HTML forzerà l'utilizzo del solo valore intero evidenziando così uno scatto competo in avanti e non i minimi progressi.

Sarà inoltre possibile per i giocatori che hanno raggiunto una certa età, che molti parametri più sensibili, come Resistenza o Riflessi, possano calare sostanzialmente se non allenati. Questo fattore che rende più realistica l'interazione con la piattaforma permette di controbilanciare la facilità di allenamento di questi due particolari parametri costringendo quindi l'utente a prestare attenzione e a non trascurarli del tutto contribuendo così a rendere il gioco più realistico.

Al fine di mantenere sempre rifornite le aste e creare nuovi giocatori da immettere nel campionato, è necessaria una routine di **generazione automatica delle aste** nella tabella *mercato*.
Essa sarà avviata più volte al giorno, a seconda delle necessità del mercato. I giocatori vengono prodotti nello stesso modo in cui essi vengono generati nel documento *newuser.php* analizzato precedentemente. Le uniche differenze sono che la loro squadra di appartenenza sarà un BOT e verranno immessi immediatamente nel mercato aggiornando la lista di record nella tabella *mercato*.
I BOT allo stesso modo vengono generati esattamente come le altre squadre ma andranno memorizzati in una diversa tabella del database al fine di riempire eventuali posti liberi e completare campionati con utenti eventualmente mancanti.
L'esatta frequenza di immissione e il livello delle caratteristiche dei giocatori saranno dipendenti dal livello globale degli elementi già fruibili in asta. E' necessario quindi, per un corretto dimensionamento della procedura studiare dei dati derivanti da una necessaria sessione di test a lavoro completo e pubblicato.

Il progetto prevede inoltre la **generazione del bilancio**. Essa viene ricavata dai dati della tabella *bilancio* già descritta. La cadenza di questo aggiornamento è settimanale (ogni sabato notte prima della partita) e si basa sul consueto valore di *ranking*, campo dei record della stessa tabella. Anche in questo caso si rendono necessari degli studi sul test del sistema per poter bilanciare i valori coinvolti nelle operazioni.

- Operazioni di manutenzione:

Questo tipo di operazioni mantiene ottimizzato il sistema ed evita che i dati obsoleti presenti nel database vi restino per sempre. La procedura di **classificazione degli utenti inattivi e la loro sostituzione**, permette di cancellare dei record non più utilizzati in modo programmato.

Ogni volta che un utente, completa la procedura di registrazione account, nel relativo record, il campo "*time*" indica il momento esatto in cui i suoi dati sono stati memorizzati. E' necessario quindi ampliare la struttura aggiungendo un campo "*log_time*", che tiene traccia del momento in cui è stata affrontata l'ultima procedura di login.
Questo valore, aggiornato ad ogni inserimento delle proprie credenziali nel sistema, può essere utilizzato per verificare da quanto tempo un utente non si connette al proprio account.
Aggiornamento relativamente al quale va prodotta una nuova routine con la quale aggiornare il documento *login.php*.

Ogni notte, quindi con cadenza giornaliera, questo cotrollo deve essere eseguito. Il codice confronterà il tempo attuale con i valore del campo "*log_time*" di ogni record. Se la differenza è maggiore di un determinato tempo valutabile sull'ordine del mese, l'utente viene dichiarato inattivo. Pertanto i campi "*username*" e "*password*" relativi vengono impostati a 0, l'"*ID*" nel rispettivo record nella tabella *campionati*, viene modificato aggiungendo il prefisso "*i*" ed il campo "*inattivi*" incrementato di 1.
In questo modo ogni dato resta inviolato, si nega solo l'accesso al proprietario e si predispone il sistema a riconoscerlo come inattivo. Essi vengono mantenuti per non turbare gli equilibri di un campionato in corso e le squadre di inattive restano nel sistema conservando tutte le ultime impostazioni salvate dall'utente. Confidando quindi nell'abilità dei compagni di serie, esso in qualche stagione dovrebbe retrocedere fino alle serie minori.
A questo punto, ogni nuova iscrizione alla piattaforma ha la possibilità di sovrascrivere la squadra, eliminando definitivamente dalle tabelle *user*, *teams*, *players*, e *bilancio*, i record relativi ai dati obsoleti. Ogni altro campo di tutte le tabelle relative alla piattaforma viene infine modificato inserendo i dati del nuovo utente iscritto.

Record della tabella *players*, che si riferiscono a giocatori troppo vecchi, vengono eliminati attraverso una query di DELETE che costituisce la procedura di **eliminazione giocatori**.
Tale sequenza di istruzioni viene eseguita ogni 16 settimane, al termine della 15°. In questa occasione, vengono passati in rassegna tutti i record della tabella *players*, dei quali viene controllato solo il campo "*Anni*". Se esso supera una determinata soglia (da definire attraverso uno studio dei test), esiste una possibilità progressivamente più alta di stagione in stagione che esso si ritiri e che quindi non sia più utilizzabile. Un numero generato casualmente dal codice decide le sorti del record, se esso rientra in un range prestabilito ma sempre più grande a seconda dell'età, il giocatore viene cancellato dal server.
I giocatori, se licenziati e non messi in vendita, subiscono la stessa sorte qualsiasi età essi abbiano.

5 - Accorgimenti sulla sicurezza web:

Le contromisure relative alla sicurezza, applicate ad un sistema, sono un mezzo importante di salvaguardia dei dati personali che gli utenti hanno permesso al sistema di immagazzinare.
Al fine di proteggerli è necessario quindi utilizzare alcune procedure di controllo sui dati inseriti e sulle interazioni tra gli elementi che compongono l'intera struttura.
I punti deboli di un sistema web sono sempre identificabili in quegli elementi che permettono un inserimento di dati che andranno ad interagire con le strutture dati della piattaforma.

Il web server che offre il servizio di hosting prevede un controllo di base sul sistema che agisce normalmente tramite degli antivirus o un costante monitoraggio sulle richieste che alcuni servizi possono ricevere. I suoi controlli sono molto estesi, il server ospitante infatti può contenere molti altri sistemi al suo interno, divisi a seconda di partizioni private della stessa memoria.

Un classico tipo di attacco, sfrutta i campi di testo di una piattaforma per inserire nella meccanica dei documenti PHP e quindi nel database delle stringhe che in realtà si rivelano codice maligno. Le istruzioni generalmente hanno lo scopo di catturare dei dati e di poterli riutilizzare acquisendo così diritti di amministratore sul sistema attaccato.
Il codice sviluppato nel progetto prevede, durante la gestione di ogni specifico dato, un controllo sui dati inseriti. Semplici validazioni sulla lunghezza minima e massima delle stringhe passate come parametri da ogni form permettono di escludere la maggior parte dei possibili attacchi. In questo modo è impossibile inserire dati composti da più caratteri di quelli contemplati.

Nel caso in cui un qualche tipo di attacco riesca comunque ad insinuarsi nel sistema ed acquisire informazioni relative ai campi username e password, l'utilizzo della conversione MD5, rende il successo dell'impresa quasi inutile. Come analizzato precedentemente infatti, al momento della registrazione utente, il campo "*password*" del relativo record non viene mai salvato utilizzando i caratteri realmente inseriti. Ancora prima di essere trattato, questo dato viene convertito attraverso la funzione di conversione. MD5 indica un algoritmo di codifica asimmetrica. Rende quindi possibile la conversione in un senso partendo da una normale stringa di caratteri, ma non concepisce il processo inverso.
Due stringhe molto simili inoltre, una volta convertite, possono essere completamente diverse. Non è possibile pertanto conoscere la reale password di un utente ricavando il valore del dato memorizzato nel campo "*password*" della tabella *users*. Dato che la procedura di login acquisisce la stringa e la confronta con quella conservato nel database solo in seguito ad un'ulteriore conversione, è necessario conoscere la stringa di partenza per potersi autenticare come quel determinato utente.
La piattaforma inoltre prevede in fase di registrazione che l'inserimento del dato password abbia almeno 9 caratteri di lunghezza.
In questo modo anche tentativi di accesso maligno di tipo *Brute-force* sono destinati a fallire considerato che 9 caratteri alfanumerici costituiscono un numero astronomico di possibili combinazioni. 26 caratteri in minuscolo, 26 in maiuscolo e 10 cifre inserite in 9 caratteri (minimi) costituiscono 62^9 combinazioni diverse.

Un altro importante accorgimento sulla sicurezza è costituito dall'utilizzo del metodo di invio dati "*_POST*" tra diversi script interni al sistema. Esso permette a differenza del contrapposto metodo "*_GET*", che ogni valore passato non sia visibile direttamente nella sintassi dell'URL come accade necessariamente invece per esempio nella richiesta a *auth_key.php* che lo sfrutta al fine di specificare la variabile di attivazione per mezzo di un URL via email.

6 - Elementi caratterizzanti:

La piattaforma è stata realizzata allo scopo di offrire qualcosa di diverso e nuovo al pubblico di utenti che solitamente partecipano a giochi di questo tipo.
Le dinamiche che mettono in relazione più utenti, permettendogli di confrontare le proprie abilità e misurarsi giocando ai classici browsergame, non è una novità. Anzi, questa caratteristica è necessaria a piattaforme di questo tipo.

Il progetto possiede delle determinate caratteristiche che lo fanno emergere rispetto a quelli esistenti. La particolarità più evidente è sicuramente lo sport di cui si occupa. Il Tchoukball è diffuso a livello mondiale, ma non è molto conosciuto. In Italia esistono più di 40 squadre che partecipano al campionato italiano, divise in serie A e B. Esse sono diffuse quasi totalmente al nord, in europa è discretamente giocato ed ogni gruppo di giocatori promuove la diffusione all'interno della propria regione. Nei paesi orientali è giocato maggiormente e per alcune nazioni è addirittura sport nazionale.

La seconda particolarità che offre il browsergame è costituita dalla già descritta tabella *specials*. Essa contiene un elenco di nomi di giocatori realmente esistenti che vengono selezionati per essere inclusi nelle formazioni attraverso la routine di generazione attivata da *newuser.php*. Su richiesta gli interessati possono essere progressivamente aggiunti rendendosi così partecipi.
Questa caratteristica potrebbe portare gli utenti a ricercare dei giocatori "famosi" e quindi a creare una squadra di elementi speciali, fornendo così una variante alla classica conquista del primo posto in classifica.

La principale innovazione però, rispetto ai più conosciuti browsergame è costituita dalla possibilità di visualizzare le partite vere e proprie. Utilizzando la tecnologia Flash, il progetto prevede la costruzione e l'inclusione nel codice di un file formato *.swf*. Come nel caso relativo alle tattiche, esso viene eseguito ed una grande mole di parametri viene generata ed inclusa attraverso le *flashvars*. La serie di istruzioni quindi contenute in *field.swf*, ha l'unico scopo di permettere la visualizzazione del confronto sul campo di due squadre. Il suo sviluppo è ancora incompleto, ma la totalità delle strutture e la maggior parte delle funzioni necessarie è già stata creata. Le istruzioni sviluppate fino ad ora in Actionscript 3.0, costituiscono un documento testuale di 3000 righe circa. L'analisi quindi tratterà solo delle porzioni salienti e caratteristiche dell'intero codice.

Tramite *flashvars*, il codice del documento *matchlist.php*, invia all'oggetto incluso una generosa collezione di informazioni. *field.swf*, le acquisisce attraverso la seguente procedura.
Una delle particolarità di questi confronti sta nel fatto che molti degli eventi che possono accadere in una partita, sono casuali. Essendo però questi oggetti delle procedure prive di contenuto, ogni variabile deve essere concepita esternamente ad essi. Pertanto valori casuali che influenzano il gioco non vengono generati all'interno di *field.swf*.
Un semplice esempio è la statistica Tecnica di un giocatore, essa permetterà di determinare il successo della ricezione di un passaggio per esempio. Un valore relativo alla caratteristica può essere compreso nel range 0 – 9, all'aumentare di questa capacità, diminuisce il rischio di insuccesso. A determinare quindi l'esito dell'azione, sarà una condizione che esamina un indice numerico casuale.
Tra i valori destinati a *flashvars* esistono quindi delle stringhe testuali composte da una generazione esterna di numeri casuali, memorizzati nei campi ausiliari relativi ai record della tabella *matchlist*, riutilizzabili ad ogni visualizzazione di una singola partita.

Primo chiaro esempio del problema appena descritto è costituito dalla variabile "*sorteggio*" che determina il possesso palla iniziale. Non è possibile che ad ogni visualizzazione questo parametro possa cambiare, pertanto, anch'esso memorizzato in *matchlist*, fornisce il primissimo parametro ricevuto da *field.swf*.
Le successive variabili descrivono relativamente ai giocatori "*home*" rispettivamente il nome della squadra, nome della tattica d'attacco utilizzata, nome di quella di difesa, un indice relativo alla ridisposizione in campo e la già citata stinga di caratteri numerici da 0 a 99, che definirà le probabilità relativamente ad ogni evento che necessiti di una produzione di un numero random generato esternamente e memorizzato.
Questa stringa verrà elaborata ed inizializzerà un array esattamente come accade in *tacts.swf*.
Le 14 variabili da "*plha1*" a "*plhd7*" contengono i valori di tattica prodotti da *tacts.swf*.
"*phome*" è una codifica in stringa alfanumerica delle caratteristiche dei 12 giocatori coinvolti.
Da "*p1hn*" a "*p12hn*" sono acquisiti i nomi dei giocatori scelti dall'utente.
Una simmetrica collezione di valori proviene poi dalla seconda parte della procedura, ma relativamente all'utente ospitato.

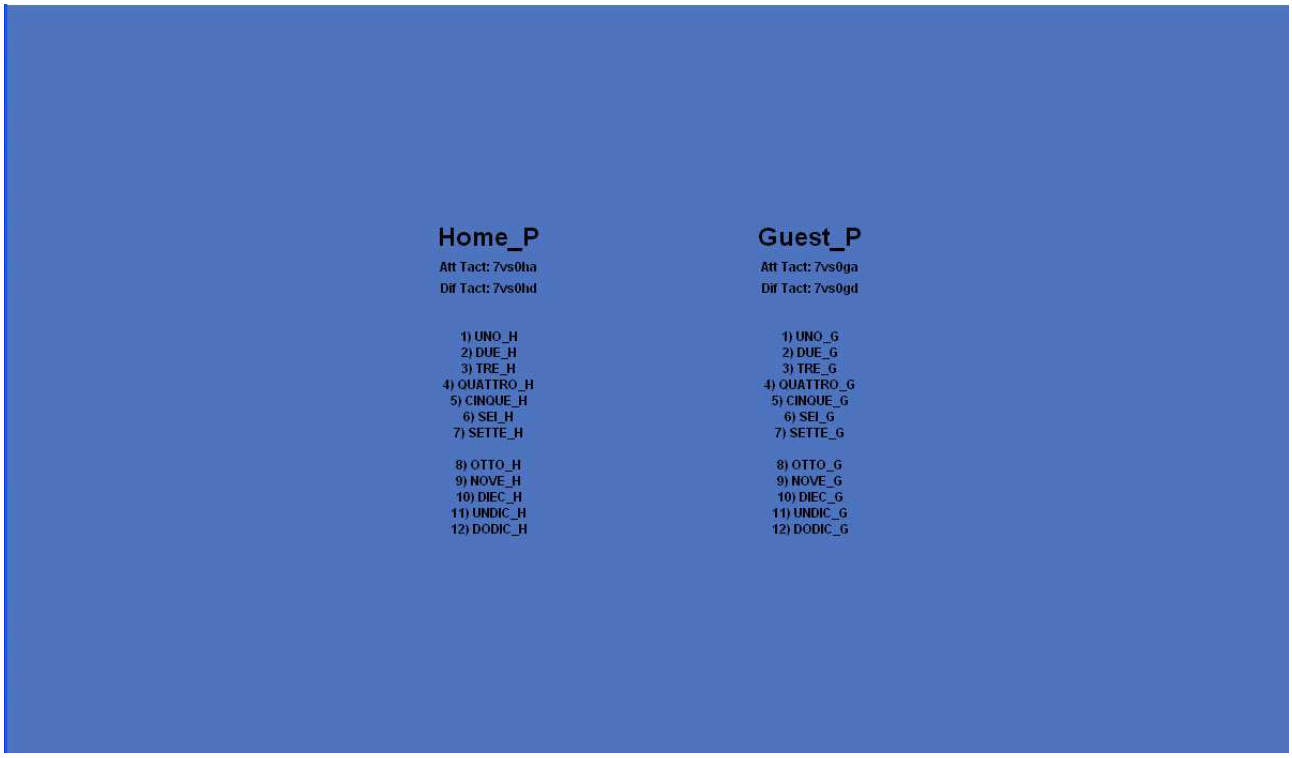
----- field.swf (flashvars) -----

```
sorteggio = root.loaderInfo.parameters.sort;

//Ricezione parametri gioctori home
nTh = root.loaderInfo.parameters.NTh;
nametha = root.loaderInfo.parameters.NAMEThа;
namethd = root.loaderInfo.parameters.NAMEThd;
Hinv = root.loaderInfo.parameters.HINV;
randH = root.loaderInfo.parameters.HRAND;

plha1 = root.loaderInfo.parameters.P1ha;
plha2 = root.loaderInfo.parameters.P2ha;
plha3 = root.loaderInfo.parameters.P3ha;
plha4 = root.loaderInfo.parameters.P4ha;
plha5 = root.loaderInfo.parameters.P5ha;
plha6 = root.loaderInfo.parameters.P6ha;
plha7 = root.loaderInfo.parameters.P7ha;
plhd1 = root.loaderInfo.parameters.P1hd;
plhd2 = root.loaderInfo.parameters.P2hd;
plhd3 = root.loaderInfo.parameters.P3hd;
plhd4 = root.loaderInfo.parameters.P4hd;
plhd5 = root.loaderInfo.parameters.P5hd;
plhd6 = root.loaderInfo.parameters.P6hd;
plhd7 = root.loaderInfo.parameters.P7hd;

phome = root.loaderInfo.parameters.PHOME;
p1hn = root.loaderInfo.parameters.P1hn;
p2hn = root.loaderInfo.parameters.P2hn;
p3hn = root.loaderInfo.parameters.P3hn;
p4hn = root.loaderInfo.parameters.P4hn;
p5hn = root.loaderInfo.parameters.P5hn;
p6hn = root.loaderInfo.parameters.P6hn;
p7hn = root.loaderInfo.parameters.P7hn;
p8hn = root.loaderInfo.parameters.P8hn;
p9hn = root.loaderInfo.parameters.P9hn;
p10hn = root.loaderInfo.parameters.P10hn;
p11hn = root.loaderInfo.parameters.P11hn;
p12hn = root.loaderInfo.parameters.P12hn;

//Ricezione parametri giocatori guest
<...>
```

----- end -----

L’analisi ora procede con la descrizione degli elementi visualizzabili in output attraverso l’esecuzione del file. Il codice successivo a quello appena descritto nel documento infatti risulta dispersivo e puramente ausiliario.
L’oggetto avviato presenta una panoramica sulle impostazioni definite dall’utente in fase di disposizione relativa alla partita. Esso è visualizzabile per una decina di secondi e fornisce in output, i nomi delle squadre, delle tattiche selezionate e, dei giocatori coinvolti.
Ogni campo di testo è visualizzabile grazie alla costruzione di elementi “*TextField*” che assumono un valore impostando l’attributo *text*.

```
var homeName:TextField = new TextField();
homeName.text = nTh;
```

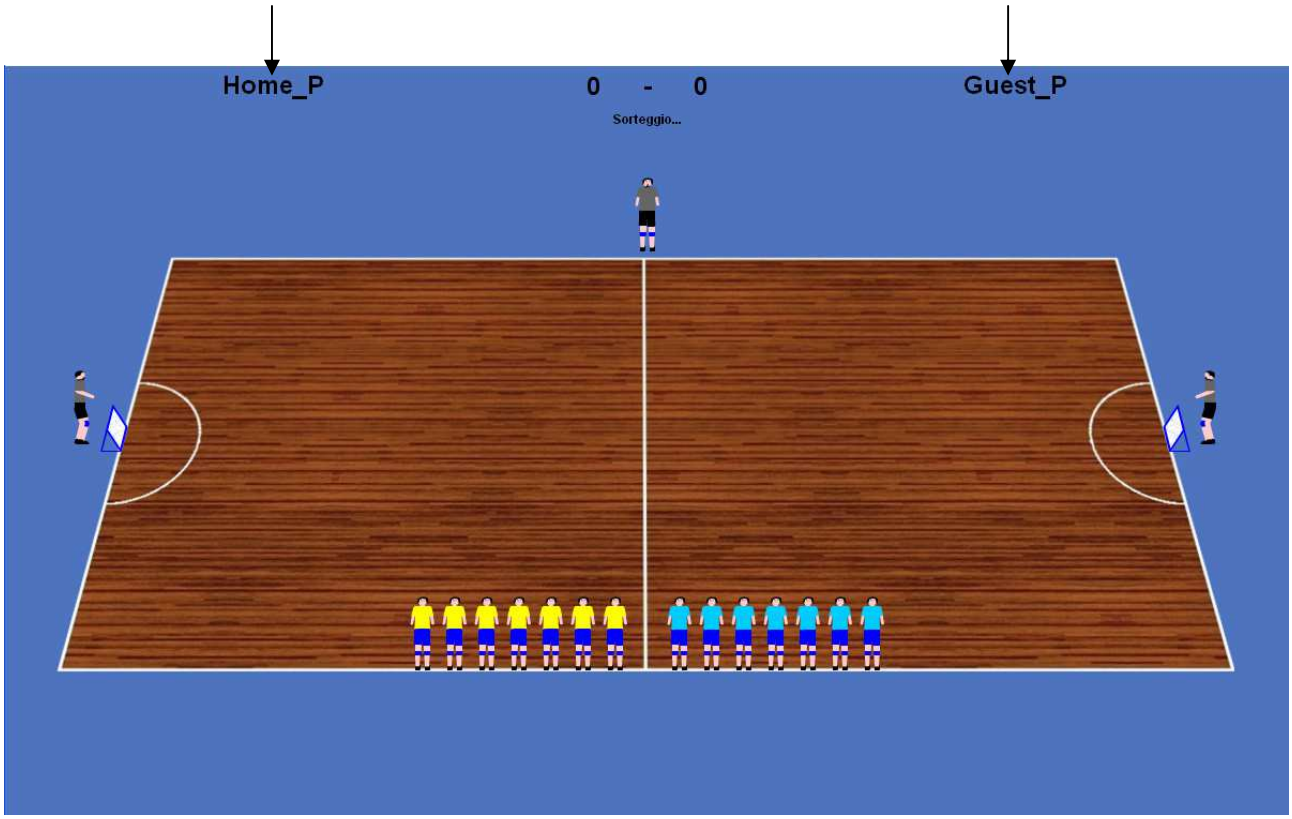


Schermata di avvio relativa alla visualizzazione della partita

Successivamente i dati scompaiono in dissolvenza tramite l’ausilio dell’attributo *alpha*, relativo a qualsiasi elemento. *alpha* è per definizione inizializzato ad 1, costruendo un ciclo che lo decrementi fino a 0 progressivamente, esso scomparirà gradualmente, per lasciare il posto alla visualizzazione costruita al livello sottostante.

```
var back:Shape = new Shape();
      back.graphics.beginFill(0x4e74c0);
      back.graphics.moveTo(0, 0);
      back.graphics.lineTo(1200, 0);
      back.graphics.lineTo(1200, 700);
      back.graphics.lineTo(0, 700);
      back.graphics.lineTo(0, 0);
      back.graphics.endFill();
addChild(back);
ball.alpha = 1;   //impostazione iniziale standard
```

Possiamo notare però che i nomi delle squadre non scompaiono ma si spostano fino alla parte superiore della schermata, costituendo così un elemento funzionale che ha lo scopo di indicare a quale delle due formazioni appartiene ogni lato del campo centrale che mostra il punteggio della partita.



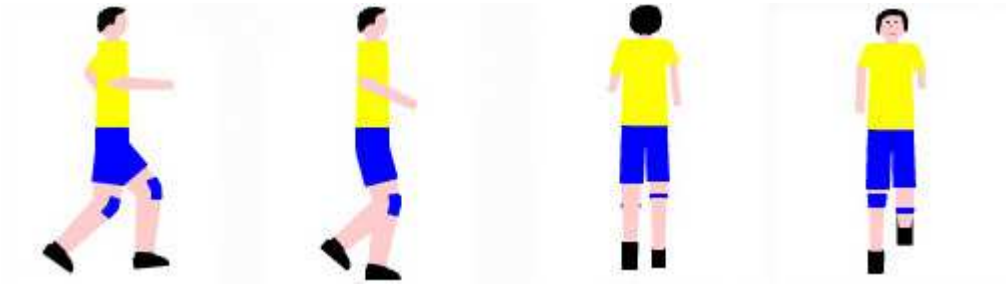
Inizializzazione della partita.

A questo punto analizzeremo brevemente la composizione della struttura che definisce il campo di gioco. Innanzitutto lo sfondo, composto dall'immagine in prospettiva del parquet fornisce la sensazione di avere a che fare con una animazione in 3D. Essa invece è una normale immagine bidimensionale esattamente come ogni elemento della struttura e viene inclusa attraverso una procedura grafica fornita dallo strumento Adobe Flash CS5.

I pannelli posti ai lati del campo, gli arbitri ed i giocatori invece sono a loro volta il risultato di un'elaborazione di file *.swf* più piccoli. La differenza tra di loro è che i file *.swf* che rappresentano arbitri e pannelli sono statici. Non devono quindi muoversi, almeno secondo gli sviluppi previsti dal progetto originario.

Arbitro, file .swf incluso nel codice.

I giocatori invece dovranno attraversare in ogni direzione il campo a seconda delle tattiche, essi dovranno quindi muoversi. Grazie agli strumenti di interpolazione grafica forniti da Adobe Flash, è stato possibile generare delle semplici animazioni cicliche che consentono il movimento, supportato successivamente in campo dal trascinamento. Queste due componenti forniscono all'utente l'idea di corsa in campo.
Le prime immagini seguenti mostrano due diversi frame della stessa animazione nella quale gambe e braccia del giocatore si muovono in modo coordinato. Le ultime due invece si riferiscono alla direzione della corsa.
Il codice riconosce la direzione di movimento del giocatore e cambia il riferimento all'elaborazione *swf* da includere, rendendo reale il movimento sulla superficie del campo.



Frame relativi alla corsa del giocatore descritta dai relativi file .swf.

Ogni animazione è stata sviluppata a 24 frame/sec, la clip quindi si aggiornerà 24 volte ogni secondo. Misura bilanciata a fornire il senso di scorrevolezza all'occhio umano.
Necessariamente quindi anche *field.swf* è impostato a 24 frame/sec, per questioni di compatibilità.

Esistono inoltre ulteriori clip che prevedono ogni possibile movimento dei giocatori, come le difese, il semplice possesso palla ed anche la corsa con palla in mano. L'elemento palla infatti semplicemente disegnato, una volta raggiunto un giocatore scompare, per ricomparire successivamente, agendo sul relativo attributo *alpha*.

Altre disposizioni relative ai giocatori durante le diverse azioni di gioco in campo.

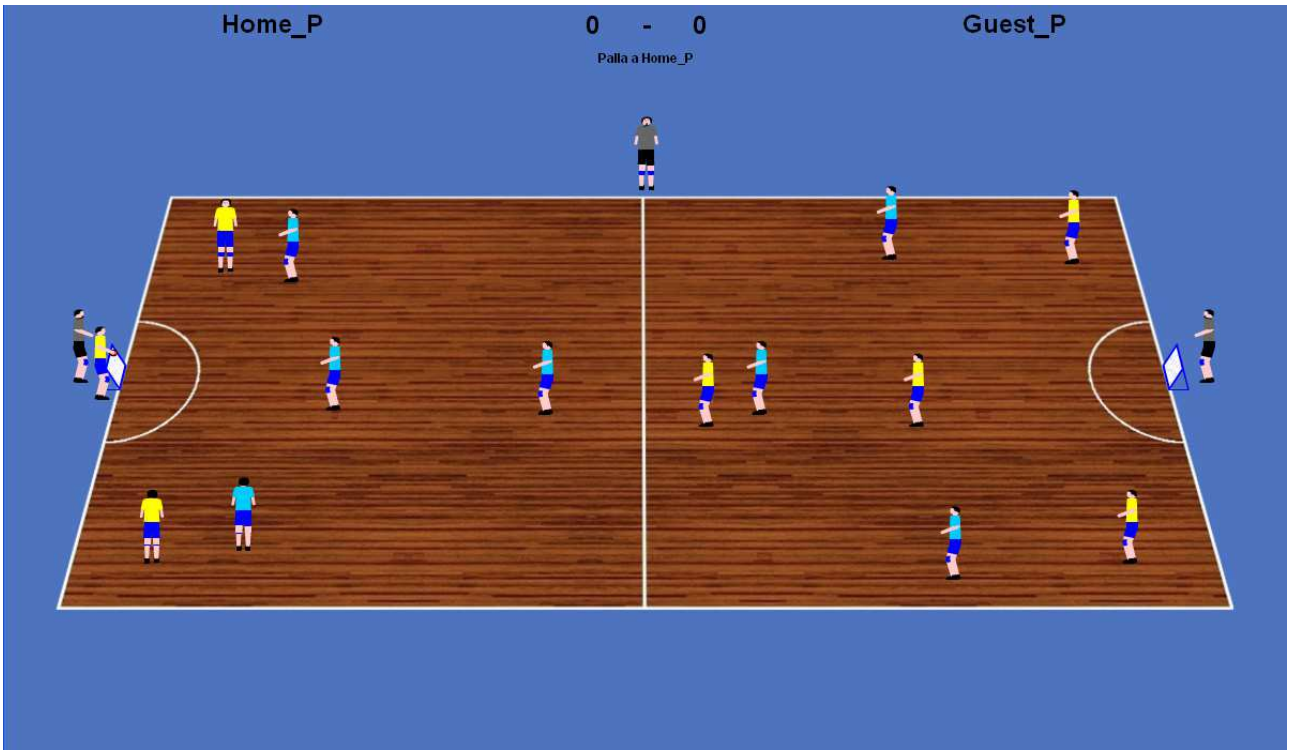
Le due squadre di base differiscono per colore della divisa, giallo per i padroni di casa e azzurro per gli ospiti. Esiste quindi una versione di tutte le azioni per ogni divisa.

Ogni singola azione relativa all'elaborazione, viene scandita da un timer principale. Esso dura per tutta la durata della riproduzione dell'animazione, è costituito dalla seguente struttura.
I parametri passati alla funzione *new Timer()* scandiscono ogni quanti millisecondi deve essere aggiornato il conteggio, e per quante volte. La configurazione specificata, indica che esso deve passare ad un nuovo stato ogni 30 millisecondi per 38000 volte, che sono 19 minuti. Tempo sufficiente a giocare i 15 minuti di partita più inizializzazione e gli intermezzi temporali tra un tempo e l'altro.

```
var movePl:Timer = new Timer(30, 38000); //ogni 30 millisecondi per 19 minuti
(1140 secondi)
      movePl.addEventListener(TimerEvent.TIMER, movePlay);
      movePl.start();
```

Al fine di garantire una buona precisione, i secondi dei tempi di gioco vengono calcolati da altri timer che aggiornano il conteggio ogni 1000 millisecondi, quindi una volta al secondo, incrementando così il valore del campo testo che segna il tempo.

La seguente immagine rappresenta l'inizio delle azioni di gioco. Lo sviluppo di *field.swf* è giunto poco oltre a questo momento creando inoltre la funzione per l'azione di passaggio della palla tra giocatori. Dato che ogni aspetto relativo al direzionamento degli elementi in campo, alla disposizione di essi secondo le tattiche prestabilite, ed alla gestione del cronometro e dei tempi di gioco, lo sviluppo dell'animazione può essere giudicato completo all'80%.
Elementi progettati ma non sviluppati sono le azioni di attacco e difesa, mentre l'aggiornamento dei punti, ed altre funzionalità non direttamente incidenti al risultato dell'animazione sono ormai complete.



Azione di gioco, passaggio iniziale.

Risulta tuttavia meritevole di analisi la procedura di conversione dei dati delle tattiche acquisiti via *flashvars*. Esse infatti, basate sui pixel di un campo largo 240 pixel e alto 440, devono essere adattate e convertite all'utilizzo su una superficie larga 1200 pixel ed alta 700. Considerando che i bordi del campo sono inclinati, si rivela molto interessante il processo di conversione al fine di utilizzarle in partita.

Come già descritto, il dato relativo alla tattica di un determinato giocatore è codificato in forma di stringa numerica formata da 132 caratteri. L'esempio seguente permette di considerare concretamente il dato acquisito dal file. Esso si ripete per 28 volte in tutto, 14 per ogni squadra coinvolta nella partita, 7 relative all'attacco e 7 alla difesa.

```
plha1 = "039030041032040030040032042031044032038036043031040037040036040034042051042049043055043059043063043054041061038065038060042066042066";

<...>       //altri parametri per un totale di 28
```

Esso, attraverso la procedura già descritta di assegnazione dei parametri ai diversi array, sfrutta le strutture qui di seguito inizializzate per decodificare la stringa iniziale in due array relativi ad ascisse ed ordinate. Ognuna delle 28 stringhe viene quindi decodificata generando in tutto 56 sottostrutture.

```
var pl1xha:Array = new Array;
var pl1yha:Array = new Array;

<...>       //strutture array per tutti gli altri 27 elementi in campo
```

A questo punto è necessario affrontare il problema della compatibilità tra i diversi formati delle tattiche. I riferimenti tra *tacts.swf* e *field.swf* sono invertiti e le dimensioni sostanziosamente ingrandite.
Questo primo ciclo for popola gli array descritti invertendo i valori relativi all'asse X ed Y, applicando ad X però delle modifiche sostanziali. Il valore viene sottratto all'indice fisso di 240 ed associato all'array di Y. Questo compie la conversione delle coordinate disposte in un sistema verticale ad uno ruotato di 90° a sinistra.
Il ciclo applica la conversione ad ogni elemento acquisito da *flashvars*.

```
for (zf=0; zf<22; zf++) {

      pl1yha[zf] = 240 - Number(plha1.substring(benS, endS));
      pl1xha[zf] = Number(plha1.substring(benS+3, endS+3));

      <...>       //routine applicata a tutti i valori della stessa tattica

      benS = benS + 6;
      endS = endS + 6;
}

<...>       //routine applicata a tutte le restanti 3 tattiche
```

Una volta disposti correttamente i coefficienti, è necessario adattarli alla forma e alla dimensione del campo relativo a *field.swf*.
Lo sfondo del campo ha le dimensioni di 1200 pixel di larghezza per 700 di altezza. Le misure del campo di gioco per quanto riguarda l'asse Y vanno da 179 a 565 pixel (386 pixel in tutto). Confrontare i valori dal bordo superiore del campo da gioco significa porli nelle stesse condizioni iniziali, la coordinata di base quindi viene decrementata di 20 pixel che identifica lo spessore del bordo esterno relativamente al file *tacts.swf*.
La proporzione di seguito descritta chiarisce la procedura:

$nY_F : 386 = (Y_T – 20) : 200$ quindi $nY_F = (Y_T – 20) * 386 / 200 = (Y_T – 20) * 1,93$

dove Y_T = valore di Y relativo alle tattiche in orizzontale, 200 = altezza del campo delle tattiche ruotato di 90° a sinistra e nY_F è l'incognita relativa alla dimensione riferita al reale campo da gioco. Una volta individuato il valore, è necessario solo sommarlo a 179, che è il valore in pixel che stabilisce il limite superiore del campo di *field.swf*.

La conversione dell'asse X è ben più complicata. I bordi esterni infatti sono individuabili attraverso delle funzioni dipendenti dalla quota Y. I limiti imposti ad essa sono quelli definiti precedentemente. Relativamente all'asse X invece, il bordo superiore è tracciato dalla quota X di 155 a 1038 e quello inferiore da 49 a 1149 pixel. Il campo quindi è definito in una forma trapezoidale racchiusa nei punti di coordinate A(155, 179), B(1038, 179), C(1149, 565) e D(49, 565). Computando i dati relativi alle coppie di punti A-D e B-C, è possibile determinare delle funzioni che rappresentino le rette passanti. Le rette che determinano il valore di X attraverso il parametro Y sono:

A-D: X = (Y – 743,2/(-3,64)
B-C: X = (Y + 3433,24)/3,48

Una volta determinato se il valore X_T proveniente dalle tattiche è <= 220 o meno, è possibile, utilizzando il relativo valore (nY_F + 179) determinare quale delle due formule usare apportando le relative modifiche dovute alla posizione destra o sinistra rispetto alla metà del campo in cui si opera determinata dal valore X di 597 (metà campo in *field.swf*).
Convertiti così i dati delle tattiche in modo proporzionale al campo in cui si devono applicare, è possibile utilizzarle liberamente nel gioco.

```
for (zf=0; zf<22; zf++) {

      //HOME - ATTACCO
      pl1yha[zf] = ((pl1yha[zf] - 20) * 1.93) + 179;
      if (pl1xha[zf] <= 220) {
            pl1xha[zf] = 597 - (((597 - ((pl1yha[zf] - 743.2)/(-1 * 3.64))) *
(220 - pl1xha[zf]) )/200);
      } else {
            pl1xha[zf] = (((((pl1yha[zf] + 3433.24) / 3.48) - 597) * (pl1xha[zf]
- 220) )/200) + 597;
      }

      <...>       //routine applicata a tutti i valori della stessa tattica

}

<...>       //routine applicata a tutte le restanti 3 tattiche
```

Allo stesso modo, sono state computate complesse formule che definiscono i limiti di ogni settore in cui la palla deve agire al fine di muovere i giocatori. Il principio di funzionamento è però identico, solo applicato più volte al fine di definire molti più limiti nel terreno di gioco contenuto in *field.swf*.

Esistono altre procedure degne di nota, ma essendo strettamente dipendenti dalla conclusione della fase di sviluppo, sono ancora incomplete, come la gestione dei livelli degli oggetti nel campo al fine di non farli sovrapporre in modo innaturale oppure l'effettiva gestione dei movimenti tra i settori dell'oggetto palla, che riprende gli algoritmi già descritti durante l'analisi di *tacts.swf*.

7 - Sviluppi futuri:

A questo punto tutti gli elementi progettati sono stati analizzati adeguatamente. Nel complesso ogni componente interagisce con gli altri fornendo all'utente un'esperienza di gioco completa sia dal punto di vista ludico che relativamente all'ottimizzazione ciclica del sistema.
Dopo la pubblicazione finale della piattaforma in rete, affidata ad un web server che offre il servizio di hosting, è possibile studiare ogni variazione del sistema al fine di ottimizzarlo in modo adeguato.

Quando il sistema verrà ritenuto stabile e non avrà bisogno di ulteriori modifiche è possibile applicare ad esso delle nuove funzionalità che vanno oltre gli obbiettivi determinati dalla fase di progettazione iniziale.

Il seguente elenco fornisce una panoramica che descrive le idee di possibile realizzazione al fine di espandere il gioco in futuro, dandogli maggiore variabilità di interfacciamento con le scelte dell'utente.

- Lo sviluppo futuro più importante riguarda la **gestione di account plus**. La creazione di questo tipo di account permette a ogni utente, che paghi una quota mensile, di accedere a funzionalità nuove e alla personalizzazione dei propri dati di gioco. Assegnazione di una divisa ai giocatori, pubblicazione di una bandiera o un simbolo del club, visualizzazione di statistiche approfondite relativamente ai parametri di crescita dei giocatori, compongono alcune delle funzionalità dell'upgrade. Mettere a frutto questa idea però fa parte di un lavoro complesso. Ogni elemento del sito sviluppato dovrà essere modificata, o per meglio dire, di ogni parte di esso dovrà essere creata una versione alternativa che dovrà sfruttare contemporaneamente nuove e vecchie strutture dati. L'importante è che tutto ciò non influisca sul rendimento della squadra o dei giocatori in modo diretto, ogni strumento messo a disposizione all'utente sarà solo un aiuto alla gestione del club. Questo è lo sviluppo futuro più importante, essendo la principale e più diretta fonte di reddito della piattaforma.

- Le partite di campionato hanno cadenza settimanale, vengono giocate ogni domenica. Il giovedì è invece possibile organizzare delle amichevoli.
Lo sviluppo che si vuole realizzare tratta dell'**organizzazione di tornei** tra gruppi di utenti. Essi potranno attribuirgli un nome e giocare le relative partite il giovedì. Questo sviluppo necessiterà di un nuovo documento PHP e di una struttura dati che collezioni i record relativi ad ogni torneo organizzato. Il calcolo delle partite da giocare verrà affidato alla stessa procedura descritta relativamente alle *operazioni temporizzate* che gestiscono le partite di campionato, ma eseguita a discrezione degli utenti.

- Durante l'analisi del capitolo relativo alle routine di allenamento, un ruolo fondamentale alla generazione degli incrementi applicati alle statistiche dei singoli giocatori è affidato agli allenatori. Nella progettazione è previsto che essi siano degli elementi standardizzati in base ad un unico giudizio di abilità. L'idea di questo sviluppo è legata alla suddivisione delle loro capacità, generando infatti un **mercato di allenatori**, sarà possibile comprarli attraverso delle aste esattamente come accade per gli elementi di *players*. Creando una routine di produzione allenatori sarà inoltre possibile suddividere loro le abilità per ogni parametro e definire dalla somma delle statistiche un giudizio globale che ne determinerà il prezzo di ingaggio, lo stipendio e, a seconda delle offerte degli utenti, un prezzo di asta variabile.

- Un ulteriore sviluppo di per sé molto semplice è costituito da una variante sull'allenamento. A seconda dello studio sui risultati del periodo iniziale di test, sarà possibile verificare se il dimensionamento dei progressi attivati dall'allenamento è corretto. Se esso risulta troppo favorevole all'utente, sarà aggiunta alla piattaforma una nuova variabile, l'**allenamento di ruoli specifici**. Con essa, oltre a dover impostare la caratteristica da allenare settimanalmente, sarà necessario scegliere anche il ruolo da allenare (Ala, Centrale, ecc…). Questa variante porterà a concentrare i miglioramenti solo su alcuni elementi della squadra, che comunque si rifletterà in parte minore anche sugli altri.

- La generazione automatica dei giocatori per il mercato potrebbe risultare talvolta insufficiente a dare abbastanza ricambio ai giocatori assegnati alle squadre. Al fine di aumentare il flusso di nuovi giovani tra le proprie fila, il sistema e quindi ogni squadra, verranno dotati della possibilità di coltivare nuovi talenti all'interno del proprio club attraverso **le giovanili**. Investendo una quota settimanale infatti i club potranno allenare dei giovani che una volta maturi possono essere venduti o utilizzati a seconda delle necessità dell'utente.

- L'ultima idea è caratterizzata da uno sviluppo a lungo termine. Se il gioco, riesce ad avere successo su larga scala, con la creazione quindi di campionati e leghe estere, sarà possibile costruire una struttura che permetta di organizzare e giocare dei **campionati mondiali** o europei.
Attraverso uno strumento per la comunicazione, **affidandosi ad un forum esterno o sviluppandone uno interno al sito che ospita la piattaforma**, possono essere eletti gli utenti più promettenti e selezionati i giocatori migliori relativamente ad ogni nazione contemplata dalla piattaforma.
Gli utenti eletti come commissari tecnici delle diverse nazionali avranno accesso a due gestioni separate al fine di non perdere il controllo del proprio club originario e gestire al meglio la propria squadra nazionale durante il calendario della manifestazione.

8 - Conclusioni:

In conclusione, l'analisi che ha descritto la piattaforma fino ad ora ha evidenziato che il progetto per la creazione di un browsergame di tchoukball è completo e permette al termine dello sviluppo di base, di ottenere una piattaforma di gioco funzionante.
Lo sviluppo vero e proprio tuttavia non è ancora stato ultimato, la struttura infatti, formata da documenti PHP che coinvolgono un server SQL attraverso l'utilizzo di tecnologie ausiliarie, non è stata prodotta in tutte le sue parti.

Nonostante la necessità di creare ulteriori elementi, la porzione fino ad ora creata permette di fornire una chiara idea del progetto sviluppato interamente. Le strutture principali di gestione di utenti, squadre, giocatori e tattiche forniscono il minimo livello utile alla comprensione dell'obbiettivo del lavoro in sviluppo. Gli elementi mancanti come il bilancio, l'allenamento ed il mercato, sono parte integrante della progettazione iniziale, ma non costituiscono una grave mancanza allo scorrevole funzionamento preteso dal gioco.
Elemento fondamentale invece, il codice che descrive l'organizzazione delle partite e le relative operazioni temporizzate. Di vitale importanza al ciclo imposto dalla temporizzazione del campionato e proprio ora in fase di sviluppo.

Data la complessità dell'intera struttura descritta, è possibile attribuire una percentuale di completamento della fase di sviluppo del 75% circa. Gli elementi di bilancio, mercato e allenamento infatti sono semplici confrontandoli al codice generato fino ad ora.

Una volta che l'intero sistema sarà stato ultimato, ne seguirà una fase di test alla quale avranno accesso un numero limitato di utenti. Questa procedura permetterà di collezionare preziosi indici sul bilanciamento dei contenuti che serviranno a modificare determinati aspetti al fine di rendere il browsergame il più realistico possibile, e di fornire agli utenti delle sfide governate da dei fattori che bilancino adeguatamente la difficoltà di gioco.
A questa fase seguirà la vera e propria pubblicazione, depositando i documenti PHP, le strutture dati e le impostazioni relative agli strumenti utilizzati dal sistema, nella memoria di un web server. Esso attraverso l'assegnazione di un URL standard, permetterà di accedere al browsergame e di giocare virtualmente a tchoukball.

Bibliografia:

[1] - http://www.wikipedia.org/

[2] - http://www.php.net/

[3] - http://www.w3schools.com/sql/

[4] - http://reference.sitepoint.com/css

[5] - http://www.javascriptkit.com/jsref/

[6] - http://help.adobe.com/it_IT/FlashPlatform/reference/actionscript/3/