ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

SOCCERFUN: ΑΝΑΠΤΥΞΗ ΔΥΝΑΜΙΚΗΣ ONLINE ΕΦΑΡΜΟΓΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Ν. Καραχρήστος

Επιβλέπων: Ιωάννης Βασιλείου, Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

SOCCERFUN: ΑΝΑΠΤΥΞΗ ΔΥΝΑΜΙΚΗΣ ONLINE ΕΦΑΡΜΟΓΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Ν. Καραχρήστος

Επιβλέπων: Ιωάννης Βασιλείου, Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 2η Νοεμβρίου 2009:

.....	.....	.....
Ιωάννης Βασιλείου	Τιμολέων Σελλής	Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2009

……..
Ιωάννης Καραχρήστος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη, ο σχεδιασμός και η υλοποίηση μιας δυναμικής online εφαρμογής. Η εφαρμογή, η οποία διαθέτει το όνομα Soccerfun, αποτελεί μια πλατφόρμα υποστήριξης παιχνιδιών σχετικών με το ποδόσφαιρο. Οι χρήστες της εφαρμογής συμμετέχουν σε μια σειρά από υπο-παιχνίδια, στα οποία συναγωνίζονται ως προς την ικανότητά τους να προβλέπουν αποτελέσματα αγώνων και πορείες ομάδων κατά τη διάρκεια μιας διοργάνωσης, όπως για παράδειγμα το UEFA Champions League.

Η παρούσα αναφορά αναλύει τη διαδικασία που ακολουθήθηκε, τόσο κατά τη φάση της ανάπτυξης της εφαρμογής γύρω από το σύστημα βάσεων δεδομένων MySQL με χρήση της γλώσσας PHP, όσο και κατά τη φάση της σχεδίασης των ιστοσελίδων με βάση τα πρότυπα web standards.

Keywords: προγραμματισμός, online, εφαρμογή, internet, βάση δεδομένων, MySQL, PHP, web standards

Abstract

The purpose of this diploma project is the design and development of a dynamic online application. The application, named Soccerfun, is a platform built to support various interactive games related to soccer. The users of Soccerfun participate in a series of sub-games, in which they compete with each other, while trying to predict the outcome of individual games and the performance of teams during a soccer event, such as the UEFA Champions League.

This report details the process that was followed during the development of the application, based on the relational database system MySQL and the PHP programming language, as well as the design concepts of the various sections of the site, which are based on the use of web standards.

Keywords: programming, online application, internet, database, MySQL, PHP, web standards

Soccerfun: Ανάπτυξη δυναμικής online εφαρμογής

Ιωάννης Καραχρήστος

Αθήνα, 2 Νοεμβρίου 2009

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Αποσπάσματα κώδικα

Κεφάλαιο 1

Εισαγωγή

> "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."
>
> *Tim Berners-Lee*

Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη ενός δυναμικού site με χρήση τεχνολογιών PHP, MySQL και Web Standards. Το site θα αποτελεί μια πλατφόρμα υποστήριξης παιχνιδιών. Για τους σκοπούς της παρούσας διπλωματικής εργασίας, το παιχνίδι με το οποίο θα ασχοληθούμε εκτυλίσσεται κατά τη διάρκεια μιας διοργάνωσης ποδοσφαίρου, όπως για παράδειγμα το Champions League.

Οι χρήστες του site προβλέπουν αποτελέσματα αγώνων και πορείες ομάδων συνολικά στη διάρκεια της διοργάνωσης και ανάλογα με την επιτυχία των προβλέψεών τους συγκεντρώνουν πόντους. Στο τέλος της διοργάνωσης, ο χρήστης με τους περισσότερους πόντους ανακηρύσσεται νικητής.

1.1 Δομή του κειμένου

Η παρούσα αναφορά αποτελείται από 6 κεφάλαια στα οποία παρουσιάζονται διεξοδικά όλες οι φάσεις ανάλυσης, σχεδίασης και υλοποίησης της εφαρμογής. Πιο συγκεκριμένα, η δομή του κειμένου είναι η εξής:

Εισαγωγή Το παρόν κεφάλαιο αποτελεί μια εισαγωγή στην εφαρμογή και περιλαμβάνει τη δομή του κειμένου, μια αναφορά στα εργαλεία που χρησιμοποιήθηκαν και τις ευχαριστίες του γράφοντος.

Παρουσίαση Στη συνέχεια, η εφαρμογή παρουσιάζεται πιο αναλυτικά και αναφέρονται οι κανόνες του παιχνιδιού, οι λειτουργικές απαιτήσεις του διαχειριστή (administrator) καθώς επίσης οι λειτουργικές απαιτήσεις του απλού χρήστη του site.

Ανάπτυξη Στο κεφάλαιο αυτό θα ασχοληθούμε με τη διαδικασία και το περιβάλλον ανάπτυξης της εφαρμογής. Πιο συγκεκριμένα, θα παρουσιάσουμε το σύστημα βάσεων δεδομένων MySQL και θα αναλύσουμε τη δομή της βάσης που υποστηρίζει την εφαρμογή. Στη συνέχεια, θα αναφερθούμε στη γλώσσα προγραμματισμού PHP, στην στενή σχέση που έχει με το σύστημα MySQL και στη διαδικασία κατασκευής των διαφόρων τμημάτων της εφαρμογής.

Σχεδίαση Στο κεφάλαιο αυτό θα περάσουμε στη διαδικασία σχεδίασης της εφαρμογής και θα ασχοληθούμε με το εικαστικό κομμάτι της εργασίας. Αφού κάνουμε μια αναφορά στα web standards και στη σημασία τους στην ανάπτυξη σύγχρονων και λειτουργικών web εφαρμογών, στη συνέχεια θα ασχοληθούμε πιο διεξοδικά με την περιγραφική γλώσσα XHTML καθώς επίσης με τη γλώσσα CSS η οποία χρησιμοποιείται για τον καθορισμό της εμφάνισης των ιστοσελίδων.

Σύνθεση Στη συνέχεια θα περάσουμε στη σύνθεση της εφαρμογής και θα αναλύσουμε τον τρόπο με τον οποίο οι διάφορες τεχνολογίες που αναφέρθηκαν παραπάνω συνεργάζονται για να παράγουν μια ολοκληρωμένη εφαρμογή. Θα εξετάσουμε την εφαρμογή τόσο από την πλευρά του διαχειριστή (administrator), όσο και από την πλευρά του απλού χρήστη και θα αναφέρουμε διάφορα παραδείγματα και σενάρια χρήσης. Τέλος, θα αναφερθούμε σε ζητήματα ασφαλείας μιας online εφαρμογής και σε τρόπους με τους οποίους μπορούμε να την προστατέψουμε.

Επίλογος Στο τελευταίο κεφάλαιο της αναφοράς θα αναφέρουμε σκέψεις και συμπεράσματα τα οποία προέκυψαν κατά την ανάπτυξη της εφαρμογής και θα παρουσιάσουμε τις πιθανές μελλοντικές επεκτάσεις που θα μπορούσαν να προστεθούν.

1.2 Τεχνολογίες ανοιχτού κώδικα

Πριν προχωρήσουμε στην παρουσίαση και ανάλυση της εφαρμογής, θα θέλαμε να αναφερθούμε στις τεχνολογίες ανοιχτού κώδικα και στη σημασία τους. Όλες οι τεχνολογίες στις οποίες βασίζεται η εφαρμογή που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας είναι τεχνολογίες ανοιχτού κώδικα. Τα σημαντικότερα πλεονεκτήματα της επιλογής αυτής συνοψίζονται στα ακόλουθα σημεία:

- Όπως υποδηλώνεται και από το όνομά τους, οι τεχνολογίες ανοιχτού κώδικα διαθέτουν στον προγραμματιστή τον πλήρη κώδικα των λειτουργιών τους. Έτσι, ο χρήστης κάποιου open source προγράμματος ή εργαλείου μπορεί να τροποποιήσει τον τρόπο που επιτελείται κάποια από τις λειτουργίες, σύμφωνα με τις απαιτήσεις του.

- Ακριβώς λόγω της διαθεσιμότητας του κώδικα, στις περισσότερες των περιπτώσεων, αναπτύσσονται από χρήστες προσθήκες ή επεκτάσεις σε ένα υπάρχον εργαλείο, τις οποίες κάθε χρήστης μπορεί να ενσωματώσει και να επωφεληθεί από τη λειτουργικότητα που αυτές προσφέρουν.

- Το open source ευνοεί την ανάπτυξη κοινοτήτων χρηστών, οι οποίοι συμμετέχουν ενεργά στην εξέλιξη των εργαλείων και τεχνολογιών γενικότερα. Έτσι, τα σφάλματα στον κώδικα ανακαλύπτονται γρηγορότερα και η διαδικασία διόρθωσής τους επιταχύνεται σημαντικά.

- Οι τεχνολογίες ανοιχτού κώδικα, συνήθως, δεν απαιτούν την ύπαρξη κάποιας άδειας χρήσης και είναι διαθέσιμες δωρεάν. Αυτό έχει σαν αποτέλεσμα να δημιουργούνται μεγάλες βάσεις χρηστών. Από τις εμπειρίες των χρηστών αυτών σχετικά με τη χρήση ενός προγράμματος ή τεχνολογίας, δημιουργείται μια γενικότερη γνωσιακή βάση (knowledge base), στην οποία κάθε χρήστης μπορεί να ανατρέξει και να αναζητήση λύση σε κάποιο πρόβλημα που έχει.

Σχεδόν σε όλες τις φάσεις σχεδιασμού, ανάπτυξης και υλοποίησης της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκαν τεχνολογίες ανοιχτού κώδικα. Στις επόμενες ενότητες θα αναφερθούμε αναλυτικά στις βασικότερες από αυτές (MySQL, PHP, Web Standards), ενώ στον επίλογο της παρούσας αναφοράς θα κάνουμε μια σύντομη παρουσίαση στα βασικότερα εργαλεία που χρησιμοποιήθηκαν.

1.3 Ευχαριστίες

Θα ήθελα να ευχαριστήσω προσωπικά τον καθηγητή κ. Ιώαννη Βασιλείου τόσο για την επίβλεψή του και τις συμβουλές του κατά την εκπόνηση της παρούσας διπλωματικής εργασίας όσο και για τη γνωριμία με το χώρο των βάσεων δεδομένων και των πληροφοριακών συστημάτων μέσω της διδασκαλίας των αντίστοιχων μαθημάτων.

Επιπλέον, θα ήθελα να ευχαριστήσω τους συμφοιτητές μου για τις συζητήσεις, την ανταλλαγή απόψεων και την παρέα κατά τη διάρκεια του ακαδημαϊκού έτους 2008-2009.

Τέλος, θα ήθελα να αφιερώσω την παρούσα αναφορά στους δικούς μου ανθρώπους για την κατανόηση και την αμέριστη συμπαράστασή τους στη διάρκεια των σπουδών μου και ιδιαιτέρως κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας.

Κεφάλαιο 2

Παρουσίαση της εφαρμογής

Στο κεφάλαιο αυτό θα αναλύσουμε την εφαρμογή διεξοδικά και θα εξηγήσουμε τους κανόνες του παιχνιδιού. Στη συνέχεια θα προσδιορίσουμε τις λειτουργικές απαιτήσεις τόσο για το διαχειριστή του site, όσο και για τον απλό χρήστη.

2.1 Οι κανόνες του παιχνιδιού

Το Soccerfun είναι είναι ένα παιχνίδι σχετικό με το ποδόσφαιρο, το οποίο μετράει ήδη 10 χρόνια ζωής. Δημιουργήθηκε από μια κοινότητα ανθρώπων με στόχο να αποτελεί ένα διαγωνισμό μεταξύ των συμμετεχόντων ως προς τη δυνατότητά τους να προβλέψουν αποτελέσματα αγώνων και πορείες ομάδων στη διάρκεια μιας διοργάνωσης ποδοσφαίρου. Το παιχνίδι ουσιαστικά αποτελείται από 4 υποπαιχνίδια, τα οποία εξηγούμε στη συνέχεια. Για την οικονομία της συζήτησης θα υποθέσουμε ότι η διοργάνωση αυτή είναι το Champions League, καθώς η διαδικασία είναι παρεμφερής και για άλλες διοργανώσεις όπως το World Cup ή το Euro.

2.1.1 Pred: Πρόβλεψη αγώνων

Στην τελική φάση του Champions League συμμετέχουν 32 ομάδες, οι οποίες είναι χωρισμένες σε 8 ομίλους των 4. Στη φάση των ομίλων οι ομάδες του κάθε ομίλου συμμετέχουν σε ένα πρωτάθλημα 6 αγωνιστικών και αντιμετωπίζουν τις υπόλοιπες ομάδες του ίδιου ομίλου, τόσο εντός όσο και εκτός έδρας. Παράδειγμα της κατάταξης ενός ομίλου παρατίθεται στον Πίνακα 2.1.

Οι χρήστες καλούνται να προβλέψουν το αποτέλεσμα κάθε αγώνα της φάσης των ομίλων, επιλέγοντας ανάμεσα σε 3 πιθανά ενδεχόμενα: νίκη γηπεδούχου ομάδας, ισοπαλία ή νίκη φιλοξενούμενης ομάδας. Για κάθε πετυχημένη πρόβλεψη ο χρήστης ανταμείβεται με 10 βαθμούς και στο τέλος της φάσης των ομίλων ο χρήστης με τους περισσότερους βαθμούς κερδίζει το παιχνίδι. Σημειώνουμε ότι, καθώς ο αριθμός των αγώνων είναι 96 (6 αγωνιστικές των 2 αγώνων για 8 ομί-

Team	Points	W	T	L	GF	GA
Barcelona	13	4	1	1	18	8
Sporting	12	4	0	2	8	8
Shakhtar Donetsk	9	3	0	3	11	7
Basel	1	0	1	5	2	16

Πίνακας 2.1: Παράδειγμα βαθμολογίας ομίλου

λους), ο μέγιστος αριθμός βαθμών που μπορεί να συγκεντρώσει ένας χρήστης είναι 960.

2.1.2 Race: Πρόβλεψη πορείας ομάδων

Μια διοργάνωση ποδοσφαίρου συνήθως αποτελείται από διάφορες φάσεις στην πορεία προς τον τελικό και την κατάκτηση του τροπαίου. Στο Champions League οι φάσεις αυτές είναι η φάση των ομίλων (group stage), η φάση των 16 (knock-out stage), η προημιτελική φάση (quarter-finals), οι ημιτελικοί (semi-finals) και ο τελικός (final), ο νικητής του οποίου καταλαμβάνει και το τρόπαιο. Ένα παράδειγμα της πορείας των ομάδων στη διοργάνωση απεικονίζεται στον Πίνακα 2.2.

Ο χρήστης καλείται να προβλέψει την πορεία κάθε ομάδας που συμμετέχει στη διοργάνωση. Για κάθε πετυχημένη πρόβλεψη ο χρήστης παίρνει 30 βαθμούς. Ανάλογα με την απόκλιση της πρόβλεψης από την πραγματική πορεία μιας ομάδας, ο χρήστης συγκεντρώνει λιγότερους βαθμούς από κάθε πρόβλεψη. Έτσι, αν για παράδειγμα κάποιος προέβλεψε ότι η Barcelona θα φτάσει μέχρι τους ημιτελικούς, αλλά τελικά κατακτήσει το τρόπαιο, τότε η απόκλιση θα είναι ίση με 2 φάσεις, και ο χρήστης θα πάρει μόνο 10 βαθμούς για την πρόβλεψη αυτή. Αντίστοιχα, αν η πρόβλεψή του ήταν ότι το τρόπαιο θα κατακτήσει η Real Madrid και τελικά η ομάδα φτάσει μέχρι τη φάση των knock-out αναμετρήσεων, τότε έχουμε μια απόκλιση ίση με 3 φάσεις, και ο χρήστης δεν θα πάρει βαθμούς για την πρόβλεψη αυτή. Και σε αυτό το παιχνίδι, καθώς έχουμε 32 ομάδες που συμμετέχουν, ο μέγιστος αριθμός βαθμών που μπορεί να συγκεντρώσει ένας χρήστης είναι 960.

2.1.3 Stock: Γενικότερη απόδοση ομάδων

Οι ομάδες που συμμετέχουν σε κάθε διοργάνωση δεν έχουν την ίδια δυναμικότητα. Έτσι, σε κάθε ομάδα αντιστοιχεί ένας συντελεστής απόδοσης, ο οποίος αντιστοιχεί στις πιθανότητες της ομάδας να διακριθεί στη διοργάνωση. Ο συντελεστής αυτός προκύπτει από τη θέση της ομάδας στην κατάταξη της UEFA, καθώς επίσης από την απόδοσή της στις πρόσφατες διοργανώσεις. Παράδειγμα συντελεστών ομάδων απεικονίζεται στον Πίνακα 2.3.

Ο χρήστης διαθέτει 100 μάρκες, τις οποίες μπορεί να "ποντάρει" πάνω σε οποιεσδήποτε ομάδες επιθυμεί. Ανάλογα με την απόδοση της ομάδας στη διορ-

Team	4th	3rd	KO	QF	SF	F	W
Roma			●				
Chelsea					●		
Bordeaux		●					
CFR Cluj	●						
Panathinaikos			●				
Internazionale			●				
Werder Bremen		●					
Anorthosis	●						
Barcelona							●
Sporting CP			●				
Shakhtar Donetsk		●					
Basel	●						
Liverpool				●			
Atletico Madrid			●				
Marseille		●					
PSV Eindhoven	●						
Manchester United						●	
Villareal				●			
Aalborg BK		●					
Celtic	●						
Bayern Munich				●			
Lyon			●				
Fiorentina		●					
Steaua	●						
Porto				●			
Arsenal					●		
Dynamo Kyiv		●					
Fenerbahce	●						
Juventus			●				
Real Madrid			●				
Zenit		●					
Bate	●						

Πίνακας 2.2: Πορείες των ομάδων στη διοργάνωση

Team	Factor	Team	Factor
Bayern	5.88	Debrecen	33.33
Bordeaux	9.09	Fiorentina	9.09
Juventus	4.00	Liverpool	4.00
Maccabi Haifa	33.33	Lyon	4.76
Besiktas	10.00	Barcelona	3.23
CSKA Moskva	10.00	Dynamo Kyiv	11.11
Manchester United	3.45	Inter	4.00
Wofsburg	9.09	Rubin Kazan	14.29
Marseille	11.11	Rangers	7.29
Milan	5.26	Sevilla	4.76
Real Madrid	3.13	Stuttgart	7.69
Zurich	33.33	Unirea Urziceni	25.00
APOEL	33.33	Alkmaar	8.33
Atletico Madrid	5.88	Arsenal	4.76
Chelsea	3.45	Olympiacos	8.33
Porto	7.69	Standard Liege	8.33

Πίνακας 2.3: Πορείες των ομάδων στη διοργάνωση

γάνωση, ο χρήστης συγκεντρώνει βαθμούς. Πιο συγκεκριμένα, για κάθε νίκη στη φάση των ομίλων η ομάδα παίρνει 3 βαθμούς ενώ για κάθε ισοπαλία παίρνει 1. Στη συνέχεια, ανάλογα με τη φάση στην οποία θα φτάσει, η ομάδα λαμβάνει 2^n βαθμούς. Αν λάβουμε ως δεδομένα τα στοιχεία του Πίνακα 2.1 για τη φάση των ομίλων και τα στοιχεία του Πίνακα 2.2. για τις επόμενες φάσεις, η Barcelona θα συγκέντρωνε:

- Στη φάση των ομίλων πραγματοποίησε 4 νίκες και μία ισοπαλία. Επομένως από τη φάση αυτή συγκεντρώνει $3 \times 4 + 1 \times 1 = 13$ βαθμούς.

- Στις επόμενες φάσεις, η ομάδα έφτασε μέχρι τον τελικό και κατέκτησε το τρόπαιο. Η κάτοχος του τροπαίου θεωρούμε ότι έφτασε στην φάση 6 (οι φάσεις κατά αύξουσα σειρά είναι: τρίτη θέση στον όμιλο, πρόκριση στη φάση των knock-out παιχνιδιών, προημιτελικοί, ημιτελικοί, φιναλίστ και νικήτρια της διοργάνωσης). Έτσι, από τις επόμενες φάσεις η ομάδα συγκεντρώνει $2^6 = 64$ βαθμούς.

Το σύνολο των βαθμών πολλαπλασιάζεται με τον συντελεστή απόδοσης της ομάδας και με τον αριθμό των μάρκων που έχει ποντάρει ο χρήστης. Έτσι, αν υποθέσουμε ότι ο συντελεστής απόδοσης είναι 3.5 και ο χρήστης έχει ποντάρει 50 μάρκες, το σύνολο των βαθμών που θα συγκεντρώσει ανέρχεται σε $(64+13) \times 3.5 \times 50 = 13,475$ βαθμούς. Ανάλογα, υπολογίζονται οι βαθμοί για όλα τα "ποντάρισματα" του χρήστη και το σύνολό τους αποτελεί το σκορ του χρήστη για

το παιχνίδι αυτό. Ο χρήστης με το μεγαλύτερο σκορ στο τέλος της διοργάνωσης, κερδίζει.

2.1.4 Bonus: Γενικότερες προβλέψεις

Στο παιχνίδι αυτό, ο χρήστης καλείται να προβλέψει τον πρώτο σκόρερ της διοργάνωσης, την ομάδα με την καλύτερη επίθεση και την ομάδα με την καλύτερη άμυνα. Ανάλογα με την πρόβλεψη για τον πρώτο σκόρερ, ο χρήστης συγκεντρώνει 30 βαθμούς για κάθε γκολ που πετυχαίνει ο παίκτης της επιλογής του. Το ίδιο ακριβώς ισχύει και για την ομάδα που έχει επιλέξει στην κατηγορία "καλύτερη επίθεση". Για την ομάδα στην κατηγορία "καλύτερη άμυνα", ο χρήστης ξεκινά με 300 βαθμούς, από τους οποίους αφαιρούνται 30 βαθμοί για κάθε γκολ που δέχεται η ομάδα της επιλογής του. Ο χρήστης με τους περισσότερους βαθμούς στο τέλος της διοργάνωσης, κερδίζει.

2.2 Ιστορία του παιχνιδιού

Στη διάρκεια του χρόνου, η διαδικασία του παιχνιδιού έχει υποστεί πολλές αλλαγές, όμως κατά βάση, το σύστημα που ακολουθείται μέχρι σήμερα δεν καταφέρνει να ξεπεράσει τα προβλήματα που αντιμετώπιζε η πρώτη έκδοση. Στη σημερινή του εκδοχή, το παιχνίδι βασίζεται σε μεγάλο βαθμό στη δραστηριότητα του διαχειριστή του παιχνιδιού.

Αρχικά, ο διαχειριστής δημιουργεί ένα αρχείο λογιστικού φύλλου (π.χ. Excel ή OpenOffice Calc) στο οποίο συμπληρώνει τις οδηγίες προς τους παίκτες και μια σειρά από πίνακες με τους αγώνες, τις ομάδες και λοιπά στοιχεία, τα οποία ο κάθε χρήστης καλείται να συμπληρώσει για να συμμετέχει στο παιχνίδι. Στη συνέχεια, το αρχείο αυτό αποστέλλεται μέσω email από χρήστη σε χρήστη με την υποσημείωση ότι η διορία για τη συμμετοχή στο παιχνίδι λήγει μια συγκεκριμένη ημερομηνία.

Οι χρήστες με τη σειρά τους, λαμβάνουν το αρχείο στο email τους, συμπληρώνουν μια σειρά από πίνακες και στη συνέχεια αποστέλλουν τη συμμετοχή τους στο διαχειριστή επίσης μέσω email. Με το πέρας της διορίας, ο διαχειριστής καλείται να συγκεντρώσει όλες τις συμμετοχές και να τις επεξεργαστεί ώστε να είναι σε θέση να υπολογίσει τους βαθμούς και την κατάταξη του κάθε χρήστη στη διάρκεια της διοργάνωσης.

Μετά από κάθε αγωνιστική, ο διαχειριστής περνάει τα αποτελέσματα και στέλνει ένα email στους χρήστες με την τρέχουσα κατάταξή τους και τους βαθμούς που έχουν συγκεντρώσει μέχρι εκείνη τη στιγμή. Είναι προφανές ότι η διαδικασία που μόλις περιγράψαμε εμφανίζει μια σειρά από προβλήματα τα οποία και αναλύουμε στη συνέχεια:

- Η διαδικασία του παιχνιδιού βασίζεται σε μεγάλο βαθμό στην πρωτοβουλία και ενασχόληση του διαχειριστή, καθώς εκείνος καλείται όχι μόνο να

δημιουργήσει το αρχείο συμμετοχών των χρηστών, αλλά επίσης να συγκεντρώσει τα δεδομένα και στη συνέχεια να ενημερώνει τους χρήστες με τα αποτελέσματα.

- Τα δεδομένα βρίσκονται σε κατακερματισμένη μορφή και η επεξεργασία τους παρουσιάζει δυσκολίες. Ακόμη και ο πιο απλός υπολογισμός απαιτεί από τον διαχειριστή να συγκεντρώσει όλες τις συμμετοχές σε ένα αρχείο χειροκίνητα (συνήθως μέσα από μια λειτουργία αντιγραφής-επικόλλησης) και στη συνέχεια να επεξεργαστεί τα δεδομένα στατιστικά.
- Είναι σαφές ότι η σημερινή διαδικασία του παιχνιδιού δεν μπορεί να υποστηρίξει μεγάλο αριθμό συμμετοχών (scalability), καθώς και μόνο η συγκέντρωση των συμμετοχών θα ήταν εξαιρετικά χρονοβόρα.
- Η εμπειρία για το χρήστη είναι ελάχιστα ελκυστική καθώς, όσο καλαίσθητο και αν είναι ένα αρχείο λογιστικού φύλλου, ο χρήστης δεν παύει να βρίσκεται αντιμέτωπος με μια σειρά από πίνακες τους οποίους καλείται να συμπληρώσει για να συμμετάσχει στο παιχνίδι.
- Η μορφή παρουσίασης των αποτελεσμάτων και της κατάταξης των χρηστών στη διάρκεια της διοργάνωσης επίσης δεν αφήνει πολλά περιθώρια για τη διατήρηση του ενδιαφέροντος των χρηστών, καθώς τα μόνα δεδομένα που είναι άμεσα διαθέσιμα σε αυτούς είναι οι λίστες με τους βαθμούς που αυτοί έχουν συγκεντρώσει και τις σειρές κατάταξής τους.
- Η διαδικασία προσφέρει ελάχιστη διαδραστικότητα στο χρήστη και δεν του επιτρέπει να έχει άμεσα στη διάθεσή του στατιστικά στοιχεία, όπως για παράδειγμα το ποσοστό των χρηστών που προέβλεψαν σωστά ένα αποτέλεσμα αγώνα.
- Η όλη λειτουργία του παιχνιδιού μοιάζει κατακερματισμένη και δεν προσφέρει περιθώρια βελτιώσεων και προσθηκών στη διάρκεια του χρόνου, όπως για παράδειγμα τη δυνατότητα μεγαλύτερης εμπλοκής των χρηστών στη διαδικασία μέσα από συζητήσεις ή σχόλια.

Τα παραπάνω προβλήματα δεν είναι δυνατόν να αντιμετωπιστούν με την υπάρχουσα πλατφόρμα του παιχνιδιού. Αντίθετα, η δημιουργία μιας online δυναμικής εφαρμογής γύρω από το παιχνίδι Soccerfun, όχι μόνο αντιμετωπίζει τα προβλήματα αυτά, αλλά επιπλέον δίνει τη δυνατότητα για την προσθήκη περισσότερων δυνατοτήτων και λειτουργιών. Στις επόμενες ενότητες αναλύουμε την εφαρμογή, τόσο από την πλευρά του διαχειριστή, όσο και από την πλευρά του απλού χρήστη που συμμετέχει στο παιχνίδι.

2.3 Η εφαρμογή για το διαχειριστή

Ο διαχειριστής της εφαρμογής είναι υπεύθυνος για την αρχική διαμόρφωση του παιχνιδιού για τη συγκεκριμένη διοργάνωση, για την ενημέρωση των αποτε-

λεσμάτων των αγώνων και στοιχείων όπως η εισαγωγή των σκόρερ ή των ομάδων που προκρίνονται στην επόμενη φάση της διοργάνωσης. Επιπλέον, στην ευθύνη του διαχειριστή εμπίπτουν εργασίες σχετικές με την ίδια την εφαρμογή και τους χρήστες που συμμετέχουν σε αυτήν. Ένα από τα βασικά προβλήματα του υπάρχοντος συστήματος, όπως αυτά αναφέρθηκαν στην προηγούμενη ενότητα, είναι η δυσκολία των διαδικασιών αυτών, η εξάλειψη της οποίας είναι βασική απαίτηση για την εφαρμογή.

2.3.1 Αρχική διαμόρφωση της διοργάνωσης

Στην αρχική φάση του παιχνιδιού, ο διαχειριστής θα πρέπει να έχει τη δυνατότητα με εύκολο και γρήγορο τρόπο να καθορίσει όλες τις παραμέτρους και τα δεδομένα που είναι απαραίτητα για μια συγκεκριμένη διοργάνωση. Πιο αναλυτικά, θα πρέπει να οριστούν:

- Ο αριθμός των ομίλων και οι ονομασίες τους.
- Οι ομάδες που συμμετέχουν στη διοργάνωση, ο όμιλος στον οποίο κάθε ομάδα ανήκει, καθώς επίσης κάποια επιπλέον στοιχεία όπως ο συντελεστής απόδοσης της κάθε ομάδας και η χώρα από την οποία αυτή προέρχεται.
- Οι παίκτες των ομάδων και οι θέσεις στις οποίες αυτοί αγωνίζονται
- Οι αγώνες ανάμεσα στις ομάδες των ομίλων.

2.3.2 Συμπλήρωση αποτελεσμάτων

Στη συνέχεια, καθώς η διοργάνωση βρίσκεται σε εξέλιση, ο διαχειριστής θα πρέπει να έχει τη δυνατότητα να ενημερώνει την εφαρμογή με τα εξής στοιχεία:

- Για κάθε αγώνα θα πρέπει να εισάγεται το αποτέλεσμα του αγώνα, καθώς επίσης και το σκορ.
- Για κάθε γκολ που σημειώνεται θα πρέπει να ενημερώνεται η κατάταξη των σκόρερ.
- Στο τέλος κάθε φάσης, θα πρέπει να υπάρχει η δυνατότητα να καθορίζονται οι ομάδες που προκρίθηκαν στην επόμενη φάση της διοργάνωσης.

2.3.3 Γενικότερη διαχείριση

Τέλος, θα πρέπει να δίνεται η δυνατότητα στο διαχειριστή να πραγματοποιεί μια σειρά από άλλες εργασίες σχετικές με την ίδια την εφαρμογή, όπως για παράδειγμα:

- Θα πρέπει να είναι σε θέση να ενεργοποιεί ή να απενεργοποιεί την εγγραφή νέων χρηστών ή τη συμμετοχή στο παιχνίδι ανάλογα με την ημερομηνία στην οποία βρισκόμαστε.

- Θα πρέπει να έχει τη δυνατότητα να εκτελεί εργασίες διαχείρισης των χρηστών, όπως για παράδειγμα να διαγράφει κάποιον χρήστη.

- Επιπλέον, θα πρέπει να μπορεί να εισάγει ανακοινώσεις ή άλλες πληροφορίες που επιθυμεί να εμφανίζονται σε κάποιες ενότητες της εφαρμογής.

Όλες οι εργασίες που αναφέρθηκαν παραπάνω θα πρέπει να γίνονται με τρόπο οικείο στο διαχειριστή, χωρίς όμως να θέτουν σε κίνδυνο την ασφάλεια και ακεραιότητα των δεδομένων.

2.4 Η εφαρμογή για το χρήστη

Το Soccerfun, πάνω απ'όλα είναι ένα παιχνίδι. Οι χρήστες θα πρέπει να έχουν τη δυνατότητα να συμμετέχουν σε αυτό με τρόπο εύκολο, γρήγορο και απλό στην κατανόηση. Στη συνέχεια αναλύουμε τις βασικές απαιτήσεις των απλών χρηστών του site.

2.4.1 Εγγραφή στο site

Καθώς πρόκειται για μια online εφαρμογή, μια από τις βασικές προϋποθέσεις για τη συμμετοχή στο Soccerfun, είναι η δημιουργία λογαριασμού. Η διαδικασία εγγραφής στο site θα πρέπει να πληροί τις εξής προϋποθέσεις:

- Ο χρήστης θα πρέπει να έχει τη δυνατότητα να επιλέξει ένα όνομα χρήστη (username) και ένα κωδικό εισόδου (password) στην εφαρμογή.

- Επιπλέον, θα πρέπει να είναι δυνατή η ανάκτηση ή η επαναφορά του κωδικού εισόδου σε περίπτωση που ο χρήστης το επιθυμεί.

- Ο κωδικός εισόδου θα πρέπει να προστατεύεται ώστε να είναι γνωστός μόνο από τον ίδιο το χρήστη.

2.4.2 Συμμετοχή στα παιχνίδια

Έχοντας δημιουργήσει το λογαριασμό του στο Soccerfun, ο χρήστης θα πρέπει να έχει τη δυνατότητα να υποβάλει τη συμμετοχή του στο ίδιο το παιχνίδι. Η διαδικασία συμμετοχής στο παιχνίδι θα πρέπει να καλύπτει τις ακόλουθες απαιτήσεις:

- Ο χρήστης θα πρέπει να έχει τη δυνατότητα να συμμετέχει και στα 4 παιχνίδια μέσα από μια απλή, γρήγορη και εύκολα κατανοητή διαδικασία. Οι διάφορες επιλογές, όπως για παράδειγμα η πρόβλεψη του αποτελέσματος

ενός αγώνα ή η επιλογή του πρώτου σκόρερ της διοργάνωσης, θα πρέπει να παρουσιάζονται με τρόπο, ο οποίος είναι σαφής και οικείος στους χρήστες online εφαρμογών.

- Μετά τη λήξη της διορίας συμμετοχής, οι χρήστες θα πρέπει να έχουν πρόσβαση στις σειρές κατάταξης για όλα τα παιχνίδια. Επιπλέον, για λόγους αξιοπιστίας και διαφάνειας του παιχνιδιού, κάθε χρήστης θα πρέπει να έχει τη δυνατότητα να δει τις προβλέψεις οποιουδήποτε άλλου χρήστη που συμμετέχει στο Soccerfun.

- Εκτός από τις σειρές κατάταξης, οι χρήστες θα μπορούν να συμβουλεύονται μια επιπλέον ενότητα με στατιστική ανάλυση των συμμετοχών. Έτσι, θα είναι σε θέση να δουν στοιχεία όπως, για παράδειγμα, το ποσοστό των χρηστών που προέβλεψε σωστά το αποτέλεσμα ενός αγώνα ή τον αριθμό των χρηστών που επέλεξαν μια ομάδα για την κατηγορία της "καλύτερης επίθεσης".

2.4.3 Δραστηριότητες επικοινωνίας

Πέρα από τα ίδια τα παιχνίδια, το site θα υποστηρίζει και δυνατότητες επικοινωνίας μεταξύ των χρηστών. Για παράδειγμα, θα υπάρχει η δυνατότητα σε κάποιους χρήστες να υποβάλλουν άρθρα σχετικά με το ποδόσφαιρο για δημοσίευση στο Soccerfun. Επιπλέον θα μπορούσε να υπάρξει μια ειδικά διαμορφωμένη ενότητα ζωντανής συζήτησης ανάμεσα στους χρήστες (chat). Τέλος, μια επίσης βασική απαίτηση των χρηστών είναι η δυνατότητα επικοινωνίας με το διαχειριστή της εφαρμογής για την επίλυση προβλημάτων ή για την υποβολή προτάσεων, παρατηρήσεων κλπ.

Κεφάλαιο 3

Περιβάλλον ανάπτυξης

Στο κεφάλαιο αυτό θα ασχοληθούμε με την ανάπτυξη της εφαρμογής και με τις τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίησή της.

3.1 Στατικές και δυναμικές σελίδες

Μια online εφαρμογή αποτελείται από μια συλλογή σελίδων, οι οποίες χωρίζονται σε δύο γενικές κατηγορίες: τις στατικές και τις δυναμικές σελίδες. Ανάλογα με την κατηγορία κάθε σελίδας, η διαδικασία που ακολουθείται για την εμφάνισή της στον browser του χρήστη είναι διαφορετική. Στη συνέχεια θα εξετάσουμε τις διαδικασίες αυτές και θα παρουσιάσουμε τα πλεονεκτήματα αλλά και τις δυνατότητες που παρέχει η κάθε κατηγορία.

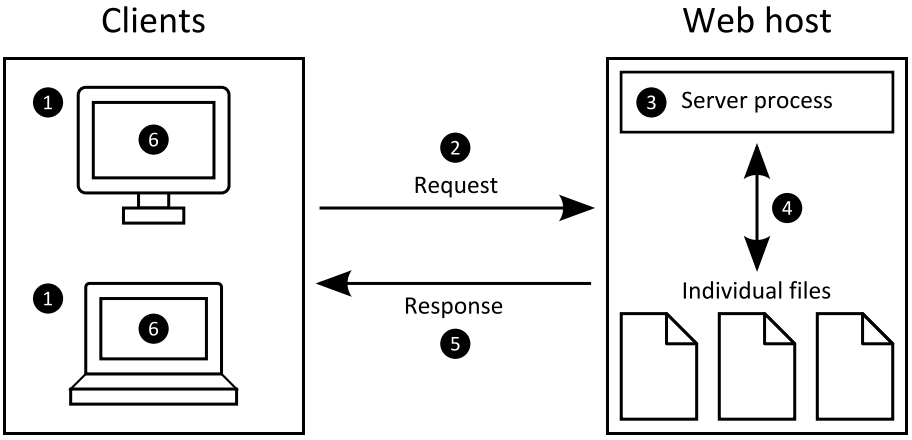
Η διαδικασία για την εμφάνιση μιας στατικής σελίδας απεικονίζεται στον σχήμα 3.1 και αποτελείται από 6 βήματα:

1. Ο χρήστης (client), χρησιμοποιώντας κάποιον browser ζητά να δει μια σελίδα η οποία φιλοξενείται σε κάποιον web host. Η αίτηση αυτή εκφράζεται μέσω μιας διεύθυνσης URL (Universal Resource Locator) η οποία έχει τη μορφή *http://www.example.com/sample.html*. Στο παράδειγμα αυτό, το αρχείο με όνομα *sample.html* αντιστοιχεί στη σελίδα που ο χρήστης αναζητά, ενώ το όνομα *www.example.com* αντιστοιχεί στον web host που φιλοξενεί τη σελίδα αυτή.

2. Η αίτηση (request) λαμβάνεται από τον πάροχο του χρήστη και προωθείται στον web host, ο οποίος φιλοξενεί τη συγκεκριμένη σελίδα.

3. Ο web host παραλαμβάνει την αίτηση και αναγνωρίζει το όνομα του αρχείου που αναζητείται.

4. Η διαδικασία του web server αναλαμβάνει να αναζητήσει στο αποθηκευτικό μέσο το συγκεκριμένο αρχείο και να το ανακτήσει.



Σχήμα 3.1: Διαδικασία εμφάνισης στατικής σελίδας

5. Στη συνέχεια, ο web server ανταποκρίνεται (response) στο αίτημα που δέχτηκε, στέλνοντας τη σελίδα που ζητήθηκε πίσω στον browser του χρήστη.

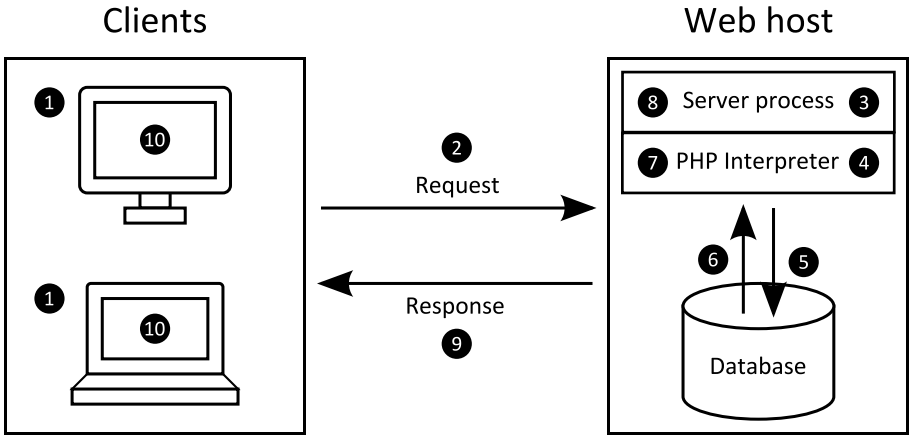
6. Ο browser λαμβάνει το αρχείο και εμφανίζει το περιεχόμενό του στον χρήστη.

Η σελίδα του παραπάνω παραδείγματος *sample.html* είναι μια στατική σελίδα. Το περιεχόμενό της είναι αυστηρά καθορισμένο και δεν επιδέχεται παραμετροποιήσεων ή αλλαγών ανάλογα με το πότε, ποιός και από πού αναζητά τη σελίδα αυτή. Αντίθετα, οι δυναμικές σελίδες επιτρέπουν την τροποποίηση του περιεχομένου τους ανάλογα με μια σειρά από παραμέτρους. Η διαδικασία που ακολουθείται για την εμφάνιση μιας δυναμικής σελίδας απεικονίζεται στο σχήμα 3.2 και αποτελείται από τα ακόλουθα 10 βήματα:

1. Ο χρήστης (client), όπως και στην προηγούμενη περίπτωση, χρησιμοποιώντας κάποιον browser ζητά να δει μια σελίδα η οποία φιλοξενείται σε κάποιον web host. Η αίτηση αυτή, καθώς πλέον πρόκειται για δυναμική σελίδα, έχει τη μορφή *http://www.example.com/sample.php*. Στο παράδειγμα αυτό, το αρχείο με όνομα *sample.php* αντιστοιχεί στη σελίδα που ο χρήστης αναζητά, ενώ το όνομα *www.example.com* αντιστοιχεί στον web host που φιλοξενεί τη σελίδα αυτή.

2. Η αίτηση (request) λαμβάνεται από τον πάροχο του χρήστη και προωθείται στον web host, ο οποίος φιλοξενεί τη συγκεκριμένη σελίδα.

3. Ο web host παραλαμβάνει την αίτηση και αναγνωρίζει το όνομα του αρχείου που αναζητείται. Αφού ανακτήσει το αρχείο και αντιληφθεί ότι πρόκειται για ένα αρχείο το οποίο χρήζει επεξεργασίας (και όχι ένα απλό HTML αρχείο), το προωθεί στον μεταφραστή PHP.



Σχήμα 3.2: Διαδικασία εμφάνισης δυναμικής σελίδας

4. Ο μεταφραστής PHP (interpreter) λαμβάνει το αρχείο και εκτελεί τον κώδικα που περιέχεται σε αυτό. Στην πλειοψηφία των περιπτώσεων, ο κώδικας αυτός περιέχει κλήσεις στη βάση δεδομένων.

5. Ο PHP interpreter ζητά από τη βάση δεδομένων να επεξεργαστεί τις κλήσεις.

6. Η διαδικασία της βάσης δεδομένων (database process) επεξεργάζεται τις κλήσεις και επιστρέφει στον PHP Interpreter τα αποτελέσματα της ερώτησης (query).

7. Ο PHP interpreter ολοκληρώνει την εκτέλεση του κώδικα PHP με τα αποτελέσματα που παρέλαβε από τη βάση δεδομένων και επιστρέφει στον web server το τελικό αρχείο.

8. Ο web server παραλαμβάνει το αρχείο σε μορφή HTML, και...

9. .. ανταποκρίνεται στην αίτηση του χρήστη στέλνοντας το τελικό αρχείο (response).

10. Ο browser λαμβάνει την πληροφορία σε μορφή HTML και την απεικονίζει στο χρήστη.

Η ύπαρξη του PHP interpreter και του συστήματος βάσης δεδομένων, προσφέρουν στην εφαρμογή τη δυνατότητα να παράγει το περιεχόμενο μιας σελίδας με τέτοιο τρόπο, ώστε η πληροφορία που περιέχεται σε αυτήν να είναι κάθε φορά διαφορετική ανάλογα με μια σειρά από παραμέτρους.

Ας υποθέσουμε, για παράδειγμα, ότι μια σελίδα περιέχει τα ονόματα των χρηστών που συμμετέχουν στο παιχνίδι και τους βαθμούς που έχει συγκεντρώσει κάθε χρήστης. Αν η σελίδα αυτή είναι στατική, τότε ο προγραμματιστής της

εφαρμογής θα πρέπει κάθε φορά να ανανεώνει “χειροκίνητα” την πληροφορία που περιέχεται σε αυτήν, με αποτέλεσμα να ελοχεύει ο κίνδυνος λάθους ή η πληροφορία να μην είναι ενημερωμένη. Αντίθετα, αν η σελίδα αυτή είναι δυναμική, τότε το περιεχόμενό της θα παράγεται κάθε φορά που αυτή ζητείται είτε με κλήσεις στη βάση δεδομένων, είτε με υπολογισμούς από τον PHP interpreter, είτε με συνδυασμό και των δύο τεχνικών.

Το συγκριτικό αυτό πλεονέκτημα των δυναμικών σελίδων, καθιστούν τη χρήση τους αναγκαία για την ανάπτυξη online εφαρμογών. Στις επόμενες ενότητες θα αναλύσουμε το σύστημα βάσης δεδομένων που υποστηρίζει την εφαρμογή, καθώς επίσης τη γλώσσα PHP και τον τρόπο με τον οποίο αυτή χρησιμοποιείται για τις ανάγκες του παιχνιδιού.

3.2 MySQL: Η βάση της εφαρμογής

Η εφαρμογή στηρίζεται σε μεγάλο βαθμό στη διαχείριση δεδομένων: οι ομάδες, οι αγώνες, τα αποτελέσματα, οι χρήστες και οι προβλέψεις τους αποτελούν ένα σύνολο από στοιχεία, τα οποία θα πρέπει να είμαστε σε θέση να αποθηκεύσουμε και να διαχειριστούμε με ευκολία και αποτελεσματικότητα. Η βασική αυτή απαίτηση για την ανάπτυξη της εφαρμογής, καθιστά αναγκαία τη χρήση ενός συστήματος διαχείρισης βάσης δεδομένων για τις ανάγκες του site.

3.2.1 Το σύστημα βάσεων δεδομένων MySQL

Η MySQL αποτελεί μια εξαιρετικά δημοφιλή και διαδεδομένη υλοποίηση ενός σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων. Παρ’ όλο που είναι μια τεχνολογία ανοιχτού κώδικα, το σύστημα υποστηρίζεται από μεγάλες εταιρίες του χώρου όπως η Sun (και πρόσφατα η Oracle) και γνωρίζει διαρκή εξάπλωση στο χώρο των online εφαρμογών. Το γεγονός ότι η χρήση του είναι δωρεάν, αλλά και η διαθεσιμότητά του σε μια πληθώρα λειτουργικών συστημάτων το έχουν καταστήσει κυρίαρχη λύση στην ανάπτυξη online εφαρμογών μικρής και μεσαίας κλίμακας.

3.2.2 Σχεδίαση της βάσης δεδομένων της εφαρμογής

Για τη σχεδίαση της βάσης δεδομένων που θα υποστηρίξει την εφαρμογή, ακολουθήσαμε το μοντέλο Οντοτήτων - Συσχετίσεων, σύμφωνα με το οποίο ο μικρόκοσμος της εφαρμογής αποτελείται από βασικά αντικείμενα, τα οποία ονομάζονται οντότητες. Ανάμεσα στα αντικείμενα αυτά υπάρχουν σχέσεις αλληλεπίδρασης ή αλλιώς συσχετίσεις.

Για την εφαρμογή που εξετάζουμε, οι βασικές οντότητες είναι οι ομάδες που συμμετέχουν στη διοργάνωση, οι αγώνες που λαμβάνουν χώρα, οι παίκτες των ομάδων και οι χρήστες της εφαρμογής που συμμετέχουν στα παιχνίδια. Στη συνέχεια, θα περιγράψουμε τις οντότητες αυτές αναλυτικά:

Team Αποτελεί την οντότητα της έννοιας **ομάδα**. Τα βασικά γνωρίσματα του αντικειμένου αυτού, είναι το όνομα της ομάδας (*name*), η χώρα από την οποία προέρχεται (*country*), ο συντελεστής απόδοσης της ομάδας (*factor*), ο όμιλος στον οποίο ανήκει (*group*) και η φάση της διοργάνωσης στην οποία βρίσκεται (*stage*).

Match Αποτελεί την οντότητα της έννοιας **αγώνας**. Τα βασικά γνωρίσματα του αντικειμένου αυτού, είναι η ημερομηνία διεξαγωγής του αγώνα (*date*), ο όμιλος (*group*) και η φάση (*stage*) που αντιστοιχούν στον αγώνα. Τέλος, ένα ακόμη γνώρισμα της οντότητας, αποτελεί το αποτέλεσμα (*result*) του αγώνα.

Player Αποτελεί την οντότητα της έννοιας **παίκτης**. Τα βασικά γνωρίσματα του αντικειμένου αυτού, είναι το όνομα του παίκτη (*name*) και η θέση στην οποία αγωνίζεται (*position*).

User Αποτελεί την οντότητα της έννοιας **χρήστης**. Τα βασικά γνωρίσματα του αντικειμένου αυτού, είναι το όνομα χρήστη (*username*), ο κωδικός εισόδου (*password*) και το *email* του χρήστη.

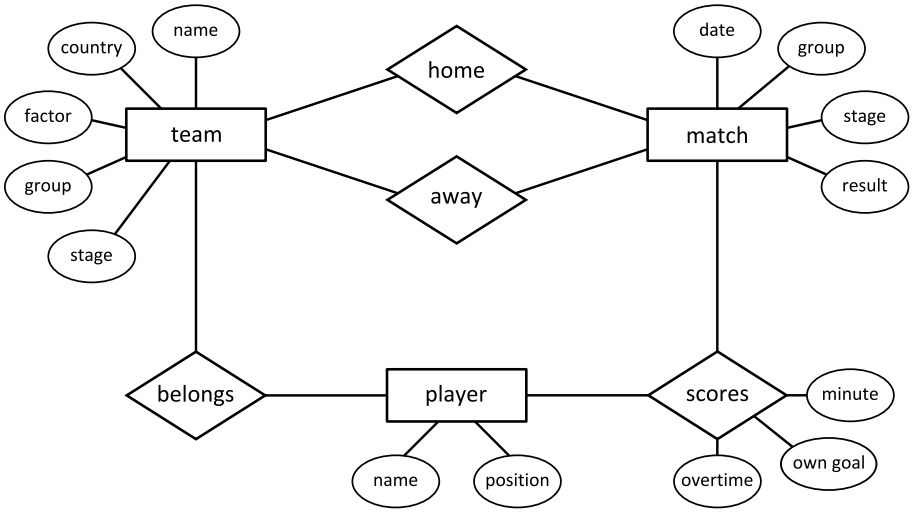
Για λόγους ευκολίας στην απεικόνιση, θα εξετάσουμε τις παραπάνω οντότητες σε δύο ενότητες: η πρώτη αφορά στην ίδια τη διοργάνωση που εξετάζουμε και περιλαμβάνει όλες τις οντότητες εκτός από την οντότητα user. Οι σχέσεις που εμφανίζονται στην ενότητα αφορούν στην ίδια τη διοργάνωση, γύρω από την οποία εκτυλίσσεται το παιχνίδι, και στην καταγραφή των γεγονότων που σχετίζονται με αυτήν:

Home Η σχέση αυτή συσχετίζει μια ομάδα (team) με έναν αγώνα (match) και υποδηλώνει ότι μια ομάδα συμμετέχει σε έναν αγώνα ως γηπεδούχος. Πρόκειται για μια σχέση 1:N, καθώς μια ομάδα μπορεί να συμμετάσχει σε πολλούς αγώνες ως γηπεδούχος, ενώ κάθε αγώνας έχει μόνο μια ομάδα σαν γηπεδούχο.

Away Η σχέση αυτή συσχετίζει μια ομάδα (team) με έναν αγώνα (match) και υποδηλώνει ότι μια ομάδα συμμετέχει σε έναν αγώνα ως φιλεξενούμενη. Όπως και στη σχέση home, πρόκειται για μια σχέση 1:N, καθώς μια ομάδα μπορεί να συμμετάσχει σε πολλούς αγώνες ως φιλοξενούμενη, ενώ κάθε αγώνας έχει μόνο μια ομάδα σαν φιλοξενούμενη.

Belongs Η σχέση αυτή συσχετίζει έναν παίκτη (player) με μια ομάδα (team) και υποδηλώνει ότι ο παίκτης αγωνίζεται με τη φανέλα μιας ομάδας. Πρόκειται για μια σχέση 1:N, καθώς μια ομάδα μπορεί να έχει πολλούς παίκτες, ενώ κάθε παίκτης αγωνίζεται σε μια και μόνο ομάδα.

Scores Η σχέση αυτή συσχετίζει έναν παίκτη (player) με έναν αγώνα (match) και υποδηλώνει ότι ένας παίκτης σκόραρε σε έναν αγώνα. Πρόκειται για μια



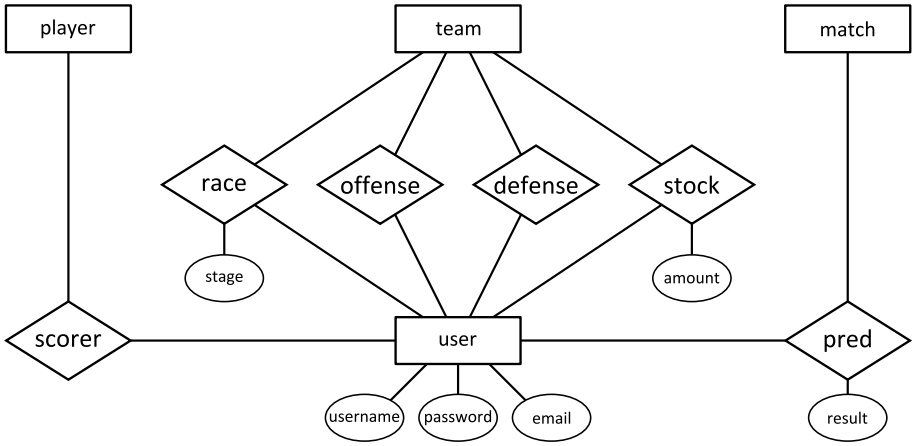
Σχήμα 3.3: Διάγραμμα οντοτήτων-συσχετίσεων (E-R), μέρος 1ο

σχέση M:N, καθώς σε έναν αγώνα είναι δυνατόν να σκοράρουν περισσότεροι του ενός παίκτη, ενώ επιπλεόν ο ίδιος παίκτης είναι δυνατόν να σκοράρει σε περισσότερους του ενός αγώνες. Η σχέση αυτή χαρακτηρίζεται από τα εξής γνωρίσματα: *minute* (λεπτό) στο οποίο ο παίκτης σκόραρε, *own goal*, το οποίο δηλώνει αν πρόκειται για αυτογκόλ, και *overtime*, το οποίο δηλώνει αν το γκολ επιτεύχθηκε στην παράταση.

Οι οντότητες που περιγράψαμε παραπάνω και οι συσχετίσεις που αντιστοιχούν στην πρώτη ενότητα του μοντέλου της βάσης δεδομένων, διαμορφώνουν το διάγραμμα οντοτήτων - συσχετίσεων που απεικονίζεται στο σχήμα 3.3.

Στην δεύτερη ενότητα του μοντέλου της βάσης δεδομένων της εφαρμογής, ασχολούμαστε με το ίδιο το παιχνίδι και με τη συμμετοχή του χρήστη σε αυτό. Υπενθυμίζουμε ότι το παιχνίδι αποτελείται από 4 υπο-παιχνίδια: Pred (πρόβλεψη αποτελεσμάτων αγώνων), Race (πρόβλεψη πορείας μιας ομάδας στη διοργάνωση), Stock ("ποντάρισμα" 100 μαρκών στις ομάδες της διοργάνωσης και συλλογή βαθμών ανάλογα με την απόδοση της ομάδας) και τέλος, Bonus (πρόβλεψη της ομάδας με την καλύτερη επίθεση, της ομάδας με την καλύτερη άμυνα και του πρώτου σκόρερ της διοργάνωσης). Έτσι, με βάση τις οντότητες που έχουμε ορίσει για το μοντέλο μας, προκύπτουν οι εξής σχέσεις:

Pred Η σχέση αυτή συσχετίζει ένα χρήστη (user) με έναν αγώνα (match) και υποδηλώνει την πρόβλεψη του χρήστη για τον αγώνα. Η πρόβλεψη αυτή, εκφράζεται μέσω του γνωρίσματος της σχέσης *result* (αποτέλεσμα). Πρόκειται για μια σχέση M:N καθώς ένας χρήστης πραγματοποιεί προβλέψεις για διαφορετικούς αγώνες, ενώ για έναν αγώνα υπάρχουν προβλέψεις διαφορετικών χρηστών.



Σχήμα 3.4: Διάγραμμα οντοτήτων-συσχετίσεων (E-R), μέρος 2ο

Race Η σχέση αυτή συσχετίζει ένα χρήστη (user) με μια ομάδα (team) και υποδηλώνει την πρόβλεψη του χρήστη για την πορεία της ομάδας στη διοργάνωση. Η πρόβλεψη αυτή εκφράζεται μέσω του γνωρίσματος της σχέσης *stage* (φάση της διοργάνωσης). Πρόκειται για μια μια σχέση M:N καθώς ένας χρήστης πραγματοποιεί προβλέψεις για διαφορετικές ομάδες, ενώ για μια ομάδα υπάρχουν προβλέψεις διαφορετικών χρηστών.

Stock Η σχέση αυτή συσχετίζει ένα χρήστη (user) με μια ομάδα (team) και υποδηλώνει το "ποντάρισμα" του χρήστη πάνω σε μια ομάδα. Το ποντάρισμα αυτό εκφράζεται μέσω του γνωρίσματος της σχέσης *amount* (ποσό), το οποίο αντιστοιχεί στον αριθμό των μαρκών του πονταρίσματος. Πρόκειται για μια σχέση M:N, καθώς ένας παίκτης μπορεί να ποντάρει σε διαφορετικές ομάδες, ενώ μια ομάδα μπορεί να περιλαμβάνεται στα πονταρίσματα διαφορετικών παικτών.

Scorer, **Offense**, **Defence** Οι σχέσεις αυτές συσχετίζουν ένα χρήστη (user) με ένα παίκτη (για τη σχέση scorer) η με μια ομάδα (για τις σχέσεις offense και defense) και, από κοινού, εκφράζουν τη συμμετοχή του χρήστη στο υποπαιχνίδι bonus. Κάθε μια υποδηλώνει την επιλογή του χρήστη για τον πρώτο σκόρερ της διοργάνωσης, την ομάδα με την καλύτερη επίθεση και την ομάδα με την καλύτερη άμυνα αντίστοιχα. Πρόκειται για σχέσεις 1:N, καθώς κάθε χρήστης δικαιούται μόνο μια επιλογή για κάθε κατηγορία, ενώ διαφορετικοί παίκτες δύνανται να έχουν τις ίδιες επιλογές.

Οι σχέσεις που αναλύσαμε παραπάνω, σε συνδυασμό με τις οντότητες στις οποίες αφορούν, διαμορφώνουν το διάγραμμα οντοτήτων-συσχετίσεων που απεικονίζεται στο σχήμα 3.4. Σημειώνουμε ότι διάγραμμα αυτό δεν αναιρεί το προηγούμενο, αλλά αντίθετα το επεκτείνει. Τα γνωρίσματα των οντοτήτων team, match

name	Το όνομα της ομάδας
country	Η χώρα προέλευσης της ομάδας
factor	Ο συντελεστής απόδοσης της ομάδας
group	Ο όμιλος στον οποίο ανήκει η ομάδα
stage	Η φάση της διοργάνωσης στην οποία βρίσκεται η ομάδα

Πίνακας 3.1: Η σχέση teams, μέρος 1ο

και player καθώς επίσης οι σχέσεις ανάμεσα στις οντότητες αυτές παραλείπονται από το διάγραμμα του σχήματος 3.4 για λόγους απλότητας.

3.2.3 Υλοποίηση της βάσης δεδομένων

Έχοντας ορίσει το θεωρητικό μοντέλο της βάσης δεδομένων που θα υποστηρίζει την εφαρμογή, στη συνέχεια, από το μοντέλο οντοτήτων-συσχετίσεων θα περάσουμε στο σχεσιακό μοντέλο και στην περιγραφή των πινάκων από τους οποίους απαρτίζεται η βάση δεδομένων. Σαν πρώτο βήμα, σε ευθεία αντιστοιχία με τις βασικές οντότητες του μοντέλου μας θα δημιουργήσουμε τις βασικές σχέσεις της βάσης δεδομένων.

Η σχέση teams, μέρος 1ο

Από την οντότητα team ορίζουμε τον πίνακα **teams**. Τα ορίσματα του πίνακα αυτού θα περιλαμβάνουν τα γνωρίσματα της οντότητας, όπως αυτά αναλύθηκαν παραπάνω. Τα βασικά ορίσματα εμφανίζονται στον πίνακα 3.1.

Παρ'όλο που το όνομα κάθε ομάδας είναι θεωρητικά μοναδικό και προσδιορίζει μια ομάδα με μοναδικό τρόπο, επιλέγουμε επιπλέον να προσθέσουμε ένα μοναδικό αριθμητικό αναγνωριστικό με το όνομα *id*, το οποίο γενικά μας διευκολύνει περισσότερο σε μια σειρά από λειτουργίες, όπως αυτό θα καταστεί σαφές στην επόμενη ενότητα, όπου και ασχολούμαστε με τον προγραμματισμό της εφαρμογής. Το πεδίο αυτό θα χρησιμοποιηθεί σαν πρωτεύον κλειδί (primary key) της σχέσης.

Από την εξέταση των παραπάνω ορισμάτων, είναι προφανές ότι τα ορίσματα *country*, *group* και *stage* αποτελούν πληροφορία η οποία επαναλαμβάνεται για περισσότερες της μιας ομάδες. Για το λόγο αυτό, δημιουργούμε αντίστοιχες βοηθητικές σχέσεις και αναφερόμαστε σε αυτές από την παρούσα σχέση.

Οι σχέσεις countries, groups και stages

Οι βοηθητικές αυτές σχέσεις, οι οποίες δημιουργούνται από την ανάγκη να διατηρήσουμε τα δεδομένα σε κανονική μορφή, έχουν παρόμοια δομή και περιλαμβάνουν τα ορίσματα του πίνακα 3.2.

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
name	Το όνομα ή η περιγραφή της εγγραφής (π.χ. το όνομα της χώρας ή της φάσης της διοργάνωσης)

Πίνακας 3.2: Οι σχέσεις countries, groups και stages

Το πεδίο *id* αποτελεί και το πρωτεύον κλειδί της κάθε σχέσης και θα το χρησιμοποιήσουμε για να δημιουργήσουμε αναφορές ξένων κλειδιών (foreign keys) σε οποιαδήποτε άλλη σχέση χρησιμοποιεί την πληροφορία των σχέσεων αυτών.

Η σχέση teams, μέρος 2ο

Έχοντας δημιουργήσει τις βοηθητικές σχέσεις που αναφέραμε παραπάνω, επιστρέφουμε και πάλι στο μοντέλο της σχέσης teams και αντικαθιστούμε τα ορίσματα *country*, *group* και *stage* με αναφορές foreign keys στις αντίστοιχες σχέσεις. Έτσι, και με την προσθήκη του πεδίου *id*, τα ορίσματα της σχέσης γίνονται όπως αυτά εμφανίζονται στον πίνακα 3.3

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
name	Το όνομα της ομάδας
country_id	Η χώρα προέλευσης της ομάδας *Foreign key στη σχέση countries*
factor	Ο συντελεστής απόδοσης της ομάδας
group_id	Ο όμιλος στον οποίο ανήκει η ομάδα *Foreign key στη σχέση groups*
stage_id	Η φάση της διοργάνωσης στην οποία βρίσκεται η ομάδα *Foreign key στη σχέση stages*

Πίνακας 3.3: Η σχέση teams, μέρος 2ο

Οι σχέσεις players και positions

Από την οντότητα player, δημιουργούμε τη σχέση players, οι εγγραφές της οποίας θα αντιστοιχούν στους παίκτες που αγωνίζονται στη διοργάνωση. Από το μοντέλο οντοτήτων-συσχετίσεων διαπιστώνουμε ότι η οντότητα έχει σαν γνωρίσματα το όνομα του παίκτη και τη θέση στην οποία αυτός αγωνίζεται. Και σε αυτήν την περίπτωση, είναι προφανές ότι περισσότεροι του ενός παίκτες είναι δυνατόν να αγωνίζονται στην ίδια θέση. Ακολουθώντας παρόμοια λογική με την περίπτωση των βοηθητικών σχέσεων που ακολουθήσαμε παραπάνω, δημιουργούμε μια βοηθητική σχέση με το όνομα **positions**, η οποία διαθέτει πεδία αντίστοιχα με αυτά του πίνακα 3.2.

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
name	Το όνομα του παίκτη
team_id	Η ομάδα στην οποία αγωνίζεται το παίκτης *Foreign key στη σχέση teams*
position_id	Η θέση στην οποία αγωνίζεται το παίκτης *Foreign key στη σχέση positions*

Πίνακας 3.4: Η σχέση players

Στη συνέχεια, από το μοντέλο οντοτήτων-συσχετίσεων διαπιστώνουμε ότι οι οντότητες player και team συνδέονται με μια συσχέτιση με το όνομα belongs. Η συσχέτιση αυτή υποδηλώνει ότι ένας παίκτης ανήκει σε μια ομάδα και είναι μια συσχέτιση 1:N. Επομένως, μπορούμε να εκφράσουμε τη συσχέτιση αυτή στο σχεσιακό μοντέλο εισάγωντας στη σχέση **players** ένα επιπλέον πεδίο με το όνομα *team_id*, το οποίο θα αναφέρεται στη σχέση **teams**. Τέλος, παρ'όλο που ο συνδυασμός του ονόματος του παίκτη και του αναγνωριστικού της ομάδας είναι κατά πάσα πιθανότητα μοναδικός, επιλέγουμε και σε αυτήν τη σχέση να προσθέσουμε ένα επιπλέον πεδίο *id*, το οποίο θα είναι ένα μοναδικό αριθμητικό αναγνωριστικό και θα επιτελεί το ρόλο του πρωτεύοντος κλειδιού της σχέσης. Τα πεδία της σχέσης απεικονίζονται στον πίνακα 3.4.

Η σχέση matches

Με βάση την οντότητα match του μοντέλου οντοτήτων-συσχετίσεων, στη συνέχεια θα δημιουργήσουμε τη σχέση matches. Τα βασικά ορίσματα της οντότητας είναι η ημερομηνία διεξαγωγής (*date*), ο όμιλος (*group*) και η φάση (*stage*) και τέλος το αποτέλεσμα του αγώνα. Για τον όμιλο και τη φάση έχουμε ήδη δημιουργήσει τις αντίστοιχες βοηθητικές σχέσεις και επομένως μπορούμε να χρησιμοποιήσουμε foreign keys για να αναφερθούμε στις σχέσεις αυτές.

Επιπλέον, επιλέγουμε να δημιουργήσουμε μια ακόμη βοηθητική σχέση με το όνομα *match_days*, τα ορίσματα της οποίας απεικονίζονται στον πίνακα 3.5.

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
day	Ένας αριθμός που αντιστοιχεί στην αγωνιστική
match_date	Η ημερομηνία της αγωνιστικής ημέρας
stage_id	Η φάση στην οποία αντιστοιχεί η αγωνιστική

Πίνακας 3.5: Η σχέση match_days

Ο λόγος της δημιουργίας της βοηθητικής αυτής σχέσης είναι το γεγονός ότι σε πολλές διοργανώσεις, όπως για παράδειγμα το champions league, μια αγωνιστική μπορεί να λαμβάνει χώρα σε περισσότερες της μιας ημερομηνίες. Η βοηθητική αυτή σχέση, όπως αυτό θα γίνει σαφές στην επόμενη ενότητα, μας διευ-

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
name	Το όνομα ή η περιγραφή της εγγραφής

Πίνακας 3.6: Η βοηθητική σχέση results

κολύνει σημαντικά σε μιας σειρά από λειτουργίες της εφαρμογής. Επιπλέον, η φάση στην οποία αντιστοιχεί κάθε ματς, θα μπορούσαμε να πούμε ότι είναι μια παραγώμενη ιδιότητα, καθώς είναι άμεσα συνυφασμένη με την ημερομηνία διεξαγωγής. Έτσι, μπορούμε να την αφαιρέσουμε από τα γνωρίσματα της σχέσης matches. Επιλέγουμε να την διατηρήσουμε σαν ξεχωριστό γνώρισμα της σχέσης match_days, καθώς η ύπαρξή της θα μας διευκολύνει στην εκτέλεση ερωτήσεων, όπως θα δούμε στη συνέχεια.

Εξετάζοντας το μοντέλο οντοτήτων-συσχετίσεων, παρατηρούμε ότι η οντότητα match συνδέεται με την οντότητα team μέσω δύο συσχετίσεων, τις home και away, οι οποίες υποδηλώνουν τις δύο ομάδες που συμμετέχουν στον αγώνα ως γηπεδούχος και φιλοξενούμενη ομάδα αντίστοιχα. Οι συσχετίσεις αυτές είναι 1:N και επομένως μπορούν να απορροφηθούν στη σχέση matches εισάγοντας κατάλληλα πεδία: το πεδίο *home_team_id*, το οποίο αντιστοιχεί στη γηπεδούχο ομάδα και αποτελεί foreign key στη σχέση teams, και το πεδίο *away_team_id*, το οποίο αντιστοιχεί στη φιλοξενούμενη ομάδα και επίσης αποτελεί foreign key στη σχέση teams.

Συνεχίζοντας την εξέταση των γνωρισμάτων της σχέσης matches, στρέφουμε την προσοχή μας στο πεδίο του αποτελέσματος του αγώνα (result). Αν υποθέσουμε ότι το περιεχόμενο του πεδίου αυτού για έναν αγώνα είναι "2-1", εύκολα διαπιστώνουμε ότι η πληροφορία σε αυτήν τη μορφή δεν μας είναι ιδιαίτερα χρήσιμη, καθώς θα πρέπει να την επεξεργαστούμε περαιτέρω για να αποφανθούμε αν πρόκειται για νίκη της γηπεδούχου, ισοπαλία ή νίκη της φιλοξενούμενης ομάδας. Επιπλέον, σύμφωνα με τις ανάγκες της εφαρμογής, θα πρέπει να είμαστε σε θέση με εύκολο και γρήγορο τρόπο να υπολογίζουμε στοιχεία όπως τα συνολικά γκολ που έχει πετύχει μια ομάδα ή το συνολικό αριθμό των νικών της. Για το λόγο αυτό εισάγουμε μια ακόμη βοηθητική σχέση με το όνομα *results*, τα πεδία της οποίας απεικονίζονται στον πίνακα 3.6.

Χρησιμοποιώντας τη βοηθητική αυτή σχέση, δημιουργούμε ένα πεδίο με το όνομα *result_id*, το οποίο θα αποτελεί foreign key στη σχέση αυτή. Έτσι, εισάγοντας στη σχέση results εγγραφές που να αντιστοιχούν σε νίκη της γηπεδούχου, ισοπαλία, νίκη της φιλοξενούμενης ομάδας και αγώνας σε εκκρεμότητα, είμαστε σε θέση εύκολα και γρήγορα να ανακτήσουμε αγώνες που μας ενδιαφέρουν με βάση το αποτέλεσμα.

Όμως, για τη διατήρηση της πληροφορίας του γνωρίσματος result, το πεδίο *result_id* δεν αρκεί, καθώς πλέον δεν γνωρίζουμε τον αριθμό των γκολ κάθε ομάδας. Έτσι, δημιουργούμε δύο ακόμη πεδία με ονόματα *home_goals* και *away_goals*, τα οποία αντιστοιχούν στον αριθμό των γκολ της γηπεδούχου και φιλοξενούμε-

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
match_day_id	Η αγωνιστική του ματς *Foreign key στη σχέση match_days*
group_id	Ο όμιλος στον οποίο αντιστοιχεί το ματς *Foreign key στη σχέση groups*
home_team_id	Η γηπεδούχος ομάδα *Foreign key στη σχέση teams*
away_team_id	Η φιλοξενούμενη ομάδα *Foreign key στη σχέση teams*
result_id	Το αποτέλεσμα του αγώνα *Foreign key στη σχέση results*
home_goals	Τα γκολ της γηπεδούχου ομάδας
away_goals	Τα γκολ της φιλοξενούμενης ομάδας
home_penalties	Τα πέναλτυ της γηπεδούχου ομάδας
away_penalties	Τα πέναλτυ της φιλοξενούμενης ομάδας

Πίνακας 3.7: Η σχέση matches

νης ομάδας αντίστοιχα. Για λόγους πληρότητας και για την περίπτωση που ένας αγώνας κριθεί στα πέναλτυ, περιλαμβάνουμε αντίστοιχα πεδία για κάθε ομάδα. Η σχέση matches, όπως αυτή διαμορφώνεται μετά από τις παραπάνω παρατηρήσεις, απεικονίζεται στον πίνακα 3.7.

Η σχέση goals

Επιστρέφοντας και πάλι στο διάγραμμα οντοτήτων-συσχετίσεων εξετάζουμε τη συσχέτιση scores, η οποία συνδέει ένα παίκτη με έναν αγώνα. Τα γνωρίσματα της συσχέτισης αυτής είναι το λεπτό στο οποίο ο παίκτης πέτυχε το γκολ (*minute*), το αν πρόκειται για αυτογκολ (*own goal*) και το αν το γκολ επιτεύχθηκε στην παράταση του αγώνα (*overtime*). Δεδομένου ότι πρόκειται για μια M:N συσχέτιση, θα δημιουργήσουμε μια νέα σχέση με το όνομα *goals*, η οποία θα περιλαμβάνει σαν γνωρίσματα τα ορίσματα της οντότητας scores. Επιπλέον θα αναφέρεται στις σχέσεις **players** και **matches** μέσω των πεδίων *player_id* και *match_id*, τα οποία αποτελούν foreign keys στις σχέσεις αυτές. Τέλος, εισάγουμε ένα ακόμη πεδίο με το όνομα *id*, το οποίο θα αποτελεί το πρωτεύον κλειδί της σχέσης. Τα πεδία της σχέσης goals απεικονίζονται στον πίνακα 3.8.

Η σχέση teams, μέρος 3ο

Με βάση τις απαιτήσεις της εφαρμογής, θα πρέπει να είμαστε σε θέση με εύκολο και γρήγορο τρόπο να απαντούμε σε ερωτήσεις όπως "πόσοι είναι οι βαθμοί που έχει συγκεντρώσει μια ομάδα" ή "πόσες είναι οι νίκες που έχει πετύχει μια ομάδα συνολικά στη διοργάνωση". Οι σχέσεις που έχουμε δημιουργήσει μέχρι

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
match_id	Ο αγώνας στον οποίο επιτεύχθηκε το γκολ *Foreign key στη σχέση match_days*
player_id	Ο παίκτης που πέτυχε το γκολ *Foreign key στη σχέση players*
minute	Το λεπτό στο οποίο επιτεύθηκε το γκολ
owngoal	Αν πρόκειται για αυτογκόλ
overtime	Αν το γκολ επιτεύχθηκε στην παράταση

Πίνακας 3.8: Η σχέση goals

στιγμής μας δίνουν τη δυνατότητα να απαντήσουμε στις ερωτήσεις αυτές πραγματοποιώντας μια σειρά από υπολογισμούς στη βάση δεδομένων.

Για παράδειγμα, για να υπολογίσουμε τους βαθμούς που έχει συγκεντρώσει μια ομάδα αρκεί να βρούμε όλους τους αγώνες στους οποίους έχει συμμετάσχει η ομάδα αυτή και για κάθε νίκη να προσθέσουμε 3 βαθμούς, ενώ για κάθε ισοπαλία να προσθέσουμε 1 βαθμό. Ένα τέτοιο ερώτημα στη βάση θα μπορούσε να έχει την ακόλουθη μορφή:

```
SELECT 3 * (
  SELECT COUNT(*)
  FROM   matches, teams, results
  WHERE  matches.result_id = results.id
  AND    results.name = 'Home'
  AND    matches.home_team_id = teams.id
  AND    teams.name = 'Arsenal')
+ 3 * (
  SELECT COUNT(*)
  FROM   matches, teams, results
  WHERE  matches.result_id = results.id
  AND    results.name = 'Away'
  AND    matches.away_team_id = teams.id
  AND    teams.name = 'Arsenal')
+ (
  SELECT COUNT(*)
  FROM   matches, teams, results
  WHERE  matches.result_id = results.id
  AND    results.name = 'Draw'
  AND    teams.name = 'Arsenal'
  AND (  matches.away_team_id = teams.id
  OR     matches.home_team_id = teams.id));
```

Κώδικας 3.1: Ερώτημα υπολογισμού των βαθμών μιας ομάδας

Όπως είναι προφανές το παραπάνω ερώτημα απαιτεί από τη βάση δεδομένων μια σειρά από υπολογισμούς για να απαντηθεί. Όσο μεγαλώνει ο αριθμός των χρηστών, τόσο ανεβαίνει το υπολογιστικό κόστος της χρήσης της εφαρμογής, αφού ακόμη και για τόσο απλή λειτουργία όσο η εμφάνιση των βαθμών μιας

ομάδας απαιτούνται μια σειρά από υπολογισμούς.

Εξετάζοντας, όμως, τη φύση της εφαρμογής, διαπιστώνουμε ότι όσο τα αποτελέσματα των αγώνων στη βάση μένουν αναλοίωτα, το αποτέλεσμα μιας ερώτησης όπως η παραπάνω θα είναι πάντοτε το ίδιο. Επομένως θα ήταν αποδοτικότερο τα αποτελέσματα ερωτήσεων που θα τίθενται συχνά στη βάση δεδομένων να αποθηκεύονται, ώστε η απάντησή τους να είναι υπόθεση ενός απλού ερωτήματος όπως το ακόλουθο:

```
SELECT group_points
FROM   teams
WHERE  name = 'Arsenal';
```

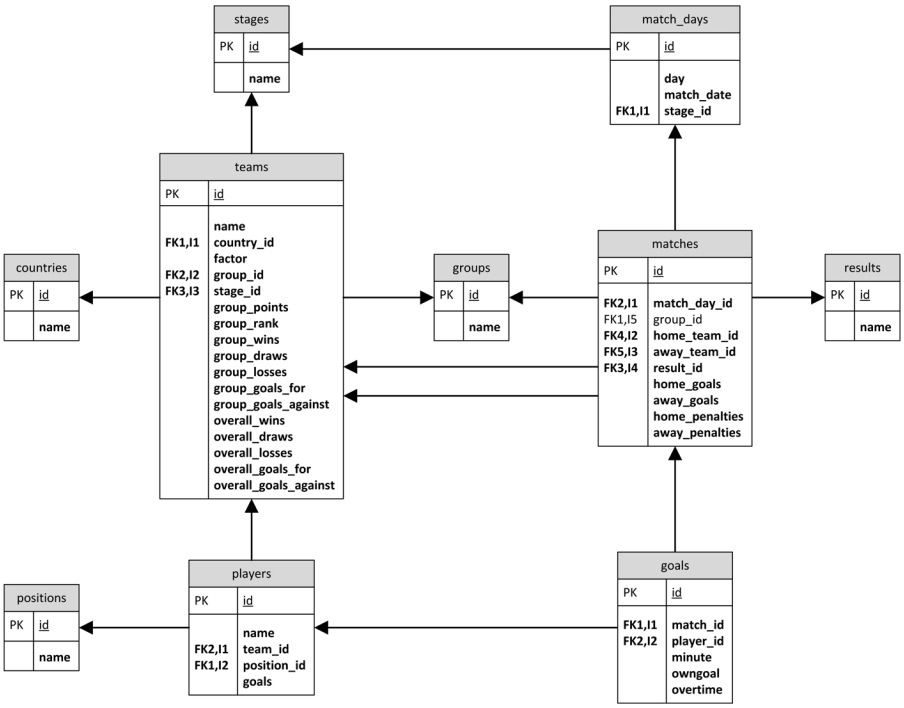
Κώδικας 3.2: Ερώτημα ανάκτησης των βαθμών μιας ομάδας

Για το σκοπό αυτό, εισάγουμε στη σχέση teams μια σειρά από πεδία τα οποία τηρούν χρήσιμα στατιστικά στοιχεία σχετικά με την απόδοση της ομάδας και ανανεώνονται μόνο μετά από ενημέρωση των αποτελεσμάτων των αγώνων. Έτσι, η σχέση teams μετά την εισαγωγή των πεδίων αυτών, παίρνει τη μορφή που απεικονίζεται στον πίνακα 3.9.

name	Το όνομα της ομάδας
country_id	Η χώρα προέλευσης της ομάδας *Foreign key στη σχέση countries*
factor	Ο συντελεστής απόδοσης της ομάδας
group_id	Ο όμιλος στον οποίο ανήκει η ομάδα *Foreign key στη σχέση groups*
stage_id	Φάση στην οποία βρίσκεται η ομάδα *Foreign key στη σχέση stages*
group_points	Οι βαθμοί της ομάδας
group_rank	Η σειρά κατάταξης στον όμιλο
group_wins	Οι νίκες στον όμιλο
group_draws	Οι ισοπαλίες στον όμιλο
group_losses	Οι ήττες στον όμιλο
group_goals_for	Γκολ υπέρ στον όμιλο
group_goals_against	Γκολ κατά στον όμιλο
overall_wins	Οι συνολικές νίκες
overall_draws	Οι συνολικές ισοπαλίες
overall_losses	Οι συνολικές ήττες
overall_goals_for	Γκολ υπέρ συνολικά
overall_goals_against	Γκολ κατά συνολικά

Πίνακας 3.9: Η σχέση teams, μέρος 3ο

Έχοντας αναλύσει τις σχέσεις που αντιστοιχούν στην πρώτη ενότητα του μοντέλου της βάσης δεδομένων, διαμορφώνεται το σχεσιακό διάγραμμα του σχήμα-



Σχήμα 3.5: Σχεσιακό διάγραμμα της βάσης δεδομένων, μέρος 1ο

τος 3.5, στο απεικονίζονται οι σχέσεις που μετέχουν στην ενότητα καθώς επίσης οι αναφορές που τις συνδέουν μέσω των foreign keys. Στη συνέχεια θα ασχοληθούμε με τη δεύτερη ενότητα του μοντέλου οντοτήτων-συσχετίσεων, το οποίο υποστηρίζει τη συμμετοχή των χρηστών στο παιχνίδι και στην εφαρμογή γενικότερα.

Η σχέση users, μέρος 1ο

Κατ' αντιστοιχία στην οντότητα user, δημιουργούμε τη σχέση users, η οποία, όπως και η οντότητα, έχει σαν βασικά γνωρίσματα το όνομα χρήστη (*username*), τον κωδικό εισόδου (*password*) και το *email* του χρήστη. Παρ' όλο που το όνομα χρήστη είναι μοναδικό, επιπλέον εισάγουμε το πεδίο *id*, το οποίο είναι ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση και επιτελεί το ρόλο του πρωτεύοντος κλειδιού. Τα πεδία της σχέσης απεικονίζονται συνοπτικά στον πίνακα 3.10.

Οι σχέσεις preds, races και stocks

Από την M:N συσχέτιση pred, δημιουργούμε τη σχέση preds, η οποία εκφράζει τη συμμετοχή των χρηστών στο ομώνυμο υπο-παιχνίδι και κάθε εγγραφή της οποίας αντιστοιχεί σε μια πρόβλεψη ενός χρήστη για το αποτέλεσμα ενός αγώνα.

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
username	Το όνομα χρήστη
password	Ο κωδικός εισόδου
email	Το email του χρήστη

Πίνακας 3.10: Η σχέση users, μέρος 1ο

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
user_id	Ο χρήστης που συμμετέχει στο παιχνίδι *Foreign key στη σχέση users*
match_id	Ο αγώνας στον οποίο αφορά η πρόβλεψη *Foreign key στη σχέση matches*
result_id	Η πρόβλεψη του αποτελέσματος *Foreign key στη σχέση results*

Πίνακας 3.11: Η σχέση preds

Για να εκφράσουμε τη συσχέτιση αυτή, εισάγουμε τα πεδία *user_id* και *match_id* τα οποία αναφέρονται στις σχέσεις **users** και **matches** και αποτελούν foreign keys στις σχέσεις αυτές. Επιπλέον, για να εκφράσουμε την πρόβλεψη του χρήστη, εισάγουμε το πεδίο *result_id*, το οποίο αναφέρεται στη σχέση **results** και επίσης αποτελεί foreign key στη σχέση αυτή. Τέλος, εισάγουμε όπως και στις προηγούμενες σχέσεις το πεδίο *id*, ένα μοναδικό αριθμητικό αναγνωριστικό, το οποίο αποτελεί το primary key. Τα πεδία της σχέσης **preds** απεικονίζονται στον πίνακα 3.11.

Ανάλογα, διαμορφώνονται και οι σχέσεις **races** και **stocks**. Η πρώτη αντιπροσωπεύει τις συμμετοχές των χρηστών στο υπο-παιχνίδι race και κάθε εγγραφή της αποτελεί μια πρόβλεψη ενός χρήστη για την πορεία μιας ομάδας στη διοργάνωση. Η πρόβλεψη αυτή εκφράζεται μέσω του πεδίου *stage_id*, το οποίο αναφέρεται στη σχέση **stages** και αποτελεί foreign key στη σχέση αυτή. Αντίστοιχα, η δεύτερη σχέση αντιπροσωπεύει τις συμμετοχές των χρηστών στο υπο-παιχνίδι stock και κάθε εγγραφή της αποτελεί ένα "ποντάρισμα" ενός αριθμού μαρκών από ένα παίκτη πάνω σε μια ομάδα. Ο αριθμός των μαρκών εκφράζεται με το πεδίο *amount*. Τα πεδία των σχέσεων αυτών απεικονίζονται συνοπτικά στους πίνακες 3.12 και 3.13.

Η σχέση bonuses

Επιστρέφοντας στο διάγραμμα οντοτήτων-συσχετίσεων του σχήματος 3.4, διαπιστώνουμε ότι το υπο-παιχνίδι bonus εκφράζεται μέσω των συσχετίσεων scorer, offense και defense. Οι συσχετίσεις αυτές είναι 1:N και αντιπροσωπεύουν τις επιλογές του χρήστη για τον χρήστη που είναι ο πρώτος σκόρερ, την

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
user_id	Ο χρήστης που συμμετέχει στο παιχνίδι *Foreign key στη σχέση users*
team_id	Η ομάδα στην οποία αφορά η πρόβλεψη *Foreign key στη σχέση teams*
stage_id	Η πρόβλεψη της πορείας της ομάδας *Foreign key στη σχέση stages*

Πίνακας 3.12: Η σχέση races

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
user_id	Ο χρήστης που συμμετέχει στο παιχνίδι *Foreign key στη σχέση users*
team_id	Η ομάδα στην οποία ποντάρει ο χρήστης *Foreign key στη σχέση teams*
amount	Ο αριθμός των μαρκών του ποντάρίσματος

Πίνακας 3.13: Η σχέση stocks

ομάδα με την καλύτερη επίθεση και την ομάδα με την καλύτερη άμυνα της διοργάνωσης. Μπορούμε να ομαδοποιήσουμε τις συσχετίσεις αυτές σε μια σχέση με το όνομα **bonuses**, η οποία θα εκφράζει τις συμμετοχές των χρηστών στο υποπαιχνίδι bonus και κάθε εγγραφή της οποίες θα περιέχει και τις τρεις επιλογές του χρήστη για το υπο-παιχνίδι αυτό μέσω των πεδίων *scorer_id*, *offense_team_id* και *defense_team_id* αντίστοιχα. Τα πεδία της σχέσης **bonuses** απεικονίζονται στον πίνακα 3.14.

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
user_id	Ο χρήστης που συμμετέχει στο παιχνίδι *Foreign key στη σχέση users*
offense_team_id	Η ομάδα με την καλύτερη επίθεση *Foreign key στη σχέση teams*
defense_team_id	Η ομάδα με την καλύτερη άμυνα *Foreign key στη σχέση teams*
scorer_id	Ο πρώτος σκόρερ *Foreign key στη σχέση players*

Πίνακας 3.14: Η σχέση bonuses

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
username	Το όνομα χρήστη
password	Ο κωδικός εισόδου
email	Το email του χρήστη
registered_on	Ημερομηνία εγγραφής
activation_code	Κωδικός ενεργοποίησης
activated	Αν ο χρήστης είναι ενεργός
played_pred	Συμμετοχή στο παιχνίδι pred
played_race	Συμμετοχή στο παιχνίδι race
played_stock	Συμμετοχή στο παιχνίδι stock
played_bonus	Συμμετοχή στο παιχνίδι bonus
points_pred	Βαθμοί στο παιχνίδι pred
points_race	Βαθμοί στο παιχνίδι race
points_stock	Βαθμοί στο παιχνίδι stock
points_bonus	Βαθμοί στο παιχνίδι bonus

Πίνακας 3.15: Η σχέση users, μέρος 2ο

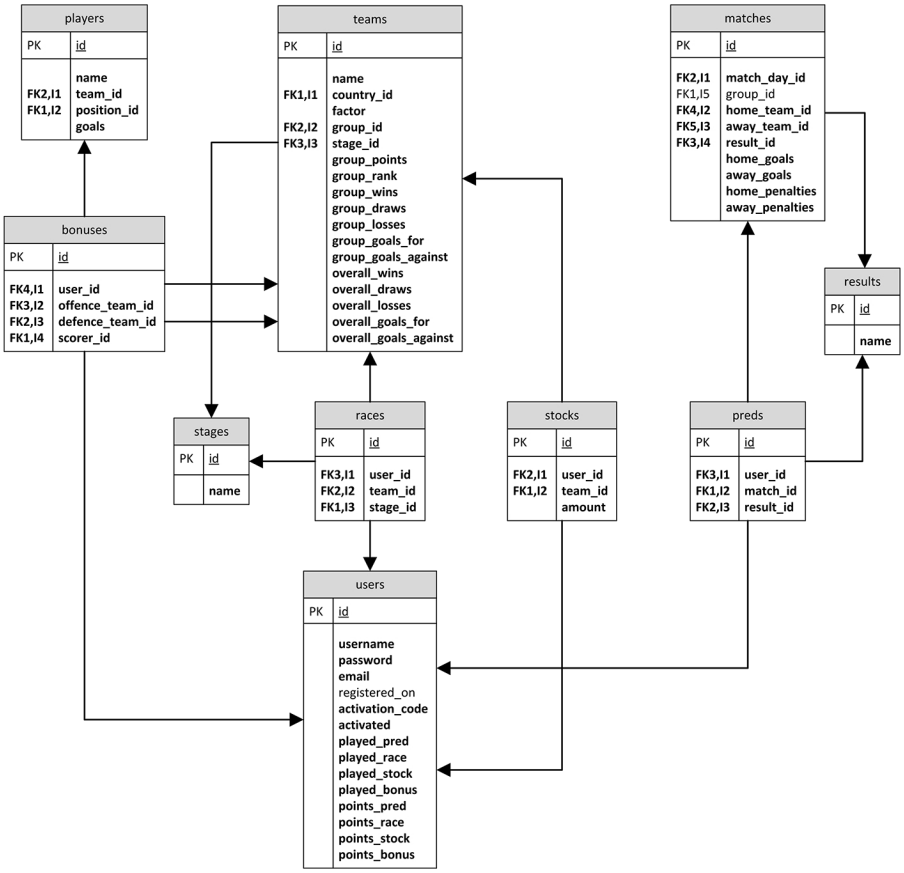
Η σχέση users, μέρος 2ο

Στον πυρήνα της εφαρμογής βρίσκονται οι συμμετοχές των χρηστών στα διάφορα παιχνίδια και ο συναγωνισμός για τη διάκριση σε αυτά. Η κατάταξη των χρηστών με βάση τους βαθμούς που συγκεντρώνουν ανάλογα με την επιτυχία των προβλέψεών τους και σύμφωνα με τους κανόνες που αναλύσαμε στο κεφάλαιο 2. Όπως και με τα στοιχεία των επιδόσεων των ομάδων στη σχέση teams, έτσι και οι βαθμολογίες των χρηστών προκύπτουν από μια σειρά υπολογισμών το αποτέλεσμα των οποίων εξαρτάται αποκλειστικά από τα αποτελέσματα των αγώνων που έχουν εισαχθεί στη βάση. Λόγω της αραιής συχνότητας των ενημερώσεων των αποτελεσμάτων, θα ήταν αποδοτικότερο οι βαθμολογίες των παικτών σε κάθε παιχνίδι να υπολογίζονται μόνο μια φορά μετά από κάθε ενημέρωση και να αποθηκεύονται, παρά οι υπολογισμοί αυτοί να πραγματοποιούνται κάθε φορά που αυτό απαιτείται.

Για το λόγο αυτό εισάγουμε τα πεδία *points_pred*, *points_race*, *points_stock* και *points_bonus* στα οποία θα αποθηκεύονται οι βαθμοί που έχει συγκεντρώσει ο χρήστης στα αντίστοιχα υπο-παιχνίδια. Επιπλέον, εισάγουμε στη σχέση τα πεδία *played_pred*, *played_race*, *played_stock* και *played_bonus* στα οποία καταγράφεται αν ο χρήστης έχει ολοκληρώσει τη διαδικασία συμμετοχής στα αντίστοιχα παιχνίδια. Η χρησιμότητα των πεδίων αυτών θα γίνει αντιληπτή σε επόμενες ενότητες.

Τέλος, εισάγουμε τα πεδία *registered_on*, *activation_code* και *activated* στα οποία θα αναφερθούμε στην ενότητα 5.3.1. Η σχέση **users**, όπως αυτή διαμορφώνεται μετά τις παραπάνω προσθήκες, απεικονίζεται στον πίνακα 3.15.

Έχοντας ολοκληρώσει την αναφορά μας στις σχέσεις που αντιστοιχούν στη



Σχήμα 3.6: Σχεσιακό διάγραμμα της βάσης δεδομένων, μέρος 2ο

δεύτερη ενότητα του μοντέλου οντοτήτων-συσχετίσεων, δημιουργούμε το σχεσιακό διάγραμμα για την ενότητα αυτή στο σχήμα 3.6. Σημειώνουμε ότι για λόγους απλότητας από το διάγραμμα αυτό έχουν παραλειφθεί οι αναφορές λόγω των foreign keys που παρουσιάστηκαν στο διάγραμμα 3.5.

Ο πλήρης κώδικας δημιουργίας της βάσης δεδομένων της εφαρμογής περιλαμβάνεται στο παράρτημα Α΄. Στην επόμενη ενότητα θα εξετάσουμε τη γλώσσα PHP και τον τρόπο με τον οποίο συνεργάζεται με το σύστημα βάσης δεδομένων για τη δημιουργία της λειτουργικότητας της εφαρμογής.

3.3 PHP: Πρόσβαση στα δεδομένα

Όπως είδαμε στην ενότητα 3.1, τα δύο βασικά χαρακτηριστικά μιας δυναμικής online εφαρμογής είναι το σύστημα βάσεων δεδομένων και ο PHP interpreter, ο οποίος αναλαμβάνει να μεταφράσει τις εντολές PHP και να δημιουργήσει τα τελικά αρχεία που λαμβάνονται και απεικονίζονται στους browsers των χρηστών

της εφαρμογής. Στην ενότητα αυτή θα εξετάσουμε αναλυτικά τη γλώσσα προγραμματισμού PHP, τον τρόπο με τον οποίο συνεργάζεται με το σύστημα βάσεων δεδομένων MySQL και τα οφέλη που προκύπτουν από την εφαρμογή της αντικειμενοστρέφειας στην ανάπτυξη μιας εφαρμογής.

3.3.1 Η γλώσσα προγραμματισμού PHP

Η γλώσσα προγραμματισμού PHP (PHP: Hypertext Preprocessor) είναι μια γλώσσα δημιουργίας κώδικα ο οποίος ενσωματώνεται σε κώδικα HTML (στην ενότητα 4.1 θα αναλύσουμε τη γλώσσα HTML και τις δυνατότητες της διεξοδικά) και προσφέρει τη δυνατότητα τροποποίησης του περιεχομένου της σελίδας ανάλογα με μια σειρά από παραμέτρους. Μπορούμε να συνοψίσουμε τα βασικά πλεονεκτήματα της γλώσσας PHP στα εξής:

- Η PHP και η MySQL δημιουργήθηκαν εξ αρχής για να συνεργάζονται αρμονικά μεταξύ τους. Έτσι, η PHP προσφέρει μια σειρά από λειτουργίες, οι οποίες εκμεταλλεύονται στο έπακρο τις δυνατότητες της MySQL.
- Η PHP είναι τεχνολογία ανοιχτού κώδικα και επομένως η χρήση της είναι δωρεάν.
- Η PHP έχει μια ευρεία βάση χρηστών και μια μεγάλη κοινότητα ενεργών χρηστών οι οποίοι συνεισφέρουν στη διόρθωση προβλημάτων, αλλά και στην περαιτέρω ανάπτυξη της γλώσσας.
- Η PHP είναι μια εξαιρετικά διαδεδομένη πλατφόρμα για την ανάπτυξη web εφαρμογών και είναι διαθέσιμη στην πλειοψηφία των web hosts ανά τον κόσμο.

3.3.2 Επικοινωνία με τη βάση δεδομένων

Μια από τις βασικότερες λειτουργίες της γλώσσας PHP είναι η αλληλεπίδραση με τη βάση δεδομένων και η δυνατότητα που προσφέρει στον προγραμματιστή να πραγματοποιεί ερωτήματα και να λαμβάνει τα αποτελέσματα της ερώτησης σε μορφή κατάλληλη για επεξεργασία. Η διαδικασία μιας τέτοιας αλληλεπίδρασης με τη βάση περιλαμβάνει τα εξής στάδια:

1. Πραγματοποίηση της σύνδεσης με το σύστημα βάσεων δεδομένων. Η σύνδεση αυτή πραγματοποιείται με τη χρήση της μεθόδου `mysql_connect`, η οποία δέχεται τρεις παραμέτρους: *host* (το όνομα του υπολογιστή που φιλοξενεί της βάση δεδομένων), *username* και *password* (το όνομα χρήστη και ο κωδικό εισόδου ενός ατόμου με δικαιώματα χρήσης της βάσης δεδομένων).
2. Επιλογή της κατάλληλης βάσης δεδομένων. Η επιλογή αυτή πραγματοποιείται με τη χρήση της μεθόδου `mysql_select_db`, η οποία δέχεται σαν παράμετρο το όνομα της βάσης δεδομένων *db_name*.

```php
// 1.Connect
$connection = mysql_connect($db_host, $db_user, $db_pass);
if (!$connection) {
  die ("Could not connect to the database: " . mysq_error());
}
// 2. Select database
$database = mysql_select_db($db_database);
if (!$database) {
  die ("Could not select the database: " . mysql_error());
}
// 3. Perform the query
$query = "SELECT * FROM users";
$result = mysql_query($query);
if (!$result) {
  die ("Could not query the database" . mysql_error());
}
// 4. Process the results
while ($row = mysql_fetch_array($result)) {
  echo "Name: " . $row["username"] . <br />;
}
// 5. Close the connection
mysql_close($connection);
```

Κώδικας 3.3: Ανάκτηση των χρηστών της εφαρμογής με χρήση PHP

3. Εκτέλεση ενός ερωτήματος στη βάση. Η υποβολή ερωτημάτων γίνεται με τη χρήση της μεθόδου `mysql_query`, η οποία δέχεται σαν παράμετρο το ερώτημα που θέλουμε να υποβάλουμε σε μορφή κώδικα SQL και επιστρέφει τα αποτελέσματα της ερωτήματος.

4. Επεξεργασία των αποτελεσμάτων του ερωτήματος. Ο πιο συνηθισμένος τρόπος εξαγωγής των αποτελεσμάτων ενός ερωτήματος είναι η επεξεργασία κάθε τούπλας ξεχωριστά με τη βοήθεια της προκαθορισμένης από τη γλώσσα μεθόδου `mysql_fetch_array`, η οποία επιστρέφει μια γραμμή σε μορφή συσχετιστικού πίνακα.

5. Τερματισμός της σύνδεσης με χρήση της μεθόδου `mysql_close`.

Ένα παράδειγμα της διαδικασίας αυτής εμφανίζεται στον κώδικα 3.3, στο οποίο φέρνουμε από τη βάση τα ονόματα όλων των χρηστών που έχουν εγγραφεί στην εφαρμογή.

Η διεξοδική παρουσίαση της γλώσσας PHP ξεφεύγει από την εμβέλεια της παρούσας διπλωματικής εργασίας. Όμως αξίζει να σημειώσουμε ορισμένες παρατηρήσεις σχετικές με το παραπάνω απόσπασμα κώδικα, οι οποίες θα μας φανούν χρήσιμες στη συνέχεια:

- Η μέθοδος `mysql_fetch_array` δέχεται σαν όρισμα το αποτέλεσμα ενός ερωτήματος στη βάση δεδομένων και επιστρέφει ένα συσχετιστικό πίνακα

που αντιστοιχεί σε μια γραμμή του αποτελέσματος. Ο πίνακας αυτός αποτελείται από ζεύγη κλειδιού-τιμής (key-value pairs) με τα κλειδιά να αντιστοιχούν στα ορίσματα των πινάκων στη βάση δεδομένων που αντιστοιχούν στο συγκεκριμένο ερώτημα και τις τιμές να αντιστοιχούν στις αντίστοιχες τιμές των εγγραφών των πινάκων αυτών. Έτσι, στο παραπάνω παράδειγμα, ο όρος `$row["username"]` αντιστοιχεί στο όνομα ενός χρήστη, ενώ για να πάρουμε το email του χρήστη αυτού θα πρέπει να χρησιμοποιήσουμε τον όρο `$row["email"]`.

- Τα παραπάνω 5 βήματα της διαδικασίας είναι υποχρεωτικά για οποιαδήποτε αλληλεπίδραση με τη βάση δεδομένων. Ουσιαστικά τα βήματα 1,2 και 5 είναι κοινά σε οποιαδήποτε αλλελεπίδραση με τη βάση δεδομένων.

- Το ίδιο το ερώτημα που τίθεται στη βάση δεδομένων από τη σκοπιά της γλώσσας PHP είναι απλά ένα string, ή αλλιώς μια σειρά χαρακτήρων. Το γεγονός αυτό προσφέρει τη δυνατότητα να δημιουργούνται δυναμικά ερωτήματα με βάση κάποια παράμετρο. Για παράδειγμα, μπορούμε να δημιουργήσουμε ερωτήματα της μορφής `SELECT * FROM teams WHERE name LIKE '$param'`, όπου ο όρος `$param` αντιστοιχεί σε μια μεταβλητή, η τιμή της οποίας τροποποιεί το ερώτημα κάθε φορά.

- Η PHP προσφέρει μια σειρά από έτοιμες μεθόδους για την αλληλεπίδραση με τη βάση δεδομένων, το όνομα των οποίων δηλώνει ότι έχουν δημιουργηθεί για χρήση με το σύστημα βάσεων δεδομένων MySQL. Αντίστοιχες μέθοδοι παρέχονται και για άλλα συστήματα βάσεων δεδομένων, με ονόματα επίσης δηλωτικά του αντίστοιχου συστήματος.

3.3.3 PHP και DRY

Μια από τις βασικότερες αρχές της σύγχρονης ανάπτυξης εφαρμογών είναι γνωστή με τα αρχικά DRY, τα οποία αντιστοιχούν στις λέξεις Don't Repeat Yourself. Σε ελεύθερη μετάφραση, ο προγραμματισμός σύμφωνα με την αρχή DRY έχει σαν στόχο την ελαχιστοποίηση της επανάληψης κώδικα σε όλη την έκταση της εφαρμογής. Αντί να ξαναγράφουμε τις ίδιες εντολές σε διαφορετικά σημεία της εφαρμογής θα ήταν αποδοτικότερο να προσπαθήσουμε να επαναχρησιμοποιήσουμε τον κώδικα που ήδη έχουμε γράψει. Στο παραπάνω στοιχειώδες παράδειγμα κώδικα PHP, όπως αναφέραμε στις παρατηρήσεις, κάθε ερώτημα στη βάση δεδομένων θα πρέπει να περιλαμβάνει τα βήματα 1 έως 5. Έτσι, αν έχουμε μια σελίδα η οποία εμφανίζει μια λίστα με τις ομάδες που μετέχουν στη διοργάνωση (*teams.php*) και μια δεύτερη σελίδα η οποία εμφανίζει τα στοιχεία των παικτών που αγωνίζονται στη διοργάνωση (*teams.php*), τότε και οι δύο σελίδες θα περιέχουν ακριβώς τις ίδιες γραμμές κώδικα για τα βήματα 1,2 και 5.

Προφανώς θα ήταν αποδοτικότερο αν μπορούσαμε να αποφύγουμε αυτήν την επανάληψη. Η λύση στη γλώσσα PHP προσφέρεται μέσω της εντολής `require`, η οποία μας δίνει τη δυνατότητα να ενσωματώσουμε τον κώδικα ενός αρχείου σε

```
// Get required file
require("database_init.php");
// 3. Perform the query
$query  = "SELECT * FROM teams ";
$query .= "WHERE group_id=" . $group_id;
$result = mysql_query($query);
if (!$result) {
  die ("Could not query the database" . mysql_error());
}
// 4. Process the results
while ($row = mysql_fetch_array($result)) {
  echo "Name: " . $row["name"] . <br />;
}
// 5. Close the connection
mysql_close($connection);
```

Κώδικας 3.4: Ανάκτηση των χρηστών της εφαρμογής με χρήση require

ένα άλλο. Έτσι, δημιουργώντας ένα αρχείο που περιλαμβάνει τον κώδικα των βημάτων 1 και 2 μπορούμε να επαναχρησιμοποιούμε τις λειτουργίες αυτές σε οποιοδήποτε σημείο της εφαρμογής χρειάζεται. Η λύση αυτή είναι ένα βήμα στη σωστή κατεύθυνση και μπορεί από μόνη της να υποστηρίξει μια εφαρμογή μερικών αρχείων. Τί συμβαίνει όμως όταν πρόκειται για μια εφαρμογή δεκάδων ή και εκατοντάδων αρχείων; Με ποιό τρόπο μπορούμε να αποφύγουμε την επανάληψη των επόμενων βημάτων της παραπάνω διαδικασίας; Η απάντηση στα ερωτήματα αυτά δίνεται από την εισαγωγή αντικειμένων, την οποία θα εξετάσουμε στη συνέχεια.

3.3.4 PHP και αντικειμενοστρέφεια

Ας υποθέσουμε ότι για τις ανάγκες της εφαρμογής μας έχουμε διαμορφώσει ένα κομμάτι κώδικα το οποίο ανακτά από τη βάση τις ομάδες που ανήκουν σε ένα συγκεκριμένο όμιλο. Το απόσπασμα αυτό θα μπορούσε να έχει τη μορφή του κώδικα 3.4.

Η μεταβλητή `$group_id` είναι η παράμετρος που τροποποιεί το ερώτημα στη βάση. Αν δύο ή περισσότερες σελίδες περιέχουν την ίδια λειτουργία, τότε είμαστε υποχρεωμένοι να επαναλάβουμε ακριβώς το ίδιο κομμάτι κώδικα. Το γεγονός αυτό από μόνο του αφαιρεί από την ευελιξία της εφαρμογής και δημιουργεί τις συνθήκες για την εμφάνιση προβλημάτων τα οποία στο μέλλον θα είναι δύσκολότερο να εντοπιστούν και να επιλυθούν.

Η λύση σε αυτό το ζήτημα προσφέρεται από την αντικειμενοστρέφεια, η οποία από την έκδοση 5.0 και μετά προσφέρεται από την PHP και την οποία χρησιμοποιήσαμε εκτεταμένα κατά την υλοποίηση της εφαρμογής. Σύμφωνα με την αρχή της αντικειμενοστρέφειας, κάθε τί στον προγραμματισμό μπορεί να θεωρηθεί σαν αντικείμενο, το οποίο έχει τα δικά του χαρακτηριστικά και προσφέρει τις

δικές του λειτουργίες. Στη γλώσσα PHP τα αντικείμενα εκφράζονται μέσω των κλάσεων, τα χαρακτηριστικά μέσω των γνωρισμάτων (attributes) και οι λειτουργίες μέσω των μεθόδων (methods).

Η κλάση της βάσης δεδομένων

Ένα από τα πρώτα αντικείμενα που θα δημιουργήσουμε είναι η κλάση της βάσης δεδομένων. Το αντικείμενο αυτό θα βρίσκεται στον πυρήνα της εφαρμογής, καθώς θα εξυπηρετεί μια πληθώρα λειτουργιών, όπως θα δούμε στη συνεχεια. Οι βασικές λειτουργίες της κλάσης αυτής, θα είναι η δυνατότητα δημιουργίας και τερματισμού συνδέσεων και η δυνατότητα εκτέλεσης ερωτημάτων. Ο κώδικας της κλάσης της βάσης δεδομένων απεικονίζεται στο απόσπασμα 3.5.

Παρατηρώντας τον ορισμό της κλάσης αυτής, θα σημειώσουμε ότι στα αποσπάσματα κώδικα που παραθέσαμε προηγουμένως, χρησιμοποιήσαμε έτοιμες μεθόδους που προσφέρονται από τη γλώσσα PHP για την εκτέλεση κάποιων λειτουργιών σχετικών με τη βάση δεδομένων. Οι μέθοδοι αυτές έχουν ονόματα που περιλαμβάνουν το πρόθεμα `mysql`, και το οποίο είναι δηλωτικό της χρήσης του συστήματος βάσεων δεδομένων MySQL από την εφαρμογή μας. Για την περίπτωση που στο μέλλον αποφασίσουμε να μεταπηδήσουμε σε κάποιο άλλο σύστημα βάσης δεδομένων (όπως για παράδειγμα ο Microsoft SQL Server), ορίζουμε την κλάση μας με το εξειδικευμένο όνομα `MySQLDatabase` και τις μεθόδους που προσφέρονται από την PHP για την αλληλεπίδραση με τη βάση δεδομένων με ονόματα γενικευμένα, τα οποία δεν περιέχουν κάποιο δηλωτικό του συγκεκριμένου συστήματος που χρησιμοποιείται.

Στην τελευταία εντολή του κώδικα 3.5 ορίζουμε το αντικείμενο της βάσης δεδομένων `$database`, το οποίο δημιουργείται με βάση την κλάση που ορίσαμε και παρέχει όλη τη λειτουργικότητα της βάσης αυτής. Όλες οι λειτουργίες που πραγματοποιούνται από την εφαρμογή και σχετίζονται με τη βάση δεδομένων, μπορούν να πραγματοποιηθούν μέσω του αντικειμένου αυτού. Με τον τρόπο αυτό επιτυγχάνουμε μια αφαίρεση της λειτουργικότητας και την ανεξαρτοποίηση της εφαρμογής από ένα συγκεκριμένο σύστημα βάσεων δεδομένων. Αν λοιπόν στο μέλλον αποφασίσουμε να μεταπηδήσουμε από το σύστημα MySQL στο σύστημα Microsoft SQL Server, το μόνο που απαιτείται για να λειτουργήσει σωστά η εφαρμογή μας, είναι η δημιουργία μιας κλάσης βάσης δεδομένων για το νέο σύστημα, η οποία να υλοποιεί τις ίδιες μεθόδους με την κλάση που δημιουργήσαμε παραπάνω και η δημιουργία ενός αντικειμένου με το όνομα `$database` με βάση τη νέα κλάση.

3.3.5 Βασικές κλάσεις της εφαρμογής

Στο κεφάλαιο αυτό θα προχωρήσουμε στον ορισμό των βασικών κλάσεων που χρησιμοποιούνται από την εφαρμογή. Οι κλάσεις αυτές χρησιμοποιούν την κλάση της βάσης δεδομένων που ορίσαμε προηγουμένως για τις λειτουργίες που

```php
<?php
require_once("database.config.php");

class MySQLDatabase {
  private $connection;

  function __construct() {
    $this->open_connection();
  }
  public function open_connection() {
    $this->connection = mysql_connect(DB_HOST, DB_USER, DB_PASS);
      if (!$this->connection) {
        die("Database connection failed: " . mysql_error());
      } else {
        $db_select = mysql_select_db(DB_NAME, $this->connection);
        if (!$db_select) {
          die("Database selection failed: " . mysql_error());
        }
    }
  }
  public function close_connection() {
    if(isset($this->connection)) {
      mysql_close($this->connection);
      unset($this->connection);
    }
  }
  public function query($sql) {
    $result = mysql_query($sql, $this->connection);
    return $result;
  }
  public function fetch_array($result_set) {
    return mysql_fetch_array($result_set);
  }
  public function num_rows($result_set) {
    return mysql_num_rows($result_set);
  }
  public function insert_id() {
    return mysql_insert_id($this->connection);
  }
  public function affected_rows() {
    return mysql_affected_rows($this->connection);
  }
}
$database = new MySQLDatabase();
?>
```

Κώδικας 3.5: Ορισμός της κλάσης MySQLDatabase

σχετίζονται με τη βάση δεδομένων. Γενικά, ο ορισμός μιας κλάσης στην PHP περιλαμβάνει τρία στοιχεία:

- Το όνομα της κλάσης
- Τα γνωρίσματα (attributes) της κλάσης
- Τις μεθόδους (methods) της κλάσης

Όμως, καθώς οι κλάσεις που θα ορίσουμε αντιστοιχούν σε σχέσεις ή πίνακες μιας βάσεις δεδομένων, μπορούμε να συγκεκριμενοποιήσουμε της δομή και τη λειτουργικότητα των κλάσεών μας ακόμη περισσότερο:

- Το όνομα της κλάσης
- Τα γνωρίσματα (attributes) της κλάσης που *αντιστοιχούν* στα γνωρίσματα της αντίστοιχης σχέσης στη βάση δεδομένων
- Τις μεθόδους (methods) της κλάσης που είναι εξειδικευμένες για την κλάση αυτή
- Τέσσερις βασικές μεθόδους που είναι κοινές σε όλα τα αντικείμενα που αντιστοιχούν σε σχέση στη βάση δεδομένων και έχουν ονόματα δηλωτικά της λειτουργικότητάς τους: *find_all*, *create*, *update* και *delete*.

Για να γίνει πιο κατανοητή η δομή αυτή, θα εξετάσουμε ενδεικτικά την κλάση `Match`, η οποία δημιουργήθηκε σε αντιστοιχία με την οντότητα match και κατ' επέκταση τη σχέση *matches* στη βάση δεδομένων. Υπενθυμίζουμε ότι τα πεδία του πίνακα matches εμφανίζονται στον πίνακα 3.7. Έτσι, και η κλάση `Match` θα έχει τα αντίστοιχα γνωρίσματα όπως φαίνεται στο ακόλουθο απόσπασμα κώδικα:

```
class Match {
  public $id;
  public $match_day_id;
  public $group_id = 0;
  public $home_team_id;
  public $away_team_id;
  public $result_id = 1;
  public $home_goals = 0;
  public $away_goals = 0;
  public $home_penalties = 0;
  public $away_penalties = 0;
}
```

Κώδικας 3.6: Ορισμός της κλάσης Match

Σχετικά με την αρχικοποίηση των τιμών των attributes, σημειώνουμε ότι η εγγραφή στον πίνακα *results* με id=1 δηλώνει ότι το αποτέλεσμα ενός αγώνα δεν έχει γίνει ακόμη γνωστό.

Έχοντας ορίσει τα γνωρίσματα της κλάσης, στη συνέχεια θα προχωρήσουμε στον ορισμό των μεθόδων της κλάσης. Όπως αναφέραμε παραπάνω, υπάρχουν 4 βασικές μέθοδοι οι οποίες είναι κοινές σε όλες τις κλάσεις που αντιστοιχούν σε αντικείμενα στη βάση δεδομένων. Στη συνέχεια θα εξετάσουμε τις μεθόδους αυτές αναλυτικά:

Οι μέθοδοι find

Οι μέθοδοι αυτές αντιστοιχούν στην πράξη SELECT της SQL και χρησιμοποιούνται για την ανάκτηση εγγραφών από τον πίνακα matches της βάσης δεδομένων. Ας υποθέσουμε για παράδειγμα ότι επιθυμούμε να ανακτήσουμε όλους τους αγώνες από τη βάση και στη συνέχεια να εκτυπώσουμε τη γηπεδούχο και τη φιλοξενούμενη ομάδα και στη συνέχεια το αποτέλεσμα του αγώνα. Ο κώδικας για αυτή τη λειτουργία, χωρίς τη χρήση της κλάσης Match, θα είχε τη μορφή:

```
$sql = "SELECT * FROM matches";
$result = $database->query($sql);

while ($row = mysql_fetch_array($result)) {
  $home_team_id = $row["home_team_id"];
  $away_team_id = $row["away_team_id"];
  // Find the home team
  $sql1  = "SELECT * FROM teams ";
  $sql1 .= "WHERE id =" . $home_team_id;
  $result1 = $database->query($sql1);
  $row1 = mysql_fetch_array($result1);
  $home_team = $row1["name"];
  // Find the away team
  $sql2  = "SELECT * FROM teams ";
  $sql2 .= "WHERE id =" . $away_team_id;
  $result2 = $database->query($sql2);
  $row2 = mysql_fetch_array($result2);
  $away_team = $row2["name"];
  // Print the match details
  echo $home_team . " - " . $away_team . " ";
  echo $row["home_goals"] . "-" . $row["away_goals"];
}
```

Κώδικας 3.7: Παράδειγμα λειτουργίας ανάκτησης εγγραφών

Διαπιστώνουμε ότι ακόμη και για μια τόσο απλή λειτουργία απαιτούνται αρκετές γραμμές κώδικα. Επιπλέον, οι εγγραφές που επιστρέφονται μέσω της εκτέλεσης της ερώτησης στη βάση, μέσω της μεθόδου `query` της κλάσης της βάσης δεδομένων που ορίσαμε προηγουμένως, είναι συσχετιστικοί πίνακες με τα ονόματα των γνωρισμάτων του πίνακα να παίζουν το ρόλο των κλειδιών και τα ίδια τα δεδομένα να παίζουν το ρόλο των τιμών.

Όμως στον ορισμό της κλάσης `Match` έχουμε ήδη δημιουργήσει τα αντίστοιχα γνωρίσματα και στην κλάση. Επομένως θα ήταν πιο ευέλικτο αντί να λαμβάνουμε μια σειρά από συσχετιστικούς πίνακες, να λαμβάνουμε μια συλλογή από **αντικείμενα** της κλάσης `Match`. Για το λόγο αυτό, δημιουργούμε τη μέθοδο *instantiate*,

η οποία αναλαμβάνει να μετατρέψει μια εγγραφή του αποτελέσματος σε ένα αντικείμενο της κλάσης. Όπως θα δούμε στη συνέχεια, η μέθοδος αυτή είναι πολύ χρήσιμη και πάνω σε αυτήν θα στηριχθούν όλες οι μέθοδοι ανάκτησης δεδομένων. Ο κώδικας της μεθόδου *instantiate* είναι ο εξής:

```
private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
  if($object->has_attribute($attribute)) {
    $object->$attribute = $value;
    }
  }
  return $object;
  }

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}
```

Κώδικας 3.8: Η μέθοδος instantiate

Πλέον, μπορούμε να χρησιμοποιήσουμε τη μέθοδο αυτή για να ορίσουμε τις μεθόδους find για την κλάση. Η πρώτη, και βασικότερη μέθοδος, ονομάζεται `find_by_sql` και εκτελεί το ερώτημα που δέχεται ως παράμετρο. Ο κώδικας της μεθόδου είναι ο εξής:

```
public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}
```

Κώδικας 3.9: Η μέθοδος find_by_sql

Από το απόσπασμα αυτό διαπιστώνουμε ότι η μέθοδος αυτή χρησιμοποιεί τη μέθοδο `instantiate` που ορίσαμε προηγουμένως για μετατρέψει τις εγγραφές που επιστρέφονται από το ερώτημα σε μια συλλογή αντικειμένων της κλάσης. Δύο πολύ συχνές περιπτώσεις ερωτημάτων που τίθενται στη βάση δεδομένων, είναι η ανάκτηση όλων των εγγραφών ενός πίνακα και η ανάκτηση μιας μόνο εγγραφής με βάση το πρωτεύον κλειδί του πίνακα. Οι σχέσεις αυτές ονομάζονται `find_all` και `find_by_id` αντίστοιχα και ουσιαστικά αποτελούν εξειδικεύσεις της `find_by_sql` που ορίσαμε προηγουμένως. Έτσι, ο ορισμός των μεθόδων αυτών, με τη βοήθεια της γενικότερης μεθόδου που ορίσαμε παραπάνω, απεικονίζεται στο απόσπασμα 3.10.

Το αποτέλεσμα της εκτέλεσης των μεθόδων find είναι αντικείμενα της κλάσης Match με τα ορίσματα του κάθε αντικειμένου να έχουν τις τιμές των αντίστοιχων

```
public static function find_all() {
  $sql = "SELECT * FROM matches";
  return self::find_by_sql($sql);
}

public static function find_by_id($id=0) {
  $sql = "SELECT * FROM matches WHERE id={$id} LIMIT 1"
  $result_array = self::find_by_sql($sql);
  return !empty($result_array)? array_shift($result_array):false;
}
```

Κώδικας 3.10: Οι μέθοδοι find_all και find_by_id

πεδίων του πίνακα matches στη βάση δεδομένων. Έτσι, αν υποθέσουμε ότι και η κλάση `Teams` διαθέτει αντίστοιχες find μεθόδους, το παράδειγμα κώδικα που εξετάσαμε στην αρχή της παρούσας ενότητας μπορεί να ξαναγραφεί ως εξής:

```
$matches = Match::find_all();

foreach ($matches as $match) {
  $home_team = Teams::find_by_id($match->home_team_id);
  $away_team = Teams::find_by_id($match->away_team_id);
  echo $home_team->name."-".$away_team->name." ";
  echo $match->home_goals."-".$match->away_goals;
}
```

Κώδικας 3.11: Παράδειγμα λειτουργίας ανάκτησης αντικειμένων

Είναι προφανές ότι το απόσπασμα αυτό κώδικα είναι πολύ πιο ευανάγνωστο και απλοποιεί σημαντικά τη σύνθεση του κώδικα των διαφόρων λειτουργιών της εφαρμογής.

Πριν προχωρήσουμε στις επόμενες βασικές μεθόδους θα θέλαμε να σημειώσουμε ότι οι μέθοδοι find είναι `static` μέθοδοι, ανήκουν δηλαδή στην κλάση και όχι σε κάποιο αντικείμενο αυτής. Αντίθετα, οι επόμενες μέθοδοι απαιτούν την ύπαρξη ενός αντικειμένου για να εκτελεστούν.

Η μέθοδος create

Η μέθοδος αυτή αντιστοιχεί στην πράξη `INSERT` της SQL και χρησιμοποιείται για την εισαγωγή νέων εγγραφών στον πίνακα matches της βάσης δεδομένων. Ο κώδικας της μεθόδου απεικονίζεται στο απόσπασμα 3.12.

Ουσιαστικά η μέθοδος *create* αναλαμβάνει να εκτελέσει μια παραμετροποιημένη εισαγωγή στον αντίστοιχο πίνακα με βάση τα πεδία του πίνακα και κατ' επέκταση με βάση τα ορίσματα του αντικειμένου. Η διαδικασία που ακολουθείται εδώ μπορεί να αναλυθεί στα εξής βήματα:

- Ένα νέο αντικείμενο της κλάσης `Match` δημιουργείται. Αρχικά τα ορίσματά του έχουν τις default τιμές, για όποια από αυτά παρέχονται τέτοιες.

```
public function create() {
  global $database;

  $sql  = "INSERT INTO matches ";
  $sql .= "(match_day_id, group_id, home_team_id, away_team_id,
            result_id, home_goals, away_goals, home_penalties,
            away_penalties) ";
  $sql .= "VALUES (";
  $sql .= "'".$database->escape_value($this->match_day_id)."', ";
  $sql .= "'".$database->escape_value($this->group_id)."', ";
  $sql .= "'".$database->escape_value($this->home_team_id)."', ";
  $sql .= "'".$database->escape_value($this->away_team_id)."', ";
  $sql .= "'".$database->escape_value($this->result_id)."', ";
  $sql .= "'".$database->escape_value($this->home_goals)."', ";
  $sql .= "'".$database->escape_value($this->away_goals)."', ";
  $sql .= "'".$database->escape_value($this->home_penalties)."',
      ";
  $sql .= "'".$database->escape_value($this->away_penalties)."')";
  if($database->query($sql)) {
    $this->id = $database->insert_id();
    return true;
  } else {
    return false;
  }
}
```

Κώδικας 3.12: Η μέθοδος create

- Οι τιμές των ορισμάτων του μεταβάλλονται σύμφωνα με το πρόγραμμα. Στο σημείο αυτό για παράδειγμα, οι τιμές θα μπορούσαν να προέρχονται από την εισαγωγή δεδομένων σε μια φόρμα.
- Η μέθοδος `create` εκτελείται μέσω του αντικειμένου αυτού και μια νέα εγγραφή εισάγεται στη βάση δεδομένων.
- Η τιμή του πεδίου id (το οποίο στις περισσότερες περιπτώσεις είναι και το πρωτεύον κλειδί με την ιδιότητα auto-increment) λαμβάνεται από τη βάση και αποδίδεται στο αντικείμενο, ώστε να είναι διαθέσιμη για επόμενες λειτουργίες.

Η μέθοδος `escape_value` της κλάσης της βάσης δεδομένων εξετάζεται αναλυτικά στην ενότητα **??**. Προς το παρόν θα αρκεστούμε να αναφέρουμε ότι προετοιμάζει τις τιμές για χρήση με τη βάση δεδομένων και αποτελεί βασική δικλείδα ασφαλείας για την τήρηση της βάσης σε ασφαλή κατάσταση.

Η μέθοδος update

Σε αναλογία με τη μέθοδο `create`, η μέθοδος `update` αντιστοιχεί στην πράξη `UPDATE` της SQL και χρησιμοποιείται για την ενημέρωση των δεδομένων μιας εγγραφής στη βάση. Ο κώδικας της μεθόδου απεικονίζεται στο απόσπασμα 3.13.

Όπως και στην περίπτωση της μεθόδου `create`, η μέθοδος `update` απαιτεί την ύπαρξη ενός αντικειμένου για την εκτέλεσή της. Θα μπορούσαμε να πούμε ότι είναι το ίδιο το αντικείμενο που ενημερώνει την εγγραφή που του αντιστοιχεί στη βάση δεδομένων. Το ίδιο ακριβώς ισχύει και για τη μέθοδο `delete` που θα δούμε στη συνέχεια.

Η μέθοδος delete

Η μέθοδος `delete`, όπως δηλώνει το όνομά της, αντιστοιχεί στην πράξη της SQL `DELETE` και χρησιμοποιείται για την αφαίρεση μιας εγγραφής από τη βάση δεδομένων. Ο κώδικας της μεθόδου απεικονίζεται στο απόσπασμα 3.14.

Όπως διαπιστώνουμε από τον παραπάνω κώδικα, η μέθοδος `delete` απαιτεί την ύπαρξη ενός αντικειμένου για να εκτελεστεί. Μάλιστα είναι το ίδιο το αντικείμενο που επιδρά στον εαυτό του και διαγράφει την αντίστοιχη εγγραφή από τη βάση δεδομένων. Το γεγονός αυτό είναι ενδεικτικό της χρήσης της αντικειμενοστρέφειας στο σχεδιασμό της εφαρμογής, τα πλεονεκτήματα της οποίας θα εξετάσουμε στη συνέχεια.

3.3.6 Πλεονεκτήματα της χρήσης κλάσεων

Η ανάπτυξη της εφαρμογής με τη χρήση αντικειμένων τα οποία διαθέτουν γνωρίσματα και μεθόδους προσφέρει μια σειρά από πλεονεκτήματα, σε σχέση με τη χρήση προστακτικού προγραμματισμού, τα οποία συνοψίζονται στις ακόλουθες παρατηρήσεις:

```
public function update() {
  global $database;

  $sql  = "UPDATE matches SET ";
  $sql .= "match_day_id = '".
    $database->escape_value($this->match_day_id)."', ";
  $sql .= "home_team_id = '".
    $database->escape_value($this->home_team_id)."', ";
  $sql .= "away_team_id = '".
    $database->escape_value($this->away_team_id)."', ";
  $sql .= "result_id = '".
    $database->escape_value($this->result_id)."', ";
  $sql .= "home_goals = '".
    $database->escape_value($this->home_goals)."', ";
  $sql .= "away_goals = '".
    $database->escape_value($this->away_goals)."', ";
  $sql .= "home_penalties = '".
    $database->escape_value($this->home_penalties)."', ";
  $sql .= "away_penalties = '".
    $database->escape_value($this->away_penalties)."' ";
  $sql .= "WHERE id=". $database->escape_value($this->id);
  $database->query($sql);
  return ($database->affected_rows() == 1) ? true : false;
}
```

Κώδικας 3.13: Η μέθοδος update

```
public function delete() {
  global $database;
  $sql = "DELETE FROM matches";
  $sql .= " WHERE id=". $database->escape_value($this->id);
  $sql .= " LIMIT 1";
  $database->query($sql);
  return ($database->affected_rows() == 1) ? true : false;
}
```

Κώδικας 3.14: Η μέθοδος delete

- Έχουμε τη δυνατότητα να αναπτύξουμε τις λειτουργίες της εφαρμογής σε τμήματα κώδικα τα οποία αλληλεπιδρούν μεν, είναι ανεξάρτητα δε μεταξύ τους και μας επιτρέπουν να εξετάζουμε ένα τμήμα τη φορά.

- Η αντικειμενοστρέφεια μας επιτρέπει σε μεγάλο βαθμό να μην επαναλαμβάνουμε τις ίδιες γραμμές κώδικα για να εκτελέσουμε κάποια λειτουργία, αποτελεί δηλαδή ένα τρόπο "αποκέντρωσης" του κώδικα, ο οποίος μας εξασφαλίζει ότι αν μια λειτουργία δουλεύει σωστά σε ένα σημείο της εφαρμογής, τότε θα δουλεύει σε σωστά σε όλα.

- Ο κώδικας για μια μεσαία ή μεγάλη εφαρμογή συχνά μπορεί να επεκταθεί σε χιλιάδες γραμμές εντολών, καθιστώντας δύσκολο τον εντοπισμό σφαλμάτων και τη διόρθωσή τους. Η χρήση κλάσεων βοηθά σημαντικά στην αποσφαλμάτωση (debugging) του κώδικα καθώς το λάθος μπορεί να εντοπιστεί σε επίπεδο κλάσης ή ακόμη και μεθόδου.

- Η αντικειμενοστρέφεια βοηθά σημαντικά στη λογική διάταξη και οργάνωση του κώδικα. Είναι, για παράδειγμα, πολύ πιο λογικό και πολύ πιο κοντά στο μοντέλο του φυσικού κόσμου να αναζητήσουμε το όνομα μιας ομάδας αναφερόμενοι στο αντίστοιχο γνώρισμα ενός αντικειμένου της κλάσης ως εξής: `$team->name`, αντί να ανατρέξουμε σε ένα πίνακα για να βρούμε την τιμή που περιέχεται σε μια του θέση ως εξής: `$row["name"]`. Η λογική οργάνωση του κώδικα μέσω κλάσεων και αντικειμένων, γίνεται ακόμη πιο σημαντική για την περίπτωση των μεθόδων, καθώς, για παράδειγμα, μπορούμε να δημιουργήσουμε μια μέθοδο στην κλάση `Match` που αναλύσαμε προηγουμένως, η οποία να επιστρέφει το αποτέλεσμα ενός αγώνα έτοιμο προς εκτύπωση σε μορφή string χαρακτήρων. Η μέθοδος αυτή απεικονίζεται στον κώδικα 3.15. Έτσι, οποτεδήποτε χρειαζόμαστε να εμφανίσουμε το αποτέλεσμα ενός αγώνα, δεν έχουμε παρά να καλέσουμε τη μέθοδο αυτή μέσω του αντικειμένου που αντιστοιχεί στον αγώνα αυτό μέσω της εντολής `get_result()`.

- Γενικότερα, η οργάνωση της εφαρμογής με βάση τις κλάσεις και τα αντικείμενα μας βοηθά σημαντικά στην εκτέλεση λειτουργιών, αλλά επίσης προσφέρει στον προγραμματιστή μεγάλη ευελιξία ως προς την τροποποίηση ή την προσθήκη νέων λειτουργιών και δυνατοτήτων στην εφαρμογή, όπως είδαμε προηγουμένως με την προσθήκη στην κλάση `Match` της μεθόδου `$match->get_result()`. Το στοιχεία αυτό μας εξασφαλίζει ότι στο μέλλον ο κώδικας της εφαρμογής θα έχει αξία, καθώς θα αρκούν λίγες αλλαγές για την αναβάθμιση της εφαρμογής.

3.3.7 Λογική οργάνωση του κώδικα

Η κλάση `Match` που αναλύσαμε ήδη, αποτελεί μια μόνο από τις κλάσεις που αντιστοιχούν σε αντικείμενα ή οντότητες του μοντέλου της βάσης δεδομένων.

```
public function get_result() {
  if ($this->result_id == 1) {
    return "Pending";
  } else {
    return $this->home_goals." - ".$this->away_goals;
  }
}
```

Κώδικας 3.15: Η μέθοδος get_result()

Κατ' αντιστοιχία, δημιουργούμε κλάσεις για όλους τους υπόλοιπους πίνακες με την ίδια δομή που αναφέραμε προηγουμένως. Κάθε μια από τις κλάσεις αυτές θα διαθέτει σαν γνωρίσματα (attributes) μεταβλητές που αναλογούν στα πεδία του αντίστοιχου πίνακα, και μεθόδους (methods) που αντιστοιχούν στις πράξεις `SELECT`, `INSERT`, `UPDATE` και `DELETE` της γλώσσας SQL. Επιπλέον, κάθε μια από τις κλάσεις αυτές θα διαθέτει επιπλέον μεθόδους, οι οποίες θα είναι εξειδικευμένες για τις λειτουργίες της κλάσης αυτής. Ο πλήρης κώδικας των κλάσεων που δημιουργήθηκαν για την εφαρμογή παρατίθεται στο παράρτημα Β′. Στην ενότητα αυτή θα εξετάσουμε ορισμένες από τις μεθόδους αυτές κα τη λειτουργία τους.

Η πρώτη μέθοδος που θα εξετάσουμε ανήκει στην κλάση `Group` και ονομάζεται `update_ranks()`. Η μέθοδος χρησιμοποιείται για να ενημερώσει την κατάταξη των ομάδων που ανήκουν στον όμιλο αυτό με βάση τους βαθμούς που έχουν συγκεντρώσει. Είναι μια μέθοδος η οποία απαιτεί την ύπαρξη ενός αντικειμένου της κλάσης `Group` για να εκτελεστεί, κάτι το οποίο συνάδει με τη λογική, καθώς είναι σαν να ζητάμε από ένα συγκεκριμένο όμιλο να ενημερώσει την κατάταξη των ομάδων που μετέχουν σε αυτόν. Όμως το αντικείμενο `$group` δεν διαθέτει κάποιο γνώρισμα που αντιστοιχεί στην έννοια της κατάταξης. Αντίθετα, τα αντικείμενα της κλάσης `Team` διαθέτουν ένα γνώρισμα με το όνομα `group_rank`, το οποίο αναλογεί στο αντίστοιχο πεδίο του πίνακα *teams* στη βάση δεδομένων και χρησιμοποιείται για το σκοπό αυτό. Επομένως οι δύο κλάσεις πρέπει να συνεργαστούν για να επιτελέσουν τη λειτουργία αυτή. Ο κώδικας της μεθόδου απεικονίζεται στο απόσπασμα 3.16 και αποτελείται από τα εξής βήματα:

1. Η μέθοδος ζητά από την κλάση `Team` να εκτελέσει ένα ερώτημα στη βάση, το οποίο ανακτά τις ομάδες που ανήκουν στον όμιλο αυτό και είναι ταξινομημένες με βάση τους βαθμούς που έχουν συγκεντρώσει στον όμιλο (πεδίο *group_points*).

2. Η μέθοδος `find_by_sql` της κλάσης `Team` εκτελείται και επιστρέφει μια συλλογή από αντικείμενα της κλάσης αυτής.

3. Η τιμή του πεδίου *group_rank* ενημερώνεται με βάση τη σειρά με την οποία ανακτήθηκαν τα αντικείμενα από τη βάση μέσω του ερωτήματος που υποβλήθηκε.

```
public function update_ranks() {
  $sql  = "SELECT * FROM teams ";
  $sql .= "WHERE group_id = ". $this->id ." ";
  $sql .= "ORDER BY group_points DESC";
  $teams = Team::find_by_sql($sql);

  for ($i=0; $i<4; $i++) {
    $team = $teams[$i];
    $team->group_rank = $i + 1;
    $team->save();
  }
}
```

Κώδικας 3.16: Η μέθοδος update_ranks()

Η διαδικασία αυτή είναι ενδεικτική του τρόπου με τον οποίο μπορούν να συνεργαστούν οι κλάσεις της εφαρμογής για να εκτελέσουν διαφορετικές λειτουργίες. Ένα δεύτερο παράδειγμα συνεργασίας ανάμεσα σε κλάσεις είναι η μέθοδος `update_stats()` της κλάσης `Team`, ένα απόσπασμα της οποίας εμφανίζεται στον κώδικα 3.17. Πρόκειται για μια μέθοδο η οποία χρησιμοποιείται για την ενημέρωση των στατιστικών στοιχείων της ομάδας μετά την εισαγωγή κάποιου αποτελέσματος αγώνα. Η διαδικασία που ακολουθείται συνοψίζεται στα ακόλουθα βήματα:

1. Η μέθοδος αρχικοποιεί τα στατιστικά στοιχεία της ομάδας μηδενίζοντας τις τιμές των αντίστοιχων γνωρισμάτων της.

2. Στη συνέχεια χρησιμοποιείται η εξειδικευμένη μέθοδος `find_by_home_id` της κλάσης `Match`, η οποία επιστρέφει όλους τους αγώνες στους οποίους η συγκεκριμένη ομάδα αγωνίστηκε σαν γηπεδούχος.

3. Για κάθε αγώνα προσδιορίζεται αν πρόκειται για αγώνα της φάσης των ομίλων ή για αγώνα μεταγενέστερης φάσης. Ο έλεγχος γίνεται σε συνεργασία με την κλάση `MatchDay`.

4. Ανάλογα με το αποτέλεσμα του αγώνα και με τη φάση της διοργάνωσης στην οποία αντιστοιχεί ενημερώνονται τα αντίστοιχα γνωρίσματα της ομάδας.

5. Τα βήματα 2, 3 και 4 επαναλαμβάνονται για τους αγώνες στους οποίους η ομάδα ήταν φιλοξενούμενη.

6. Μετά το πέρας των υπολογισμών τα δεδομένα της αντίστοιχης εγγραφής στη βάση δεδομένων ενημερώνονται.

```
public function update_stats() {
  ...
  $matches = Match::find_by_home_team_id($this->id);
  foreach ($matches as $match) {
    $group_stage_match =
        (MatchDay::find_by_id($match->match_day_id)->stage_id ==
        1) ? true : false;

    $this->overall_goals_for += $match->home_goals;
    $this->overall_goals_against += $match->away_goals;
    if ($group_stage_match) {
      $this->group_goals_for += $match->home_goals;
      $this->group_goals_against += $match->away_goals;
    }
    if ($match->result_id == 2) {
      $this->overall_wins++;
      if ($group_stage_match) {
        $this->group_wins++; $this->group_points += 3;
      }
    } elseif ($match->result_id == 3) {
      $this->overall_draws++;
      if ($group_stage_match) {
        $this->group_draws++; $this->group_points += 1;
      }
    } elseif ($match->result_id == 4) {
      $this->overall_losses++;
      if ($group_stage_match) {
        $this->group_losses++;
      }
    }
    ...
  }

  $this->save();
}
```

Κώδικας 3.17: Η μέθοδος update_stats()

Τα δύο αυτά παραδείγματα εξειδικευμένων μεθόδων που εξετάσαμε φανερώνουν τη δύναμη που ποσφέρει η αντικειμενοστρέφεια και η χρήση της στην ανάπτυξη εφαρμογών. Για τις ανάγκες της εφαρμογής έχουμε αναπτύξη μια σειρά από εξειδικευμένες μεθόδους οι οποίες εντάσσονται στις κλάσεις και παρέχουν χρήσιμες λειτουργίες. Ο αναγνώστης μπορεί να ανατρέξει στο παράρτημα Β΄ για τον πλήρη κώδικα των κλάσεων της εφαρμογής.

Έχοντας χτίσει τον σκελετό της εφαρμογής, στη συνέχεια θα περάσουμε στην ανάλυση του σχεδιασμού των σελίδων της εφαρμογής και στην απεικόνιση της πληροφορίας στο χρήστη.

Κεφάλαιο 4

Διαδικασία σχεδίασης

Στην ενότητα αυτή θα ασχοληθούμε με το περιβάλλον αλληλεπίδρασης με το χρήστη της εφαρμογής, τη διάταξη της πληροφορίας στις διάφορες σελίδες του site και τις τεχνικές που χρησιμοποιούνται για τη σχεδίαση μιας σύγχρονης online εφαρμογής γενικότερα.

4.1 Browsers και XHTML

Κάθε online εφαρμογή, σε όποια τεχνολογία κι αν στηρίζεται, προϋποθέτει από το χρήστη να χρησιμοποιήσει έναν browser για να έχει πρόσβαση στην πληροφορία. Οι browsers είναι προγράμματα σχεδιασμένα να τρέχουν σε υπολογιστές, κινητά τηλέφωνα και άλλες συσκευές, και τα οποία είναι σε θέση να αποκωδικοποιήσουν αρχεία γραμμένα σε γλώσσα HTML και να απεικονίσουν την πληροφορία που περιέχεται σε αυτά στο χρήστη.

Η γλώσσα HTML είναι μια περιγραφική γλώσσα (mark-up language), η οποία περιέχει συγκεκριμένες εντολές που ονομάζονται *tags* και καθοδηγούν τον browser ως προς τη σύνταξη της πληροφορίας που περιέχεται στο κείμενο. Παράδειγμα μιας τέτοιας εντολής αποτελεί το απόσπασμα 4.1, το οποίο δηλώνει ότι το κείμενο που περιέχεται ανάμεσα στα tags `<p>` και `</p>` θα πρέπει να απεικονιστεί σαν παράγραφος.

```
<p>This is a new paragraph.</p>
```

Κώδικας 4.1: Παράδειγμα εντολής HTML

Οι εντολές που απευθύνονται στους browser δεν εμφανίζονται στο χρήστη, αλλά επηρεάζουν τον τρόπο με τον οποίο η πληροφορία απεικονίζεται σε αυτόν. Όπως οι μέθοδοι σε μια γλώσσα σαν την PHP δέχονται παραμέτρους που καθορίζουν το αποτέλεσμα εκτέλεσής τους, έτσι και οι εντολές της γλώσσας HTML δέχονται παραμέτρους ή attributes, τα οποία καθορίζουν τη συμπεριφορά ή και την εμφάνιση ενός στοιχείου της σελίδας. Παράδειγμα χρήσης μιας τέτοιας παραμέτρου αποτελεί το απόσπασμα 4.2. Ο browser θα διαβάσει αυτήν την εντολή

```
<p color="red">This is a new paragraph.</p>
```

Κώδικας 4.2: Παράδειγμα εντολής HTML με attribute

και θα μορφοποιήσει κατάλληλα το περιεχόμενο σύμφωνα με τις όποιες παραμετρους. Σε αυτήν την περίπτωση το κείμενο της παραγράφου θα έχει κόκκινο χρώμα.

Υπάρχει μια πληθώρα εντολών που απευθύνονται στον browser και ορίζουν τον τρόπο με τον οποίο θα απεικονιστεί η πληροφορία. Η διεξοδική παρουσίαση των εντολών αυτών ξεφεύγει από τα πλαίσια της παρούσας διπλωματικής εργασίας[1]. Όμως πριν προχωρήσουμε θα ήταν χρήσιμο να αναφέρουμε ορισμένα από τα πιο διαδεδομένα tags και attributes, στα οποίες θα αναφερθούμε στη συνέχεια:

`<p>` Ορίζει μια νέα παράγραφο

`<h1>, ..., <h6>` Ορίζει διάφορα επίπεδα επικεφαλίδων, με το νούμερο 1 να αντιστοιχεί στην πιο σημαντική και το νούμερο 6 στη λιγότερο σημαντική επικεφαλίδα.

`<div>` Ορίζει μια λογική ενότητα της πληροφορίας που εμφανίζεται στη σελίδα.

`<form>` Ορίζει μια φόρμα, η οποία αποτελείται από πεδία διαφόρων ειδών και συμπληρώνεται από το χρήστη για την παροχή δεδομένων στην εφαρμογή.

`<table>` Ορίζει ένα πίνακα από στοιχεία κατανεμημένα σε γραμμές και στήλες.

`id` Αποτελεί ένα attribute, το οποίο μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις περισσότερες εντολές HTML για να ταυτοποιήσει ένα στοιχείο με μοναδικό τρόπο.

`class` Αποτελεί ένα attribute, το οποίο μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις περισσότερες εντολές HTML για να δηλώσει ότι ένα στοιχείο της σελίδας ανήκει σε ένα συγκεκριμένο σύνολο στοιχείων.

Οι εντολές HTML, όπως αυτές που αναφέραμε παραπάνω, περιγράφουν τη δομή της πληροφορίας που εμφανίζεται σε μια σελίδα. Όμως παράλληλα καθορίζουν, έστω και σε μικρό βαθμό, τον τρόπο με τον οποίο η πληροφορία αυτή θα εμφανιστεί στην οθόνη του χρήστη. Για παράδειγμα, μια επικεφαλίδα τύπου `<h1>` συνήθως απεικονίζεται με γράμματα πολύ μεγαλύτερα από μια επικεφαλίδα τύπου `<h3>`. Αντίστοιχα, ένας σύνδεσμος προς μια άλλη σελίδα συνήθως εμφανίζεται με υπογραμμισμένο κείμενο χρώματος μπλε. Το μέσο που αποφασίζει για τον τρόπο με τον οποίο θα απεικονίζονται τα διάφορα στοιχεία μιας σελίδας είναι ο browser.

[1] Ο αναγνώστης μπορεί να ανατρέξει στην αντίστοιχη ενότητα της βιβλιογραφίας για περισσότερες πληροφορίες.

```
<HTML>
  <head>
    <title>This is the bad page</title>
  </head>
  <BODY>
    <h1>Some title here
    <p color=red>This is some paragraph text with <b>bold and
        <i>italic</b></i> text.
  </body>
</HTML>
```

Κώδικας 4.3: Παράδειγμα σελίδας HTML με εσφαλμένη σύνταξη

Οι προγραμματιστές και σχεδιαστές των σελίδων, από τις πρώτες μέρες του internet, θέλησαν να μπορούν να καθορίζουν τον τρόπο με τον οποίο θα απεικονίζεται η πληροφορία στην οθόνη του χρήστη. Για το σκοπό αυτό, άρχισαν να χρησιμοποιούν διάφορες τεχνικές εκμεταλλευόμενοι τους μηχανισμούς που προσέφερε η HTML μέσω attributes όπως `color` και `background`. Επιπλέον, για τη διάταξη της πληροφορίας στο χώρο η πιο διαδεδομένη τεχνική ήταν η χρήση ενός μεγάλου πίνακα με τα διάφορα τμήματα της σελίδας να μοιράζονται ανάμεσα στα κελιά του πίνακα αυτού. Οι τεχνικές αυτές δημιούργησαν περισσότερα προβλήματα, παρά έδωσαν λύσεις στο ζήτημα της αισθητικής παρέμβασης στον τρόπο εμφάνισης του περιεχομένου των σελίδων. Στην προβληματική αυτή κατάσταση συνέβαλαν και οι διάφοροι browsers, καθώς και οι εταιρίες που τους υποστήριζαν, καθώς στον πόλεμο για μεγαλύτερο μερίδιο αγοράς, κάθε browser εισήγαγε δικές του πρόσθετες λειτουργίες και δυνατότητες τις οποίες μια online εφαρμογή καλούνταν να υλοποιήσει. Τη λύση στην κατάσταση αυτή έδωσαν τα web standards στα οποία θα αναφερθούμε στη συνέχεια.

4.2 Τα web standards και η σημασία τους

Ο όμιλος W3C (World Wide Web Consortium) δημιουργήθηκε με σκοπό να συντάξει τα πρότυπα που θα καθορίζουν μια σειρά από τεχνολογίες που σχετίζονται με το internet. Μια από τις βασικότερες δραστηριότητες του W3C ήταν να επανακαθορίσει τη γλώσσα HTML με ένα τρόπο καθορισμένο και εννιαίο, ο οποίος σήμερα υιοθετείται από όλους τους σύγχρονους browsers.

4.2.1 Η περιγραφική γλώσσα XHTML

Η γλώσσα HTML ήταν εξ αρχής μια γλώσσα ορισμένη με χαλαρό τρόπο και επέτρεπε στο σχεδιαστή της σελίδας να προσεγγίζει τη διαδικασία σχεδίασης με τρόπο μη συστηματικό χωρίς αυτό να έχει επίπτωση στην εμφάνιση της πληροφορίας στη σελίδα. Παράδειγμα των εντολών μιας σελίδας απεικονίζεται στον κώδικα 4.3.

Στο απόσπασμα αυτό κώδικα υπάρχει μια σειρά από συντακτικά λάθη, τα οποία όμως διορθώνονται από τον browser χωρίς να δημιουργείται πρόβλημα στην παρουσίαση της σελίδας. Τα λάθη αυτά συνοψίζονται στις ακόλουθες κατηγορίες:

1. Ορισμένες από τις εντολές έχουν tag έναρξης, αλλά δεν έχουν tag τερματισμού.
2. Η τιμή του attribute color της παραγράφου δεν βρίσκεται εντός εισαγωγικών.
3. Οι εντολές `<b>` και `<i>` είναι ενσωματωμένες η μια μέσα στην άλλη (nested).
4. Ορισμένες από τις εντολές είναι γραμμένες με κεφαλαία, ενώ άλλες είναι γραμμένες με πεζά.

Με βάση τη γλώσσα HTML, το W3C όρισε μια νέα έκδοση η οποία ονομάστηκε XHTML (Extensible Hypertext Mark-up Language). Η γλώσσα αυτή είναι πλήρως συμβατή με την προκάτοχό της και υποστηρίζει την πλειονότητα των εντολών της. Η βασικότερη όμως διαφορά που εισήγαγε ήταν ένας πιο αυστηρός ορισμός της γλώσσας ο οποίος δεν επέτρεπε τέτοιου είδους ασυνέπειες στον κώδικα. Πιο συγκεκριμένα, για να είναι ένα αρχείο σύμφωνο με τις προδιαγραφές της γλώσσας XHTML θα πρέπει να πληροί τις ακόλουθες βασικές προϋποθέσεις:

1. Όλες οι εντολές θα πρέπει να έχουν tag έναρξης και τερματισμού.
2. Δύο εντολές δεν πρέπει να ενσωματώνουν η μια την άλλη (neste tags).
3. Οι τιμές των ορισμάτων θα πρέπει να περιέχονται σε διπλά εισαγωγικά (double quotes).
4. Όλες οι εντολές θα πρέπει να είναι γραμμένες με πεζά γράμματα.

Με βάση τις προδιαγραφές αυτές, στη συνέχεια θα εξετάσουμε τη δομή ενός εγγράφου XHTML.

4.2.2 Δομή ενός εγγράφου XHTML

Η δομή ενός σωστά ορισμένου XHTML εγγράφου εμφανίζεται στον κώδικα 4.4. Η πρώτη εντολή του αρχείου ορίζει τον τύπο της πληροφορίας που περιέχεται στο έγγραφο. Με τον όρο `DOCTYPE` και τις παραμέτρους που ακολουθούν, ο σχεδιαστής της σελίδας ενημερώνει τον browser ότι το έγγραφο έχει οριστεί με βάση τις προδιαγραφές που ορίζει το πρότυπο XHTML, όπως αυτό καθορίζεται από το W3C. Έτσι, ο browser γνωρίζει ότι πρόκειται για ένα σωστά γραμμένο έγγραφο και το μορφοποιεί σύμφωνα με το πρότυπο που δηλώθηκε.

Οι σύγχρονοι browsers μπορούν, στην πλειονότητά τους, να υποστηρίξουν τα πρότυπα που ορίζονται από το W3C. Το γεγονός αυτό μας εξασφαλίζει ότι

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
    lang="en">
<head>
  <meta http-equiv="Content-type" content="text/html;
      charset=utf-8" />
  <title>This is a proper page</title>
</head>
<body>
  <h1>A heading</h1>
  <p>Page content goes here...</p>
</body>
</html>
```

Κώδικας 4.4: Παράδειγμα σωστά ορισμένης σελίδας XHTML

αν ορίσουμε τις σελίδες της εφαρμογής μας σύμφωνα με το πρότυπο του W3C, η πληροφορία που περιέχεται σε αυτές θα εμφανίζεται με τον ίδιο τρόπο στο χρήστη ανεξαρτήτως του browser χρησιμοποιεί.

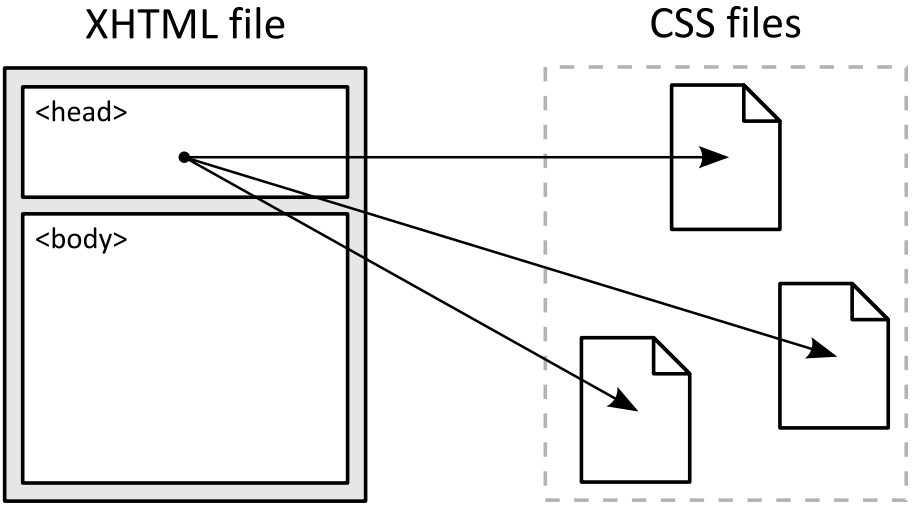
Επιπλέον, η τήρηση των προδιαγραφών της γλώσσας XHTML διασφαλίζει τη βιωσιμότητα της εφαρμογής μας στο μέλλον, καθώς θα παραμείνει συμβατή με τους μελλοντικούς browsers όσο το πρότυπό στο οποίο βασίζεται υποστηρίζεται από το W3C.

4.3 Cascading Stylesheets

Μια ακόμη τεχνολογία που προωθείται από τον όμιλο W3C είναι η χρήση των CSS (Cascading Stylesheets) για τον καθορισμό της εμφάνισης της σελίδας. Όπως αναφέραμε προηγουμένως, η γλώσσα HTML εξ αρχής προσέφερε κάποιους υποτυπώδεις μηχανισμούς παρέμβασης στον τρόπο με τον οποίο ο browser θα απεικονίσει μια σελίδα στην οθόνη του χρήστη.

Οι μηχανισμοί όμως αυτοί επέβαλαν ήταν απλές προσθήκες στη γλώσσα και δεν αποτελούσαν μια ολοκληρωμένη τεχνολογία με ουσιαστικές δυνατότητες καθορισμού της αισθητικής μιας σελίδας. Για το σκοπό αυτό δημιουργήθηκαν τα CSS τα οποία έχουν τη δικιά τους περιγραφική γλώσσα και προσφέρουν σημαντικά πλεονεκτήματα στο σχεδιαστή της σελίδας, επιτρέποντάς του να ορίσει με μεγάλη ακρίβεια το αισθητικό περιεχόμενο της σελίδας. Η λογική χρήσης ενός αρχείου CSS απεικονίζεται στο σχήμα 4.1.

Ουσιαστικά πρόκειται για ξεχωριστά αρχεία στα οποία ένα έγγραφο αναφέρεται μέσω μιας εντολής που περιλαμβάνεται στην επικεφαλίδα (head) του εγγράφου. Ένα αρχείο CSS αποτελείται από μια σειρά κανόνων, κάθε ένας από τους οποίους καθορίζει την εμφάνιση ενός στοιχείου της σελίδας. Κάθε κανόνας μπορεί να περιλαμβάνει μια σειρά από παραμέτρους οι οποίες εκφράζονται με τη μορφή ζευγών κλειδιού-τιμής, όπως απεικονίζεται στον κώδικα 4.5.



Σχήμα 4.1: Λογική χρήσης των αρχείων CSS

```
body {
  font-family: Calibri, Helvetica, Arial, sans-serif;
  font-size: 13px;
  line-height: 20px;
  color: #2e3436;
  background: #d3d7cf url(../images/bg_body.png) top left
      repeat-x;
}
```

Κώδικας 4.5: Παράδειγμα κανόνα CSS

Στο συγκεκριμένο παράδειγμα, ο κανόνας αφορά στην εμφάνιση του στοιχείου που καθορίζεται από το tag `<body>` και καθορίζει ότι η γραμματοσειρά που χρησιμοποιείται από τη σελίδα είναι μια από τις Calibri, Helvetica ή Arial, ανάλογα με τη διαθεσιμότητά τους στο μέσο που χρησιμοποιείται για πρόσβαση στη σελίδα. Επιπλέον ορίζεται το μέγεθος και το διάστιχο της γραμματοσειράς, καθώς επίσης το χρώμα του κειμένου και του φόντου της σελίδας. Υπάρχει μια πληθώρα κανόνων και γνωρισμάτων (attributes) που μπορούν να χρησιμοποιηθούν σε ένα αρχείο CSS, η διεξοδική παρουσίαση των οποίων ξεφεύγει από τα όρια της παρούσας διπλωματικής εργασίας. Αξίζει όμως να σταθούμε στα πλεονεκτήματα που αποκομίζονται από τη χρήση των CSS για τον καθορισμό της εμφάνισης της εφαρμογής, στα οποία θα αναφερθούμε στη συνέχεια.

4.3.1 Οφέλη από την χρήση των CSS

Η τεχνολογία των CSS προσφέρει μια σειρά από δυνατότητες και πλεονεκτήματα, που καθιστούν τη χρήση τους απαραίτητη στο σχεδιασμό μιας σύγχρονης online εφαρμογής. Στη συνέχεια θα αναλύσουμε τα κυριότερα από αυτά:

Η ιεραρχία των κανόνων

Τα διάφορα στοιχεία (elements) ενός XHTML εγγράφου αποτελούν μια ιεραρχία από tags. Έτσι, όπως είδαμε και στο παράδειγμα του αποσπάσματος 4.4, το στοιχείο που ορίζεται με τα tags `<p>` και `</p>` περιέχεται στο στοιχείο `<body>`. Καθώς ένα έγκυρο XHTML έγγραφο ακολουθεί αυστηρούς κανόνες σύνταξης, ανάμεσα στα στοιχεία του εγγράφου δημιουργείται μια ιεραρχία επιπέδων.

Η λέξη *cascading* στον όρο CSS δηλώνει ότι οι εντολές που ορίζονται για ένα στοιχείο μιας σελίδας, θα ισχύουν και για όλα τα στοιχεία που εμφανίζονται ως απόγονοι ή παιδιά του στοιχείου αυτού στο δέντρο της ιεραρχίας του εγγράφου. Έτσι, αν το χρώμα του κειμένου για το στοιχείο `<body>` οριστεί να είναι κόκκινο, τότε και το χρώμα του κειμένου της παραγράφου `<p>` θα είναι επίσης κόκκινο, εκτός και αν υπάρχει κάποιος κανόνας που να αναιρεί τις τιμές που κληρονομούνται.

Ο μηχανισμός αυτός προσφέρει στο σχεδιαστή τη δυνατότητα να έχει τον πλήρη έλεγχο της εμφάνισης των στοιχείων των σελίδων της εφαρμογής, καθώς μπορεί να χρησιμοποιήσει τόσο γενικούς κανόνες οι οποίοι κληρονομούνται από πολλά επίπεδα της ιεραρχίας, όσο και ειδικούς κανόνες για να εστιάσει σε ορισμένα μόνο από αυτά.

Η αρχή της μη επανάληψης

Η αρχή ανάπτυξης DRY βρίσκει άμεση εφαρμογή στη χρήση των CSS. Στην πλειοψηφία τους, οι online εφαρμογές αποτελούνται από διαφορετικές ενότητες και σελίδες, κάθε μια από τις οποίες επιτελεί και διαφορετικές λειτουργίες. Συνήθως όμως, οι σελίδες αυτές μοιράζονται, σε γενικές γραμμές, την ίδια εμφάνιση,

καθώς με τον τρόπο αυτό δίνεται στο χρήστη η αίσθηση πως πρόκειται για μια εννιαία εφαρμογή.

Με τις παλιές τεχνικές, ο σχεδιαστής της εφαρμογής θα έπρεπε να επαναλάβει τις περισσότερες από τις εντολές καθορισμού της εμφάνισης της πληροφορίας σε κάθε μια από τις σελίδες που απαρτίζουν την εφαρμογή. Με τα αρχεία CSS όμως, το μόνο που χρειάζεται είναι να δημιουργήσουμε ένα και μόνο αρχείο που να περιέχει όλους τους κανόνες που είναι απαραίτητοι για την εφαρμογή μας και να αναφερθούμε στο αρχείο αυτό από τις σελίδες. Επιπλέον, η μέθοδος αυτή διευκολύνει σημαντικά στις όποιες αλλαγές ή διορθώσεις επιθυμούμε να πραγματοποιήσουμε, καθώς αρκεί να τροποποιήσουμε μόνο ένα αρχείο και οι αλλαγές αυτές θα επηρεάσουν όλες τις σελίδες που αναφέρονται στο αρχείο αυτό.

Διαχωρισμός εμφάνισης-περιεχομένου

Η χρήση των CSS μας επιτρέπει να εστιάσουμε στις δύο όψεις μιας σελίδας της εφαρμογής μας: το έγγραφο XHTML καθορίζει με τρόπο σαφή και συντακτικά ορθό την πληροφορία μιας σελίδας. Για παράδειγμα, ένας πίνακας χρησιμοποιείται για να απεικονίσει δεδομένα που αντιστοιχούν λογικά σε μια πινακοειδή μορφή, και όχι για να ορίσει ότι η σελίδα θα αποτελείται από δύο στήλες. Ο καθορισμός της μορφής της πληροφορίας και της αισθητικής αναπαράστασής της αναλαμβάνεται από τα CSS. Χρησιμοποιώντας σωστά τα στοιχεία και τις εντολές της γλώσσας XHTML εξασφαλίζουμε ότι η πληροφορία και τα δεδομένα μας θα έχουν πάντοτε τη σωστή δομή με οποιοδήποτε μέσο και αν χρησιμοποιείται για πρόσβαση σε αυτά. Για παράδειγμα, άτομα με δυσκολίες στην όραση, θα είναι σε θέση να επισκεφθούν και να χρησιμοποιήσουν την εφαρμογή με τον ίδιο τρόπο με τους υπόλοιπους χρήστες, χωρίς να επιβαρύνονται από πλεονάζουσες εντολές στις σελίδες που σκοπό δεν σχετίζονται με την πληροφορία αυτή καθ'εαυτή, παρά με την εμφάνισή της.

Διαφορετικά μέσα πρόσβασης

Η εντολή που χρησιμοποιείται για να συνδέσει ένα έγγραφο XHTML με ένα αρχείο CSS εντάσσεται στο τμήμα `<head>` του εγγράφου και έχει τη μορφή που απεικονίζεται στον κώδικα 4.6. Όπως διαπιστώνουμε από το απόσπασμα αυτό, ένα από τα γνωρίσματα της εντολής είναι ο όρος *media*, η τιμή του οποίου στη συγκεκριμένη περίπτωση έχει τεθεί ίση με `screen`. Ο όρος αυτός δηλώνει, ότι οι κανόνες που περιέχονται στο αρχείο CSS στο οποίο αναφέρεται η εντολή, απευθύνονται σε μέσα όπως οι browsers ενός προσωπικού υπολογιστή.

Με τον τρόπο αυτό μπορούμε να ορίσουμε διαφορετικά αρχεία, τα οποία να καθορίζουν διαφορετικούς κανόνες ανάλογα με το μέσο με το οποίο πραγματοποιείται η πρόσβαση στη σελίδα. Οι πιο διαδεδομένες εναλλακτικές περιπτώσεις είναι οι `handheld`, για την περίπτωση που χρησιμοποιείται ο browser ενός κινητού τηλεφώνου και `print`, για την περίπτωση που ο χρήστης επιθυμεί να τυπώσει μια σελίδα. Περισσότερα για το χαρακτηριστικό αυτό των CSS θα αναλύσουμε

```
<link href="/path/to/file/main.css"
      rel="stylesheet"
      type="text/css"
      media="screen" />
```

Κώδικας 4.6: Εντολή αναφοράς σε αρχείο CSS

σε επόμενη ενότητα.

4.4 Σύγχρονες τάσεις στη σχεδίαση ιστοσελίδων

Μετά τη σύντομη αυτή αναφορά στην τεχνολογία CSS και στα πλεονεκτήματά της, και πριν περάσουμε στην ανάλυση της εμφάνισης της εφαρμογής, αξίζει να σταθούμε σε ορισμένες παρατηρήσεις σχετικά με το σχεδιασμό σύγχρονων ιστοσελίδων και web εφαρμογών. Σε αντίθεση με τα προγράμματα που είναι εγκατεστημένα σε ένα προσωπικό υπολογιστή, μια online εφαρμογή δεν χρησιμοποιεί ένα προκαθορισμένο από κάποιο λειτουργικό σύστημα περιβάλλον διαπροσωπίας με το χρήστη (user interface). Αντίθετα, ο σχεδιαστής της εφαρμογής οφείλει να σχεδιάσει το περιβάλλον αυτό έτσι, ώστε να είναι λειτουργικό και εύκολα κατανοητό από το χρήστη.

Κατά την υλοποίηση του user interface, μια σειρά από παραμέτρους πρέπει να ληφθούν υπ' όψην. Κατ' αρχήν η εφαρμογή απευθύνεται κυρίως σε χρήστες οι οποίοι θα χρησιμοποιούν τον browser ενός προσωπικού υπολογιστή για έχουν πρόσβαση σε αυτήν. Οι χρήστες μπορούν να διαθέτουν διαφορετικά λειτουργικά συστήματα, να χρησιμοποιούν διαφορετικούς browsers σε διαφορετικά μεγέθη και αναλύσεις απεικόνισης. Έτσι, η διάταξη της πληροφορίας θα πρέπει να είναι τέτοια που να εξυπηρετεί όσο το δυνατόν περισσότερες περιπτώσεις χρηστών.

Οι διαστάσεις της σελίδας

Η τήρηση των web standards κατά τη σχεδίαση των σελίδων της εφαρμογής μας εξασφαλίζει ότι η εμφάνισή τους θα είναι ίδια ανεξάρτητα από το λειτουργικό σύστημα και τον browser του χρήστη, καθώς όλοι οι σύγχρονοι browsers ακολουθούν τα σύγχρονα πρότυπα και απεικονίζουν την πληροφορία με τον ίδιο τρόπο. Ως προς το μέγεθος όμως και την ανάλυση της οθόνης του χρήστη καλούμαστε να πάρουμε μια απόφαση. Οι χρήστες είναι εξοικειωμένοι με την κατακόρυφη κύλιση (scrolling) της σελίδας, όμως βρίσκουν εξαιρετικά άβολη την οριζόντια κύλιση του παραθύρου του browser. Για το λόγο αυτό, το πλάτος της σελίδας θα πρέπει να είναι τέτοιο ώστε να αποτρέπεται η ανάγκη να οριζόντια επέκταση του παραθύρου. Μια από τις πιο διαδεδομένες διαστάσεις στο σύγχρονο σχεδιασμό σελίδων είναι αυτή των *960px*. Η διάσταση αυτή έχει το μεγάλο προσόν ότι διαιρείται με ένα πλήθος αριθμών ακριβώς (2, 3, 4, 6, 8, κλπ) και επομένως μας διευκολύνει να χωρίσουμε οπτικά τη διάταξή μας σε στήλες ή ενότητες.

```
h1 {
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size: 24px;
  font-weight: normal;
}
p{
  font-family: Calibri, Helvetica, Arial, sans-serif;
  font-size: 11px;
  line-height: 18px;
  color: #202020;
}
```

Κώδικας 4.7: Καθορισμός γραμματοσειράς με CSS

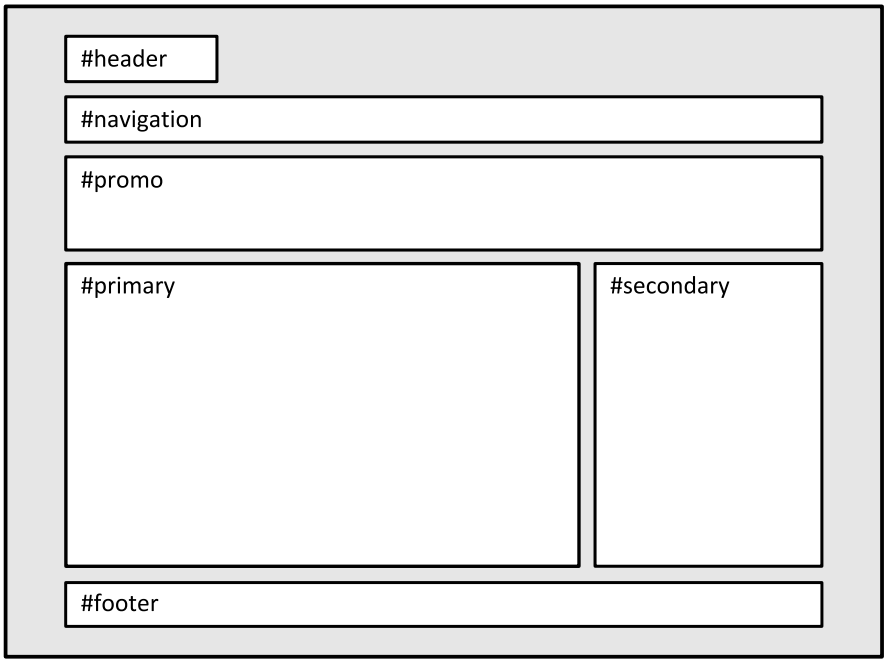
Επιπλέον, μας εξασφαλίζει ότι οι χρήστες, στη συντριπτική τους πλειοψηφία δεν θα χρειαστεί να μετακινήσουν οριζόντια το παράθυρο του browser, καθώς στατιστικά στοιχεία δείχνουν ότι αναλύσεις οθόνης με πλάτος μικρότερο από 1024px ανήκουν πια στο παρελθόν.

Η τυπογραφία και η σημασία της

Ένας σημαντικός όγκος πληροφορίας στο internet αντιστοιχεί σε απλό κείμενο. Επομένως είναι σημαντικό η παρουσίαση του κειμένου να είναι ευανάγνωστη και ξεκούραστη για το χρήστη. Οι γραμματοσειρές που μπορούν να χρησιμοποιηθούν από μια εφαρμογή για την απεικόνιση του κειμένου της, αναγκαστικά θα πρέπει να είναι διαθέσιμες στον υπολογιστή του χρήστη. Όμως, όπως ήδη αναφέραμε, οι χρήστες μπορούν να χρησιμοποιούν διαφορετικά λειτουργικά συστήματα και επομένως να διαθέτουν διαφορετικές εγκατεστημένες γραμματοσειρές. Για να αντιμετωπιστεί αυτό το πρόβλημα, τα CSS προσφέρουν ένα μηχανισμό που επιτρέπει στο σχεδιαστή της σελίδας να καθορίσει μια σειρά από γραμματοσειρές, όπως φαίνεται στον κώδικα 4.7.

Ο browser διαβάζει τους αντίστοιχους κανόνες και επιχειρεί να βρει μια από τις γραμματοσειρές στο λειτουργικό σύστημα με σειρά προτεραιότητας τη σειρά με την οποία αυτές αναφέρονται στον κανόνα. Αν καμια από τις γραμματοσειρές δεν είναι διαθέσιμη, τότε οι επιλογές `serif` και `sans-serif` επιλέγονται και οι γραμματοσειρές που χρησιμοποιούνται είναι εκείνες που καθορίζονται από τον ίδιο τον browser σε συνεργασία με το λειτουργικό σύστημα.

Μέσω των κανόνων CSS, ο σχεδιαστής έχει τη δυνατότητα να καθορίσει μια σειρά από παραμέτρους που σχετίζονται με την εμφάνιση του κειμένου, όπως το μέγεθός του, το διάστιχο, το χρώμα και το στυλ των γραμμάτων. Για περισσότερες πληροφορίες σχετικά με τις παραμέτρους αυτές, ο αναγνώστης μπορεί να ανατρέξει στην αντίστοιχη ενότητα της βιβλιογραφίας.



Σχήμα 4.2: Γενική διάταξη των σελίδων της εφαρμογής

Χρώμα και γραφικά

Το χρώμα και τα γραφικά μπορούν να αναδείξουν ή να υποβαθμίσουν την πληροφορία που περιέχεται σε μια σελίδα. Επιπλέον, θα πρέπει να διασφαλίσουμε ότι η σελίδα θα είναι προσβάσιμη από όλους του χρήστες, ακόμη και από εκείνους με προβλήματα όρασης. Για παράδειγμα, ιδιαίτερη προσοχή απαιτείται ώστε οι αποχρώσεις που χρησιμοποιούμε να μην προκαλούν προβλήματα στους ανθρώπους με αχρωματοψία ή άλλες παρόμοιες παθήσεις και να παρέχουν αρκετή αντίθεση ώστε η δομή της σελίδας να είναι πλήρως λειτουργική.

4.5 Ο σκελετός των σελίδων της εφαρμογής

Με βάση τα όσα αναφέραμε στην παρούσα ενότητα, θα προχωρήσουμε στη συνέχεια στη σχεδίαση των σελίδων της εφαρμογής. Η γενική διάταξη στης πληροφορίας σε μια σελίδα απεικονίζεται στο σχήμα 4.2.

Όπως φαίνεται στη γενική αυτή διάταξη, η σελίδα αποτελείται από ενότητες, κάθε μια από τις οποίες αντιστοιχεί σε διαφορετικά τμήματα της πληροφορίας:

header Περιέχει το λογότυπο της εφαρμογής, καθώς επίσης κάποιους γενικούς συνδέσμους για διαδικασίες όπως η είσοδος (login) ή η εγγραφή (register) στην εφαρμογή.

navigation Αποτελεί το βασικό μέσο πλοήγησης στις διάφορες ενότητες της εφαρμογής και απαρτίζεται από μια σειρά από συνδέσμους με ενδεικτικά ονόματα. Η χρήση του οριζόντιου "μενού" πλοήγησης αποτελεί μια εξαιρετικά διαδεδομένη τεχνική και επομένως είναι οικεία στους χρήστες.

promo Περιέχει διακοσμητικά στοιχεία, όπως ένα εικονίδιο και σύντομο εισαγωγικό κείμενο. Επίσης, μπορεί να χρησιμοποιηθεί για σημαντικές ανακοινώσεις, όπως για παράδειγμα για να ενημερώσει τους επισκέπτες του site ότι η συμμετοχή στο παιχνίδι είναι δυνατή για περιορισμένο χρονικό διάστημα.

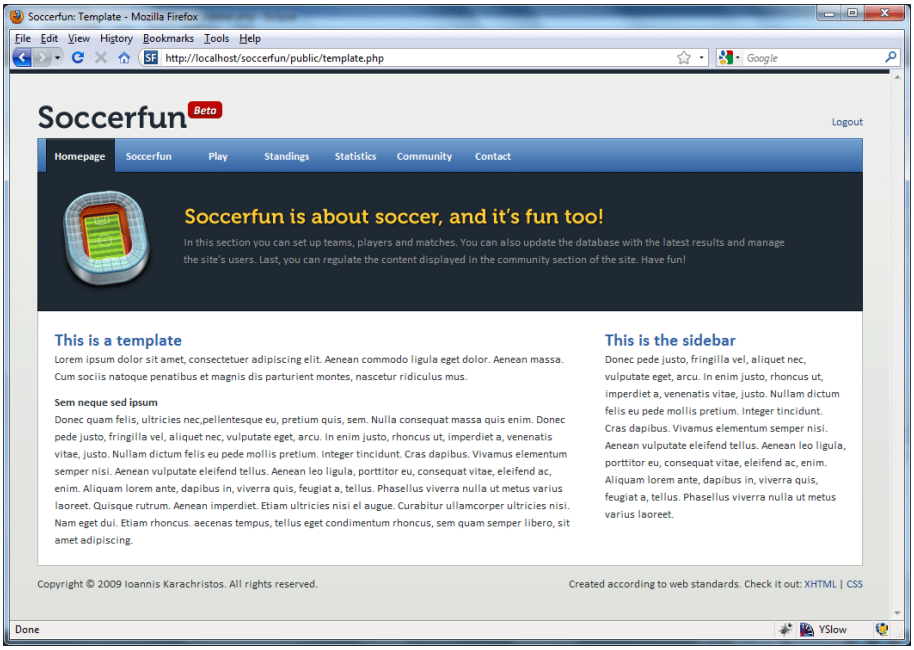
primary Αποτελεί το βασικό περιεχόμενο κάθε σελίδας και περιέχει φόρμες συμμετοχής στο παιχνίδι, πίνακες αποτελεσμάτων και κατάταξης των χρηστών κλπ.

secondary Περιέχει το δευτερεύον περιεχόμενο κάθε σελίδας, όπως για παράδειγμα οδηγίες συμπλήρωσης μιας φόρμας ή μια σύνοψη της συμμετοχής του τρέχοντος χρήστη στο παιχνίδι.

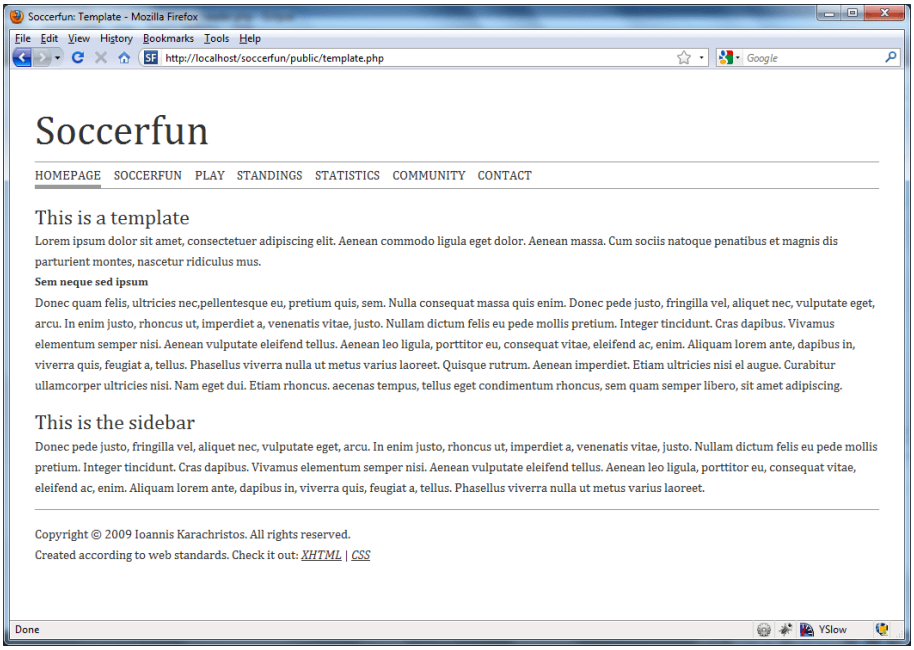
footer Περιέχει τη δήλωση δικαιωμάτων της σελίδας, καθώς επίσης συνδέσμους που προωθούν τη χρήση και εφαρμογή των web standards.

Η πραγματική μορφή της σελίδας, όπως αυτή εμφανίζεται στην οθόνη του χρήστη απεικονίζεται στο σχήμα 4.3. Στο σχήμα 4.4 απεικονίζεται η ίδια σελίδα αν ο χρήστης επιθυμεί να την τυπώσει σε κάποιον εκτυπωτή. Παρατηρούμε ότι για την περίπτωση αυτή, το χρώμα και τα γραφικά αφαιρούνται από την εμφάνιση της πληροφορίας για λόγους οικονομίας και η έμφαση δίνεται στο περιεχόμενο αυτό καθ' εαυτό. Τέλος, στο σχήμα 4.5 απεικονίζεται η ίδια σελίδα αν ο χρήστης πλοηγηθεί σε αυτήν μέσω του browser ενός κινητού τηλεφώνου. Σε αυτήν την περίπτωση, λόγω του περιορισμένου μεγέθους της οθόνης αλλά και των περιορισμένων δυνατοτήτων του μέσου, όπως για παράδειγμα η απουσία υποστήριξης επιπλέον γραμματοσειρών, το design της σελίδας τροποποιείται σημαντικά και περιοριζόμαστε μόνο σε μικρές αλλά ουσιαστικές παρεμβάσεις.

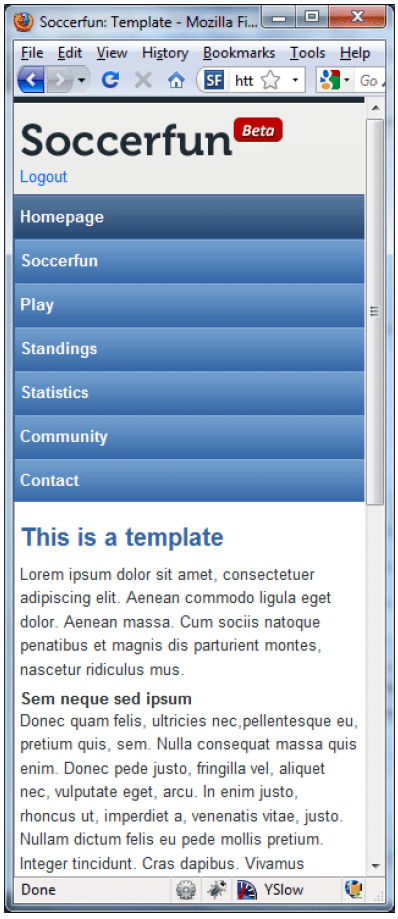
Οι τρεις αυτές εκδόσεις της σελίδας έγιναν δυνατές μέσω της υλοποίησης τριών διαφορετικών αρχείων CSS και της αναφοράς σε αυτά από τη σελίδα. Ο κώδικας των αρχείων αυτών παρατίθεται στο παράρτημα Δ′. Στην επόμενη ενότητα θα αναλύσουμε τις διάφορες ενότητες της εφαρμογής πιο διεξοδικά και θα εξετάσουμε διαφορετικά σενάρια χρήσης.



Σχήμα 4.3: Εμφάνιση της σελίδας σύμφωνα με το main.css



Σχήμα 4.4: Εμφάνιση της σελίδας σύμφωνα με το print.css



Σχήμα 4.5: Εμφάνιση της σελίδας σύμφωνα με το mobile.css

Κεφάλαιο 5

Σύνθεση της εφαρμογής

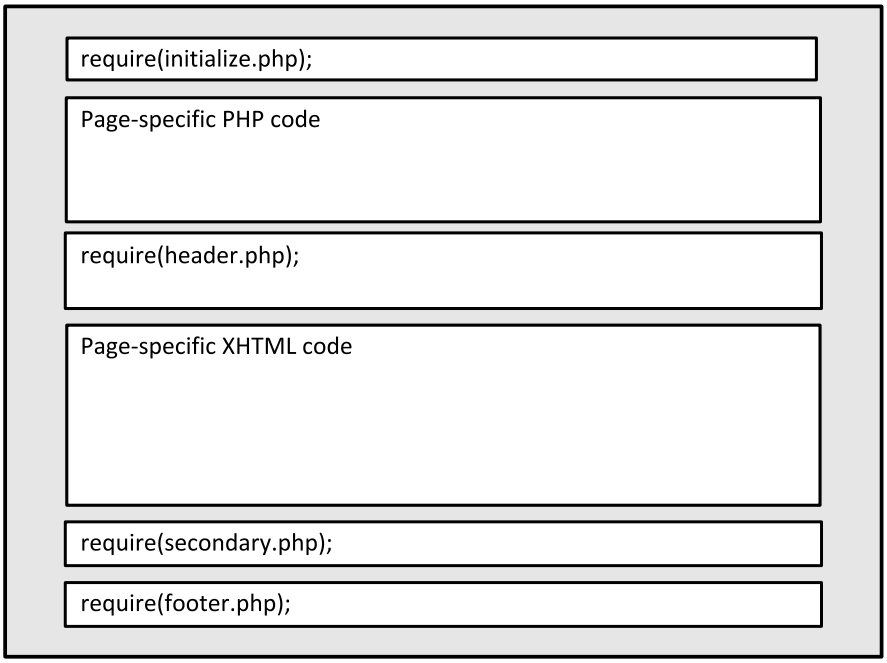
Στην ενότητα αυτή θα αναλύσουμε τις διάφορες ενότητες της εφαρμογής, τόσο από την πλευρά του διαχειριστή, όσο και από την πλευρά του απλού χρήστη που συμμετέχει στο παιχνίδι. Πριν προχωρήσουμε όμως στην ανάλυση των λειτουργιών της σελίδας, θα παρουσιάσουμε την εσωτερική δομή μιας τυπικής σελίδας της εφαρμογής.

5.1 Δομή των σελίδων της εφαρμογής

Οι περισσότερες σελίδες της εφαρμογής συνδυάζουν μια σειρά από επί μέρους στοιχεία. Αφ' ενός, κάθε σελίδα απεικονίζει κάποιας μορφής πληροφορία στην οθόνη του χρήστη. Για το σκοπό αυτό χρησιμοποιεί κώδικα XHTML και αναφέρεται σε αρχεία CSS για τον καθορισμό της μορφής και της διάταξης της πληροφορίας αυτής, όπως είδαμε στο προηγούμενο κεφάλαιο. Αφ' ετέρου, κάθε σελίδα περιέχει καθαρό κώδικα PHP, ο οποίος καθορίζει τις λειτουργίες που επιτελούνται από τη σελίδα αυτή. Τα κοινά αυτά στοιχεία ανάμεσα σε όλες τις σελίδες που απαρτίζουν την εφαρμογή, συνεπάγονται μια επανάληψη πολλών γραμμών κώδικα, η οποία μπορεί να αντιμετωπισθεί με την αφαίρεση (abstraction) των τμημάτων της σελίδας και τη δημιουργία μιας εννιαίας δομή που θα χρησιμοποιείται από ολόκληρη την εφαρμογή. Η δομή αυτή που δημιουργήσαμε απεικονίζεται στο σχήμα 5.1 και αποτελείται από τα εξής τμήματα:

1. Η πρώτη εντολή περιλαμβάνει τον κώδικα του αρχείου *initialize.php*, ο οποίος παρατίθεται στο παράρτημα Β′. Το αρχείο αυτό με τη σειρά του περιλαμβάνει τον κώδικα όλων των κλάσεων της εφαρμογής, παρέχοντας έτσι σε όλες τις σελίδες τη λειτουργικότητα των κλάσεων αυτών.

2. Το δεύτερο τμήμα της δομής περιέχει τον κώδικα PHP, ο οποίος είναι γραμμένος για τη συγκεκριμένη σελίδα. Έτσι, στο τμήμα αυτό μπορούν να επιτελούνται εργασίες όπως η ανάκτηση δεδομένων από τη βάση δεδομένων, ο χειρισμός της υποβολής μιας φόρμας της σελίδας, ό έλεγχος των στοιχείων μιας φόρμας για σφάλματα κλπ.



Σχήμα 5.1: Δομή μιας τυπικής σελίδας της εφαρμογής

3. Το τρίτο τμήμα της δομής περιέχει τις εντολές που ορίζουν την έναρξη του κώδικα XHTML και αναφέρεται στα εξωτερικά αρχεία, όπως τα αρχεία CSS που χρησιμοποιούνται από τη σελίδα αυτή. Επιπλέον, περιέχει τα τμήματα *header*, *navigation*, και *promo*, όπως αυτά παρουσιάστηκαν στο προηγούμενο κεφάλαιο.

4. Το τέταρτο τμήμα της δομής περιέχει το ουσιαστικό περιεχόμενο της παρούσας σελίδας και αντιστοιχεί στο κομμάτι *primary*. Στο σημείο αυτό μπορούν να απεικονίζονται οι φόρμες επικοινωνίας, πίνακες κατάταξη και αποτελεσμάτων, κλπ.

5. Το πέμπτο τμήμα της δομής αντιστοιχεί στο δευτερεύον περιεχόμενο της σελίδας και η χρήση του είναι προαιρετική.

6. Το έκτο και τελευταίο τμήμα της σελίδας, αντιστοιχεί στην ενότητα *footer* και επιπλέον περιέχει κώδικα PHP, ο οποίος είναι κοινός σε όλες τις σελίδες, όπως για παράδειγμα ο τερματισμός της σύνδεσης με τη βάση δεδομένων.

Η ανάπτυξη μιας εννιαίας δομής συμβάλει σημαντικά στη μείωση του απαιτούμενου κώδικα για τη δημιουργία μιας σελίδας και μας εξασφαλίζει ότι αν ένα μοιραζόμενο στοιχείο της δομή λειτουργεί σε μια σελίδα σωστά, τότε θα λειτουρ-

γεί σε όλες. Με βάση αυτό το σκεπτικό, στη συνέχεια θα ασχοληθούμε με τις επί μέρους ενότητες της εφαρμογής.

5.2 Περιβάλλον του διαχειριστή

Ανατρέχοντας στις λειτουργικές απαιτήσεις του διαχειριστή της εφαρμογής, διαπιστώνουμε ότι πρέπει να του παρέχεται η δυνατότητα αρχικά να διαμορφώνει το παιχνίδι, στη συνέχεια να ενημερώνει τη βάση δεδομένων με τα αποτελέσματα των αγώνων της διοργάνωσης και τέλος να διαχειρίζεται τους χρήστες που μετέχουν στην εφαρμογή. Όλες οι παραπάνω εργασίες θα πρέπει να επιτελούνται με τρόπο απλό, γρήγορο και ασφαλή, ώστε να εξασφαλίζεται η ακεραιότητα των δεδομένων της βάσης.

5.2.1 Αρχική διαμόρφωση του παιχνιδιού

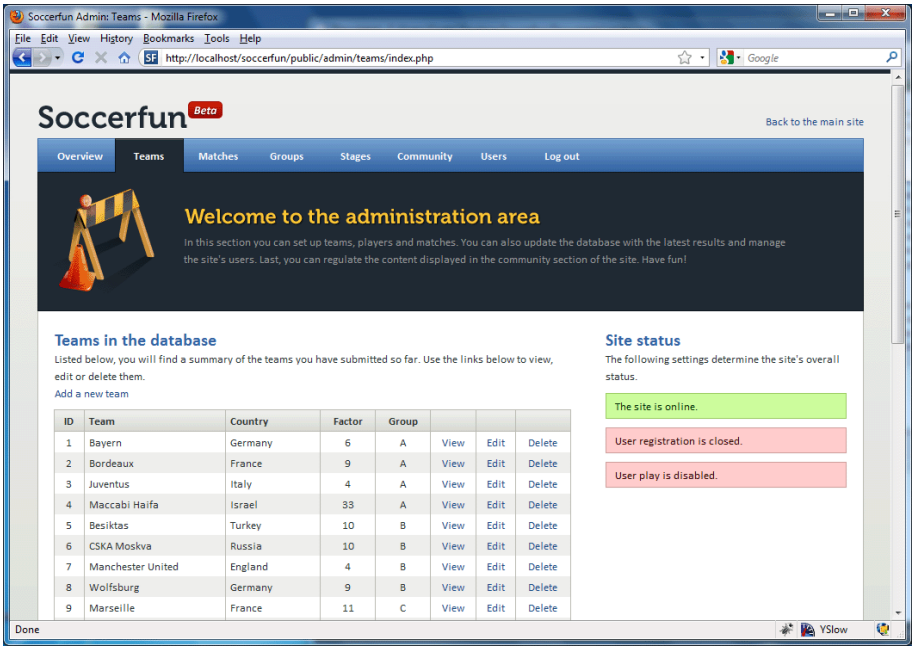
Η αρχική διαμόρφωση του παιχνιδιού συνίσταται στην εισαγωγή των ομάδων που συμμετέχουν στη διοργάνωση και των παικτών που ανήκουν στις ομάδες αυτές, στην κατάταξη των ομάδων αυτών σε ομίλους και στον ορισμό των αγώνων στη φάση των ομίλων. Για το σκοπό αυτό διαμορφώνουμε τις ενότητες **Teams** και **Matches** οι οποίες παρέχουν στο διαχειριστή τα μέσα για την εισαγωγή των αρχικών στοιχείων του παιχνιδιού.

Η ενότητα Teams

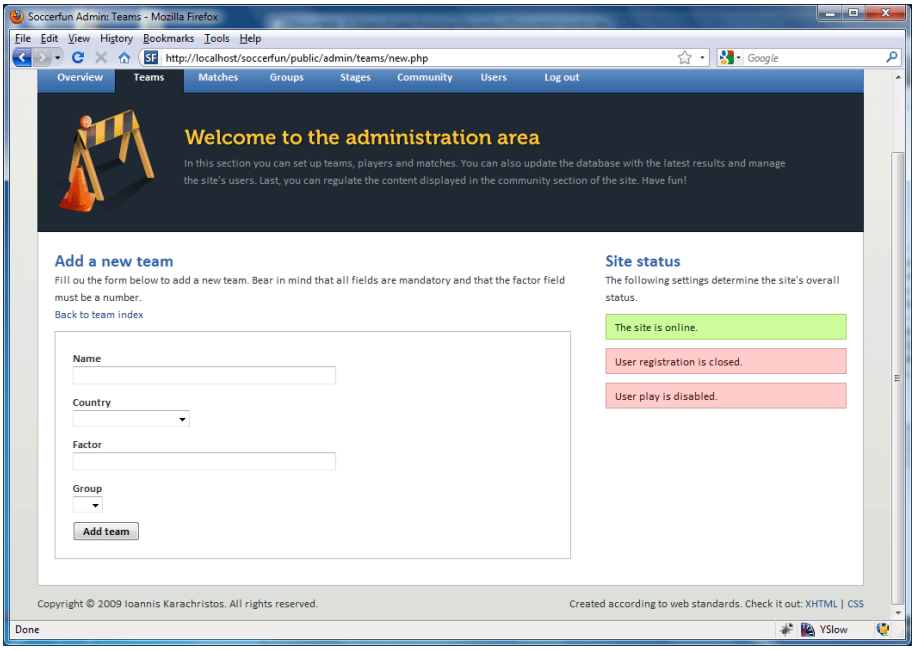
Μια τυπική εικόνα αυτής της ενότητα απεικονίζεται στο σχήμα 5.2. Στο μενού πλοήγησης η "καρτέλα" Teams είναι μορφοποιημένη με διαφορετικό τρόπο ώστε να υπογραμμίζεται ότι η παρούσα σελίδα αντιστοιχεί στη συγκεκριμένη ενότητα. Επιπλέον στο τμήμα του δευτερεύοντος περιεχομένου (secondary) εμφανίζονται τρεις βασικές ρυθμίσεις οι οποίες είναι ενδεικτικές της κατάστασης της εφαρμογής. Η πρώτη αφορά στο αν η εφαρμογή είναι διαθέσιμη (online) ή όχι για τους επισκέπτες του site. Η δεύτερη και η τρίτη ρύθμιση αφορούν στο αν οι χρήστες έχουν τη δυνατότητα να εγγραφούν στην εφαρμογή και να συμμετάσχουν στο παιχνίδι αντιστοίχως. Περισσότερα για τις ρυθμίσεις αυτές θα δούμε σε επόμενη ενότητα.

Η κεντρική σελίδα της ενότητας Teams απεικονίζει τις ομάδες οι οποίες έχουν ήδη εισαχθεί στη βάση δεδομένων. Επιπλέον υπάρχει ένας σύνδεσμος ο οποίος επιτρέπει στο διαχειριστή να εισάγει μια νέα ομάδα στο παιχνίδι. Τα στοιχεία που απαιτούνται για τη δημιουργία της νέας εγγραφής είναι το όνομα της ομάδας, η χώρα προέλευσης, ο συντελεστής απόδοσης και ο όμιλος στον οποίο θα ενταχθεί η ομάδα αυτή. Τα στοιχεία αυτά υποβάλονται με τη συμπλήρωση των αντίστοιχων πεδίων μιας φόρμας, όπως απεικονίζεται στην εικόνα του σχήματος 5.3.

Η σελίδα εισαγωγής μιας νέας ομάδας είναι υπεύθυνη για τον έλεγχο των στοιχείων που εισάγονται. Για τη διευκόλυνση του διαχειριστή η φόρμα έχει



Σχήμα 5.2: Οψη της ενότητας Teams



Σχήμα 5.3: Σελίδα εισαγωγής μιας νέας ομάδας

εμπλουτιστεί, όπου αυτό είναι δυνατόν, με στοιχεία που περιορίζουν τα δεδομένα που ζητούνται σε ορισμένες επιλογές. Έτσι, το πεδίο της χώρας προέλευσης έχει αντικατασταθεί από ένα μενού επιλογής με όλες τις υπάρχουσες χώρες. Αντίστοιχα, η επιλογή του ομίλου στον οποίο ανήκει η ομάδα, επίσης γίνεται μέσα από ένα μενού επιλογής. Χρησιμοποιώντας τέτοιου είδους μεθόδους εισόδου δεδομένων, όχι μόνο διευκολύνουμε το χρήστη της σελίδας, αλλά επιπλέον εξασφαλίζουμε ότι τα δεδομένα αυτά θα έχουν την κατάλληλη μορφή.

Επιπλέον, η σελίδα ελέγχει τις τιμές που ο διαχειριστής υποβάλει για ενδεχόμενα λάθη. Κατ' αρχήν επιβεβαιώνει ότι όλα τα απαραίτητα πεδία έχουν συμπληρωθεί κατάλληλα. Στη συνέχεια, πριν πραγματοποιήσει μια νέα εγγραφή στον αντίστοιχο πίνακα της βάσης δεδομένων, ελέγχει αν υπάρχει ήδη κάποια άλλη ομάδα με το ίδιο όνομα. Τέλος, ενημερώνει το χρήστη για το αποτέλεσμα της υποβολής των στοιχείων, με ένα εμφανές μήνυμα, το οποίο τον ενημερώνει για την επιτυχία της καταχώρησης ή για τυχόν σφάλματα που βρέθηκαν και απέτρεψαν την εισαγωγή της νέας ομάδας.

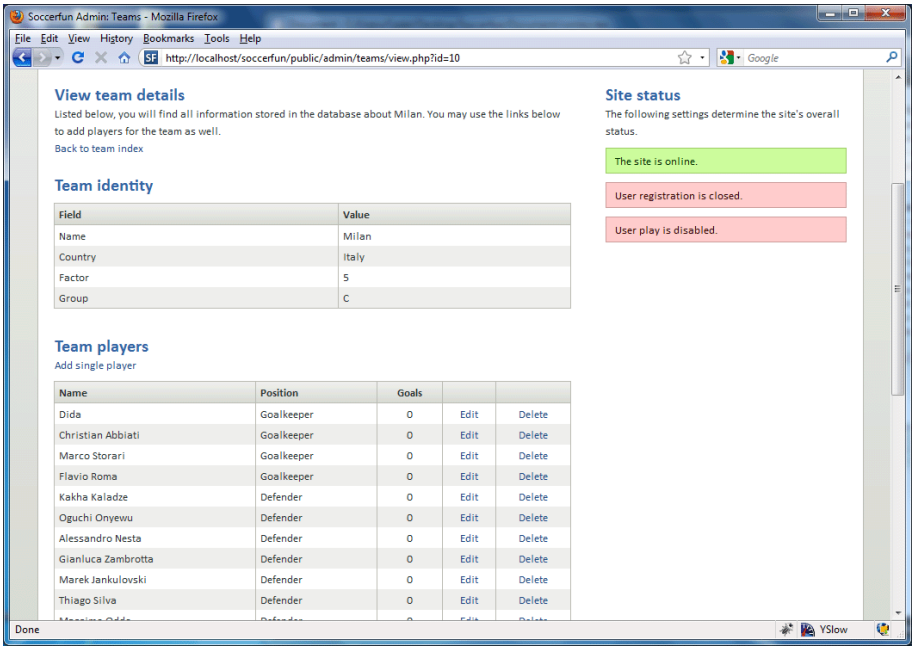
Επιστρέφοντας στην αρχική σελίδα της ενότητας Teams, διαπιστώνουμε ότι για κάθε ομάδα διατίθενται τρεις σύνδεσμοι με τα ονόματα *view*, *edit* και *delete*. Ο σύνδεσμος *edit* οδηγεί σε μια νέα σελίδα, στην οποία ο διαχειριστής έχει τη δυνατότητα να τροποποιήσει τα στοιχεία μιας ομάδας. Η σελίδα είναι παρόμοια με τη σελίδα δημιουργίας ομάδας, με τη διαφορά ότι τα πεδία της φόρμας είναι αρχικοποιημένα στις τιμές που ήδη έχουν τεθεί για την εν λόγω ομάδα. Ο σύνδεσμος *delete* διαγράφει την ομάδα από το παιχνίδι, ενώ ο σύνδεσμος *view* οδηγεί σε μια σελίδα στην οποία ο διαχειριστής μπορεί να έχει μια γενικότερη άποψη για την ομάδα.

Η σελίδα *view* απεικονίζεται στην εικόνα 5.4 και περιλαμβάνει τα βασικά στοιχεία της ομάδας, τους παίκτες που αγωνίζονται στην ομάδα αυτή και μια σύνοψη των αγώνων στους οποίους έχει συμμετάσχει η ομάδα. Επιπλέον, δίνεται η δυνατότητα στο διαχειριστή να εισάγει νέους παίκτες για την ομάδα, να τροποποιήσει τα στοιχεία που έχουν ήδη καταχωρηθεί ή να διαγράψει κάποιον παίκτη. Όλες οι λειτουργίες αυτές, επιτελούνται σε αντίστοιχες ξεχωριστές σελίδες, κάθε μια από τις οποίες είναι υπεύθυνη για τον έλεγχο των δεδομένων που υποβάλονται κάθε φορά.

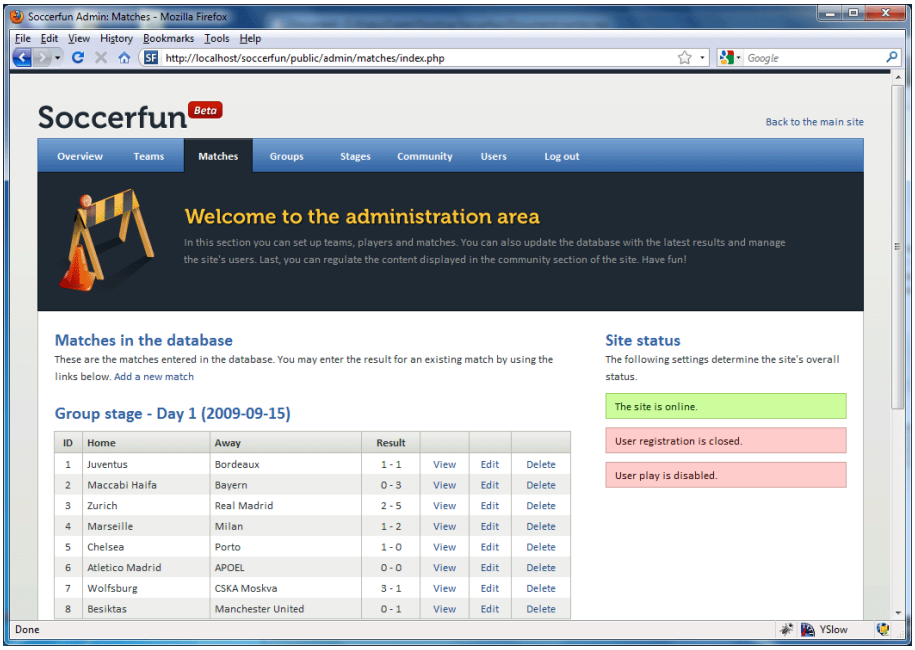
Η ενότητα Matches

Έχοντας ολοκληρώσει την εισαγωγή των ομάδων και των παικτών που αγωνίζονται σε αυτές, καθώς επίσης των καθορισμό των ομίλων στους οποίους αυτές ανήκουν, στην ενότητα αυτή καθορίζουμε τους αγώνες της διοργάνωσης. Η κεντρική σελίδα της ενότητας απεικονίζεται στο σχήμα 5.5.

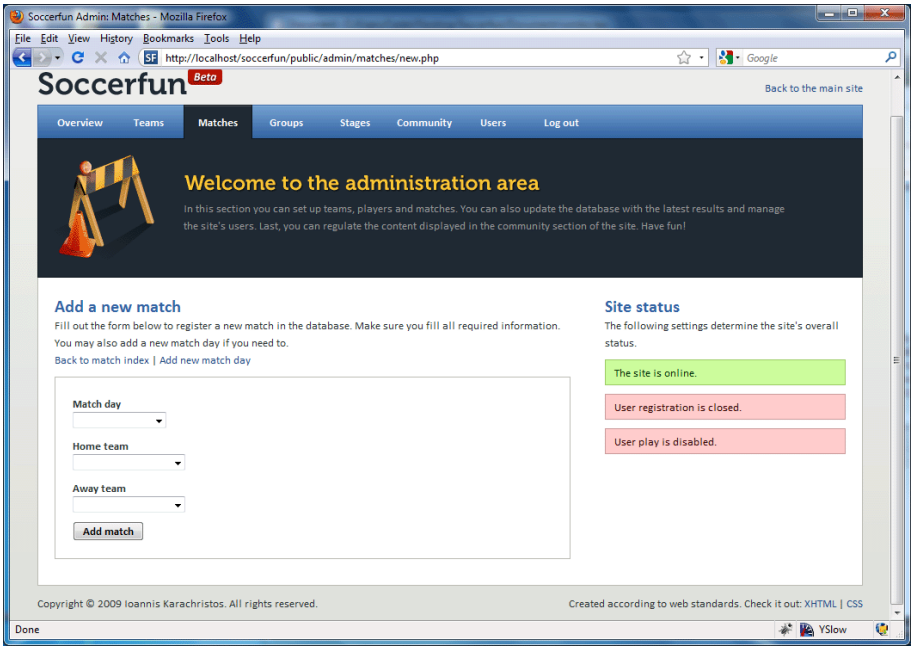
Στη σελίδα απεικονίζονται οι αγώνες οι οποίοι ήδη έχουν εισαχθεί στη βάση δεδομένων. Για κάθε αγώνα εμφανίζεται η γηπεδούχος και η φιλοξενούμενη ομάδα, καθώς επίσης το αποτέλεσμα του αγώνα. Επιπλέον, προσφέρεται η δυνατότητα προσθήκης ενός νέου αγώνα μέσω ενός συνδέσμου ο οποίος οδηγεί σε μια νέα σελίδα η οποία απεικονίζεται στο σχήμα 5.6. Όπως και στην περίπτωση εισαγω-



Σχήμα 5.4: Σελίδα όψης των στοιχείων μιας ομάδας



Σχήμα 5.5: Κεντρική σελίδα της ενότητας Matches

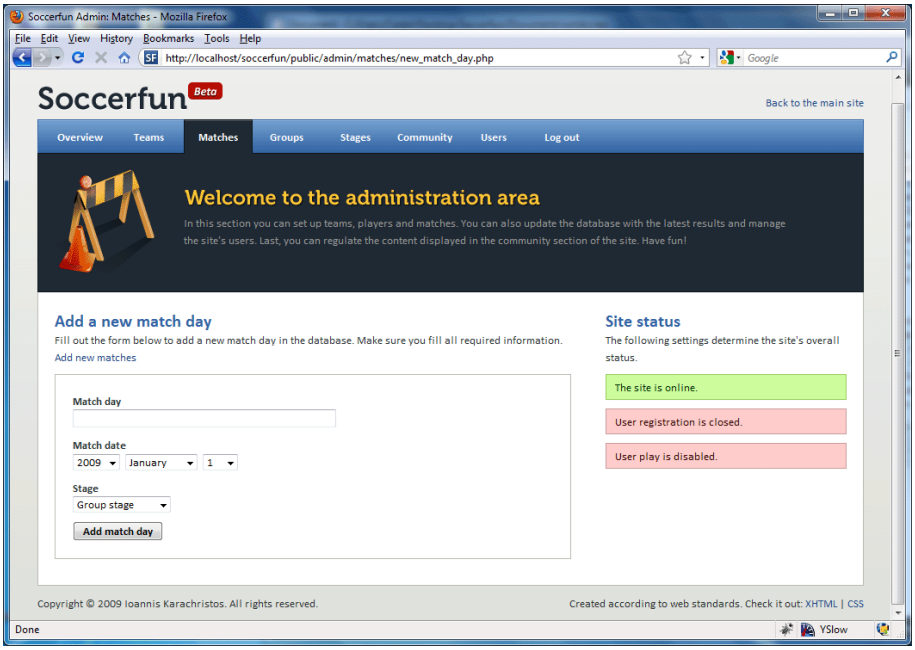


Σχήμα 5.6: Σελίδα εισαγωγής ενός νέου αγώνα

γής νέας ομάδας, και σε αυτήν την περίπτωση χρησιμοποιούμε το μηχανισμό των μενού επιλογής για να διευκολύνουμε την επιλογή των ομάδων. Έτσι, ο διαχειριστής επιλέγει την αγωνιστική ημέρα, τη φιλοξενούμενη και τη γηπεδούχο ομάδα και υποβάλει τη φόρμα για να δημιουργήσει ένα νέο αγώνα στη βάση δεδομένων.

Η σελίδα είναι υπεύθυνη για τον έλεγχο της ορθότητας των δεδομένων που υποβάλονται. Έτσι, ελέγχει ότι οι δύο ομάδες ανήκουν στον ίδιο όμιλο, αν πρόκειται για αγώνα της φάσης των ομίλων και ότι δεν υπάρχει άλλος αγώνας της ίδιας ομάδας για την ίδια αγωνιστική. Αν οι διαθέσιμες επιλογές ως προς την αγωνιστική δεν είναι επαρκείς, ο διαχειριστής μπορεί να δημιουργήσει μια νέα αγωνιστική ημέρα μέσω ενός συνδέσμου που οδηγεί σε μια σελίδα εισαγωγής αγωνιστικής ημέρας. Η σελίδα αυτή εμφανίζεται στο σχήμα 5.7.

Επιστρέφοντας στην κεντρική σελίδα της ενότητας των αγώνων, για κάθε ματς προσφέρονται τρεις σύνδεσμοι με ονόματα *view*, *edit* και *delete*. Ο πρώτος σύνδεσμος οδηγεί σε μια σελίδα στην οποία μπορούμε να δούμε τα στοιχεία που έχουν καταχωρηθεί για τον παιχνίδι αυτό, ενώ ο δεύτερος μας επιτρέπει να τροποποιήσουμε τα στοιχεία αυτά. Τέλος, ο τρίτος σύνδεσμος μας επιτρέπει να διαγράψουμε έναν αγώνα από τη βάση. Περισσότερα για τις λειτουργίες *view* και *edit* θα δούμε στην επόμενη ενότητα, στην οποία αναλύουμε την ενημέρωση των αποτελεσμάτων.



Σχήμα 5.7: Σελίδα εισαγωγής νέας αγωνιστικής ημέρας

5.2.2 Ενημέρωση αποτελεσμάτων

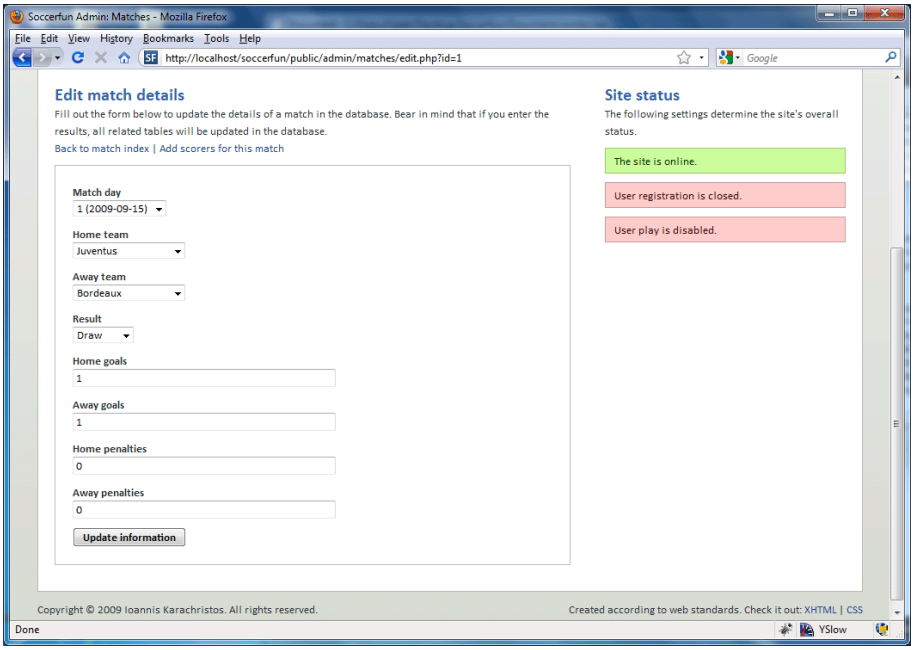
Η ενημέρωση των αποτελεσμάτων συνίσταται στην εισαγωγή των σκορ των αγώνων, μόλις αυτοί ολοκληρωθούν, τον καθορισμό των σκορερ του κάθε γκολ ενός αγώνα και την ενημέρωση των ομάδων που προκρίνονται σε επόμενη φάση της διοργάνωσης.

Για την ενημέρωση των αποτελεσμάτων των αγώνων, χρησιμοποιούμε και πάλι την ενότητα Matches και συγκεκριμένα τη λειτουργία *edit* που προσφέρεται για κάθε αγώνα. Ο σύνδεσμος αυτός οδηγεί το χρήστη σε μια σελίδα η μορφή της οποίας απεικονίζεται στο σχήμα 5.8. Στη σελίδα αυτή ο διαχειριστής έχει τη δυνατότητα να συμπληρώσει την εικονιζόμενη φόρμα, καθορίζοντας έτσι το αποτέλεσμα του αγώνα (πεδίο *result*), καθώς επίσης τα γκολ της κάθε ομάδας.

Στη συνέχεια, έχουμε τη δυνατότητα να προσθέσουμε τους σκόρερς για τον αγώνα, τα στοιχεία του οποίου συμπληρώνουμε. Έτσι, ο σύνδεσμος *view* του κάθε αγώνα στην κεντρική σελίδα της ενότητας Matches, οδηγεί με τη σειρά της σε μια σελίδα που απεικονίζει τα στοιχεία που έχουν καταχωρηθεί για τον αγώνα αυτό, συμπεριλαμβανομένων των σκόρερ.

Η ενότητα Groups

Καθώς η διοργάνωση εξελίσσεται, οι ομάδες συγκεντρώνουν βαθμούς από τις νίκες και τις ισοπαλίες και έτσι σε κάθε όμιλο δημιουργείται μια κατάταξη



Σχήμα 5.8: Σελίδα ενημέρωσης αποτελέσματος ενός αγώνα

σύμφωνα με την απόδοση κάθε ομάδας. Όπως είδαμε σε προηγούμενο κεφάλαιο, η κλάση Group διαθέτει τη μέθοδο `update_ranks()`, ο κώδικας της οποίας εμφανίζεται στο απόσπασμα 3.16, η οποία αναλαμβάνει να ενημερώσει τη σειρά κατάταξης των ομάδων στον όμιλο με βάση τη βαθμολογία τους.

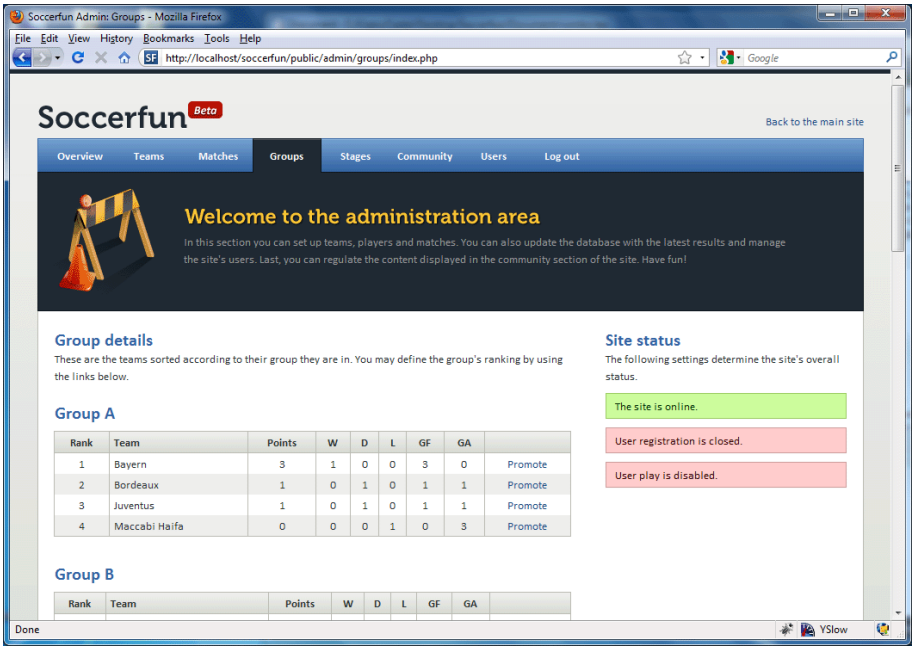
Όμως η κάθε διοργάνωση μπορεί να ορίζει διαφορετικά κριτήρια για τη σειρά κατάταξης των ομάδων που ισοβαθμούν. Για το λόγο αυτό, δημιουργούμε την ενότητα Groups, η κεντρική σελίδα της οποίας απεικονίζεται στο σχήμα 5.9.

Η σελίδα απεικονίζει όλους τους ομίλους της διοργάνωσης με βάση την υπάρχουσα σειρά κατάταξης των ομάδων. Ο διαχειριστής έχει τη δυνατότητα "χειροκίνητα" να προωθήσει (λειτουργία promote) μια ομάδα στη σειρά κατάταξης καθορίζοντας τη σωστή σειρά των ομάδων.

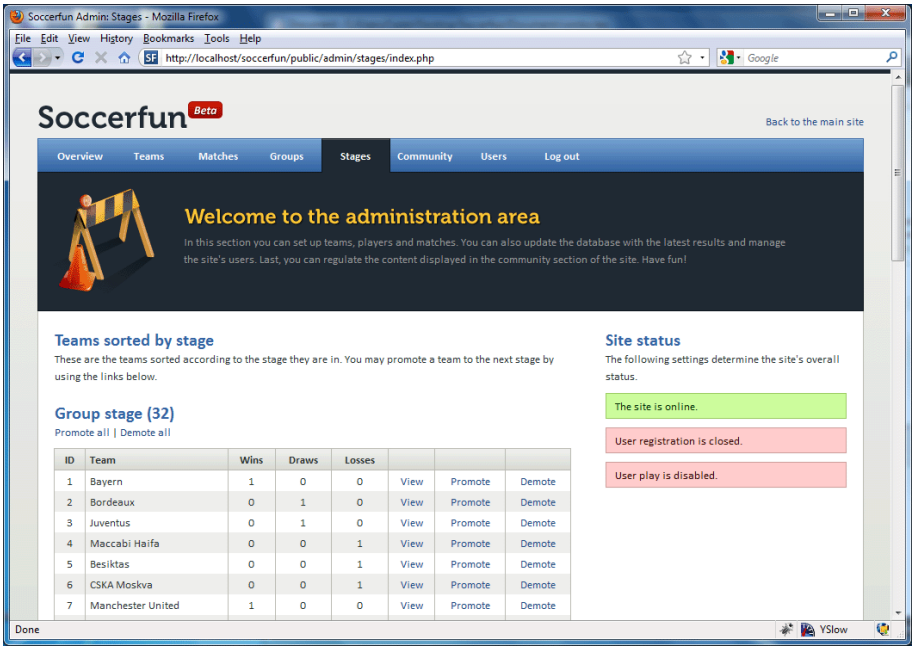
Η ενότητα Stages

Καθώς η διοργάνωση εκτυλίσσεται οι ομάδες αγωνίζονται για την πρόκριση στην επόμενη φάση μέχρι και τον τελικό. Στην ενότητα αυτή, η κεντρική σελίδα της οποίας απεικονίζεται στο σχήμα 5.10, ο διαχειριστής έχει τη δυνατότητα να δει τις ομάδες όπως αυτές κατάσσονται με βάση την φάση στην οποία βρίσκονται.

Έτσι, με χρήση των συνδέσμων πρόκρισης (promote), μπορούμε να ορίσουμε τις ομάδες που προκρίνονται από κάθε φάση της διοργάνωσης. Για να μπορούμε να αναιρέσουμε μια εσφαλμένη ενημέρωση, προσθέτουμε και το σύνδεσμο επα-



Σχήμα 5.9: Σελίδα ενημέρωσης κατάταξης των ομίλων



Σχήμα 5.10: Σελίδα ενημέρωσης της φάσης των ομάδων

ναφοράς (demote), ο οποίος επιτελεί την ακριβώς αντίθετη λειτουργία.

Συνοψίζοντας, με τις ενότητες Teams, Matches, Groups και Stages ο διαχειριστής μπορεί να διαμορφώσει τα αρχικά δεδομένα του παιχνιδιού και στη συνέχεια να ενημερώσει τη βάση με τα αποτελέσματα των αγώνων και τις πορείες των ομάδων στη διοργάνωση. Στη συνέχεια θα αναλύσουμε τις επιπλέον εργασίες που μπορούν να γίνουν μέσω άλλων ενοτήτων.

5.2.3 Γενικότερες εργασίες διαχείρισης

Εκτός από τις βασικές λειτουργίες που αναφέραμε προηγουμένως, η ενότητα Overview προσφέρει τη δυνατότητα στο διαχειριστή να ορίσει βασικές ρυθμίσεις της εφαρμογής. Οι ρυθμίσεις αυτές αντιστοιχούν σε εγγραφές σε ένα πρόσθετο πίνακα που δημιουργήσαμε στη βάση για αυτόν το σκοπό και ονομάζεται settings. Οι κυριότερες εγγραφές του πίνακα αυτού, επηρεάζουν τη συνολική κατάσταση της εφαρμογής και είναι οι εξής:

Site online Η ρύθμιση αυτή καθορίζει αν η εφαρμογή είναι διαθέσιμη στους χρήστες ή όχι. Έτσι, ο διαχειριστής μπορεί να απενεργοποιήσει τη ρύθμιση αυτή, όσο επιτελεί εργασίες συντήρησης ή τροποποίησης δομών της εφαρμογής, οι οποίες ενδέχεται να προκαλέσουν προβλήματα στους συνδεδεμένους χρήστες. Αν η εφαρμογή είναι απενεργοποιημένη, τότε σε οποιαδήποτε σελίδα κι αν επιχειρήσουν να πλοηγηθούν οι χρήστες, το αίτημά τους θα εκτραπεί σε μια σελίδα η οποία τους ενημερώνει ότι γίνονται εργασίες συντήρησης.

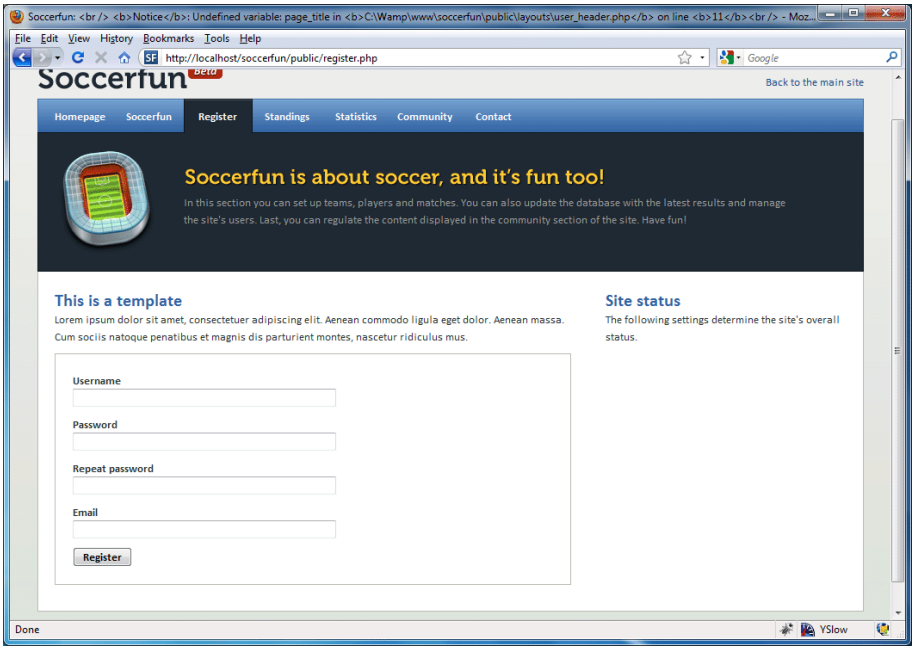
User register Η ρύθμιση αυτή καθορίζει αν επιτρέπεται ή όχι η εγγραφή νέων χρηστών στην εφαρμογή.

User play Η ρύθμιση αυτή καθορίζει αν επιτρέπεται ή όχι η συμμετοχή στα παιχνίδια. Έτσι, ο διαχειριστής έχει τη δυνατότητα να αποτρέψει την υποβολή νέων συμμετοχών στο παιχνίδι μόλις η διοργάνωση ξεκινήσει.

Όλες οι σελίδες που απευθύνονται στο χρήστη επιτρέποντάς του να επιτελέσει μια σειρά από εργασίες λαμβάνουν υπ όψην τους τις τιμές των ρυθμίσεων αυτών και καθορίζουν τη συμπεριφορά τους ανάλογα.

5.3 Περιβάλλον του χρήστη

Ανατρέχοντας και πάλι στις λειτουργικές απαιτήσεις της εφαρμογής, αυτή τη φορά από την πλευρά του χρήστη, διαπιστώνουμε ότι πρέπει να του παρέχεται η δυνατότητα να εγγραφεί στην εφαρμογή, να συμμετάσχει στο παιχνίδι και στη συνέχεια να μπορεί να βλέπει την κατάταξη των χρηστών στα διάφορα παιχνίδια, καθώς επίσης να συμβουλευτεί διάφορα στατιστικά σχετικά με τις συμμετοχές των χρηστών στα παιχνίδια συνολικά. Τέλος, για λόγους διαφάνειας των



Σχήμα 5.11: Σελίδα εγγραφής στην εφαρμογή

συμμετοχών, θα πρέπει να υπάρχει η δυνατότητα σε κάθε χρήστη να μπορεί να βλέπει τις συμμετοχές των υπολοίπων χρηστών. Στη συνέχεια θα αναλύσουμε κάθε μια από τις απαιτήσεις αυτές ξεχωριστά.

5.3.1 Εγγραφή στο site

Η εγγραφή στο site γίνεται μέσα από την ενότητα **Register** η οποία είναι διαθέσιμη μόνο αν ο διαχειριστής έχει επιτρέψει τη λειτουργία αυτή, όπως αναλύσαμε προηγουμένως. Η ενότητα αυτή απεικονίζεται στην εικόνα 5.11, στην οποία εμφανίζεται η φόρμα την οποία ο χρήστης καλείται να συμπληρώσει για να πραγματοποιήσει την εγγραφή του στην εφαρμογή. Έτσι, ο χρήστης παρέχει ένα όνομα χρήστη (username), ένα κωδικό εισόδου (password) και ένα email, το οποίο είναι απαραίτητο για την ολοκλήρωση της διαδικασίας. Ο χρήστης καλείται να εισάγει τον κωδικό εισόδου δύο φορές για να διασφαλίσουμε ότι δεν υπήρξε κάποιο λάθος κατά την πληκτρολόγηση.

Η σελίδα, αφού πραγματοποιήσει ένα πρώτο βασικό έλεγχο για την παρουσία πληροφορίας σε όλα τα πεδία της φόρμας, στη συνέχεια προχωράει στον έλεγχο της ίδιας της πληροφορίας. Έτσι, ελέγχει ότι δεν υπάρχει ήδη κάποια εγγραφή στη βάση δεδομένων με το ίδιο όνομα χρήστη ή το ίδιο email, καθώς τα δύο αυτά πεδία οφείλουν να είναι μοναδικά. Αν ο έλεγχος αυτός είναι ανεπιτυχής, ο χρήστης ειδοποιείται με κατάλληλο μήνυμα. Αν ο έλεγχος είναι επιτυχής, μια νέα εγγραφή εισάγεται στη βάση δεδομένων στον πίνακα users και η διαδι-

κασίας εγγραφής περνά στην επόμενη φάση.

Κατά την εισαγωγή της εγγραφής στη βάση δεδομένων, δημιουργούμε ένα τυχαίο μοναδικό αναγνωριστικό αριθμό τον οποίο αποθηκεύουμε στην εγγραφή αυτή. Στη συνέχεια στέλνουμε ένα email στο λογαριασμό που ο χρήστης δήλωσε κατά τη συμπλήρωση της φόρμας το οποίο περιέχει ένα σύνδεσμο προς τη σελίδα *activate.php* της εφαρμογής. Στο σύνδεσμο αυτό προστίθενται ως παράμετροι η τιμή του πεδίου *id* και η τιμή του πεδίου *activation_code* της εγγραφής του χρήστη. Μόλις ο χρήστης ακολουθήσει το σύνδεσμο αυτό, ο λογαριασμός του ενεργοποιείται και η διαδικασία εγγραφής ολοκληρώνεται. Με τον τρόπο αυτό εξασφαλίζουμε ότι ο χρήστης είναι όντως κάτοχος του συγκεκριμένου λογαριασμού email, κάτι το οποίο είναι απαραίτητο για τυχόν επαναφορά του κωδικού εισόδου του χρήστη, όπως θα δούμε στην ενότητα της ασφάλειας.

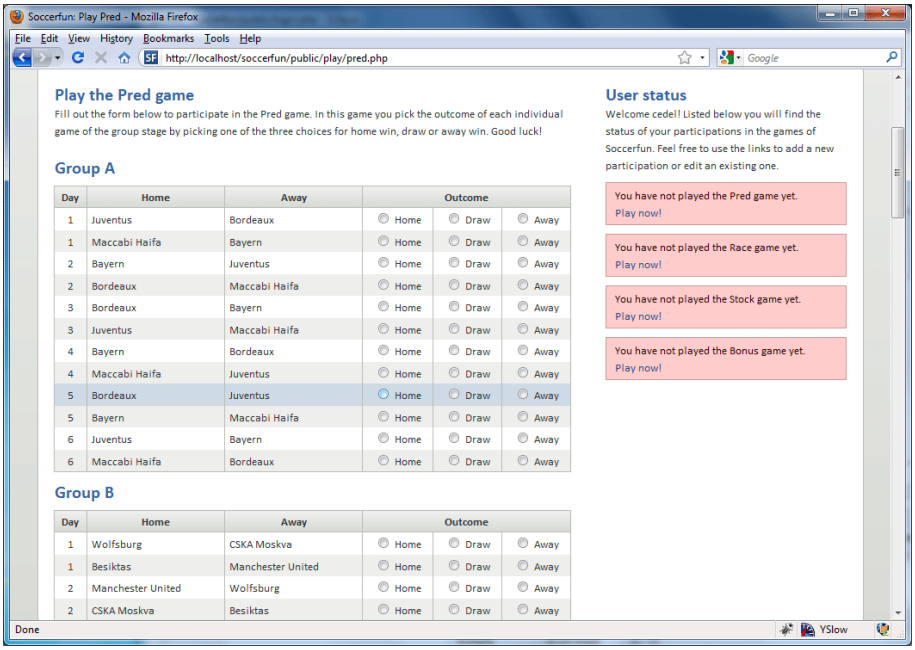
5.3.2 Συμμετοχή στο παιχνίδι

Μόλις ο χρήστης ολοκληρώσει τη διαδικασία εγγραφής και συνδεθεί στην εφαρμογή, η καρτέλα **Play** γίνεται διαθέσιμη και ο χρήστης μπορεί να συμμετάσχει στα υπο-παιχνίδια του site. Στο σημείο αυτό σημειώνουμε ότι η διαθεσιμότητα της ενότητας **Play** εξαρτάται επίσης από την αντίστοιχη ρύθμιση της ενότητας διαχείρισης της εφαρμογής, όπως εξηγήσαμε παραπάνω. Μια ενδεικτική όψη της ενότητας απεικονίζεται στο σχήμα 5.12. Στα δεξιά της σελίδας, ο χρήστης μπορεί να δει σε ποιό από τα παιχνίδια έχει συμπληρώσει τη διαδικασία συμμετοχής και μέσω κατάλληλων συνδέσμων να συμμετάσχει στα υπόλοιπα παιχνίδια ή να τροποποιήσει τα δεδομένα που έχει ήδη υποβάλει.

Το στιγμιότυπο της εικόνας 5.12 αφορά στο υπο-παιχνίδι Pred. Ο χρήστης καλείται να επιλέξει το αποτέλεσμα των αγώνων της φάσης των ομίλων ανάμεσα σε: νίκη γηπεδούχου, ισοπαλία ή νίκη φιλοξενούμενης ομάδας. Οι αγώνες εμφανίζονται ομαδοποιημένοι κατά ομίλους και η πρόβλεψη του αποτελέσματος γίνεται με τη χρήση του μηχανισμού των radio buttons, τα οποία επιτρέπουν μια μόνο επιλογή. Ο μηχανισμός αυτός επιπλέον εξασφαλίζει ότι η επιλογή θα είναι έγκυρη, σε αντίθεση με κάποιο άλλο σύστημα, όπως για παράδειγμα ένα πεδίο κειμένου στο οποίο ο χρήστης θα πληκτρολογήσει το αποτέλεσμα με τη μορφή κάποιου σημείου από τα 1, 2 ή X.

Επιπλέον, ο μηχανισμός των radio buttons διευκολύνει στον έλεγχο των στοιχείων που ο χρήστης υποβάλει. Έτσι, μόλις η φόρμα συμπληρωθεί και η συμμετοχή στο παιχνίδι υποβληθεί, ελέγχεται αν ο χρήστης έχει υποβάλει πρόβλεψη για όλους τους αγώνες και στη συνέχεια η συμμετοχή του καταχωρείται στον αντίστοιχο πίνακα της βάσης δεδομένων. Ο έλεγχος αυτός είναι εύκολος καθώς τα radio buttons είναι ομαδοποιημένα σε ένα και μόνο αντικείμενο τύπου array, το οποίο ονομάζουμε `preds`. Αν ο χρήστης δεν επιλέξει κάποια επιλογή για έναν αγώνα, τότε το αντικείμενο θα έχει μέγεθος μικρότερο από το πλήθος των αγώνων στη φάση των ομίλων και έτσι η καταχώρηση δεν θα πραγματοποιηθεί.

Αντίστοιχη λογική ακολουθούμε και για τα υπόλοιπα παιχνίδια. Οι εικόνες 5.13, 5.14 και 5.15 απεικονίζουν στιγμιότυπα της διαδικασίας συμμετοχής στα παι-



Σχήμα 5.12: Ενδεικτική σελίδα συμμετοχής στο παιχνίδι Pred

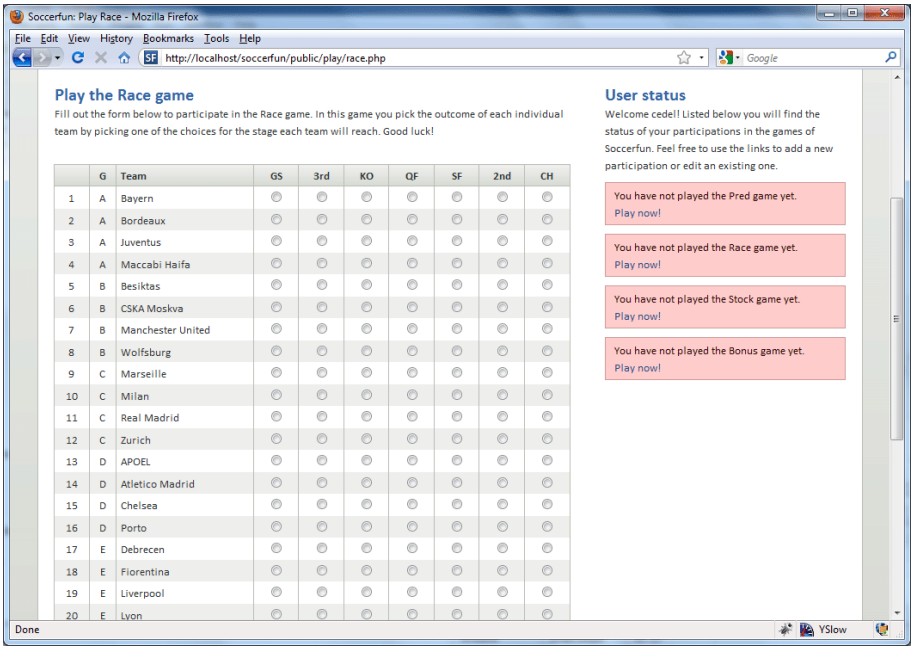
χνίδια Race, Stock και Bonus αντίστοιχα.

Σε όλες τις φόρμες συμμετοχής στα παιχνίδια οι διαθέσιμες επιλογές του χρήστη έχουν προκαθοριστεί, όπου αυτό είναι δυνατό. Μοναδική εξαίρεση αποτελεί η φόρμα συμμετοχής στο παιχνίδι Stock, όπου ο χρήστης καλείται να συμπληρώσει ο ίδιος τον αριθμό των μαρκών που ποντάρει σε κάθε ομάδα. Ειδικά για την περίπτωση αυτή υπάρχει ένα ακόμη πεδίο το οποίο ενημερώνεται αυτόματα κάθε φορά που ο χρήστης μεταβάλει την τιμή ενός πεδίου, ώστε ο χρήστης να είναι ενήμερος για το ποσό που έχει ποντάρει ως εκείνη τη στιγμή.

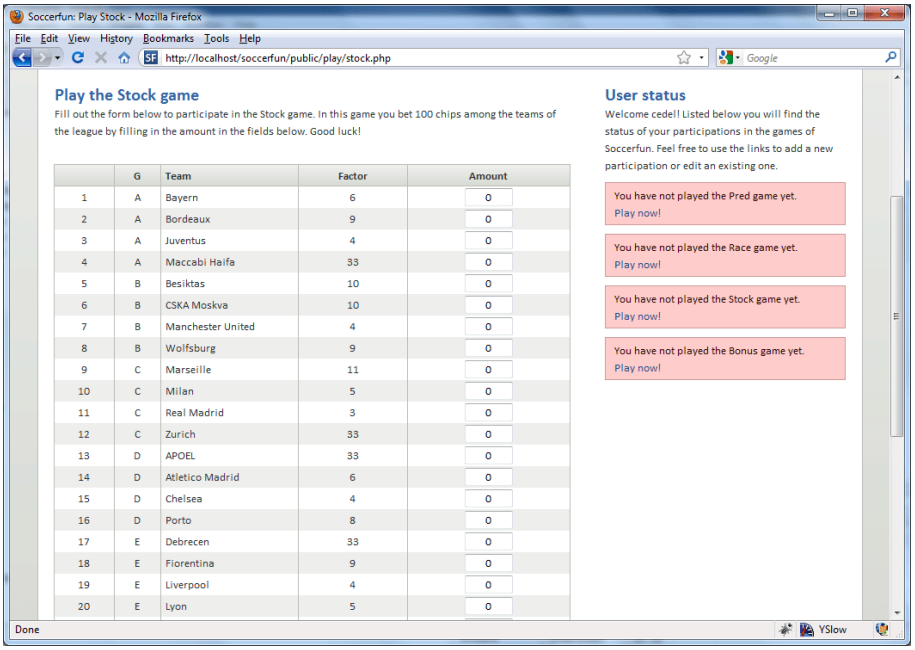
5.3.3 Παρακολούθηση κατάταξης, στατιστικά και διαφάνεια

Μετά τη λήξη της διορίας συμμετοχής στα παιχνίδια της εφαρμογής και την έναρξη της διοργάνωσης, ο χρήστης θα πρέπει να είναι σε θέση να παρακολουθεί την κατάταξή του στα διάφορα υπο-παιχνίδια. Η ενότητα Standings έχει δημιουργηθεί ακριβώς για αυτό το σκοπό. Η κατάταξη στο κάθε παιχνίδι παρουσιάζεται με τη μορφή ενός πίνακα στον οποίο η θέση του συνδεδεμένου χρήστη υποδεικνύεται με ειδική μορφοποίηση της αντίστοιχης γραμμής του πίνακα.

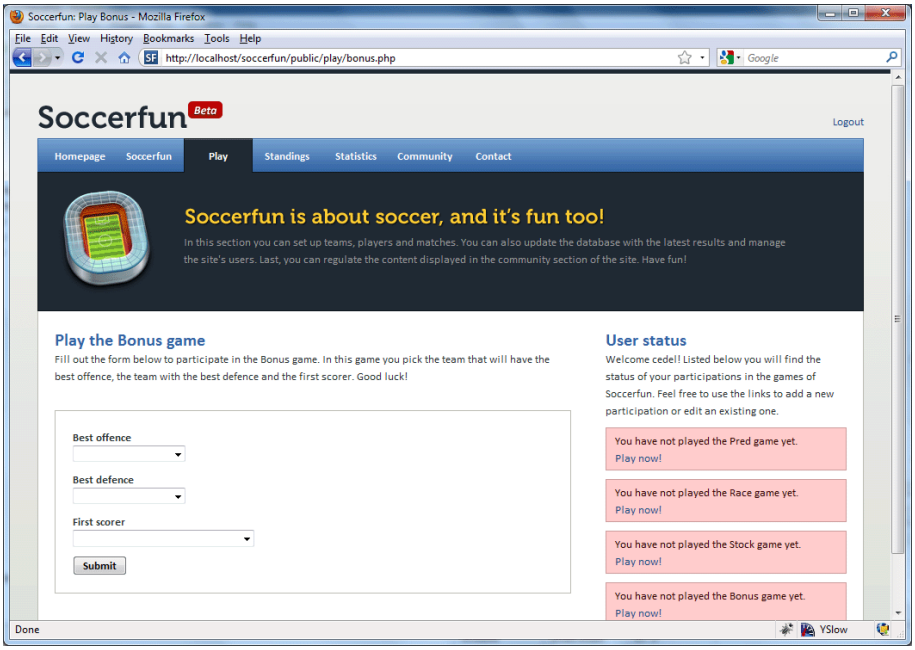
Για λόγους διαφάνειας και αξιοπιστίας της εφαρμογής, το όνομα του κάθε χρήστη αποτελεί ένα σύνδεσμο προς μια νέα σελίδα, στην οποία ο συνδεδεμένος χρήστης μπορεί να δει όλες τις επιλογές της συμμετοχής οποιουδήποτε άλλου χρήστη και να τις συγκρίνει με τις δικές του. Έτσι, διασφαλίζουμε ότι κάθε χρήστης μπορεί να επιβεβαιώσει ότι η βαθμολογία των άλλων χρηστών είναι σωστή.



Σχήμα 5.13: Ενδεικτική σελίδα συμμετοχής στο παιχνίδι Race



Σχήμα 5.14: Ενδεικτική σελίδα συμμετοχής στο παιχνίδι Stock



Σχήμα 5.15: Ενδεικτική σελίδα συμμετοχής στο παιχνίδι Bonus

Επιπλέον, στην ενότητα Statistics, ο χρήστης μπορεί να συμβουλευτεί συνολικά στατιστικά για τις συμμετοχές στην εφαρμογή, όπως για παράδειγμα πόσοι χρήστες προέβλεψαν ότι το αποτέλεσμα ενός αγώνα θα είναι ισοπαλία ή ότι μια ομάδα θα φτάσει στον τελικό της διοργάνωσης.

5.4 Ζητήματα ασφάλειας

Ένα από τα σημαντικότερα στοιχεία κάθε εφαρμογής είναι η ασφάλεια και η προστασία της από εσφαλμένη χρήση, κακόβουλη ή μη. Μέχρι στιγμής, έχουμε αναφέρει ένα πρώτο αλλά σημαντικό βήμα στην προστασία των δεδομένων της εφαρμογής: όπου αυτό είναι δυνατό, τα πεδία σε μια φόρμα εισαγωγής ή τροποποίησης δεδομένων αντικαθίστανται από μηχανισμούς επιλογής ανάμεσα σε προκαθορισμένα δεδομένα. Για παράδειγμα η επιλογή των ομάδων που μετέχουν σε έναν αγώνα γίνεται μέσα από μενού επιλογής, καθώς η πληκτρολόγηση των ονομάτων των ομάδων όχι μόνο δε διευκολύνει το χρήστη, αλλά επιπλέον δημιουργεί τις συνθήκες για λανθασμένες εισαγωγές στοιχείων.

Επιπλέον, ο έλεγχος μιας φόρμας για εσφαλμένα δεδομένα είναι όχι μόνο χρήσιμος αλλά και αναγκαίος για τη διατήρηση της ακεραιότητας των δεδομένων που βρίσκονται στη βάση. Για παράδειγμα, χωρίς τον έλεγχο των δεδομένων εισαγωγής μιας φόρμας πριν προβούμε σε λειτουργίες πάνω στα δεδομένα της εφαρμογής, θα μπορούσαν να δημιουργηθούν εγγραφές στον πίνακα των ομάδων

```
public function open_connection() {
  $this->connection = mysql_connect(DB_HOST, DB_USER, DB_PASS);
  if (!$this->connection) {
    die("Database connection failed: " . mysql_error());
  } else {
    $db_select = mysql_select_db(DB_NAME, $this->connection);
    if (!$db_select) {
      die("Database selection failed: " . mysql_error());
    }
  }
}
```

Κώδικας 5.1: Η μέθοδος open_connection()

με το ίδιο όνομα ή χωρίς κάποια τιμή στο πεδίο του συντελεστή. Στη συνέχεια, θα εστιάσουμε ακόμη περισσότερο στη βάση δεδομένων και στα μέτρα προστασίας που μπορούμε να πάρουμε.

5.4.1 Προστασία της βάσης δεδομένων

Η βάση δεδομένων βρίσκεται στην καρδιά της εφαρμογής μας και επομένως η τήρησή της σε καλή κατάσταση αποτελεί βασική μας προτεραιότητα. Ένα απλό αλλά πολύ σημαντικό βήμα είναι ο καθορισμός των δικαιωμάτων του χρήστη της βάσης.

Οι λειτουργίες που επιτελούνται από την εφαρμογή πάνω στη βάση δεδομένων γίνονται εξ ολοκλήρου μέσω του αντικειμένου `$database`. Το αντικείμενο αυτό αποτελεί ένα στιγμιότυπο της κλάσης *MySQLDatabase* και διαθέτει όλες τις μεθόδους της κλάσης αυτής. Η μέθοδος που μας απασχολεί στο σημείο αυτό είναι η `open_connection()` και την υπενθυμίζουμε στον κώδικα 5.1.

Η μέθοδος αυτή ξεκινά μια νέα σύνδεση με το σύστημα βάσεων δεδομένων και επιλέγει τη συγκεκριμένη βάση που υποστηρίζει την εφαρμογή. Οι παράμετροι `DB_USER` και `DB_PASS` ορίζουν το όνομα χρήστη και τον κωδικό εισόδου που χρησιμοποιούνται για τη σύνδεση. Επομένως οι λειτουργίες που επιτελούνται στη βάση δεδομένων μέσω της εφαρμογής μας περιορίζονται σύμφωνα με τα δικαιώματα του χρήστη αυτού. Καθώς οι λειτουργίες που περιγράψαμε μέχρι τώρα, περιορίζονται μόνο σε πράξεις `SELECT`, `INSERT`, `UPDATE` και `DELETE` είναι λογικό να περιορίσουμε τα δικαιώματα του χρήστη στις πράξεις αυτές. Για το σκοπό αυτό αποφεύγουμε να κάνουμε χρήση του λογαριασμού του κεντρικού διαχειριστή της βάσης δεδομένων (root) και δημιουργούμε ένα νέο χρήστη με περιορισμένα δικαιώματα ειδικά για την εφαρμογή μας.

Με τον τρόπο αυτό, ακόμη κι αν το όνομα και ο κωδικός του χρήστη αυτού γίνουν γνωστά, το σύστημα βάσεων δεδομένων είναι προστατευμένο καθώς οι άλλες βάσεις που φιλοξενούνται σε αυτό παραμένουν προστατευμένες. Επιπλέον η βάση δεδομένων της εφαρμογής έχει μια, έστω βασική προστασία, καθώς η δομή της δεν μπορεί να μεταβληθεί καθώς ο χρήστης δεν έχει δικαιώματα σε

```
$sql  = "INSERT INTO users (username, password, email) ";
$sql .= "VALUES (";
$sql .= "'".$username."',";
$sql .= "'".$password."',";
$sql .= "'".$email."')";
$database->query($sql);
```

Κώδικας 5.2: Παράδειγμα εισαγωγής χρήστη στη βάση δεδομένων (PHP)

```
INSERT INTO users (username, password, email)
VALUES ('joe', 'secret', 'joe@sample.com');
```

Κώδικας 5.3: Παράδειγμα εισαγωγής χρήστη στη βάση δεδομένων (SQL)

λειτουργίες όπως η εισαγωγή ή η διαγραφή πινάκων κλπ.

Ανάμεσα όμως στα δικαιώματα του χρήστη που δημιουργήσαμε για τις ανάγκες τις εφαρμογής μας, είναι και το δικαίωμα της διαγραφής εγγραφών από τη βάση δεδομένων. Για το λόγο αυτό, ιδιαίτερη προσοχή πρέπει να ληφθεί στο χειρισμό των δεδομένων που λαμβάνονται από το χρήστη της εφαρμογής μέσω των πεδίων μιας φόρμας. Για να γίνει κατανοητός ο κίνδυνος θα εξετάσουμε ένα πρακτικό παράδειγμα.

Ας υποθέσουμε ότι ένας κακόβουλος χρήστης επιχειρεί να εκμεταλευτεί προβλήματα στον κώδικα της εφαρμογής και για το σκοπό αυτό χρησιμοποιεί τη φόρμα εγγραφής στην εφαρμογή, όπως αυτή απεικονίζεται στην εικόνα 5.11. Η σελίδα που χειρίζεται την εγγραφή, λαμβάνει τις τιμές που παρέχονται από το χρήστη και τελικά καταλήγει στην εκτέλεση ενός ερωτήματος SQL, το οποίο θα μπορούσε να έχει τη δομή που απεικονίζεται στον κώδικα 5.2.

Στον κώδικα αυτό οι μεταβλητές `$username`, `$password` και `$email` αναλογούν στις τιμές των αντίστοιχων πεδίων της φόρμας. Έτσι αν για παράδειγμα οι τιμές αυτές είναι *joe*, *secret* και *joe@sample.com*, τότε ο κώδικας SQL που θα προκύψει θα είναι αυτός που απεικονίζεται στο απόσπασμα 5.3.

Τί θα συμβεί όμως αν ο κακόβουλος χρήστης στο πεδίο email, αντί για κάποια κανονική διεύθυνση, εισάγει το κείμενο *'); DELETE FROM users*; Τότε το παραγώμενο ερώτημα θα είναι αυτό του κώδικα 5.4 και θα έχει σαν αποτέλεσμα τη διαγραφή όλων των χρηστών από τη βάση δεδομένων δημιουργώντας τεράστιο πρόβλημα στην εφαρμογή μας.

Ο κίνδυνος από τέτοιου είδους ενέργειες εκμηδενίζεται με τη χρήση της με-

```
INSERT INTO users (username, password, email)
VALUES ('joe', 'secret', ''); DELETE FROM users;
```

Κώδικας 5.4: Παράδειγμα κακόβουλου ερωτήματος στη βάση δεδομένων (SQL)

θόδου `escape_value()` της κλάσης της βάσης δεδομένων, η οποία αναλαμβάνει να μετατρέψει τα δεδομένα που εισάγονται από το χρήστη σε μορφή ασφαλή για χρήση με τη βάση δεδομένων. Ο κώδικας της μεθόδου παρατίθεται μαζί με τον πλήρη κώδικα της κλάσης στο παράρτημα Β′ και ουσιαστικά μετατρέπει του επικίνδυνους χαρακτήρες όπως τα εισαγωγικά και τα ερωτηματικά σε ασφαλή μορφή μέσω της μεθόδου escape. Έτσι, κάθε εισαγωγή από το χρήστη αντιμετωπίζεται σαν ένα εννιαίο string χαρακτήρων και όχι σαν πιθανές εντολές SQL.

Όλες οι μέθοδοι των κλάσεων της εφαρμογής που σχετίζονται με λειτουργίες πάνω στη βάση δεδομένων κάνουν χρήση της μεθόδου `escape_value()` για να διασφαλίσουν ότι οι τιμές των ερωτημάτων είναι ασφαλείς για τη βάση δεδομένων. Επιπλέον, το γεγονός ότι οι λειτουργίες αυτές επιτελούνται πάντοτε με τη χρήση των μεθόδων που έχουμε δημιουργήσει στις διάφορες κλάσεις της εφαρμογής, εξασφαλίζει ότι η βάση θα είναι πάντοτε προστατευμένη από τέτοιου είδους επιθέσεις.

5.4.2 Προστασία των δεδομένων των χρηστών

Κατά την εγγραφή τους, οι χρήστες δηλώνουν το όνομα χρήστη (username), το email και τον κωδικό εισόδου (password) που επιθυμούν να χρησιμοποιούν για την πρόσβαση στην εφαρμογή. Είναι σύνηθες για τους χρήστες να χρησιμοποιούν τον ίδιο ή παρόμοιο κωδικό για πολλές διαφορετικές online εφαρμογές. Έτσι, αν κάποιος αποκτήσει πρόσβαση στον κωδικό αυτό μέσω της εφαρμογής, θα μπορέσει να τον χρησιμοποιήσει και σε άλλες περιπτώσεις. Για το λόγο αυτό θα πρέπει το πεδίο αυτό του πίνακα **users** της βάσης δεδομένων να είναι προστατευμένο.

Η προστασία αυτή προσφέρεται με την κρυπτογράφηση της πληροφορίας. Η μέθοδος `sha1()` της PHP μετατρέπει ένα string εισόδου σε κρυπτογραφημένη μορφή. Η μορφή αυτή είναι μια σειρά 40 αλφαριθμητικών χαρακτήρων ανεξάρτητα από το μέγεθος της εισόδου. Για παράδειγμα η είσοδος `hello` αντιστοιχεί στην κρυπτογραφημένη μορφή *aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d*. Η διαδικασία αυτή ισχύει μόνο προς μια κατεύθυνση, με την έννοια ότι από την κρυπτογραφημένη μορφή είναι πρακτικά αδύνατο να εξαχθεί η αρχική πληροφορία. Έτσι, κατά την εγγραφή ενός χρήστη στην εφαρμογή, και κατ' επέκταση κατά τη δημιουργία μιας νέας εγγραφής στον πίνακα users στη βάση δεδομένων, ο κωδικό εισόδου αποθηκεύεται σε κρυπτογραφημένη μορφή.

Όποτε ο χρήστης συνδέεται με την εφαρμογή παρέχοντας το όνομα χρήστη και τον κωδικό εισόδου του, ο παρεχόμενος αυτός κωδικός κρυπτογραφείται μέσω της συνάρτησης `sha1()` και το αποτέλεσμα συγκρίνεται με την πληροφορία που είναι αποθηκευμένη στη βάση δεδομένων. Όπως είναι προφανές, με τη διαδικασία αυτή εξασφαλίζεται ότι ο μόνος που γνωρίζει τον κωδικό εισόδου είναι ο ίδιος ο χρήστης που τον καθορίζει. Ακόμη και ο διαχειριστής της βάσης δεδομένων δεν είναι σε θέση να εξάγει τον κωδικό, καθώς αυτός αποθηκεύεται εξαρχής σε κρυπτογραφημένη μορφή.

Βεβαίως, η τεχνική αυτή παρουσιάζει ένα βασικό μειονέκτημα: ακριβώς επειδή

κανείς άλλος δεν είναι σε θέση να διαβάσει το αρχικό password, αν ο χρήστης ξεχάσει τον κωδικό εισόδου του δεν υπάρχει η δυνατότητα να του το υπενθυμίσουμε με κάποιο τρόπο. Η συνήθης πρακτική σε αυτήν την περίπτωση είναι να στέλνουμε στον χρήστη ένα email στο λογαριασμό που δήλωσε κατά την εγγραφή του, το οποίο θα του επιτρέπει να ορίσει εκ νέου ένα κωδικό εισόδου. Ο συμβιβασμός αυτός είναι μια διαδεδομένη πρακτική, της οποίας το τίμημα αξίζει να πληρωθεί προκειμένου τα δεδομένα των χρηστών να είναι ασφαλή.

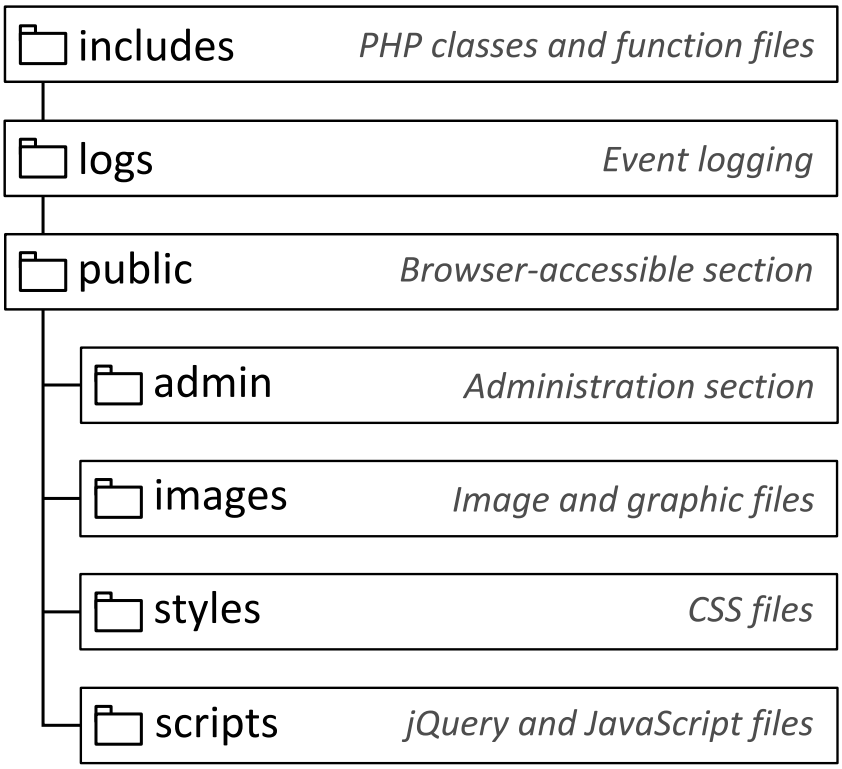
5.4.3 Πρόσβαση στις λειτουργίες της εφαρμογής

Η εφαρμογή απαρτίζεται από πολλαπλά αρχεία και διαφορετικές λειτουργίες, οι πρόσβαση στις οποίες πρέπει να καθορίζεται για κάθε χρήστη. Για παράδειγμα, όλες οι λειτουργίες διαχείρισης πρέπει να είναι προσβάσιμες μόνο από το διαχειριστή της εφαρμογής. Επιπλέον, δεν έχει νόημα κάποιος χρήστης να έχει πρόσβαση στην ενότητα συμμετοχής στο παιχνίδι, αν προηγουμένως δεν έχει ολοκληρώσει τη διαδικασία εγγραφής. Πριν αναλύσουμε όμως τη μέθοδο που ρυθμίζει την πρόσβαση στις διάφορες ενότητες του παιχνιδιού, θα εξετάσουμε την προστασία των ίδιων των αρχείων της εφαρμογής.

Η δομή σε επίπεδο αρχείων

Κατά βάση, η εφαρμογή αποτελείται από ένα σύνολο αρχείων. Τα αρχεία αυτά μπορούν είναι κλάσεις τις εφαρμογής, αρχεία CSS, εικόνες και γραφικά που χρησιμοποιούνται για την εμφάνιση της πληροφορίας, αρχεία PHP που χειρίζονται την αλληλεπίδραση με το χρήστη, κλπ. Η διάταξη των αρχείων αυτών στο επίπεδο του δίσκου απεικονίζεται στο σχήμα 5.16.

Ο φάκελος *public* αποτελεί το τμήμα της εφαρμογής που είναι προσβάσιμο μέσα από έναν browser και περιέχει όλες τις σελίδες που απευθύνονται τόσο στον απλό χρήστη, όσο και στο διαχειριστή της εφαρμογής. Ο φάκελος *logs* περιέχει αρχεία κειμένου, στα οποία καταγράφονται διάφορα γεγονότα, όπως για παράδειγμα η ημερομηνία και η ώρα της κάθε σύνδεσης του διαχειριστή στην εφαρμογή. Τα αρχεία αυτά δεν είναι προσβάσιμα μέσω κάποιου browser, παρά μόνο μέσω πρόσβασης στο ίδιο το αρχείο σε επίπεδο δίσκου. Έτσι, ο διαχειριστής μπορεί με κάποιο τρόπο, όπως για παράδειγμα μέσω FTP, να έχει πρόσβαση στο αρχείο αυτό και να ελέγξει τα περιεχόμενά του. Τέλος, ο φάκελος *includes* περιέχει όλες τις κλάσεις που υλοποιήσαμε για την εφαρμογή και οι οποίες παρατίθενται στο παράρτημα Β′. Τα αρχεία αυτά επίσης δεν είναι προσβάσιμα μέσω του browser των χρηστών. Αντίθετα, είναι διαθέσιμα προς χρήση από τα υπόλοιπα αρχεία της εφαρμογής, τα οποία βασίζονται σε αυτά για να επιτελέσουν τις διάφορες λειτουργίες τους, όπως εξηγήσαμε παραπάνω. Με τον τρόπο αυτό εξασφαλίζουμε ότι μόνο ο προγραμματιστής της εφαρμογής έχει πρόσβαση στα αρχεία αυτά και επομένως ότι διατηρούμε τον πλήρη έλεγχο της λειτουργικότητας της εφαρμογής.



Σχήμα 5.16: Διάταξη της εφαρμογής σε επίπεδο αρχείων

Η κλάση session

Ένας χρήστης της εφαρμογής χρησιμοποεί κάποιον browser για να έχει πρόσβαση σε αυτήν. Ο browser όμως δεν διατηρεί την πληροφορία που ο χρήστης εισάγει από σελίδα σε σελίδα με αποτέλεσμα η πληροφορία αυτή να μην μπορεί να επαναχρησιμοποιηθεί. Για παράδειγμα αν ο χρήστης συνδεθεί με την εφαρμογή σε μια σελίδα και πλοηγηθεί σε κάποια άλλη ενότητα, ο browser δεν είναι σε θέση να γνωρίζει από μόνος του ότι η σύνδεση έχει ήδη πραγματοποιηθεί και έτσι ο χρήστης θα πρέπει και πάλι να συμπληρώσει τα στοιχεία εισόδου του.

Η λύση σε αυτό το πρόβλημα παρέχεται μέσω των sessions. Κάθε φορά που ένας χρήστης επισκέπτεται το site της εφαρμογής, ο server δημιουργεί αυτόματα ένα μοναδικό αναγνωριστικό *Session ID*, το οποίο αποθηκεύεται στον browser του χρήστη. Στη μεριά του server δημιουργείται ένα ξεχωριστό αρχείο για το συγκεκριμένο αναγνωριστικό, στο οποίο μπορεί να αποθηκευτεί πληροφορία. Έτσι, όσο το παράθυρο του browser μένει ανοιχτό σε κάποια από τις σελίδες της εφαρμογής, η πληροφορία αυτή είναι διαθέσιμη ανεξάρτητως της σελίδας που επισκέπτεται ο χρήστης. Το γεγονός ότι η πληροφορία των sessions αποθηκεύεται σε αρχεία στον server διασφαλίζει ότι μόνο η εφαρμογή μας έχει πρόσβαση στα αρχεία αυτά και στην πληροφορία που περιέχουν και όχι ο ίδιος ο χρήστης. Η αποθήκευση και ανάκτηση της πληροφορίας που περιέχεται στα αρχεία αυτά γίνεται μέσω της global μεταβλητής `$_SESSION`, η οποία ουσιαστικά είναι ένας συσχετιστικός πίνακας. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα νέο πεδίο της μεταβλητής αυτής και να θέσουμε την τιμή της με την εντολή `$_SESSION["user_id"]=4`. Έτσι, σε κάποια επόμενη σελίδα μπορούμε να ανακτήσουμε την πληροφορία αυτή μέσω της μεταβλητής `$_SESSION["user_id"]`.

Για τις ανάγκες της εφαρμογής μας χρησιμοποιούμε τα sessions κυρίως για να ελέγχουμε αν κάποιος χρήστης είναι συνδεδεμένος στην εφαρμογή. Έτσι δημιουργούμε την κλάση **session**, ο κώδικας της οποίας παρατίθεται στο παράρτημα Β′. Η βασική μεταβλητή της κλάσης είναι η `user_id`. Μόλις κάποιος χρήστης συνδεθεί στην εφαρμογή παρέχοντας τα σωστά στοιχεία σύνδεσης, η τιμή της μεταβλητής αυτής τίθεται ίση με την τιμή του πεδίου *id* του συγκεκριμένου χρήστη. Η τιμή αυτή θα παραμείνει διαθέσιμη όσο διαρκεί το συγκεκριμένο session και όσο ο χρήστης δεν πραγματοποιήσει κάποια αποσύνδεση (log out).

Έχοντας την τιμή αυτή στη διάθεσή μας, είμαστε σε θέση να ελέγχουμε αν κάποιος χρήστης έχει πρόσβαση σε κάποια ενότητα της εφαρμογής. Έτσι, αν κάποιος χρήστης επιχειρήσει να πλοηγηθεί στην ενότητα διαχείρισης της εφαρμογής (admin), μπορούμε εύκολα να ελέγξουμε αν ο χρήστης αυτός έχει δικαίωμα πρόσβασης στην ενότητα αυτή και αν όχι να αποτρέψουμε την πρόσβαση αυτή επαναφέροντας για παράδειγμα το χρήστη στην αρχική σελίδα της εφαρμογής. Αντίστοιχα, αν κάποιος χρήστης επιχειρήσει να συμμετάσχει στο παιχνίδι χωρίς προηγουμένως να έχει συνδεθεί στην εφαρμογή, με άλλα λόγια αν δεν έχει τεθεί η τιμή στη μεταβλητή `user_id`, μπορούμε να επαναφέρουμε το χρήστη στη σελίδα σύνδεσης στην εφαρμογή.

Επιπλέον, όπως αναφέραμε προηγουμένως, στα αρχεία session μπορούν να

αποθηκευθούν πολλών ειδών πληροφορίες οι οποίες θα είναι διαθέσιμες από άλλες σελίδες της εφαρμογής. Έτσι, μπορούμε να χρησιμοποιήσουμε τα sessions για να αποθηκεύουμε μηνύματα λάθους ή άλλες ειδοποιήσεις προς τον χρήστη, οι οποίες θα είναι διαθέσιμες από την επόμενη σελίδα που θα εμφανιστεί. Για παράδειγμα, ας υποθέσουμε ότι ο χρήστης επιχειρεί να τροποποιήσει κάποια συμμετοχή του στα παιχνίδια. Αφού πραγματοποιήσει τις αλλαγές που επιθυμεί και υποβάλει το αίτημα του, στην επόμενη σελίδα μπορούμε να τον ενημερώσουμε για το αποτέλεσμα του αιτήματος μέσω κάποιου μηνύματος κατάλληλα μορφοποιημένου (ανάλογα με το αν η διαδικασία ήταν επιτυχής ή όχι). Στη συνέχεια αυτό το μήνυμα μπορεί να διαγραφεί από το αρχείο session καθώς δεν έχει νόημα η απεικόνισή του στην επόμενη σελίδα. Ο μηχανισμός των μηνυμάτων είναι τμήμα της κλάσης **session** και οι μέθοδοι που χρησιμοποιούνται παρατίθενται στο παράρτημα Β΄.

Κεφάλαιο 6

Επίλογος

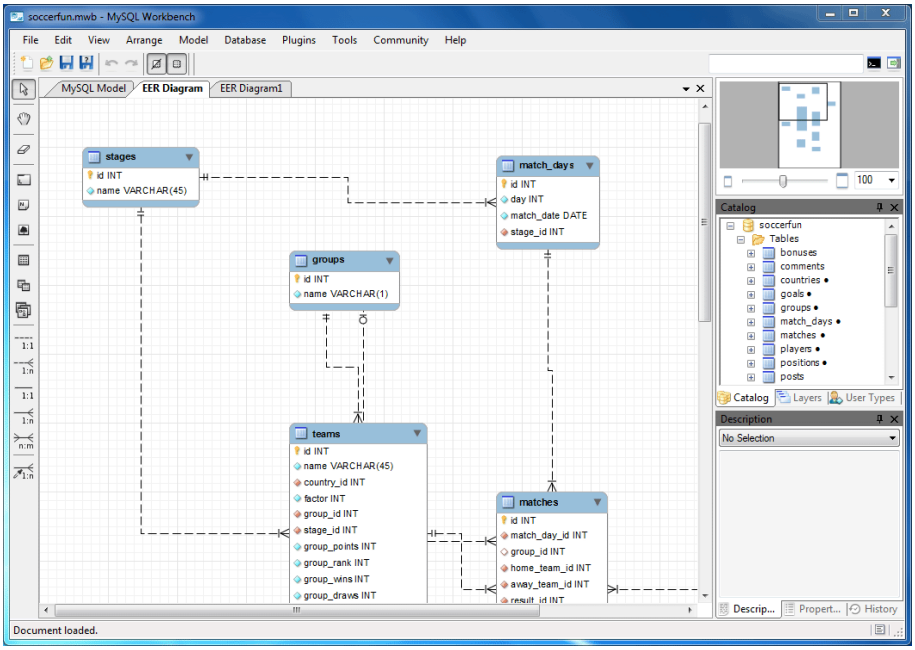
Στην ενότητα αυτή θα αναφερθούμε σε κάποιες παρατηρήσεις που έγιναν κατά την υλοποίηση της εφαρμογής και στη σημασία της δομής της εφαρμογής στη βιωσιμότητά της. Στη συνέχεια θα αναλύσουμε τις δυνατότητες που προσφέρονται για μελλοντικές επεκτάσεις και τροποποιήσεις της εφαρμογής και θα αναφερθούμε ενδεικτικά στην ενότητα Community ως παράδειγμα επέκτασης.

6.1 Σημειώσεις κατά την υλοποίηση

Όπως αναφέραμε αναλυτικά στις προηγούμενες ενότητες, μια δυναμική online εφαρμογή περιλαμβάνει πολλαπλά επίπεδα λειτουργιών και αποτελείται από διαφορετικά συστατικά τα οποία συνεργάζονται για την εκτέλεση των λειτουργιών αυτών. Είναι σημαντικό να χρησιμοποιούμε εργαλεία τα οποία μας επιτρέπουν να έχουμε τον πλήρη έλεγχο όλων των στοιχείων της εφαρμογής και να μας διευκολύνουν τόσο κατά την ανάπτυξη, όσο και κατά την υλοποίηση της εφαρμογής. Στη συνέχεια θα αναφερθούμε στα σημαντικότερα από τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

MySQL Workbench

Το MySQL Workbench είναι ένα δωρεάν εργαλείο το οποίο σχεδιάστηκε αποκλειστικά για χρήση με το σύστημα βάσεων δεδομένων MySQL. Στην εικόνα 6.1 απεικονίζεται ένα χαρακτηριστικό στιγμιότυπο του προγράμματος. Με τη βοήθεια του εργαλείου, μπορούμε να σχεδιάσουμε τη βάση δεδομένων της εφαρμογής, ορίζοντας τα πεδία του κάθε πίνακα και τις σχέσεις που εμφανίζονται ανάμεσα στους πίνακες μέσω των foreign keys. Επιπλέον, μπορούμε να χρησιμοποιήσουμε μια ακόμη λειτουργία της εφαρμογής η οποία μας επιτρέπει να παράγουμε σχεσιακά διαγράμματα για καλύτερη εποπτεία των πινάκων και των σχέσεων μεταξύ τους.



Σχήμα 6.1: Ενδεικτική εικόνα του MySQL Workbench

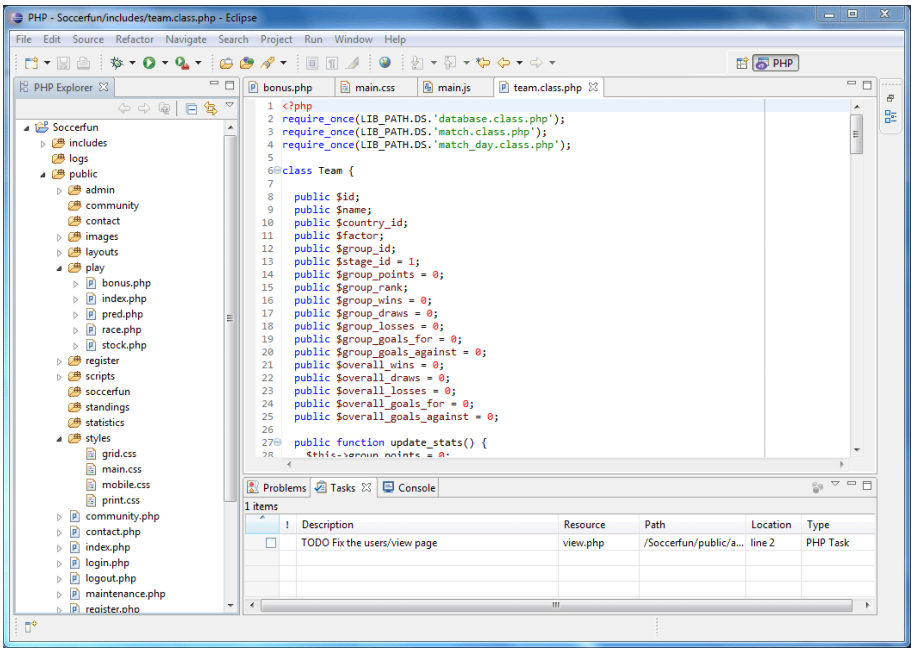
Eclipse PHP Development Tools

Το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse αποτελεί μια από τις πλέον διαδεδομένες εφαρμογές ανάπτυξης λογισμικού. Ειδικά για την ανάπτυξη εφαρμογών βασισμένες σε PHP, η έκδοση Eclipse PHP Development Tools είναι μια από τις πιο ολοκληρωμένες λύσεις. Στην εικόνα 6.2 απεικονίζεται ένα χαρακτηριστικό στιγμιότυπο χρήσης του εργαλείου.

Η διεξοδική παρουσίαση του Eclipse PDT ξεφεύγει από τα πλαίσια της παρούσας διπλωματικής εργασίας. Όμως αξίζει να σταθούμε σε ορισμένα χαρακτηριστικά του, τα οποία αποδείχθηκαν εξαιρετικά χρήσιμα κατά την ανάπτυξη της εφαρμογής.

- Το Eclipse PDT υποστηρίζει πλήρως τον προγραμματισμό σε γλώσσα PHP. Έτσι, διαθέτει μια σειρά από διευκολύνσεις για τον προγραμματιστή, όπως η χρωματική μορφοποίηση του κώδικα, ενσωματωμένο σύστημα αναφοράς στις λειτουργίες της γλώσσας PHP, αλλά επίσης το σύστημα code assist, το οποίο βοηθά στην πρόσβαση στα γνωρίσματα (attributes) και τις μεθόδους (methods) των κλάσεων που έχουμε ορίσει. Το γεγονός αυτό επιταχύνει σημαντικά τη διαδικασία ανάπτυξης και μειώνει αισθητά τα σφάλματα κατά την πληκτρολόγηση.

- Μια από τις ενότητες του περιβάλλοντος ανάπτυξης είναι ο PHP Explorer, ο οποίος εμφανίζεται στο αριστερό τμήμα της εικόνας 6.2. Χρησιμοποιώ-



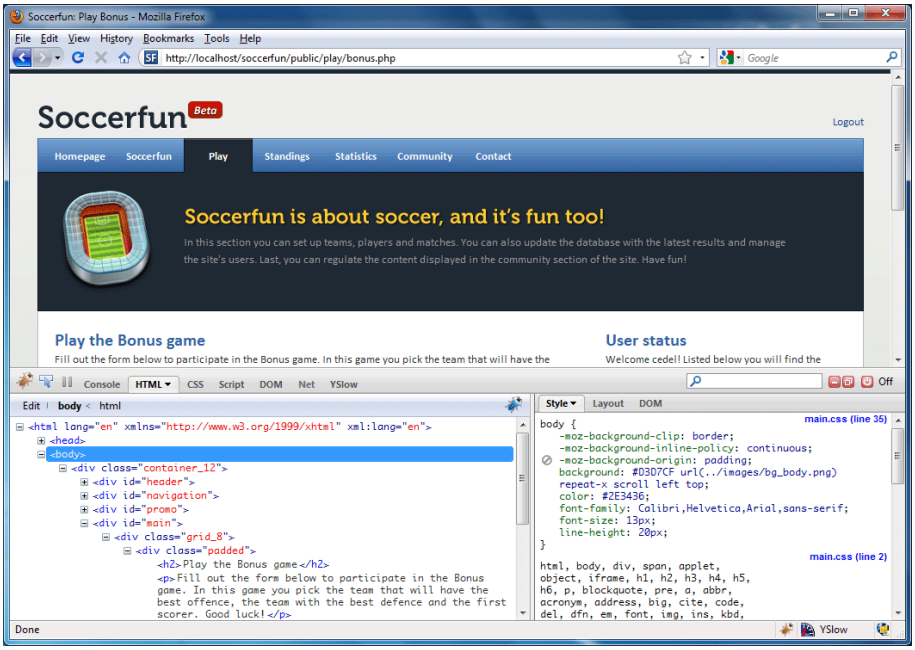
Σχήμα 6.2: Ενδεικτική εικόνα του Eclipse PHP Development Tools

ντας την ενότητα αυτή, ο προγραμματιστής μπορεί να καθορίσει τη δομή της εφαρμογής σε επίπεδο αρχείων και να έχει άμεση πρόσβαση στα διαφορετικά αρχεία κλάσεων ή άλλων λειτουργιών της εφαρμογής.

- Εκτός από τον προγραμματισμό σε γλώσσα PHP, το εργαλείο επιπλέον υποστηρίζει μια σειρά από επιπλέον τεχνολογίες οι οποίες σχετίζονται άμεσα με την ανάπτυξη μιας online εφαρμογής. Έτσι, χρησιμοποιώντας το Eclipse PDT, μπορούμε να συντάξουμε κώδικα τόσο για τα αρχεία CSS όσο και για τα αρχεία JavaScript της εφαρμογής, διατηρώντας τον έλεγχο της διαδικασίας ανάπτυξης σε ένα και μόνο εργαλείο.

Ο browser Firefox

Το τελευταίο εργαλείο στο οποίο θα αναφερθούμε είναι ο browser Firefox. Μια δυναμική online εφαρμογή είναι ουσιαστικά ένα site, στο οποίο οι χρήστες έχουν πρόσβαση μέσω κάποιου browser. Είναι επομένως σημαντικό να εξασφαλίσουμε ότι ο σχεδιασμός των σελίδων της εφαρμογής έχει γίνει σωστά και ότι η απεικόνιση της πληροφορίας στο χρήστη γίνεται με τον τρόπο που επιθυμούμε. Ο Firefox είναι από τους πιο δημοφιλείς browsers και ένας από τους λόγους της εξάπλωσής του είναι η υποστήριξη επεκτάσεων (add-ons), οι οποίες προσδίδουν επιπλέον χρήσιμες λειτουργίες στο πρόγραμμα. Δύο από τις χρησιμότερες επεκτάσεις είναι το εργαλείο Firebug και η σουίτα Web Developer Toolbar. Ενδεικτικό στιγμιό-



Σχήμα 6.3: Ενδεικτική εικόνα του browser Firefox

τυπο της χρήσης του Firebug απεικονίζεται στην εικόνα 6.3. Χρησιμοποιώντας το εργαλείο αυτό, μπορούμε να εξετάσουμε τη δομή των σελίδων, τους παραγώμενους κανόνες CSS για κάποιο στοιχείο της σελίδας, τον τρόπο λειτουργίας του κώδικα JavaScript της σελίδας και να επιτελέσουμε πολλούς ακόμη ελέγχους και λειτουργίες.

Αντίστοιχα, χρησιμοποιώντας το Web Developer Toolbar, μπορούμε να ελέγξουμε άλλα στοιχεία που άπτονται του σχεδιασμού της σελίδας, όπως για παράδειγμα να πιστοποιήσουμε ότι η σελίδα τηρεί τα web standards, τόσο από την πλευρά του παραγώμενου XHTML κώδικα, όσο και από την πλευρά των κανόνων CSS που εφαρμόζονται σε αυτήν. Τέλος, μέσω του εργαλείου αυτού, μπορούμε να ελέγξουμε την όψη των σελίδων της εφαρμογής μας όταν αυτή απευθύνεται σε άλλα μέσα, όπως για παράδειγμα ένα κινητό τηλέφωνο ή ένας εκτυπωτής.

6.2 Μελλοντικές εκδόσεις και επεκτάσεις

Η εφαρμογή Soccerfun σχεδιάστηκε και υλοποιήθηκε με το σκεπτικό να είναι ευέλικτη και να μπορεί να υποστηρίξει επεκτάσεις και προσθήκες στο μέλλον. Η εκτεταμένη χρήστη της αντικειμενοστρέφειας και η συγκέντρωση των λειτουργιών των διαφόρων ενοτήτων στις αντίστοιχες κλάσεις της εφαρμογής εξασφαλίζουν ότι θα μπορούμε να επεκτείνουμε το μοντέλο της εφαρμογής για την υποστήριξη νέων λειτουργιών χωρίς προβλήματα. Ενδεικτικά θα αναφέρ-

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
title	Ο τίτλος του άρθρου
content	Το κείμενο του άρθρου
posted_at	Η ημερομηνία εισαγωγής του άρθρου
published	Αν το άρθρο είναι δημοσιευμένο ή όχι

Πίνακας 6.1: Η σχέση posts

id	Ένα μοναδικό αριθμητικό αναγνωριστικό για τη σχέση
post_id	Το άρθρο στο οποίο αφορά το σχόλιο Foreign key στη σχέση posts
user_id	Ο χρήστης που πραγματοποιεί το σχόλιο Foreign key στη σχέση users
content	Το περιεχόμενο του σχολίου

Πίνακας 6.2: Η σχέση comments

θούμε στην προσθήκη της ενότητας Community η οποία επιτρέπει στο διαχειριστή να προσθέτει άρθρα και ανακοινώσεις και στους χρήστες να συζητούν με βάση τα άρθρα αυτά μέσω του μηχανισμού των σχολίων.

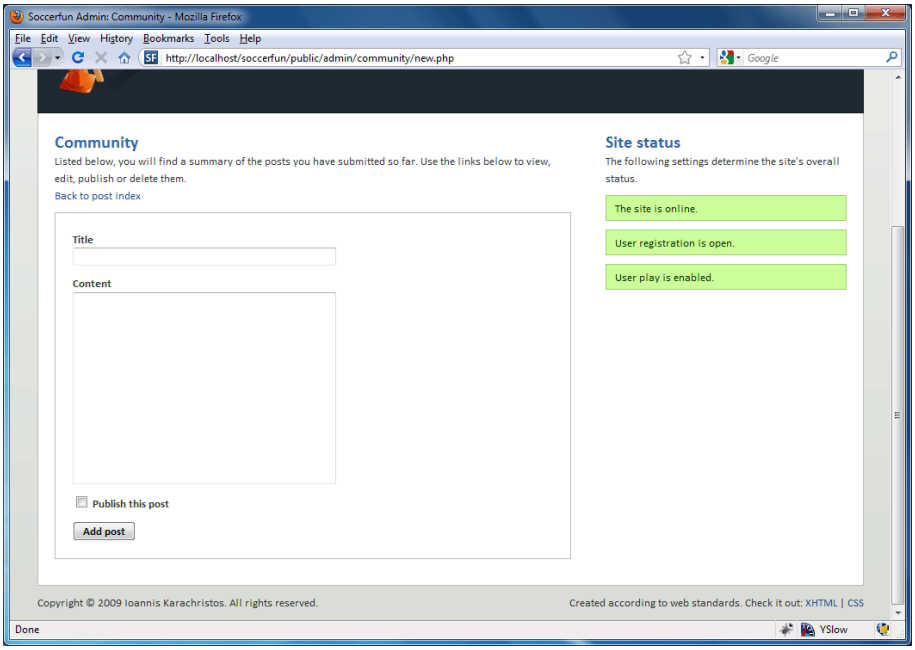
6.2.1 Η ενότητα Community

Η εφαρμογή, κατά βάση, απευθύνεται στους φίλους του ποδοσφαίρου. Η προσθήκη μιας ενότητας στην εφαρμογή, στην οποία οι χρήστες θα μπορούν να διαβάζουν άρθρα και να συζητούν γύρω από αυτά θα ήταν ένα επιπλέον στοιχείο διαδραστικότητας της εφαρμογής, αλλά επίσης θα αύξανε την αίσθηση της συμμετοχής του χρήστη στην εφαρμογή.

Το πρώτο βήμα για την υποστήριξη της ενότητας Community είναι η τροποποίηση του μοντέλου της βάσης δεδομένων. Στο παράδειγμα που εξετάζουμε, επιθυμούμε ο διαχειριστής να είναι σε θέση να δημοσιεύει άρθρα και ανακοινώσεις και οι χρήστες να μπορούν να σχολιάζουν τα άρθρα αυτά. Οι πίνακες οι οποίοι προστίθενται στην εφαρμογή για την υποστήριξη των λειτουργιών αυτών απεικονίζονται στους πίνακες 6.1 και 6.2 αντίστοιχα.

Το επόμενο βήμα της διαδικασίας είναι η δημιουργία των αντίστοιχων κλάσεων, οι οποίες θα υποστηρίζουν τις λειτουργίες της ενότητας αυτής. Ο κώδικας των κλάσεων αυτών παρατίθεται στο παράρτημα Β′ και ακολουθεί την ίδια λογική με τις υπόλοιπες κλάσεις της εφαρμογής, όπως αυτή περιγράφηκε παραπάνω.

Το τελευταίο βήμα ενσωμάτωσης της ενότητας Community είναι η δημιουργία των κατάλληλων σελίδων, οι οποίες θα αποτελούν το περιβάλλον αλληλεπίδρασης τόσο με το διαχειριστή, όσο και με το χρήστη. Ενδεικτικά, παραθέτουμε



Σχήμα 6.4: Παράδειγμα εισαγωγής δημοσίευσης

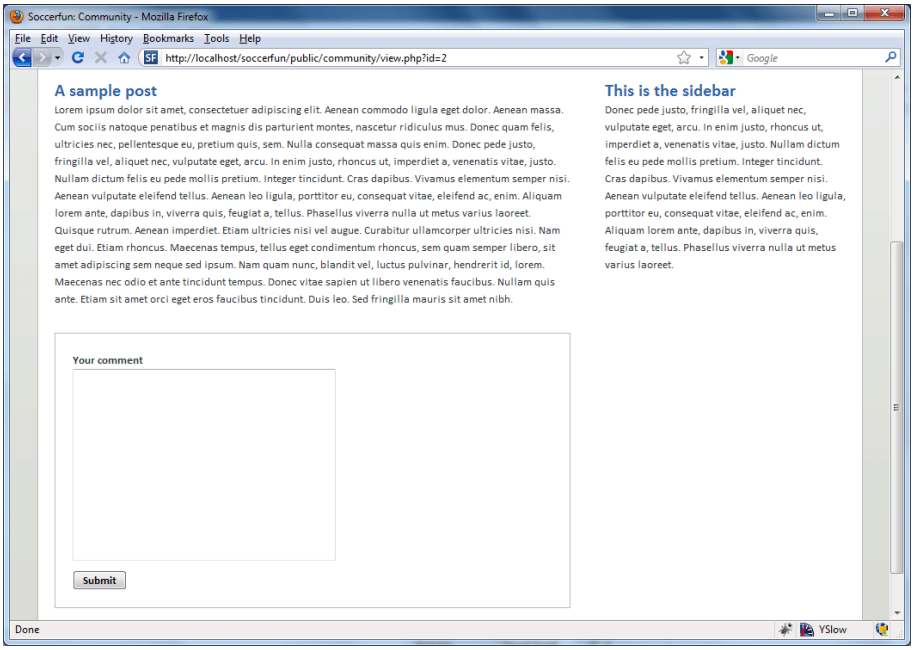
στιγμιότυπα των ενοτήτων αυτών στις εικόνες 6.4 και 6.5. Το πρώτο στιγμιότυπο αφορά στη δημιουργία μιας νέας δημοσίευσης από το διαχειριστή, ενώ το δεύτερο αφορά στην ανάγνωση ενός άρθρου από κάποιο συνδεδεμένο χρήστη με τη δυνατότητα προσθήκης ενός σχολίου.

Από την σύνοψη αυτή της διαδικασίας προσθήκης λειτουργικότητας, διαπιστώνουμε ότι η δομή της εφαρμογής, τόσο σε επίπεδο διασύνδεσης των συστατικών που την απαρτίζουν, όσο και σε επίπεδο αρχείων στο δίσκο, επιτρέπει την προσθήκη νέων στοιχείων χωρίς επίδραση στις ήδη υλοποιημένες λειτουργίες.

6.2.2 Άλλες πιθανές επεκτάσεις

Είναι προφανές ότι το παραπάνω παράδειγμα επέκτασης της εφαρμογής είναι απλά ενδεικτικό της διαδικασίας και όχι μια ολοκληρωμένη επέκταση. Έτσι, μια ενότητα που να αφορά στις γενικότερες δραστηριότητες της κοινότητας των χρηστών θα μπορούσε να περιλαμβάνει τη δημιουργία ενός φόρουμ επικοινωνίας ή ακόμη και μιας πλατφόρμας ζωντανής ανταλλαγής μηνυμάτων ανάμεσα στους χρήστες.

Μια άλλη προσθήκη στην εφαρμογή, θα μπορούσε να είναι μια γενικότερη λειτουργία καταγραφής στατιστικών γύρω από τα γεγονότα μιας διοργάνωσης με απώτερο σκοπό τη δημιουργία μιας βάσης δεδομένων με ιστορικά στοιχεία από τις διοργανώσεις των περασμένων ετών. Οι χρήστες θα μπορούν να συμβουλεύονται τα στοιχεία αυτά, ανεξάρτητα αν συμμετέχουν στο παιχνίδι ή όχι.



Σχήμα 6.5: Παράδειγμα εισαγωγής σχολίου

Τέλος, η άνοδος της πλατφόρμας των κινητών τηλεφώνων, με την κυκλοφορία του iPhone και των συσκευών βασισμένων στο λειτουργικό σύστημα Android, ήδη αλλάζει το χάρτη των χρηστών του internet. Τα μέσα αυτά προσφέρουν αυξημένες δυνατότητες στο χρήστη, σε σχέση με το πρόσφατο παρελθόν των κινητών τηλεφώνων, και σε ορισμένες περιπτώσεις, κυρίως λόγω της οθόνης αφής που διαθέτουν, προσφέρουν μια ακόμη πιο διαδραστική εμπειρία σε σχέση με την πρόσβαση μέσω του browser ενός προσωπικού υπολογιστή. Έτσι, θα μπορούσε να δημιουργηθεί μια εξειδικευμένη έκδοση της εφαρμογής, η οποία να απευθύνεται στις συσκευές αυτές και να συνεργάζεται με την υπάρχουσα πλατφόρμα.

Το internet είναι ένα μέσο, το οποίο εξελίσσεται με ραγδαίο ρυθμό. Ο οργανισμός W3C βρίσκεται σε φάση μελέτης και σύντομα θα δημοσιεύσει τις προδιαγραφές των προτύπων HTML5 και CSS3, τα οποία θα προσφέρουν ακόμη περισσότερες δυνατότητες στην υλοποίηση online εφαρμογών. Από την άλλη μεριά, τόσο η γλώσσα PHP, όσο και το σύστημα MySQL βρίσκονται σε διαρκή εξέλιξη και καθ' οδόν προς την επόμενή τους έκδοση. Το γεγονός αυτό καθιστά την υιοθέτηση των προτύπων αυτών κατά την ανάπτυξη των σύγχρονων εφαρμογών όχι μόνο χρήσιμη, αλλά και απαραίτητη για την προσαρμογή στις μελλοντικές συνθήκες. Και με αυτό ακριβώς το σκεπτικό υλοποιήθηκε η εφαρμογή Soccerfun στα πλαίσια της παρούσας διπλωματικής εργασίας.

Παράρτημα Α΄

Κώδικας MySQL

```
CREATE TABLE IF NOT EXISTS bonuses (
id int(11) NOT NULL AUTO_INCREMENT,
user_id int(11) NOT NULL,
offence_team_id int(11) NOT NULL,
defence_team_id int(11) NOT NULL,
scorer_id int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_bonuses_users (user_id),
KEY fk_bonuses_teams_off (offence_team_id),
KEY fk_bonuses_teams_def (defence_team_id),
KEY fk_bonuses_players (scorer_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS comments (
id int(11) NOT NULL AUTO_INCREMENT,
post_id int(11) NOT NULL,
user_id int(11) NOT NULL,
content text NOT NULL,
PRIMARY KEY (id),
KEY fk_comments_posts (post_id),
KEY fk_comments_users (user_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS countries (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS goals (
id int(11) NOT NULL AUTO_INCREMENT,
match_id int(11) NOT NULL,
player_id int(11) NOT NULL,
minute int(11) NOT NULL,
owngoal tinyint(4) NOT NULL DEFAULT '0',
overtime tinyint(4) NOT NULL DEFAULT '0',
PRIMARY KEY (id),
```

```
KEY fk_goals_matches (match_id),
KEY fk_goals_players (player_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS groups (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(1) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS matches (
id int(11) NOT NULL AUTO_INCREMENT,
match_day_id int(11) NOT NULL,
group_id int(11) DEFAULT NULL,
home_team_id int(11) NOT NULL,
away_team_id int(11) NOT NULL,
result_id int(11) NOT NULL DEFAULT '1',
home_goals int(11) NOT NULL DEFAULT '0',
away_goals int(11) NOT NULL DEFAULT '0',
home_penalties int(11) NOT NULL DEFAULT '0',
away_penalties int(11) NOT NULL DEFAULT '0',
PRIMARY KEY (id),
KEY fk_matches_match_days (match_day_id),
KEY fk_matches_teams_home (home_team_id),
KEY fk_matches_teams_away (away_team_id),
KEY fk_matches_results (result_id),
KEY fk_matches_groups (group_id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS match_days (
id int(11) NOT NULL AUTO_INCREMENT,
day int(11) NOT NULL,
match_date date NOT NULL,
stage_id int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_match_days_stages (stage_id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS players (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
team_id int(11) NOT NULL,
position_id int(11) NOT NULL,
goals int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_players_teams (team_id),
KEY fk_players_positions (position_id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS positions (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

```
CREATE TABLE IF NOT EXISTS posts (
id int(11) NOT NULL AUTO_INCREMENT,
title varchar(50) NOT NULL,
content text NOT NULL,
posted_at timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
published tinyint(4) NOT NULL DEFAULT '0',
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS preds (
id int(11) NOT NULL AUTO_INCREMENT,
user_id int(11) NOT NULL,
match_id int(11) NOT NULL,
result_id int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_preds_users (user_id),
KEY fk_preds_matches (match_id),
KEY fk_preds_results (result_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS races (
id int(11) NOT NULL AUTO_INCREMENT,
user_id int(11) NOT NULL,
team_id int(11) NOT NULL,
stage_id int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_races_users (user_id),
KEY fk_races_teams (team_id),
KEY fk_races_stages (stage_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS results (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(20) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS settings (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(50) NOT NULL,
status tinyint(4) NOT NULL DEFAULT '0',
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS stages (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS stocks (
id int(11) NOT NULL AUTO_INCREMENT,
```

```
user_id int(11) NOT NULL,
team_id int(11) NOT NULL,
amount int(11) NOT NULL,
PRIMARY KEY (id),
KEY fk_stocks_users (user_id),
KEY fk_stocks_teams (team_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS teams (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(45) NOT NULL,
country_id int(11) NOT NULL,
factor int(11) NOT NULL,
group_id int(11) NOT NULL,
stage_id int(11) NOT NULL DEFAULT '1',
group_points int(11) NOT NULL DEFAULT '0',
group_rank int(11) NOT NULL DEFAULT '0',
group_wins int(11) NOT NULL DEFAULT '0',
group_draws int(11) NOT NULL DEFAULT '0',
group_losses int(11) NOT NULL DEFAULT '0',
group_goals_for int(11) NOT NULL DEFAULT '0',
group_goals_against int(11) NOT NULL DEFAULT '0',
overall_wins int(11) NOT NULL DEFAULT '0',
overall_draws int(11) NOT NULL DEFAULT '0',
overall_losses int(11) NOT NULL DEFAULT '0',
overall_goals_for int(11) NOT NULL DEFAULT '0',
overall_goals_against int(11) NOT NULL DEFAULT '0',
PRIMARY KEY (id),
KEY fk_teams_countries (country_id),
KEY fk_teams_groups (group_id),
KEY fk_teams_stages (stage_id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS users (
id int(11) NOT NULL AUTO_INCREMENT,
username varchar(45) NOT NULL,
password varchar(40) NOT NULL,
email varchar(45) NOT NULL,
registered_on timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP,
activation_code varchar(45) NOT NULL,
activated tinyint(4) NOT NULL DEFAULT '0',
played_pred tinyint(4) NOT NULL DEFAULT '0',
played_race tinyint(4) NOT NULL DEFAULT '0',
played_stock tinyint(4) NOT NULL DEFAULT '0',
played_bonus tinyint(4) NOT NULL DEFAULT '0',
points_pred int(11) NOT NULL DEFAULT '0',
points_race int(11) NOT NULL DEFAULT '0',
points_stock int(11) NOT NULL DEFAULT '0',
points_bonus int(11) NOT NULL DEFAULT '0',
PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

```
ALTER TABLE bonuses
ADD CONSTRAINT fk_bonuses_players FOREIGN KEY (scorer_id)
REFERENCES players (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_bonuses_teams_def FOREIGN KEY (defence_team_id)
REFERENCES teams (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_bonuses_teams_off FOREIGN KEY (offence_team_id)
REFERENCES teams (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_bonuses_users FOREIGN KEY (user_id)
REFERENCES users (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE comments
ADD CONSTRAINT fk_comments_posts FOREIGN KEY (post_id)
REFERENCES posts (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_comments_users FOREIGN KEY (user_id)
REFERENCES users (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE goals
ADD CONSTRAINT fk_goals_matches FOREIGN KEY (match_id)
REFERENCES matches (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_goals_players FOREIGN KEY (player_id)
REFERENCES players (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE matches
ADD CONSTRAINT fk_matches_groups FOREIGN KEY (group_id)
REFERENCES groups (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_matches_match_days FOREIGN KEY (match_day_id)
REFERENCES match_days (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_matches_results FOREIGN KEY (result_id)
REFERENCES results (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_matches_teams_away FOREIGN KEY (away_team_id)
REFERENCES teams (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_matches_teams_home FOREIGN KEY (home_team_id)
REFERENCES teams (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE match_days
ADD CONSTRAINT fk_match_days_stages FOREIGN KEY (stage_id)
REFERENCES stages (id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE players
ADD CONSTRAINT fk_players_positions FOREIGN KEY (position_id)
REFERENCES positions (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT fk_players_teams FOREIGN KEY (team_id)
REFERENCES teams (id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE preds
ADD CONSTRAINT fk_preds_matches FOREIGN KEY (match_id)
REFERENCES matches (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT fk_preds_results FOREIGN KEY (result_id)
REFERENCES results (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT fk_preds_users FOREIGN KEY (user_id)
REFERENCES users (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE races
ADD CONSTRAINT fk_races_stages FOREIGN KEY (stage_id)
```

```
REFERENCES stages (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT fk_races_teams FOREIGN KEY (team_id)
REFERENCES teams (id) ON DELETE NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT fk_races_users FOREIGN KEY (user_id)
REFERENCES users (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE stocks
ADD CONSTRAINT fk_stocks_teams FOREIGN KEY (team_id)
REFERENCES teams (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_stocks_users FOREIGN KEY (user_id)
REFERENCES users (id) ON DELETE CASCADE ON UPDATE NO ACTION;

ALTER TABLE teams
ADD CONSTRAINT fk_teams_countries FOREIGN KEY (country_id)
REFERENCES countries (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_teams_groups FOREIGN KEY (group_id)
REFERENCES groups (id) ON DELETE CASCADE ON UPDATE NO ACTION,
ADD CONSTRAINT fk_teams_stages FOREIGN KEY (stage_id)
REFERENCES stages (id) ON DELETE CASCADE ON UPDATE NO ACTION;
```

Παράρτημα Β΄

Κώδικας PHP

Στο παράρτημα αυτό παραθέτουμε τον κώδικα των βασικότερων κλάσεων, καθώς επίσης κάποια απαραίτητα αρχεία που χρησιμοποιούνται από την εφαρμογή.

Το αρχείο Initialize

```
<?php
// Define the core paths
defined('DS') ? null :
  define('DS', DIRECTORY_SEPARATOR);
defined('SITE_ROOT') ? null :
  define('SITE_ROOT', 'C:'.DS.'Wamp'.DS.'www'.DS.'soccerfun');
defined('ADMIN_ROOT') ? null :
  define('ADMIN_ROOT', SITE_ROOT.DS.'public'.DS.'admin');
defined('LIB_PATH') ? null :
  define('LIB_PATH', SITE_ROOT.DS.'includes');

// Load database config file first
require_once(LIB_PATH.DS.'database.config.php');

// Load basic functions
require_once(LIB_PATH.DS.'functions.php');

// Load core objects
require_once(LIB_PATH.DS.'session.class.php');
require_once(LIB_PATH.DS.'database.class.php');

// Load project-related classes
require_once(LIB_PATH.DS.'comment.class.php');
require_once(LIB_PATH.DS.'country.class.php');
require_once(LIB_PATH.DS.'group.class.php');
require_once(LIB_PATH.DS.'match_day.class.php');
require_once(LIB_PATH.DS.'match.class.php');
require_once(LIB_PATH.DS.'player.class.php');
require_once(LIB_PATH.DS.'position.class.php');
require_once(LIB_PATH.DS.'post.class.php');
```

```
require_once(LIB_PATH.DS.'result.class.php');
require_once(LIB_PATH.DS.'setting.class.php');
require_once(LIB_PATH.DS.'stage.class.php');
require_once(LIB_PATH.DS.'team.class.php');
require_once(LIB_PATH.DS.'user.class.php');
require_once(LIB_PATH.DS.'pred.class.php');
require_once(LIB_PATH.DS.'race.class.php');
require_once(LIB_PATH.DS.'stock.class.php');
require_once(LIB_PATH.DS.'bonus.class.php');
?>
```

Το αρχείο Functions

```
<?php

function redirect_to( $location = NULL ) {
  if ($location != NULL) {
    header("Location: {$location}");
    exit;
  }
}

function output_message($message="") {
  if (!empty($message)) {
    return "<p class=\"message\">{$message}</p>";
  } else {
    return "";
  }
}

function __autoload($class_name) {
  $class_name = strtolower($class_name);
  $path = LIB_PATH.DS."{$class_name}.php";
  if(file_exists($path)) {
    require_once($path);
  } else {
    die("The file {$class_name}.php could not be found.");
  }
}

function log_action($action, $message="") {
  $logfile = SITE_ROOT.DS.'logs'.DS.'log.txt';
  $new = file_exists($logfile) ? false : true;
  if($handle = fopen($logfile, 'a')) { // append
    $timestamp = strftime("%Y-%m-%d %H:%M:%S", time());
    $content = "{$timestamp} | {$action}: {$message}\n";
    fwrite($handle, $content);
    fclose($handle);
    if($new) { chmod($logfile, 0755); }
  } else {
    echo "Could not open log file for writing.";
```

```
  }
}

function include_layout_template($template="") {
  include(SITE_ROOT.DS.'public'.DS.'layouts'.DS.$template);
}

function datetime_to_text($datetime="") {
  $unixdatetime = strtotime($datetime);
  return strftime("%B %d, %Y at %I:%M %p", $unixdatetime);
}

function generate_activation_code() {
  $chars =
      'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
  $string = '';
  for ($i = 0; $i < 40; $i++) {
      $string .= $chars[rand(0, strlen($chars) - 1)];
  }
  return $string;
}
?>
```

Η κλάση Comment

```
<?php
require_once(LIB_PATH.DS.'database.class.php');

class Comment {

  public $id;
  public $posted_at;
  public $post_id;
  public $user_id;
  public $content;

  public function find_by_post_id($id=0) {
    $sql  = "SELECT * FROM comments ";
    $sql .= "WHERE post_id={$id}";
    return self::find_by_sql($sql);
  }

  // Common Database Methods
  public static function find_all() {
    $sql = "SELECT * FROM comments";
    return self::find_by_sql($sql);
  }

  public static function find_by_id($id=0) {
    $sql  = "SELECT * FROM comments ";
    $sql .= "WHERE id={$id} LIMIT 1";
```

```
  $result_array = self::find_by_sql($sql);
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM comments";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}

public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}

public function create() {
  global $database;
  $sql  = "INSERT INTO comments (post_id, user_id, content) ";
  $sql .= "VALUES (";
  $sql .= "'".$database->escape_value($this->post_id)."', ";
  $sql .= "'".$database->escape_value($this->user_id)."', ";
  $sql .= "'".$database->escape_value($this->content)."')";
  if($database->query($sql)) {
    $this->id = $database->insert_id();
    return true;
  } else {
    return false;
```

```
    }
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM comments";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Database

```
<?php
require_once(LIB_PATH.DS."database.config.php");

class MySQLDatabase {

  private $connection;
  public $last_query;
  private $magic_quotes_active;
  private $real_escape_string_exists;

  function __construct() {
    $this->open_connection();
    $this->magic_quotes_active = get_magic_quotes_gpc();
    $this->real_escape_string_exists =
      function_exists( "mysql_real_escape_string" );
  }

  public function open_connection() {
    $this->connection = mysql_connect(DB_HOST, DB_USER, DB_PASS);
    if (!$this->connection) {
      die("Database connection failed: " . mysql_error());
    } else {
      $db_select = mysql_select_db(DB_NAME, $this->connection);
      if (!$db_select) {
        die("Database selection failed: " . mysql_error());
      }
    }
  }

  public function close_connection() {
    if(isset($this->connection)) {
      mysql_close($this->connection);
      unset($this->connection);
    }
  }
```

```
  public function query($sql) {
    $this->last_query = $sql;
    $result = mysql_query($sql, $this->connection);
    $this->confirm_query($result);
    return $result;
  }

  public function escape_value( $value ) {
    if( $this->real_escape_string_exists ) {
      if( $this->magic_quotes_active ) { $value = stripslashes(
          $value ); }
      $value = mysql_real_escape_string( $value );
    } else {
      if( !$this->magic_quotes_active ) { $value = addslashes(
          $value ); }
    }
    return $value;
  }

  public function fetch_array($result_set) {
    return mysql_fetch_array($result_set);
  }

  public function num_rows($result_set) {
   return mysql_num_rows($result_set);
  }

  public function insert_id() {
    return mysql_insert_id($this->connection);
  }

  public function affected_rows() {
    return mysql_affected_rows($this->connection);
  }

  private function confirm_query($result) {
    if (!$result) {
      $output = "Database query failed: " . mysql_error() . "<br
          /><br />";
      die( $output );
    }
  }
}

$database = new MySQLDatabase();
?>
```

Η κλάση Group

```
<?php
```

```
require_once(LIB_PATH.DS.'database.class.php');
require_once(LIB_PATH.DS.'team.class.php');

class Group {

  public $id;
  public $name;

  public function update_ranks() {
    $sql  = "SELECT * FROM teams ";
    $sql .= "WHERE group_id = ". $this->id ." ";
    $sql .= "ORDER BY group_points DESC";
    $teams = Team::find_by_sql($sql);

    for ($i=0; $i<4; $i++) {
      $team = $teams[$i];
      $team->group_rank = $i + 1;
      $team->save();
    }
  }

  // Common Database Methods
  public static function find_all() {
    $sql = "SELECT * FROM groups";
    return self::find_by_sql($sql);
  }

  public static function find_by_id($id=0) {
    $sql  = "SELECT * FROM groups ";
    $sql .= "WHERE id={$id} LIMIT 1";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_sql($sql="") {
    global $database;
    $result_set = $database->query($sql);
    $object_array = array();
    while ($row = $database->fetch_array($result_set)) {
      $object_array[] = self::instantiate($row);
    }
    return $object_array;
  }

  public static function count_all() {
    global $database;
    $sql = "SELECT COUNT(*) FROM groups";
    $result_set = $database->query($sql);
    $row = $database->fetch_array($result_set);
    return array_shift($row);
  }

  private static function instantiate($record) {
```

```
    $object = new self;
    foreach($record as $attribute=>$value){
      if($object->has_attribute($attribute)) {
        $object->$attribute = $value;
      }
    }
    return $object;
  }

  private function has_attribute($attribute) {
    $object_vars = get_object_vars($this);
    return array_key_exists($attribute, $object_vars);
  }
}
?>
```

Η κλάση Match

```
<?php
require_once(LIB_PATH.DS.'database.class.php');

class Match {

  public $id;
  public $match_day_id;
  public $group_id = 0;
  public $home_team_id;
  public $away_team_id;
  public $result_id = 1;
  public $home_goals = 0;
  public $away_goals = 0;
  public $home_penalties = 0;
  public $away_penalties = 0;

  public function get_result() {
    if ($this->result_id == 1) {
      return "Pending";
    } else {
      return $this->home_goals." - ".$this->away_goals;
    }
  }

  public static function find_by_team_id($id=0) {
    return self::find_by_sql("SELECT * FROM matches WHERE
        home_team_id={$id} OR away_team_id={$id}");
  }

  public static function find_by_home_team_id($id=0) {
    return self::find_by_sql("SELECT * FROM matches WHERE
        home_team_id={$id}");
  }
```

```
public static function find_by_away_team_id($id=0) {
  return self::find_by_sql("SELECT * FROM matches WHERE
      away_team_id={$id}");
}

public static function find_by_match_day_id($id=0) {
  return self::find_by_sql("SELECT * FROM matches WHERE
      match_day_id={$id}");
}

public static function find_by_group_id($id=0) {
  return self::find_by_sql("SELECT * FROM matches WHERE
      group_id={$id}");
}

public static function find_by_result_id($id=0) {
  return self::find_by_sql("SELECT * FROM matches WHERE
      result_id={$id}");
}

// Common Database Methods
public static function find_all() {
  return self::find_by_sql("SELECT * FROM matches ");
}

public static function find_by_id($id=0) {
  $result_array = self::find_by_sql("SELECT * FROM matches
      WHERE id={$id} LIMIT 1");
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM matches";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
```

```
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}


public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}

public function create() {
  global $database;
  $sql  = "INSERT INTO matches ";
  $sql .= "(match_day_id, group_id, home_team_id, away_team_id,
      result_id, home_goals, away_goals, home_penalties,
      away_penalties) ";
  $sql .= "VALUES (";
  $sql .= "'".$database->escape_value($this->match_day_id)."',
      ";
  $sql .= "'".$database->escape_value($this->group_id)."', ";
  $sql .= "'".$database->escape_value($this->home_team_id)."',
      ";
  $sql .= "'".$database->escape_value($this->away_team_id)."',
      ";
  $sql .= "'".$database->escape_value($this->result_id)."', ";
  $sql .= "'".$database->escape_value($this->home_goals)."', ";
  $sql .= "'".$database->escape_value($this->away_goals)."', ";
  $sql .=
      "'".$database->escape_value($this->home_penalties)."', ";
  $sql .=
      "'".$database->escape_value($this->away_penalties)."')";
  if($database->query($sql)) {
    $this->id = $database->insert_id();
    return true;
  } else {
    return false;
  }
}

public function update() {
  global $database;
  $sql  = "UPDATE matches SET ";
  $sql .= "match_day_id = '".
      $database->escape_value($this->match_day_id)."', ";
  $sql .= "home_team_id = '".
      $database->escape_value($this->home_team_id)."', ";
```

```
    $sql .= "away_team_id = '".
        $database->escape_value($this->away_team_id)."', ";
    $sql .= "result_id = '".
        $database->escape_value($this->result_id)."', ";
    $sql .= "home_goals = '".
        $database->escape_value($this->home_goals)."', ";
    $sql .= "away_goals = '".
        $database->escape_value($this->away_goals)."', ";
    $sql .= "home_penalties = '".
        $database->escape_value($this->home_penalties)."', ";
    $sql .= "away_penalties = '".
        $database->escape_value($this->away_penalties)."' ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM matches";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Player

```
<?php
require_once(LIB_PATH.DS.'database.class.php');

class Player {

  public $id;
  public $name;
  public $team_id;
  public $position_id;
  public $goals = 0;

  public function update_stats() {

  }

  public static function find_scorers() {
    return self::find_by_sql("SELECT * FROM players WHERE goals >
        0");
  }

  public static function find_by_team_id($id=0) {
```

```
  return self::find_by_sql("SELECT * FROM players WHERE
      team_id={$id}");
}

// Common Database Methods
public static function find_all() {
  return self::find_by_sql("SELECT * FROM players ORDER BY
      name");
}

public static function find_by_id($id=0) {
  $result_array = self::find_by_sql("SELECT * FROM players
      WHERE id={$id} LIMIT 1");
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM players";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}


public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}
```

```
  public function create() {
    global $database;
    $sql  = "INSERT INTO players (name, team_id, position_id,
        goals) ";
    $sql .= "VALUES (";
    $sql .= "'".$database->escape_value($this->name)."', ";
    $sql .= "'".$database->escape_value($this->team_id)."', ";
    $sql .= "'".$database->escape_value($this->position_id)."', ";
    $sql .= "'".$database->escape_value($this->goals)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $sql  = "UPDATE players SET ";
    $sql .= "name = '". $database->escape_value($this->name)."',
        ";
    $sql .= "team_id = '".
        $database->escape_value($this->team_id)."', ";
    $sql .= "position_id = '".
        $database->escape_value($this->position_id)."', ";
    $sql .= "goals = '". $database->escape_value($this->goals)."'
        ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM players";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Post

```
<?php
require_once(LIB_PATH.DS.'database.class.php');

class Post {
```

```
public $id;
public $posted_at;
public $title;
public $content;
public $published = 0;

public static function find_published() {
  $sql  = "SELECT * FROM posts ";
  $sql .= "WHERE published = 1 ";
  $sql .= "ORDER BY posted_at DESC";
  return self::find_by_sql($sql);
}

// Common Database Methods
public static function find_all() {
  $sql = "SELECT * FROM posts";
  return self::find_by_sql($sql);
}

public static function find_by_id($id=0) {
  $sql  = "SELECT * FROM posts ";
  $sql .= "WHERE id={$id} LIMIT 1";
  $result_array = self::find_by_sql($sql);
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM posts";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
```

```
  }

  private function has_attribute($attribute) {
    $object_vars = get_object_vars($this);
    return array_key_exists($attribute, $object_vars);
  }

  public function save() {
    return isset($this->id) ? $this->update() : $this->create();
  }

  public function create() {
    global $database;
    $sql  = "INSERT INTO posts (title, content, published) ";
    $sql .= "VALUES (";
    $sql .= "'".$database->escape_value($this->title)."', ";
    $sql .= "'".$database->escape_value($this->content)."', ";
    $sql .= "'".$database->escape_value($this->published)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $sql  = "UPDATE posts SET ";
    $sql .= "title = '".$database->escape_value($this->title)."',
        ";
    $sql .= "content =
        '".$database->escape_value($this->content)."', ";
    $sql .= "published =
        '".$database->escape_value($this->published)."' ";
    $sql .= "WHERE id = ".$database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM posts";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Session

```
<?php
class Session {

  private $logged_in=false;
  public $user_id;
  public $error_message;
  public $success_message;

  function __construct() {
    session_start();
    $this->check_messages();
    $this->check_login();
  }

  public function is_logged_in() {
    return $this->logged_in;
  }

  public function login($user) {
    if($user){
      $this->user_id = $_SESSION['user_id'] = $user->id;
      $this->logged_in = true;
    }
  }

  public function logout() {
    unset($_SESSION['user_id']);
    unset($this->user_id);
    $this->logged_in = false;
  }

  public function error_message($msg="") {
    if(!empty($msg)) {
      $_SESSION['error_message'] = $msg;
    } else {
      return $this->error_message;
    }
  }

  public function success_message($msg="") {
    if(!empty($msg)) {
      $_SESSION['success_message'] = $msg;
    } else {
      return $this->success_message;
    }
  }

  private function check_login() {
    if(isset($_SESSION['user_id'])) {
      $this->user_id = $_SESSION['user_id'];
      $this->logged_in = true;
```

```
    } else {
      unset($this->user_id);
      $this->logged_in = false;
    }
  }

  private function check_messages() {
    if(isset($_SESSION['error_message'])) {
      $this->error_message = $_SESSION['error_message'];
      unset($_SESSION['error_message']);
    } else {
      $this->error_message = "";
    }
    if(isset($_SESSION['success_message'])) {
      $this->success_message = $_SESSION['success_message'];
      unset($_SESSION['success_message']);
    } else {
      $this->success_message = "";
    }
  }
}

$session = new Session();
?>
```

Η κλάση Team

```
<?php
require_once(LIB_PATH.DS.'database.class.php');
require_once(LIB_PATH.DS.'match.class.php');
require_once(LIB_PATH.DS.'match_day.class.php');

class Team {

  public $id;
  public $name;
  public $country_id;
  public $factor;
  public $group_id;
  public $stage_id = 1;
  public $group_points = 0;
  public $group_rank;
  public $group_wins = 0;
  public $group_draws = 0;
  public $group_losses = 0;
  public $group_goals_for = 0;
  public $group_goals_against = 0;
  public $overall_wins = 0;
  public $overall_draws = 0;
  public $overall_losses = 0;
  public $overall_goals_for = 0;
```

```
public $overall_goals_against = 0;

public function update_stats() {
  $this->group_points = 0;
  $this->group_wins = 0;
  $this->group_draws = 0;
  $this->group_losses = 0;
  $this->group_goals_for = 0;
  $this->group_goals_against = 0;
  $this->overall_wins = 0;
  $this->overall_draws = 0;
  $this->overall_losses = 0;
  $this->overall_goals_for = 0;
  $this->overall_goals_against = 0;

  $matches = Match::find_by_home_team_id($this->id);
  foreach ($matches as $match) {
    $group_stage_match =
        (MatchDay::find_by_id($match->match_day_id)->stage_id
        == 1) ? true : false;

    $this->overall_goals_for += $match->home_goals;
    $this->overall_goals_against += $match->away_goals;
    if ($group_stage_match) {
      $this->group_goals_for += $match->home_goals;
      $this->group_goals_against += $match->away_goals;
    }

    if ($match->result_id == 2) {
      $this->overall_wins++;
      if ($group_stage_match) {$this->group_wins++;
          $this->group_points += 3;}
    } elseif ($match->result_id == 3) {
      $this->overall_draws++;
      if ($group_stage_match) {$this->group_draws++;
          $this->group_points += 1;}
    } elseif ($match->result_id == 4) {
      $this->overall_losses++;
      if ($group_stage_match) {$this->group_losses++;}
    }
  }

  $matches = Match::find_by_away_team_id($this->id);
  foreach ($matches as $match) {
    $group_stage_match =
        (MatchDay::find_by_id($match->match_day_id)->stage_id
        == 1) ? true : false;

    $this->overall_goals_for += $match->away_goals;
    $this->overall_goals_against += $match->home_goals;
    if ($group_stage_match) {
      $this->group_goals_for += $match->away_goals;
      $this->group_goals_against += $match->home_goals;
    }
```

```
    if ($match->result_id == 4) {
      $this->overall_wins++;
      if ($group_stage_match) {$this->group_wins++;
          $this->group_points += 3;}
    } elseif ($match->result_id == 3) {
      $this->overall_draws++;
      if ($group_stage_match) {$this->group_draws++;
          $this->group_points += 1;}
    } elseif ($match->result_id == 2) {
      $this->overall_losses++;
      if ($group_stage_match) {$this->group_losses++;}
    }
  }

  $this->save();
}

public static function find_by_name($name="") {
  $result_array = self::find_by_sql("SELECT * FROM teams WHERE
      name LIKE '{$name}' LIMIT 1");
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_group_id($id="") {
  return self::find_by_sql("SELECT * FROM teams WHERE
      group_id={$id} ORDER BY group_rank");
}

public static function find_by_stage_id($id="") {
  return self::find_by_sql("SELECT * FROM teams WHERE
      stage_id={$id}");
}

// Common Database Methods
public static function find_all() {
  return self::find_by_sql("SELECT * FROM teams");
}

public static function find_by_id($id=0) {
  $result_array = self::find_by_sql("SELECT * FROM teams WHERE
      id={$id} LIMIT 1");
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
```

```
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM teams";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}


public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}

public function create() {
  global $database;
  $count = sizeof(self::find_by_group_id($this->group_id));
  $this->group_rank = $count + 1;
  $sql  = "INSERT INTO teams ";
  $sql .= "(name, country_id, factor, group_id, group_rank) ";
  $sql .= "VALUES (";
  $sql .= "'".$database->escape_value($this->name)."', ";
  $sql .= "'".$database->escape_value($this->country_id)."', ";
  $sql .= "'".$database->escape_value($this->factor)."', ";
  $sql .= "'".$database->escape_value($this->group_id)."', ";
  $sql .= "'".$database->escape_value($this->group_rank)."')";
  if($database->query($sql)) {
    $this->id = $database->insert_id();
    return true;
  } else {
    return false;
  }
}

public function update() {
  global $database;
  $sql  = "UPDATE teams SET ";
```

```
    $sql .= "name = '". $database->escape_value($this->name)."',
        ";
    $sql .= "country_id = '".
        $database->escape_value($this->country_id)."', ";
    $sql .= "factor = '".
        $database->escape_value($this->factor)."', ";
    $sql .= "group_id = '".
        $database->escape_value($this->group_id)."', ";
    $sql .= "stage_id = '".
        $database->escape_value($this->stage_id)."', ";
    $sql .= "group_points = '".
        $database->escape_value($this->group_points)."', ";
    $sql .= "group_rank = '".
        $database->escape_value($this->group_rank)."', ";
    $sql .= "group_wins = '".
        $database->escape_value($this->group_wins)."', ";
    $sql .= "group_draws = '".
        $database->escape_value($this->group_draws)."', ";
    $sql .= "group_losses = '".
        $database->escape_value($this->group_losses)."', ";
    $sql .= "group_goals_for = '".
        $database->escape_value($this->group_goals_for)."', ";
    $sql .= "group_goals_against = '".
        $database->escape_value($this->group_goals_against)."', ";
    $sql .= "overall_wins = '".
        $database->escape_value($this->overall_wins)."', ";
    $sql .= "overall_draws = '".
        $database->escape_value($this->overall_draws)."', ";
    $sql .= "overall_losses = '".
        $database->escape_value($this->overall_losses)."', ";
    $sql .= "overall_goals_for = '".
        $database->escape_value($this->overall_goals_for)."', ";
    $sql .= "overall_goals_against = '".
        $database->escape_value($this->overall_goals_against)."'
        ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM teams";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση User

```
<?php
require_once(LIB_PATH.DS.'database.class.php');

class User {

  public $id;
  public $username;
  public $password;
  public $email;
  public $registered_on;
  public $activated = 0;
  public $activation_code;
  public $played_pred = 0;
  public $played_race = 0;
  public $played_stock = 0;
  public $played_bonus = 0;
  public $points_pred = 0;
  public $points_race = 0;
  public $points_stock = 0;
  public $points_bonus = 0;

  public static function authenticate($username="", $password="")
      {
    global $database;
    $username = $database->escape_value($username);
    $password = sha1($database->escape_value($password));

    $sql  = "SELECT * FROM users ";
    $sql .= "WHERE username = '{$username}' ";
    $sql .= "AND password = '{$password}' ";
    $sql .= "LIMIT 1";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_username($name="") {
    $result_array = self::find_by_sql("SELECT * FROM users WHERE
        username='{$name}' LIMIT 1");
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_email($email="") {
    $result_array = self::find_by_sql("SELECT * FROM users WHERE
        email='{$email}' LIMIT 1");
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  // Common Database Methods
```

```
public static function find_all() {
  return self::find_by_sql("SELECT * FROM users");
}

public static function find_by_id($id=0) {
  $result_array = self::find_by_sql("SELECT * FROM users WHERE
      id={$id} LIMIT 1");
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM users";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}


public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}

public function create() {
  global $database;
  $sql  = "INSERT INTO users (username, password, email,
      registered_on, activation_code) ";
  $sql .= "VALUES (";
```

```
    $sql .= "'".$database->escape_value($this->username)."', ";
    $sql .= "'".$database->escape_value($this->password)."', ";
    $sql .= "'".$database->escape_value($this->email)."', ";
    $sql .= "'".date("Y-m-d")."', ";
    $sql .=
        "'".$database->escape_value($this->activation_code)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $attributes = $this->sanitized_attributes();
    $attribute_pairs = array();
    foreach($attributes as $key => $value) {
      $attribute_pairs[] = "{$key}='{$value}'";
    }
    $sql = "UPDATE ".self::$table_name." SET ";
    $sql .= join(", ", $attribute_pairs);
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM ".self::$table_name;
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Pred

```
<?php
require_once(LIB_PATH.DS.'database.class.php');
require_once(LIB_PATH.DS.'team.class.php');

class Pred {

  public $id;
  public $user_id;
  public $match_id;
  public $result_id;
```

```
public static function find_by_user_and_match($user, $match) {
  $sql  = "SELECT * FROM preds ";
  $sql .= "WHERE user_id={$user} ";
  $sql .= "AND match_id={$match}";
  $result_array = self::find_by_sql($sql);
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

// Common Database Methods
public static function find_all() {
  $sql = "SELECT * FROM preds";
  return self::find_by_sql($sql);
}

public static function find_by_id($id=0) {
  $sql  = "SELECT * FROM preds ";
  $sql .= "WHERE id={$id} LIMIT 1";
  $result_array = self::find_by_sql($sql);
  return !empty($result_array) ? array_shift($result_array) :
      false;
}

public static function find_by_sql($sql="") {
  global $database;
  $result_set = $database->query($sql);
  $object_array = array();
  while ($row = $database->fetch_array($result_set)) {
    $object_array[] = self::instantiate($row);
  }
  return $object_array;
}

public static function count_all() {
  global $database;
  $sql = "SELECT COUNT(*) FROM preds";
  $result_set = $database->query($sql);
  $row = $database->fetch_array($result_set);
  return array_shift($row);
}

private static function instantiate($record) {
  $object = new self;
  foreach($record as $attribute=>$value){
    if($object->has_attribute($attribute)) {
      $object->$attribute = $value;
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
```

```
    return array_key_exists($attribute, $object_vars);
  }

  public function save() {
    return isset($this->id) ? $this->update() : $this->create();
  }

  public function create() {
    global $database;
    $sql  = "INSERT INTO preds (user_id, match_id, result_id) ";
    $sql .= "VALUES (";
    $sql .= "'".$database->escape_value($this->user_id)."', ";
    $sql .= "'".$database->escape_value($this->match_id)."', ";
    $sql .= "'".$database->escape_value($this->result_id)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $sql  = "UPDATE preds SET ";
    $sql .= "user_id = '".
        $database->escape_value($this->user_id)."', ";
    $sql .= "match_id = '".
        $database->escape_value($this->match_id)."', ";
    $sql .= "result_id = '".
        $database->escape_value($this->result_id)."' ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM preds";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Race

```
<?php
require_once(LIB_PATH.DS.'database.class.php');
```

```
require_once(LIB_PATH.DS.'team.class.php');

class Race {

  public $id;
  public $user_id;
  public $team_id;
  public $stage_id;

  public static function find_by_user_and_team($user, $team) {
    $sql  = "SELECT * FROM races ";
    $sql .= "WHERE user_id={$user} ";
    $sql .= "AND team_id={$team}";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  // Common Database Methods
  public static function find_all() {
    $sql = "SELECT * FROM races";
    return self::find_by_sql($sql);
  }

  public static function find_by_id($id=0) {
    $sql  = "SELECT * FROM races ";
    $sql .= "WHERE id={$id} LIMIT 1";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_sql($sql="") {
    global $database;
    $result_set = $database->query($sql);
    $object_array = array();
    while ($row = $database->fetch_array($result_set)) {
      $object_array[] = self::instantiate($row);
    }
    return $object_array;
  }

  public static function count_all() {
    global $database;
    $sql = "SELECT COUNT(*) FROM races";
    $result_set = $database->query($sql);
    $row = $database->fetch_array($result_set);
    return array_shift($row);
  }

  private static function instantiate($record) {
    $object = new self;
    foreach($record as $attribute=>$value){
      if($object->has_attribute($attribute)) {
```

```
        $object->$attribute = $value;
      }
    }
    return $object;
  }

  private function has_attribute($attribute) {
    $object_vars = get_object_vars($this);
    return array_key_exists($attribute, $object_vars);
  }

  public function save() {
    return isset($this->id) ? $this->update() : $this->create();
  }

  public function create() {
    global $database;
    $sql  = "INSERT INTO races (user_id, team_id, stage_id) ";
    $sql .= "VALUES (";
    $sql .= "'".$database->escape_value($this->user_id)."', ";
    $sql .= "'".$database->escape_value($this->team_id)."', ";
    $sql .= "'".$database->escape_value($this->stage_id)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $sql  = "UPDATE races SET ";
    $sql .= "user_id = '".
        $database->escape_value($this->user_id)."', ";
    $sql .= "team_id = '".
        $database->escape_value($this->team_id)."', ";
    $sql .= "stage_id = '".
        $database->escape_value($this->stage_id)."' ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM races";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Stock

```
<?php
require_once(LIB_PATH.DS.'database.class.php');
require_once(LIB_PATH.DS.'team.class.php');

class Stock {

  public $id;
  public $user_id;
  public $team_id;
  public $amount;

  // Common Database Methods
  public static function find_all() {
    $sql = "SELECT * FROM stocks";
    return self::find_by_sql($sql);
  }

  public static function find_by_id($id=0) {
    $sql  = "SELECT * FROM stocks ";
    $sql .= "WHERE id={$id} LIMIT 1";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_sql($sql="") {
    global $database;
    $result_set = $database->query($sql);
    $object_array = array();
    while ($row = $database->fetch_array($result_set)) {
      $object_array[] = self::instantiate($row);
    }
    return $object_array;
  }

  public static function count_all() {
    global $database;
    $sql = "SELECT COUNT(*) FROM stocks";
    $result_set = $database->query($sql);
    $row = $database->fetch_array($result_set);
    return array_shift($row);
  }

  private static function instantiate($record) {
    $object = new self;
    foreach($record as $attribute=>$value){
      if($object->has_attribute($attribute)) {
        $object->$attribute = $value;
      }
```

```
    }
    return $object;
  }

  private function has_attribute($attribute) {
    $object_vars = get_object_vars($this);
    return array_key_exists($attribute, $object_vars);
  }

  public function save() {
    return isset($this->id) ? $this->update() : $this->create();
  }

  public function create() {
    global $database;
    $sql  = "INSERT INTO stocks (user_id, team_id, amount) ";
    $sql .= "VALUES (";
    $sql .= "'".$database->escape_value($this->user_id)."', ";
    $sql .= "'".$database->escape_value($this->team_id)."', ";
    $sql .= "'".$database->escape_value($this->amount)."')";
    if($database->query($sql)) {
      $this->id = $database->insert_id();
      return true;
    } else {
      return false;
    }
  }

  public function update() {
    global $database;
    $sql  = "UPDATE stocks SET ";
    $sql .= "user_id = '".
        $database->escape_value($this->user_id)."', ";
    $sql .= "team_id = '".
        $database->escape_value($this->team_id)."', ";
    $sql .= "amount = '".
        $database->escape_value($this->amount)."' ";
    $sql .= "WHERE id=". $database->escape_value($this->id);
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }

  public function delete() {
    global $database;
    $sql = "DELETE FROM stocks";
    $sql .= " WHERE id=". $database->escape_value($this->id);
    $sql .= " LIMIT 1";
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Η κλάση Bonus

```
<?php
require_once(LIB_PATH.DS.'database.class.php');
require_once(LIB_PATH.DS.'team.class.php');

class Bonus {

  public $id;
  public $user_id;
  public $offence_team_id;
  public $defence_team_id;
  public $scorer_id;

  // Common Database Methods
  public static function find_all() {
    $sql = "SELECT * FROM bonuses";
    return self::find_by_sql($sql);
  }

  public static function find_by_id($id=0) {
    $sql  = "SELECT * FROM bonuses ";
    $sql .= "WHERE id={$id} LIMIT 1";
    $result_array = self::find_by_sql($sql);
    return !empty($result_array) ? array_shift($result_array) :
        false;
  }

  public static function find_by_sql($sql="") {
    global $database;
    $result_set = $database->query($sql);
    $object_array = array();
    while ($row = $database->fetch_array($result_set)) {
      $object_array[] = self::instantiate($row);
    }
    return $object_array;
  }

  public static function count_all() {
    global $database;
    $sql = "SELECT COUNT(*) FROM bonuses";
    $result_set = $database->query($sql);
    $row = $database->fetch_array($result_set);
    return array_shift($row);
  }

  private static function instantiate($record) {
    $object = new self;
    foreach($record as $attribute=>$value){
      if($object->has_attribute($attribute)) {
        $object->$attribute = $value;
```

```
    }
  }
  return $object;
}

private function has_attribute($attribute) {
  $object_vars = get_object_vars($this);
  return array_key_exists($attribute, $object_vars);
}

public function save() {
  return isset($this->id) ? $this->update() : $this->create();
}

public function create() {
  global $database;
  $sql  = "INSERT INTO bonuses (user_id, offence_team_id,
      defence_team_id, scorer_id) ";
  $sql .= "VALUES (";
  $sql .= "'".$database->escape_value($this->user_id)."', ";
  $sql .=
      "'".$database->escape_value($this->offence_team_id)."', ";
  $sql .=
      "'".$database->escape_value($this->defence_team_id)."', ";
  $sql .= "'".$database->escape_value($this->scorer_id)."')";
  if($database->query($sql)) {
    $this->id = $database->insert_id();
    return true;
  } else {
    return false;
  }
}

public function update() {
  global $database;
  $sql  = "UPDATE bonuses SET ";
  $sql .= "user_id = '".
      $database->escape_value($this->user_id)."', ";
  $sql .= "offence_team_id = '".
      $database->escape_value($this->offence_team_id)."', ";
  $sql .= "defence_team_id = '".
      $database->escape_value($this->defence_team_id)."', ";
  $sql .= "scorer_id = '".
      $database->escape_value($this->scorer_id)."' ";
  $sql .= "WHERE id=". $database->escape_value($this->id);
  $database->query($sql);
  return ($database->affected_rows() == 1) ? true : false;
}

public function delete() {
  global $database;
  $sql = "DELETE FROM bonuses";
  $sql .= " WHERE id=". $database->escape_value($this->id);
  $sql .= " LIMIT 1";
```

```
    $database->query($sql);
    return ($database->affected_rows() == 1) ? true : false;
  }
}
?>
```

Παράρτημα Γ΄

Κώδικας XHTML

Στο παράρτημα αυτό παραθέτουμε ενδεικτικά τον κώδικα του αρχείου *template.html*. Το αρχείο αυτό αποτελεί ένα πρότυπο πάνω στο οποίο στηρίζονται όλες οι σελίδες της εφαρμογής. Λόγω του πλήθους των σελίδων, αλλά και της επανάληψης της πληροφορίας δεν κρίνεται σκόπιμο να παραθέσουμε των κώδικα όλων των σελίδων.

Το αρχείο template.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
    lang="en">
<head>
<meta http-equiv="Content-type" content="text/html;
    charset=utf-8" />
<title>Soccerfun</title>
<link href="/soccerfun/public/styles/grid.css" media="screen"
    rel="stylesheet" type="text/css" />
<link href="/soccerfun/public/styles/main.css" media="screen"
    rel="stylesheet" type="text/css" />
<link href="/soccerfun/public/styles/mobile.css" media="handheld"
    rel="stylesheet" type="text/css" />
<link href="/soccerfun/public/styles/print.css" media="print"
    rel="stylesheet" type="text/css" />
<script type="text/javascript"
src="/soccerfun/public/scripts/jquery.min.js"></script>
<script type="text/javascript"
src="/soccerfun/public/scripts/jquery.validate.min.js"></script>
<script type="text/javascript"
src="/soccerfun/public/scripts/main.js"></script>
</head>
<body>

<div class="container_12">
```

```
<div id="header">

<div id="brand">
<h1>Soccerfun</h1>
</div>

<div id="login">
<p><a href="/soccerfun/public/index.php">Login</a></p>
</div>

<div class="clear"></div>
</div>

<div id="navigation">
<div class="grid_12">
<ul>
<li>
  <a href="/soccerfun/public/homepage/index.php">Homepage</a>
</li>
<li>
  <a href="/soccerfun/public/soccerfun/index.php">Soccerfun</a>
</li>
<li>
  <a href="/soccerfun/public/register/index.php">Register</a>
</li>
<li>
  <a href="/soccerfun/public/play/index.php">Play</a>
</li>
<li>
  <a href="/soccerfun/public/standings/index.php">Standings</a>
</li>
<li>
  <a href="/soccerfun/public/statistics/index.php">Statistics</a>
</li>
<li>
  <a href="/soccerfun/public/community/index.php">Community</a>
</li>
<li>
  <a href="/soccerfun/public/contact/index.php">Contact</a>
</li>
</ul>
</div>
<div class="clear"></div>
</div>

<div id="promo">

<div class="grid_2">
<img src="/soccerfun/public/images/stadium.png" alt="Soccer
    stadium" />
</div>

<div class="grid_9 suffix_1">
<h2>Welcome to the administration area</h2>
```

```
<p>In this section you can set up teams, players and matches. You
    can also
update the database with the latest results and manage the site's
    users. Last,
you can regulate the content displayed in the community section
    of the site.
Have fun!</p>
</div>

<div class="clear"></div>
</div>

<div id="main">

<div class="grid_8">
<div class="padded">
<h2>This is a template</h2>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
    Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et
    magnis dis
parturient montes, nascetur ridiculus mus.</p>

<h3>Sem neque sed ipsum</h3>
<p>Donec quam felis, ultricies nec,pellentesque eu, pretium quis,
    sem. Nulla
consequat massa quis enim. Donec pede justo, fringilla vel,
    aliquet nec,
vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a,
    venenatis vitae,
justo. Nullam dictum felis eu pede mollis pretium. Integer
    tincidunt. Cras
dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend
    tellus.
Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac,
    enim. Aliquam
lorem ante, dapibus in, viverra quis, feugiat a, tellus.
    Phasellus viverra nulla
ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam
    ultricies nisi
el augue. Curabitur ullamcorper ultricies nisi. Nam eget dui.
    Etiam rhoncus.
aecenas tempus, tellus eget condimentum rhoncus, sem quam semper
    libero, sit
amet adipiscing.</p>

</div>
</div>

<div class="grid_4">
<div class="padded">
<h2>Site status</h2>
<p>The following settings determine the site's overall status.</p>
</div>
```

```
</div>

<div class="clear"></div>
</div><!-- End of main content div -->

<div id="footer">

<div id="copyright">
<p>Copyright &copy; 2009 Ioannis Karachristos. All rights
    reserved.</p>
</div>

<div id="standards">
  <p>Created according to web standards. Check it out: <a
      href="#">XHTML</a> |
  <a href="#">CSS</a></p>
</div>

<div class="clear"></div>
</div>

</div>

</body>
</html>
```

Παράρτημα Δ΄

Κώδικας CSS

Στο παράρτημα αυτό παραθέτουμε τον κώδικα των CSS αρχείων που χρησιμοποιούνται για τον καθορισμό της εμφάνισης της εφαρμογής. Πιο συγκεκριμένα, περιλαμβάνουμε το αρχείο *main.css*, το οποίο καθορίζει την εμφάνιση της εφαρμογής όταν ο χρήστης επισκέπτεται το site μέσω ενός browser προσωπικού υπολογιστή, το αρχείο *print.css*, το οποίο καθορίζει την εμφάνιση μιας σελίδας όταν αυτή εκτυπώνεται από το χρήστη, και το αρχείο *mobile.css*, το οποίο καθορίζει την εμφάνιση της εφαρμογής όταν ο χρήστης επισκέπτεται το site μέσω κινητού τηλεφώνου.

Το αρχείο main.css

```
/* Reset rules */
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
  margin: 0;
  padding: 0;
  border: 0;
  outline: 0;
  font-weight: inherit;
  font-style: inherit;
  font-size: 100%;
  font-family: inherit;
  vertical-align: baseline;
}

table {
  border-collapse: separate;
  border-spacing: 0;
}
```

```
caption, th, td {
  text-align: left;
  font-weight: normal;
}

/* Basic colors and typography */
html {
  overflow-y: scroll;
}

body {
  font-family: Calibri, Helvetica, Arial, sans-serif;
  font-size: 13px;
  line-height: 20px;
  color: #2e3436;
  background: #d3d7cf url(../images/bg_body.png) top left
      repeat-x;
}

h2 {
  font-size: 20px;
  line-height: 25px;
  font-weight: bold;
}

h3 {
  margin-top: 10px;
  font-size: 13px;
  line-height: 20px;
  font-weight: bold;
}

a:link, a:visited {
  text-decoration: none;
  color: #204a87;
}

a:hover, a:active {
  text-decoration: underline;
}

strong {
  font-weight: bold;
}

/* Rules about the header */
#header {
  padding: 30px 0px 10px;
}

#brand {
  float: left;
  width: 220px;
  height: 40px;
```

```
  text-indent: -9999px;
  background: url(../images/logo.png) top left no-repeat;
}

#login {
  padding-top: 20px;
  float: right;
  text-align: right;
}

/* Rules about the navigation */
#navigation  {
  background: #3465a4 url(../images/bg_navigation.png) top left
      no-repeat;
}

#navigation ul {
  list-style: none;
}

#navigation ul li {
  float: left;
}

#navigation ul li a {
  display: block;
  width: 80px;
  padding: 10px 0px;
  font-weight: bold;
  text-align: center;
  color: #ffffff;
}

#navigation ul li a:hover {
  text-decoration: none;
  background: url(../images/bg_nav_hover.png) top left repeat-x;
}

#navigation ul li.current a {
  background: #1f2933 url(../images/bg_current.png) top left
      no-repeat;
}

/* Rules about the promo area */
#promo {
  padding: 10px 0px;
  background: #0f151a url(../images/bg_promo.png) top left
      repeat-y;
  border-bottom: 1px solid #0f151a;
}

#promo h2 {
  margin-top: 30px;
  height: 30px;
```

```
  text-indent: -9999px;
  background: url(../images/promo_title.png) top left no-repeat;
}

#promo.admin h2 {
  background: url(../images/promo_title_admin.png) top left
      no-repeat;
}

#promo.update h2 {
  background: url(../images/promo_title_update.png) top left
      no-repeat;
}

#promo p {
  color: #999999;
}

/* Rules about the main area */
#main {
  padding: 20px 0px;
  background: #ffffff url(../images/bg_main.png) top left
      repeat-y;
  border-bottom: 1px solid #babdb6;
}

#main .padded {
  padding: 0px 10px;
}

#main h2 {
  color: #3465a4;
}

/* Rules about the footer area */
#footer {
  padding: 10px 0px;
}

#copyright {
  float: left;
}

#standards {
  float: right;
  text-align: right;
}

/* Rules about forms */
form {
  margin: 10px 0px;
  padding: 20px 20px 10px;
  border: 1px solid #babdb6;
}
```

```
form.alt {
  margin: 0px;
  padding: 0px;
  border: none;
}

form p {
  margin-bottom: 10px;
}

form p span.invalid {
  padding-left: 10px;
  vertical-align: top;
  color: #cc0000;
}

label {
  font-weight: bold;
}

label span {
  padding: 0;
  font-weight: normal;
  color: #888a85;
}

input, select, textarea {
  font-family: Calibri, Helvetica, Arial, sans-serif;
  font-size: 13px;
  line-height: 20px;
}

input[type="text"], input[type="password"], textarea {
  width: 300px;
}

input[type="text"].alt {
  width: 50px;
  text-align: center;
}

input[type="submit"] {
  font-weight: bold;
}

/* Rules about messages */
div.error {
  margin-top: 10px;
  padding: 4px 10px;
  color: #330000;
  background-color: #ffcccc;
  border: 1px solid #cc9999;
}
```

```
div.success {
  margin-top: 10px;
  padding: 4px 10px;
  color: #003300;
  background-color: #ccff99;
  border: 1px solid #99cc66;
}

div.boxed {
  margin: 10px 0px;
  padding: 20px;
  border: 1px solid #babdb6;
}


/* Rules about tables */
table {
  margin: 10px 0px;
  border-collapse: collapse;
  width: 100%;
  border: 1px solid #babdb6;
}

col {
  border-right: 1px solid #babdb6;
}

col.team {
  width: 160px;
}

thead {
  background: url(../images/bg_thead.png) top left repeat-x;
  border-bottom: 1px solid #babdb6;
}

th, td {
  padding: 2px 5px;
}

th {
  font-weight: bold;
  text-align: center;
}

td {
  text-align: center;
}

th.left, td.left {
  text-align: left;
}
```

```
tr.alt {
  background-color: #eeeeec;
}

tr:hover {
  background-color: #cfdbe5;
}

span.over {
  color: #cc0000;
}
```

Το αρχείο print.css

```
* {
  margin: 0;
  padding: 0;
}

body {
  margin: 2em;
  font-family: Cambria, serif;
  font-size: 11pt;
  line-height: 18pt;
  color: #333333;
}

h1 {
  font-size: 36pt;
  line-height: 36pt;
  font-weight: normal;
}

h2 {
  font-size: 18pt;
  line-height: 24pt;
  font-weight: normal;
}

h3 {
  font-size: 10pt;
}

a:link, a:visited {
  font-style: italic;
  color: #333333;
}

/* Rules about the header */
#header {
  padding: 1em 0;
```

```
  border-bottom: 1pt solid #999999;
}

/* Rules about navigation */
#navigation .grid_12 {
  float: left;
  width: 100%;
}

#navigation ul {
  float: left;
  list-style: none;
}

#navigation ul li {
  float: left;
  margin-right: 1em;
}

#navigation ul li a {
  display: block;
  padding-top: 2pt;
  font-style: normal;
  text-decoration: none;
  text-transform: uppercase;
  color: #333333;
}

#navigation ul li.current a {
  border-bottom: 3pt solid #999999;
}


/* Rules about the login and promo area */
#login, #promo {
  display: none;
}

/* Rules about the main area */
#main {
  padding: 1em 0;
  border-top: 1pt solid #999999;
  border-bottom: 1pt solid #999999;
}

#main div.grid_8 {
  margin-bottom: 1em;
}


/* Rules about the footer area */
#footer {
  padding-top: 1em;
}
```

```
/* Rules about forms */
form {
  margin: 10pt 0pt;
  padding: 19pt 19pt 9pt;
  border: 1pt solid #999999;
}

form p {
  margin-bottom: 10pt;
}

form p span.invalid {
  padding-left: 10pt;
  vertical-align: top;
  color: #cc0000;
}

label {
  font-weight: bold;
}

label span {
  padding: 0;
  font-weight: normal;
  color: #888a85;
}

input[type="text"], input[type="password"] {
  font-size: 14pt;
  width: 300pt;
  border: 1pt solid #999999;
}

textarea {
  width: 300pt;
  border: 1pt solid #999999;
}

input[type="submit"] {
  padding: 0pt 5pt;
  font-family: Cambria, serif;
  font-size: 11pt;
  line-height: 18pt;
  color: #333333;
  background: none;
  border: 1pt solid #999999;
}

/* Rules about messages */
div.error {
  margin-top: 10px;
  padding: 4px 10px;
  color: #330000;
```

```
  background-color: #ffcccc;
  border: 1px solid #cc9999;
}

div.success {
  margin-top: 10px;
  padding: 4px 10px;
  color: #003300;
  background-color: #ccff99;
  border: 1px solid #99cc66;
}

div.boxed {
  margin: 10px 0px;
  padding: 20px;
  border: 1px solid #babdb6;
}


/* Rules about tables */
table {
  margin: 10pt 0;
  border-collapse: collapse;
  width: 100%;
  border: 1pt solid #999999;
}

col {
  border-right: 1pt solid #999999;
}

thead {
  background-color: #e8e8e8;
  border-bottom: 1pt solid #999999;
}

th, td {
  padding: 2pt 5pt;
}

th {
  font-weight: bold;
  text-align: center;
}

td {
  text-align: center;
}

th.left, td.left {
  text-align: left;
}

tr.alt {
```

```
  background-color: #f2f2f2;
}


/* Rules for special cases */
div.clear {
  clear: both;
}
```

Το αρχείο mobile.css

```
/* Reset rules */
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
  margin: 0;
  padding: 0;
  border: 0;
  outline: 0;
  font-weight: inherit;
  font-style: inherit;
  font-size: 100%;
  font-family: inherit;
  vertical-align: baseline;
}

:focus {
  outline: 0;
}

table {
  border-collapse: separate;
  border-spacing: 0;
}
caption, th, td {
  text-align: left;
  font-weight: normal;
}

/* Basic colors and typography */
body {
  font-family: sans-serif;
  font-size: small;
  color: #2e3436;
  background: #d3d7cf url(../images/bg_body.png) top left
      repeat-x;
```

```
}

p {
  line-height: 1.5em;
}

h2 {
  margin-bottom: 0.5em;
  font-size: 1.5em;
  line-height: 1em;
  font-weight: bold;
}

h3 {
  margin-top: 0.5em;
  font-size: 1em;
  font-weight: bold;
}

a:link, a:visited {
  text-decoration: none;
  color: #0066ff;
}

a:hover, a:active {
  color: #ffffff;
  background-color: #0066ff;
}

strong {
  font-weight: bold;
}

/* Rules about the header */
#brand {
  margin: 10px 0px 0px 5px;
  width: 220px;
  height: 40px;
  text-indent: -9999px;
  background: url(../images/logo.png) top left no-repeat;
}

#login {
  padding: 5px;
}

/* Rules about the navigation */
#navigation ul {
  list-style: none;
}

#navigation ul li a {
  display: block;
  padding: 10px 5px;
```

```
  font-weight: bold;
  color: #ffffff;
  background: #3465a4 url(../images/bg_navigation.png) top left
      no-repeat;
}

#navigation ul li a:hover {
  text-decoration: none;
  background: url(../images/bg_nav_hover.png) top left repeat-x;
}

#navigation ul li.current a {
  background: url(../images/bg_nav_hover.png) top left repeat-x;
}

/* Rules about the promo area */
#promo {
  display: none;
}

/* Rules about the main area */
#main {
  padding: 20px 0px;
  background: #ffffff url(../images/bg_main.png) top left
      repeat-y;
  border-bottom: 1px solid #babdb6;
}

#main .padded {
  padding: 0px 5px;
}

#main h2 {
  color: #3465a4;
}

#secondary {
  margin-top: 20px;
}

/* Rules about the footer area */
#footer {
  padding: 10px 5px;
}

/* Rules about forms */
form {
  margin: 10px 0px;
  padding: 10px 9px 0px;
  border: 1px solid #babdb6;
}

form p {
  margin-bottom: 10px;
```

```
}

form p span.invalid {
  padding-left: 10px;
  vertical-align: top;
  color: #cc0000;
}

label {
  font-weight: bold;
}

label span {
  padding: 0;
  font-weight: normal;
  color: #888a85;
}

/* Rules about messages */
div.error {
  margin-top: 10px;
  padding: 4px 9px;
  color: #330000;
  background-color: #ffcccc;
  border: 1px solid #cc9999;
}

div.success {
  margin-top: 10px;
  padding: 4px 9px;
  color: #003300;
  background-color: #ccff99;
  border: 1px solid #99cc66;
}

div.boxed {
  margin-top: 10px;
  padding: 4px 9px;
  border: 1px solid #babdb6;
}


/* Rules about tables */
table {
  margin: 10px 0px;
  border-collapse: collapse;
  width: 100%;
  border: 1px solid #babdb6;
}

col {
  border-right: 1px solid #babdb6;
}
```

```
thead {
  background: url(../images/bg_thead.png) top left repeat-x;
  border-bottom: 1px solid #babdb6;
}

th, td {
  padding: 2px 5px;
}

th {
  font-weight: bold;
  text-align: center;
}

td {
  text-align: center;
}

th.left, td.left {
  text-align: left;
}

tr.alt {
  background-color: #eeeeec;
}

tr:hover {
  background-color: #cfdbe5;
}
```

Βιβλιογραφία

[1] Abraham Silberschatz, Henry F. Korth and S. Sudarshan: *Database System Concepts*, 4th Edition, McGraw-Hill

[2] Michele E. Davis & Jon A. Phillips: *Learning PHP & Mysql*, O'Reilly

[3] Larry Ullman: *PHP and MySQL for dynamic web sites*, 2nd Edition, Peachpit Press

[4] Paul Dubois, Stefan Hinz & Carsten Pedersen: *MySQL 5.0 Certification Study Guide*, MySQL Press

[5] Luke Welling & Laura Thomson: *PHP and MySQL Web Development*, 3rd Edition, Sams Publishing

[6] George Schlossnagle: *Advanced PHP Programming*, Sams Publishing

[7] Chuck Musciano & Bill Kennedy, *HTML & XHTML The definitive guide*, 6th Edition, O'Reilly

[8] Eric Meyer, *Eric Meyer on CSS*, New Riders

[9] Dan Cederholm, *Web standards solutions*, FriendsofED

[10] Dan Cederholm, *Bulletproof web design*, New Riders

[11] Jeremy Keith, *DOM Scripting*, FriendsofED

[12] Bear Bibeault & Yehuda Katz, *JQuery in action*, Manning