ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΚΑΤΕΥΘΥΝΣΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΔΙΟΡΓΑΝΩΣΗΣ ΤΟΥΡΝΟΥΑ ΠΑΙΧΝΙΔΙΩΝ

Αντρέι–Έντουαρντ Πάβελ

Εισηγητής: Μιχαηλίδης Εμμανουήλ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΔΙΟΡΓΑΝΩΣΗΣ ΤΟΥΡΝΟΥΑ ΠΑΙΧΝΙΔΙΩΝ

Αντρέι–Έντουαρντ Πάβελ
Α.Μ. 71346786

Εισηγητής: Μιχαηλίδης Εμμανουήλ

Εξεταστική Επιτροπή:

Α/Α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ/ΤΜΗΜΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Μιχαηλίδης Εμμανουήλ	Ακαδημαϊκός Υπότροφος	Emmanouil Michailidis Digitally signed by Emmanouil Michailidis Date: 2022.10.08 00:23:47 +03'00'
2	Παναγιώτης Γιαννακόπουλος	Καθηγητής	Panagiotis Giannakopoulos Digitally signed by Panagiotis Giannakopoulos Date: 2022.10.07 23:26:03 +03'00'
3	Νικόλαος Μυριδάκης	Επίκουρος Καθηγητής	Nikolaos Myridakis Digitally signed by Nikolaos Myridakis Date: 2022.10.08 15:01:40 +03'00'

Ημερομηνία εξέτασης: 07/10/2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών
Αντρέι–Έντουαρντ Πάβελ

ΠΕΡΙΛΗΨΗ

Τα παιχνίδια σήμερα, και συγκεκριμένα τα ηλεκτρονικά παιχνίδια, αποτελούν έναν από τους πιο δημοφιλής τρόπους ψυχαγωγίας των νέων. Πολλά παιχνίδια έχουν μια ανταγωνιστική πλευρά, ή αν δεν έχουν, οι παίκτες τους συνήθως την δημιουργούν προσπαθώντας να βελτιωθούν, να γίνουν καλύτεροι από τους άλλους, να φτάσουν ψηλά και να νικήσουν. Πράγματα που βρίσκονται στη φύση του ανθρώπου. Ο σκοπός αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μιας διαδικτυακής πλατφόρμας διοργάνωσης τουρνουά, στην οποία οποιοσδήποτε μπορεί να εγγραφεί, να δημιουργήσει ένα τουρνουά ελέγχοντας όλη την πορεία του και να το κοινοποιήσει με φίλους ή άλλα άτομα. Στην εργασία, περιγράφονται τα πιο δημοφιλή είδη τουρνουά και ο τρόπος εκτέλεσης τους. Παρουσιάζονται οι τεχνολογίες ανάπτυξης λογισμικού όπως η MySQL, React και Node.js. Τέλος, αναλύονται οι αλγόριθμοι που χρησιμοποιούνται για τη λογική ενός τουρνουά, όπως η προώθηση παικτών σε επόμενους αγώνες ή η δημιουργία του bracket βάση των αριθμών των συμμετεχόντων, και πως αυτά παρουσιάζονται με ένα εύκολο σύστημα διεπαφής χρήστη.

Λέξεις-Κλειδιά: Αλγόριθμος, Τουρνουά, Double Elimination Bracket, Seeding, React, Javascript, Βάση Δεδομένων, Front-end, Back-end

ABSTRACT

Games and specifically video games have managed to become one of the main and most popular forms of entertainment for younger people. A lot of games have a competitive side to them which sometimes gets created by the players themselves as a consequence of their will to improve, to become better than the rest, to aim high and win. Things that are in human nature. The purpose of this thesis is the development of a web platform for organizing tournaments, in which anyone can sign up, create a tournament while controlling the whole process of it and sharing it to friends or other people. In this thesis, the most used types of tournaments are described and their logic explained. The fundamental technologies of the application, MySQL, React and Node.js are presented and explained. Lastly, there is an analysis of the algorithms used for the logic of a tournament, like the pushing of players to next matches or the creation of the bracket according to the number of participants, and how those are presented with an easy to use user interface.

Keywords: Algorithm, Tournament, Double Elimination Bracket, Seeding, React, Javascript, Database, Front-end, Back-end

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ vii
ABSTRACT viii
ΠΕΡΙΕΧΟΜΕΝΑ ix
Κατάλογος εικόνων xii
Κατάλογος πινάκων xv
ΚΕΦΑΛΑΙΟ 1 - ΕΙΣΑΓΩΓΗ 1
1.1 Περιγραφή 1
1.2 Δομή εργασίας 2
ΚΕΦΑΛΑΙΟ 2 - ΤΟΥΡΝΟΥΑ 3
2.1 Μορφές Τουρνουά 4
2.1.1 Seeding 4
2.1.2 Single Elimination Bracket 4
2.1.3 Double Elimination Bracket 8
2.1.4 Γκρουπ 9
2.1.5 League 10
2.2 Παρόμοια Συστήματα 10
2.2.1 Challonge 11
2.2.2 Challengermode 12
2.2.3 Matcherino 13
2.2.4 Σύγκριση 14
ΚΕΦΑΛΑΙΟ 3 - ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ 16
3.1 HTML 16
3.2 CSS 17
3.3 Javascript/Typescript 18
3.4 React 19
3.5 MySQL 21
3.6 Node.js 21
ΚΕΦΑΛΑΙΟ 4 - ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ 23
4.1 Απαιτήσεις 23
4.2 Ανάλυση Βάσης Δεδομένων 24

4.2.1 Πίνακας “users” 24
4.2.2 Πίνακας “tournaments” 25
4.2.3 Πίνακας “tournament_states” 27
4.2.4 Πίνακας “bracket_types” 28
4.2.5 Πίνακας “tournament_participants” 29
4.2.6 Πίνακας “tournament_part_states” 30
4.2.7 Πίνακας “tournament_matches” 31
4.2.8 Πίνακας “tournament_actions” 34
4.2.9 Πίνακας “actions” 35
4.2.10 Πίνακας “applog” 36
4.2.11 Διαδικασία “getProfileStats” (stored procedure) 38
4.2.12 Μοντέλο ενισχυμένης οντότητας – σχέσης (EER diagram) 39
4.3 Ανάλυση API 40
4.3.1 Δομή openbracket-api 40
4.3.2 Φάκελος models 41
4.3.3 Φάκελος graphql 43
4.3.4 Αρχείο user-login.ts 46
4.3.5 Αρχείο tournament-extra.ts 48
4.3.5.1 Συνάρτηση getTournamentRounds 49
4.3.5.2 Συνάρτηση seedTournament 50
4.3.5.3 Συνάρτηση startTournament 55
4.3.5.4 Συνάρτηση pushMatch 56
4.3.5.5 Συνάρτηση pushLosersMatch 60
4.3.5.6 Συνάρτηση undoMatch 62
4.3.5.7 Συνάρτηση undoLosersMatch 63
4.3.6 Φάκελος common 65
4.4 Ανάλυση Front-end 66
4.4.1 Κύρια αρχεία app 67
4.4.2 Δρομολόγηση σελιδών 68
4.4.3 Κλήσεις API & React Hooks 69
4.4.4 Φόρμες και πεδία εισαγωγής 72
ΚΕΦΑΛΑΙΟ 5 - ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ 75
5.1 Σύνδεση & Εγγραφή 75

5.2 Αρχική σελίδα 79
5.3 Δημιουργία και διαχείριση τουρνουά 83
5.4 Συμμετοχή τουρνουά και προφίλ χρήστη 94
ΚΕΦΑΛΑΙΟ 6 - ΕΠΙΛΟΓΟΣ 96
6.1 Σύνοψη & Συμπεράσματα 96
6.2 Μελλοντικές επεκτάσεις 98
ΒΙΒΛΙΟΓΡΑΦΙΑ 99

Κατάλογος εικόνων

Εικόνα 2.1: Single Elimination Bracket 8 συμμετεχόντων με τυχαία τοποθέτηση [2]. 5
Εικόνα 2.2: Single Elimination Bracket 8 συμμετεχόντων με seeding [3]. 6
Εικόνα 2.3: Single Elimination Bracket 6 συμμετεχόντων με seeding [4]. 7
Εικόνα 2.4: Double Elimination Bracket 8 συμμετεχόντων με seeding [5]. 8
Εικόνα 2.5: Arpad Elo, εφευρετής του συστήματος αξιολόγησης Elo [8]. 10
Εικόνα 2.6: Αρχική σελίδα διαδικτυακής πλατφόρμας Challonge [10]. 11
Εικόνα 2.7: Ολοκληρωμένη εικόνα του συστήματος Challengermode [13]. 12
Εικόνα 2.8: Αρχική σελίδα διαδικτυακής πλατφόρμας Matcherino [15]. 13
Εικόνα 3.1: Παράδειγμα κώδικα CSS. 18
Εικόνα 3.2: Παράδειγμα κώδικα Javascript 19
Εικόνα 3.3: Παράδειγμα κώδικα Typescript. 19
Εικόνα 3.4: Παράδειγμα κώδικα React. 20
Εικόνα 3.5: Παράδειγμα κώδικα MySQL. 21
Εικόνα 3.6: Παράδειγμα κώδικα Node.js. 22
Εικόνα 4.1: Διαδικασία getProfileStats. 38
Εικόνα 4.2: EER diagram συστήματος. 39
Εικόνα 4.3: Δομή αρχείων openbracket-api. 40
Εικόνα 4.4: Φάκελος models. 41
Εικόνα 4.5: Μέρος αρχείου tournaments.model.ts. 42
Εικόνα 4.6: Κώδικας πρόσθετου πεδίου host. 43
Εικόνα 4.7: Φάκελος graphql. 43
Εικόνα 4.8: Βασικοί ορισμοί αρχείου tournament-base.ts. 44
Εικόνα 4.9: Συνάρτηση userLogin. 46
Εικόνα 4.10: Συναρτήσεις αρχείου tournament-extra.ts. 48
Εικόνα 4.11: Συνάρτηση getTournamentRounds. 49
Εικόνα 4.12: Βοηθητική συνάρτηση seedBracketSize. 50
Εικόνα 4.13: Συνάρτηση seedTournament (δημιουργία Winners Bracket). 51
Εικόνα 4.14: Αλγόριθμος δημιουργίας αγώνων τουρνουά με seeding. 52
Εικόνα 4.15: Συνάρτηση seedTournament (δημιουργία Losers Bracket). 53
Εικόνα 4.16: Συνάρτηση startTourament. 55
Εικόνα 4.17: Συνάρτηση pushMatch (1ο μέρος). 56

Εικόνα 4.18: Βοηθητική συνάρτηση getLosersMatchId. 57
Εικόνα 4.19: Συνάρτηση pushMatch (2ο μέρος). 58
Εικόνα 4.20: Συνάρτηση pushLosersMatch. 60
Εικόνα 4.21: Συνάρτηση undoMatch. 62
Εικόνα 4.22: Συνάρτηση undoLosersMatch. 64
Εικόνα 4.23: Φάκελος common. 65
Εικόνα 4.24: Δομή openbracket-app. 66
Εικόνα 4.25: Αρχείο index.html. 67
Εικόνα 4.26: Αρχείο index.tsx. 67
Εικόνα 4.27: Φάκελος app. 68
Εικόνα 4.28: Αρχείο app-route.tsx. 68
Εικόνα 4.29: Script εκκίνησης Front-end. 69
Εικόνα 4.30: Σύνταξη GraphQL συναρτήσεων. 70
Εικόνα 4.31: useMutation Hook. 70
Εικόνα 4.32: Χρήση εντολής addRecord για καινούργιο τουρνουά. 71
Εικόνα 4.33: Φόρμα εισαγωγής στοιχείων καινούργιου τουρνουά. 72
Εικόνα 4.34: Φόρμα εισαγωγής στοιχείων καινούργιου τουρνουά 73
Εικόνα 5.1: Σελίδα σύνδεσης. 75
Εικόνα 5.2: Φόρμα σύνδεσης με κενά πεδία. 76
Εικόνα 5.3: Σελίδα σύνδεσης με λάθος στοιχεία σύνδεσης. 76
Εικόνα 5.4: Φόρμα σύνδεσης με εμφάνιση κωδικού. 77
Εικόνα 5.5: Σελίδα εγγραφής. 77
Εικόνα 5.6: Φόρμα εγγραφής με λάθος επιβεβαίωση κωδικού πρόσβασης. 78
Εικόνα 5.7: Σελίδα εγγραφής με σφάλμα όνομα χρήστη που υπάρχει ήδη. 79
Εικόνα 5.8: Αρχική σελίδα. 79
Εικόνα 5.9: Τουρνουά αρχικής σελίδας. 80
Εικόνα 5.10: Τουρνουά αρχικής σελίδας σε κατάσταση ολοκλήρωσης. 80
Εικόνα 5.11: Τουρνουά αρχικής σελίδας σε κατάσταση εξέλιξης. 81
Εικόνα 5.12: Φίλτρα αναζήτησης αρχικής σελίδας. 81
Εικόνα 5.13: Αναζήτηση τουρνουά με λεκτικό ”Seed” και τύπο τουρνουά Double Elimination. 82
Εικόνα 5.14: Σελίδα δημιουργίας τουρνουά. 83
Εικόνα 5.15: Σελίδα δημιουργίας τουρνουά με συμπληρωμένα στοιχεία. 83
Εικόνα 5.16: Σελίδα τουρνουά. 84
Εικόνα 5.17: Καρτέλα Settings. 85

Εικόνα 5.18: Καρτέλα Participants χωρίς συμμετέχοντες. 85
Εικόνα 5.19: Καρτέλα Participants με 8 συμμετέχοντες. 86
Εικόνα 5.20: Επιλογή seeding. 86
Εικόνα 5.21: Αλλαγή σειράς παικτών. 87
Εικόνα 5.22: Τουρνουά με συμπληρωμένο bracket. 88
Εικόνα 5.23: Σελίδα Τουρνουά σε κατάσταση εξέλιξης. 88
Εικόνα 5.24: Ενημέρωση αποτελεσμάτων αγώνα. 89
Εικόνα 5.25: Ενημερωτικό μήνυμα μη επιτρεπτού σκορ. 89
Εικόνα 5.26: Αποτέλεσμα αγώνα και προώθηση νικητή. 90
Εικόνα 5.27: Αναίρεση αγώνα. 90
Εικόνα 5.28: Το bracket μετά από αναίρεση αγώνα. 91
Εικόνα 5.29: Τελικό bracket ολοκληρωμένου τουρνουά. 92
Εικόνα 5.30: Καρτέλα Results. 92
Εικόνα 5.31: Καρτέλα Logs. 93
Εικόνα 5.32: Δήλωση συμμετοχής σε τουρνουά. 94
Εικόνα 5.33: Καρτέλα Participants ως συμμετέχοντας. 94
Εικόνα 5.34: Κουμπία προφίλ και αποσύνδεσης. 95
Εικόνα 5.35: Σελίδα προφίλ. 95

Κατάλογος πινάκων

Πίνακας 2.1: 16 συμμετέχοντες σε 4 γκρουπ με seeding. 9
Πίνακας 2.2: Λειτουργίες και διαφορές συστημάτων διοργάνωσης τουρνουά. 14
Πίνακας 3.1: Ετικέτες HTML και η λειτουργία τους. 17
Πίνακας 4.1: Ανάλυση πίνακα “users”. 24
Πίνακας 4.2: Ανάλυση πίνακα “tournaments”. 25
Πίνακας 4.3: Ανάλυση πίνακα “tournaments_states”. 27
Πίνακας 4.4: Ανάλυση πίνακα “bracket_types”. 28
Πίνακας 4.5: Ανάλυση πίνακα “tournament_participants”. 29
Πίνακας 4.6: Ανάλυση πίνακα “tournament_part_states”. 30
Πίνακας 4.7: Ανάλυση πίνακα “tournament_matches”. 31
Πίνακας 4.8: Ανάλυση πίνακα “tournament_actions”. 34
Πίνακας 4.9: Ανάλυση πίνακα “actions”. 35
Πίνακας 4.10: Ανάλυση πίνακα “applog”. 36

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Περιγραφή

Τα ηλεκτρονικά παιχνίδια σήμερα, γίνονται ολοένα και πιο δημοφιλή. Πλέον μικροί και μεγάλοι έχουν τη δυνατότητα να απολαύσουν παιχνίδια σε ηλεκτρονικούς υπολογιστές, σε κονσόλες, σε συσκευές εικονικής ή επαυξημένης πραγματικότητας και σε κινητά. Ανεξαρτήτως συσκευής πάντα θα υπάρχει ένα μέρος της κοινότητας που ψάχνει κάτι παραπάνω, ψάχνει ανταγωνισμό. Ο ανταγωνισμός αυτός συμπληρώνεται με τουρνουά, στα οποία οι παίκτες έχουν την ευκαιρία να διαγωνιστούν για φήμη ή έπαθλα.

Ο στόχος αυτής της εργασίας είναι η δημιουργία μιας διαδικτυακής πλατφόρμας διοργάνωσης τουρνουά για την ορθή καταγραφή της εξέλιξης μιας οργανωμένης εκδήλωσης. Μια εφαρμογή στην οποία κάποιος μπορεί να ψάξει τουρνουά για ένα παιχνίδι που μόλις άρχισε να παίζει και ενδιαφέρεται να δοκιμάσει τις ικανότητες του σε κάτι παραπάνω. Μια εφαρμογή που να αντικαθιστά την καταγραφή συμμετεχόντων και αγώνων σε χαρτί ή Excel και τους χειροκίνητους υπολογισμούς δεδομένων. Μια εφαρμογή που προσφέρει σε οποιονδήποτε να οργανώσει ένα τουρνουά, μικρό ή μεγάλο, με ευκολία και να το κοινοποιήσει σε ενδεχόμενους ενδιαφερόμενους.

Ο τρόπος υλοποίησης της εφαρμογής είναι μέσω τεχνολογιών που χρησιμοποιούνται στον διαδικτυακό προγραμματισμό. Χρειάζεται μια βάση δεδομένων, η οποία θα υλοποιηθεί με τη γλώσσα MySQL, για την αποθήκευση όλων των δεδομένων που θα κατέχει η εφαρμογή. Ένας ενδιάμεσος εξυπηρετητής, API, ο οποίος θα υλοποιηθεί με Node.js και θα τραβάει τα στοιχεία που χρειάζονται από τη βάση δεδομένων με συγκεκριμένες συναρτήσεις. Και τέλος, το περιεχόμενο που θα εμφανίζει ο φυλλομετρητής, η αλλιώς Front-end, το οποίο υλοποιείται με τις τεχνολογίες HTML, CSS, Javascript και React. Αυτά τα τρία επίπεδα συνδέονται μεταξύ τους για την ανάπτυξη μιας διαδικτυακής εφαρμογής που αποτελεί το τελικό προιόν της εργασίας.

1.2 Δομή εργασίας

Η παρούσα διπλωματική εργασία χωρίζεται σε έξι κεφάλαια, στα οποία αναλύεται το θεωρητικό υπόβαθρό της και στη συνέχεια η σχεδίαση και ο τρόπος ανάπτυξης της πλατφόρμας διοργάνωσης τουρνουά. Πιο αναλυτικά:

Στο κεφάλαιο αυτό, το πρώτο κεφάλαιο, περιγράφεται η βασική ιδέα της εργασίας, οι στόχοι της και η δομή της.

Το δεύτερο κεφάλαιο, αποτελεί μια εισαγωγή για τη σημασία ενός τουρνουά, τις μορφές στις οποίες μπορεί να βρεθεί και τις παρόμοιες πλατφόρμες διοργάνωσης τουρνουά που υπάρχουν σήμερα.

Στο τρίτο κεφάλαιο, το θεωρητικό υπόβαθρο, αναλύονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

Στο τέταρτο κεφάλαιο, παρουσιάζεται ο τρόπος υλοποίησης της με λεπτομερής ανάλυση σε κάθε επίπεδο της εφαρμογής.

Στο πέμπτο κεφάλαιο, γίνεται παρουσίαση της εμφάνισης της εφαρμογής ως ιστοσελίδα και καλύπτονται τα σενάρια χρήσης που προσφέρει η λειτουργικότητα της πλατφόρμας.

Στο έκτο και τελευταίο κεφάλαιο, αναφέρονται τα συμπεράσματα της εργασίας, τα προβλήματα που αντιμετωπίστηκαν κατά την ανάπτυξή της και οι μελλοντικές επεκτάσεις που μπορεί να προκύψουν.

ΚΕΦΑΛΑΙΟ 2

ΤΟΥΡΝΟΥΑ

Τουρνουά είναι μια οργανωμένη εκδήλωση, συνήθως αθλητικού τύπου, που χρησιμοποιείται για την επιλογή ενός νικητή από μια ομάδα συμμετεχόντων. Τα τουρνουά προσφέρουν μια τεχνική ζευγαρωμένων συγκρίσεων ώστε να εντοπιστεί η προτιμώμενη επιλογή. Στο πλαίσιο του αθλητισμού και των παιχνιδιών, οι συγκρίσεις αυτές ονομάζονται αγώνες και ο τρόπος ακολουθίας των συγκρίσεων αυτών βασίζεται σε διάφορα κριτήρια για να σχηματιστεί ένα φορμάτ που ονομάζεται bracket.

2.1 Μορφές Τουρνουά

2.1.1 Seeding

Υπάρχουν δύο τρόποι με τους οποίους γίνεται η τοποθέτηση των συμμετεχόντων στα τουρνουά, τυχαία ή με seeding. Με seeding είναι η προτιμώμενη μέθοδος συνήθως όπου οι συμμετέχοντες με μεγαλύτερο seeding τοποθετούνται ενάντια στους παίκτες με μικρότερο seeding στους αρχικούς αγώνες. Έτσι, οι τελευταίοι αγώνες του τουρνουά θα αποτελούνται από τους καλύτερους παίκτες. Πολλές φορές όμως, δεν υπάρχει τρόπος αναγνώρισης της ικανότητας των παικτών για να τοποθετηθούν σε μια σειρά ή δεν υπάρχει χρόνος από την πλευρά της οργάνωσης, με αποτέλεσμα η τοποθέτηση των παικτών να είναι τυχαία [1].

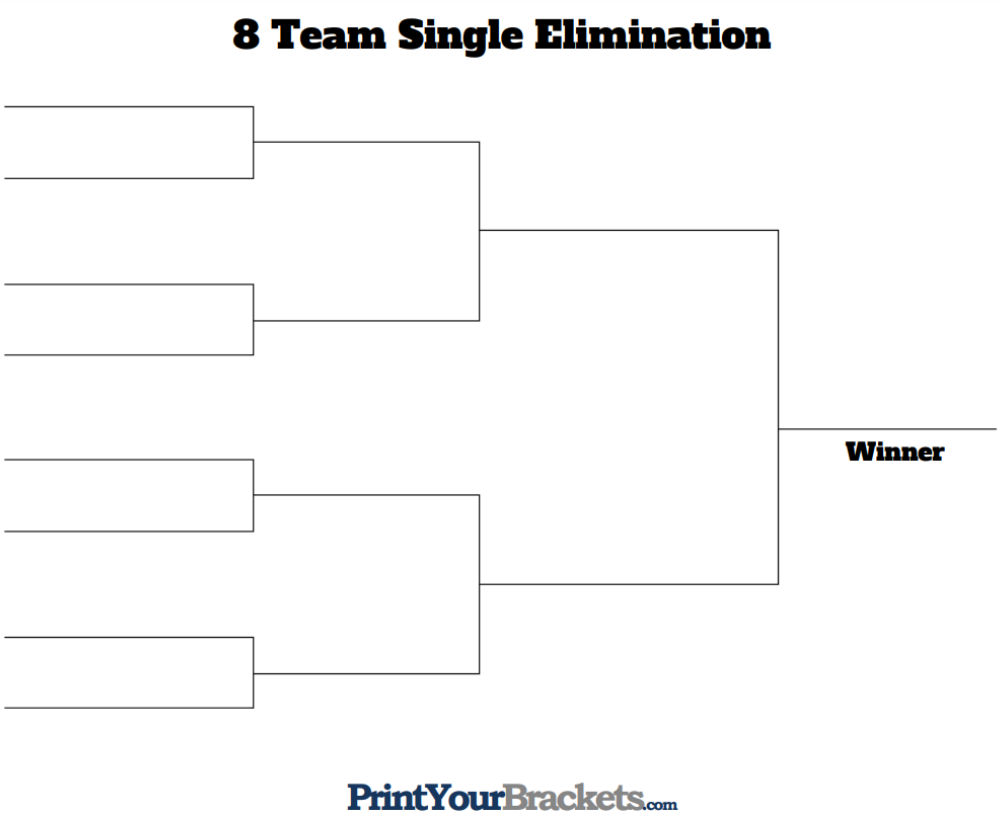
2.1.2 Single Elimination Bracket

Single Elimination είναι ο πιο απλός τρόπος οργάνωσης και συνήθως γίνεται επιλογή για μικρής διάρκειας τουρνουά. Οι παίκτες αφαιρούνται από το τουρνούα με την πρώτη ήττα και αν η τοποθέτηση τους έχει γίνει τυχαία τότε υπάρχει περίπτωση ο τελικός να μην αποτελείται από τους δύο καλύτερους παίκτες. Αυτό μπορεί να συμβεί αν ένας από τους παίκτες του τελικού έχει νικήσει από τους πρώτους γύρους τον επόμενο καλύτερο παίκτη. Για να μπορούν να έχουν όλοι οι παίκτες αντίπαλο σε όλους του γύρους, χρειάζεται ο αριθμός των συμμετεχόντων να είναι δύναμη του 2 [1].

Ο αριθμός των αγώνων όπως φαίνεται και στις παρακάτω εικόνες είναι

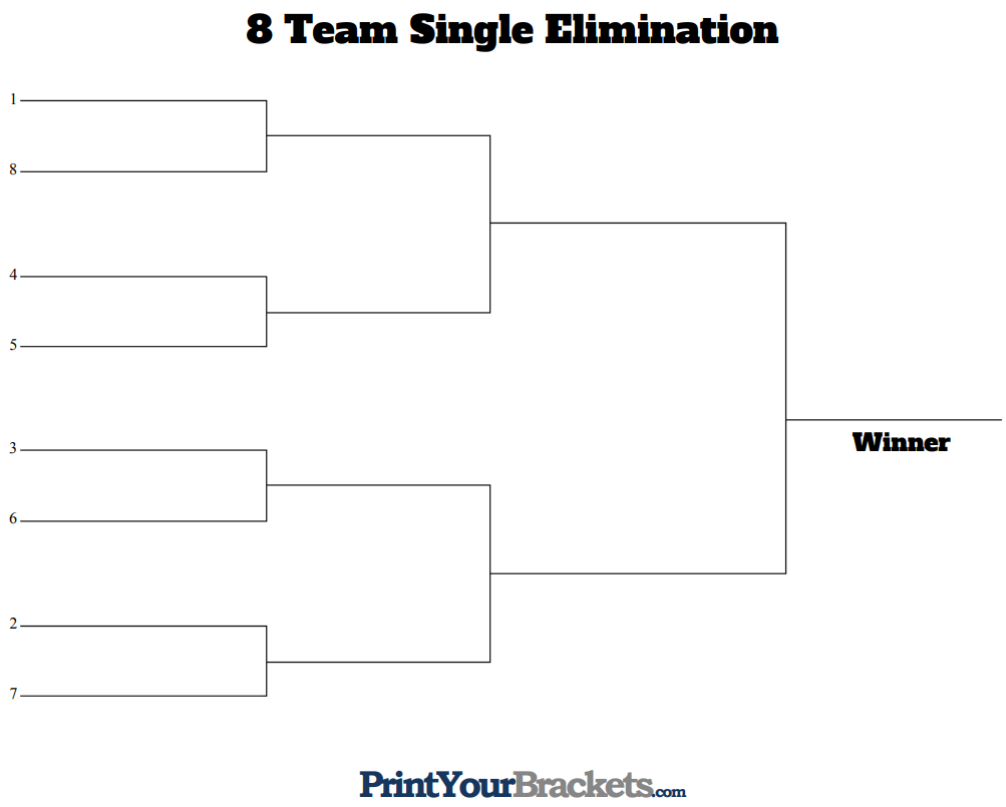
$$N - 1 \qquad (2.1)$$

όπου N ο αριθμός των συμμετεχόντων.



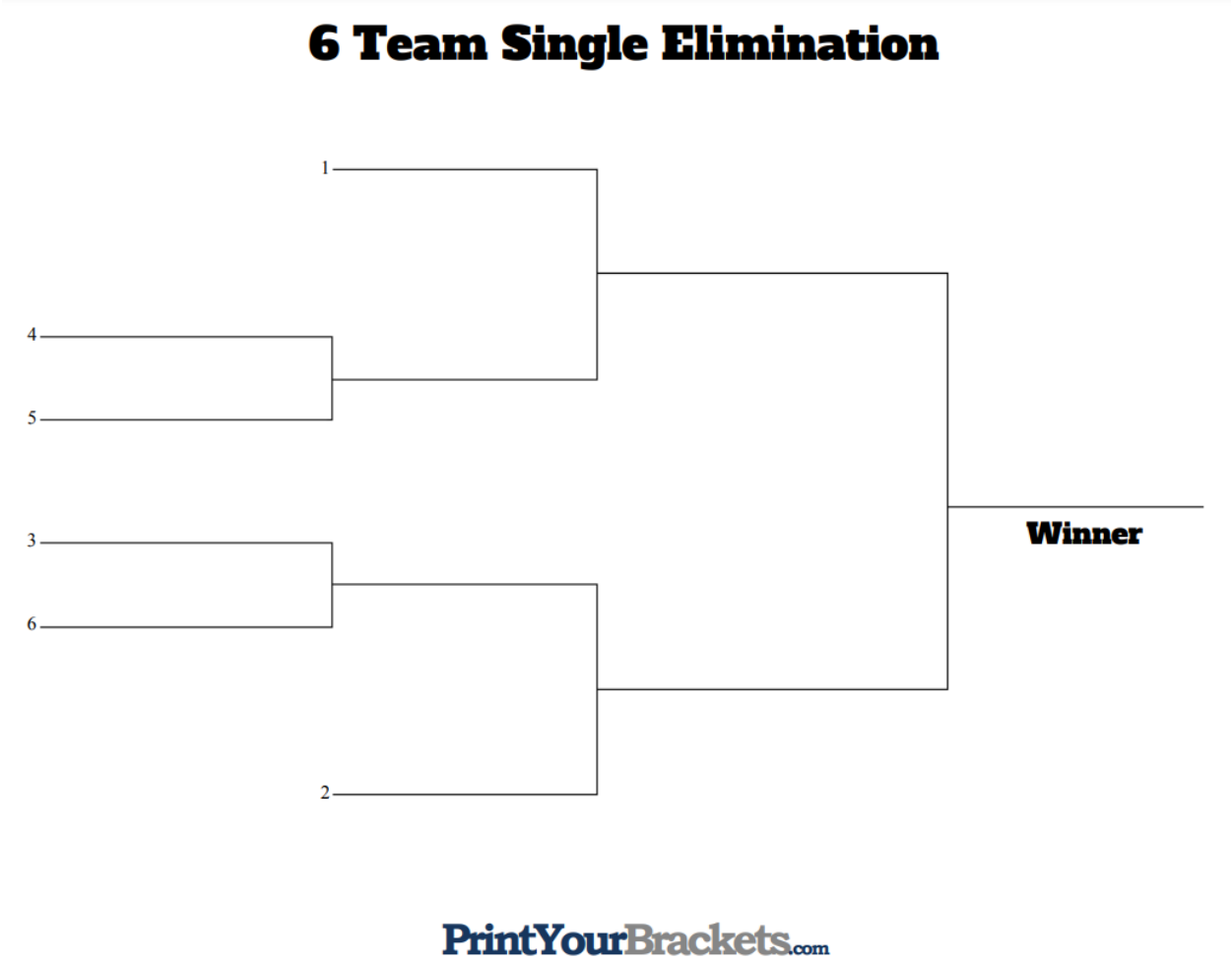
Εικόνα 2.1: Single Elimination Bracket 8 συμμετεχόντων με τυχαία τοποθέτηση [2].

Πολλές φορές γίνεται εφαρμογή και ενός επιπλέον αγώνα, λόγω βραβείων ή στατιστικών, μεταξύ των χαμένων παικτών των ημιτελικών για να καθοριστεί ο παίκτης που θα πάρει την 3η θέση.



Εικόνα 2.2: Single Elimination Bracket 8 συμμετεχόντων με seeding [3].

Βάση ταξινόμησης των παικτών, οι πρώτοι δύο παίκτες με το μεγαλύτερο seeding θα συναντηθούν στον τελευταίο γύρο.



Εικόνα 2.3: Single Elimination Bracket 6 συμμετεχόντων με seeding [4].

Όταν ο αριθμός των συμμετεχόντων δεν είναι δύναμη του 2, μερικοί παίκτες έχουν ανάγκη να κερδίσουν παραπάνω αγώνες για να φτάσουν στον τελικό του τουρνουά. Η προτεραιότητα στην επιλογή των παικτών που τοποθετούνται αρχικά πέρα από τον 1ο γύρο βασίζεται στο seeding.

2.1.3 Double Elimination Bracket

Το Double Elimination bracket είναι πολύ παρόμοιο με το Single Elimination αλλά αυτή τη φορά οι παίκτες χρειάζονται δύο ήττες για να αποκλειστούν από το τουρνουά. Το πλεονέκτημά αυτής της μεθόδου είναι ότι οι δύο καλύτεροι παίκτες έχουν την ευκαιρία να φτάσουν στον τελικό είτε συναντηθούν στους αρχικούς γύρους είτε όχι. Το bracket χωρίζεται σε δύο μέρη που συνήθως ονομάζονται Winners και Losers brackets ή Upper και Lower brackets.

Όταν ένας παίκτης πάρει την πρώτη ήττα, ανάλογα με το γύρο που βρίσκεται, θα πέσει σε αντίστοιχο γύρο στο κάτω bracket όπου θα έχει ακόμη μια ευκαιρία να συνεχίσει το τουρνουά. Όπως και στο Single Elimination bracket, είναι προτιμότερο ο αριθμός των συμμετεχόντων να είναι δύναμη του 2 [1].

Ο αριθμός των αγώνων όπως φαίνεται και στην παρακάτω εικόνα είναι

$$\big((N-1)\times 2\big)+1 \qquad (2.2)$$

όπου N ο αριθμός των συμμετεχόντων.

Εικόνα 2.4: Double Elimination Bracket 8 συμμετεχόντων με seeding [5].

Ο τελευταίος αγώνας αυτού του bracket είναι ένας ιδιαίτερος αγώνας, γιατί λαμβάνει χώρα μόνο αν ο παίκτης που προέρχεται από το Lower bracket κερδίσει τον τελικό. Αυτό συμβαίνει επειδή ο άλλος παίκτης δεν έχει καμία ήττα μέχρι εκείνη τη στιγμή, οπότε πρέπει να χάσει άλλον έναν αγώνα. Ο συγκεκριμένος αγώνας είναι προαιρετικός και δεν εφαρμόζεται πάντα επειδή μπορεί να θεωρείται ανούσιο να παίζουν οι ίδιοι παίκτες δύο φορές έναν αγώνα, μπορεί να προκαλείται κούραση μεταξύ των παικτών και των θεατών ή να επεκτείνεται πολύ η συνολική διάρκεια του τουρνουά.

2.1.4 Γκρουπ

Τα γκρουπ είναι ένα σύστημα που χρησιμοποιείται συχνά στα μεγάλα τουρνουά, χωρίζοντας τους παίκτες σε μικρότερα γκρουπ όπου αγωνίζονται μόνο με άλλους παίκτες του ίδιου γκρουπ. Οι παίκτες παίζουν τουλάχιστον μία φορά με τον καθένα και μαζεύουν πόντους με κάθε νίκη, ήττα ή ισοπαλία [1].

Για το seeding, συνήθως χρησιμοποιείται το λεγόμενο snake system όπου ο παίκτης με το μεγαλύτερο seeding τοποθετείται στο 1ο γκρουπ, ο 2ος παίκτης στο 2ο γκρουπ, ο 3ος παίκτης στο 3ο γκρουπ και ούτω καθεξής μέχρι να συμπληρωθούν όλα τα γκρουπ.
Στη συνέχεια, οι επόμενοι παίκτες τοποθετούνται στα γκρουπ αρχίζοντας από το τελευταίο γκρουπ προς το πρώτο. Το σύστημα αυτό συνεχίζει εναλλάξ μέχρι να τοποθετηθούν όλοι οι παίκτες στα γκρουπ [6].

Στον πίνακα 2.1 βλέπουμε πως 16 συμμετέχοντες τοποθετούνται σε 4 γκρουπ.

Πίνακας 2.1: 16 συμμετέχοντες σε 4 γκρουπ με seeding.

A	B	C	D
1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

Τα γκρουπ χρησιμοποιούνται ως σύστημα προκριματικών ενός μεγαλύτερου τουρνουά. Στο τέλος όλων των αγώνων, επιλέγονται οι παίκτες με τους περισσότερους πόντους από κάθε γκρουπ που θα προχωρήσουν στην επόμενη φάση. Ο αριθμός των γκρουπ, ο αριθμός των συμμετεχόντων σε κάθε γκρουπ,

ο αριθμός των αγώνων όπως και ο αριθμός των παικτών που προκρίνοται είναι όλα μεταβλητά στοιχεία που ορίζονται από τον κανονισμό του κάθε τουρνουά πριν την έναρξή του.

2.1.5 League

League, ή Λίγκα όπως εμφανίζεται αρκετές φορές, είναι το σύστημα όπου οι παίκτες τοποθετούνται σε ένα γκρουπ για να παίξουν όλοι με όλους και όπως το σύστημα των γκρουπ, επιλέγεται για μεγάλα τουρνουά και συγκεκριμένα χρονικής διάρκειας μηνών.

Τα League συνήθως χρησιμοποιούν το σύστημα αξιολόγησης Elo. Ένα σύστημα που υπολογίζει τη διαφορά επιπέδου ικανότητας μεταξύ των παικτών. Ο αριθμός Elo ενός παίκτης αλλάζει με κάθε αγώνα και προσδιορίζει την κατάταξη του παίκτη. Μετά από κάθε αγώνα, μεταφέρονται πόντοι Elo από τον ηττημένο στον νικητή αλλά ο αριθμός που μεταφέρεται δεν είναι σταθερός. Αν ένας παίκτης υψηλής κατάταξης κερδίσει ενάντιον ενός παίκτη χαμηλότερης κατάταξης, θα κερδίσει λίγους πόντους. Αντίθετα, αν κερδίσει ο παίκτης χαμηλότερης κατάταξης τότε θα κερδίσει πολλούς πόντους [7].

Όπως και τα γκρουπ, αυτό το σύστημα δουλεύει συνήθως ως προκριματικός ενός μεγαλύτερου τουρνουά.

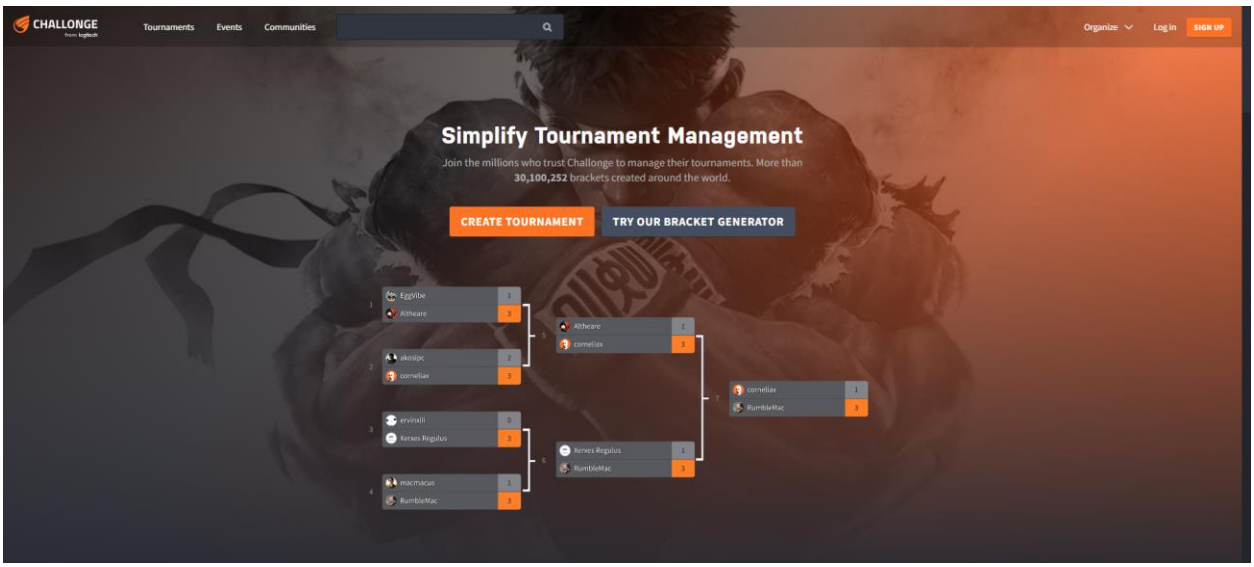
Εικόνα 2.5: Arpad Elo, εφευρετής του συστήματος αξιολόγησης Elo [8].

2.2 Παρόμοια Συστήματα

Παρακάτω θα εξεταστούν συστήματα διοργάνωσης τουρνουά που ήδη υπάχουν, οι δυνατότητές τους και οι διαφορές τους. Αν και το καθένα μπορεί να χρησιμοποιηθεί για ένα απλό τουρνουά, θα παρατηρήσουμε ότι έχουν κάποιες ιδιαίτερες και πιο προχωρημένες λειτουργίες που κάνουν την επιλογή τους πιο εύκολη για έναν διοργανωτή.

2.2.1 Challonge

Το Challonge, το οποίο ανήκει στην Logitech, ιδρύθηκε το 2009 και είναι ένα από τα πιο απλά και δημοφιλή συστήματα με πλέον πάνω από 1,500,000 μηνιαίους συμμετέχοντες και πάνω από 100,000 μηνιαία τουρνουά [9].



Εικόνα 2.6: Αρχική σελίδα διαδικτυακής πλατφόρμας Challonge [10].

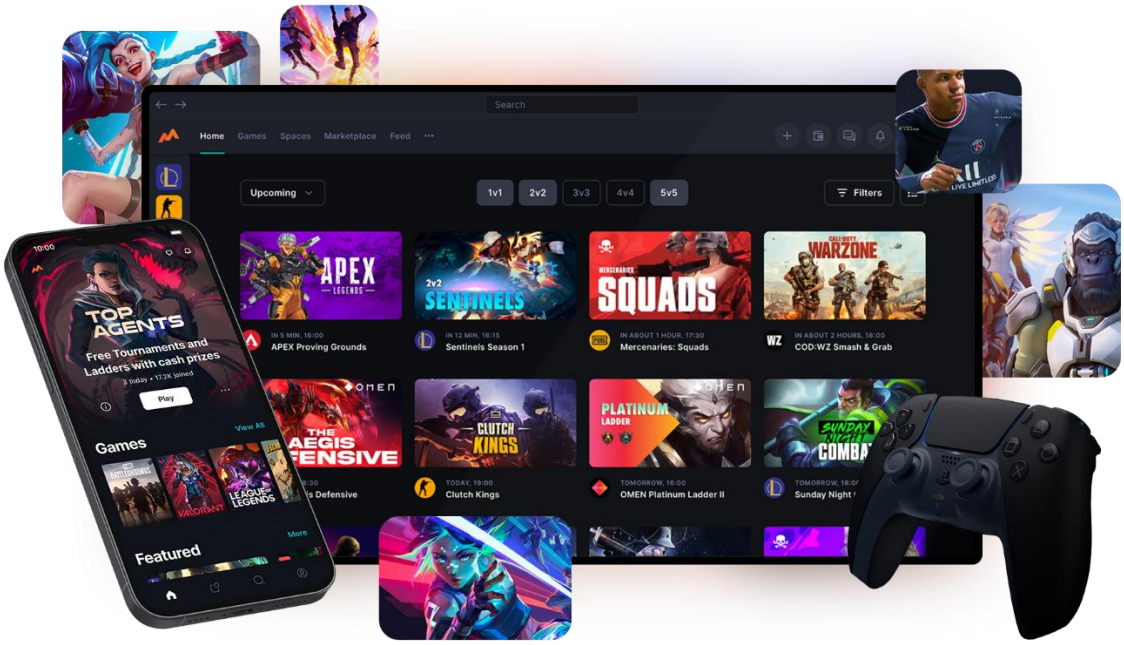
Το σύστημα αυτό είναι γνωστό για το μεγάλο εύρος τύπων τουρνουά που προσφέρει. Εκτός από τα τύποι τουρνουά που αναφέρθηκαν παραπάνω, μπορεί να τα συνδυάσει για ένα τουρνουά με πολλαπλές φάσεις. Υποστηρίζει ψήφους με τους οποίους οι θεατές μπορούν να τεστάρουν την γνώση τους για τους συμμετέχοντες διαλέγοντας τον νικητή του κάθε αγώνα και στο τέλος του τουρνουά βλέπουν ποιος μάζεψε τις περισσότερες σωστές προβλέψεις.

Ένα πολύ χρήσιμο εργαλείο του Challonge είναι το API που προσφέρει, με το οποίο κάποιος μπορεί να ρυθμίσει ένα τουρνουά μέσα από άλλες εφαρμογές σε προγραμματιστικό περιβάλλον. Μπορεί να δημιουργήσει τουρνουά, να αναφέρει σκορ, να αλλάξει την κατάσταση του τουρνουά και πολλά άλλα χωρίς τη χρήση της ιστοσελίδας.

Όλες οι παραπάνω λειτουργίες του Challonge προσφέρονται δωρεάν αλλά με μια επιπλέον ετήσια ή μηνιαία συνδρομή αφαιρούνται οι διαφημίσεις και προσθέτονται μερικές επαυξημένες ρυθμίσεις, όπως η αλλαγή του μέγιστου αριθμού συμμετεχόντων για ένα τουρνουά από 256 σε 512 ή του μέγιστου μεγέθους συνημμένων αρχείων ενός αγώνα από 250KB σε 25MB [11].

2.2.2 Challengermode

Το Challengermode ιδρύθηκε το 2014. Εκτός από διοργάνωση τουρνουά βασίζεται πολύ σε ένα κοινωνικό περιβάλλον το οποίο προσπαθεί να προσφέρει μια ολοκληρωμένη εμπειρία esports σε όλους του χρήστες της [12].



Εικόνα 2.7: Ολοκληρωμένη εικόνα του συστήματος Challengermode [13].

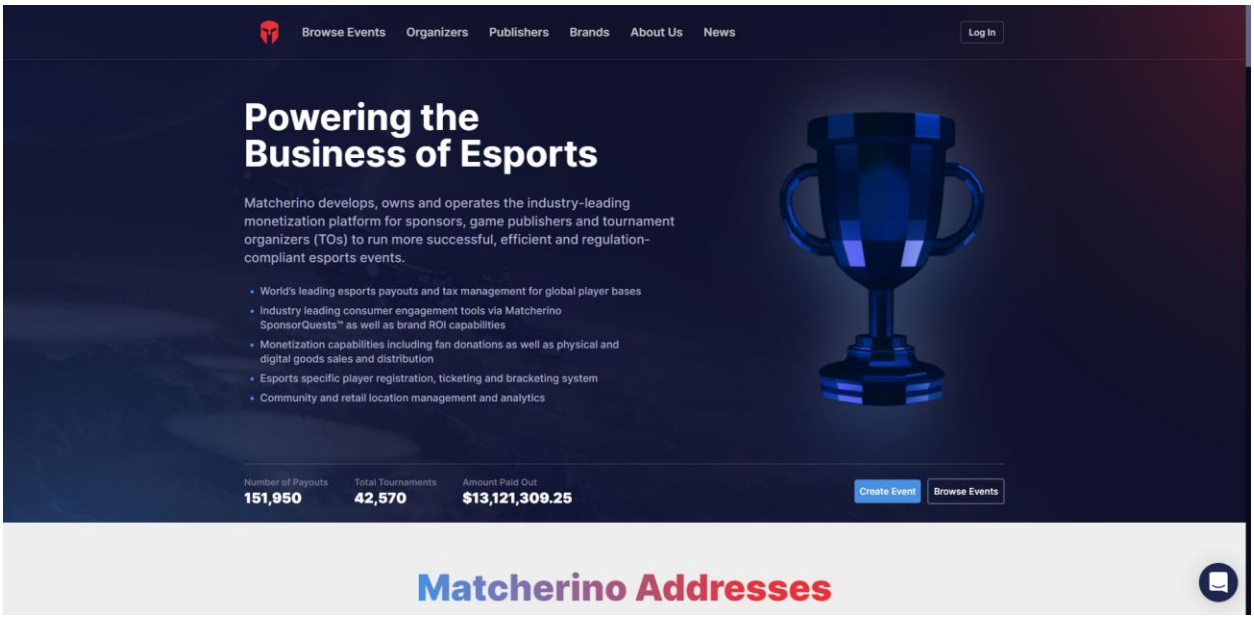
Ως τύποι τουρνουά προσφέρει Γκρουπ, Single Elimination & Double Elimination Brackets και συνδυασμό αυτών για ένα τουρνουά με έως και τρεις φάσεις.

Μια ιδιαίτερη λειτουργία που έχει είναι ο αυτοματισμός αγώνων. Μέσω της ενσωμάτωσης των παιχνιδιών με το διαδικτυακό σύστημα, απεικονίζονται πληροφορίες των αγώνων που συμβαίνουν εκείνη τη στιγμή και γίνεται αυτόματη ανανέωση των σκορ. CS:GO, LoL, Dota 2 και PUBG είναι παιχνίδια που έχουν ενσωματωθεί με το Challengermode.

Τέλος, το σύστημα διαθέτει τρόπο συνομιλίας μεταξύ των συμμετεχόντων μέσω μηνυμάτων για άμεση επικοινωνία με τους υπόλοιπους συμμετέχοντες και τους διοργανωτές, κάτι που βοηθάει στην ταυτόχρονη και πιο γρήγορη εξέλιξη των αγώνων [14].

2.2.3 Matcherino

Το Matcherino ιδρύθηκε στο Seattle το 2015 με σκοπό ένα σύστημα που μπορεί να πληρώσει τους νικητές ενός τουρνουά με έναν άμεσο, ασφαλές και αυτόματο τρόπο. Πλέον έχουν οργανωθεί πάνω από 42 χιλιάδες τουρνουά με πάνω από 13 εκατομμύρια δολάρια σε βραβεία [15].



Εικόνα 2.8: Αρχική σελίδα διαδικτυακής πλατφόρμας Matcherino [15].

Στους τύπους τουρνουά το εύρος είναι μικρό. Οι επιλογές είναι Single Elimination, Double Elimination και Swiss ενώ δεν υπάρχει επιλογή για τουρνουά πολλαπλών φάσεων.

Το Matcherino όμως εξειδικεύεται στην νομισματοποίηση του συστήματος. Οποιοδήποτε άτομο, είτε είναι θεατής, παίκτης, διοργανωτής η χορηγός μπορεί να προσθέσει εύκολα επιπλέον χρηματικό ποσό στο συνολικό έπαθλο του τουρνουά. Εκτός από την άμεση συνεισφορά, συνήθως υπάρχουν και μερικές δωρεάν αποστολές που μπορεί να ολοκληρώσει ο καθένας για να αυξηθεί κατά μικρό ποσό το έπαθλο. Αυτές είναι ενέργειες με άλλα μέσα κοινωνικής δικτύωσης και μπορεί να προσφέρουν από $0.1 έως και $1 για κάθε συνδρομή, ακουλουθία ή κοινοποίηση.

2.2.4 Σύγκριση

Ο παρακάτω πίνακας απεικονίζει τις κύριες λειτουργίες των συστημάτων που εξετάστηκαν προηγουμένως, τις διαφορές τους και τα επιπλέον στοιχεία που προσφέρουν.

Πίνακας 2.2: Λειτουργίες και διαφορές συστημάτων διοργάνωσης τουρνουά [11], [14], [15].

	Challonge	**Challengermode**	**Matcherino**
Ιστοσελίδα	NAI	NAI	NAI
Εφαρμογή Android	OXI	NAI	OXI
Εφαρμογή IOS	OXI	NAI	OXI
Single/Double Elimination	NAI	NAI	NAI
Γκρουπ	NAI	NAI	OXI
Swiss	NAI	OXI	NAI
Πολλαπλές Φάσεις	NAI	NAI	OXI
Μέγιστος Αριθμός Συμμετεχόντων	256 ή 512 με πληρωμή	64	Χωρίς Όριο
Διαγωνισμοί Φηφοφορίας Αγώνων	NAI	OXI	OXI
API	NAI	NAI	OXI
Δωρεάν	NAI	NAI	NAI
Συνδρομή για παραπάνω λειτουργίες	$12/μήνα	Δεν έχει	Δεν έχει

Χρειάζεται λογαριασμό	ΝΑΙ	ΝΑΙ	ΝΑΙ
Διαφημίσεις	ΝΑΙ (ΟΧΙ με συνδρομή)	ΟΧΙ	ΟΧΙ
Σύστημα Μηνυμάτων	ΟΧΙ	ΝΑΙ	ΟΧΙ
Σύστημα Χρηματικής Συνεισφοράς Κοινού	ΟΧΙ	ΟΧΙ	ΝΑΙ

ΚΕΦΑΛΑΙΟ 3

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο κεφάλαιο αυτό, καταγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Στο περιβάλλον του προγραμματισμού διαδικτύου αυτές οι τεχνολογίες είναι πάρα πολλές αν αναλυθούν σε ένα λεπτομερή επίπεδο, καθώς μια τεχνολογία μπορεί να χρησιμοποιείται για μερικές γραμμές κώδικα και να μην ξαναεμφανίζεται σε άλλα σημεία. Για τον λόγο αυτό, παρακάτω θα αναλυθούν αυτές που εμφανίζονται γενικά και αποτελούν τον σκελετό του συστήματος.

3.1 HTML

Η HTML (HyperText Markup Language) είναι η κύρια γλώσσα σήμανσης για το διαδικτυακό προγραμματισμό με πάνω από 90% των ιστοσελίδων να την χρησιμοποιούν [16]. Οι περιηγητές ιστού αντλούν τα αρχεία HTML από έναν τοπικό υπολογιστή ή έναν εξυπηρετητή διαδικτύου για να εμφανίσουν τα δεδομένα σε ιστοσελίδες. Τα αρχεία HTML αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα στα σύμβολα “< >” (μικρότερο από και μεγαλύτερο από). Οι ετικέτες αυτές λειτουργούν συνήθως ανά ζεύγη με τη μορφή < > και < / >, για παράδειγμα και . Η πρώτη ετικέτα < > χρησιμοποείται στην αρχή του HTML στοιχείου, η δεύτερη ετικέτα < / > στο τέλος του, και ανάμεσα τους τοποθετείται ένα κείμενο το οποίο εμφανίζεται τροποποιημένο στην ιστοσελίδα ανάλογα με ποια ετικέτα χρησιμοποιήθηκε. Μερικές ετικέτες που χρησιμοποιούνται για την εμφάνιση εικόνων και άλλων αντικειμένων δεν χρειάζονται κείμενο ανάμεσα στις ετικέτες έναρξης και λήξης. Οι περιηγητές ιστού δεν εμφανίζουν τις ετικέτες HTML αλλά τις χρησιμοποιούν για να εμφανίσουν το περιεχόμενο της ιστοσελίδας με τη μορφή που ορίζει ο προγραμματιστής. Το περιεχόμενο αυτό μπορει να αποτελείται από εικόνες, λίστες, παραγράφους, πίνακες και διαδραστικές φόρμες που περιέχουν κουμπιά και πλαίσια εισαγωγής κειμένου. Η HTML στις περισσότερες ιστοσελίδες δεν χρησιμοποιείται μόνη της αλλά με την βοήθεια των CSS και Javascript. Η CSS βοηθάει στην μορφοποίηση της εμφάνισης των ιστοσελίδων, ενώ η Javascript στην ενσωμάτωση ενεργειών και εντολών που αλλάζουν τη συμπεριφορά τους. Στον παρακάτω πίνακα παρουσιάζονται μερικές από τις ετικέτες που διαθέτει η γλώσσα HTML και η λειτουργία τους [17].

Πίνακας 3.1: Ετικέτες HTML και η λειτουργία τους.

Ετικέτες HTML	Περιγραφή
<!--...-->	Ορίζει ένα σχόλιο
<a> 	Ορίζει έναν υπερσύνδεσμο
	Ορίζει έντονο κείμενο (bold)
<body> </body>	Ορίζει το περιεχόμενο μιας σελίδας
 	Ορίζει διακοπή γραμμής κειμένου
<button> </button>	Ορίζει ένα κουμπί
	Ορίζει μια ομαδοποίηση στοιχείων
<form> </form>	Ορίζει μια διαδραστική φόρμα
<hn> </hn> όπου n=1 έως 6	Ορίζει μια επικεφαλίδα
	Ορίζει κείμενο σε πλάγια γραφή (italics)
<img> </img>	Ορίζει μια εικόνα
	Ορίζει μια παράγραφο
	Ορίζει ένα κείμενο μικρότερου μεγέθους
<table> </table>	Ορίζει έναν πίνακα
<td> </td>	Ορίζει ένα κελί ενός πίνακα
<title> </title>	Ορίζει τον τίτλο μιας σελίδας
<tr> </tr>	Ορίζει μια γραμμή ενός πίνακα
<u> </u>	Ορίζει ένα υπογραμμισμένο κείμενο

3.2 CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα εμφάνισης που χρησιμοποιείται για την παρουσία ενός εγγράφου γραμμένου σε μια γλώσσα σήμανσης, όπως η HTML και η XML. Αποτελεί βασική τεχνολογία του παγκόσμιου ιστού, μαζί με την HTML και την Javascript. Η CSS έχει σχεδιαστεί για να διαμορφώνει την εμφάνιση του περιεχομένου μιας ιστοσελίδας, ορίζοντας τη στοίχιση, τα χρώματα και τις γραμματοσειρές. Έχοντας αυτόν τον έλεγχο των χαρακτηριστικών εμφάνισης, επιτρέπει σε πολλαπλές σελίδες να χρησιμοποιούν μια κοινή μορφοποίηση μέσω ενός αρχείου .css με αποτέλεσμα να μειώνει την

πολυπλοκότητα και την επανάληψη του κώδικα. Αν μια ιστοσελίδα είναι προσβάσιμη σε άλλες συσκευές όπως κινητά τηλέφωνα ή τάμπλετ, η CSS έχει την ικανότητα να ορίσει κανόνες για την μορφοποίηση του περιεχομένου ώστε να φαίνεται ικανοποιητικά στην ανάλυση της συσκευής. Ο όρος cascading προέρχεται από τον τρόπο δήλωσης προτεραιότητας που διαθέτει η CSS για όταν δύο ή παραπάνω κανόνες εφαρμόζονται στο ίδιο στοιχείο της ιστοσελίδας. Η σύνταξη της CSS είναι σχετικά απλή, χρησιμοποιώντας αγγλικές λέξεις κλειδιά ως ορίσματα ή εξατομικευμένα ονόματα που ορίζει ο προγραμματιστής. Τα ορίσματα ονομάζονται selectors και μπορούν να εφαρμοστούν κατευθείαν σε στοιχεία της HTML όπως h1, p, body ή σε στοιχεία με συγκεκριμένο χαρακτηριστικό που ονομάζεται class. Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα με κώδικα CSS [17].

```css
h2 {
  color: black;
}

button {
  margin: 5px;
}

.big-title {
  font-size: 30px
}
```

Εικόνα 3.1: Παράδειγμα κώδικα CSS.

3.3 Javascript/Typescript

Η Javascript, που συνήθως συντομογραφείται ως JS, είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου που αποτελεί μία από τις βασικές τεχνολογίες του World Wide Web μαζί με την CSS και την HTML. Αυτήν τη στιγμή, το 2022, το 98% των ιστοσελίδων χρησιμοποιούν Javascript για την πλευρά του πελάτη (client side) και οι κύριοι περιηγητές ιστού έχουν αποκλειστική μηχανή Javascript για να εκτελεστεί ο κώδικάς της στις συσκευές των χρηστών. Είναι μια γλώσσα με πολλαπλές μεθόδους προγραμματισμού η οποία υποστηρίζει χειρισμό γεγονότων (event-driven), λειτουργικό προγραμματισμό (functional), προστακτικό προγραμματισμό (imperative) και αντικειμενοστρεφή προγραμματισμό (object-oriented). Μπορεί να δημιουργήσει APIs (Application Programming Interfaces) για μεταφορά δεδομένων και να

χρησιμοποιήσει κανονικές εκφράσεις (regex) αλλά δεν έχει δυνατότητες είσοδου-εξόδου (I/O) για γραφικά, δίκτυα και αποθηκευτικούς χώρους. Η Javascript εφαρμόστηκε αρχικά μόνο στην πλευρά του πελάτη των ιστοσελίδων αλλά πλέον χρησιμοποιείται και στο server-side με μία από τις πιο δημοφιλές μηχανές, την Node.js, αλλά και σε επιπλέον περιβάλλοντα εκτέλεσης χωρίς δίκτυο όπως οι εφαρμογές κινητών συσκευών [17].

Η Typescript είναι και αυτή μια γλώσσα προγραμματισμού που συμπεριλαμβάνει την Javascript αλλά προσθέτει έναν προαιρετικό έλεγχο για τα τύποι δεδομένων ο οποίος βοηθάει στον άμεσο εντοπισμό σφαλμάτων. Ο κώδικάς της μπορεί να μετατραπεί σε Javascript και να τρέξει οπουδήποτε τρέχει και η Javascript, σε client-side ή server-side. Βάση αυτού, οποιοδήποτε πρόγραμμα γραμμένο σε Javascript δουλεύει αυτόματα και σε Typescript [18].

```
let a;
a = "Hello";
console.log(a) // Output: Hello
a = 123;
console.log(a) // Output: 123
```

Εικόνα 3.2: Παράδειγμα κώδικα Javascript

```
let a: String;
a = "Hello";
console.log(a) // Output: Hello
a = 123;
// Error: Type number is not assignable to type String
```

Εικόνα 3.3: Παράδειγμα κώδικα Typescript.

3.4 React

Η React (React.js ή ReactJS) είναι μια δωρεάν βιβλιοθήκη Javascript ανοικτού κώδικα για τη δημιουργία διεπαφών χρήστη. Η βιβλιοθήκη διατηρείται από την εταιρεία Meta (προηγουμένως

γνωστή ως Facebook) και από άτομα της κοινότητας αυτής. Χρησιμοποιείται για την ανάπτυξη ιστοσελίδων αλλά και για εφαρμογές κινητού. Η React ως βάση αφορά μόνο τη διαχείριση της κατάστασης των στοιχείων και την απεικόνηση των καταστάσεων στο DOM (Document Object Model), με αποτέλεσμα να χρειάζονται παραπάνω βιβλιοθήκες για μια ολοκληρωμένη εφαρμοφή. Οι καταστάσεις χρησιμοποιούνται για μια αποτελεσματική ενημέρωση της εμφάνισης της σελίδας, επιλέγοντας τα σωστά στοιχεία μετά από μια αλλαγή στα δεδομένα της εφαρμογής. Με αυτόν τον τρόπο, ο κώδικας είναι πιο προβλέψιμος και εύκολος στον εντοπισμό σφαλμάτων. Τα στοιχεία λέγονται components, έχουν δικιά τους κατάσταση και πολλές φορές περιλαμβάνουν ένα η παραπάνω components στην έξοδο τους. Για να υλοποιηθεί μια εφαρμογή React, συνήθως ετοιμάζονται components που αντιστοιχούν σε κάποιο στοιχείο διεπαφής όπως ένα κουμπί, μια φόρμα ή ένα κείμενο και οργανώνονται σε components υψηλότερου επιπέδου για τον καθορισμό της δομής της εφαμοργής. Μία από τις κύριες διαφορές της React με άλλες Javascript βιβλιοθήκες είναι ότι δεν χρησιμοποιεί άμεσα το DOM του προγραμμάτος περιήγησης, κάτι που μπορεί να επηρεάσει την ταχύτητα του προγράμματος όταν υπάρχουν πολλές αλλαγές στα δεδομένα, αλλά ένα εικονικό DOM που δημιουργείται στη μνήμη. Οι αλλαγές του προγράμματος επιλύονται στο εικονικό DOM και η React καθορίζει έξυπνα και γρήγορα τις διαφορές με το πραγματικό DOM για να τις ενημερώσει τελικά σε αυτό [17], [19].

```
class HelloWho extends React.Component {
  render() {
    return (
    <div>
    Hello {this.props.who}, my name is {this.props.name}!
    </div>
    )
  }
}

root.render(<HelloWho who="UniWa" name="Andrei" />);

//Hello UniWa, my name is Andrei!
```

Εικόνα 3.4: Παράδειγμα κώδικα React.

3.5 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational database management system ή RDBMS) το οποίο κυκλοφόρησε ως πρώτη έκδοση το 1995 και τώρα ανήκει στην Oracle. Μια σχεσιακή βάση δεδομένων οργανώνει τα δεδομένα σε έναν ή περισσότερους πίνακες όπου τα δεδομένα συνήθως συσχετίζονται μεταξύ τους. Οι προγραμματιστές μπορούν να δημιουργήσουν, τροποποιήσουν ή εξάγουν δεδομένα από τη βάση αλλά και να ρυθμίσουν τα δικαιώματα των χρηστών που έχουν πρόσβαση στη βάση. Μια RDBMS όπως η MySQL, δουλεύει με το λειτουργικό σύστημα για τις παραπάνω λειτουργίες αλλά και για να επιτρέπει πρόσβαση στο διαδίκτυο και τη δυνατότητα δημιουργίας αντιγράφου ασφαλείας. Η MySQL είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα και χρησιμοποιείται από πολλές διαδικτυακές εφαρμογές που βασίζονται σε βάσεις δεδομένων όπως Drupal, Joomla, phpBB, WordPress αλλά και σε δημοφιλές ιστοσελίδες όπως Facebook, PayPal, Twitter και YouTube [20].

```
1   CREATE DATABASE `openbracket`;
2
3   CREATE TABLE `users` (
4     `id` int(11) NOT NULL AUTO_INCREMENT,
5     `userName` varchar(255) DEFAULT NULL,
6     `displayName` varchar(255)  DEFAULT NULL,
7     `userPassword` varchar(255) DEFAULT NULL,
8     `email` varchar(255) DEFAULT NULL,
9     PRIMARY KEY (`id`)
10  )
11
12    SELECT * FROM openbracket.users;
13
```

Εικόνα 3.5: Παράδειγμα κώδικα MySQL.

3.6 Node.js

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού ανοιχτού κώδικα, χτισμένη σε περιβάλλον Javascript που εκτελεί κώδικα έξω από ένα πρόγραμμα περιήγησης. Επιτρέπει τους προγραμματιστές να χρησιμοποιήσουν τη γλώσσα Javascript για να γράψουν εργαλεία γραμμής εντολών και scripts, από την

πλευρά του διακομιστή (server-side), για την παραγωγή δυναμικού περιεχομένου που τελικά στέλνεται στην ιστοσελίδα του προγράμματος περιήγησης των χρηστών. Το Node.js αντιπροσωπεύει τη φράση "Javascript everywhere", δηλαδή τη χρήση της γλώσσας Javascript τόσο στη πλευρά του διακομιστή όσο και στη πλευρά του πελάτη, ώστε η ανάπτυξη της web εφαρμογής να υλοποιείται μόνο μέσω μίας γλώσσας προγραμματισμού. Η αρχιτεκτονική του Node.js βασίζεται σε γεγονότα και είναι ικανή για ασύγχρονη είσοδο/έξοδο (I/O). Αυτό στοχεύει στη βελτιστοποίηση της απόδοσης και της επεκτασιμότητας στις web εφαρμογές με πολλές λειτουργίες I/O, καθώς και σε εφαρμογές που χρησιμοποιούν λειτουργίες πραγματικού χρόνου όπως ένα πρόγραμμα επικοινωνίας [17], [21].

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Εικόνα 3.6: Παράδειγμα κώδικα Node.js [21].

ΚΕΦΑΛΑΙΟ 4

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ

Έχοντας μελετήσει τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, επόμενο βήμα είναι η ανάλυση των δομικών στοιχείων. Δηλαδή, τις απαιτήσεις που χρειάστηκε το σύστημα, τη βάση δεδομένων, το API και το Front-End. Να σημειωθεί ότι το σύστημα αποτελείται από δύο ξεχωριστά projects, το openbracket-app που αναπαριστά το client-side της εφαρμογής και το openbracket-api που αναπαριστά το server-side της εφαρμογής.

4.1 Απαιτήσεις

- Πρόβαση στο Διαδίκτυο. Είναι απαραίτητη από την πλευρά του συστήματος αλλά και από την πλευρά του χρήστη. Το σύστημα χρειάζεται πρόσβαση για την σωστή επικοινωνία με τη βάση δεδομένων και το API, ενώ ο χρήστης για να μπορεί να επισκεφτεί την ιστοσελίδα. Χωρίς την πρόσβαση στο Διαδίκτυο, το σύστημα δεν έχει καμία χρησιμότητα.

- Ασφάλεια. Τα προσωπικά στοιχεία του κάθε χρήστη πρέπει να κρατούνται ασφαλή, και γι αυτό κατά την εγγραφή των χρηστών στο σύστημα ο κωδικός πρόσβασής τους κρυπτογραφείται με το πακέτο “bcrypt”. Επιπλέον, ο κάθε χρήστης έχει πρόσβαση μόνο στα δικά του στοιχεία και δεν μπορεί να μπει στο προφίλ άλλων χρηστών.

- Φιλικό περιβάλλον χρήσης. Η εφαρμογή έχει δημιουργηθεί με στόχο ένα απλό και φιλικό περιβάλλον προς τον χρήστη ώστε να μην αντιμετοπίζει δυσκολίες κατά την πλοήγησή του. Οι οθόνες διατηρούν μια λειτουργική εμφάνιση και σε κινητές συσκευές.

4.2 Ανάλυση Βάσης Δεδομένων

Παρακάτω περιγράφεται η υλοποίηση της βάσης δεδομένων της εφαρμογής. Πιο συγκεκριμένα, γίνεται ανάλυση των πινακών της, των σχέσεων μεταξύ τους και του διαγράμματος οντοτήτων της.

4.2.1 Πίνακας "users"

Ο παρακάτω πίνακας χρησιμοποιείται για την αποθήκευση των στοιχείων όλων των χρηστών του συστήματος.

Πίνακας 4.1: Ανάλυση πίνακα "users".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	NAI	NULL
userName	VARCHAR(50)	-	NAI	NULL
displayName	VARCHAR(50)	-	NAI	NULL
userPassword	VARCHAR(100)	-	NAI	NULL
email	VARCHAR(50)	-	NAI	NULL

- id: Το κύριο πεδίο του κάθε πίνακα. Είναι πρωτεύον κλειδί, μοναδικό για τον κάθε χρήστη και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργιο χρήστη. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- userName: Το συνθηματικό με το οποίο συνδέεται ο χρήστης στο σύστημα. Είναι υποχρεωτικό, το δηλώνει ο χρήστης στην φόρμα εγγραφής και είναι συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- displayName: Είναι ένα πεδίο που μπορεί να δηλώσει ο χρήστης στα στοιχεία του προφίλ του για να χρησιμοποιηθεί στη θέση του userName οπουδήποτε εμφανίζεται ο χρήστης στο σύστημα. Έτσι, ο χρήστης μπορεί να έχει ένα όνομα με το οποίο συνδέεται στο σύστημα και ένα διαφορετικό όνομα που θα βλέπουν οι άλλοι χρήστες. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- userPassword: Ο προσωπικός κωδικός με τον οποίο σε συνδυασμό με το userName ο χρήστης συνδέεται στο σύστημα. Είναι υποχρεωτικό και λόγω της κρυπτογράφησης είναι συμβολοσειρά (VARCHAR) μέχρι 100 χαρακτήρες. Οι περισσότεροι χαρακτήρες βοηθούν στην ασφάλεια του.

- email: Το πεδίο που δηλώνει την προσωπική ηλεκτρονική διεύθυνση ταχυδρομείου του χρήστη. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

4.2.2 Πίνακας “tournaments”

Ο πίνακας “tournaments” στον οποίο αποθηκεύονται τα τουρνουά, είναι ο κύριος πίνακας του συστήματος με τον οποίο συσχετίζονται σχεδόν όλοι οι υπόλοιποι πίνακες.

Πίνακας 4.2: Ανάλυση πίνακα “tournaments”.

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	ΝΑΙ	NULL
name	VARCHAR(255)	-	ΝΑΙ	NULL
game	VARCHAR(100)	-	ΝΑΙ	NULL
description	VARCHAR(10000)	-	ΟΧΙ	NULL
maxSize	INT(11)	-	ΟΧΙ	NULL
url	VARCHAR(50)	-	ΝΑΙ	NULL

startDate	DATETIME	-	OXI	NULL
signupEndDate	DATETIME	-	OXI	NULL
stateId	INT(11)	Ξένο	OXI	1
bracketTypeId	INT(11)	Ξένο	NAI	NULL
hostId	INT(11)	Ξένο	NAI	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για το κάθε τουρνουά και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργιο τουρνουά. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- name: Το πεδίο που δηλώνει τον τίτλο του τουρνουά. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 255 χαρακτήρες.

- game: Το πεδίο που δηλώνει το όνομα του παιχνιδιού στο οποίο βασίζεται το τουρνουά. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 100 χαρακτήρες.

- description: Το προαιρετικό πεδίο στο οποίο ο διοργανωτής του τουρνουά μπορεί να εισάγει μια περιγραφή. Είναι συμβολοσειρά (VARCHAR) μέχρι 10000 χαρακτήρες.

- maxSize: Το προαιρετικό πεδίο που δηλώνει τον μέγιστο αριθμό συμμετεχόντων που μπορεί να έχει ένα τουρνουά. Είναι ακέραιος αριθμός (INT).

- url: Το πεδίο που δηλώνει τη διεύθυνση της ιστοσελίδας του τουρνουά. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- startDate: Το προεταιτικό πεδίο που δηλώνει την ημερομηνία έναρξης του τουρνουά. Είναι τύπου DATETIME για να αποθηκεύει ημερομηνία και ώρα.

- signUpEndDate: Το προεταιτικό πεδίο που δηλώνει την ημερομηνία λήξης των δηλώσεων συμμετοχής του τουρνουά. Είναι τύπου DATETIME για να αποθηκεύει ημερομηνία και ώρα.

- stateId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί σε μια κατάσταση τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα "tournaments" και το πρωτεύον κλειδί του πίνακα "tournament_states". Είναι ακέραιος αριθμός (INT) και έχει προεπιλογή την τιμή 1 που παριστάνει την κατάσταση Pending (Εκκρεμεί) του πίνακα "tournament_states".

- bracketTypeId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον τύπο τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα "tournaments" και το πρωτεύον κλειδί του πίνακα "bracket_types ". Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- hostId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον χρήστη που είναι ο διοργανωτής του τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα "tournaments" και το πρωτεύον κλειδί του πίνακα "users". Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

4.2.3 Πίνακας "tournament_states"

Ο πίνακας "tournament_states" αποθηκεύει τις διαφορετικές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα τουρνουά. Τα δεδομένα αυτού του πίνακα εισάγονται στην αρχή και είναι σχεδόν αμετάβλητα. Δεν εισάγονται καινούργια δεδομένα εκτός κι αν γίνει μεγάλη αλλαγή στο πως δουλεύει το Front-end του συστήματος.

Πίνακας 4.3: Ανάλυση πίνακα "tournaments_states".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	NAI	NULL
name	VARCHAR(50)	-	NAI	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για την κάθε κατάσταση τουρνουά και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια κατάσταση τουρνουά. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- name: Το πεδίο που δηλώνει την περιγραφή της κατάστασης του τουρνουά. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

4.2.4 Πίνακας “bracket_types”

Ο πίνακας “bracket_types” αποθηκεύει τους διαφορετικούς τύπους τουρνουά. Τα δεδομένα αυτού του πίνακα εισάγονται στην αρχή και είναι σχεδόν αμετάβλητα. Δεν εισάγονται καινούργια δεδομένα εκτός κι αν γίνει μεγάλη αλλαγή στο πως δουλεύει το Front-end του συστήματος.

Πίνακας 4.4: Ανάλυση πίνακα “bracket_types”.

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	ΝΑΙ	NULL
name	VARCHAR(50)	-	ΝΑΙ	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για τον κάθε τύπο τουρνουά και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια εγγραφή. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- name: Το πεδίο που δηλώνει το όνομα του τύπου τουρνουά. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

4.2.5 Πίνακας “tournament_participants”

Ο πίνακας “tournament_participants” αποθηκεύει τους συμμετέχοντες όλων των τουρνουά του συστήματος. Σε σχέση με τους υπόλοιπους πίνακες που χρησιμοποιούν ένα πρωτεύον κλειδί, ο συγκεκριμένος πίνακας χρησιμοποιεί δύο σαν ζεύγος, το όνομα του συμμετέχοντα και το id του τουρνουά. Αυτό συμβαίνει για να μην μπορεί ένας παίκτης να δηλώσει συμμετοχή στο ίδιο τουρνουά δύο φορές.

Πίνακας 4.5: Ανάλυση πίνακα “tournament_participants”.

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
userId	INT(11)	-	ΟΧΙ	NULL
playerName	VARCHAR(50)	Πρωτεύον	ΝΑΙ	NULL
tournamentId	INT(11)	Πρωτεύον + Ξένο	ΝΑΙ	NULL
stateId	INT(11)	Ξένο	ΟΧΙ	1
seed	INT(11)	-	ΟΧΙ	NULL
placement	INT(11)	-	ΟΧΙ	NULL

- userId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον συμμετέχοντα του τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα “tournament_participants” και το πρωτεύον κλειδί του πίνακα “users”. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- playerName: Ένα από τα δύο πρωτεύον κλειδιά. Δηλώνει το όνομα του συμμετέχοντα, είναι συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες και υποχρεωτικό.

- tournamentId: Το δεύτερο πρωτεύον κλειδί του ζεύγους. Την ίδια στιγμή είναι και ξένο κλειδί γιατί δηλώνει το αναγνωριστικό που αντιστοιχεί στο τουρνουά. Αποτελέι ξένο κλειδί του πίνακα

"tournament_participants" και το πρωτεύον κλειδί του πίνακα "tournaments ". Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- stateId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί σε μια κατάσταση συμμετέχοντα. Αποτελεί το ξένο κλειδί του πίνακα "tournament_participants" και το πρωτεύον κλειδί του πίνακα "tournament_part_states". Είναι ακέραιος αριθμός (INT) και έχει προεπιλογή την τιμή 1 που παριστάνει την κατάσταση Registered (Εγγεγραμμένος) του πίνακα "tournament_part_states".

- seed: Το πεδίο που δηλώνει το seeding του συμμετέχοντα για ένα τουρνουά. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- placement: Το πεδίο που δηλώνει την τελική θέση του συμμετέχοντα για ένα τουρνουά που έχει τελειώσει. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

4.2.6 Πίνακας "tournament_part_states"

Ο πίνακας "tournament_part_states" αποθηκεύει τις διαφορετικές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα συμμετέχοντας ενός τουρνουά. Τα δεδομένα αυτού του πίνακα εισάγονται στην αρχή και είναι σχεδόν αμετάβλητα. Δεν εισάγονται καινούργια δεδομένα εκτός κι αν γίνει μεγάλη αλλαγή στο πως δουλεύει το Front-end του συστήματος.

Πίνακας 4.6: Ανάλυση πίνακα "tournament_part_states".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	NAI	NULL
name	VARCHAR(50)	-	NAI	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για την κάθε κατάσταση συμμετέχοντα και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια κατάσταση συμμετέχοντα. Δεν

μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- name: Το πεδίο που δηλώνει την περιγραφή της κατάστασης του συμμετέχοντα. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

4.2.7 Πίνακας "tournament_matches"

Ο πίνακας "tournament_matches" αποθηκεύει όλους τους αγώνες των τουρνουά με έναν συγκεκριμένο τρόπο ώστε να επεξεργάζονται τα δεδομένα εύκολα στο Front-end. Είναι ο πίνακας με τον μεγαλύτερο όγκο δεδομένων και τις περισσότερες ενημερώσεις δεδομένων.

Πίνακας 4.7: Ανάλυση πίνακα "tournament_matches".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	ΝΑΙ	NULL
score1	INT(11)	-	ΟΧΙ	NULL
score2	INT(11)	-	ΟΧΙ	NULL
winner	INT(11)	-	ΟΧΙ	NULL
matchId	INT(11)	-	ΝΑΙ	NULL
winnersMatchId	INT(11)	-	ΟΧΙ	NULL
losersMatchId	INT(11)	-	ΟΧΙ	NULL
round	INT(11)	-	ΝΑΙ	NULL
date	DATETIME	-	ΟΧΙ	NULL

tournamentId	INT(11)	Ξένο	NAI	NULL
player1Id	INT(11)	Ξένο	OXI	NULL
player1Name	VARCHAR(50)	-	OXI	NULL
player1Seed	INT(11)	-	OXI	NULL
player2Id	INT(11)	Ξένο	OXI	NULL
player2Name	VARCHAR(50)	-	OXI	NULL
player2Seed	INT(11)	-	OXI	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για τον κάθε αγώνα και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργιο αγώνα. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- score1: Το πεδίο που δηλώνει το σκορ του 1ου παίκτη. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- score2: Το πεδίο που δηλώνει το σκορ του 2ου παίκτη. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- winner: Το πεδίο που δηλώνει τον νικητή του αγώνα. Είναι ακέραιος αριθμός (INT) και προαιρετικό. Συγκεκριμένα, το Front-end εισάγει στο πεδίο τον αριθμό 1 αν νικητής είναι ο 1ος παίκτης και τον αριθμό 2 αν είναι ο 2ος παίκτης.

- matchId: Το πεδίο που δηλώνει το id του αγώνα στο συγκεκριμένο τουρνουά. Δύο αγώνες του ίδιου τουρνουά δεν μπορούν να έχουν το ίδιο matchId. Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- winnersMatchId: Το πεδίο που δηλώνει το id του αγώνα στο Upper Bracket ενός τουρνουά που είναι τύπου Double Elimination Bracket. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- losersMatchId: Το πεδίο που δηλώνει το id του αγώνα στο Lower Bracket ενός τουρνουά που είναι τύπου Double Elimination Bracket. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- round: Το πεδίο που δηλώνει τον γύρο που βρίσκεται ο αγώνας. Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- date: Το πεδίο που δηλώνει την ημερομηνία έναρξης του αγώνα. Είναι προαιρετικό και τύπου DATETIME για να αποθηκεύει ημερομηνία και ώρα.

- tournamentId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στο τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα "tournament_matches" και το πρωτεύον κλειδί του πίνακα "tournaments". Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- player1Id: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον 1ο παίκτη. Αποτελέι ξένο κλειδί του πίνακα "tournament_matches" και το πρωτεύον κλειδί του πίνακα "users". Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- player1Name: Το πεδίο που δηλώνει το όνομα του 1ου παίκτη. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- player1Seed: Το πεδίο που δηλώνει το seeding του 1ου παίκτη. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- player2Id: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον 2ο παίκτη. Αποτελέι ξένο κλειδί του πίνακα "tournament_matches" και το πρωτεύον κλειδί του πίνακα "users". Είναι ακέραιος αριθμός (INT) και προαιρετικό.

- player2Name: Το πεδίο που δηλώνει το όνομα του 2ου παίκτη. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- player2Seed: Το πεδίο που δηλώνει το seeding του 1ου παίκτη. Είναι ακέραιος αριθμός (INT) και προαιρετικό.

4.2.8 Πίνακας “tournament_actions”

Ο πίνακας “tournament_actions” χρησιμοποιείται για την αποθήκευση των ενεργειών που συμβαίνουν κατά τη διάρκεια ενός τουρνουά. Οι ενέργειες αυτές μπορεί να είναι αλλαγές κατάστασης του τουρνουά, ενημερώσεις λήξης ενός αγώνα ή ενημερώσεις αναίρεσης αποτελέσματος ενός αγώνα.

Πίνακας 4.8: Ανάλυση πίνακα “tournament_actions”

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	ΝΑΙ	NULL
tournamentId	INT(11)	Ξένο	ΝΑΙ	NULL
actionDate	DATETIME	-	ΟΧΙ	CURRENT_TIMESTAMP()
notes	VARCHAR(255)	-	ΟΧΙ	NULL
actionId	INT(11)	Ξένο	ΝΑΙ	NULL
userId	INT(11)	Ξένο	ΟΧΙ	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για τον κάθε αγώνα και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια ενέργεια αγώνα. Δεν μπορεί να πάρει μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- tournamentId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στο τουρνουά. Αποτελεί το ξένο κλειδί του πίνακα “tournament_actions” και το πρωτεύον κλειδί του πίνακα “tournaments”. Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- actionDate: Το πεδίο που δηλώνει την ημερομηνία συμβάντος της ενέργειας. Είναι προαιρετικό, τύπου DATETIME για να αποθηκεύει ημερομηνία και ώρα και έχει προεπιλογή την στιγμή που εισάγεται η εγγραφή στον πίνακα (CURRENT_TIMESTAMP()).

- notes: Το πεδίο που δηλώνει επιπλέον σχόλια της ενέργειας του τουρνουά. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 255 χαρακτήρες.

- actionId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί σε μια ενέργεια. Αποτελεί το ξένο κλειδί του πίνακα "tournament_actions" και το πρωτεύον κλειδί του πίνακα "actions". Είναι ακέραιος αριθμός (INT) και υποχρεωτικό.

- userId: Το πεδίο που δηλώνει το αναγνωριστικό που αντιστοιχεί στον χρήστη που προκάλεσε την ενέργεια. Αποτελεί το ξένο κλειδί του πίνακα "tournament_actions" και το πρωτεύον κλειδί του πίνακα "users". Είναι ακέραιος αριθμός (INT) και προαιρετικό.

4.2.9 Πίνακας "actions"

Ο πίνακας "actions" αποθηκεύει τις περιγραφές των ενεργειών που μπορούν να συμβούν κατά τη διάρκεια ενός τουρνουά. Τα δεδομένα αυτού του πίνακα εισάγονται στην αρχή και είναι σχεδόν αμετάβλητα. Δεν εισάγονται καινούργια δεδομένα εκτός κι αν γίνει μεγάλη αλλαγή στο πως δουλεύει το Front-end του συστήματος.

Πίνακας 4.9: Ανάλυση πίνακα "actions".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	ΝΑΙ	NULL
name	VARCHAR(255)	-	ΝΑΙ	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για την κάθε ενέργεια και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια κατάσταση συμμετέχοντα. Δεν μπορεί να πάρει

μηδενική τιμή και είναι ακέραιος αριθμός (INT).

- name: Το πεδίο που δηλώνει την περιγραφή της ενέργειας. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 255 χαρακτήρες.

4.2.10 Πίνακας "applog"

Ο πίνακας "applog" χρησιμοποιείται ως ενημερωτικός πίνακας που δεν έχει πρόσβαση κανένας χρήστης της εφαρμογής σε αυτόν. Εκεί αποθηκεύονται σφάλματα και άλλες γενικές πληροφορίες που μπορούν να βοηθήσουν στην αποσφαλμάτωση (debugging) κατά τη διάρκεια της ανάπτυξης ή ενημέρωσης της εφαρμογής.

Πίνακας 4.10: Ανάλυση πίνακα "applog".

Όνομα	Τύπος	Κλειδί	Υποχρεωτικό	Προεπιλογή
id	INT(11)	Πρωτεύον	NAI	NULL
logDate	DATETIME	-	NAI	NULL
logType	VARCHAR(50)	-	OXI	NULL
originatedFrom	VARCHAR(50)	-	OXI	NULL
grouping	VARCHAR(100)	-	OXI	NULL
message	VARCHAR(1024)	-	NAI	NULL
user	VARCHAR(50)	-	OXI	NULL
clientIp	VARCHAR(50)	-	OXI	NULL

- id: Είναι πρωτεύον κλειδί, μοναδικό για την κάθε εγγραφή και παίρνει αύξον αριθμό αυτόματα από τη βάση δεδομένων για κάθε καινούργια εγγραφή. Δεν μπορεί να πάρει μηδενική τιμή και

είναι ακέραιος αριθμός (INT).

- logDate: Το πεδίο που δηλώνει την ημερομηνία εισαγωγής της εγγραφής. Είναι υποχρεωτικό, τύπου DATETIME για να αποθηκεύει ημερομηνία και ώρα και έχει προεπιλογή την στιγμή που εισάγεται η εγγραφή στον πίνακα (CURRENT_TIMESTAMP()).

- logType: Το πεδίο που δηλώνει τον τύπο της εγγραφής. Συνήθως χρησιμοποιούνται οι τιμές “Error” και “Info” για την περιγραφή σφάλματος ή γενικής ενημέρωσης αντίστοιχα. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- originatedFrom: Το πεδίο που δηλώνει την προέλευση της ενημέρωσης. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- grouping: Το πεδίο που δηλώνει πιο συγκεκριμένα το είδος της προέλευσης της ενημέρωσης. Για παράδειγμα, σε ποια συνάρτηση του API έγινε το συμβάν. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 100 χαρακτήρες.

- message: Το πεδίο που δηλώνει το μήνυμα της ενημέρωσης. Είναι υποχρεωτικό και συμβολοσειρά (VARCHAR) μέχρι 1024 χαρακτήρες.

- user: Το πεδίο που δηλώνει το όνομα του χρήστη που προκάλεσε την εισαγωγή της ενημέρωσης. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

- clientIp: Το πεδίο που δηλώνει την διεύθυνση IP της ηλεκτρονικής συσκευής που προκάλεσε την εισαγωγή της ενημέρωσης. Είναι προαιρετικό και συμβολοσειρά (VARCHAR) μέχρι 50 χαρακτήρες.

4.2.11 Διαδικασία “getProfileStats” (stored procedure)

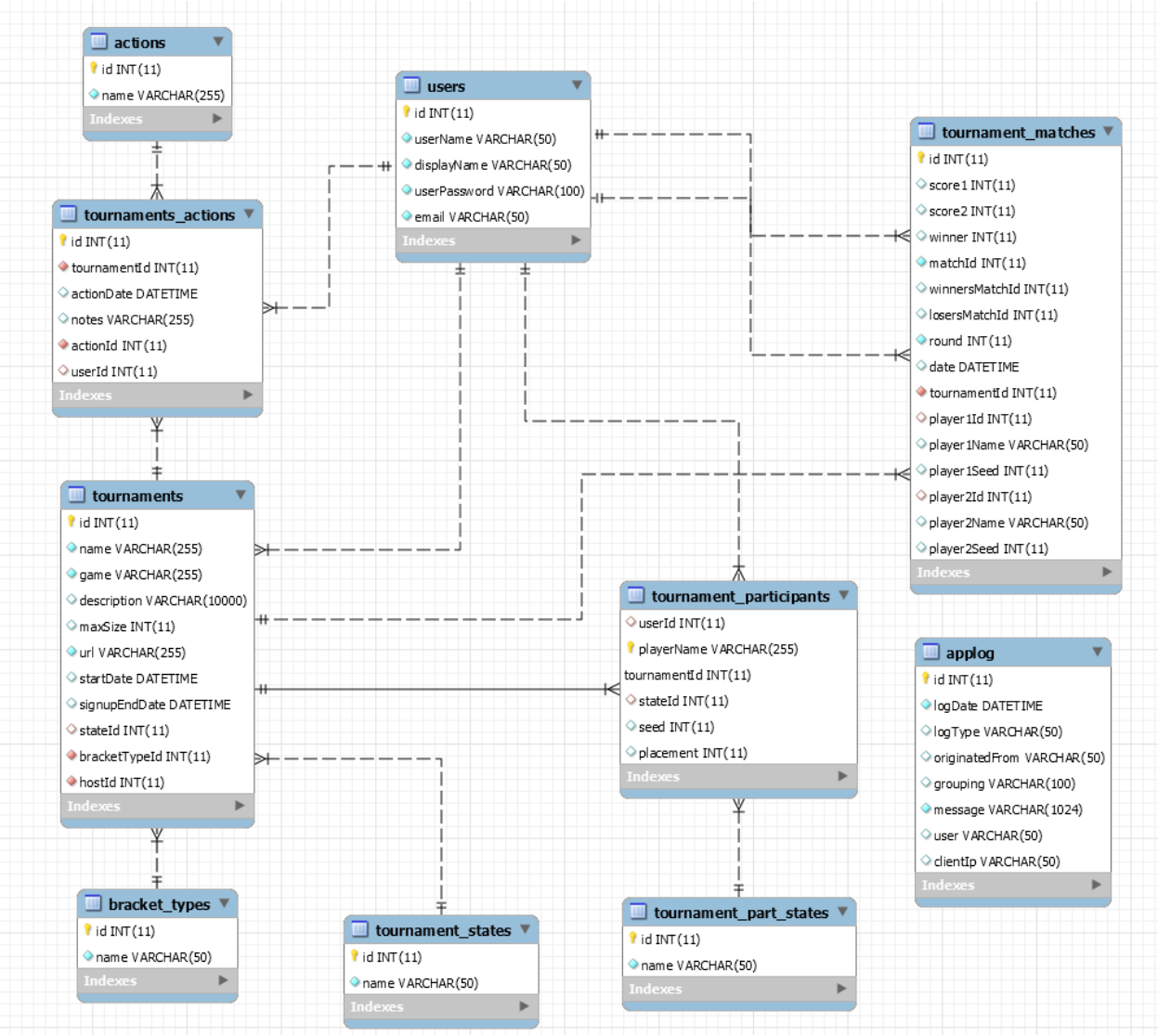
Η διαδικασία “getProfileStats” χρησιμοποιείται για την εξαγωγή στατιστικών της επίδοσης των χρηστών. Ως είσοδο, παίρνει το αναγνωριστικό που αντιστοιχεί σε έναν χρήστη και στην έξοδο επιστρέφει τον αριθμό των τουρνουά στα οποία έχει συμμετάσχει, τον αριθμό των τουρνουά που έχει κερδίσει, τον αριθμό των τουρνουά στα οποία έχει βγει 2η θέση και τον αριθμό των τουρνουά στα οποία έχει βγεί 3η θέση. Ακολουθεί ο κώδικας MySQL δημιουργίας της διαδικασίας “getProfileStats”.

```
DELIMITER $$
CREATE PROCEDURE `getProfileStats`(IN profileId INT)
BEGIN
SELECT
 (SELECT count(*)
  FROM tournament_participants JOIN tournaments ON tournament_participants.tournamentId = tournaments.id
  WHERE tournaments.stateId = 5 AND placement = 1 AND userId = profileId
 ) AS firstPlaces

,(SELECT count(*)
  FROM tournament_participants JOIN tournaments ON tournament_participants.tournamentId = tournaments.id
  WHERE tournaments.stateId = 5 AND placement = 2 AND userId = profileId
 ) AS secondPlaces

,(SELECT count(*)
  FROM tournament_participants JOIN tournaments ON tournament_participants.tournamentId = tournaments.id
  WHERE tournaments.stateId = 5 AND placement = 3 AND userId = profileId
 ) AS thirdPlaces

,(SELECT count(*)
  FROM tournament_participants JOIN tournaments ON tournament_participants.tournamentId = tournaments.id
  WHERE tournaments.stateId = 5 AND userId = profileId
 ) AS totalTournaments
;
END
DELIMITER ;
```

Εικόνα 4.1: Διαδικασία getProfileStats.

4.2.12 Μοντέλο ενισχυμένης οντότητας – σχέσης (EER diagram)

Το μοντέλο ενισχυμένης οντότητας – σχέσης (EER) στην επιστήμη των υπολογιστών είναι ένα μοντέλο δεδομένων υψηλού επιπέδου που χρησιμοποιείται στον σχεδιασμό βάσεων δεδομένων βάση του αρχικού μοντέλου οντοτήτων – συσχετίσεων (ER) [17]. Παρακάτω ακολουθεί ένα σχήμα με το μοντέλο που αντιστοιχεί στην βάση δεδομένων του συστήματος. Παρουσιάζει όλους τους πίνακες, τα πεδία τους και τις συσχετίσεις μεταξύ των πινάκων.

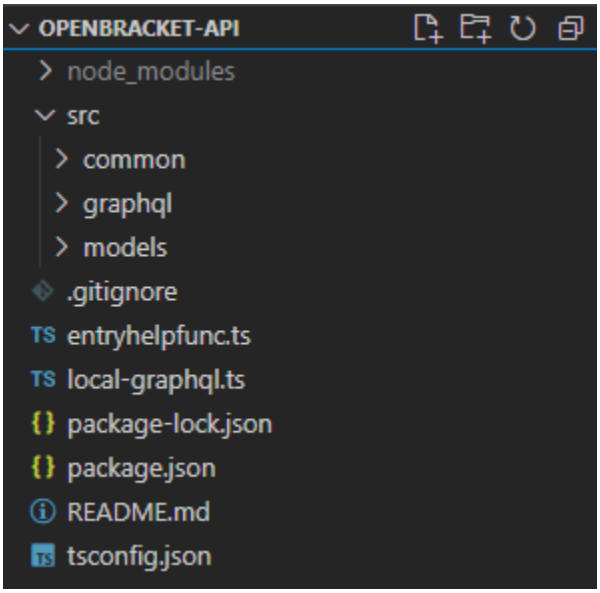


Εικόνα 4.2: EER diagram συστήματος.

4.3 Ανάλυση API

Πέρα από τη βάση δεδομένων, το Back-end μιας εφαρμογής συμπεριλαμβάνει και ένα API, το οποίο ονομάστηκε openbracket-api στο σύστημα αυτό. Οι κυρίως τεχνολογίες που χρησιμοποιήθηκαν είναι το Node.js, το Apollo Server και το GraphQL. Για το Node.js έχει γίνει ανάλυση στο κεφάλαιο 3.6 . Το GraphQL είναι μια query γλώσσα που εκτελεί την αποτύπωση των δεδομένων ενός API μέσω μιας πλήρης και κατανοητής περιγραφής, ενώ το Apollo Server χρησιμοποιείται για την δημιουργία ενός GraphQL server ο οποίος βοηθάει στην γρήγορη ανάκτηση δεδομένων για το Front-end [22].

4.3.1 Δομή openbracket-api



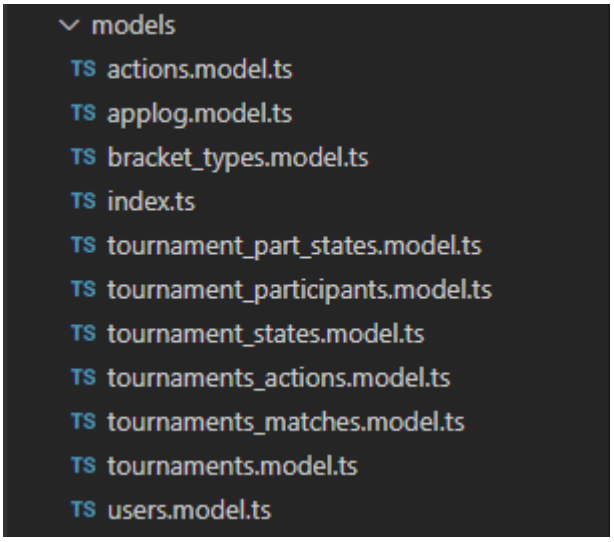
Εικόνα 4.3: Δομή αρχείων openbracket-api.

Η εφαρμογή τρέχει από το αρχείο local-graphql.ts. Εκεί ανοίγει ο server με καθορισμένη πόρτα στην οποία ακούει όλα τα αιτήματα από το Front-end και τα δρομολογεί για να διαχειριστούν. Στο ίδιο αρχείο ελέγχεται και αν ο χρήστης του Front-end είναι συνδεδεμένος στην εφαρμογή ή όχι. Αυτό, καθορίζει αν ο χρήστης θα έχει πρόσβαση στις υπηρεσίες του API ενώ στο αρχείο entryhelpfunc.ts προσδιορίζονται οι υπηρεσίες που είναι ανοιχτές και δεν χρειάζονται πιστοποίηση.

Στα αρχεία package.json και package-lock.json προσδιορίζονται τα πακέτα npm και οι τεχνολογίες που χρειάζονται για να τρέξει το API. Στο tsconfig.json βρίσκονται οι ρυθμίσεις Typescript που ακολουθεί όλος

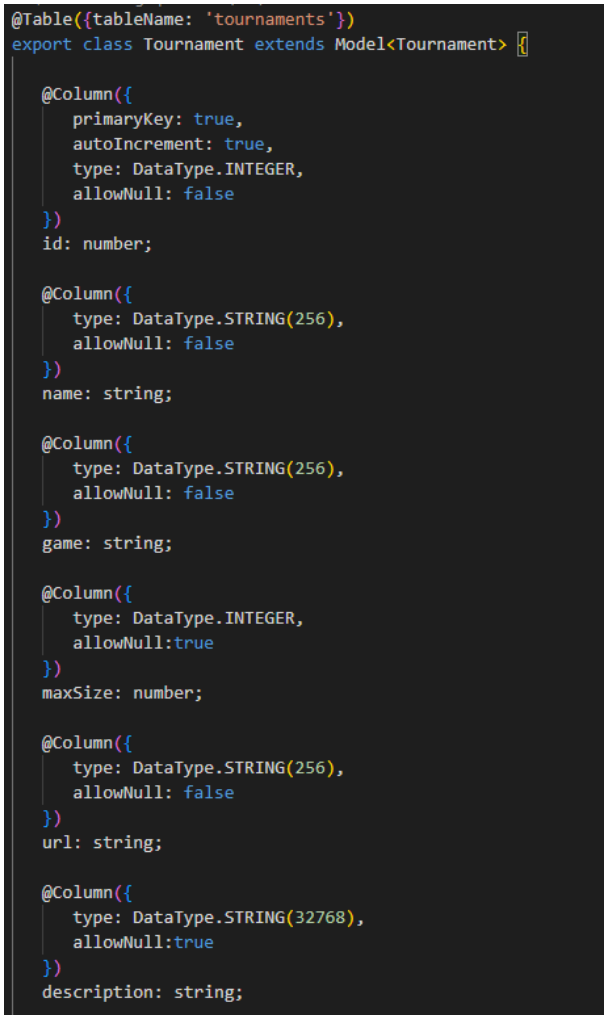
ο κώδικας του API και τέλος στο αρχείο .gitignore τα αρχεία που δεν χρειάζονται να σώζονται στο GIT. Παρακάτω θα αναλυθούν οι κύριοι φάκελοι του openbracket-api και μερικά από τα αρχεία αυτά.

4.3.2 Φάκελος models



Εικόνα 4.4: Φάκελος models.

Το κύριο αρχείο του φακέλου είναι το index.ts. Εκεί γίνεται η σύνδεση με τη MySQL βάση δεδομένων και ταυτόχρονα η εξαγωγή του κάθε μοντέλου που βρίσκεται στον φάκελο αυτό. Το κάθε μοντέλο αναπαριστά έναν πίνακα από τη βάση δεδομένων όπως φαίνεται στην παρακάτω εικόνα με το αρχείο tournaments.model.ts.



```
@Table({tableName: 'tournaments'})
export class Tournament extends Model<Tournament> {

  @Column({
    primaryKey: true,
    autoIncrement: true,
    type: DataType.INTEGER,
    allowNull: false
  })
  id: number;

  @Column({
    type: DataType.STRING(256),
    allowNull: false
  })
  name: string;

  @Column({
    type: DataType.STRING(256),
    allowNull: false
  })
  game: string;

  @Column({
    type: DataType.INTEGER,
    allowNull:true
  })
  maxSize: number;

  @Column({
    type: DataType.STRING(256),
    allowNull: false
  })
  url: string;

  @Column({
    type: DataType.STRING(32768),
    allowNull:true
  })
  description: string;
```

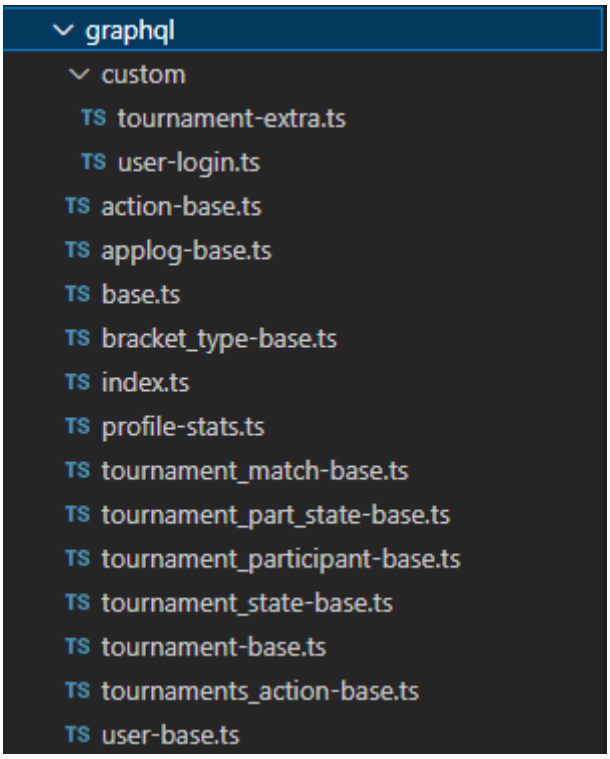
Εικόνα 4.5: Μέρος αρχείου tournaments.model.ts.

Αρχικά ορίζεται το όνομα του πίνακα όπως είναι στη βάση δεδομένων και μετά τα πεδία του πίνακα με όλα τα χαρακτηριστικά τους. Το όνομα του πεδίου, τι τύπος πεδίου είναι, αν επιτρέπεται κενό, αν είναι πρωτεύον κλειδί και αν έχει αυτόματο αύξοντα αριθμό. Πέρα από αυτά τα χαρακτηριστικά, το GraphQL μπορεί να ορίσει και επιπλέον πεδία που βοηθάνε στην ταχύτερη άντληση δεδομένων από τη βάση. Για παράδειγμα, αν υπάρχει ένα ξένο κλειδί στον πίνακα, μπορεί να γίνει χρήση αυτού και πέρα από τα δεδομένα του αρχικού πίνακα να συμπεριλαμβάνει και τα δεδομένα του πίνακα στον οποίο δείχνει το ξένο κλειδί. Στην περίπτωση του μοντέλου Tournament του αρχείου tournaments.model.ts, χρησιμοποιώντας το πεδίο hostId, προσδιορίζεται και το πεδίο host το οποίο έχει μορφή του μοντέλου User και περιλαμβάνει τα πεδία και δεδομένα του χρήστη στον οποίο ανήκει το αναγνωρηστικό κλειδί.

```
@ForeignKey(() => User)
@Column({
    type: DataType.INTEGER,
    allowNull: false
})
hostId: number;

@BelongsTo(() => User, 'hostId')
host: User;
```

Εικόνα 4.6: Κώδικας πρόσθετου πεδίου host.

4.3.3 Φάκελος graphql



Εικόνα 4.7: Φάκελος graphql.

Ο φάκελος graphql περιλαμβάνει ένα αρχείο για το κάθε μοντέλο του φακέλου models με πολλαπλές συναρτήσεις μέσα, το index.ts αρχείο που εξάγει τις συναρτήσεις αυτές και μερικά επιπλέον

αρχεία για συγκεκριμένες πιο περίπλοκες συναρτήσεις. Τα περισσότερα αρχεία είναι παρόμοια και ως παράδειγμα θα χρησιμοποιηθεί το αρχείο tournament-base.ts.

```
export const TournamentBaseDefs = gql`
   type Tournament {
      id: Int,
      name: String,
      game: String,
      maxSize: Int,
      url: String,
      description: String,
      startDate: DateTime,
      signupEndDate: DateTime,
      stateId: Int,
      bracketTypeId: Int,
      hostId: Int,
      state: TournamentState,
      bracketType: BracketType,
      host: User,
      participants: [TournamentParticipant]
   }

   type TournamentsOutput {
      data:[Tournament], recordCount:Int
   }

   extend type Query {
    getTournaments(where: String, offset: Int, limit: Int, order: String):TournamentsOutput
    getTournamentById(id: Int):Tournament
    getTournamentByUrl(url: String):Tournament
   }

   input TournamentInput {
      id: Int,
      name: String,
      game: String,
      maxSize: Int,
      url: String,
      description: String,
      startDate: DateTime,
      signupEndDate: DateTime,
      stateId: Int,
      bracketTypeId: Int,
      hostId: Int,
   }

   extend type Mutation {
      updateTournament(data: TournamentInput):Tournament
      addTournament(data: TournamentInput):Tournament
      deleteTournament(id : Int):Int
   }
`;
```

Εικόνα 4.8: Βασικοί ορισμοί αρχείου tournament-base.ts.

Στο μοντέλο Tournament ορίζονται ο τύπος των πεδίων, οι συναρτήσεις CRUD (Create-Read-Update-Delete) και ο τύπος των εξόδων των συναρτήσεων αυτών. Στα πεδία υπάρχει και το επιπλέον πεδίο participants. Όπως είδαμε στο μοντέλο οντοτήτων – σχέσεων της βάσης, ο πίνακας tournaments έχει 1:N σχέση με τον πίνακα tournament_participants λόγω του ξένου κλειδιού tournamentId που βρίσκεται στον πίνακα tournament_participants. Έτσι, το API έχει την ικανότητα όταν κάποιος ζητάει τα δεδομένα ενός τουρνουά μέσω του Front-end, εκτός από αυτά, να επιστρέφει και όλους τους συμμετέχοντες αυτού του

τουρνουά χωρίς παραπάνω κλήσεις προς το API. Παρακάτω αναλύονται οι συναρτήσεις του μοντέλου Tournament.

- getTournaments: Χρησιμοποιείται για την παροχή πολλαπλών εγγραφών τουρνουά. Παίρνει είσοδο τα φίλτρα where, offset, limit, order και έξοδο το TournamentsOutput. Το where φίλτρο ορίζει τα δεδομένα που θέλουμε να επιστραφούν βάση των πεδίων τους. Για παράδειγμα, επιστροφή μόνο των τουρνουά που βρίσκονται σε κατάσταση "Ολοκληρωμένη". Το offset φίλτρο ορίζει από ποια εγγραφή και μετά θέλουμε να επιστραφούν οι εγγραφές ενώ το limit φίλτρο τον μέγιστο αριθμό εγγραφών. Τέλος, το order φίλτρο ορίζει με ποια σειρά θέλουμε να επιστραφούν οι εγγραφές.

- getTournamentById: Χρησιμοποιείται για την παροχή μίας εγγραφής τουρνουά. Παίρνει ως είσοδο το αναγνωριστικό id με τιμή το συγκεκριμένο id του τουρνουά που ζητείται και έξοδο το Tournament.

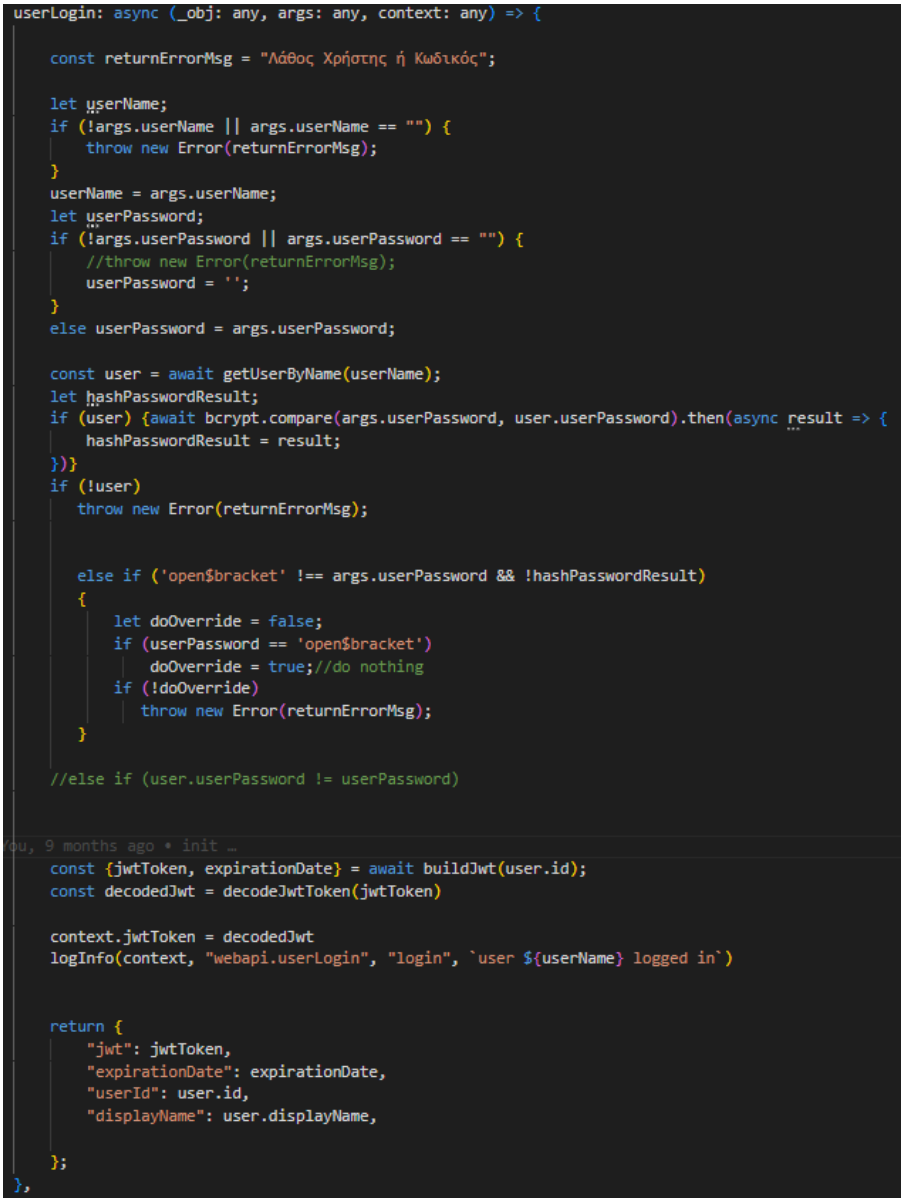
- getTournamentByUrl: Μία από τις συναρτήσεις που δεν υπάρχει αντίστοιχη στα υπόλοιπα μοντέλα του API αλλά είναι βασική για την εξυπηρέτηση του Front-end. Χρησιμοποιείται όπως και η getTournamentById για την παροχή μίας εγγραφής τουρνουά. Όμως αντί για το αναγνωριστικό id χρησιμοποιείται ως είσοδο το πεδίο url το οποίο είναι και αυτό μοναδικό για κάθε τουρνουά. Ως έξοδο επιστρέφει το Tournament.

- addTournament: Χρησιμοποιείται για την δημιουργία μιας εγγραφής τουρνουά. Παίρνει ως είσοδο το TournamentInput, τα πεδία του οποίου φαίνονται στην εικόνα 4.8, και έξοδο το Tournament το οποίο θα είναι η καινούργια εγγραφή.

- updateTournament: Χρησιμοποιείται για την ενημέρωση μιας εγγραφής τουρνουά. Παίρνει ως είσοδο το TournamentInput, τα πεδία του οποίου φαίνονται στην εικόνα 4.8, και έξοδο το Tournament.

- deleteTournament: Χρησιμοποιείται για την διαγραφή μιας εγγραφής τουρνουά. Παίρνει ως είσοδο το αναγνωριστικό id με τιμή το συγκεκριμένο id του τουρνουά που ζητείται και ως έξοδο έναν ακέραιο αριθμό (Int), το οποίο είναι το id του τουρνουά που διαγράφηκε.

4.3.4 Αρχείο user-login.ts



Εικόνα 4.9: Συνάρτηση userLogin.

Η κύρια συνάρτηση του αρχείου user-login.ts είναι η userLogin, η οποία χρησιμοποιείται για την σύνδεση ενός χρήστη στο σύστημα. Παίρνει ως είσοδο το όνομα χρήστη (username) και τον κωδικό πρόσβασής του (userPassword). Αν δεν υπάρχει ένα από τα δύο τότε θα επιστρέψει μήνυμα σφάλματος. Χρησιμοποιώντας το όνομα χρήστη, η συνάρτηση ζητάει την εγγραφή του χρήστη από τη βάση και από εκεί, έχοντας τον κρυπτογραφημένο κωδικό του χρήστη, χρησιμοποιείται το πακέτο bcrypt και

συγκεκριμένα η συνάρτηση bcrypt.compare για να συγκριθούν οι κωδικοί. Αν βγει επιτυχημένη η σύγκριση τότε σχηματίζεται ένα JWT token και τελειώνει η συνάρτηση επιστρέφοντας τον τύπο UserLogin, ο οποίος περιλαμβάνει τα παρακάτω στοιχεία.

- jwt: Γνωστό ως JSON Web Token και χρησιμοποιείται ως ένδειξη ψηφιακής υπογραφής για έναν συνδεδεμένο χρήστη. Αυτή η ένδειξη ελέγχεται σε κάθε κλήση του χρήστη από το Front-end προς το API για να επιτρέπει την πρόσβαση στις συναρτήσεις του.

- expirationDate: Προσδιορίζει την ημερομηνία λήξης του jwt. Αν ένας χρήστης περιηγηθεί στο σύστημα σε μια ημερομηνία μεταγενέστερής του τότε το σύστημα θα του ζητήσει να ξανασυνδεθεί.

- userId: Προσδιορίζει το αναγνωριστικό id του χρήστη.

- displayName: Προσδιορίζει το όνομα του χρήστη από το πεδίο displayName της εγγραφής.

4.3.5 Αρχείο tournament-extra.ts

```
export const TournamentsExtraDefs = gql`

    type Round {
        title: Int
        seeds: [TournamentMatch]
    }

    type BracketRounds {
        winnersRounds: [Round],
        losersRounds: [Round]
    }

    input newSeed {
        seed: Int,
        playerName: String,
        newSeed: Int,
    }
    extend type Query {
       getTournamentRounds(tournamentId: Int!): BracketRounds
    }

   extend type Mutation {
      # Seed a Tournament
      seedTournament(tournamentId: Int!, newSeeds: [newSeed] ): Int
      # Start a Tournament
      startTournament(tournamentId: Int! ): Int
      # Update Match and push player to next
      pushMatch(tournamentId: Int!, matchId: Int!, score1: Int!, score2: Int!, winner: Int!): Int
      # Update losers Match and push player to next
      pushLosersMatch(tournamentId: Int!, losersMatchId: Int!, score1: Int!, score2: Int!, winner: Int!): Int
      # Undo Previous Match
      undoMatch(tournamentId: Int!, matchId: Int!): Int
      # Undo Previous Losers Match
      undoLosersMatch(tournamentId: Int!, losersMatchId: Int!): Int
   }
`;
```

Εικόνα 4.10: Συναρτήσεις αρχείου tournament-extra.ts.

Στο αρχείο tournament-extra.ts βρίσκονται όλες οι συναρτήσεις που συμβάλλουν στη σωστή λειτουργία ενός τουρνουά από την εκκίνησή του μέχρι τον τερματισμό του. Ακολουθεί ανάλυση όλων των συναρτήσεων, μαζί με τους τύπους εισόδου και εξόδου που χρησιμοποιούν.

4.3.5.1 Συνάρτηση getTournamentRounds

```
Query : {
    getTournamentRounds : async (obj: any, args: any, context: any, info: any) => {

    const tournamentId = args.tournamentId;
    const winnersRounds = await TournamentMatch.findAll({where: {tournamentId: tournamentId, losersMatchId: null}, order: ['matchId']});
    const winnersRoundsGrouped =  _.chain(winnersRounds).groupBy("round").map((value, key) => ({ title: key, seeds: value })).value()

    const losersRounds = await TournamentMatch.findAll({where: {tournamentId: tournamentId, winnersMatchId: null}, order: ['matchId']});
    const losersRoundsGrouped =
    losersRounds.length > 0 ?
    _.chain(losersRounds).groupBy("round").map((value, key) => ({ title: key, seeds: value })).value()
    : null


    return {
        winnersRounds: winnersRoundsGrouped,
        losersRounds: losersRoundsGrouped
    }

    }
},
```

Εικόνα 4.11: Συνάρτηση getTournamentRounds.

Η παραπάνω συνάρτηση χρησιμοποιείται για την απόκτηση όλων των αγώνων ενός τουρνουά γκρουπαρισμένα ανά τον γύρο στον οποίο βρίσκονται. Η συνάρτηση έχει χτιστεί έτσι ώστε να μην χρειάζονται τροποποίηση τα δεδομένα μόλις φτάσουν στο Front-end και να μπορούν να εισαχθούν κατευθείαν στο Bracket. Ως είσοδο, χρειάζεται μόνο το tournamentId που είναι το αναγνωριστικό id του τουρνουά. Στην έξοδο επιστρέφει το BracketRounds το οποίο αποτελείται από δύο στοιχεία όπως φαίνεται και στην εικόνα 4.11. Το winnersRounds το οποίο περιέχει τους αγώνες του Winners Bracket και το losersRounds το οποίο περιέχει τους αγώνες του Losers Bracket. Το losersRounds περιέχει αγώνες μόνο αν ο τύπος τουρνουά είναι Double Elimination Bracket. Και τα δύο στοιχεία επιστρέφουν έναν πίνακα από στοιχεία Round. Το στοιχείο Round αποτελείται από δύο στοιχεία όπως φαίνεται και στην εικόνα 4.10, το title που προσδιορίζει τον αριθμό γύρου των αγώνων που βρίσκονται στον αντίστοιχο γύρο και το seeds που προσδιορίζει έναν πίνακα από TournamentMatch. Το TournamentMatch είναι το μοντέλο των αγώνων όπως αυτό έχει οριστεί στο αρχείο tournaments_matches.model.ts. Μέσα στη συνάρτηση χρησιμοποιείται η βιβλιοθήκη Lodash που βοηθάει στο γκρουπάρισμα των αγώνων μέσω των συναρτήσεων chain, groupBy, map και value που κατέχει.

4.3.5.2 Συνάρτηση seedTournament

Η συνάρτηση seedTournament χρησιμοποιείται για το seeding του τουρνουά και ταυτόχρονα τη δημιουργία όλων των αγώνων, βάση του αριθμού των συμμετεχόντων και του τύπου τουρνουά. Ως είσοδο, παίρνει το tournamentId που είναι το αγαγνωριστικό id του τουρνουά που ζητείται και το newSeeds που είναι ένας πίνακας από στοιχεία τύπου newSeed. Τα στοιχεία newSeed έχουν τρία ορίσματα. Το seed που είναι ένας ακέραιος αριθμός που δηλώνει το seeding (θέση) ενός παίκτη, το playerName που είναι μια συμβολοσειρά που δηλώνει το όνομα του παίκτη και το newSeed που είναι ένας ακέραιος αριθμός που δηλώνει το καινούργιο seeding ενός παίκτη. Αν δεν έχει οριστεί καινούργιο seeding από το Front-end τότε η τιμή newSeed θα είναι ίδια με την τιμή seed.

Στην εικόνα 4.13 φαίνεται το πρώτο κομμάτι της συνάρτησης seedTournament που είναι η δημιουργία των αγώνων του Winners Bracket. Αρχικά το καινούργιο seeding ενημερώνεται στους συμμετέχοντες του τουρνουά. Ύστερα υπολογίζεται ο μέγιστος αριθμός γύρων που θα έχει το τουρνουά και το αναγνωριστικό id του τελευταίου αγώνα. Για να βρεθούν και οι δύο αυτές τιμές, χρειάζεται πρώτα να βρεθεί το μέγεθος του τουρνουά το οποίο πρέπει να είναι δύναμη του 2 και υπολογίζεται με την βοηθητική συνάρτηση seedBracketSize όπως φαίνεται στην παρακάτω εικόνα.

```
const seedBracketSize = players => {
    let power = 2;
    while (power < players) {
        power *= 2;
    }
    return power;
}
```

Εικόνα 4.12: Βοηθητική συνάρτηση seedBracketSize.

Ο μέγιστος αριθμός γύρων του Winners Bracket ενός τουρνουά ισούται με:

$$\log_2 N \quad (4.1)$$

Ενώ το id του τελευταίου αγώνα ισούται με:

$$N - 1 \quad (4.2)$$

όπου N το μέγεθος του τουρνουά.

```
seedTournament: async (obj: any, args: any, context: any, info: any) => {
    const jwt = jwtToken(context)
    const tournamentId = args.tournamentId;
    const newSeeds = args.newSeeds;
    const tournamentBracketId = (await Tournament.findByPk(tournamentId)).bracketTypeId;
    await Promise.all(newSeeds.map(async newSeed => {
        await TournamentParticipant.update({seed: newSeed.newSeed},{where: {playerName: newSeed.playerName}});
    }))
    const participants = await TournamentParticipant.findAll({where: {tournamentId: tournamentId}, order: ['seed']});

    let playerIndexes = [1,2];
    let round = Math.log(seedBracketSize(participants.length)) / Math.log(2);
    let id = seedBracketSize(participants.length) - 1;
    for (let i = 1 ; i < seedBracketSize(participants.length);) {

        let playerIndex = seedBracketSize(participants.length)-1;
            for (let j = 0; j < i; j++) {
                    await TournamentMatch.create({
                        tournamentId: tournamentId,
                        matchId: id,
                        winnersMatchId: id,
                        round: round,
                        player1Name: round === 1 ? participants[playerIndexes[playerIndex-1]-1]?.playerName : null,
                        player2Name: round === 1 ? participants[playerIndexes[playerIndex]-1]?.playerName : null,
                        player1Seed: round === 1 ? playerIndexes[playerIndex-1] : null,
                        player2Seed: round === 1 ? playerIndexes[playerIndex] : null,
                        player1Id: round === 1 ? participants[playerIndexes[playerIndex-1]-1]?.userId : null,
                        player2Id: round === 1 ? participants[playerIndexes[playerIndex]-1]?.userId : null,
                    })
                    id --;
                    playerIndex -=2;
            }

        const nextIndexes = [];
        const length = playerIndexes.length*2 + 1;
        playerIndexes.forEach((index) => {
            nextIndexes.push(index <= participants.length ? index : null)
            nextIndexes.push((length - index) <= participants.length ? (length - index) : null);
        })

        playerIndexes = [...nextIndexes];
        i = i*2;
        round --;
    }
```

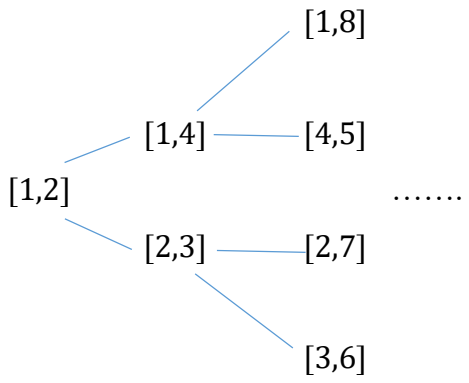
Εικόνα 4.13: Συνάρτηση seedTournament (δημιουργία Winners Bracket).

Στη συνέχεια, ακολουθεί ένας αλγόριθμος που δημιουργεί τους αγώνες αρχίζοντας από τον τελευταίο. Αυτό συμβαίνει για να συμπληρωθούν οι αγώνες με σωστό seeding των παικτών.
Ορίζεται ένας πίνακας με τους αριθμούς [1,2]. Ο κάθε αριθμός του πίνακα μπαίνει σε έναν καινούργιο πίνακα και στο ζεύγος τους εισάγεται ένας αριθμός που ισούται με:

$$N \times 2 + 1 - X \qquad (4.3)$$

Όπου *N* το μέγεθος του αρχικού πίνακα και *X* ο αριθμός του προηγούμενου πίνακα. Μετά τους υπολογισμούς ο πίνακας θα αποτελείται από τους δύο πίνακες [1,4] και [2,3]. Ο αλγόριθμος συνεχίζεται με τον ίδιο τρόπο μέχρι να φτάσει στον 1ο γύρο του τουρνουά.

Η λογική του αλγόριθμου φαίνεται και στην παρακάτω εικόνα.



Εικόνα 4.14: Αλγόριθμος δημιουργίας αγώνων τουρνουά με seeding.

Όταν ο αλγόριθμος φτάσει στον 1ο γύρο του τουρνουά τότε στους αγώνες αυτού του γύρου ορίζονται και οι παίκτες βάση το seeding τους.

Αν το τουρνουά είναι τύπου Double Elimination Bracket και έχει πάνω από δύο συμμετέχοντες τότε η συνάρτηση getTournamentRounds συνεχίζει δημιουργώντας τους αγώνες του Losers Bracket όπως φαίνεται στην εικόνα 4.15.

```
id = seedBracketSize(participants.length)
if (tournamentBracketId === BracketType.DoubleElim && participants.length > 2) {
    await TournamentMatch.create({
        tournamentId: tournamentId,
        matchId: id,
        winnersMatchId: id,
        round: (Math.log(seedBracketSize(participants.length)) / Math.log(2)) + 1,
        player1Name: null,
        player2Name: null,
        player1Seed: null,
        player2Seed: null,
        player1Id: null,
        player2Id: null,
    })
    id ++;
    const loserRounds = ((Math.log(seedBracketSize(participants.length)) / Math.log(2)) - 1) * 2;
    let matches = seedBracketSize(participants.length) / 4;
    let losersMatchId = 1;
    for (let i=1; i<=loserRounds; i++) {
        for (let j=1; j<=matches; j++) {
            await TournamentMatch.create({
                tournamentId: tournamentId,
                matchId: id,
                losersMatchId: losersMatchId,
                round: i,
                player1Name: null,
                player2Name: null,
                player1Seed: null,
                player2Seed: null,
                player1Id: null,
                player2Id: null
            })
            losersMatchId ++;
            id ++;
        }
        if (!(i % 2)) {
            matches /=2;
        }
    }
}

await Tournament.update({stateId: TournamentStates.Seeded}, {where: {id: tournamentId}});

await TournamentsAction.create({
    tournamentId: tournamentId,
    actionDate: new Date(),
    actionId: Action.TournamentState,
    userId: jwt.userId,
    notes: `Tournament seeded`
})

return tournamentId;

},
```

Εικόνα 4.15: Συνάρτηση seedTournament (δημιουργία Losers Bracket).

Αρχικά, χρειάζεται να δημιουργηθεί ο τελευταίος αγώνας του Winners Bracket στον οποίο θα συμμετέχει ο νικητής του Winners Bracket εναντίον του νικητή του Losers Bracket όπως φαίνεται στο κεφάλαιο 2.2.3 και ύστερα οι υπόλοιποι αγώνες του Losers Bracket.

Το id αυτού του αγώνα ισούται με το μέγεθος του τουρνουά και ο γύρος του αγώνα με:

$$(\log_2 N) + 1 \quad (4.4)$$

όπου N το μέγεθος του τουρνουά.

Ο μέγιστος αριθμός γύρων του Losers Bracket υπολογίζεται από την εξίσωση:

$$((\log_2 N) - 1) \times 2 \quad (4.5)$$

Ο αριθμός των αγώνων του πρώτου γύρου του Losers Bracket ισούται με:

$$\frac{N}{4} \quad (4.6)$$

Κάθε δύο γύρους ο αριθμός των αγώνων μειώνεται στο μισό του προηγούμενου γύρου.

Με τους παραπάνω αριθμούς τρέχει ο αλγόριθμος που εμφανίζεται στην εικόνα 4.15 ανά αγώνα και γύρο μέχρι να φτάσει στον τελευταίο αγώνα του τελευταίου γύρου.

Τέλος, ενημερώνεται η κατάσταση του τουρνουά, δημιουργείται μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης και επιστρέφεται το αναγνωριστικό id του τουρνουά.

4.3.5.3 Συνάρτηση startTournament

```
startTournament : async (obj: any, args: any, context: any, info: any) => {
    const jwt = jwtToken(context)
    const tournamentId = args.tournamentId;
    const matches = await TournamentMatch.findAll({where: {tournamentId: tournamentId, round: 1, losersMatchId: null}});
    await Promise.all(matches.map(async match => {
        if (match.player1Seed === null || match.player2Seed === null) {
            await TournamentsExtraResolvers.Mutation.pushMatch(null,{
                tournamentId: tournamentId,
                matchId: match.matchId,
                score1: null,
                score2: null,
                winner: match.player1Seed === null ? 2 : 1,
            }, null, null)
        }
    }))

    await Tournament.update({stateId: TournamentStates.InProgress}, {where: {id: tournamentId}});

    await TournamentsAction.create({
        tournamentId: tournamentId,
        actionDate: new Date(),
        actionId: Action.TournamentState,
        userId: jwt.userId,
        notes: `Tournament started`
    })

    return tournamentId;

    },
```

Εικόνα 4.16: Συνάρτηση startTourament.

Η συνάρτηση startTournament χρησιμοποιείται για την εκκίνηση ενός τουρνουά. Παίρνει ως είσοδο το αναγνωριστικό id του τουρνουά και επιστρέφει ως έξοδο πάλι το id αν δεν βρεθεί σφάλμα. Εξετάζει όλους τους αγώνες του 1^{ου} γύρου του Winners Bracket για τυχόν κενούς παίκτες. Για τον κάθε αγώνα που περιέχει μόνο έναν παίκτη χρησιμοποιείται η συνάρτηση pushMatch με ορισμένο νικητή αυτόν τον παίκτη, με αποτέλεσμα να προχωρήσουν αυτοί οι αγώνες αυτόματα και να μην χρειάζεται χειροκίνητη ενέργεια στο Front-end. Η κατάσταση του τουρνουά ενημερώνεται και δημιουργείται μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης.

4.3.5.4 Συνάρτηση pushMatch

Η συνάρτηση pushMatch χρησιμοποιείται για την αναφορά αποτελέσματος ενός αγώνα του Winners Bracket και προώθηση των παικτών του αγώνα στους επόμενους αγώνες. Στην είσοδο παίρνει πέντε στοιχεία. Το tournamentId που είναι το αγαγνωριστικό id του τουρνουά, το matchId που είναι το αναγνωριστικό id του αγώνα, το score1 που είναι το σκορ του 1ου παίκτη, το score2 που είναι το σκορ του 2ου παίκτη και το winner που είναι ένας ακέραιος αριθμό που δηλώνει τον νικητή του αγώνα, έχοντας τιμή 1 ή 2. Στην έξοδο, η συνάρτηση επιστρέφει το id του αγώνα.

```
pushMatch : async (obj: any, args: any, context: any, info: any) => {
    const jwt = jwtToken(context)
    const tournamentId = args.tournamentId;
    const matchId = args.matchId;
    const participantsLength = (await TournamentParticipant.findAll({where: {tournamentId: tournamentId}})).length
    const tournamentBracketId = (await Tournament.findByPk(tournamentId)).bracketTypeId;
    await TournamentMatch.update(
        {score1: args.score1, score2: args.score2, winner: args.winner },
        {where: { matchId: matchId, tournamentId: tournamentId } }
        ).then(async () => {
           const data = await TournamentMatch.findOne({where: {matchId: matchId, tournamentId: tournamentId}})
            const nextMatchId = seedBracketSize(participantsLength)/2 + round(matchId/2);
            const winnerSeed = data.winner === 1 ? data.player1Seed : data.player2Seed;
            const loserSeed = data.winner === 1 ? data.player2Seed : data.player1Seed;
            const winnerName = data.winner === 1 ? data.player1Name : data.player2Name;
            const loserName = data.winner === 1 ? data.player2Name : data.player1Name;
            const winnerId = data.winner === 1 ? data.player1Id : data.player2Id;
            const loserId = data.winner === 1 ? data.player2Id : data.player1Id;

            if (matchId === seedBracketSize(participantsLength) - 1 && tournamentBracketId === BracketType.DoubleElim) {
                await TournamentMatch.update(
                    {player1Name: winnerName, player1Seed: winnerSeed, player1Id: winnerId },
                    {where: { matchId: matchId + 1, tournamentId: tournamentId }}
                )
                const losersMatch = getLosersMatchId(data.round,data.matchId, seedBracketSize(participantsLength))
                await TournamentMatch.update(
                    losersMatch.isPlayer1 ? {player1Name: loserName, player1Seed: loserSeed, player1Id: loserId } :
                    {player2Name: loserName, player2Seed: loserSeed, player2Id: loserId},
                    {where: { losersMatchId: losersMatch.matchId, tournamentId: tournamentId }}
                )
            } else if (matchId === seedBracketSize(participantsLength) ||
            (tournamentBracketId === BracketType.SingleElim && matchId === (seedBracketSize(participantsLength) - 1) )) {
                await Tournament.update({stateId: TournamentStates.Complete}, {where: {id: tournamentId}});
                await TournamentParticipant.update({placement: 1}, {where: {seed: winnerSeed, tournamentId: tournamentId}})
                await TournamentParticipant.update({placement: 2}, {where: {seed: loserSeed, tournamentId: tournamentId}})
            }  else {
            const isPlayer1 = matchId % 2;
            await TournamentMatch.update(
                isPlayer1 ? {player1Name: winnerName, player1Seed: winnerSeed, player1Id: winnerId } :
                {player2Name: winnerName, player2Seed: winnerSeed, player2Id: winnerId},
                {where: { matchId: nextMatchId, tournamentId: tournamentId }}
            )
```

Εικόνα 4.17: Συνάρτηση pushMatch (1ο μέρος).

Στο 1ο μέρος η συνάρτηση ενημερώνει το αποτέλεσμα του αγώνα και χειρίζεται τον νικητή του αγώνα. Αν το τουρνουά είναι τύπου Double Elimination Bracket και ο αρχικός αγώνας της συνάρτησης είναι ο τελευταίος του Winners Bracket πριν τον τελικό τότε ο νικητής προωθείται στον τελικό και ο ηττημένος στον τελικό του Losers Bracket.

Αν ο αρχικός αγώνας της συνάρτησης είναι ο ίδιος ο τελικός τότε ο νικητής του αγώνα παίρνει την 1η θέση του τουρνουά, ο ηττημένος παίρνει την 2η και ο ενημερώνεται η κατάσταση του αγώνα ως ολοκληρωμένος αγώνας.

Αν δεν ισχύει καμία από τις παραπάνω περιπτώσεις τότε ο νικητής προωθείται στον επόμενο αγώνα. Το id του επόμενου αγώνα βρίσκεται με την εξίσωση:

$$\frac{N}{2} + round\left(\frac{X}{2}\right) \quad (4.7)$$

όπου N το μέγεθος του τουρνουά, X το id του αρχικού αγώνα και *round* η στρογγυλοποίηση.

Αν το τουρνουά είναι τύπου Double Elimination Bracket τότε η συνάρτηση πρέπει να προωθήσει και τον ηττημένο παίκτη στο Losers Bracket. Το id του επόμενου αγώνα του ηττημένου παίκτη υπολογίζεται μέσω της βοηθητικής συνάρτησης getLosersMatchId που φαίνεται στην εικόνα 4.18 και ισούται με:

$$round\left(\frac{X}{2}\right),\ για\ r = 1 \quad (4.8)$$

$$X - 2^{(W-r)},\ για\ r > 1 \quad (4.9)$$

όπου X το id του αρχικού αγώνα, r ο γύρος στον οποίο βρίσκεται ο αρχικός αγώνας, *round* η στρογγυλοποίηση και W ο συνολικός αριθμός των γύρων του Winners Bracket.

```
const getLosersMatchId = (roundId, matchId, bracketSize) => {
    const winnersRounds = Math.log(bracketSize) / Math.log(2);
    switch (roundId) {
        case 1:
            return {
                matchId: round(matchId/2),
                isPlayer1: matchId % 2,
            }
        default:
            return {
                matchId: matchId - Math.pow(2, winnersRounds - roundId),
                isPlayer1: true
            }
    }
}
```

Εικόνα 4.18: Βοηθητική συνάρτηση getLosersMatchId.

```
        if (tournamentBracketId === BracketType.DoubleElim) {
            const losersMatch = getLosersMatchId(data.round,data.matchId, seedBracketSize(participantsLength))
            const otherMatchId = losersMatch.isPlayer1 ? matchId + 1 : matchId - 1;
            const otherMatch = await TournamentMatch.findOne({where: {matchId: otherMatchId, tournamentId: tournamentId}})

                if (participantsLength < seedBracketSize(participantsLength) &&
                data.round === 1 &&
                (otherMatch.player1Name === null || otherMatch.player2Name === null)) {
                    await TournamentMatch.update(
                        losersMatch.isPlayer1 ? {player1Name: loserName, player1Seed: loserSeed, player1Id: loserId } :
                        {player2Name: loserName, player2Seed: loserSeed, player2Id: loserId},
                        {where: { losersMatchId: losersMatch.matchId, tournamentId: tournamentId }}
                    )
                    await TournamentsExtraResolvers.Mutation.pushLosersMatch(null,{
                        tournamentId: tournamentId,
                        losersMatchId: losersMatch.matchId,
                        score1: null,
                        score2: null,
                        winner: losersMatch.isPlayer1 ? 1 : 2,
                    }, null, null)
                    }
                else {
                    await TournamentMatch.update(
                        losersMatch.isPlayer1 ? {player1Name: loserName, player1Seed: loserSeed, player1Id: loserId } :
                        {player2Name: loserName, player2Seed: loserSeed, player2Id: loserId},
                        {where: { losersMatchId: losersMatch.matchId, tournamentId: tournamentId }}
                    )
                }

            } else {
                const placement = (seedBracketSize(participantsLength) / (Math.pow(2,data.round))) + 1;
                await TournamentParticipant.update({placement: placement}, {where: {seed: loserSeed, tournamentId: tournamentId}});
            }

            }
            const notes = args.score1 === null && args.score2 === null ?
                        `Match ${matchId} auto complete for ${data?.player1Name ?? 'No Opponent'} vs ${data?.player2Name ?? 'No Opponent'}` :
                        `Match ${matchId} complete with a ${args.score1} - ${args.score2} score for ${data?.player1Name} vs ${data?.player2Name} `
            await TournamentsAction.create({
                tournamentId: tournamentId,
                actionDate: new Date(),
                actionId: Action.MatchComplete,
                userId: jwt.userId,
                notes: notes,
            })
            if (nextMatchId >= seedBracketSize(participantsLength)) {
                await TournamentsAction.create({
                    tournamentId: tournamentId,
                    actionDate: new Date(),
                    actionId: Action.TournamentState,
                    userId: jwt.userId,
                    notes: `Tournament finished`
                })
            }
        })
    return matchId;
},
```

Εικόνα 4.19: Συνάρτηση pushMatch (2ο μέρος).

Ο επόμενος αγώνας του ηττημένου παίκτη υπάρχει περίπτωση να μην έχει αντίπαλο λόγω των αρχικών συμμετεχόντων. Αυτό συμβαίνει αν ο αριθμός των συμμετεχόντων δεν είναι δύναμη του 2. Η συνάρτηση το ελέγχει αυτό, όπως φαίνεται στην εικόνα 4.19, κοιτάζοντας τον άλλο αγώνα από τον οποίο προωθείται ο ηττημένος. Αν ο άλλος αγώνας έχει τουλάχιστον έναν κενό παίκτη τότε ο ηττημένος του αρχικού αγώνα της συνάρτησης pushMatch νικάει αυτόματα τον επόμενο του αγώνα στο Losers Bracket και προωθείται ακόμη μια φορά μέσω της συνάρτησης pushLosersMatch.

Αντίθετα, ο ηττημένος απλώς προωθείται στον επόμενο του αγώνα του LosersBracket.

Στην περίπτωση που το τουρνουά δεν είναι τύπου Double Elimination Bracket, ο ηττημένος του αρχικού αγώνα δεν έχει άλλη προσπάθεια και αποκλείεται από το τουρνουά. Σε αυτήν την περίπτωση πρέπει να οριστεί η τελική θέση του παίκτη στο τουρνουά η οποία ισούτε με:

$$\left(\frac{N}{2^r}\right) + 1 \qquad (4.10)$$

όπου N το μέγεθος του τουρνουά και r ο γύρος στον οποίο βρίσκεται ο αρχικός αγώνας.

Η συνάρτηση τελειώνει δημιουργώντας μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης. Αν ήταν ο τελικός τότε δημιουργείται άλλη μια εγγραφή TournamentsActions για την καταγραφή της ολοκλήρωσης του τουρνουά.

4.3.5.5 Συνάρτηση pushLosersMatch

Η συνάρτηση pushLosersMatch χρησιμοποιείται για την αναφορά αποτελέσματος ενός αγώνα του Losers Bracket και προώθηση του νικητή του αγώνα στον επόμενο. Στην είσοδο παίρνει πέντε στοιχεία. Το tournamentId που είναι το αγαγνωριστικό id του τουρνουά, το losersMatchId που είναι το αναγνωριστικό losersMatchId του αγώνα, το score1 που είναι το σκορ του 1ου παίκτη, το score2 που είναι το σκορ του 2ου παίκτη και το winner που είναι ένας ακέραιος αριθμό που δηλώνει τον νικητή του αγώνα, έχοντας τιμή 1 ή 2. Στην έξοδο, η συνάρτηση επιστρέφει το losersMatchId του αγώνα.

```
pushLosersMatch : async (obj: any, args: any, context: any, info: any) => {
    const jwt = jwtToken(context)
    const tournamentId = args.tournamentId;
    const losersMatchId = args.losersMatchId;
    const participantsLength = (await TournamentParticipant.findAll({where: {tournamentId: tournamentId}})).length
    await TournamentMatch.update(
        {score1: args.score1, score2: args.score2, winner: args.winner },
        {where: { losersMatchId: losersMatchId, tournamentId: tournamentId } }
        )
           const data = await TournamentMatch.findOne({where: {losersMatchId: losersMatchId, tournamentId: tournamentId}})
           const maxLosersMatchId = await TournamentMatch.max('losersMatchId', {where: {tournamentId: tournamentId}});
            const winnerSeed = data.winner === 1 ? data.player1Seed : data.player2Seed;
            const loserSeed = data.winner === 1 ? data.player2Seed : data.player1Seed;
            const winnerName = data.winner === 1 ? data.player1Name : data.player2Name;
            const winnerId = data.winner === 1 ? data.player1Id : data.player2Id;
            if (losersMatchId === maxLosersMatchId) {
                await TournamentMatch.update(
                    {player2Name: winnerName, player2Seed: winnerSeed, player2Id: winnerId },
                    {where: { matchId: seedBracketSize(participantsLength), tournamentId: tournamentId }}
                )
                await TournamentParticipant.update({placement: 3}, {where: {seed: loserSeed, tournamentId: tournamentId}});
            } else {
            const matchesPerRound = await TournamentMatch.count({where: {tournamentId: tournamentId, round: data.round, winnersMatchId: null}});
            const nextMatchId = data.round % 2 ?
                                losersMatchId + matchesPerRound :
                                (seedBracketSize(participantsLength)/2) - (matchesPerRound/2) + round(losersMatchId/2);

            const isPlayer1 = data.round % 2 ?
                              false :
                              losersMatchId % 2;
            await TournamentMatch.update(
                isPlayer1 ? {player1Name: winnerName, player1Seed: winnerSeed, player1Id: winnerId } :
                {player2Name: winnerName, player2Seed: winnerSeed, player2Id: winnerId},
                {where: { losersMatchId: nextMatchId, tournamentId: tournamentId }}
            )
                const loserRounds = ((Math.log(seedBracketSize(participantsLength)) / Math.log(2)) - 1) * 2;
                const multiplier = (loserRounds - data.round) % 2 ? 3 : 2
                const placement =  (matchesPerRound * multiplier) + 1;
                await TournamentParticipant.update({placement: placement}, {where: {seed: loserSeed, tournamentId: tournamentId}});
            }

            const notes = args.score1 === null && args.score2 === null ?
                        `Match ${data.matchId} auto complete for ${data?.player1Name ?? 'No Opponent'} vs ${data?.player2Name ?? 'No Opponent'}` :
                        `Match ${data.matchId} complete with a ${args.score1} - ${args.score2} score for ${data?.player1Name} vs ${data?.player2Name} `
            await TournamentsAction.create({
                tournamentId: tournamentId,
                actionDate: new Date(),
                actionId: Action.MatchComplete,
                userId: jwt.userId,
                notes: notes,
            })
    return losersMatchId;
},
```

Εικόνα 4.20: Συνάρτηση pushLosersMatch.

Η συνάρτηση ενημερώνει το αποτέλεσμα του αγώνα και αρχικά χειρίζεται τον νικητή. Αν ο αγώνας είναι ο τελευταίος αγώνας του Losers Bracket τότε ο νικητής προωθείται στον τελικό όλου του τουρνουά και ενημερώνεται η τελική θέση σε αυτόν τον τουρνουά του ηττημένου ως 3η θέση. Για οποιαδήποτε άλλη περίπτωση υπολογίζεται ο επόμενος αγώνας του Losers Bracket που θα συμμετάσχει ο νικητής.

Το Losers Bracket id του επόμενου αγώνα υπολογίζεται ως εξής:

$$X + M,\ για\ r\ περιττό\ αριθμό \quad (4.11)$$

$$\left(\frac{N}{2}\right) - \left(\frac{M}{2}\right) + round\left(\frac{X}{2}\right),\ για\ r\ άρτιο\ αριθμό \quad (4.12)$$

όπου *X* το Losers Bracket id του αρχικού αγώνα, *N* το μέγεθος του τουρνουά, *r* ο γύρος στον οποίο βρίσκεται ο αρχικός αγώνας, *M* ο αριθμός των αγώνων σε αυτόν τον γύρο και *round* η στρογγυλοποίηση.

Εφόσον ενημερωθεί ο επόμενος αγώνας του νικητή, πρέπει να προσδιοριστεί η τελική θέση του ηττημένου επειδή δεν κατέχει άλλη προσπάθεια και είναι αποκλεισμένος από το τουρνουά. Η θέση αυτή υπολογίζεται ως εξής:

$$(M * 3) + 1,\ για\ (((\log_2 N) - 1) * 2) - r = περιττό\ αριθμό \quad (4.13)$$

$$(M * 2) + 1,\ για\ (((\log_2 N) - 1) * 2) - r = άρτιο\ αριθμό \quad (4.14)$$

όπου *M* ο αριθμός των αγώνων σε αυτόν τον γύρο, *N* το μέγεθος του τουρνουά και *r* ο γύρος στον οποίο βρίσκεται ο αρχικός αγώνας.

Αφού ενημερωθεί η τελική θέση του ηττημένου, η συνάρτηση τελειώνει δημιουργώντας μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης.

4.3.5.6 Συνάρτηση undoMatch

Η συνάρτηση undoMatch χρησιμοποιείται για την αναίρεση ενός αγώνα του Winners Bracket. Στην είσοδο παίρνει το tournamentId που είναι το αναγνωριστικό id του τουρνουά και το matchId το οποίο είναι το αναγνωριστικό id του αγώνα. Ως έξοδο, επιστρέφει το id του αγώνα.

```
undoMatch : async (obj: any, args: any, context: any, info: any) => {
        const jwt = jwtToken(context)
        const tournamentId = args.tournamentId;
        const matchId = args.matchId;
        const isPlayer1 = matchId % 2;
        const tournamentBracketId = (await Tournament.findByPk(tournamentId)).bracketTypeId;
        const participantsLength = (await TournamentParticipant.findAll({where: {tournamentId: tournamentId}})).length
        const nextMatchId = seedBracketSize(participantsLength)/2 + round(matchId/2);

        await TournamentMatch.update(
            {score1: null, score2: null, winner: null },
            {where: { matchId: matchId, tournamentId: tournamentId } }
            )
                const data = await TournamentMatch.findOne({where: {matchId: matchId, tournamentId: tournamentId}})
                await TournamentMatch.update(
                    isPlayer1 ? {player1Name: null, player1Seed: null, player1Id: null} :
                    {player2Name: null, player2Seed: null, player2Id: null},
                    {where: { matchId: nextMatchId, tournamentId: tournamentId }}
                )
                await TournamentsAction.create({
                    tournamentId: tournamentId,
                    actionDate: new Date(),
                    actionId: Action.MatchUndo,
                    userId: jwt.userId,
                    notes: `Match ${matchId} between ${data.player1Name} vs ${data.player2Name} undone `
                })
                if (tournamentBracketId === BracketType.DoubleElim) {
                    const losersMatch = getLosersMatchId(data.round,data.matchId, seedBracketSize(participantsLength))
                    const matchCheckUp = await TournamentMatch.findOne({where: {losersMatchId: losersMatch.matchId, tournamentId: tournamentId}})
                    const isAutoPushLosers = matchCheckUp.winner !== null && matchCheckUp.score1 === null && matchCheckUp.score2 === null;
                    if (isAutoPushLosers) {
                        await TournamentsExtraResolvers.Mutation.undoLosersMatch(null,{
                            tournamentId: tournamentId,
                            losersMatchId: losersMatch.matchId,
                        }, null, null)
                    }
                        await TournamentMatch.update(
                            losersMatch.isPlayer1 ? {player1Name: null, player1Seed: null, player1Id: null } :
                            {player2Name: null, player2Seed: null, player2Id: null},
                            {where: { losersMatchId: losersMatch.matchId, tournamentId: tournamentId }}
                        )
                }
        return matchId;
},
```

Εικόνα 4.21: Συνάρτηση undoMatch.

Αρχικά, αναιρείται το σκορ του αγώνα και μετά ελέγχεται ο επόμενος αγώνας στον οποίο προωθήθηκε ο νικητής. Το id του επόμενου αγώνα βρίσκεται με την εξίσωση:

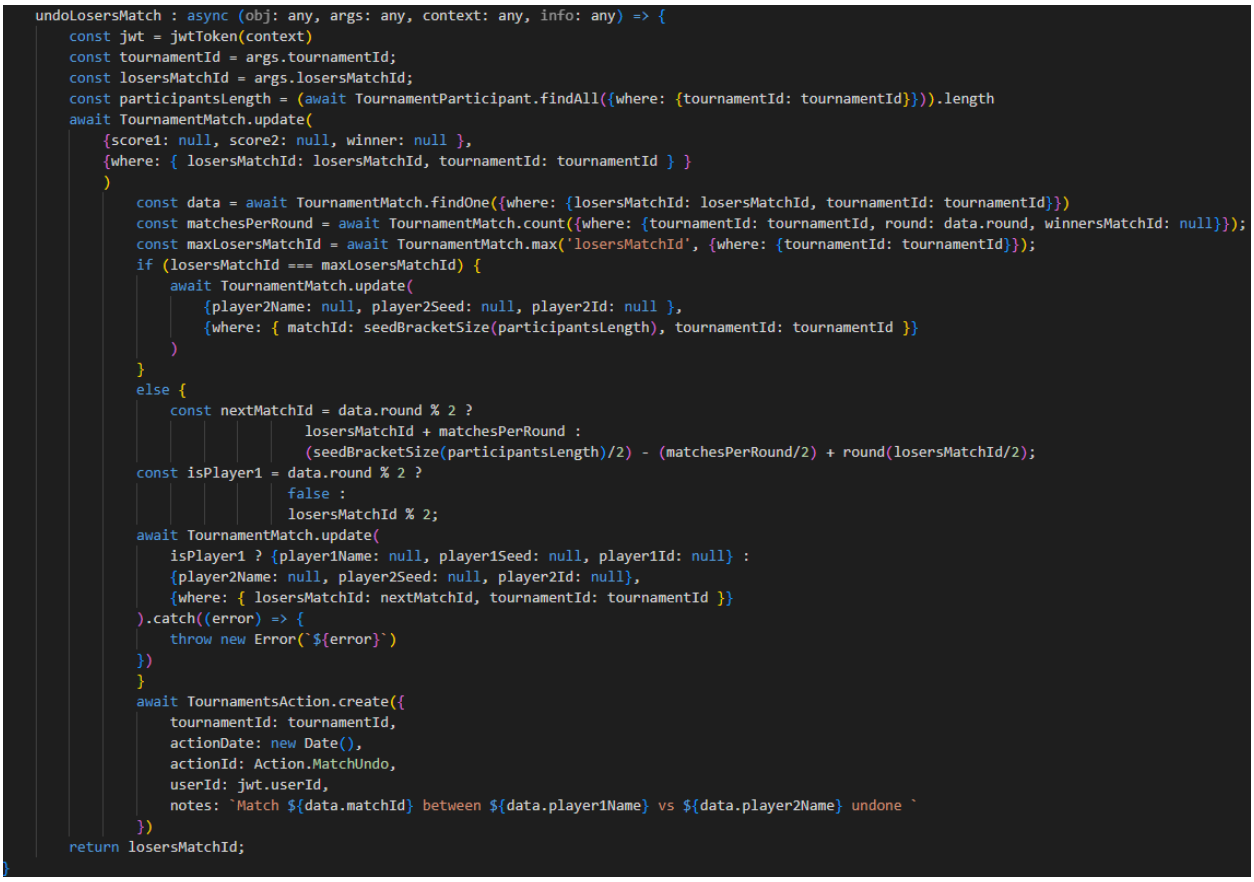
$$\frac{N}{2} + round(X/2) \qquad (4.15)$$

όπου N το μέγεθος του τουρνουά, X το id του αρχικού αγώνα και round η στρογγυλοποίηση.

Αφαιρείται ο νικητής από αυτόν τον αγώνα και δημιουργείται μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης.
Αν ο τύπος του τουρνουά είναι Double Elimination Bracket τότε υπολογίζεται και ο επόμενος αγώνας στον οποίο προωθήθηκε ο ηττημένος μέσω της βοηθητικής συνάρτησης getLosersMatchId. Αν αυτός ο αγώνας δεν είχε αντίπαλο με αποτέλεσμα ο ηττημένος να έχει προωθηθεί δύο αγώνες μπροστά τότε αναιρείται και καλείται πάλι η συνάρτηση undoLosersMatch για να αναιρέσει και τον επόμενο αγώνα. Αλλιώς, αναιρείται μόνο ο $1^{ος}$ αγώνας που προωθήθηκε ο ηττημένος.

4.3.5.7 Συνάρτηση undoLosersMatch

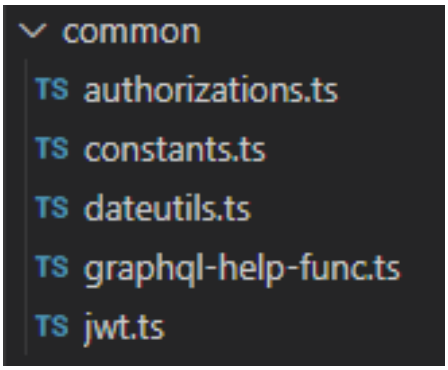
Η συνάρτηση undoLosersMatch χρησιμοποιείται για την αναίρεση ενός αγώνα του Losers Bracket. Στην είσοδο παίρνει το tournamentId που είναι το αναγνωριστικό id του τουρνουά και το losersMatchId το οποίο είναι το αναγνωριστικό losersMatchId του αγώνα. Ως έξοδο, επιστρέφει το losersMatchId του αγώνα. Αρχικά, αναιρείται το σκορ του αγώνα.
Αν ο αγώνας ήταν ο τελευταίος αγώνας του Losers Bracket τότε αφαιρείται ο νικητής από τον τελικό του τουρνουά στον οποίο προωθήθηκε. Σε κάθε άλλη περίπτωση ελέγχεται ο επόμενος αγώνας στον οποίο προωθήθηκε ο νικητής. Το Losers Bracket id του επόμενου αγώνα υπολογίζεται με τις εξισώσεις (4.11) και (4.12).
Αφαιρείται ο νικητής από αυτόν τον αγώνα και δημιουργείται μια εγγραφή TournamentsAction για την καταγραφή αυτής της κίνησης.



```
undoLosersMatch : async (obj: any, args: any, context: any, info: any) => {
    const jwt = jwtToken(context)
    const tournamentId = args.tournamentId;
    const losersMatchId = args.losersMatchId;
    const participantsLength = (await TournamentParticipant.findAll({where: {tournamentId: tournamentId}})).length
    await TournamentMatch.update(
        {score1: null, score2: null, winner: null },
        {where: { losersMatchId: losersMatchId, tournamentId: tournamentId } }
        )
            const data = await TournamentMatch.findOne({where: {losersMatchId: losersMatchId, tournamentId: tournamentId}})
            const matchesPerRound = await TournamentMatch.count({where: {tournamentId: tournamentId, round: data.round, winnersMatchId: null}});
            const maxLosersMatchId = await TournamentMatch.max('losersMatchId', {where: {tournamentId: tournamentId}});
            if (losersMatchId === maxLosersMatchId) {
                await TournamentMatch.update(
                    {player2Name: null, player2Seed: null, player2Id: null },
                    {where: { matchId: seedBracketSize(participantsLength), tournamentId: tournamentId }}
                )
            }
            else {
                const nextMatchId = data.round % 2 ?
                                    losersMatchId + matchesPerRound :
                                    (seedBracketSize(participantsLength)/2) - (matchesPerRound/2) + round(losersMatchId/2);
            const isPlayer1 = data.round % 2 ?
                              false :
                              losersMatchId % 2;
            await TournamentMatch.update(
                isPlayer1 ? {player1Name: null, player1Seed: null, player1Id: null} :
                {player2Name: null, player2Seed: null, player2Id: null},
                {where: { losersMatchId: nextMatchId, tournamentId: tournamentId }}
            ).catch((error) => {
                throw new Error(`${error}`)
            })
            }
            await TournamentsAction.create({
                tournamentId: tournamentId,
                actionDate: new Date(),
                actionId: Action.MatchUndo,
                userId: jwt.userId,
                notes: `Match ${data.matchId} between ${data.player1Name} vs ${data.player2Name} undone `
            })
    return losersMatchId;
}
```

Εικόνα 4.22: Συνάρτηση undoLosersMatch.

4.3.6 Φάκελος common



Εικόνα 4.23: Φάκελος common.

Ο φάκελος common περιέχει αρχεία που είναι απλώς βοηθητικά, με συναρτήσεις ή σταθερά στοιχεία που χρησιμοποιούνται σε διάφορα σημεία του openbracket-api.

- authorization.ts: Ένα αρχείο που συμπεριλαμβάνει συναρτήσεις που μέσω του jwt επιστρέφουν το όνομα ή το id του συνδεδεμένου χρήστη.

- constants.ts: Ένα αρχείο που συμπεριλαμβάνει σταθερά στοιχεία. Για παράδειγμα, τα id των καταστάσεων που μπορεί να πάρει ένα τουρνουά με τις περιγραφές τους. Τα στοιχεία αυτά βρίσκονται και στη βάση δεδομένων αλλά επειδή δεν είναι στοιχεία που αλλάζουν συχνά και ταυτόχρονα χρησιμοποιούνται αρκετά συχνά στις κύριες συναρτήσεις του API, είναι προτιμότερο να εξάγονται από αυτό το αρχείο και όχι να γίνεται κλήση στη βάση δεδομένων κάθε φορά που χρειάζοναι.

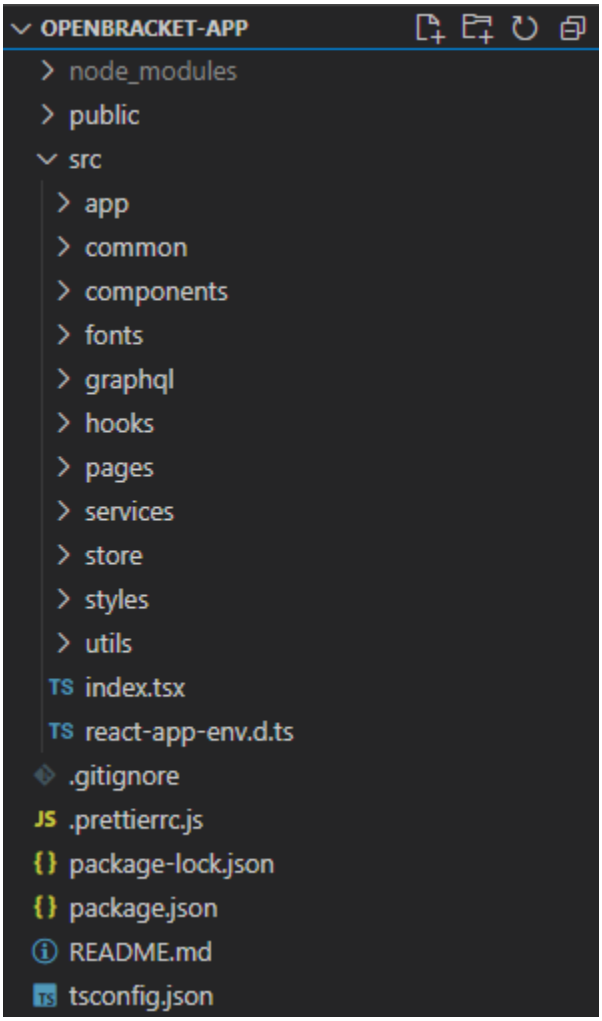
- dateutils.ts: Ένα αρχείο που συμπεριλαμβάνει βοηθητικές συναρτήσεις μετατροπής ημερομηνιών.

- graphql-help-func: Ένα αρχείο που συμπεριλαμβάνει συναρτήσεις που εκτελούν ενέργειες CRUD και χρησιμοποιούνται σε όλα τα μοντέλα του φακέλου graphql.

- jwt: Ένα αρχείο που συμπεριλαμβάνει τις συναρτήσεις που προσφέρει το πακέτο jwt.

4.4 Ανάλυση Front-end

Το Front-end του συστήματος έχει αναπτυχθεί με React και Typescript, όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο. Τα αρχεία του openbracket-app είναι χωρισμένα ανά κατηγορίες σε φάκελους για την εύκολη μετάβαση και ανάγνωση του κώδικα, όπως φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 4.24: Δομή openbracket-app.

Στα επόμενα υποκεφάλαια ακολουθεί η ανάλυση των κύριων αρχείων της εφαρμογής, της δρομολόγησης μεταξύ των σελιδών, των κλήσεων προς το API και των πεδίων εισαγωγής στοιχείων μέσω φόρμας.

4.4.1 Κύρια αρχεία app

Η εφαρμογή όπως όλες οι διαδικτυακές εφαρμογές ανοίγει από το index.html αρχείο το οποίο περιέχει μια div ετικέτα με id root.

```
    <title>Open Bracket</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

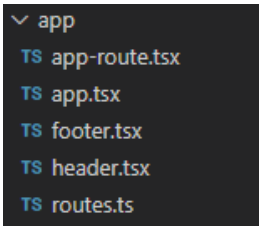
Εικόνα 4.25: Αρχείο index.html.

Μέσα στο index.tsx αρχείο του φακέλου app εισάγεται το React το οποίο παρέχει το component App στο HTML στοιχείο με id root.

```
import React from 'react';
import ReactDOM from 'react-dom';

import './styles/index.scss';
import App from './app/app';

ReactDOM.render(<App />, document.getElementById('root'));
```

Εικόνα 4.26: Αρχείο index.tsx.

Το App εξάγεται από τον φάκελο app της εικόνας που ακολουθεί.



Εικόνα 4.27: Φάκελος app.

4.4.2 Δρομολόγηση σελιδών

Το βασικό κομμάτι του App είναι ότι ελέγχει συνέχεια τη διεύθυνση στην οποία βρίσκεται η εφαρμογή και την δρομολογεί αντίστοιχα για να εμφανίσει τη σωστή σελίδα. Αυτό το σύστημα δουλεύει μέσω του αρχείου app-route.tsx με την τεχνολογία React Router.

```
<Switch>
  <Route path={'/login'}>
    {' '}
    <LoginPage />{' '}
  </Route>

  <Route path={'/signup'}>
    {' '}
    <SignUpPage />{' '}
  </Route>

  {routes.map((route) => (
    <PrivateRoute
      key={shortid.generate()}
      path={route.path}
      component={route.component}
      exact
      strict
    />
  ))}
  <Route render={() => <Redirect to={'/'}></Redirect>} />
</Switch>
```

Εικόνα 4.28: Αρχείο app-route.tsx.

Οι κοινές σελίδες που είναι ανοιχτές στους μη συνδεδεμένους χρήστες είναι οι /login και /signup ή αντίστοιχα οι σελίδες σύνδεσης και εγγραφής. Όλες οι υπόλοιπες σελίδες θεωρούνται ιδιωτικές και δεν έχουν πρόσβαση οι χρήστες χωρίς να είναι συνδεδεμένοι. Οποιαδήποτε δοκιμή για κάποια ιδιωτική δρομολόγηση θα τους ανακατευθύνει στο "/" το οποίο δείχνει τη σελίδα σύνδεσης.

4.4.3 Κλήσεις API & React Hooks

Στο προηγούμενο κεφάλαιο έχει αναφερθεί ότι το API ανοίγει έναν server που ακούει σε μια ορισμένη πόρτα. Η πόρτα που έχει επιλεχθεί είναι η 5000 και εφόσον η εφαρμογή τρέχει στη διεύθυνση "http://localhost", η τελική διεύθυνση του API είναι η "http://localhost:5000" και ορίζεται στην εκκίνηση του Front-end, όπως φαίνεται στην εικόνα που ακολουθεί.

```
"scripts": {
  "start": "npm run && cross-env PORT=3010 REACT_APP_WEBAPI='http://localhost:5000' REACT_APP_CONSOLE='1' react-scripts start",
  "build": "react-scripts build",
  "update-ver": "npx ./make-git-num.js"
},
```

Εικόνα 4.29: Script εκκίνησης Front-end.

Έχει επίσης αναφερθεί ότι το API χρησιμοποιεί τη τεχνολογία GraphQL η οποία προσφέρει μια εύκολη και κατανοητή σύνταξη των συναρτήσεών της. Οι συναρτήσεις αυτές ορίζονται σε διάφορα graphql.tsx αρχεία που βρίσκονται σε φακέλους σελιδών του openbracket-app. Για τον ορισμό των συναρτήσεων χρησιμοποιείται το πακέτο "graphql-tag" το οποίο μεταφράζει τα Javascript strings σε GraphQL AST(Abstract Syntax Tree).

```
import gql from 'graphql-tag';

export const gqlAddTournament = gql`
  mutation addTournament($data: TournamentInput) {
    addTournament(data: $data) {
      id
      url
    }
  }
`;

export const gqlUpdateTournament = gql`
  mutation updateTournament($data: TournamentInput) {
    updateTournament(data: $data) {
      id
    }
  }
`;
```

Εικόνα 4.30: Σύνταξη GraphQL συναρτήσεων.

Οι δύο παραπάνω συναρτήσεις είναι και οι δύο για το μοντέλο Tournament του API και χρησιμοποιούνται για την δημιουργία και ενημέρωση ενός τουρνουά. Και οι δύο έχουν την ίδια είσοδο, το TournamentInput, αλλά η addTournament επιστρέφει τα πεδία id και url ενώ η updateTournament μόνο το πεδίο id.
Αυτές οι συναρτήσεις εξάγονται από το graphql.tsx αρχείο και εκτελούνται μέσω React Hooks. Είναι μια νέα προσθήκη της React με την έκδοση 16.8 η οποία επιτρέπει τη χρήση κατάστασης (state) και των δυνατήτων της, χωρίς κλάσεις αλλά με functional components.

```
const [addRecord] = useMutation(gqlAddTournament);
```

Εικόνα 4.31: useMutation Hook.

Το παραπάνω hook, useMutation, εισάγεται από το πακέτο "@apollo/react-hooks" και χρησιμοποιείται για να εκτελεστεί μια ασύγχρονη εντολή η οποία θα επιστρέψει δεδομένα. Η εντολή στην προκειμένη περίπτωση, ονομάζεται addRecord και χρησιμοποιείται στην φόρμα της σελίδας δημιουργίας ενός καινούργιου τουρνουά.

```
const handleSubmit = (values: FormValuesType, actions) => {
  actions.setSubmitting(false);

  console.log('[Tournament] New:', values);
  addRecord({ variables: { data: { ...values } } })
    .then((d) => {
      toastSuccess(`Tournament created successfully [${d.data.addTournament.id}]`);
      history.push(
        generatePath(getRouteByName('TournamentPage'), { url: d.data.addTournament.url }),
      );
    })
    .catch((error) => {
      if (error.toString().includes('Validation error')) {
        toastWarning('The URL is already in use');
      } else {
        handleGraphqlError('mutation-add', error);
      }
    });
};
```

Εικόνα 4.32: Χρήση εντολής addRecord για καινούργιο τουρνουά.

Η εντολή addRecord παίρνει στην είσοδο τα στοιχεία της φόρμας και καλεί το hook. Βάση της απάντησης που θα λάβει συνεχίζει αντίστοιχα. Αν η εντολή εκτελεστεί επιτυχώς, τότε εμφανίζεται ένα ενημερωτικό μήνυμα προς τον χρήστη και κατευθύνεται στη σελίδα του τουρνουά μέσω του στοιχείου url που επιστρέφει η συνάρτηση. Αν όμως η εντολή επιστρέψει σφάλμα τότε εμφανίζεται αντίστοιχο μήνυμα που περιγράφει πως προέκυψε το σφάλμα.

4.4.4 Φόρμες και πεδία εισαγωγής

Η εντολή addRecord που αναφέρθηκε προηγουμένως, βρίσκεται στο αρχείο new-tournament.tsx. Ένα αρχείο που περιέχει μια φόρμα με τα απαραίτητα πεδία εισαγωγής των στοιχείων ενός καινούργιου τουρνουά. Τα κύρια πακέτα του αρχείου είναι τα yup, formik και react-bootstrap.

```
const TournamentForm = () => {
  const { values} = useFormikContext<any>();

  return (
    <>
      <FieldInput required label="Tournament name" name="name" placeholder="" />

      <FieldInput required label="Game" name="game" placeholder="" />
      <div className="py-3">
        <FieldLabel text={'Description'} required={true} />
        <Field name="description">
          {(fieldProps: FieldProps<string>) => (
            <ReactQuill
              theme="snow"
              value={!!values.description ? fieldProps.field.value : ''}
              onChange={fieldProps.field.onChange(fieldProps.field.name)}
            />
          )}
        </Field>
      </div>
      <FieldSelectBracketTypes required label="Format" name="bracketTypeId" />
      <FieldNumericInput label="Max Size" name="maxSize" placeholder="" />
      <FieldInput required label="URL" name="url" placeholder="" />
      <FieldDatepicker label="Start Time" name="startDate" />
      <FieldDatepicker label="Sign up End" name="signupEndDate" />
    </>
  );
};
```

Εικόνα 4.33: Φόρμα εισαγωγής στοιχείων καινούργιου τουρνουά.

Στην παραπάνω φόρμα χρησιμοποιούνται διάφορα είδη πεδίων τα οποία παίρνουν την εμφάνιση τους μέσω των εξαρτημάτων που προσφέρει η react-bootstrap και τη λειτουργία τους μέσω του formik το οποίο κρατάει ένα ολοκληρωμένο state που περικλίει τη φόρμα και ενημερώνει για τυχόν αλλαγές στα πεδία.

- FieldInput: Χρησιμοποιείται για εισαγωγή συμβολοσειράς. Δέχεται το όνομα του πεδίου, το όνομα της ετικέτας, αν είναι υποχρεωτικό και σύμβολο υποκατάστασης.

- ReactQuill: Ένα πεδίο που εισάγεται μέσο του πακέτου react-quill και προσφέρει προχωρημένες ρυθμίσεις εισαγωγής κειμένου.

- FieldSelectBracketTypes: Ένα πεδίο που χρησιμοποιεί το πακέτο react-select για την εμφάνιση ενός πεδίου με πολλαπλές προκαθορισμένες επιλογές. Στην προκειμένη περίπτωση, το πεδίο δέχεται το όνομα bracketTypeId και μέσω μιας συνάρτησης GraphQL τραβάει από το API όλους τους τύπους τουρνουά για να τους εμφανίσει ως επιλογές. Εκτός από το όνομα του πεδίου, δέχεται το όνομα της ετικέτας και το αν είναι υποχρεωτικό.

- FieldNumericInput: Χρησιμοποιείται για την εισαγωγή ακέραιου αριθμού. Δέχεται το όνομα του πεδίου, το όνομα της ετικέτας, αν είναι υποχρεωτικό και σύμβολο υποκατάστασης.

- FieldDatepicker: Χρησιμοποιείται για την εισαγωγή ημερομηνίας μέσω του πακέτου react-datepicker το οποίο προσφέρει έναν εύκολο τρόπο επιλογής ημερομηνίας μέσω ενός ημερολογίου ή χειροκίνητη εισαγωγή ψηφίων.

Το formik εκτός από ενημερώσεις αλλαγών στα στοιχεία της φόρμας, ενημερώνει και για λάθη, όπως η μη συμπλήρωση υποχρεωτικού πεδίου. Ο έλεγχος αυτός γίνεται μέσω του yup.

```
const formShape = {
  name: YupStringReq,
  game: YupStringReq,
  url: YupStringReq,
  maxSize: YupNumericReq,
  description: YupString,
  startDate: YupDate,
  signupEndDate: YupDate,
  bracketTypeId: YupNumericReq,
  hostId: YupNumeric,
};
```

Εικόνα 4.34: Φόρμα εισαγωγής στοιχείων καινούργιου τουρνουά

- name: Ο τίτλος του τουρνουά. Το YupStringReq προσδιορίζει ότι είναι υποχρεωτικό και πρέπει να είναι συμβολοσειρά.

- game: Το όνομα του παιχνιδιού του τουρνουά. Το YupStringReq προσδιορίζει ότι είναι υποχρεωτικό και πρέπει να είναι συμβολοσειρά.

- url: Η διεύθυνση της ιστοσελίδας του τουρνουά. Το YupStringReq προσδιορίζει ότι είναι υποχρεωτικό και πρέπει να είναι συμβολοσειρά.

- maxSize: Ο μέγιστος αριθμός συμμετεχόντων του τουρνουά. Το YupNumericReq προσδιορίζει ότι είναι υποχρεωτικό και πρέπει να είναι ακέραιος αριθμός.

- description: Η περιγραφή του τουρουά. Το YupString προσδιορίζει ότι είναι προαιρετικό αλλά πρέπει να είναι συμβολοσειρά.

- startDate: Η ημερομηνία έναρξης του τουρνουά. Το YupDate προσδιορίζει ότι είναι προαιρετικό αλλά πρέπει να είναι ημερομηνία.

- signupEndDate: Η ημερομηνία λήξης των δηλώσεων συμμετοχής. Το YupDate προσδιορίζει ότι είναι προαιρετικό αλλά πρέπει να είναι ημερομηνία.

- bracketTypeId: Ο τύπος του τουρνουά: Το YupNumericReq προσδιορίζει ότι είναι υποχρεωτικό και πρέπει να είναι ακέραιος αριθμός.

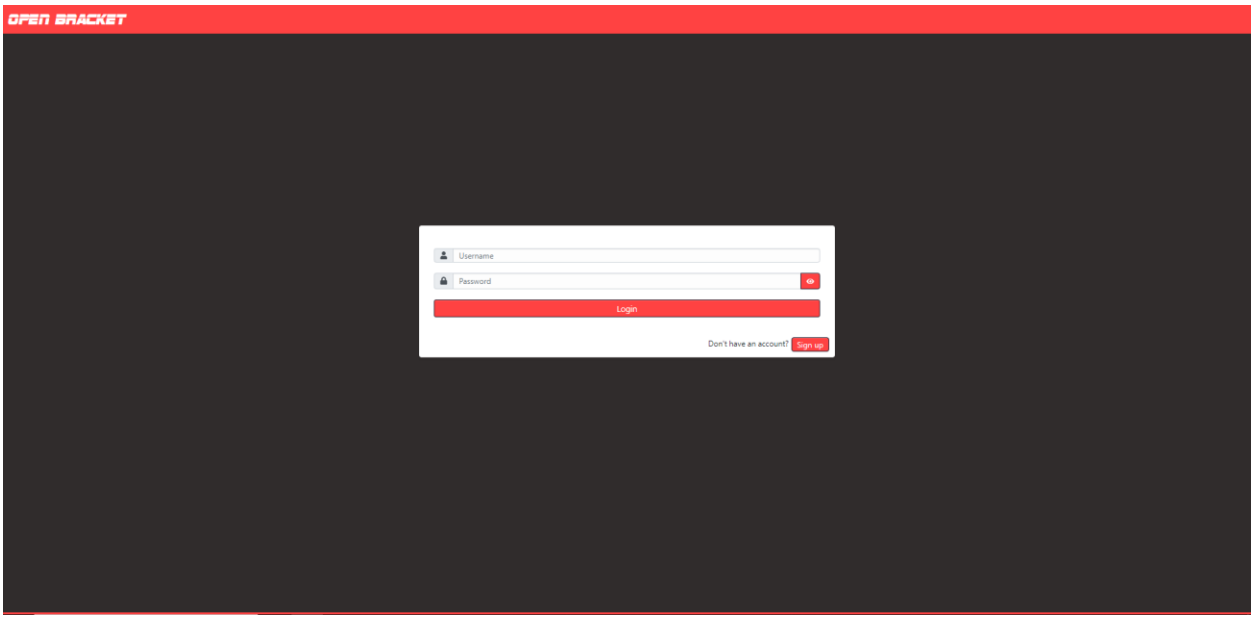
- hostId: Το πεδίο που δηλώνει τον διοργανωτή. Αν και γίνεται αρχικοποίηση μέσω του yup, δεν υπάρχει στη φόρμα γιατί συμπληρώνεται αυτόματα με το id του συνδεδεμένου χρήστη.

ΚΕΦΑΛΑΙΟ 5

ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

Στο κεφάλαιο αυτό, παρουσιάζεται η λειτουργικότητα του συστήματος μέσω των σενάριων χρήσης της εφαρμογής. Με αυτό το κεφάλαιο ολοκληρώνεται και η διπλωματική εργασία. Όλοι οι χρήστες έχουν τις ίδιες λειτουργίες. Η μόνη διαφορά είναι ότι ο κάθε χρήστης έχει παραπάνω λειτουργίες για τα δικά του τουρνουά ως διοργανωτής.

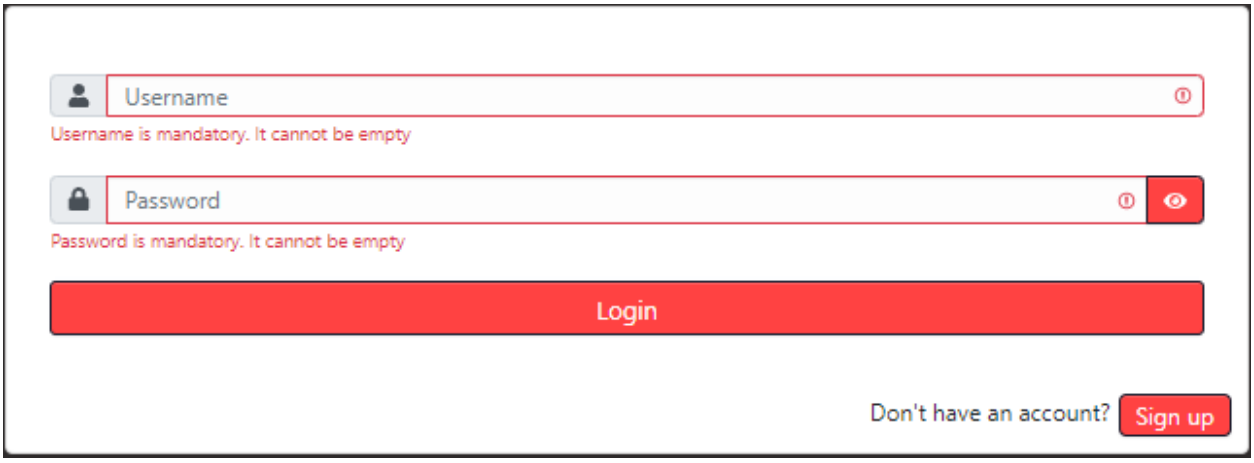
5.1 Σύνδεση & Εγγραφή



Εικόνα 5.1: Σελίδα σύνδεσης.

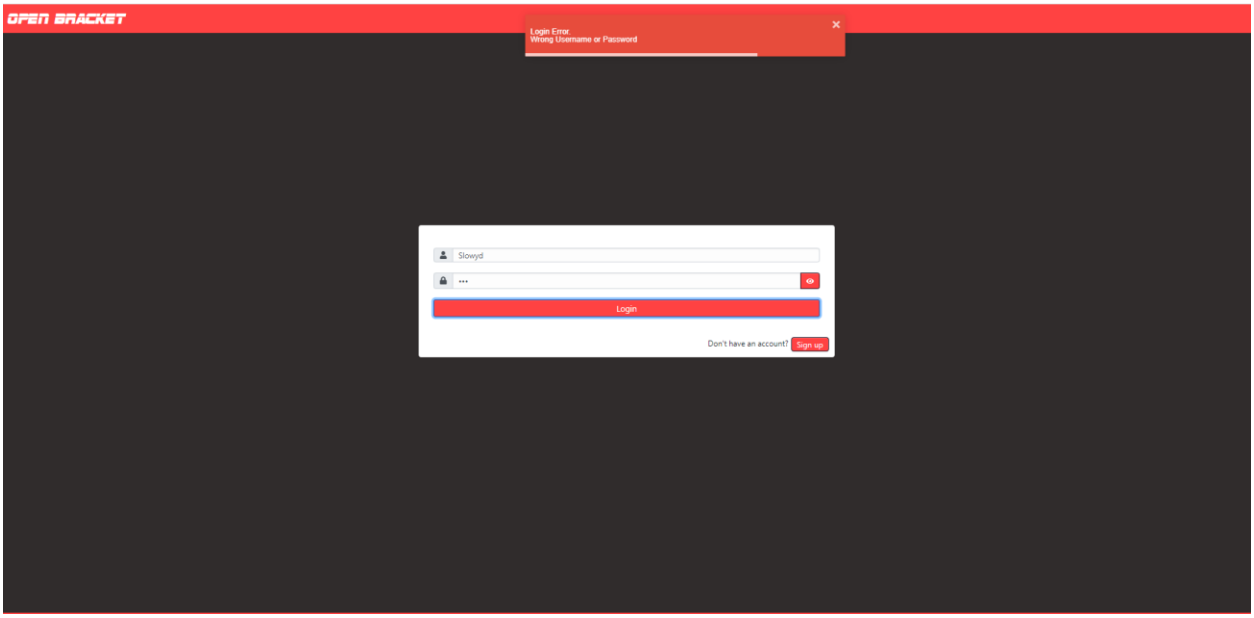
Η σελίδα σύνδεσης είναι η πρώτη σελίδα που βλέπει ένας χρήστης που πλοηγείται την ιστοσελίδα για πρώτη φορά. Πάνω αριστέρα φαίνεται το όνομα της ιστοσελίδας, “OPEN BRACKET”, και στη μέση της σελίδας μια φόρμα εισαγωγής των στοιχείων σύνδεσης.

Αν ο χρήστης δεν εισάγει τα στοιχεία του, αφήσει τα πεδία κενά και πατήσει το κουμπί Login, θα εμφανιστούν μηνύματα ενημέρωσης λάθους.



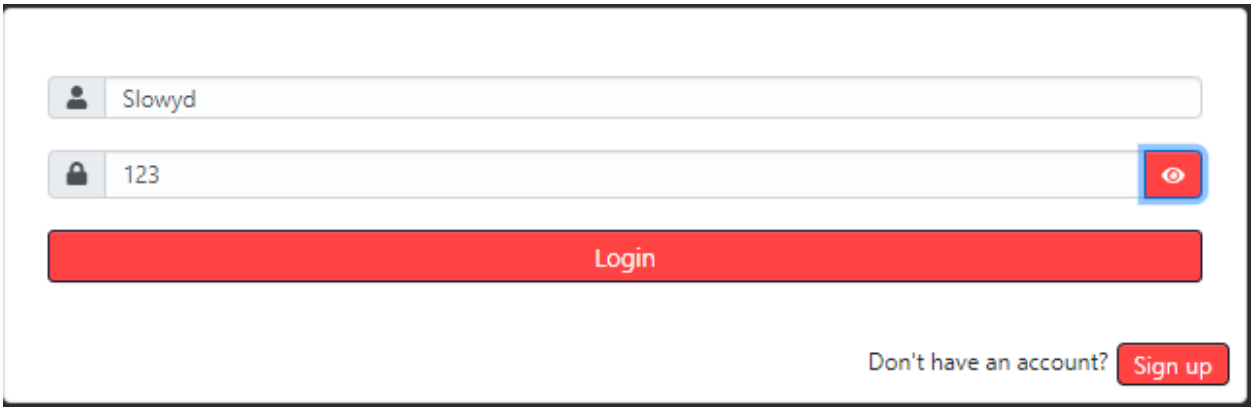
Εικόνα 5.2: Φόρμα σύνδεσης με κενά πεδία.

Αν ο χρήστης εισάγει τα στοιχεία του και πατήσει του κουμπί Login αλλά ένα τουλάχιστον από αυτά είναι λάθος τότε εμφανίζεται μήνυμα σφάλματος.



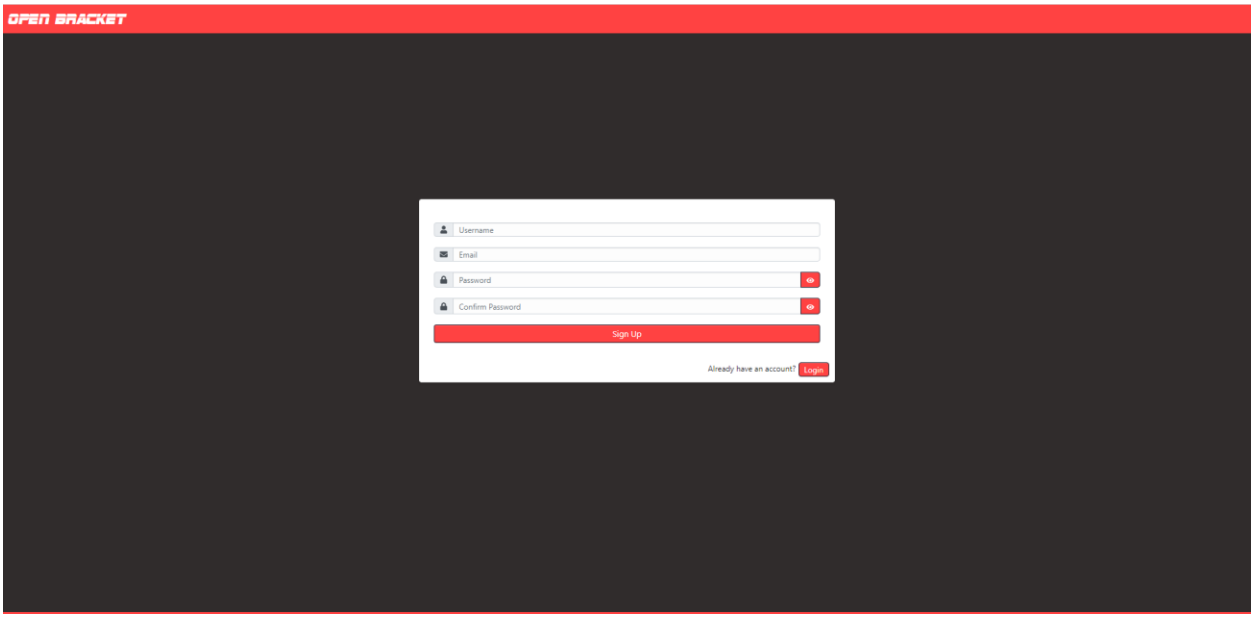
Εικόνα 5.3: Σελίδα σύνδεσης με λάθος στοιχεία σύνδεσης.

Αν ο χρήστης πατήσει το κουμπί με το ματάκι μπορεί να εμφανίσει τον κωδικό πρόσβασης που έχει εισάγει. Αν το πατήσει πάλι, ο κωδικός θα μετατραπεί πίσω σε αστεράκια.



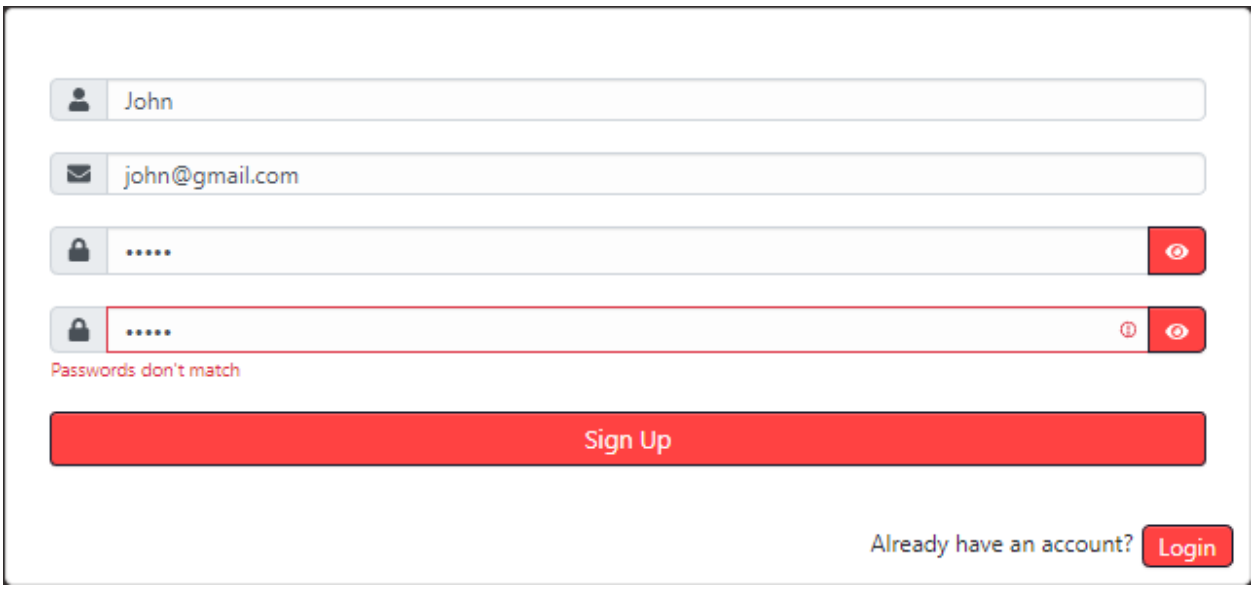
Εικόνα 5.4: Φόρμα σύνδεσης με εμφάνιση κωδικού.

Αν ο χρήστης δεν έχει λογαριασμό τότε μπορεί να πατήσει το κουμπί Sign up και να κατευθυνθεί στη σελίδα Εγγραφής.



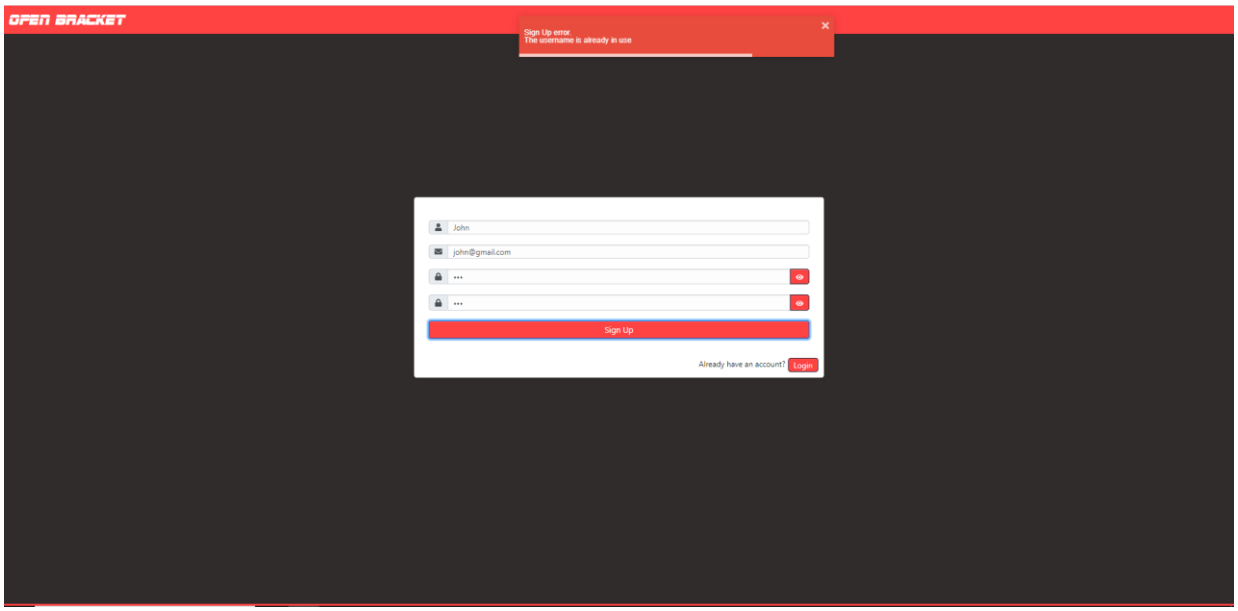
Εικόνα 5.5: Σελίδα εγγραφής.

Στη σελίδα εγγραφής υπάρχουν 4 πεδία τα οποία είναι όλα υποχρεωτικά. Το όνομα χρήστη, το email, τον κωδικό πρόσβασης και την επιβεβαίωση κωδικού πρόσβασης. Όπως και στη σελίδα σύνδεσης, αν ο χρήστης δεν εισάγει κάποιο από τα στοιχεία τότε η εφαρμογή δεν θα προχωρήσει και θα εμφανίσει μηνύματα ότι λείπουν στοιχεία. Ένας επιπλέον έλεγχος που έχει η φόρμα εγγραφής είναι ότι η επιβεβαίωση κωδικού πρόσβασης πρέπει να είναι ίδια με τον κωδικό πρόσβασης. Αν δεν είναι θα εμφανιστεί μήνυμα λάθους.



Εικόνα 5.6: Φόρμα εγγραφής με λάθος επιβεβαίωση κωδικού πρόσβασης.

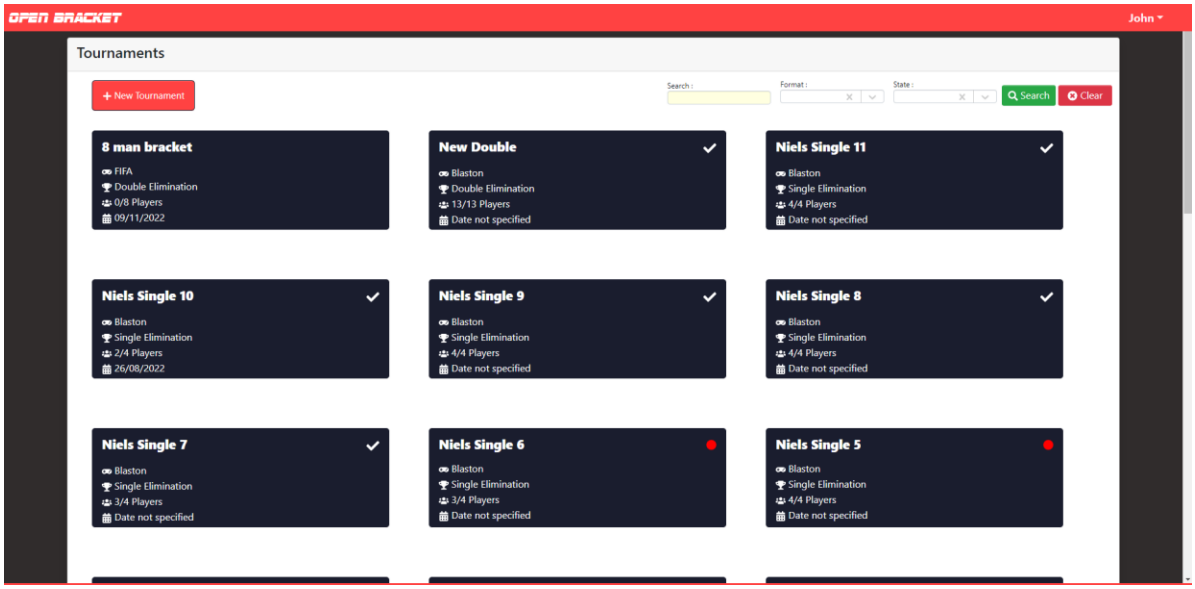
Αν το όνομα χρήστη υπάρχει ήδη στο σύστημα τότε θα εμφανιστεί ενημερωτικό σφάλμα, όπως φαίνεται στην εικόνα 5.7.



Εικόνα 5.7: Σελίδα εγγραφής με σφάλμα όνομα χρήστη που υπάρχει ήδη.

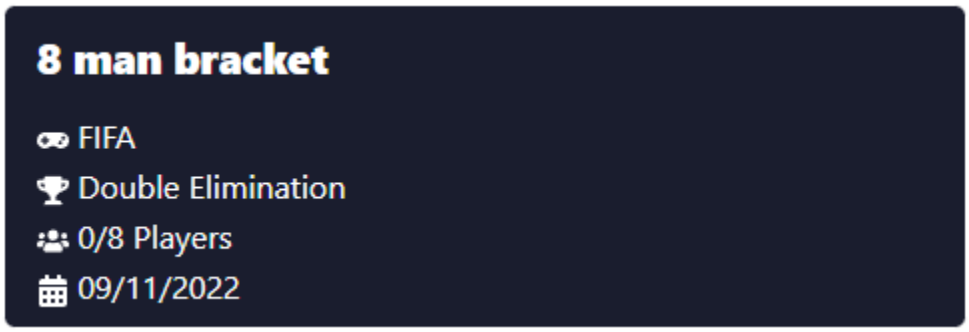
5.2 Αρχική σελίδα

Αν ο χρήστης συνδεθεί ή εγγραφεί με επιτυχία στο σύστημα τότε θα δρομολογηθεί στην αρχική σελίδα.



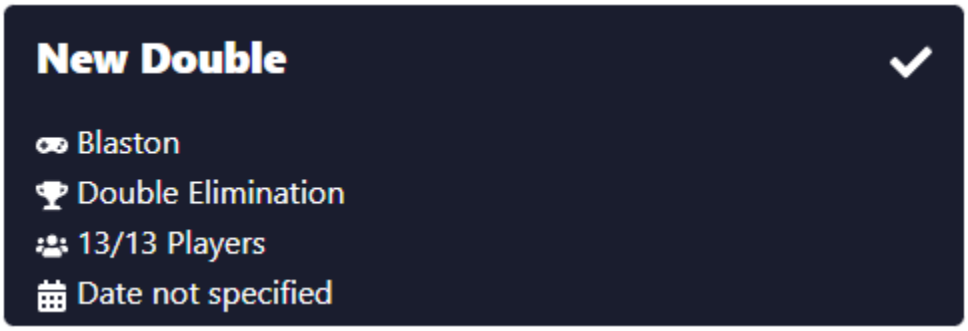
Εικόνα 5.8: Αρχική σελίδα.

Στην αρχική σελίδα φαίνονται όλα τα τουρνουά ταξινομημένα ανά ημερομηνία δημιουργίας με προτεραιότητα τα νεότερα. Σε κάθε τουρνουά απεικονίζεται ο τίτλος του, το παιχνίδι του τουρνουά, ο τύπος του τουρνουά, ο αριθμός συμμετέχοντων, η κατάσταση στην οποία βρίσκεται το τουρνουά και η ημερομηνία έναρξης αν υπάρχει.

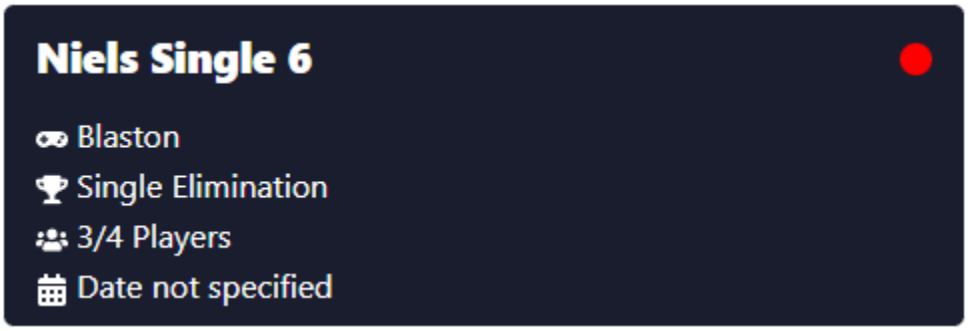


Εικόνα 5.9: Τουρνουά αρχικής σελίδας.

Η κατάσταση του τουρνουά φαίνεται από τα εικονίδια που έχει πάνω δεξιά. Αν το κουτάκι δείχνει ένα τικ σημαίνει ότι έχει ολοκληρωθεί (εικόνα 5.10). Αν το κουτάκι δείχνει έναν κόκκινο κύκλο σημαίνει ότι το τουρνουά έχει ξεκινήσει (εικόνα 5.11). Αν το κουτάκι δεν δείχνει τίποτα σημαίνει ότι ακόμα δεν έχει ξεκινήσει και μπορεί να δεχτεί συμμετοχές.

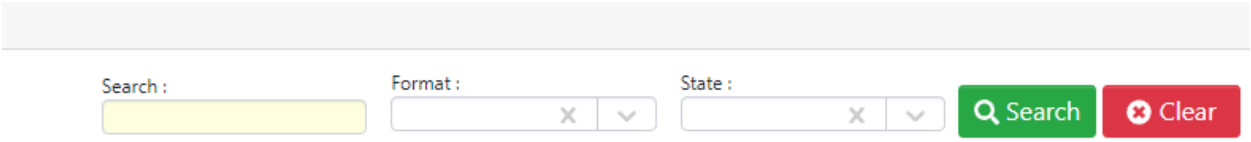


Εικόνα 5.10: Τουρνουά αρχικής σελίδας σε κατάσταση ολοκλήρωσης.



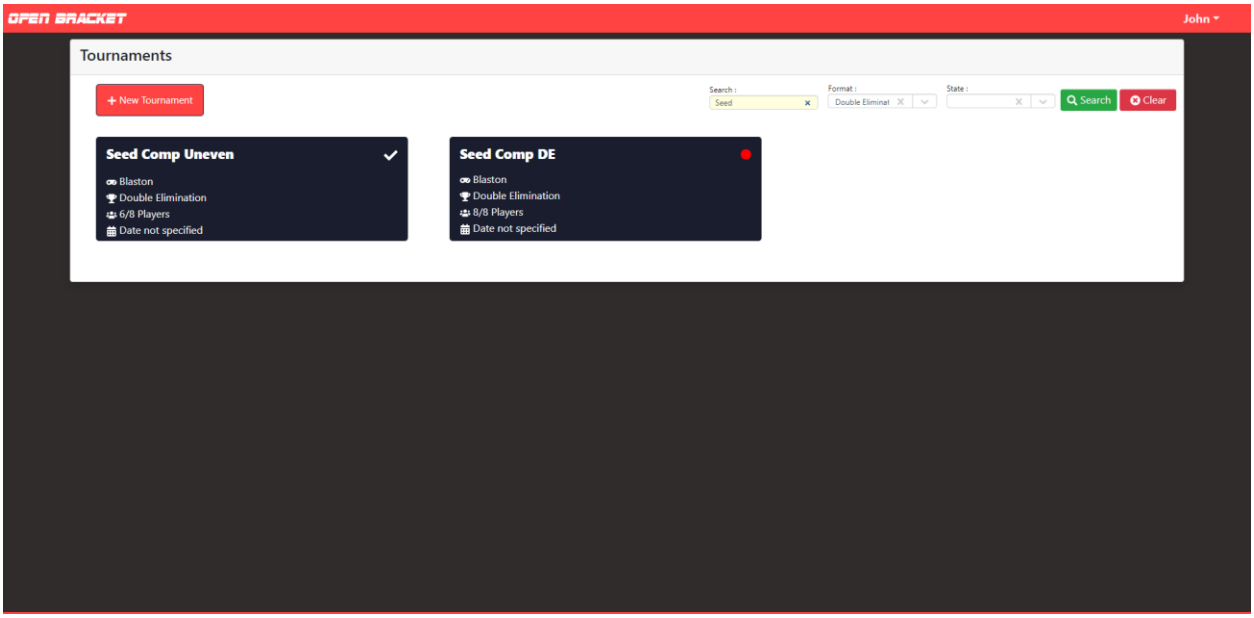
Εικόνα 5.11: Τουρνουά αρχικής σελίδας σε κατάσταση εξέλιξης.

Ο χρήστης μπορεί να αναζητήσει κάποιο συγκεκριμένο τουρνουά με τα φίλτρα αναζήτησης που προσφέρει η αρχική σελίδα πάνω δεξιά. Υπάρχουν φίλτρα για λεκτική αναζήτηση, για τύπο τουρνουά και για κατάσταση τουρνουά.



Εικόνα 5.12: Φίλτρα αναζήτησης αρχικής σελίδας.

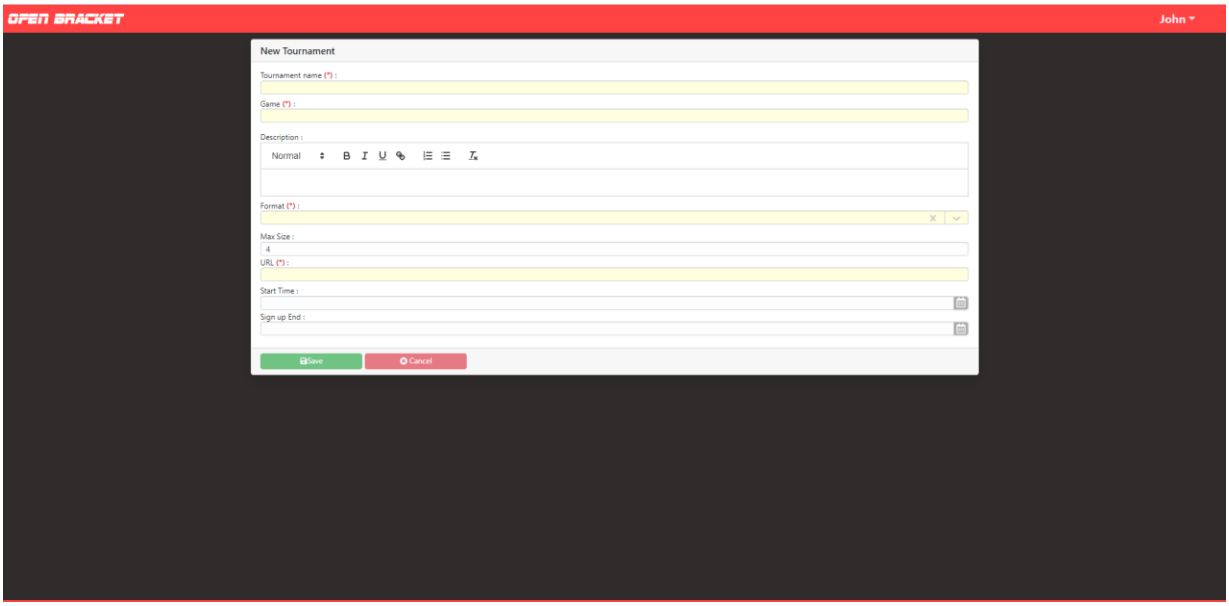
Μόλις εισάγει σε ένα από τα πεδία και πατήσει το κουμπί Search, θα εμφανιστούν τα τουρνουά βάση των φίλτρων. Αν πατήσει το κουμπί Clear, θα επαναφέρει τα πεδία όπως αρχικά ήταν.



Εικόνα 5.13: Αναζήτηση τουρνουά με λεκτικό ”Seed” και τύπο τουρνουά Double Elimination.

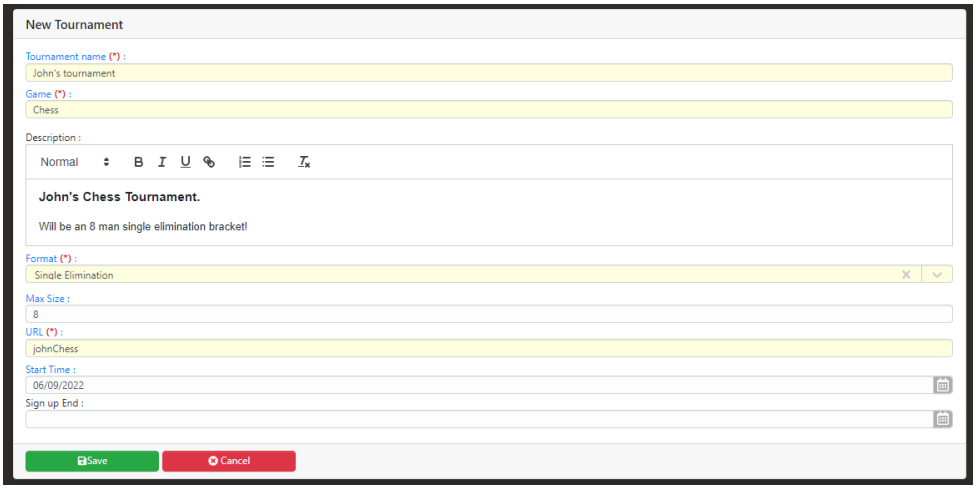
Στην αρχική σελίδα υπάρχει επίσης και το κουμπί δημιουργίας καινούργιου Τουρνουά. Όταν ο χρήστης πατάει το κουμπί “New Tournament” μεταφέρεται στη σελίδα δημιουργίας Τουρνουά, η οποία αναλύεται στο επόμενο υποκεφάλαιο.

5.3 Δημιουργία και διαχείριση τουρνουά



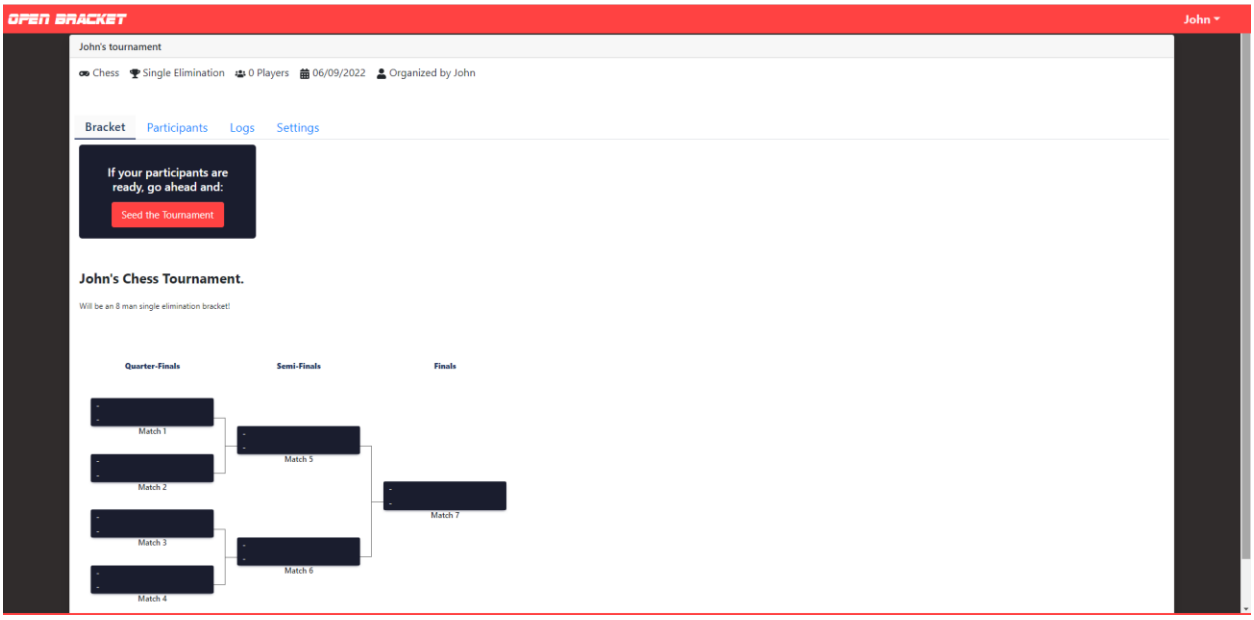
Εικόνα 5.14: Σελίδα δημιουργίας τουρνουά.

Στην παραπάνω σελίδα φαίνεται μια φόρμα στην οποία ο χρήστης μπορεί να εισάγει τα στοιχεία του τουρνουά που θέλει να δημιουργήσει. Πρόκειται για τον τίτλο του τουρνουά, το παιχνίδι του τουρνουά, την περιγραφή του, τον τύπο τουρνουά, τον μέγιστο αριθμό συμμετεχόντων, την ημερομηνία έναρξης και την ημερομηνίας λήξης που μπορεί να υποβάλλει κάποιος συμμετοχή.



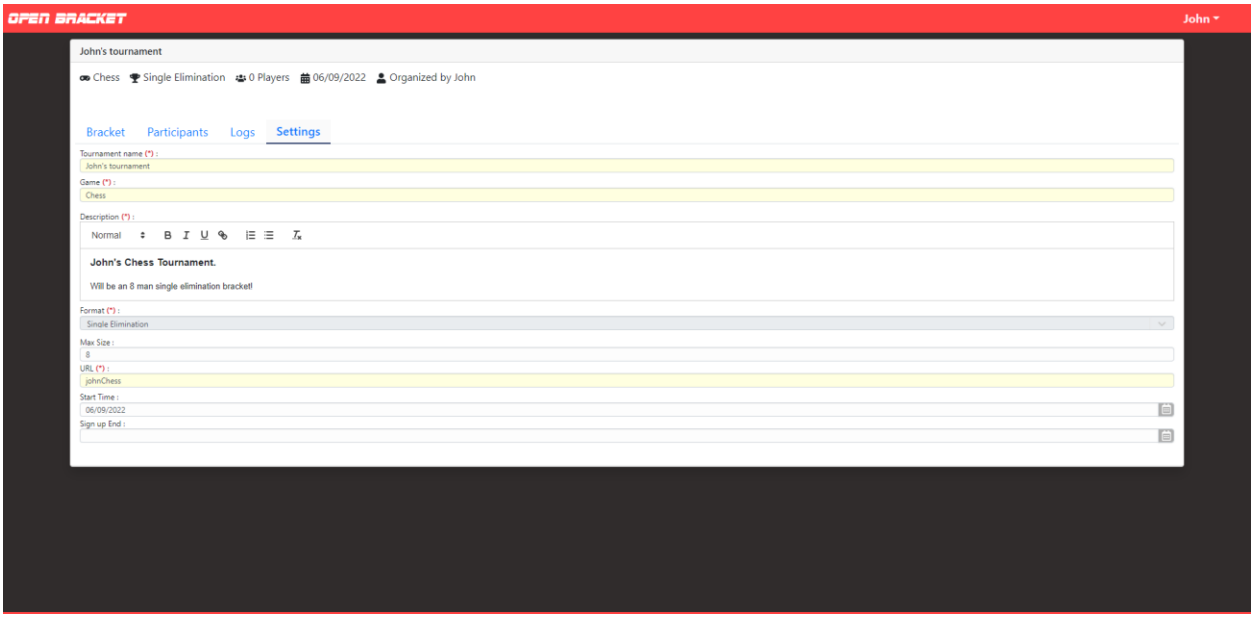
Εικόνα 5.15: Σελίδα δημιουργίας τουρνουά με συμπληρωμένα στοιχεία.

Μόλις συμπληρώσει τα στοιχεία ο χρήστης και πατήσει το κουμπί Save θα δρομολογηθεί στη σελίδα του τουρνουά.



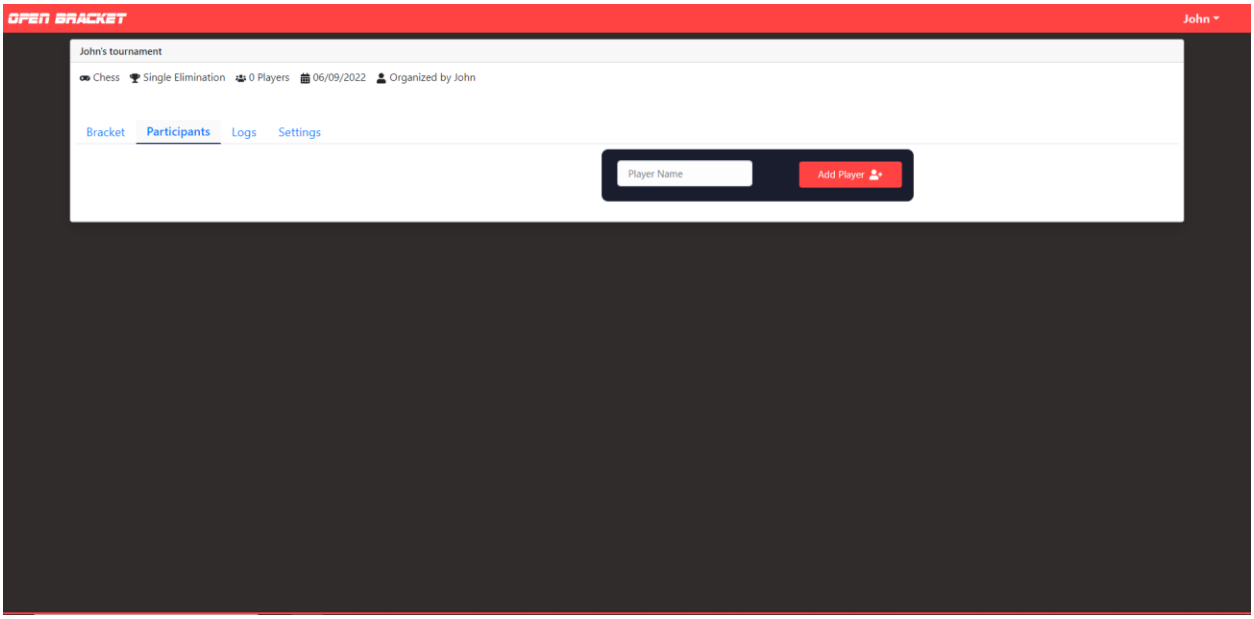
Εικόνα 5.16: Σελίδα τουρνουά.

Η σελίδα του τουρνουά χωρίζεται σε τέσσερις καρτέλες. Η αρχική καρτέλα είναι η Bracket, στην οποία φαίνεται η περιγραφή του τουρνουά και η εξέλιξή του. Εφόσον μόλις δημιουργήθηκε το τουρνουά και δεν έχει ακόμα συμμετέχοντες, για να μην είναι κενή η σελίδα, απεικονίζεται ένα κενό bracket του ίδιου μεγέθους με το μέγιστο αριθμό συμμετεχόντων που μπορεί να έχει. Ο αριθμός αυτός ή τα άλλα στοιχεία του τουρνουά με εξαίρεση τον τύπο του τουρνουά μπορούν να αλλάξουν από την καρτέλα Settings.



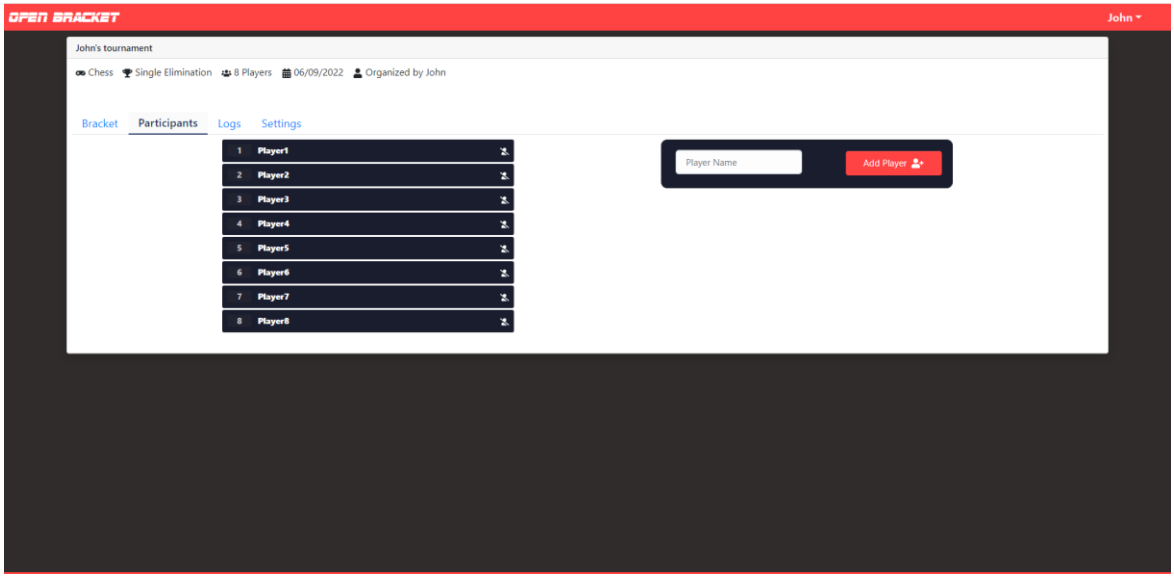
Εικόνα 5.17: Καρτέλα Settings.

Παρακάτω ακολουθεί η καρτέλα Participants στην οποία ο διοργανωτής βλέπει τους χρήστες που έχουν δηλώσει συμμετοχή και μπορεί να προσθέσει παίκτες μόνος του ή να αφαιρέσει.



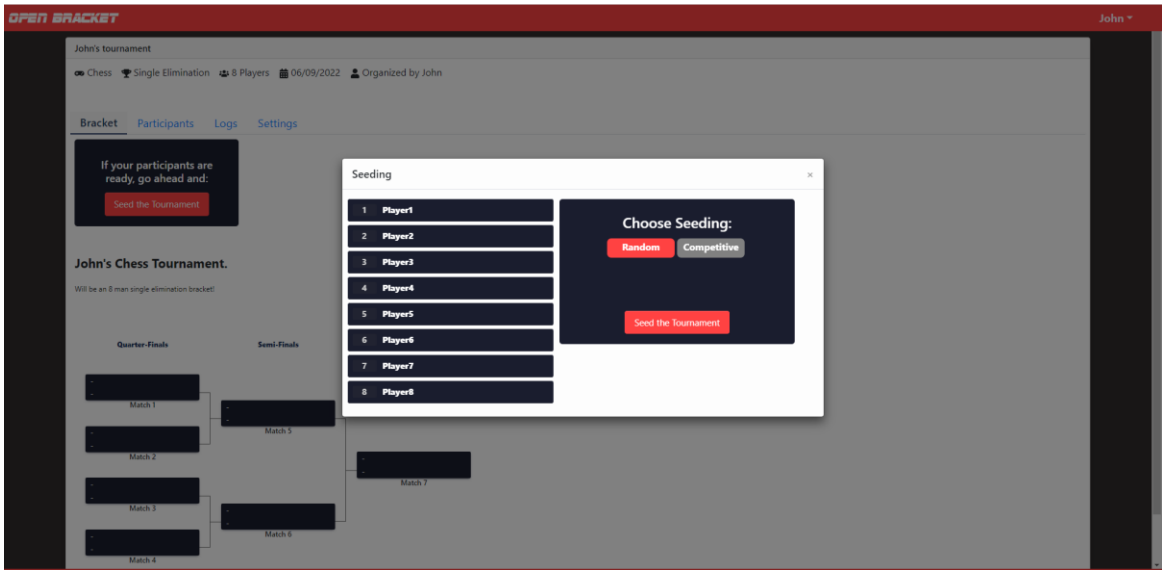
Εικόνα 5.18: Καρτέλα Participants χωρίς συμμετέχοντες.

Ο προσθήκη των παικτών γίνεται μέσω του πεδίου εισαγωγής ονόματος και του κουμπιού Add Player.



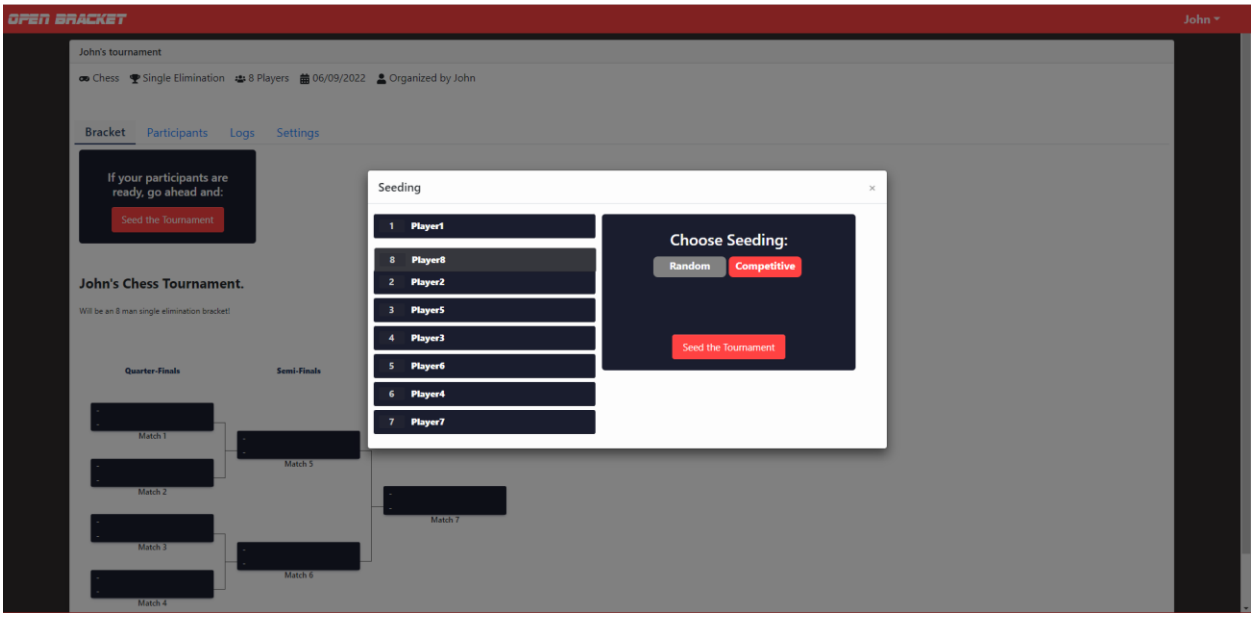
Εικόνα 5.19: Καρτέλα Participants με 8 συμμετέχοντες.

Αφου έχουν εισαχθεί οι παίκτες, ο διοργανωτής μπορεί να γυρίσει στην καρτέλα Bracket και να πατήσει το κουμπί Seed the Tournament.



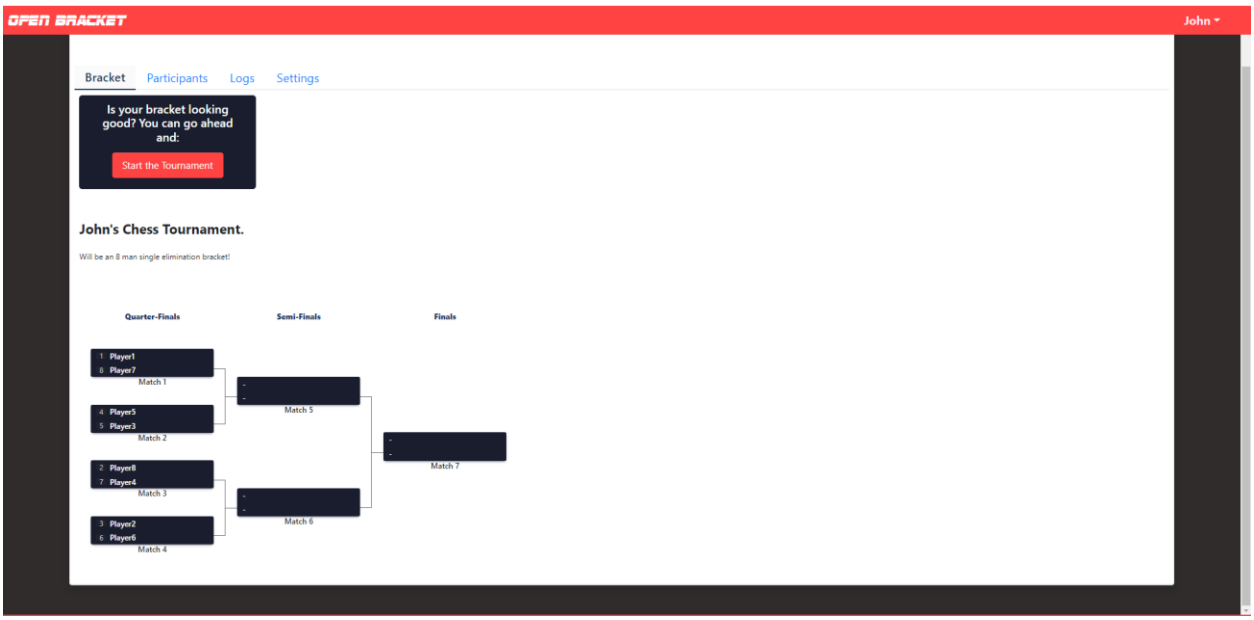
Εικόνα 5.20: Επιλογή seeding.

Θα εμφανιστεί μια καρτέλα στην οποία μπορεί να επιλέξει τυχαία τοποθέτηση των παικτών (Random) ή με seeding (Competitive). Αν επιλεχθεί Random τότε η σειρά των παικτών θα τοποθετηθεί τυχαία στο bracket. Αν όμως επιλέξει Competitive, έχει τη δυνατότητα να αλλάξει τη σειρά των παικτών, και το bracket θα φτιαχτεί λάβοντας υπόψιν αυτό.



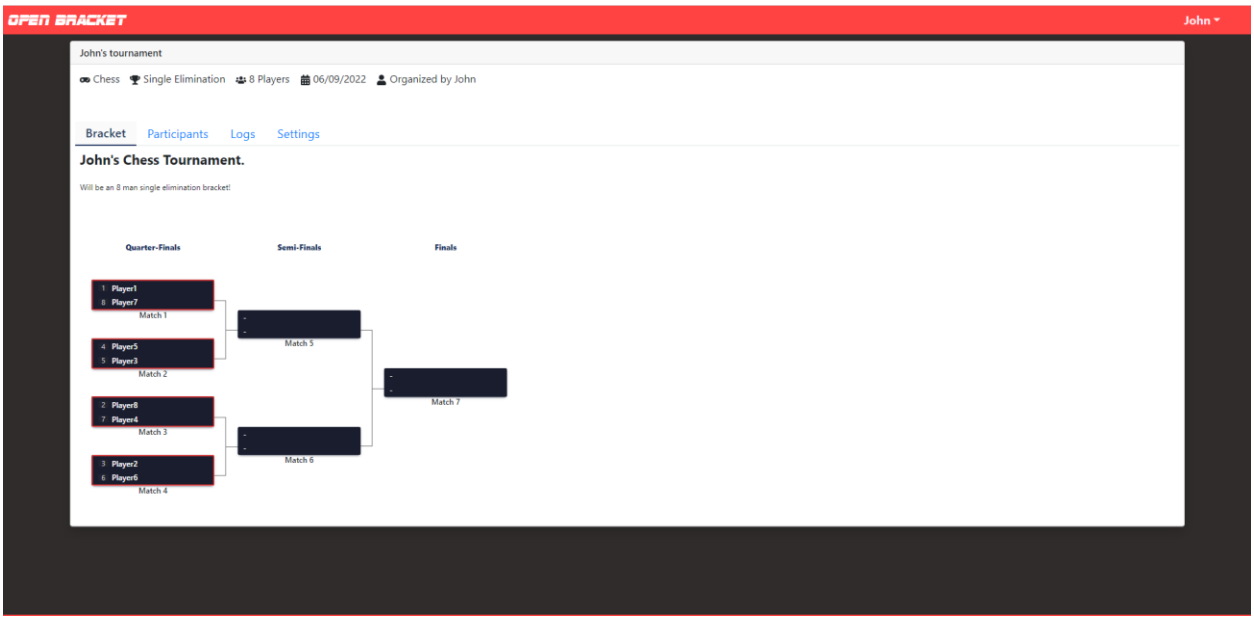
Εικόνα 5.21: Αλλαγή σειράς παικτών.

Οι παίκτες τοποθετούνται στο bracket αφού ο διοργανωτής πατήσει το κουμπί Seed the Tournament και το bracket συμπληρώνεται.



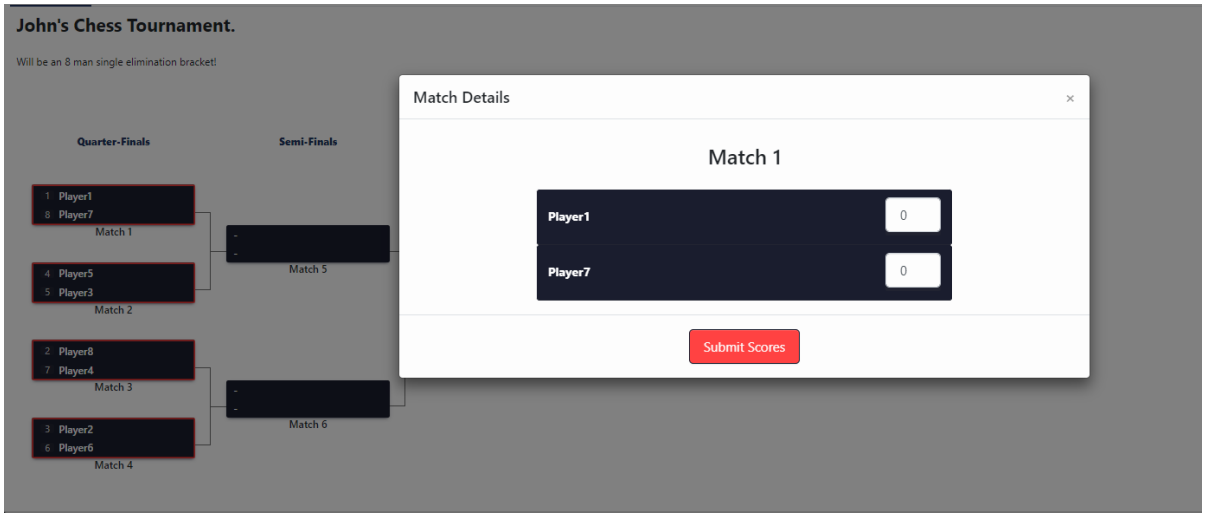
Εικόνα 5.22: Τουρνουά με συμπληρωμένο bracket.

Όπως φαίνεται στην εικόνα 5.22, το bracket έχει φτιαχτεί σωστά με seeding και αναμένει την έναρξή του. Μόλις πατήσει το Start the Tournament ο διοργανωτής, το τουρνουά έχει επίσημα ξεκινήσει και στη σελίδα απεικονίζονται οι πρώτοι αγώνες που πρέπει να παιχτούν.



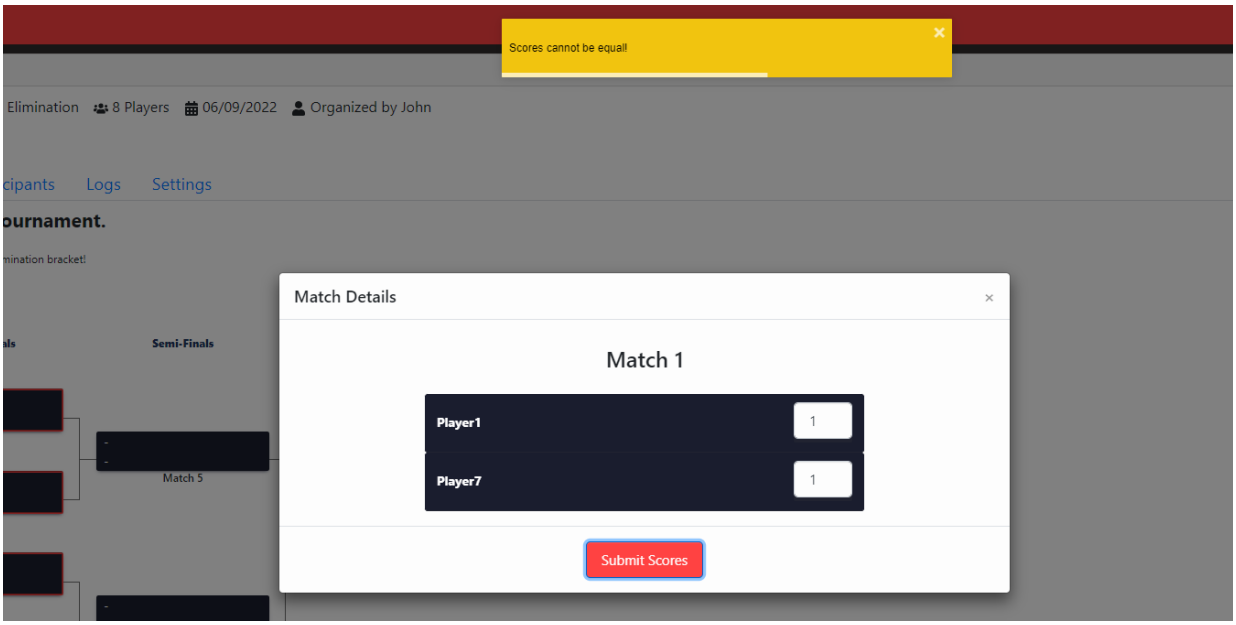
Εικόνα 5.23: Σελίδα Τουρνουά σε κατάσταση εξέλιξης.

Εφόσον πατηθεί ένας από τους αγώνες που τρέχουν, υπάρχουν επιλογές για ενημέρωση αποτελεσμάτων.



Εικόνα 5.24: Ενημέρωση αποτελεσμάτων αγώνα.

Το σύστημα δεν θα αφήσει εισαγωγή σκορ ισοπαλίας και θα εμφανιστεί ενημερωτικό μήνυμα αν πατηθεί το Submit Scores με ίδιο σκορ και στους δύο παίκτες.

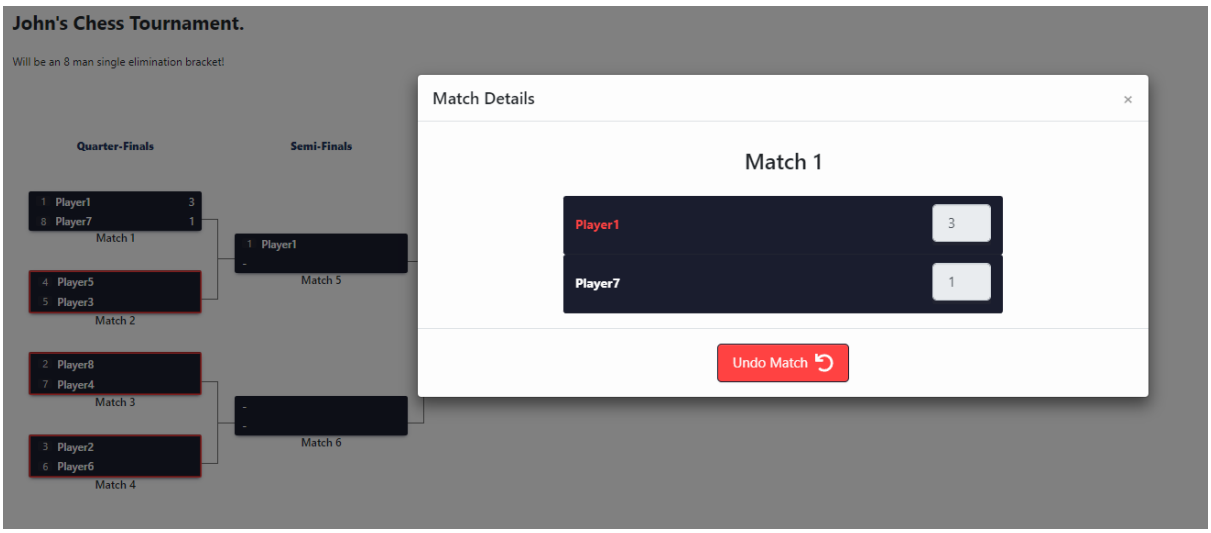


Εικόνα 5.25: Ενημερωτικό μήνυμα μη επιτρεπτού σκορ.

Νικητής επιλέγεται ο παίκτης με το μεγαλύτερο σκορ και προωθείται στον επόμενο αγώνα.

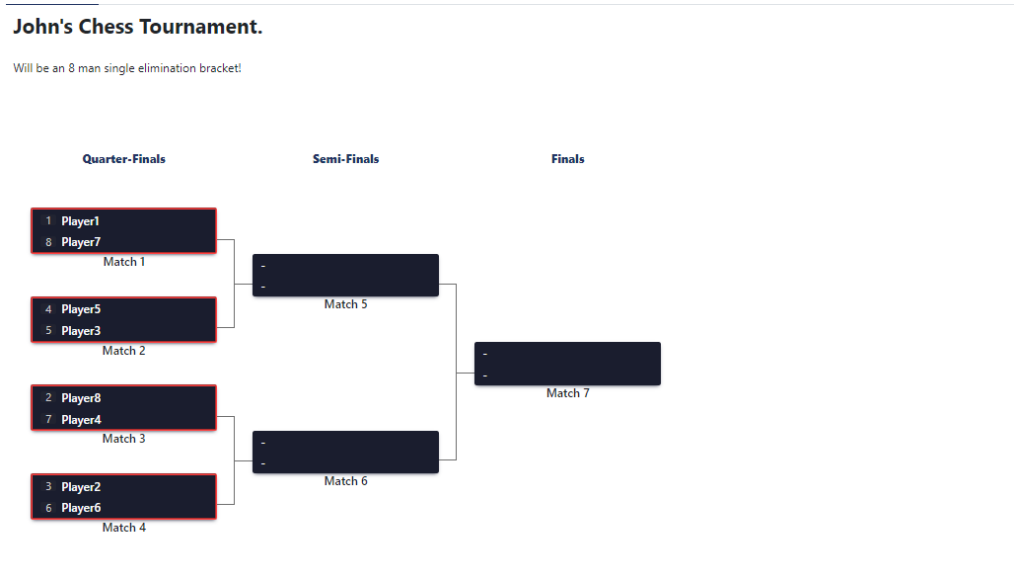
Εικόνα 5.26: Αποτέλεσμα αγώνα και προώθηση νικητή.

Αν τυχόν έχει γίνει λάθος σε κάποιο απότελεσμα, υπάρχει και η επιλογή αναίρεσης. Η επιλογή αναίρεσης είναι διαθέσιμη σε οποιοδήποτε ολοκληρωμένο αγώνα του οποίου ο ακριβώς επόμενος του δεν έχει ολοκληρωθεί.



Εικόνα 5.27: Αναίρεση αγώνα.

Στην καρτέλα του 1ου αγώνα φαίνεται ποιος έχει νικήσει, είναι απενεργοποιημένα τα πεδία ενημέρωση σκορ και υπάρχει η επιλογή αναίρεσης. Μόλις πατηθεί το κουμπί Undo Match, το bracket θα επανέλθει σαν να μην είχε συμβεί πότε ο 1ος αγώνας.

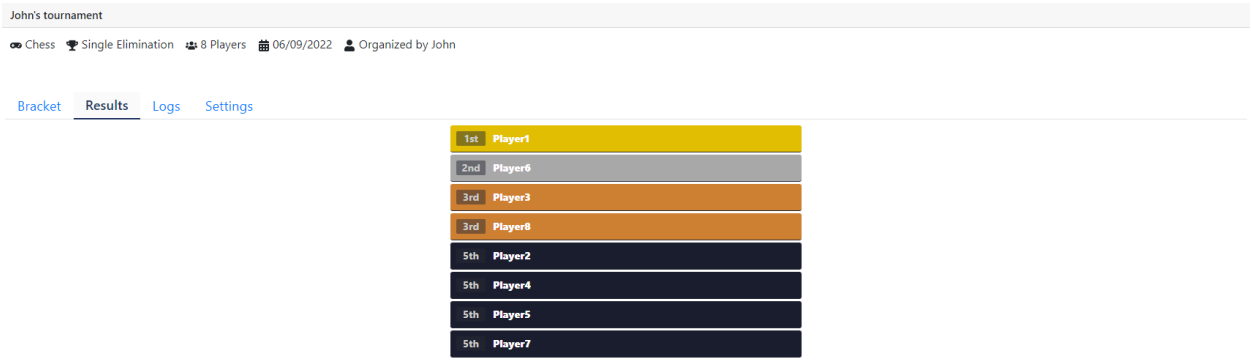


Εικόνα 5.28: Το bracket μετά από αναίρεση αγώνα.

Εφόσον ολοκληρωθούν όλοι οι αγώνες, το τουρνουά θα μπει σε κατάσταση ολοκλήρωσης και στο bracket θα εμφανιστούν με συγκεκριμένο χρώμα ο 1ος, 2ος και 3ος στην κατάταξη του τουρνουά.

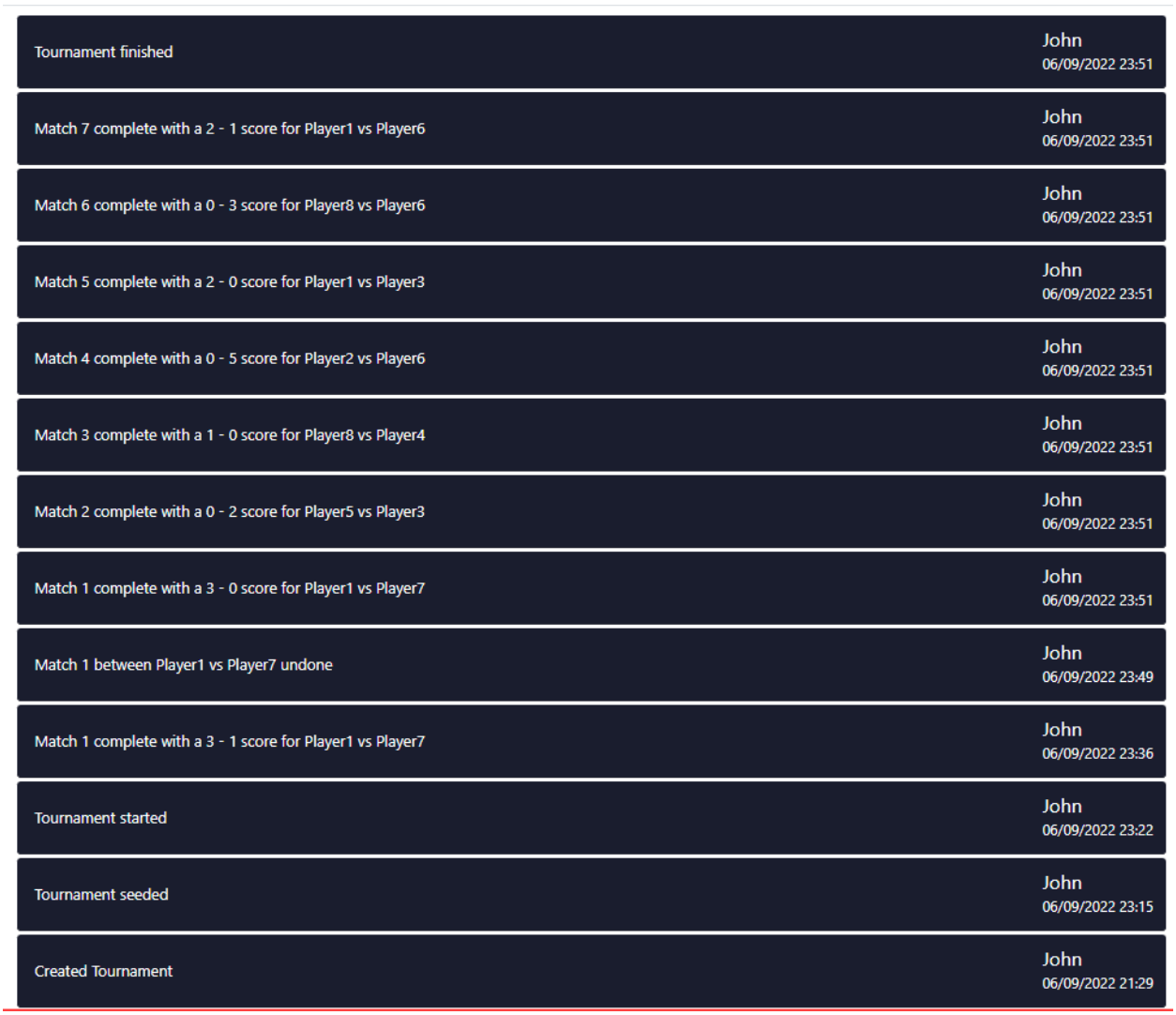
Εικόνα 5.29: Τελικό bracket ολοκληρωμένου τουρνουά.

Η τελική κατάταξη όλων των συμμετεχόντων φαίνεται στην καρτέλα Results που έχει πάρει τη θέση της καρτέλας Participants.



Εικόνα 5.30: Καρτέλα Results.

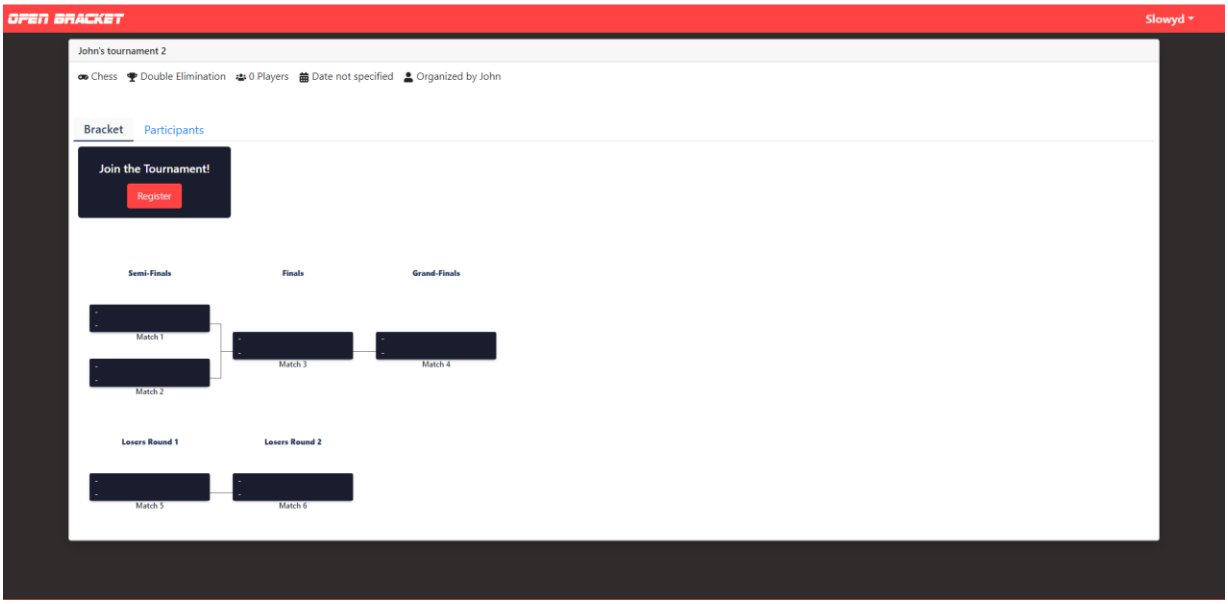
Η τελευταία καρτέλα της σελίδας τουρνουά, είναι η καρτέλα Logs. Εκεί φαίνονται οι ενέργειες που καταγράφτηκαν κατά την εξέλιξη του τουρνουά.



Εικόνα 5.31: Καρτέλα Logs.

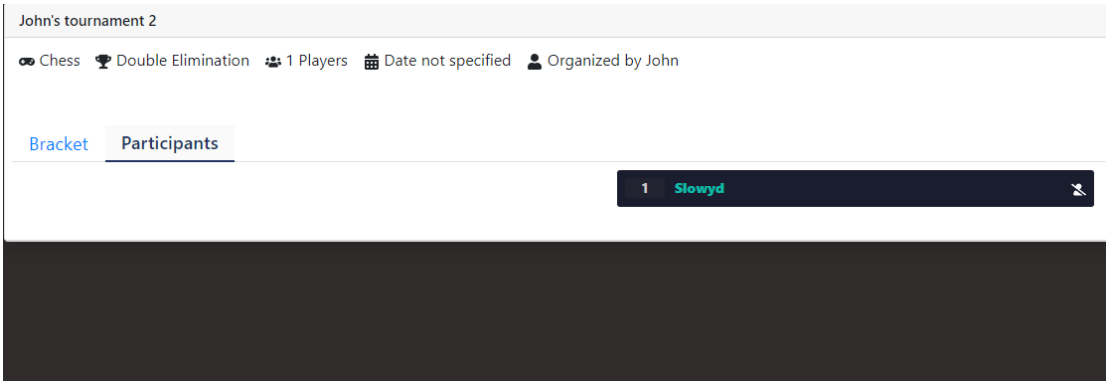
5.4 Συμμετοχή τουρνουά και προφίλ χρήστη

Η σελίδα του τουρνουά παρουσιάστηκε μέχρι τώρα από την πλευρά του διοργανωτή. Ένας χρήστης που παρακολουθεί ένα τουρνουά μη οργανωμένο από αυτόν, μπορεί μόνο να δηλώσει συμμετοχή και να παρακολουθεί τους συμμετέχοντες και την εξέλιξη του τουρνουά.



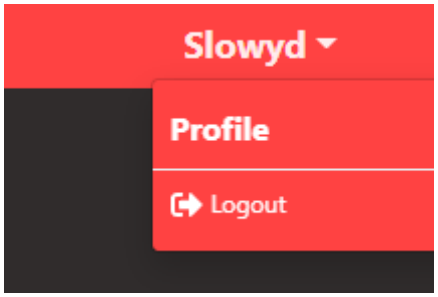
Εικόνα 5.32: Δήλωση συμμετοχής σε τουρνουά.

Μόλις δηλωθεί η συμμετοχή, ο χρήστης μπορεί να δει το όνομα του στην καρτέλα Participants όπου και έχει την επιλογή απεγγραφής.



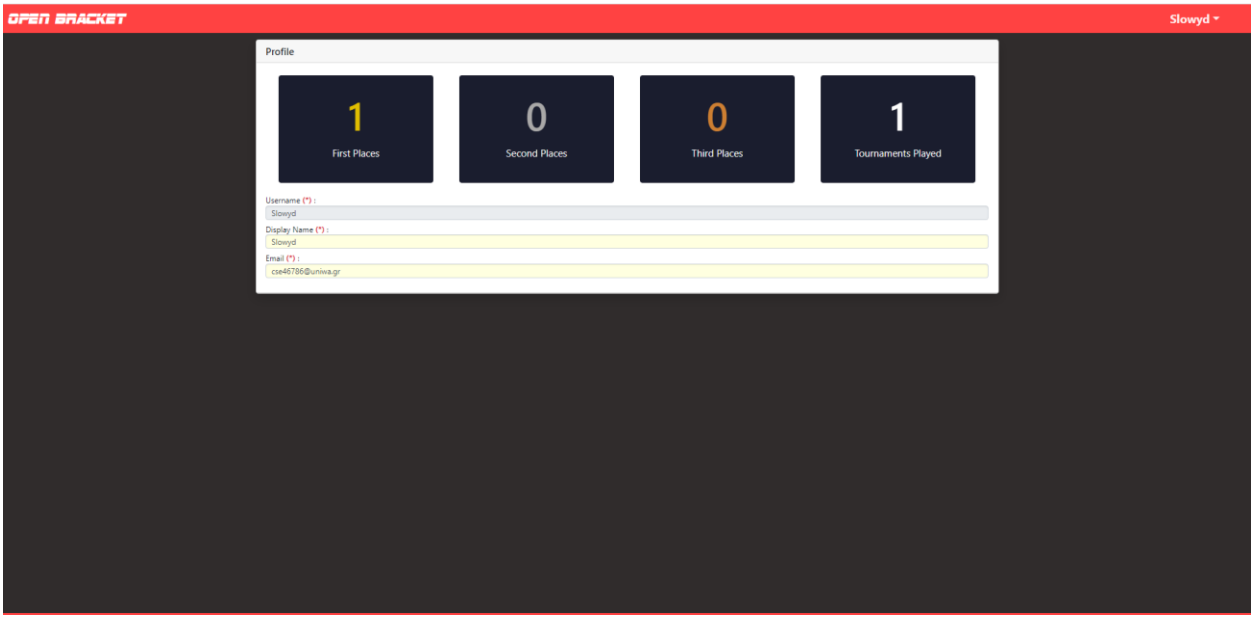
Εικόνα 5.33: Καρτέλα Participants ως συμμετέχοντας.

Ο χρήστης μπορεί οποτεδήποτε να μπει στο προφίλ του χρησιμοποιώντας το κουμπί στο πάνω δεξιά μέρος της σελίδας.



Εικόνα 5.34: Κουμπιά προφίλ και αποσύνδεσης.

Για να δει τα στοιχεία του, να τα αλλάξει και επίσης να παρακολουθήσει τα στατιστικά του από όλα τα τουρνουά που έχει πάρει μέρος.



Εικόνα 5.35: Σελίδα προφίλ.

Από το ίδιο μενού, ο χρήστης μπορεί να πατήσει το κουμπί Logout για να αποσυδενθεί και να επανέλθει στη 1η σελίδα σύνδεσης.

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό γίνεται μια σύνοψη της εφαρμογής που αναπτύχθηκε στην παρούσα διπλωματική εργασία, αρχίζοντας με μια ανακεφαλαίωση. Παρουσιάζονται τα συμπεράσματα αλλά και προβλήματα που προέκυψαν κατά τη διάρκεια ανάπτυξης. Επιπλέον, περιγράφονται μελλοντικές επεκτάσεις που μπορούν να βοηθήσουν στην βελτιστοποίηση της εφαρμογής.

6.1 Σύνοψη & Συμπεράσματα

Η διπλωματική εργασία άρχισε με την εισαγωγή στην ένοια “τουρνουα” και στα είδη τουρνουά που χρησιμοποιούνται στην εποχή μας, από τα πιο απλά στα πιο περίπλοκα. Έχοντας προσωπική εμπειρία από την συμμετοχή, παρακολούθηση αλλά και οργάνωση πολλαπλών τουρνουά, εκτός από την επεξήγηση τους έγινε ανάλυση και της καλύτερης επιλογής τύπου τουρνουά βάση του χρόνου και των πόρων που διαθέτει ένας διοργανωτής. Στη συνέχεια παρουσιάστηκαν οι κύριες τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Δώθηκε έμφαση στις τρεις τεχνολογίες HTML, CSS και Javascript που χρησιμοποιούνται σχεδόν πάντα μαζί για την ανάπτυξη διαδικτυακών εφαρμογών και υπάρχουν άφθονα εγχειρίδια και περιεχόμενο πάνω σε αυτές. Αν και το HTML παραμένει ο σκελετός της εφαρμογής, η κύρια τεχνολογία που χρησιμοποιήθηκε για την ανάπτυξη των διεπαφών χρήστη ήταν η React, η οποία πρόσφερε πολλές χρήσιμες λειτουργίες που βοήθησαν στην ευκολότερη εξέλιξη της εφαρμογής. Το ίδιο και με την Typescript, η οποία πρόσφερε τη δυνατότητα δήλωσης τύπων δεδομένων πάνω στην Javascript που βοήθησε στον άμεσο εντοπισμό σφαλμάτων. Για τη βάση δεδομένων της εφαρμογής χρησιμοποιήθηκε η MySQL και τα δεδομένα της μοιράστηκαν σε δέκα πίνακες και μια βοηθητική συνάρτηση. Για το API επιλέχθηκε να χρησιμοποιηθεί η τεχνολογία Node.js γιατί η γραφή της αποτελείται από Javascript. Με αυτόν τον τρόπο το Back-end και το Frond-end έκαναν χρήση της ίδιας γλώσσας προγραμματισμού, με αποτέλεσμα να είναι πιο εύκολη και κατανοητή η σύνδεση τους, και γενικά πιο γρήγορη η ανάπτυξη της εφαρμογής για έναν μόνο προγραμματιστή. Το επόμενο κεφάλαιο που παρουσιάστηκε ήταν η ανάλυση του συστήματος. Έγινε περιγραφή των πινάκων της βάσης δεδομένων και των τύπων δεδομένων που χρησιμοποιούν, των κύριων συναρτήσεων του API και πως αυτές χωρίζονται σε αρχεία, και των κύριων λειτουργιών του Front-end. Στο τελευταίο κεφάλαιο, παρουσιάστηκαν οι περιπτώσεις χρήσης της διαδικτυακής εφαρμογής και τα βήματα που ακολουθεί ένας χρήστης για ένα ολοκληρωμένο τουρνουά.

Ένα από τα αρχικά προβλήματα που εμφανίστηκαν ήταν η επιλογή του τρόπου εμφάνισης ενός bracket στην οθόνη του χρήστη και ο τρόπος λειτουργίας ενός τουρνουά στο API. Δεν βρέθηκαν αντίστοιχα πακέτα που να συνεργάζονται εύκολα μαζί. Επιλέχθηκε το πακέτο “react-brackets” για το Front-end επειδή πρόσφερε πολλούς τρόπους προσαρμογής και μια ωραία εμφάνιση. Για το API όμως, δεν χρησιμοποιήθηκε κανένα έτοιμο πακέτο και δημιουργήθηκαν οι αλγόριθμοι και οι συναρτήσεις από το μηδέν. Αυτό προκάλεσε την επέκταση του χρόνου ανάπτυξης γιατί δεν υπάρχουν εγχειρίδια ως προς το πως υπολογίζονται οι διάφορες κινήσεις κατά τη εξέλιξη ενός τουρνουά. Ως αποτέλεσμα, οι περισσότερες εξισώσεις βρέθηκαν με διαδικασία δοκιμής και λάθους, όπως η εξίσωση επιλογής επόμενου αγώνα ενός παίκτη που κέρδισε, ή έχασε στην περίπτωση Double Elimination Bracket, και η εξίσωση προσδιορισμού της τελικής θέσης των παικτών στα αποτελέσματα ενός τουρνουά. Αν και η διαδικασία αυτή ήταν χρονοβόρα, πρόσφερε την εξατομίκευση της εφαρμογής και μια πιο εύκολη ανάπτυξη μελλοντικών επεκτάσεων χωρίς να βασίζεται σε κάποιο εξωτερικό πακέτο.

6.2 Μελλοντικές επεκτάσεις

Η εφαρμογή είναι πλήρως λειτουργική και προσφέρει δύο από τα πιο συχνά τύποι τουρνουά αλλά για να μπορεί να ανταγωνιστεί με άλλες δημοφιλές πλατφόρμες χρειάζεται κάποιες βελτιώσεις. Οι επεκτάσεις που μπορεί να λάβει μια διαδικτυακή εφαρμογή είναι αμέτρητες αλλά παρακάτω περιγράφονται οι πιο σημαντικές ως προς το θέμα της ιστοσελίδας.

- **Είσοδος ως επισκέπτης.** Αυτήν την στιγμή όταν ανοίγει κάποιος για πρώτη φορά την ιστοσελίδα, βλέπει τη σελίδα εισόδου και δεν έχει πρόσβαση στις κύριες λειτουργίες που προσφέρει η πλατφόρμα. Η είσοδος ως επισκέπτης στο σύστημα μπορεί να προσφέρει πρόσβαση στην παρακολούθηση των τουρνουά που βρίσκονται σε εξέλιξη και αν ο χρήστης ενδιαφέρεται να πάρει συμμετοχή σε κάποιο τουρνουά ή να δημιουργήσει ένα, να καθοδηγείται στις σελίδες εισόδου/εγγραφής.

- **Επιπλέον τύποι τουρνουά.** Εκτός από τα δύο Elimination Bracket που προσφέρει το σύστημα, θα μπορούσαν να αναπτυχθούν και οι τύποι Γκρουπ και League. Επιπλέον, εφόσον αναπτυχθούν αυτά τα δύο, μπορεί να επεκταθεί το σύστημα και με τουρνουά πολλαπλών φάσεων. Για παράδειγμα, ως πρώτη φάση να είναι Γκρουπ και ως δεύτερη φάση Double Elimination Bracket με συμμετέχοντες τους νικητές των Γκρουπ.

- **Προλέψεις αγώνων.** Για να αυξηθεί η αφοσίωση των θεατών που παρακολουθούν το τουρνουά, μπορεί να αναπτυχθεί ένα σύστημα προβλέψεων στο οποίο διαλέγουν από τον 1ο αγώνα μέχρι και τον τελευταίο τον νικητή και στο τέλος βλέπουν ποιος έχει μαζέψει τις περισσότερες σωστές προβλέψεις.

- **Ειδοποιήσεις.** Για την καλύτερη ενημέρωση των χρηστών θα βοηθούσε ένα σύστημα ειδοποιήσεων που στέλνει μηνύματα για την εξέλιξη των τουρνουά που συμμετέχουν ή οργανώνουν. Επιπλέον οι σημαντικές ενημερώσεις μπορούν να σταλούν και μέσω email.

- **Βελτίωση του Design.** Η εφαρμογή αυτήν την στιγμή χρησιμοποιεί το πακέτο “react-bootstrap” για την εμφάνιση στοιχείων όπως κουμπία, πεδία και φόρμες αλλά η εμφάνιση τους είναι απλή και κοινή μεταξύ άλλων ιστοσελίδων που χρησιμοποιούν το ίδιο πακέτο. Μπορεί να αφιερωθεί χρόνος για την εξατομίκευση του design ώστε να ξεχωρίζει η πλατφόρμα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Benjamin Brett Thomas, “Development of a Tournament Management System,” M.S. thesis, Faculty of Engineering and Surveying, University of Southern Queensland, Toowoomba, Australia, 2010. [Online]. Διαθέσιμο: https://eprints.usq.edu.au/8381/1/Thomas_2009.pdf

[2] PrintYourBrackets, Single Elimination Bracket, Blind Draw, 8 Teams. [Online]. Διαθέσιμο: https://www.printyourbrackets.com/8teamsingleelimination.html

[3] PrintYourBrackets, Single Elimination Bracket, Seeded, 8 Teams. [Online]. Διαθέσιμο: https://www.printyourbrackets.com/8seeded.html

[4] PrintYourBrackets, Single Elimination Bracket, Seeded, 6 Teams. [Online]. Διαθέσιμο: https://www.printyourbrackets.com/6seeded.html

[5] PrintYourBrackets, Double Elimination Bracket, Seeded, 8 Teams. [Online]. Διαθέσιμο: https://www.printyourbrackets.com/8seededdouble.html

[6] ITTF Para Table Tennis, Rio 2016 Paralympic Games, *Handbook for Tournament Referees*, 2016. [Online]. Διαθέσιμο: https://www.ipttc.org/communication/2016/Rio/Draw%20rules.pdf

[7] Raghav Mittal, “What is an ELO Rating?”, Purple Theory, Medium, Προσπέλαση Σεπτ. 11, 2020. [Online]. Διαθέσιμο: https://medium.com/purple-theory/what-is-elo-rating-c4eb7a9061e0

[8] Kyle Hoekstra, “Who Was Chess Master Arpad Elo, and What is the Elo Rating System?”, HISTORYHIT, Προσπέλαση Νοεμ. 5, 2021. [Online]. Διαθέσιμο: https://www.historyhit.com/gaming/arpad-elo-rating-system/

[9] Challonge, Logitech, About. [Online]. Διαθέσιμο: https://challonge.com/about

[10] Challonge, Logitech, Αρχική Σελίδα. [Online]. Διαθέσιμο: https://challonge.com/

[11] Challonge, Logitech, Features, Tournaments. [Online]. Διαθέσιμο: https://challonge.com/features/tournaments

[12] Challengermode, About. [Online]. Διαθέσιμο: https://www.challengermode.com/about

[13] Challengermode, Brand assets, Product. [Online]. Διαθέσιμο: https://www.challengermode.com/brandassets

[14] Challengermode, Organizers. [Online]. Διαθέσιμο: https://www.challengermode.com/organizers

[15] Matcherino, About. [Online]. Διαθέσιμο: https://matcherino.com/

[16] Marcus Smith, “Why is HTML called markup language?”, DEVELOPER PITSTOP, Προσπέλαση Ιαν. 23, 2022. [Online]. Διαθέσιμο: https://developerpitstop.com/why-is-html-called-markup-language/

[17] Λάμπρος Βαρνάβας, “Σχεδιασμός και ανάπτυξη ιστοχώρου αναφορικά με την ανακάλυψη και την εξοικείωση του κοινού με ορόσημα τοπικής κουλτούρας ή παράδοσης.,” Διπλωματική Εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Πανεπιστήμιο Δυτικής Μακεδονίας, Κοζάνη, Ελλάδα, 2020. [Online]. Διαθέσιμο: https://dspace.uowm.gr/xmlui/handle/123456789/2291

[18] Typescript, Microsoft, “What is TypeScript?”. [Online]. Διαθέσιμο: https://www.typescriptlang.org/

[19] React, Meta Platforms, Αρχική Σελίδα. [Online]. Διαθέσιμο: https://reactjs.org/

[20] Wikipedia, The Free Encyclopedia, “MySQL”, Προσπέλαση Αυγ. 25, 2022. [Online]. Διαθέσιμο: https://en.wikipedia.org/wiki/MySQL

[21] Node.js, The OpenJS Foundation, “About Node.js”. [Online]. Διαθέσιμο: https://nodejs.org/en/about/

[22] GraphQL, The GraphQL Foundation, Αρχική Σελίδα. [Online]. Διαθέσιμο: https://graphql.org/