ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2024-2025

ΟΜΑΔΑ

ΜΙΛΤΙΑΔΗΣ ΧΑΛΑΪΔΟΠΟΥΛΟΣ, 5384

ΟΜΗΡΟΣ ΧΑΤΖΗΪΟΡΔΑΝΗΣ, 5388

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΑΪΟΣ 2025

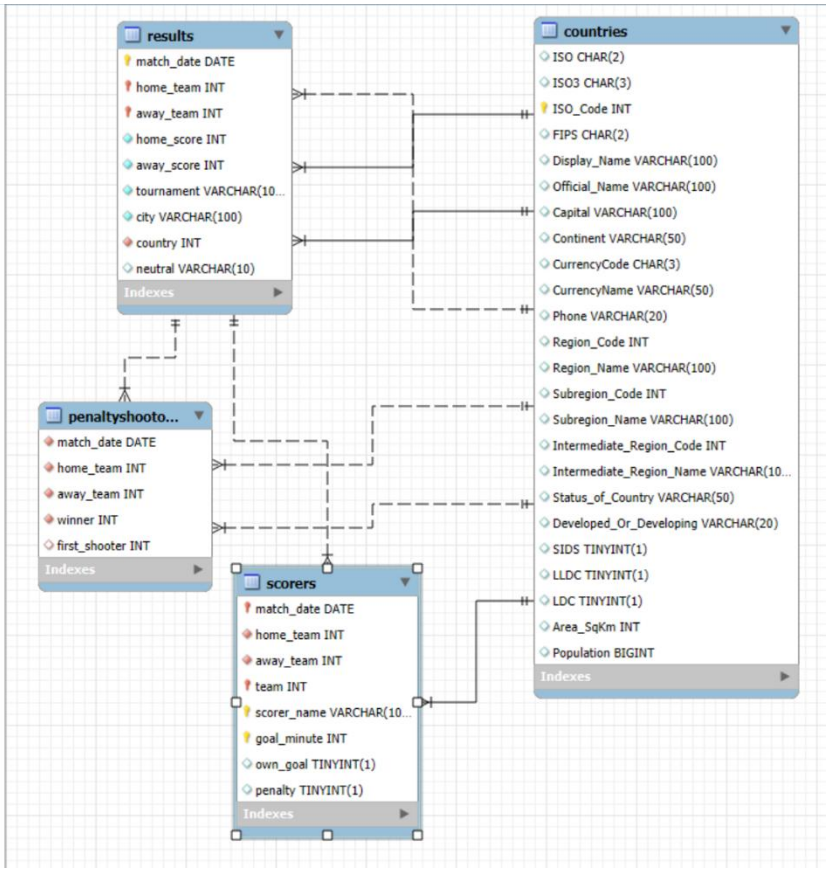
ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφέας
2025/05/27	1.0	Η πρώτη πλήρως λειτουργική κατάσταση της εφαρμογής.	Μιλτιάδης Χαλαϊδόπουλος Όμηρος Χατζηιορδάνης

1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Στην παρούσα ενότητα περιγράφονται τα σχήματα της βάσης δεδομένων που χρησιμοποιούνται στο project.

1.1 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΛΟΓΙΚΟ ΕΠΙΠΕΔΟ



Σχήμα 1.1 Σχεσιακό σχήμα της βάσης δεδομένων του συστήματος

```
--countries table

CREATE TABLE countries ( ISO CHAR(12) ,
ISO3 CHAR(13), ISO_Code INT PRIMARY KEY,
FIPS CHAR(12) NULL, Display_Name
VARCHAR(100), Official_Name VARCHAR(100),
Capital VARCHAR(100), Continent
VARCHAR(50), CurrencyCode CHAR(13) NULL,
CurrencyName VARCHAR(50) NULL, Phone
VARCHAR(20) NULL, Region_Code INT NULL,
Region_Name VARCHAR(100) NULL,
Subregion_Code INT NULL, Subregion_Name
VARCHAR(100) NULL, Intermediate_Region_Code
INT NULL, Intermediate_Region_Name
VARCHAR(100) NULL, Status_of_Country
VARCHAR(50), Developed_Or_Developing
VARCHAR(20) NULL, SIDS BOOLEAN NULL, LLDC
BOOLEAN NULL, LDC BOOLEAN NULL, Area_SqKm
INT, Population BIGINT );
```

```
--results table


CREATE TABLE results ( match_date DATE NOT
NULL, home_team INT NOT NULL, away_team INT
NOT NULL, home_score INT NOT NULL,
away_score INT NOT NULL, tournament
VARCHAR(100) NOT NULL, city VARCHAR(100)
NOT NULL, country INT NOT NULL, neutral
VARCHAR(10), PRIMARY KEY (match_date,
home_team, away_team), FOREIGN KEY
(home_team) REFERENCES countries(ISO_Code),
FOREIGN KEY (away_team) REFERENCES
countries(ISO_Code), FOREIGN KEY (country)
REFERENCES countries(ISO_Code) );


-- penaltyShootouts table

CREATE TABLE penaltyShootouts ( match_date
DATE NOT NULL, home_team INT NOT NULL,
```

```
away_team INT NOT NULL, winner INT NOT
NULL, first_shooter INT, PRIMARY KEY
(match_date,home_team,away_team), FOREIGN
KEY (match_date, home_team, away_team)
REFERENCES results(match_date, home_team,
away_team), FOREIGN KEY (winner) REFERENCES
countries(ISO_Code), FOREIGN KEY
(first_shooter) REFERENCES
countries(ISO_Code) );


-- scorers table
CREATE TABLE scorers ( match_date DATE NOT
NULL, home_team INT NOT NULL, away_team INT
NOT NULL, team INT, scorer_name
VARCHAR(100) NOT NULL, goal_minute INT NOT
NULL, own_goal BOOLEAN, penalty BOOLEAN,
PRIMARY KEY (goal_minute, scorer_name,
match_date,team), FOREIGN KEY (match_date,
home_team, away_team) REFERENCES
results(match_date, home_team, away_team),
FOREIGN KEY (team) REFERENCES
countries(ISO_Code) );
```

1.2 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ

Παρακάτω καταγράφονται οι ρυθμίσεις της βάσης σε φυσικό επίπεδο

1.2.1 ΡΥΘΜΙΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ DBMS

Το σύστημα της βάσης χρησιμοποιεί την InnoDb engine με buffer pool size στα 22,99MB (στα συστήματα που το δοκιμάσαμε να βάλουμε τη τιμή αυτή δεν ήταν δυνατό, οπότε μπήκε το ελάχιστο δυνατό μέγεθος αυτόματα).

1.2.2 ΡΥΘΜΙΣΗ ΤΟΥ ΦΥΣΙΚΟΥ ΣΧΗΜΑΤΟΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Για να βελτιωθεί η απόδοση χρησιμοποιούμε primary keys ως indexes. Πιο συγκεκριμένα το primary key ISO_Code του countries table χρησιμοποιείται ως ξένο κλειδί κάθε φορά που υπάρχει αναφορά σε χώρα στους άλλους 3 πίνακες. Έπειτα ο πίνακας results έχει για πρωτεύον κλειδί την τριάδα match_date,home_team,away_team το οποίο χρησιμοποιείται σαν ξένο κλειδί στους πίνακες με τα πέναλτι και με τους σκόρερ. Με όλα αυτά δεδομένα , δεν παρατηρήσαμε μεγάλη καθυστέρηση όταν τρέχαμε queries για αυτό δεν βρήκαμε σκόπιμο να περιπλέξουμε περεταίρω τον κώδικα. Παρ' όλα αυτά για βελτίωση περαιτέρω της ταχύτητας θα μπορούσαμε να προσθέσουμε indexes στα στην τριάδα match_date , home_team , away_team καθώς ίσως και κάποιο view με τα στατιστικά της.

2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ

2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΔΟΜΗ ETL

Η διαδικασία φόρτωσης των δεδομένων στην βάση χωρίστηκε σε 3 κομμάτια

1. Extract: Τα δεδομένα προέχονται από τα csv αρχεία στο kaggle

2. Transform : Επεξεργασία των δεδομένων με python scripts και την βιβλιοθήκη pandas έτσι ώστε να ταιριάζουν στις ανάγκες της εφαρμογής , πιο συγκεκριμένα οι αλλαγές ήταν οι εξής :

I. Στον στο αρχείο Countries :

- Προστέθηκαν αυτόματα εγγραφές για να αντιπροσωπεύουνε όλες τις χώρες που υπήρχαν στον πίνακα results
- Τροποποιήθηκαν κατάλληλα τα πεδία SIDS,LLDC,LDC ώστε να μπορούν να περαστούν Boolean τιμές
- Αφαιρέθηκαν οι Null τιμές από κάθε πεδίο

II. Στον αρχείο results

- Αλλαγή ονομάτων χωρών με τα ISO_CODE τους
- Αφαίρεση εγγραφών με ίδια κλειδιά (match_date,home,away_team)

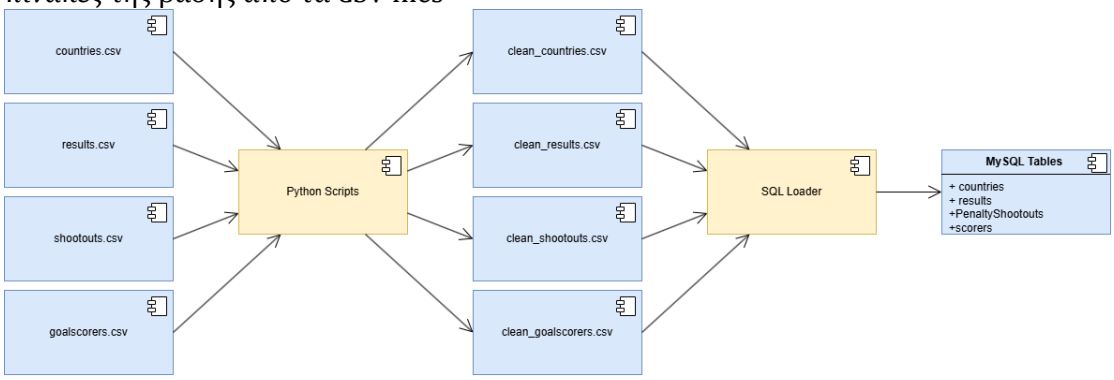
III. Στον αρχείο goalscorers:

- Αλλαγή ονομάτων χωρών με τα ISO_CODE τους
- Κατάλληλη τροποποίηση των πεδίων own_goal,penalty ώστε να μπορούν να περαστούν Boolean τιμές

IV. Στον πίνακα Penalty shootouts:

- Αλλαγή ονομάτων χωρών με τα ISO_CODE τους

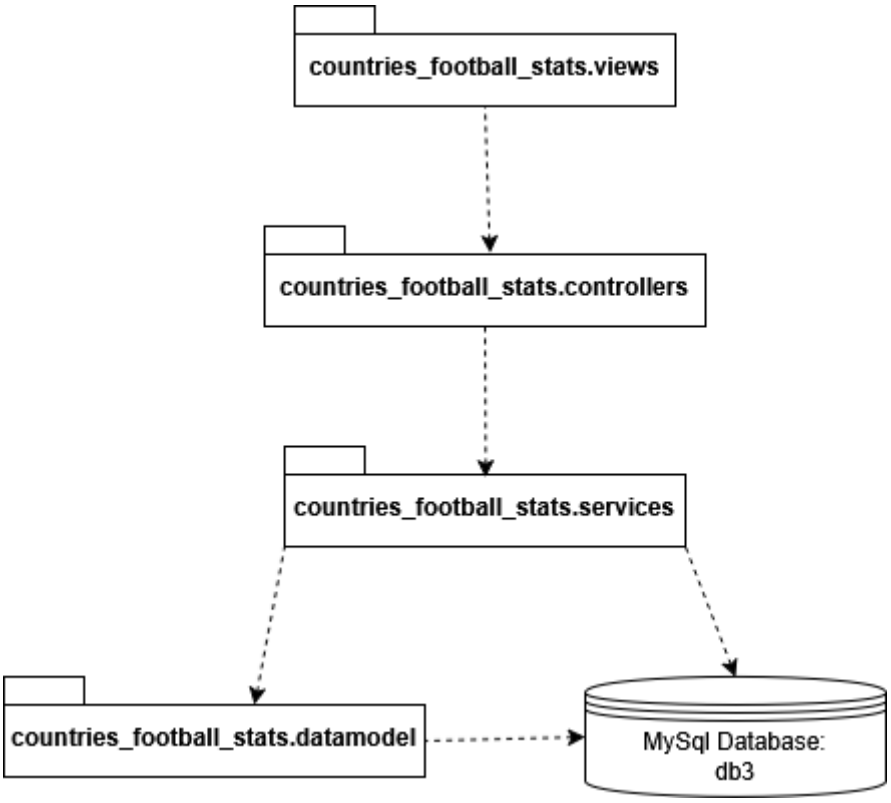
3.Load: Φορτώνουμε τα επεξεργασμένα δεδομένα στην βάση με χρήση απλών LOAD DATA LOCAL INFILE οι οποίες επιτρέπουν την μαζική εισαγωγή πλειάδων στους πίνακες της βάσης από τα CSV files



Σχήμα 2.1 UML component diagram

2.2 ΔΙΑΓΡΑΜΜΑΤΑ ΠΑΚΕΤΩΝ / ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

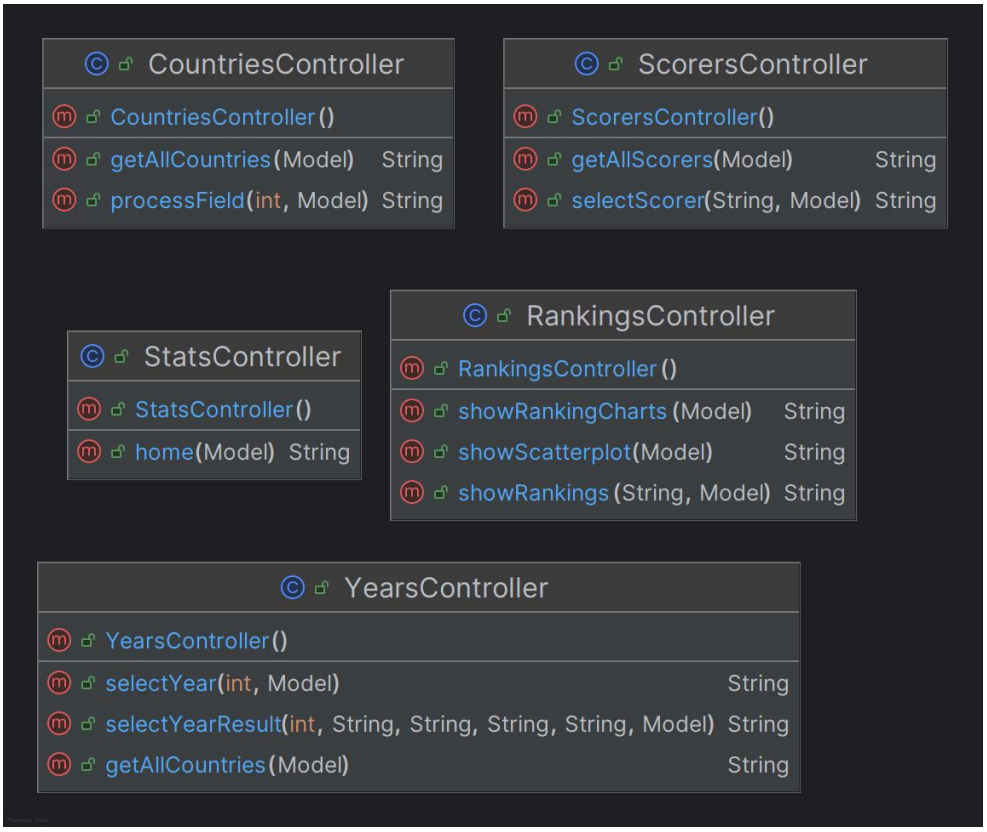
Το διάγραμμα για τα υποσυστήματα / πακέτα του λογισμικού που κατασκευάσατε ως κεντρική εφαρμογή επερώτησης. Ο στόχος είναι να φανεί η high-level αρχιτεκτονική του συστήματος, χωρίς λεπτομέρειες των επί μέρους κλάσεων. Κάποιος πολύ σύντομος σχολιασμός επίσης.



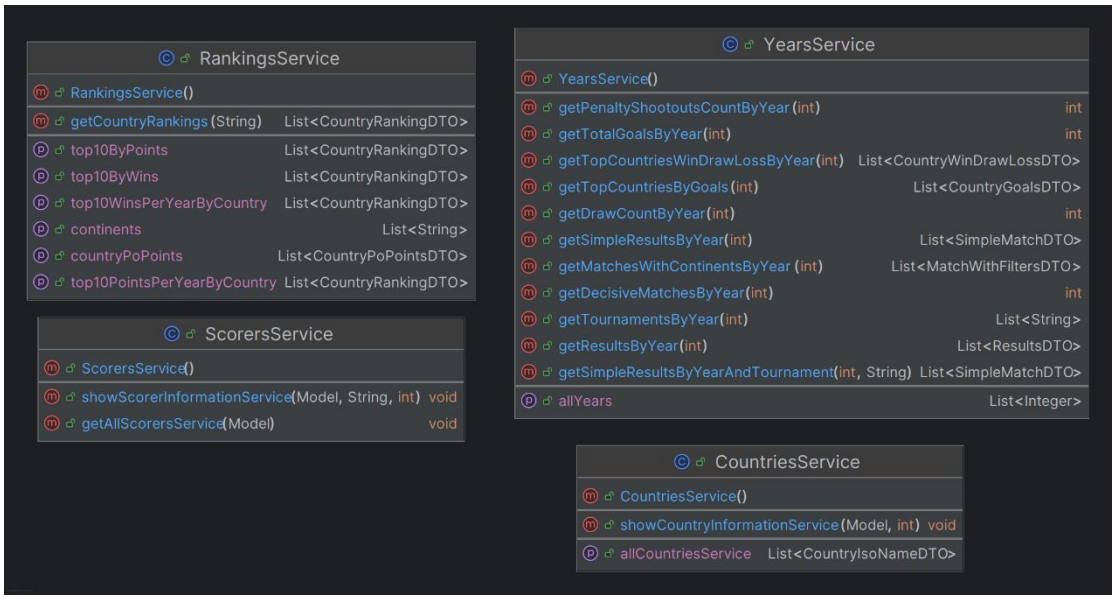
Σχήμα 2.1

Η εφαρμογή χρησιμοποιεί πολλαπλά επίπεδα. Το πακέτο views αποτελεί το κομμάτι αλληλεπίδρασης με τον χρήστη, εδώ παρουσιάζονται δεδομένα και γίνεται χειρισμός αιτημάτων από τους χρήστες ή στο ίδιο επίπεδο ή στο επόμενο. Τα controllers συνδέουν τα views με το business logic κομμάτι (δηλαδή τα services) και διαχειρίζονται αιτήματα από αυτά. Τα services είναι υπεύθυνα να καλέσουν και να αναθέσουν στα datamodel που υπάρχουν, δεδομένα από τη βάση δεδομένων, και να τα δώσουν όλα αυτά στο controller για να παρουσιαστούν στα views, σύμφωνα με το ανάλογο αίτημα. Τα datamodel αποτελούν αναπαραστάσεις δεδομένων από τη βάση, τις οποίες μπορούν να τις διαχειριστούν τα υπόλοιπα επίπεδα. Τέλος υπάρχει η βάση η οποία κρατάει όλα τα δεδομένα για την εφαρμογή.

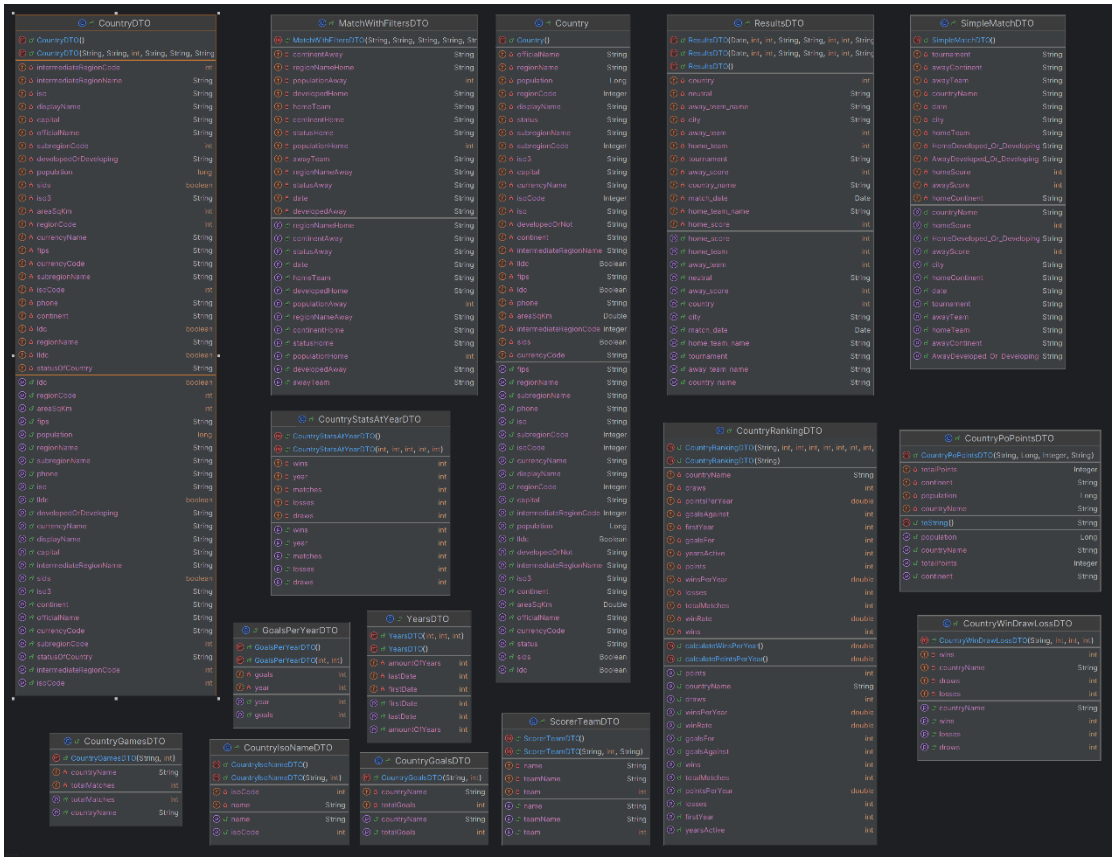
2.3 ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ



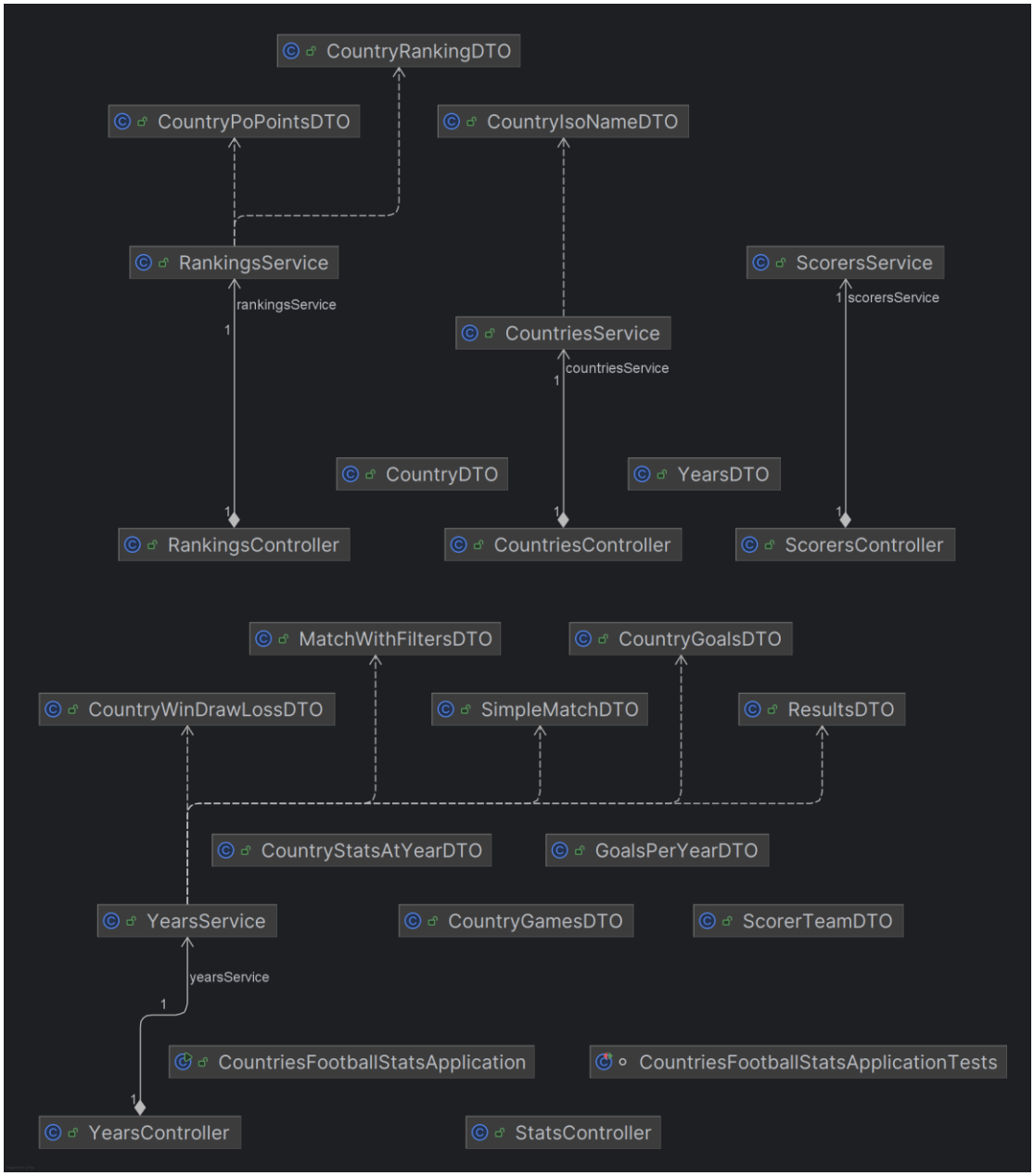
Controller package class diagram



Service package class diagram



datamodel package class diagram



class relations diagram

Η εφαρμογή μας όπως περιεγράφηκε παραπάνω αποτελείται κυρίως από κλάσεις controller, datamodel και service.

Οι controller κλάσεις είναι υπεύθυνες για τη διαχείριση αιτημάτων ενός χρήστη μέσω url mappings. Η κάθε μέθοδος ενός controller αντιστοιχεί σε μια κατάλληλη μέθοδο get από τα views, και καλεί την κατάλληλη service μέθοδο για να φορτώσει τα δεδομένα που πρέπει να εμφανιστούν στο view, και να φορτωθούν στο model που κουβαλάει ο controller.

Οι services κλάσεις είναι υπεύθυνες να μόνο να δίνουν στο DBMS τα κατάλληλα ερωτήματα SQL και να τοποθετήσουν τα αποτελέσματα τους στα κατάλληλα DTOs ή απλές μεταβλητές άμα χρειαστεί, και αυτά με τη σειρά τους να τοποθετηθούν μέσα στο model. Οι service κλάσεις δεν κάνουν καμία επεξεργασία στα δεδομένα, ούτε κάποιον έλεγχο μέσα στο περιβάλλον της java. Όλα αυτά έχουν αφεθεί επάνω στο DBMS.

Στο πρόγραμμα υπάρχουν κυρίως τέσσερις κατηγορίες στατιστικών που μπορεί να δει και να μελετήσει ένας χρήστης. Οπότε έχουν φτιαχτεί για την κάθε κατηγορία, μία αντίστοιχη controller και μία αντίστοιχη service κλάση, και ένα controller για την κεντρική σελίδα. Τα controller και τα services της μίας κατηγορίας αλληλοεπιδρούν μεταξύ τους, όμως ως προς άλλες κατηγορίες παραμένουν απομονωμένα.

Η κύρια αναπαράσταση των δεδομένων γίνεται από data transfer objects. Όταν επρόκειτο για απάντηση σε ερώτημα SQL που περιέχει πολλαπλές στήλες, που σημαίνει ότι δεν μπορεί μία πλειάδα να τοποθετηθεί σε μία primitive μεταβλητή, χρησιμοποιείται το κατάλληλο DTO που περιέχει πεδία να αναπαριστούν τη κάθε στήλη. Έτσι γίνεται η επεξεργασία μόνο στο DBMS, και τα δεδομένα μπορούν απλά να αναπαρασταθούν στα views και δίνει τη δυνατότητα μιας εύκολης επαναχρησιμοποίησης ή επέκτασης σε περίπτωση εμπλουτισμού της εφαρμογής. Τα περισσότερα DTOs χρησιμοποιούνται σε μία από τις τέσσερις κατηγορίες, για την κατάλληλη αναπαράσταση στα αντίστοιχα views. Όμως υπάρχουν και κάποια όπως το Results DTO τα οποία χρησιμοποιούνται από πολλαπλές κατηγορίες.

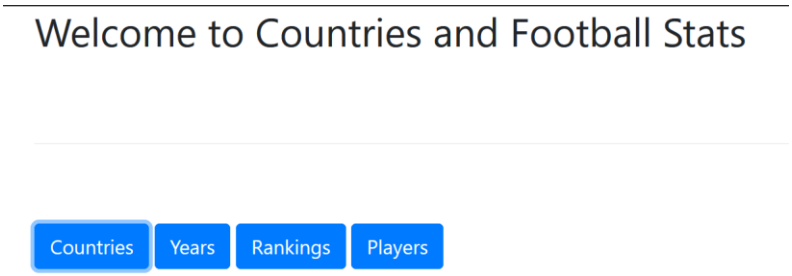
2.4 VIEWS ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Το frontend της εφαρμογής υλοποιείται μέσω αρχείων html εμπλουτισμένα με thymeleaf και javascript. Το thymeleaf προσφέρει ευκολότερη διαχείριση της παρουσίασης και της αποστολής στοιχείων και δεδομένων μέσα στο model που έχει δωθεί από τις controller κλάσεις. Για μια πιο εμπλουτισμένη και πιο διαδραστική αλληλεπίδραση και αναπαράσταση, χρησιμοποιήθηκαν ορισμένα script της javascript, είτε για το πιθανό φιλτράρισμα σε πίνακες δεδομένων ή, με τη βοήθεια της βιβλιοθήκης chart.js, για τη δημιουργία γραφικών παραστάσεων όπως linecharts ή scatterplots.

Όλα τα html views συνδέονται με τους controllers μέσω κατάλληλων url.

3 ΥΠΟΔΕΙΓΜΑΤΑ ΕΡΩΤΗΣΕΩΝ ΚΑΙ ΑΠΑΝΤΗΣΕΩΝ

Έστω θέλουμε να εμφανίσουμε τα στατιστικά για μια συγκεκριμένη χώρα (πχ Γερμανία) ξεκινώντας από την αρχική οθόνη θα μεταβούμε Countries.



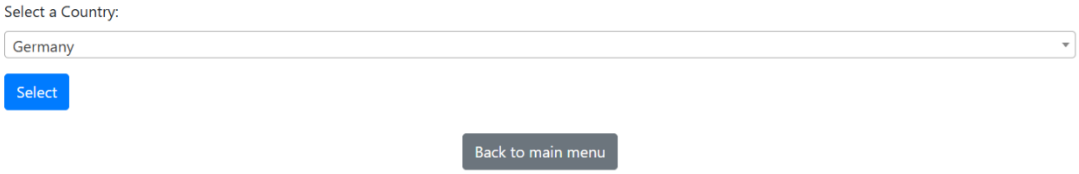
Ο controller για την σελίδα αυτή καλεί την μέθοδο από το service getAllCountriesService()

```
@GetMapping("/all_countries")
public String getAllCountries(Model model) {
    model.addAttribute("countries", countriesService.getAllCountriesService());
    return "countries/all_countries";
}
```

αυτή η μέθοδος εκτελεί ένα query το οποίο επιστρέφει όλες τις χώρες που έχουν display name και Iso_Code

```
public List<CountryIsoNameDTO> getAllCountriesService(){
    return jdbcTemplate.query(
            "SELECT Display_Name, ISO_Code FROM countries",
            (rs, rowNum) -> new CountryIsoNameDTO(
                rs.getString("Display_Name"),
                rs.getInt("ISO_Code")
            )
        );
}
```

Βλέποντας όλες αυτές τις χώρες μπορούμε να επιλέξουμε μία από το dropdown menu ή πληκτρολογώντας την (στην περίπτωση μας Germany)



Αυτή την φορά ο controller θα καλέσει την μέθοδο του sevice showCountryInformationService() για το ISO_Ccde της χώρας που επιλέξαμε .

```
@GetMapping("/select_one")
public String processField(@RequestParam int isoCode, Model model) {
    System.err.println(isoCode);
    countriesService.showCountryInformationService(model, isoCode);
    return "countries/country_information";
}
```

Η μέθοδος από το service φτιάχνει ένα DTO για χώρες καλεί διαδοχικά τα εξής queries

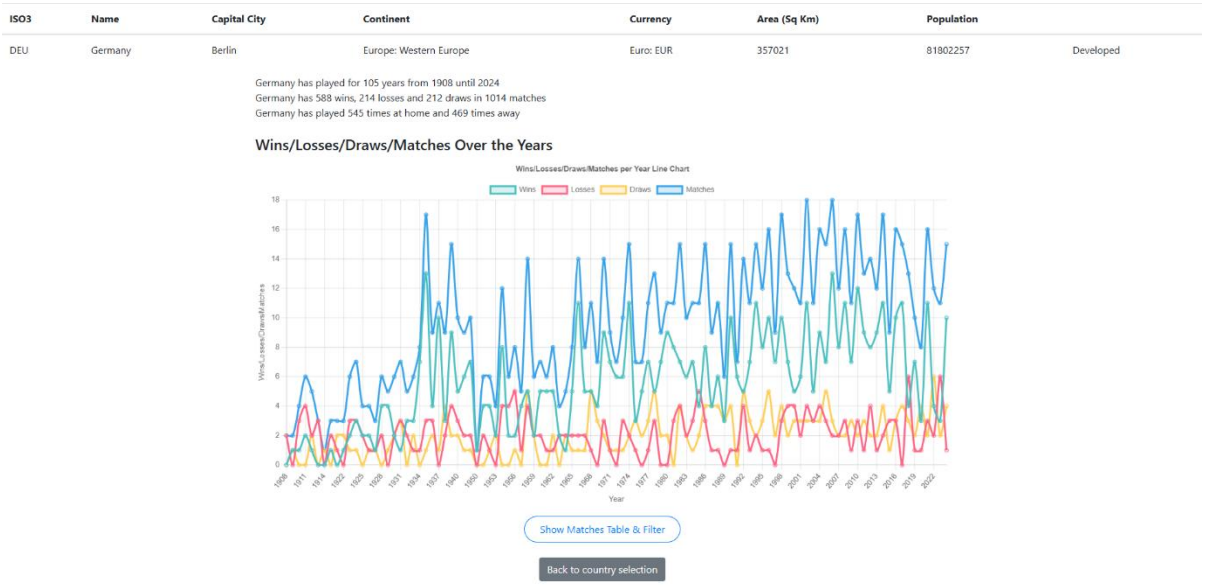
SELECT MIN(YEAR(m.match_date)) AS firstYear, MAX(YEAR(m.match_date)) AS lastYear, COUNT(DISTINCT YEAR(m.match_date)) AS totalYears FROM results m WHERE m.home_team = ? OR m.away_team = ?
(Επιστρέφει το πρώτο και τελευταίο έτος που έπαιξε μια χώρα και πόσα διαφορετικά έτη συμμετείχε)

(SELECT COUNT() FROM results WHERE home_team = ? AND home_score > away_score) + (SELECT COUNT() FROM results WHERE away_team = ? AND away_score > home_score)
(υπολογίζει τις συνολικές νίκες της χώρας (και ως home team και ως away))
Παρόμοια queries τρέχει για να υπολογίσει και τις ήττες και τις ισπαλίες

Επιπλεόν υπολογίζει το άθροισμα των ματς ματς με την
(SELECT COUNT() FROM results WHERE home_team = ?)+(SELECT COUNT() FROM results WHERE away_team = ?)

Τέλος βρίσκει τα ματς εκτός και εντός με :
SELECT COUNT() FROM results WHERE home_team = ? SELECT COUNT() FROM results WHERE away_team = ?

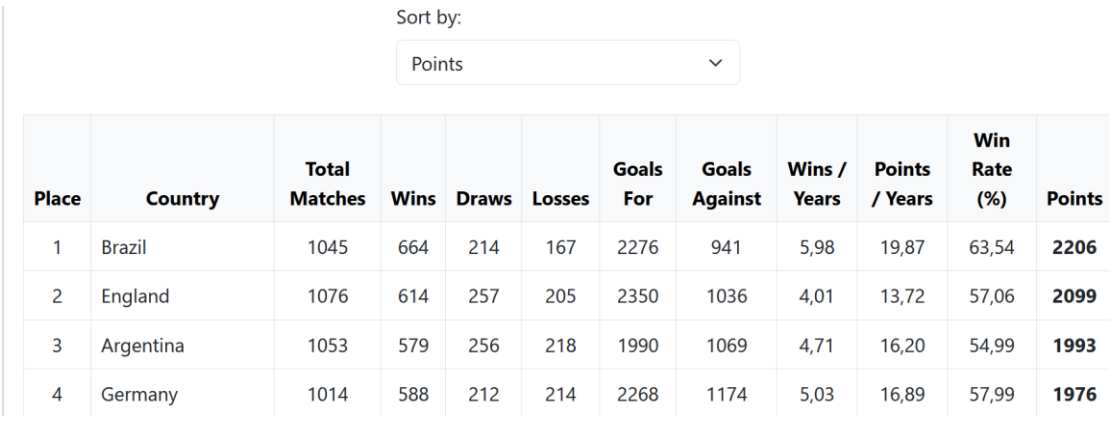
Αυτά τα δεδομένα με την σειρά τους χρησιμοποιούνται από ένα view το country information.html και έτσι εμφανίζεται το τελικό αποτέλεσμα :



Παρόμοια μπορεί ο χρήστης να περιηγηθεί σε άλλες κατηγορίες στατιστικών, με φυσικά διαφορετικά δεδομένα επιλογής και με διαφορετικούς τρόπους παρουσίασης τους.

Σε άλλες κατηγορίες περιέχονται και επιλογές που μπορεί να αλλάζουν τη παρουσίαση των δεδομένων.

Όπως στις καθολικές κατατάξεις των χωρών μέσω της επιλογής Rankings. Εκεί εμφανίζεται ένας πίνακας με τις χώρες ταξινομημένες βάση ενός συγκεκριμένου στατιστικού, το οποίο μπορεί να επιλέξει ο χρήστης και λόγω του html script να αλλάξει η ταξινόμηση.



Place	Country	Total Matches	Wins	Draws	Losses	Goals For	Goals Against	Wins / Years	Points / Years	Win Rate (%)	Points
1	Brazil	1045	664	214	167	2276	941	5,98	19,87	63,54	**2206**
2	England	1076	614	257	205	2350	1036	4,01	13,72	57,06	**2099**
3	Argentina	1053	579	256	218	1990	1069	4,71	16,20	54,99	**1993**
4	Germany	1014	588	212	214	2268	1174	5,03	16,89	57,99	**1976**

Sort by:

Total Matches

Place	Country	Total Matches	Wins	Draws	Losses	Goals For	Goals Against	Wins / Years	Points / Years	Win Rate (%)	Points
1	Ireland	1300	416	326	564	1555	2018	2,91	11,01	32,00	**1574**
2	Sweden	1081	535	230	316	2130	1382	4,57	15,68	49,49	**1835**
3	England	1076	614	257	205	2350	1036	4,01	13,72	57,06	**2099**
4	Argentina	1053	579	256	218	1990	1069	4,71	16,20	54,99	**1993**

Τέλος, περιέχονται πάντα κουμπιά επιστροφής σε προηγούμενη σελίδα ή στην αρχική, και πολλές φορές σε μια κατηγορία υπάρχουν κι άλλες σελίδες με παρουσιάσεις, που είναι προσβάσιμες από την αρχική τους σελίδα.

4 ΛΟΙΠΑ ΣΧΟΛΙΑ

Η εφαρμογή είναι ένα springboot project και έχει αναπτυχθεί σε περιβάλλον eclipse. Για την εκκίνηση της, πρέπει να τρέξει το αρχείο CountriesFootballStatsApplication.java ως εφαρμογή Java (δεξί κλικ -> Run As -> Java Application).

Δεν έχει γίνει έλεγχος σε άλλα IDEs, οπότε μπορεί να προκύψουν προβλήματα αν η εφαρμογή τρέξει σε άλλο πρόγραμμα.

Συστήνεται να είναι ανοιχτό το mysql workbench κατά τη διάρκεια της εφαρμογής και να έχει γίνει η κατάλληλη σύνδεση.

Τα στοιχεία της σύνδεσης πρέπει να εισαχθούν στο αρχείο application properties, μέσα στο φάκελο resources, αλλάζοντας τα στοιχεία spring.datasource.username και spring.datasource.password κατάλληλα ανάλογα τον χρήστη.

Απαραίτητη είναι η δημιουργία των tables όπως έχουν δοθεί στα sql scripts, ώστε να μπορεί να λειτουργήσει η εφαρμογή.

Για να χρησιμοποιηθεί η εφαρμογή, πρέπει να πληκτρολογηθεί **localhost:8080/** στη μπάρα αναζήτησης του περιηγητή.