

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ  
ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ  
ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

---

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ  
ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2022-2023

ΟΜΑΔΑ ΤΟΥΜΠΑΝΙΑΣΜΕΝΟΙ

---

---

ΓΙΑΝΝΙΤΣΑΚΗΣ ΔΗΜΗΤΡΙΟΣ, 4338

ΣΙΝΤΟΣ ΔΗΜΗΤΡΙΟΣ, 4012

ΚΑΛΥΒΙΩΤΗΣ ΑΘΑΝΑΣΙΟΣ, 4607

---

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΑΪΟΣ 2023

## ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

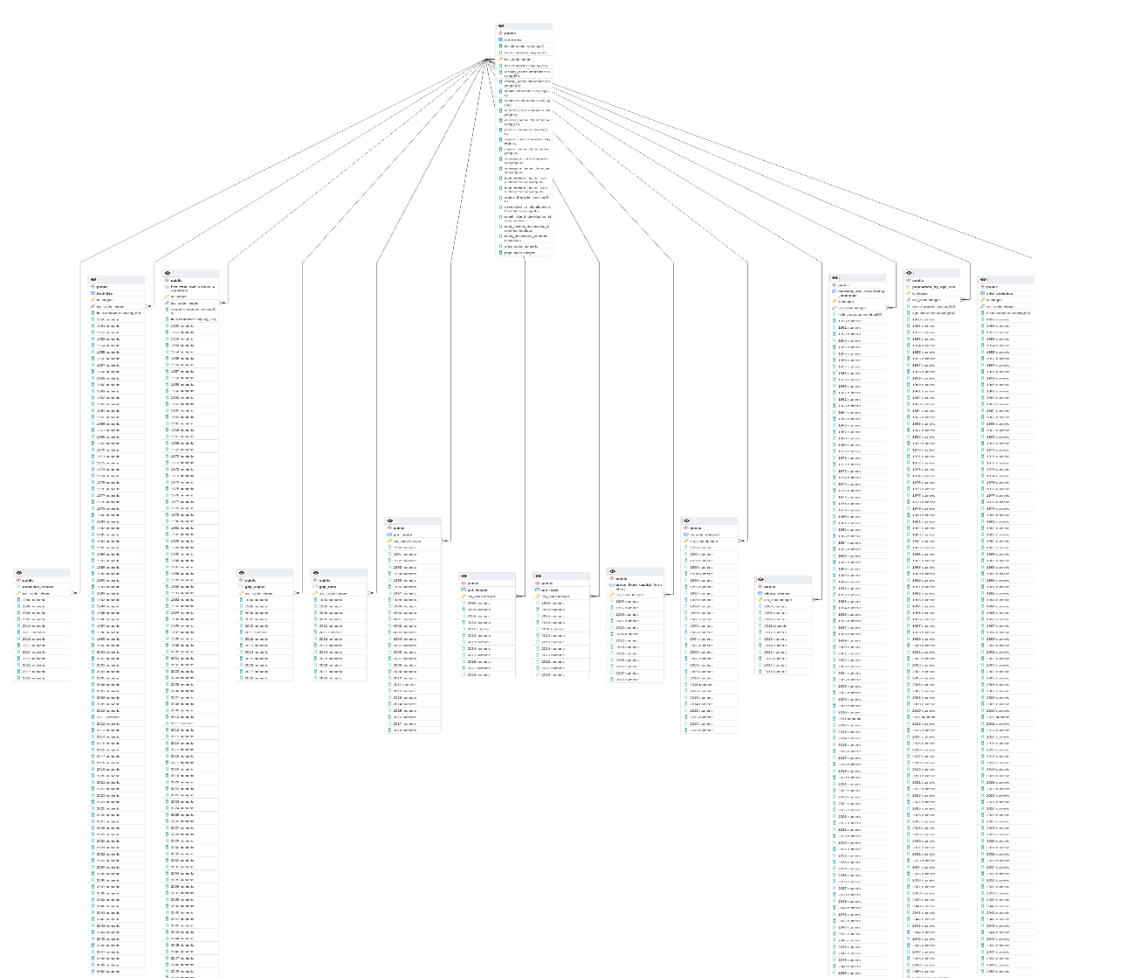
Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφέας
2023/05/30	v.1	Τελική αναφορά	Σιντος Δημήτριος, Γιαννιτσάκης Δημήτριος, Καλυβιώτης Αθανάσιος

Το κείμενο συμπληρώνεται προοδευτικά, όπως προχωρείτε στις φάσεις του Project.

## 1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

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

### 1.1 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΛΟΓΙΚΟ ΕΠΙΠΕΔΟ



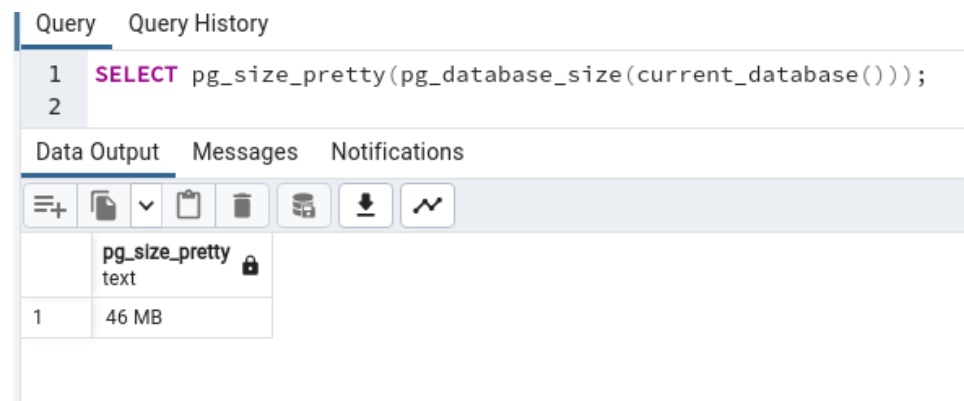
Σχήμα 1.1 Σχεσιακό σχήμα της βάσης δεδομένων του συστήματος

### 1.2 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ

Όταν θα έχετε στήσει και ρυθμίσει τη βάση δεδομένων σας, εδώ καταγράφονται και οι ρυθμίσεις σε φυσικό επίπεδο. Ενδεικτικά:

#### 1.2.1 ΡΥΘΜΙΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ DBMS

Κάνοντας `cd /etc/postgresql/14/main/` και ανοίγοντας το αρχείο `postgresql.conf` στο πεδίο `shared_buffers` (αντίστοιχη μεταβλητή με `data buffers` στην `Mysql`) βλέπουμε ότι είναι σεταρισμένω `by default` στα 128MB. Η βάση μας έχει μέγεθος 46 MB.

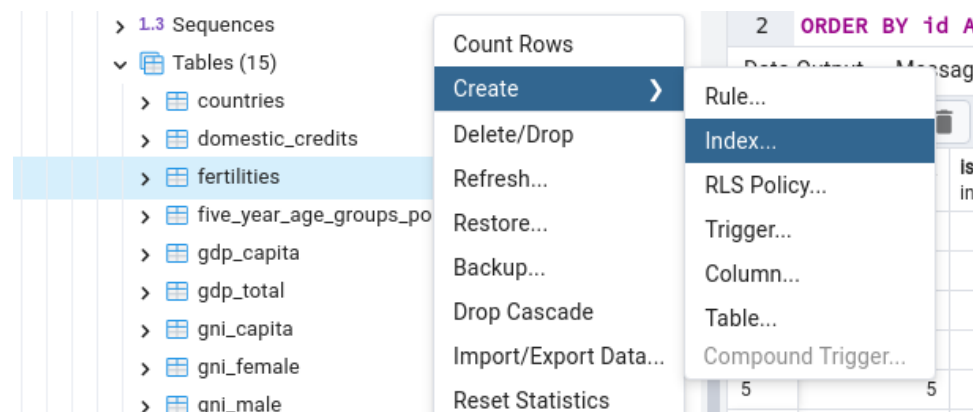


Οπότε θα μπορούσαμε να επιλέξουμε ένα μικρότερο μέγεθος. Δηλαδή το μέγεθος της βάσης συν το 10%.

### 1.2.2 ΡΥΘΜΙΣΗ ΤΟΥ ΦΥΣΙΚΟΥ ΣΧΗΜΑΤΟΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Σε όλους τους πίνακες θέσαμε το `iso_code` ως `index`. Ο λόγος είναι επειδή στα `queries` που κάνουμε προς την βάση στα `WHERE` clause έχουμε συνθήκες όπως `"iso_code = "` και `"iso_code in ()"`. Τα βήματα που ακολουθήσαμε στο `pgAdmin`:

1)



2)

**Create - Index**

General Definition SQL

Name: iso\_code\_index

Tablespace: Select an item...

Comment:

**! You must specify at least one column.**

Close Reset Save

3)

**Create - Index**

General Definition SQL

Access Method: btree

Fill factor:

Unique?: ☐

Clustered?: ☐

Concurrent build?: ☐

Constraint: 1

**Columns**

Column	Operator class	Sort order	NULLs	Collation
iso_code	Select an item...	ASC	LAST	Select an item...

Include columns: Select the column(s)

Close Reset Save

4)



Με αυτόν τον τρόπο το μέγεθος της βάσης αυξάνεται αλλά κερδίζουμε σε ταχύτητα.

### 1.2.3 ΡΥΘΜΙΣΗ ΑΣΦΑΛΕΙΑΣ

Για να περάσουμε τα δεδομένα στην βάση δημιουργήσαμε τον χρήστη “myuser”. Στον χρήστη δώσαμε όλα τα PRIVILEGES (Superuser, Create role, Create DB, Replication, Bypass RLS). Αυτό γίνεται αυτοματοποιημένα στο script που δημιουργεί την βάση.

```
CountriesApplication.java  createDB.sh  csvToPostgres.py
Scripts > $ createDB.sh
1  #!/bin/bash
2
3  # Step 1: Start psql as the postgres user
4  sudo -u postgres psql <<EOF
5  DROP DATABASE IF EXISTS MYE030;
6  CREATE DATABASE MYE030;
7  CREATE USER myuser WITH PASSWORD 'mye030';
8  GRANT ALL PRIVILEGES ON DATABASE MYE030 TO myuser;
9  EOF
```

Και στο script που πάμε να δημιουργήσουμε τα tables συνδεόμαστε με αυτόν τον χρήστη.

```

countriesApplication.java  createDB.sh  csvToPostgres.py x
cripts > csvToPostgres.py > ...
4 from createIncomeTables import create_income_tables
5 from createIncomeTables import create_income_tables
6
7
8 def connect_to_db():
9     # define the connection parameters
10    conn = psycopg2.connect(
11        host="localhost",
12        database="mye030",
13        user="myuser",
14        password="mye030"
15    )
16    conn.autocommit = True
17
18    return conn
19

```

Για το backend της εφαρμογής μας δημιουργήσαμε ένα νέο χρήστη myuser3 για να κάνει της ερωτήσεις προς την βάση.

Αρχικά ανοίξαμε το εργαλείο psql ως myuser : psql -U myuser -d mye030 και του δώσαμε μόνο δικαιώματα για select.

```

=# create user myuser3 with login password '12345678';

```

```

postgres=# grant connect on database mye030 to myuser3;
GRANT
postgres=# grant usage on schema public to myuser3;
GRANT

```

```

mye030=# grant select on all tables in schema public to myuser3;
GRANT

```

Τέλος το αρχείο application.properties:

```

J countriesApplication.java  createDB.sh  csvToPostgres.py  application.properties M x  presentation_link.tx
web app > countries > src > main > resources > application.properties
1 # JDBC properties
2
3 spring.datasource.url=jdbc:postgresql://localhost:5432/mye030?useSSL=false&serverTimezone=UTC
4 spring.datasource.username=myuser3
5 spring.datasource.password=12345678
6 spring.datasource.driver-class-name=org.postgresql.Driver
7 spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
8

```

## 1.2.4 BACK UP AND RESTORE



Για το back up τρέχουμε την εντολή **pg\_dump -d mye030 -h localhost -U myuser -n public > backup.sql**

Για το recovery τρέχουμε την εντολή **psql dbname < backup.sql** όπου το dbname είναι το όνομα της νέας βάσης.

Περισσότερες πληροφορίες θα βρείτε στον σύνδεσμο:

<https://www.postgresql.org/docs/8.0/backup.html#BACKUP-DUMP-RESTORE>

## 2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ

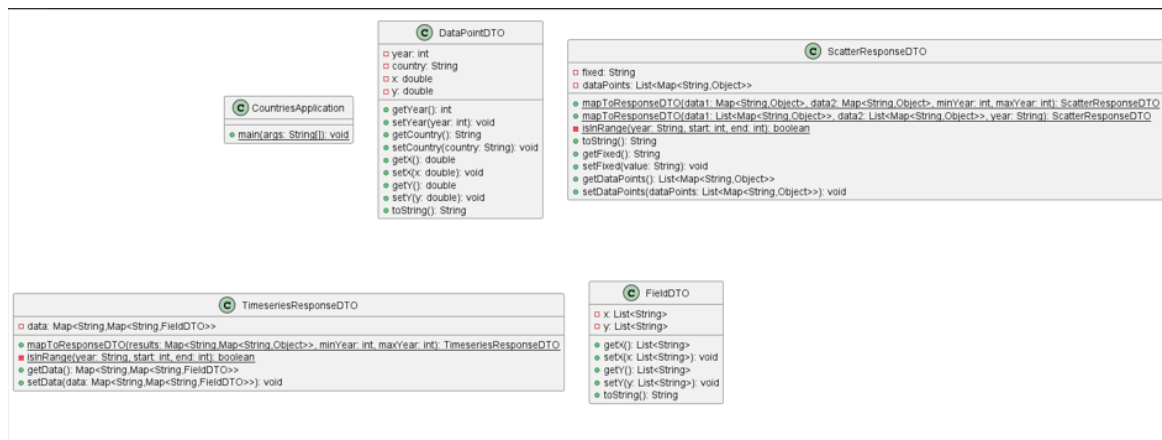
### 2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΔΟΜΗ ETL

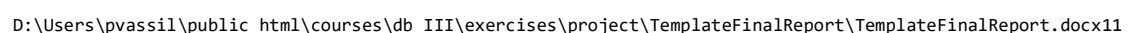
Για το φόρτωμα των csv στην βάση scripts σε Python. Τα δημογραφικά δεδομένα χρειάστηκε να γίνουν transform (τα year να γίνουν columns) για να γίνει αυτό χρησιμοποιήσαμε τα Dataframes της βιβλιοθήκης pandas και έπειτα δημιουργούμε νέα csv αρχεία (π.χ mortality\_life\_updated.csv) τα οποία τα φορτώνουμε με την εντολή copy\_from της βιβλιοθήκης pyscopg2 η οποία στο background εκτελεί την εντολή COPY ... FROM (αντίστοιχη LOAD DATA της My sql).

Αυτό είναι μια ακριβή διαδικασία σε χρόνο, καθώς για τα μεγάλα csv αρχεία το trasformation παίρνει αρκετή ώρα. Αυτό το κόστος το πληρώσαμε μια φορά αφού όταν δημιουργηθούν τα updated csv άμα τρέξουμε ξανά τα scripts η βάση δημιουργείται σε ελάχιστα δευτερόλεπτα.

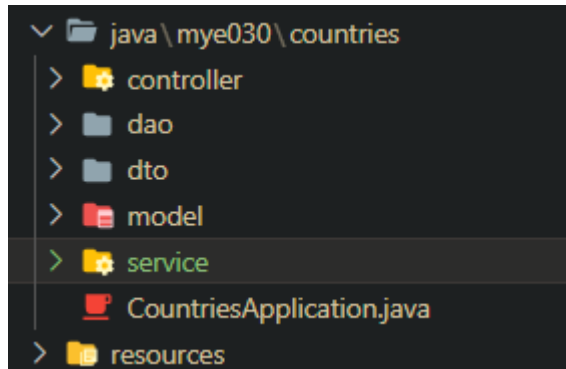
### 2.2 ΔΙΑΓΡΑΜΜΑΤΑ ΠΑΚΕΤΩΝ / ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

### 2.3 ΔΙΑΓΡΑΜΜΑΤΑ ΚΛΑΣΕΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ





Παραπάνω φαίνεται το διάγραμμα κλάσεων της κεντρικής εφαρμογής. Η κεντρική εφαρμογή είναι υλοποιημένη στο framework Spring Boot σε Java. Όσον αφορά την αρχιτεκτονική της εφαρμογής, αυτή αποτελείται από τα πακέτα service, dao, dto, controller και model.



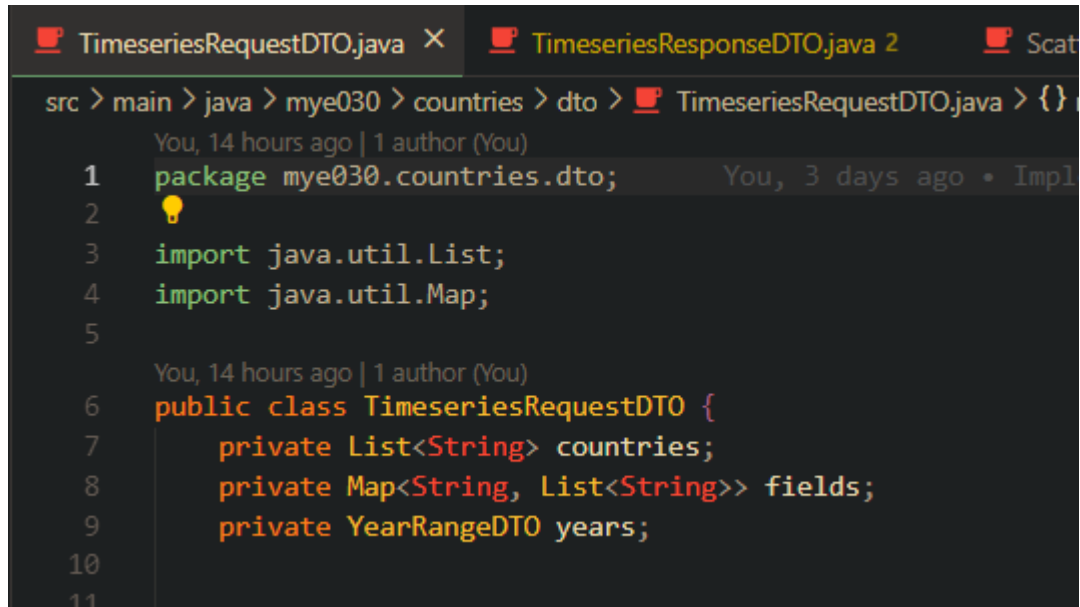
Πιο συγκεκριμένα, στο πακέτο controller διαχειριζόμαστε τα request της κεντρικής σελίδας της web εφαρμογής και τις διάφορες κλήσεις προς το api για το fetching των δεδομένων για την παραγωγή διαγραμμάτων.

```
ApiController.java X
src > main > java > mye030 > countries > controller > ApiController.java > {} mye030.countries.controller
You, 14 hours ago | 1 author (You)
1 package mye030.countries.controller;
2
3
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PostMapping;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10 import mye030.countries.dto.TimeseriesRequestDTO;
11 import mye030.countries.dto.TimeseriesResponseDTO;
12 import mye030.countries.dto.ScatterRequestDTO;
13 import mye030.countries.dto.ScatterResponseDTO;
14 import mye030.countries.service.countryService;
15
16 @RestController
17 @RequestMapping("/api")
18 public class ApiController {
19     @Autowired
20     private countryService countryService;
21
22     @PostMapping("/getTimeseriesData")
23     public ResponseEntity<TimeseriesResponseDTO> getTimeseriesData(@RequestBody TimeseriesRequestDTO request) {
24
25         // Process the request and generate the response
26         TimeseriesResponseDTO response = countryService.processTimeseriesRequest(request);
27         return ResponseEntity.ok(response);
28     }
29
30     @PostMapping("/getScatterData")
31     public ResponseEntity<ScatterResponseDTO> getScatterData(@RequestBody ScatterRequestDTO request) {
32
33         // Process the request and generate the response
34         ScatterResponseDTO response = countryService.processScatterRequest(request);
35         return ResponseEntity.ok(response);
36     }
37
38
39
40 }
41
```

Εδώ βλέπουμε την κλάση ApiController για την διαχείριση των κλήσεων του API.

Δύο είναι τα url για το fetching το /api/getTimeseriesData για διαγράμματα χρονοσειρών ή bar chart και το /api/getScatterData για τα scatter plots.

Στο πακέτο DTO μπορούμε να διακρίνουμε κάποιες κλάσεις για την δημιουργία object που σκοπό έχουν την μετατροπή των JSON Requests σε αντικείμενα για να μπορούμε να τα χρησιμοποιήσουμε και να τα διαχειριστούμε μέσα στον κώδικα. Υπάρχουν Data Transfer Objects που αναφέρονται σε αιτήματα για δεδομένα διαγραμμάτων χρονοσειράς και scatterplots αντίστοιχα, ενώ και DTOs που αναφέρονται στις απαντήσεις του Server με τα δεδομένα αυτά.



```
src > main > java > mye030 > countries > dto > TimeseriesRequestDTO.java > {}  
You, 14 hours ago | 1 author (You)  
1 package mye030.countries.dto;  
2  
3 import java.util.List;  
4 import java.util.Map;  
5  
6 public class TimeseriesRequestDTO {  
7     private List<String> countries;  
8     private Map<String, List<String>> fields;  
9     private YearRangeDTO years;  
10  
11
```



```
src > main > java > mye030 > countries > dto > ScatterRequestDTO.java > {} mye030.countries.dto  
You, 14 hours ago | 1 author (You)  
1 package mye030.countries.dto;  
2  
3 import java.util.List;  
4 import java.util.Map;  
5  
6 public class ScatterRequestDTO {  
7  
8     private String fixedField;  
9     private Map<String, String> xField;  
10    private Map<String, String> yField;  
11    private List<String> countries;  
12    private YearRangeDTO years;  
13
```

```
TimeseriesRequestDTO.java  TimeseriesResponseDTO.java 2 X  ScatterRequestDTO.java  ScatterResponseDTO.java

src > main > java > mye030 > countries > dto > TimeseriesResponseDTO.java > TimeseriesResponseDTO

You, 14 hours ago | 1 author (You)
1 package mye030.countries.dto;
2
3 import java.util.ArrayList;
4 import java.util.HashMap;
5 import java.util.List;
6 import java.util.Map;
7
8
9 You, 14 hours ago | 1 author (You)
10 public class TimeseriesResponseDTO {
11     private Map<String, Map<String, FieldDTO>> data;
12
13     You, 3 days ago * Implement timeseries data fetch api
14
15     // mapper method to turn a List<Map<String, Object>> into a ResponseDTO
16     public static TimeseriesResponseDTO mapToResponseDTO(Map<String, Map<String, Object>> results, int minYear, int maxYear) {
17         if (results.isEmpty()) {
18             // Handle empty result case if needed
19             return null;
20         }
21
22         TimeseriesResponseDTO responseDTO = new TimeseriesResponseDTO();
23         Map<String, Map<String, FieldDTO>> data = new HashMap<>();
24
25         // go through each entry of the results map
26         for (Map.Entry<String, Map<String, Object>> country : results.entrySet()) {
27             // extract the metric name and the corresponding results
28             String isoCode = country.getKey();
29             Map<String, Object> countryData = country.getValue();
30
31             // add new entry for the current iso_code
32             if (!data.containsKey(isoCode)) {
33                 data.put(isoCode, new HashMap<>());
34             }
35
36             for (Map.Entry<String, Object> entry : countryData.entrySet()) {
37                 Object metrics = entry.getValue();
38
39                 Map<String, Object> metricsMap = (Map<String, Object>) metrics;
40                 for (Map.Entry<String, Object> metric : metricsMap.entrySet()) {
41                     String metricName = metric.getKey();
42                     Object row = metric.getValue();
43
44                     FieldDTO fieldDTO = new FieldDTO();
45                     Map<String, Object> rowMap = (Map<String, Object>) row;
46                     for (Map.Entry<String, Object> rowEntry : rowMap.entrySet()) {
47                         String year = rowEntry.getKey();
48                         Object value = rowEntry.getValue();
49
50                         List<String> xValues = fieldDTO.getX();
51                         List<String> yValues = fieldDTO.getY();
52
53                         if (xValues == null || yValues == null) {
54                             xValues = new ArrayList<>();
55                         }
```

```

TimeseriesRequestDTO.java TimeseriesResponseDTO.java 2 ScatterRequestDTO.java ScatterResponseDTO.java X
src > main > java > mye030 > countries > dto > ScatterResponseDTO.java > {} mye030.countries.dto
You, 2 minutes ago | 1 author (You)
1 package mye030.countries.dto; You, 14 hours ago * Implement scatter data fetching for fixed country
2
3 import java.util.*;
4
5
6 public class ScatterResponseDTO {
7     private String fixed;
8     private List<Map<String, Object>> dataPoints;
9
10
11     public static ScatterResponseDTO mapToResponseDTO(Map<String, Object> data1, Map<String, Object> data2, int minYear, int maxYear) {
12         ScatterResponseDTO scatterResponseDTO = new ScatterResponseDTO();
13
14         if (data1 == null || data2 == null || data1.isEmpty() || data2.isEmpty()) {
15             return scatterResponseDTO;
16         }
17
18         // Extract iso_code from data1
19         String isoCode = String.valueOf(data1.get("iso_code"));
20
21         // Extract the common years between data1 and data2
22         Set<String> commonYears = new HashSet<>(data1.keySet());
23         commonYears.retainAll(data2.keySet());
24
25         // Create a list to hold the data points
26         List<Map<String, Object>> dataPoints = new ArrayList<>();
27
28         // Iterate over the common years and create data points
29         for (String year : commonYears) {
30             if (!year.matches("\\d{4}") || !isInRange(year, minYear, maxYear)) {
31                 continue;
32             }
33             Map<String, Object> dataPoint = new HashMap<>();
34             dataPoint.put("year", year);
35             dataPoint.put("x", data1.get(year));
36             dataPoint.put("y", data2.get(year));
37             dataPoints.add(dataPoint);
38         }
39
40         scatterResponseDTO.setFixed(isoCode);
41         scatterResponseDTO.setDataPoints(dataPoints);
42         return scatterResponseDTO;
43     }
44 }
45
46

```

Εδώ φαίνονται οι προαναφερόμενες κλάσεις. Ειδικά για τα Response αντικείμενα, υλοποιούνται μέθοδοι που αναλαμβάνουν την κατασκευή του Response Object από κάποια μεμονωμένα ακατέργαστα δεδομένα από τη βάση (rows πρακτικά).

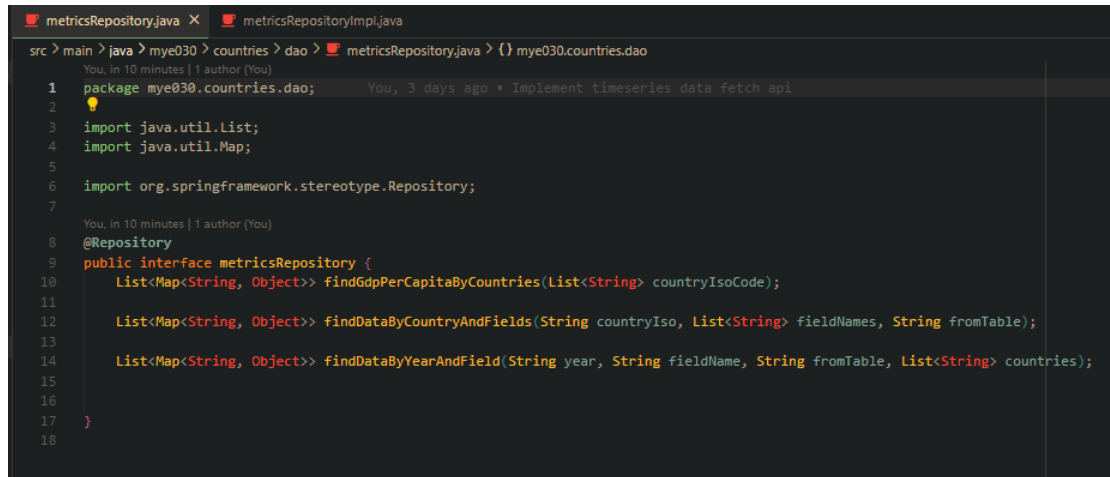
Την επικοινωνία με τη βάση την αναλαμβάνουν οι κλάσεις του πακέτου dao και πιο συγκεκριμένα τα interface countryRepository, metricsRepository και η υλοποίησή του metricsRepositoryImpl.

```

countryRepository.java X
src > main > java > mye030 > countries > dao > countryRepository.java > {} mye030.countries.dao
You, 4 days ago | 1 author (You)
1 package mye030.countries.dao; You, 4 days ago * fetch countries in select box ...
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 import mye030.countries.model.Country;
8
9 public interface countryRepository extends JpaRepository<Country, Integer>{
10     List<Country> findAll();
11 }
12

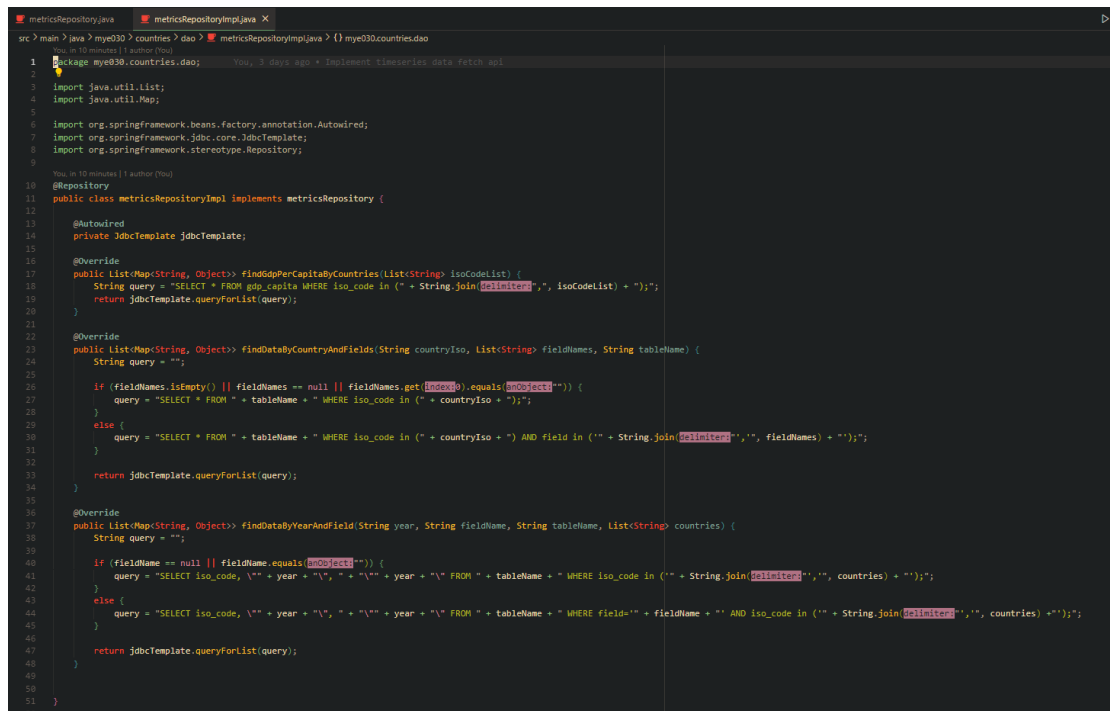
```

Η Spring αναλαμβάνει την εσωτερική υλοποίηση του `countryRepository`, το οποίο έχει στηθεί για να επιτελεί έναν κύριο σκοπό: Να ζητάει από τη βάση όλα τα στοιχεία για τις χώρες, προκειμένου να έχει πρόσβαση ο τελικός χρήστης στην λίστα με όλες τις διαθέσιμες χώρες στη διεπαφή της εφαρμογής.



```
src > main > java > mye030 > countries > dao > metricsRepository.java > {} mye030.countries.dao
You, in 10 minutes | 1 author (You)
1 package mye030.countries.dao; You, 3 days ago • Implement timeseries data fetch api
2
3 import java.util.List;
4 import java.util.Map;
5
6 import org.springframework.stereotype.Repository;
7
8 @Repository
9 public interface metricsRepository {
10     List<Map<String, Object>> findGdpPerCapitaByCountries(List<String> countryIsoCode);
11
12     List<Map<String, Object>> findDataByCountryAndFields(String countryIso, List<String> fieldNames, String fromTable);
13
14     List<Map<String, Object>> findDataByYearAndField(String year, String fieldName, String fromTable, List<String> countries);
15
16
17 }
18
```

Εδώ βλέπουμε την διεπαφή για το selection συγκεκριμένων μετρικών από τη βάση φιλτράροντας ανά χώρα ή ανά έτος, ανάλογα με το είδος του διαγράμματος. Η υλοποίηση για αυτό το interface δεν αφήνεται στα χέρια του framework αλλά αντιθέτως την προσφέρουμε χειροκίνητα μαζί με τα απαραίτητα sql queries.



```
src > main > java > mye030 > countries > dao > metricsRepositoryImpl.java > {} mye030.countries.dao
You, in 10 minutes | 1 author (You)
1 package mye030.countries.dao; You, 3 days ago • Implement timeseries data fetch api
2
3 import java.util.List;
4 import java.util.Map;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.jdbc.core.JdbcTemplate;
8 import org.springframework.stereotype.Repository;
9
10 @Repository
11 public class metricsRepositoryImpl implements metricsRepository {
12
13     @Autowired
14     private JdbcTemplate jdbcTemplate;
15
16     @Override
17     public List<Map<String, Object>> findGdpPerCapitaByCountries(List<String> isoCodeList) {
18         String query = "SELECT * FROM gdp_capita WHERE iso_code in (" + String.join(",", isoCodeList) + ")";
19         return jdbcTemplate.queryForList(query);
20     }
21
22     @Override
23     public List<Map<String, Object>> findDataByCountryAndFields(String countryIso, List<String> fieldNames, String tableName) {
24         String query = "";
25
26         if (fieldNames.isEmpty() || fieldNames == null || fieldNames.get(0).equals("")) {
27             query = "SELECT * FROM " + tableName + " WHERE iso_code in (" + countryIso + ")";
28         }
29         else {
30             query = "SELECT * FROM " + tableName + " WHERE iso_code in (" + countryIso + ") AND field in (" + String.join(",", fieldNames) + ")";
31         }
32
33         return jdbcTemplate.queryForList(query);
34     }
35
36     @Override
37     public List<Map<String, Object>> findDataByYearAndField(String year, String fieldName, String tableName, List<String> countries) {
38         String query = "";
39
40         if (fieldName == null || fieldName.equals("")) {
41             query = "SELECT iso_code, " + year + " FROM " + tableName + " WHERE iso_code in (" + String.join(",", countries) + ")";
42         }
43         else {
44             query = "SELECT iso_code, " + year + " FROM " + tableName + " WHERE field = " + fieldName + " AND iso_code in (" + String.join(",", countries) + ")";
45         }
46
47         return jdbcTemplate.queryForList(query);
48     }
49
50 }
51
52
```



Αυτό που δεν περιγράψαμε είναι το πακέτο service, το οποίο αποτελεί ένα ενδιάμεσο επίπεδο στη διαστρωμάτωση της εφαρμογής, αναλαμβάνοντας την επικοινωνία του επιπέδου dao και των controller της εφαρμογής.

```
countryService.java X countryServiceImpl.java
src > main > java > mye030 > countries > service > countryService.java > {} mye030.countries.service
You, 14 hours ago | 1 author (You)
1 package mye030.countries.service;
2
3 import java.util.List;
4
5 import org.springframework.stereotype.Service;
6
7 import mye030.countries.dto.TimeseriesResponseDTO;
8 import mye030.countries.dto.TimeseriesRequestDTO;
9 import mye030.countries.dto.ScatterResponseDTO;
10 import mye030.countries.dto.ScatterRequestDTO;
11
12 import mye030.countries.model.Country;
13
14 @Service
15 public interface countryService {
16     List<Country> getAllCountries();
17
18     TimeseriesResponseDTO processTimeseriesRequest(TimeseriesRequestDTO request);
19
20     ScatterResponseDTO processScatterRequest(ScatterRequestDTO request);
21 }
22
23
```

Εδώ φαίνεται ο ορισμός του interface countryService και παρακάτω η concrete υλοποίησή της countryServiceImpl.

```
You, 14 hours ago | 1 author (You)
19 @Service
20 public class countryServiceImpl implements countryService {
21     @Autowired
22     private countryRepository countryRepository;
23
24     @Autowired
25     private metricsRepository metricsRepository;
26
27
28
29     @Override
30     public List<Country> getAllCountries() {
31         return countryRepository.findAll();
32     }
33
34
```

Εδώ φαίνεται το call delegation στο dao για το fetching όλων των χωρών.

```
35 @Override
36 public TimeseriesResponseDTO processTimeseriesRequest(TimeseriesRequestDTO request) {
37     // get the countries from the request
38     List<String> countries = request.getCountries();
39
40     // get the fields from the request
41     Map<String, List<String>> fields = request.getFields();
42
43     // get the min and max years from the request
44     Integer minYear = Integer.valueOf(request.getYears().getMin());
45     Integer maxYear = Integer.valueOf(request.getYears().getMax());
46
47     // keep track of all the individual country-fields sets
48     Map<String, Map<String, Object>> results = new HashMap<>();
49
50     // go through every country in the request
51     for (String countryIso: countries) {
52         // initialize the results for the current country
53         results.put(countryIso, new HashMap<>());
54
55         // go through every field (metric) in the request
56         for (String tableName: fields.keySet()) {
57
58             // get all the field (metric) names for the specific table
59             List<String> fieldNames = fields.get(tableName);
60
61             // get the data for the current country and metric
62             Map<String, Map<String, Object>> data = findDataForCountry(countryIso, fieldNames, tableName);
63
64             // add the data for the current countryIsoCode to the results
65             results.get(countryIso).put(tableName, data);
66         }
67     }
68
69     TimeseriesResponseDTO response = TimeseriesResponseDTO.mapToResponseDTO(results, minYear, maxYear);
70     return response;
71 }
72
73 }
```

Εδώ φαίνεται πως επεξεργάζεται ένα αίτημα για Timeseries δεδομένα. Στην ουσία κάνει parse όσες παραμέτρους χρειάζεται από το request που έρχεται και μετά τα περνάει σε μία βοηθητική ρουτίνα findDataForCountry στη γραμμή 62. Η ρουτίνα αυτή παρουσιάζεται παρακάτω.

```
74     @Override
75     public ScatterResponseDTO processScatterRequest(ScatterRequestDTO request) throws IllegalArgumentException {
76         // get the fixed field from the request
77         String fixedField = request.getFixedField();
78
79         // depending on what the fixed field is, call the appropriate method
80         if (fixedField.equals(anObject:"country")) {
81             return fixedCountryScatter(request);
82         }
83         else if (fixedField.equals(anObject:"year")) {
84             return fixedYearScatter(request);
85         }
86         else {
87             throw new IllegalArgumentException(anObject:"Fixed field must be either 'country' or 'year'");
88             // return null;
89         }
90     }
91
92
93
94     private Map<String, Map<String, Object>> findDataForCountry(String isoCode, List<String> fieldNames, String tableName) {
95         List<Map<String, Object>> resultRows = metricsRepository.findDataByCountryAndFields(isoCode, fieldNames, tableName);
96
97         Map<String, Map<String, Object>> result = new HashMap<>();
98
99         // for the case where the tableName is the same as the fieldName (usually economic metrics)
100         if (fieldNames == null || fieldNames.isEmpty()) {
101             String fieldName = tableName;
102             result.put(fieldName, resultRows.get(index:0));
103         }
104
105         else {
106             for (int i = 0; i < resultRows.size(); i++) {
107                 String fieldName = fieldNames.get(i);
108                 result.put(fieldName, resultRows.get(i));
109             }
110         }
111
112         return result;
113     }
114
115     private List<Map<String, Object>> findDataForYear(String year, String fieldName, String tableName, List<String> countries) {
116         List<Map<String, Object>> resultRows = metricsRepository.findDataByYearAndField(year, fieldName, tableName, countries);
117
118         return resultRows;
119     }
120 }
```

Εδώ φαίνονται οι 2 βοηθητικές ρουτίνες findDataForCountry και findDataForYear που μεταφέρουν το αίτημα στο metricRepository του πακέτου dao.

Κάτω, με τη χρήση των βοηθητικών συναρτήσεων που αναφέραμε μόλις, γίνεται η επεξεργασία του JSON αιτήματος για τα scatter plots και ανάλογα με το αν είναι με σταθερή χώρα ή σταθερό χρόνο. Έπειτα επιστρέφεται το αντικείμενο response που παράγεται στον controller ο οποίος το προωθεί ως απάντηση JSON στο frontend.

```
122 private ScatterResponseDTO fixedCountryScatter(ScatterRequestDTO request) throws IllegalArgumentException {
123     // get the country from the request
124     String countryIso = request.getCountries().get(0);
125
126     // get the fields from the request
127     Map<String, String> xField = request.getXField();
128     Map<String, String> yField = request.getYField();
129
130
131     if (xField.isEmpty() || yField.isEmpty()) {
132         throw new IllegalArgumentException("Both xField and yField must be present");
133     }
134
135     // get the table names from the request
136     String xTable = xField.keySet().iterator().next();
137     String yTable = yField.keySet().iterator().next();
138     // get the field names from the request
139     String xMetric = xField.values().iterator().next();
140     List<String> xMetricList = Arrays.asList(xMetric);
141
142     String yMetric = yField.values().iterator().next();
143     List<String> yMetricList = Arrays.asList(yMetric);
144
145     // get the min and max years from the request
146     Integer minYear = Integer.valueOf(request.getYears().getMin());
147     Integer maxYear = Integer.valueOf(request.getYears().getMax());
148
149
150     Map<String, Map<String, Object>> result1;
151     Map<String, Map<String, Object>> result2;
152     Map<String, Object> data1 = null;
153     Map<String, Object> data2 = null;
154
155     result1 = findDataForCountry(countryIso, xMetricList, xTable);
156     result2 = findDataForCountry(countryIso, yMetricList, yTable);
157     try {
158         data1 = result1.values().iterator().next();
159         data2 = result2.values().iterator().next();
160     }
161     catch (Exception e) {
162         System.out.println("No data found for country " + countryIso);
163     }
164
165
166     ScatterResponseDTO response = ScatterResponseDTO.mapToResponseDTO(data1, data2, minYear, maxYear);
167     return response;
168
169 }
170
```

```

171 private ScatterResponseDTO fixedYearScatter(ScatterRequestDTO request) throws IllegalArgumentException {
172     // get the fixed year from the request
173     String year = request.getYears().getMin();
174
175     // get the countries from the request
176     List<String> countries = request.getCountries();
177
178     // get the fields from the request
179     Map<String, String> xField = request.getXField();
180     Map<String, String> yField = request.getYField();
181
182
183     if (xField.isEmpty() || yField.isEmpty()) {
184         throw new IllegalArgumentException("Both xField and yField must be present");
185     }
186
187     // get the table names from the request
188     String xTable = xField.keySet().iterator().next();
189     String yTable = yField.keySet().iterator().next();
190     // get the field names from the request
191     String xMetric = xField.values().iterator().next();
192     String yMetric = yField.values().iterator().next();
193
194
195     List<Map<String, Object>> data1 = findDataForYear(year, xMetric, xTable, countries);
196     List<Map<String, Object>> data2 = findDataForYear(year, yMetric, yTable, countries);
197
198     ScatterResponseDTO response = ScatterResponseDTO.mapToResponseDTO(data1, data2, year);
199     return response;
200 }
201
202 }

```

### 3 ΥΠΟΔΕΙΓΜΑΤΑ ΕΡΩΤΗΣΕΩΝ ΚΑΙ ΑΠΑΝΤΗΣΕΩΝ

Ας ρίξουμε μια ματιά σε κάποια υποδείγματα ερωτήσεων που γίνονται στη βάση μας κατά την λειτουργία της εφαρμογής.

Query Query History

1 SELECT \* FROM countries 5

Scratch Pad

Data Output Messages Notifications

	iso	iso3	iso_code	type	display_name	official_name	capital	continent
	character varying (2)	character varying (3)	(PK) integer	character varying (50)	character varying (50)	character varying (100)	character varying (50)	character varying (50)
1	AF	AFG	4	AF	Afghanistan	Afghanistan	Kabul	Asia
2	AX	ALA	248	[null]	Aland Islands	Aland Islands	Mariehamn	Europe
3	AL	ALB	8	AL	Albania	Albania	Tirana	Europe
4	DZ	DZA	12	AG	Algeria	Algeria	Algiers	Africa
5	AS	ASM	16	AQ	American Samoa	American Samoa	Pago Pago	Oceania
6	AD	AND	20	AN	Andorra	Andorra	Andorra la Vella	Europe
7	AO	AGO	24	AO	Angola	Angola	Luanda	Africa
8	AI	AIA	660	AI	Anguilla	Anguilla	The Valley	North America
9	AQ	ATA	10	AY	Antarctica	Antarctica		Antarctica
10	AG	ATG	28	AC	Antigua and Barbuda	Antigua and Barbuda	St. John's	North America
11	AR	ARG	32	AR	Argentina	Argentina	Buenos Aires	South America
12	AM	ARM	51	AM	Armenia	Armenia	Yerevan	Asia
13	AW	ABW	533	AA	Aruba	Aruba	Oranjestad	North America
14	AU	AUS	36	AS	Australia	Australia	Canberra	Oceania

Total rows: 252 of 252 Query complete 00:00:00.176 Ln 1, Col 26

Query Query History Scratch Pad

```
1 SELECT * FROM gdp_capita WHERE iso_code IN (300, 4)
```

Data Output Messages Notifications

	iso_code [PK] integer	1990 numeric	1995 numeric	2000 numeric	2005 numeric	2010 numeric	2011 numeric	2012 numeric	2013 numeric	2014 numeric	2015 numeric	2016 numeric	2017 numeric	2018 numeric
1	4	[null]	[null]	[null]	1099	1672	1627	1773	1808	1796	1767	1757	1758	1735
2	300	20686	21247	24839	29559	28726	26141	24364	23746	24082	24135	24189	24602	25141

Total rows: 2 of 2 Query complete 00:00:00.088 Ln 1, Col 52

Query Query History Scratch Pad

```
1 SELECT * FROM vital_statistics WHERE iso_code IN (262) and field IN ('crude_birth_rate')
```

Data Output Messages Notifications

	id [PK] integer	iso_code integer	field character varying (50)	1950 numeric	1951 numeric	1952 numeric	1953 numeric	1954 numeric	1955 numeric	1956 numeric	1957 numeric	1958 numeric	1959 numeric	1960 numeric	1961 numeric	1962 numeric	1963 numeric
1	896	262	crude_birth_rate	42.87	42.85	42.83	42.8	42.78	42.76	42.74	42.68	42.62	42.55	42.48	42.44	42.37	

Total rows: 1 of 1 Query complete 00:00:00.090 Ln 1, Col 52

Query Query History Scratch Pad

```
1 SELECT iso_code, "2015"
2 FROM vital_statistics
3 WHERE iso_code IN (262)
4 AND field='crude_birth_rate'
```

Data Output Messages Notifications

	iso_code integer	2015 numeric
1	262	23.65

Total rows: 1 of 1 Query complete 00:00:00.155 Ln 1, Col 29

Ερωτήσεις τύπου JOIN αν και ενδεχομένως πιο αποδοτικές για το fetching συνδιασμού πολλών μετρικών για scatter plots, στην δική μας περίπτωση δεν βολεύει λόγω του ότι τα έτη βρίσκονται κατά μήκος των table ως columns. Για αυτό γίνεται με κώδικα στο backend. Το θετικό με την δομή των πινάκων μας, είναι η ευκολία στα SELECT queries όταν πρόκειται για plotting timeseries δεδομένων. Όσοι μας περιορίζει στην περίπτωση που χρειάζεται να εισάγουμε μια καινούργια χρονία για παράδειγμα και στο ότι όπως προαναφέραμε δεν έχουμε την δυνατότητα να φιλτράρουμε τα δεδομένα να βρίσκονται μέσα σε ένα range από χρόνια εύκολα γιατί αυτό προϋποθέτει μία ερώτηση της μορφής WHERE YEAR < max AND YEAR > min. Αυτό το πρόβλημα παρακάμτεται με τη χρήση κώδικα σε java κατά την επεξεργασία των rows που επιστρέφονται από τη βάση σε ύστερο χρόνο.

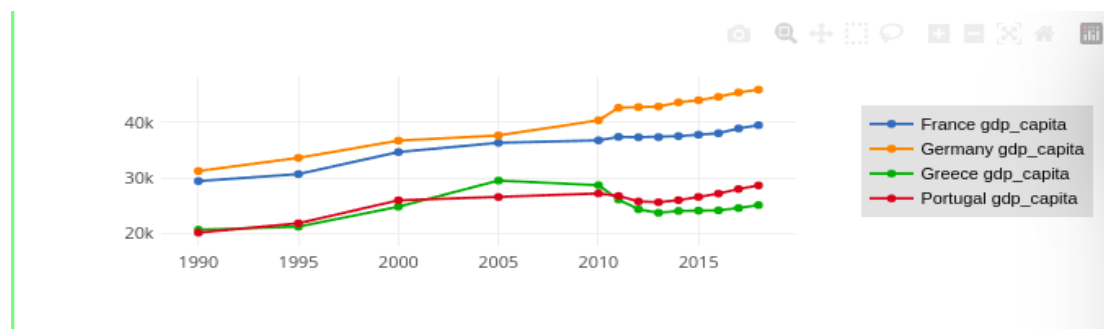
## 4 FRONT END

Η βιβλιοθήκη που χρησιμοποιήσαμε για να δημιουργήσαμε τα διαγράμματα είναι η plotly.js η οποία είναι χτυσμένη πάνω στην d3.js.

Η βασική υλοποίηση είναι ότι φτιάξαμε μια μετρική κλάση Chart και άλλες δύο που την κάνουν extend για να ζωγραφίσουμε τα Timeseries και Scatter διαγράμματα.

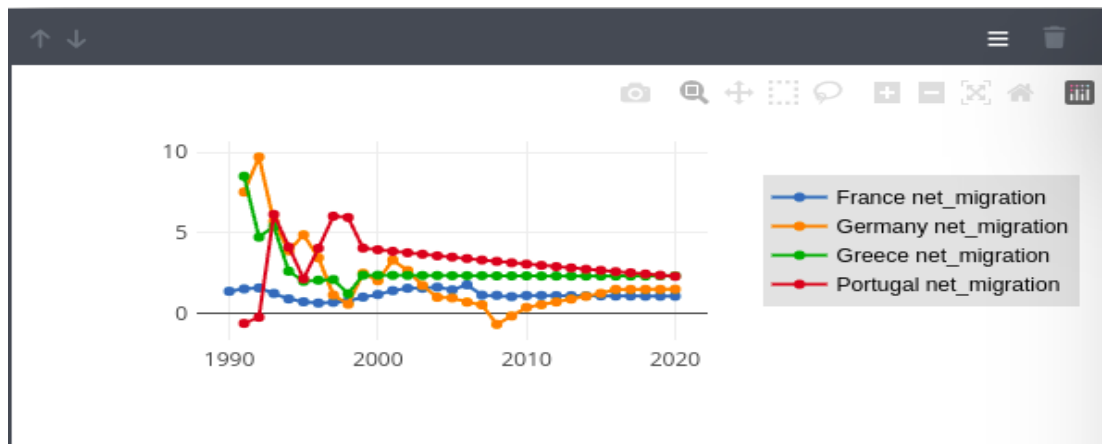
## 5 ΜΕΛΕΤΗ ΔΕΔΟΜΕΛΩΝ

GDP per capita timeline



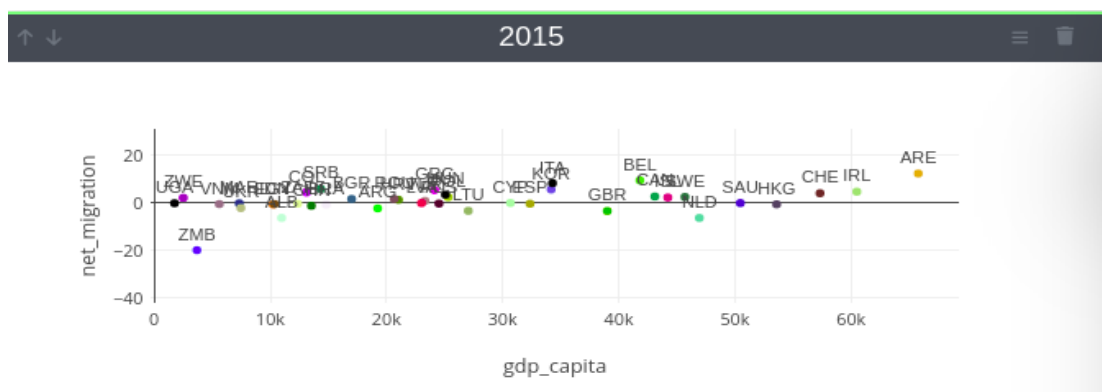
Στο γράφημα βλέπουμε ότι υπάρχει μία θετική τάση αύξησης του κατά κεφαλήν ΑΕΠ κατά τη διάρκεια των χρόνων. Επιπλέον βλέπουμε ότι από την κρίση του 2007 επηρεάστηκε περισσότερο η Πορτογαλία και η Ελλάδα, χώρες δηλαδή της περιφέρειας της Ευρώπης ενώ οι χώρες του πυρήνα φαίνεται να μην επηρεάστηκαν εξίσου πολύ από την κρίση.

Net migration timeline



Επομένως βλέπουμε ότι από την κρίση και μετά το net migration της γερμανίας να αυξάνεται και άρα να δέχεται περισσότερους μετανάστες ενώ η πορτογαλία άρχισε να έχει όλο και λιγότερο, ενώ δεν βλέπουμε ιδιαίτερες διαφορές σε Ελλάδα και Γαλλία. Επομένως φαίνεται ότι όσο πιο πολύ αυξάνεται το ΑΕΠ τόσο περισσότερους μετανάστες δέχεται μία χώρα.

#### Net migration- GDP per capita scatter plot



Έτσι βλέπουμε ότι συνολικά, παίρνοντας μεγάλο δείγμα χωρών για τις οποίες έχουμε δεδομένα από το έτος 2015 παρατηρούμε μία θετική συσχέτιση μεταξύ του ΑΕΠ και της μετανάστευσης προς την χώρα. Άρα χώρες που έχουν έχουν μεγαλύτερη παραγωγή προσελκύουν και περισσότερους ανθρώπους, λογικά από φτωχότερες χώρες που είναι πιο πιθανό όπως βλέπουμε οι κατοικοί τους να φύγουν.

#### Bar charts: GFCF & Labor Shares

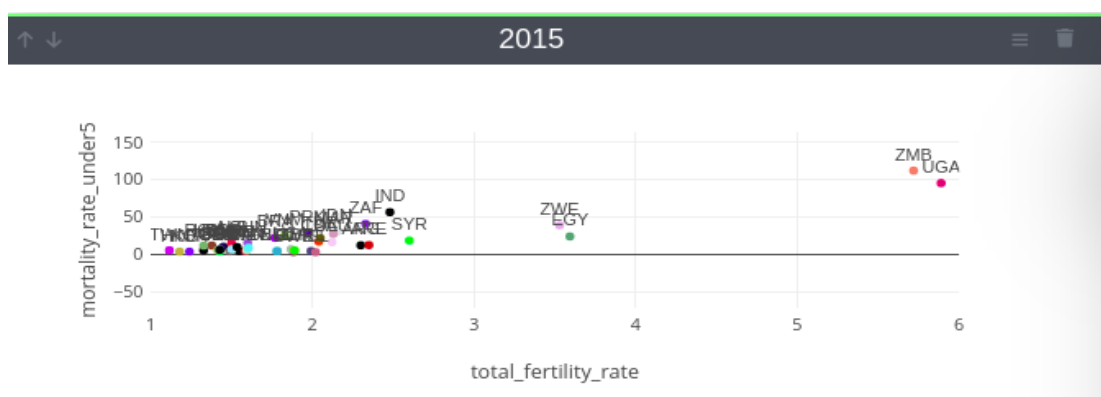




Παρατηρούμε από το πρώτο γράφημα το πόσο χαμηλό investment έχει η Ελλάδα τα τελευταία χρόνια το οποίο έχει να κάνει με την κρίση και άρα την πτώση των επενδύσεων στην χώρα ενώ αντίθετα μια σχετικά φτωχή χώρα, η Ουγκάντα, έχει αρκετές επενδύσεις, λογικά από ξένες επιχορηγήσεις το οποίο συμβάλλει στην ανάπτυξή της. Βέβαια, η διαφορά στις συνολικές επενδύσεις έχει να κάνει σίγουρα και με τον πληθυσμό αφού χώρες με λιγότερους ανθρώπους όπως η Ελλάδα έχουν και συνολικά μικρότερες επενδύσεις όπως βλέπουμε.

Από το άλλο γράφημα παρατηρούμε ότι το συνολικό μερίδιο της αγοράς της εργασίας σε κάθε χώρα παραμένει σχετικά σταθερό σε κάθε χρονιά. Κάτι άλλο που βλέπουμε είναι ότι χώρες που έχουν περισσότερα εργατικά δικαιώματα έχουν μεγαλύτερο μερίδιο αγοράς οι εργαζόμενοι (πχ Καναδας) ενώ αντίθετα σε χώρες όπως η Τουρκία οι εργαζόμενοι έχουν σχετικά μικρότερο μερίδιο.

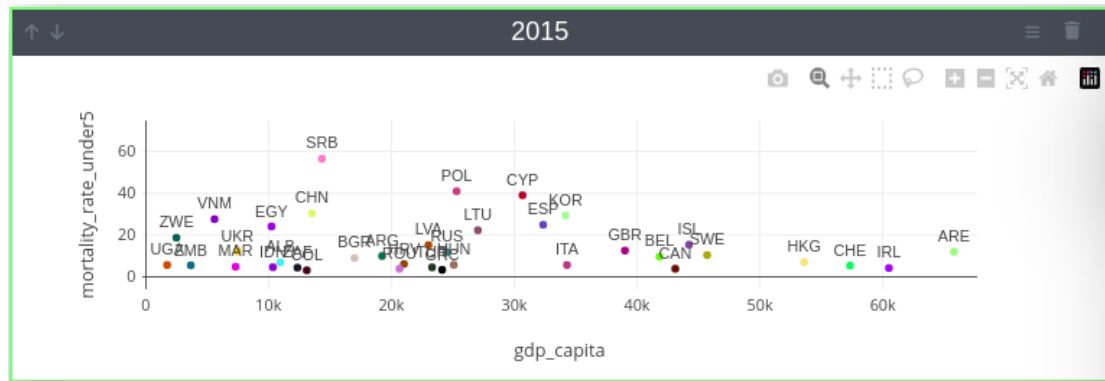
#### Mortality and fertility rate - scatter plot



Ένα ακόμα ενδιαφέρον δημογραφικό στοιχείο παρατηρούμε από το παραπάνω γράφημα. Χώρες με μεγαλύτερη θνησιμότητα σε νεαρές ηλικίες (κάτω των 5 ετών)

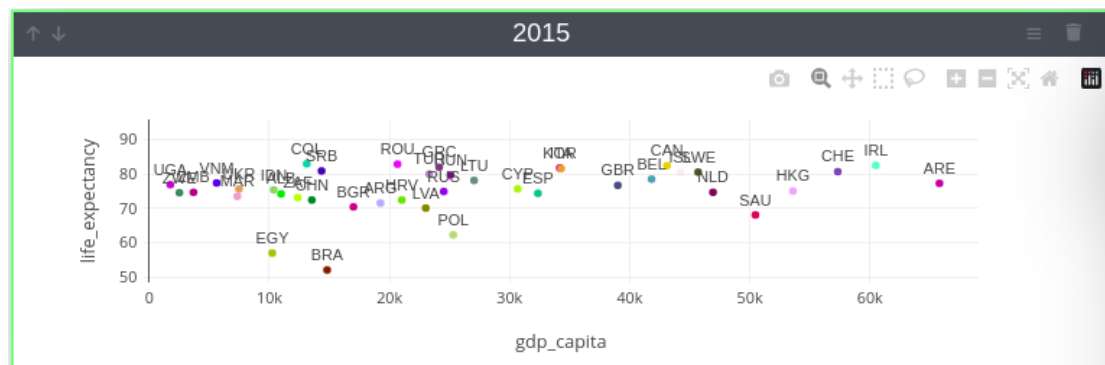
έχουν και περισσότερα παιδιά. Άρα έχουμε θετική συσχέτιση θνησιμότητας και γονιμότητας.

GDP per capita & mortality - scatter plot



Εδώ βλέπουμε ότι όσο πιο πολύ αυξάνεται το συνολικό εισόδημα μιας χώρας τόσο λιγότερα μικρά παιδιά πεθαίνουν κατά μέσο όρο, και άρα υπάρχει αρνητική συσχέτιση μεταξύ mortality rate και GDP per capita.

Life expectancy & GDP per capita -scatter plot



Βλέπουμε μία θετική συσχέτιση στο ΑΕΠ και στα συνολικά χρόνια που ζούνε οι άνθρωποι σε μία χώρα. Άρα χώρες που είναι πιο πλούσιες οι άνθρωποι ζούνε περισσότερο. Αυτό φαίνεται να δικαιολογεί ότι το ΑΕΠ είναι δείκτης ευημερίας. Βέβαια δεν είναι μόνο το ΑΕΠ που φαίνεται να επηρεάζει το life expectancy αφού χώρες με υψηλό inequality όπως η Σαουδική Αραβία φαίνεται να έχουν μικρότερο προσδόκιμο ζωής.