

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ
ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ
ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ
ΤΟ ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2020-2021

ΟΜΑΔΑ

ΜΑΡΙΟΣ ΖΙΧΝΑΛΗΣ, 3226

ΕΛΕΝΗ ΜΟΥΖΑΚΗ, 3280

ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΪΩΑΝΝΟΥ, 3309

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΑΪΟΣ 2021

ΙΣΤΟΡΙΚΟ ΠΡΟΗΓΟΥΜΕΝΩΝ ΕΚΔΟΣΕΩΝ

Ημερομηνία	Έκδοση	Περιγραφή	Συγγραφέας
20/3/2021	0.1	λήψη, τροποποίηση και φόρτωμα δεδομένων στη βάση δεδομένων	3226-3280-3309
15/4/2021	0.2	Στήσιμο Spring Boot back-end, σύνδεση με τη βάση δεδομένων, δοκιμαστικά queries	3226-3280-3309
26/5/2021	1.0	Στήσιμο React front-end και δημιουργία διαγραμμάτων με d3	3226-3280-3309

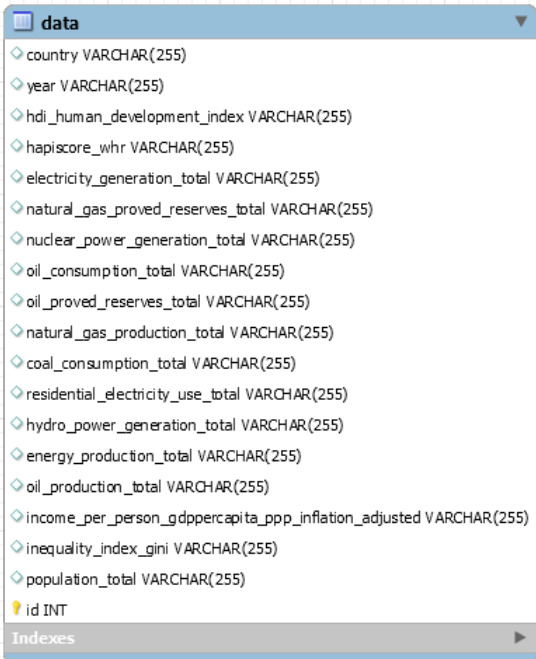
1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Από τον οργανισμό GapMinder επιλέξαμε από την κατηγορία “Energy” τις συνολικές τιμές για κάθε είδος ενέργειας (Coal, Electricity, Hydro, Natural gas, Nuclear, Oil) και χρησιμοποιήσαμε ακόμα και το total energy production. Ακόμα από τους κοινωνικούς δείκτες χρησιμοποιήσαμε τα Happiness score (WHR) και Human Development Index (HDI) και τρεις επιπλέον δείκτες με στοιχεία αναφοράς, τους GDP, total population και inequality index. Συνολικά τα δεδομένα μας απαρτίζονται από 16 αρχεία CSV, με συνολικό μέγεθος 1.1MB.

1.1 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΛΟΓΙΚΟ ΕΠΙΠΕΔΟ

Αρχικά τροποποιήσαμε τα αρχεία δεδομένων μας ώστε να γραφτούν σωστά εντός της βάσης, με τη βοήθεια προγραμμάτων που γράψαμε σε γλώσσα python. Καταλήξαμε η βάση δεδομένων μας να αποτελείται από ένα μεγάλο table, το οποίο περιέχει όλα τα δεδομένα με κατηγορίες Country, Year, Index1,..., IndexN, id. Σε πλειάδες που δεν υπήρχαν μετρήσεις για κάποιο index, προσθέσαμε None. Επιλέξαμε να δημιουργήσουμε τη βάση δεδομένων με αυτή τη μορφή έτσι ώστε να αποφύγουμε περίπλοκη σχεσιακή άλγεβρα, θυσιάζοντας βέβαια την ταχύτητα.

Το παρακάτω είναι το τελικό schema:



data	
country	VARCHAR(255)
year	VARCHAR(255)
hdi_human_development_index	VARCHAR(255)
hapiscorw_whr	VARCHAR(255)
electricity_generation_total	VARCHAR(255)
natural_gas_proved_reserves_total	VARCHAR(255)
nuclear_power_generation_total	VARCHAR(255)
oil_consumption_total	VARCHAR(255)
oil_proved_reserves_total	VARCHAR(255)
natural_gas_production_total	VARCHAR(255)
coal_consumption_total	VARCHAR(255)
residential_electricity_use_total	VARCHAR(255)
hydro_power_generation_total	VARCHAR(255)
energy_production_total	VARCHAR(255)
oil_production_total	VARCHAR(255)
income_per_person_gdppercapita_ppp_inflation_adjusted	VARCHAR(255)
inequality_index_gini	VARCHAR(255)
population_total	VARCHAR(255)
id	INT
Indexes	

1.2 ΣΧΕΣΙΑΚΟ ΣΧΗΜΑ ΣΕ ΦΥΣΙΚΟ ΕΠΙΠΕΔΟ

1.2.1 ΡΥΘΜΙΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΟΥ DBMS

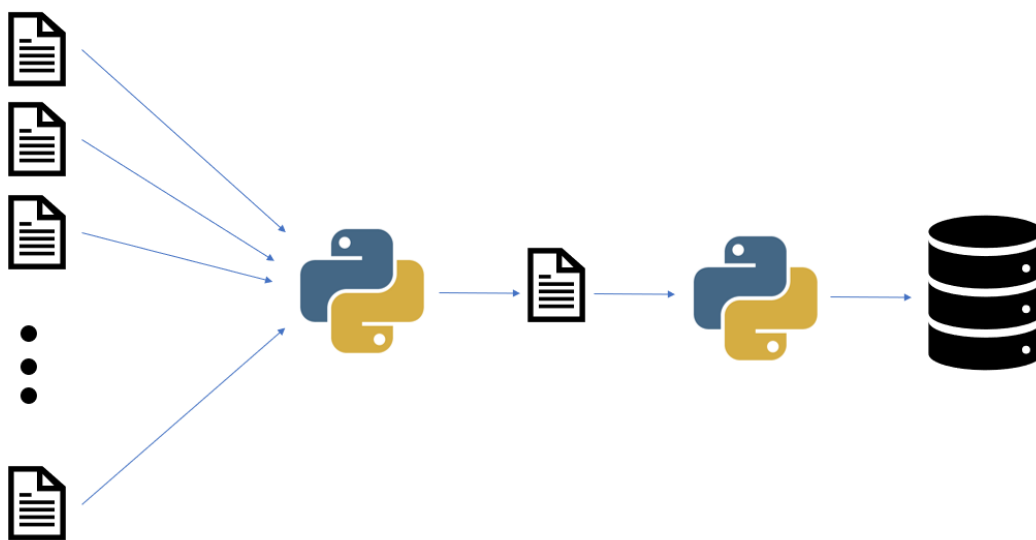
Στις παραμέτρους του DBMS κάναμε μόνο μία τροποποίηση. Η τροποποίηση αυτή έγινε με την εντολή “# SET GLOBAL local_infile = 1;”, η οποία μας επέτρεψε να φορτώσουμε το τελικό αρχείο όλων των δεδομένων (final.tsv) στο DBMS.

1.2.2 ΡΥΘΜΙΣΗ ΑΣΦΑΛΕΙΑΣ

Κατά τη διαδικασία ανάπτυξης της εφαρμογής δημιουργήσαμε έναν χρήστη στη MySQL ο οποίος είχε πρόσβαση στο table της βάσης δεδομένων.

2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ

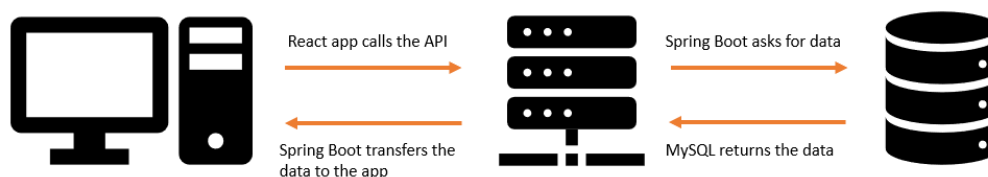
2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΔΟΜΗ ETL



Αρχικά δημιουργήσαμε το αρχείο combine_data_files.py, το οποίο είναι υπεύθυνο για τη δημιουργία ενός αρχείου .tsv που περιέχει τον συνδυασμό όλων των δεδομένων μας. Για την εκτέλεσή του γράφουμε `python3 combine_data_files.py "data_path"`, όπου data_path είναι το path του φακέλου που περιέχει τα αρχεία που θέλουμε να συμπεριλάβουμε στο DB. Μετά την εκτέλεσή του, δημιουργείται το αρχείο final.tsv. Στη συνέχεια εκτελώντας το αρχείο schema_creation.py, δημιουργείται το table της βάσης δεδομένων και γεμίζει με τα στοιχεία από το αρχείο final.tsv που δημιουργήσαμε. Να σημειωθεί σε αυτό το σημείο ότι εντός του κώδικα του schema_creation.py, ορίζεται το password του χρήστη root της MySQL, οποίος έχει τα προνόμια να δημιουργεί και να γεμίζει tables.

2.2 ΔΙΑΓΡΑΜΜΑΤΑ ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε για back-end η Spring Boot και για front-end η React. Παρακάτω φαίνεται η επικοινωνία μεταξύ back-end, front-end και database.



2.3 ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ ΚΕΝΤΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

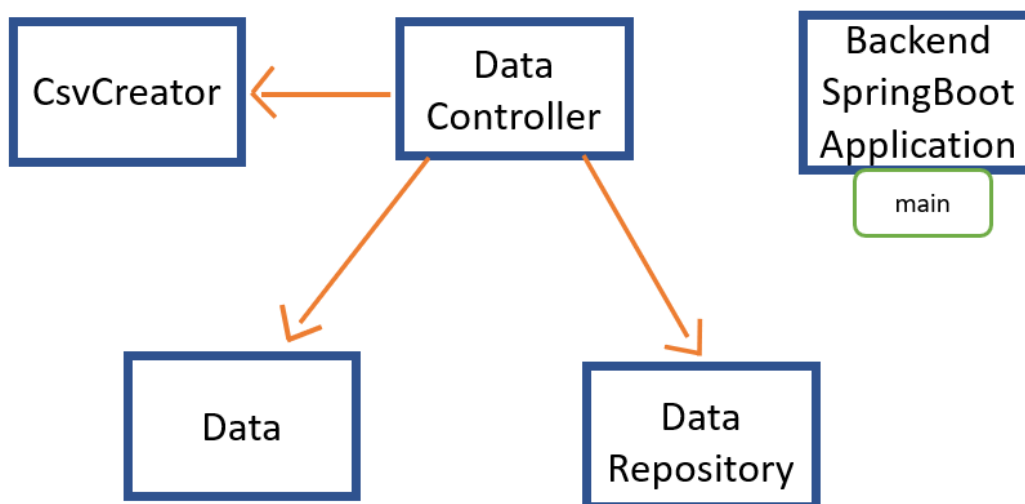
Στο back-end μας υπάρχουν οι παρακάτω κλάσεις:

1. Η κλάση Data αναπαριστά τα δεδομένα του DB μας, έχει ως πεδία όλες τις πληροφορίες του κάθε data entry και έχει ακόμα getters για να έχουμε πρόσβαση στις πληροφορίες αυτές.
2. Η διεπαφή Data Repository, η οποία κάνει extend ένα CrudRepository τύπου Data, στο οποίο ορίζουμε τις συναρτήσεις που θα χρησιμοποιεί το API μας.
3. Η κλάση DataController, αυτή περιέχει τις κλήσεις του API, χρησιμοποιεί ένα αντικείμενο τύπου DataRepository και στην ουσία μέσω αυτής της κλάσης γίνεται η επικοινωνία μεταξύ back-end και front-end. Δημιουργήσαμε τα εξής API functions τα οποία λειτουργούν και ως βήματα συλλογής πληροφοριών:
 - a. `getDiagramType()` με path `"/data/d/{diagramType}"` η οποία σύμφωνα με την επιλογή του χρήστη στο front-end ορίζει το είδος του διαγράμματος το οποίο θα πρέπει να παράγουμε για να ξέρουμε το είδος των τελικών δεδομένων που θα χρειαστούν.
 - b. `getCountries()` με path `"/data/i/{indexes}"` η οποία σύμφωνα με την επιλογή του χρήστη στο front-end ορίζει τους δείκτες για τους οποίους θα γίνουν τα ερωτήματα. Σε αυτό το σημείο επιτάσσονται ερωτήματα τύπου SQL που γίνονται για κάθε δείκτη που επιλέχθηκε και «επιστρέφουν» τις χώρες που έχουν μετρήσεις για αυτούς τους δείκτες, επίσης «επιστρέφεται» και το id της κάθε πλειάδας. Σε περίπτωση πολλαπλών index πρέπει να σιγουρευτούμε πως όλες οι χώρες που «επιστρέφονται» έχουν μετρήσεις για όλους τους δείκτες, αυτό το επιτυγχάνουμε χρησιμοποιώντας sets και παίρνοντας την τομή τους.
 - c. `getYears()` με path `"/data/c/{countries}"` η οποία σύμφωνα με την επιλογή του χρήστη στο front-end ορίζει τις χώρες για τις οποίες θα αναζητήσουμε τις μετρήσεις. Αυτή η συνάρτηση χρησιμοποιεί τα ids που

συλλέχτηκαν από το προηγούμενο βήμα και «κρατάει» μόνο τα ids που αντιστοιχούν στις επιλεγθέν χώρες.

- d. `getFinalData()` με path `"/data/y/{years}"` η οποία σύμφωνα με την επιλογή του χρήστη στο front-end ορίζει την χρονική περίοδο για την οποία θα αναζητήσουμε τις μετρήσεις. Αυτή η συνάρτηση «κρατάει» μόνο όσα ids τηρούν την χρονική συνθήκη.
Αυτή είναι η τελευταία συνάρτηση που χρειάζεται user input, αφού γίνει και το τελικό prune των ids είμαστε έτοιμοι να δημιουργήσουμε το αρχείο .csv, το οποίο χρειάζονται τα διαγράμματα της d3 για να δημιουργήσουν το σχετικό διάγραμμα.

4. Η κλάση `CsvCreator`, την οποία καλεί η `DataController` όταν έχει συλλέξει όλες τις πληροφορίες από τον χρήστη, ώστε να δημιουργηθούν τα αντίστοιχα διαγράμματα. Τα .csv αρχεία που δημιουργούνται παρέχονται στα path: `"/data/plot/lineResult.csv"`, `"/data/plot/barResult.csv"` και `"/data/plot/scatterResult.csv"`.
5. Η κλάση `BackendSpringBootApplication`, που αποτελεί τη main κλάση του συστήματος και «στήνει» τον server του back-end.



Το Front-end αποτελείται από React components τα οποία αλληλοεπιδρούν μεταξύ τους για να δημιουργήσουν μια φόρμα συμπλήρωσης στοιχείων η οποία συλλέγει τις απαραίτητες πληροφορίες για τη δημιουργία των τελικών διαγραμμάτων.

- `Stepper`, κυρίως component που αποτελεί την διαδικασία συλλογής στοιχείων
- `PlotSelection`, component επιλογής του επιθυμητού διαγράμματος, με βάση την επιλογή του χρήστη καθορίζει το επόμενο βήμα.
- `IndexSelection/IndexSelectionScatter`, components με τα οποία ο χρήστης επιλέγει τα επιθυμητά indexes. Στη γενική περίπτωση ο χρήστης μπορεί να επιλέξει όσους δείκτες επιθυμεί, στην περίπτωση του scatter plot όμως γίνεται να επιλέξει μόνο δυο δείκτες.

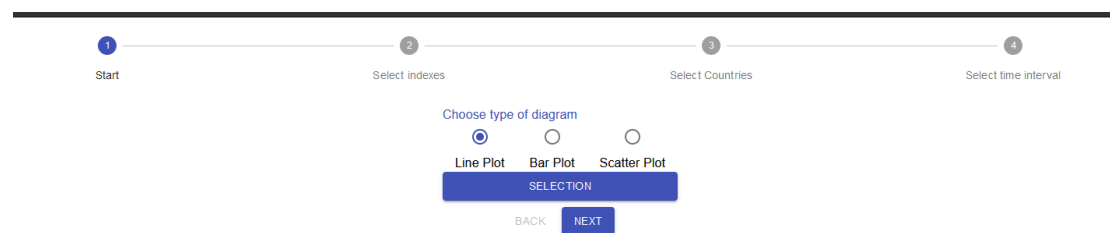
- CountrySelection, component που μετά την επιλογή των δεικτών δίνει την δυνατότητα στον χρήστη να επιλέξει τις χώρες που έχουν μετρήσεις για αυτούς τους δείκτες.
- YearSelection, τελικό component για την επιλογή της χρονικής περιόδου.

Υπάρχει ένα ακόμα component, το DataService το οποίο αναλαμβάνει να εκπληρώνει τα request του κάθε βήματος του Stepper καλώντας τα path του API με τις ανάλογες παραμέτρους.

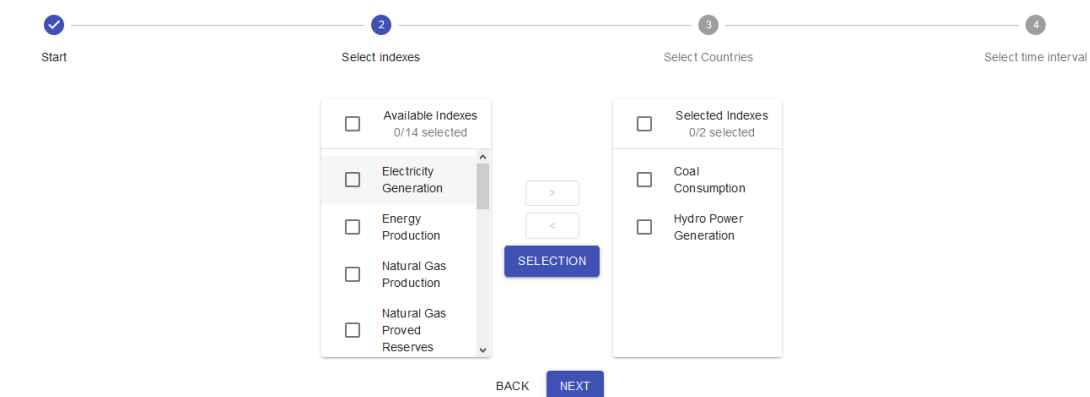
Τέλος υπάρχουν τα αρχεία Lineplot.html, Barplot.html, Scatterplot.html που περιέχουν τον κώδικα της d3 για την δημιουργία διαγραμμάτων και χρησιμοποιούν ως δεδομένα αρχεία .csv που παρέχει το backend.

3 ΥΠΟΔΕΙΓΜΑΤΑ ΕΡΩΤΗΣΕΩΝ ΚΑΙ ΑΠΑΝΤΗΣΕΩΝ

Πρώτα πρέπει να διαλέξουμε τι είδους διάγραμμα επιθυμούμε. Μετά πατώντας “SELECTION” επιβεβαιώνουμε την επιλογή μας και πατάμε “NEXT”.



Στο επόμενο βήμα καλούμαστε να επιλέξουμε indices, πατάμε το βελάκι για να μεταφερθούν στο δεξί κουτάκι, “SELECTION” και “NEXT”.



Start Select indexes Select Countries Select time interval

Available Indexes
0/67 selected

Australia
Austria
Azerbaijan
Bangladesh
Belarus

Selected Indexes
0/2 selected

Algeria
Argentina

SELECTION

BACK NEXT

Έπειτα πρέπει να επιλέξουμε τις χώρες για τις οποίες θέλουμε πληροφορίες, ακολουθώντας την ίδια διαδικασία. Στο κουτί στα αριστερά θα εμφανίζονται μόνο οι χώρες για τις οποίες έχουμε πληροφορίες.

Start Select indexes Select Countries Select time interval

From 1973 To 1983

Please choose starting year Please choose ending year

SELECTION

BACK FINISH

Στο τελικό βήμα, πρέπει να διαλέξουμε την χρονική περίοδο για την οποία θέλουμε να πάρουμε πληροφορίες, μετά πατάμε “SELECTION” και μετά “FINISH”.

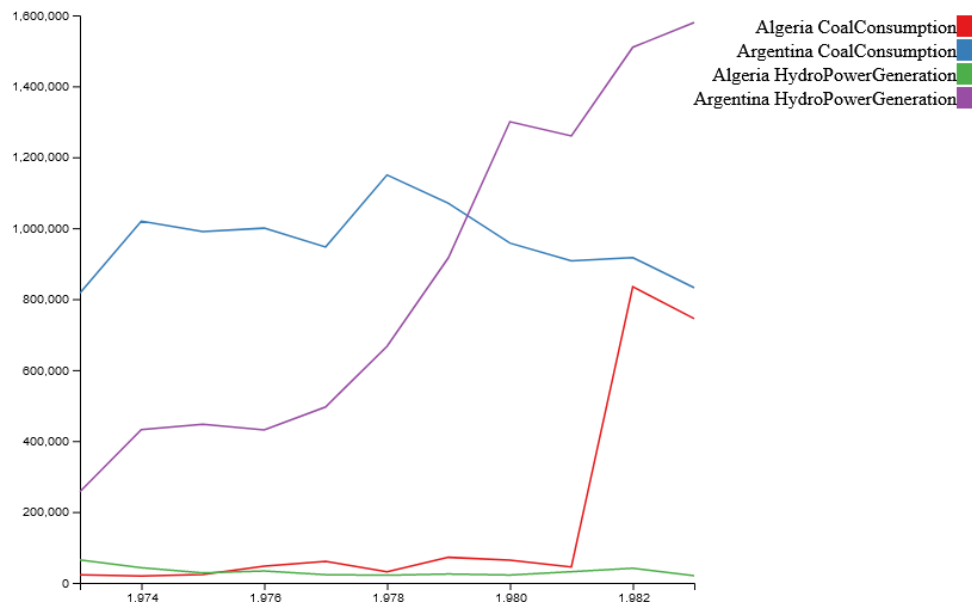
Start Select indexes Select Countries Select time interval

All done

RESET

Σε αυτό το παράθυρο έχει τελειώσει η διαδικασία. Για να δούμε το plot που επιθυμούμε (το οποίο δεν φαίνεται εκεί που θα έπρεπε) πρέπει να ανοίξουμε το αντίστοιχο .html αρχείο.

Για να δει κανείς το αποτέλεσμα-διάγραμμα από την παραπάνω διαδικασία θα πρέπει να ανοίξει το αντίστοιχο αρχείο HTML στον browser του.



4 ΤΕΚΜΗΡΙΩΣΗ ΚΑΙ ΛΟΙΠΑ ΣΧΟΛΙΑ

Στο front-end λείπουν έλεγχοι για την αποφυγή λαθών χρήστη και είναι αρκετά εύκολο να δοθεί λάθος input στο back-end.

Δεν υλοποιήθηκε η λειτουργία για επιλογή μέσου όρου πενταετίας, δεκαετίας, κοκ.