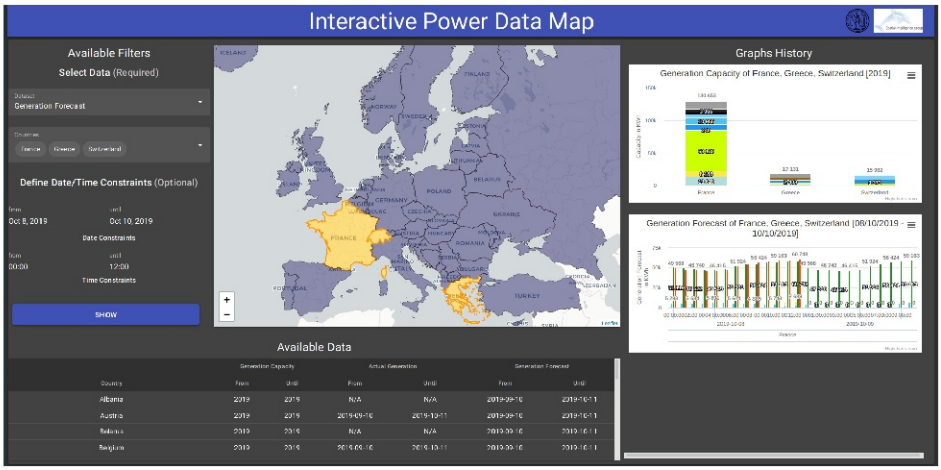
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΑΔΡΑΣΤΙΚΟΣ ΧΑΡΤΗΣ ΤΗΣ ΕΛΕΥΘΕΡΗΣ ΑΓΟΡΑΣ ΕΝΕΡΓΕΙΑΣ ΣΤΗΝ ΕΥΡΩΠΑΙΚΗ ΕΝΩΣΗ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ GIS ΕΦΑΡΜΟΓΗΣ ΜΕ ΧΡΗΣΗ ΣΥΓΧΡΟΝΩΝ ΕΡΓΑΛΕΙΩΝ ΚΑΙ ΤΕΧΝΙΚΩΝ



ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: **ΒΑΣΙΛΕΙΟΣ ΒΕΣΚΟΥΚΗΣ**
ΣΥΓΓΡΑΦΗ: **ΙΩΑΝΝΗΣ ΜΟΥΤΑΦΗΣ**

ΑΘΗΝΑ, 2019

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον επιβλέποντα καθηγητή κ. Βασίλειο Βεσκούκη που ανέλαβε την επίβλεψη της εκπόνησης της εργασίας και καθοδήγησε την πορεία υλοποίησης της.
Θέλω ακόμη να ευχαριστήσω τον Δρ. Πουλίκο Πραστάκο (ΙΤΕ) και τον καθηγητή Τζίτζικα Ιωάννη (Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης) που συνετέλεσαν στη μέχρι τώρα πορεία μου.
Επιπλέον ευχαριστώ τις εταιρίες Vermantia[1] και Intelligems[2], στις οποίες είχα τη τιμή να εργαστώ, πρωτίστως για την ευκαιρία που μου δόθηκε να εργάζομαι και να παρακολουθώ ταυτόχρονα το Μεταπτυχιακό Πρόγραμμα Σπουδών Γεωπληροφορικής του ΕΜΠ αλλά και για την εμπειρία που απέκτησα, η οποία μου επέτρεψε να υλοποιήσω τη παρούσα εφαρμογή.
Τέλος θέλω να ευχαριστήσω την οικογένεια μου που με δίδαξε ότι με θέληση, προσπάθεια και επιμονή όλα τα εμπόδια καταλύονται.

[1] Vermantia Home Page: https://www.vermantia.com/
[2] Intelligems LinkedIn Page: https://www.linkedin.com/company/intelligems-technologies/

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η σύγχρονη οικονομική τάση γύρω από την παγκόσμια αγορά ηλεκτρικής ενέργειας υπαγορεύει την απελευθέρωση της με σκοπό την αύξηση της ανταγωνιστικότητας, την απόπειρα διάσπασης των μονοπωλίων, τον διαχωρισμό της συντήρησης του δικτύου διανομής από τη παραγωγή, τη δημιουργία μηχανισμού για το καθορισμό τιμών σε ένα περιβάλλον ανταγωνισμού και την ιδιωτικοποίηση Κρατικών πόρων (κατά το Βρετανικό Μοντέλο)[1].
Για τις ανάγκες της εύρυθμης λειτουργίας ενός τέτοιου μοντέλου είναι αναγκαία η σύσταση ενός νομικού πλαισίου καθώς και συστημάτων ελέγχων που θα βασίζονται στα δεδομένα παραγωγής και προμήθειας των επιχειρήσεων που εμπλέκονται με την παραγωγή και διάθεση ενέργειας. Η επίλυση αυτού του προβλήματος αποτελεί ακόμη και σήμερα πρόκληση τόσο σε επίπεδο χωρών, όσο και σε Παγκόσμιο επίπεδο.
Κινήσεις προς τη βιώσιμη εφαρμογή του Βρετανικού Μοντέλου έχουν εκτελεσθεί εντός της Ευρωπαϊκής Ένωσης που θεωρείται και πρωτοπόρος στο ζήτημα.

Η ΕΛΕΥΘΕΡΗ ΑΓΟΡΑ ΕΝΕΡΓΕΙΑΣ ΣΤΗΝ ΕΥΡΩΠΗ

Η διαδικασία απελευθέρωσης της αγοράς ενέργειας εντός της Ευρωπαϊκής Ένωσης ξεκινάει από το 1952 με την αγορά του χάλυβα και του άνθρακα και ακολουθείται το 1957 από την αγορά της ατομικής ενέργειας, ενώ κατά το τέλος της δεκαετίας του 90 εντάσσονται και οι αγορές του φυσικού αερίου και της ηλεκτρικής ενέργειας. Με την επιτυχή προσάρτηση των παραπάνω αγορών συγκροτείται η Εσωτερική Αγορά Ενέργειας (Internal Electricity Market, IEM ή από την ελληνική ονομασία ΕΑΕ) και θεσπίζονται κοινοί κανονισμοί μεταξύ των Κρατών-Μελών της Ένωσης για τον έλεγχο της τόσο νευραλγικής αυτής αγοράς.
Οι βασικοί πυλώνες πάνω στου οποίους διαμορφώνονται οι κανονισμοί που διέπουν την ΕΑΕ είναι η διασφάλιση της προμήθειας ενέργειας, καθώς και της πλήρους κάλυψης των αναγκών (ειδικά δε κατά τις ώρες αιχμής/αυξημένης ζήτησης), όλων ανεξαιρέτως και χωρίς διακρίσεις, των χωρών της Ένωσης.
Η τεχνική φύση όμως του προϊόντος καθιστά δύσκολη την εφαρμογή της κλασσικής εμπορικής εκμετάλλευσης που επιδέχονται άλλοι πόροι (όπως για παράδειγμα τρόφιμα, χάλυβας κ.α) καθώς η παραγόμενη ενέργεια δεν είναι εύκολα αποθηκεύσιμη. Ως συνέπεια αυτής της ιδιαιτερότητας, το διασυνδεδεμένο δίκτυο μονάδων παραγωγής που συμμετέχουν στην ΕΑΕ πρέπει να βρίσκεται συνεχώς υπό τέλειο συγχρονισμό παραγωγής και κατανάλωσης, διαφορετικά θα κινδυνεύει με τοπική ή ακόμη και ολική αστάθεια που δύναται να οδηγήσει σε κατάρρευση. Σε συνδυασμό με την τεχνική δυσκολία της παραγωγής και προμήθειας ηλεκτρικής ενέργειας εγείρεται και το πρόβλημα της οικονομικής ανισορροπίας εντός των ζωνών παραγωγής της ΕΑΕ, καθώς είναι υπαρκτός ο κίνδυνος ανάδυσης ιδιωτικών, ακόμη και "φυσικών"[2], μονοπωλίων γύρω από έναν τόσο σημαντικό πόρο χωρίς τον κατάλληλο ελεγκτικό μηχανισμό.

---

[1] Απελευθέρωση της αγοράς ηλεκτρικής ενέργειας: https://en.wikipedia.org/wiki/Energy_liberalisation

[2] Κατάσταση κατά την οποία λόγω υψηλού κόστους υποδομών, μεγέθους της αγοράς και άλλων εμποδίων, ο μεγαλύτερος προμηθευτής (και συνήθως ο πρώτος προμηθευτής που εμφανίστηκε σε αυτή την αγορά) κατέχει ένα υπερμέγεθες πλεονέκτημα έναντι των ανταγωνιστών του: https://en.wikipedia.org/wiki/Natural_monopoly

ΑΠΑΙΤΗΣΕΙΣ ΔΙΑΦΑΝΕΙΑΣ ΚΑΙ ΤΟ ΠΡΟΓΡΑΜΜΑ ENTSO-E

Όπως καθίσταται προφανές από τις προαναφερθείσες δυσκολίες που διέπουν την υλοποίηση της ΕΑΕ, είναι αναγκαία η ύπαρξη διαφάνειας όσον αφορά το σύνολο της αγοράς ενέργειας εντός της ΕΕ.
Με την ύπαρξη ελεύθερων και ενημερωμένων δεδομένων που αφορούν την παραγωγή, την κατανάλωση αλλά και την διανομή της ενέργειας, διευκολύνεται η αναγκαία διαδικασία ελέγχου.

ΠΡΟΓΡΑΜΜΑ ENTSO-E

Η European Network of Transmission System Operators for Electricity (ENTSO-E) εγκαθιδρύθηκε το 2009 με το 3ο Νομοθετικό Πακέτο για την ΕΑΕ της Ευρωπαϊκής Ένωσης. Η πλατφόρμα υφίσταται από τις 5 Ιανουαρίου 2015, σύμφωνα με τον *Ευρωπαϊκό Κανονισμό 543/2013*[1]. Σε αυτή συμμετέχουν 43 Διαχειριστές Δικτύων Διανομής ενέργειας (Transmission System Operators ή TCOs) από 36 Ευρωπαϊκές χώρες, που λειτουργούν ανεξάρτητα από τις εταιρίες που μετέχουν στην διαδικασία παραγωγής ηλεκτρικής ενέργειας και παρέχουν σε αυτές πρόσβαση στο δίκτυο μέσω αμερόληπτων και διαφανών κανόνων. Ακόμη, στις περισσότερες χώρες οι TSOs είναι υπεύθυνοι για την ανάπτυξη και συντήρηση του δικτύου.
Στόχος των συμμετεχόντων στο ENTSO-E είναι να εγκαθιδρύσουν την ΕΑΕ και να εξασφαλίσουν την εύρυθμη λειτουργία της, καθώς επίσης και να υποστηρίξουν το κλιματικό πρόγραμμα της ΕΕ με την χρήση όλο και περισσότερων δομών παραγωγής μέσω ανανεώσιμων πηγών ενέργειας.

ΔΕΔΟΜΕΝΑ ΔΙΑΦΑΝΕΙΑΣ ΜΕΣΩ ΤΟΥ ENTSO-E

Στα πλαίσια του προγράμματος ENTSO-E έχει δημιουργηθεί η Πλατφόρμα Διαφάνειας (ENTSO-E Transparency Platform)[2] στην οποία αποθηκεύονται δεδομένα παραγωγής σύμφωνα με κανόνες που αναφέρονται στο Εγχειρίδιο Διαδικασιών[3], όπως αυτό ορίζεται από το άρθρο 5 του 543/2013. Τα δεδομένα αυτά είναι ανοικτά και η πρόσβαση ελεύθερη προς το κοινό.

Τα ενεργειακά δεδομένα χωρίζονται σε 5 κύριες θεματικές ενότητες[4]:

- Παραγωγή (Δεδομένα: παρελθοντικά, επίκαιρα και πρόγνωση)
- Κατανάλωση & Φόρτος Δικτύου (Δεδομένα: παρελθοντικά, επίκαιρα και πρόγνωση)
- Συναλλαγές (Δεδομένα: παρελθοντικά, επίκαιρα και πρόγνωση)
- Εξισορρόπηση και Αποθέματα (Δεδομένα: επίκαιρα)
- Διακοπές Παροχής (Δεδομένα: επίκαιρα)

---

[1] EUR-Lex 543/2013: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32013R0543
[2] ENTSO-E Transparency Platform: https://transparency.entsoe.eu/
[3] Manual of Procedures: https://www.entsoe.eu/data/transparency-platform/mop/
[4] Αναλυτική Περιγραφή Δεδομένων: https://www.entsoe.eu/fileadmin/user_upload/_library/resources/Transparency/02_MoP%20Ref02%20-%20DDD_V2R5.pdf

ΑΠΑΙΤΗΣΕΙΣ ΧΩΡΙΚΗΣ ΑΠΕΙΚΟΝΙΣΗΣ ΔΕΔΟΜΕΝΩΝ

Κατά τη χρονική περίοδο συγγραφής της παρούσας μελέτης, η πλατφόρμα διαφάνειας του ENTSO-E δεν παρέχει τη δυνατότητα χωρικής απεικόνισης της πληροφορίας και η μοναδική χωρική σύνδεση γίνεται με αναφορά στο όνομα των χωρών η των ζωνών (TCOs) παραγωγής. Ακόμη η πλατφόρμα δεν διευκολύνει την απευθείας σύγκριση δεδομένων των θεματικών ενοτήτων μεταξύ χωρών ή ζωνών. Μοναδική εξαίρεση αποτελεί η θεματική ενότητα των Συναλλαγών, με μια περιορισμένη, ανά χώρα, οπτικοποίηση των κινήσεων ηλεκτρικής ενέργειας σε χάρτη, χωρίς όμως να δίνει τη δυνατότητα και αυτή για σύγκριση των δεδομένων.
Η έλλειψη χωρικής απεικόνισης αλλά και δυνατοτήτων σύγκρισης του μεγάλου όγκου δεδομένων που αποθηκεύονται καθημερινά στη πλατφόρμα, καθιστούν τη παρεχόμενη πληροφορία δύσχρηστη.

ΕΣΤΙΑΣΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Η παρούσα μελέτη επικεντρώνεται στη διαδικασία δημιουργίας μιας διαδικτυακής GIS εφαρμογής που υλοποιεί την χωρική σύνδεση επιλεγμένων δεδομένων, από τις υπάρχουσες θεματικές ενότητες, με τις χώρες από τις οποίες προέρχεται η πληροφορία. Η εφαρμογή αντλεί δεδομένα απευθείας από την πλατφόρμα του ENTSO-E και αναλαμβάνει την χωρική τους απεικόνιση καθώς και την οπτικοποίηση τους μέσω παραστατικών γραφημάτων. Ακόμη δίνει τη δυνατότητα σύγκρισης της συλλεγμένης πληροφορίας μεταξύ χωρών, με τη ταυτόχρονη χρήση ενός διαδραστικού χάρτη και των γραφικών αναπαραστάσεων των δεδομένων.

ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στόχο της υλοποίησης αποτελεί η βελτιστοποίηση της παραστατικότητας και της χρήσης των δεδομένων της πλατφόρμας ENTSO-E. Για την επίτευξη του σκοπού αυτού η εφαρμογή εκτελεί 2 κεντρικές λειτουργίες:

- **Συλλογή Δεδομένων από το ENTSO-E:**

  Για την υλοποίηση της εφαρμογής επιλέχθηκαν 3 πηγές δεδομένων από τη θεματική ενότητα της Παραγωγής, καθώς αποτελούν ικανό σύνολο για την ανάλυση της ενεργειακής κατάστασης κάθε χώρας. Οι πηγές δεδομένων συνοπτικά[1] είναι οι ακόλουθες:

  1. Δυνατότητες παραγωγής ενέργειας σε μεγαβατώρες ανά είδος (ηλιακή, πυρηνική κτλ.) των εγκαταστάσεων κάθε χώρας.

  2. Ημερήσια ωριαία παραγωγή ενέργειας σε μεγαβατώρες ανά είδος, κάθε χώρας.

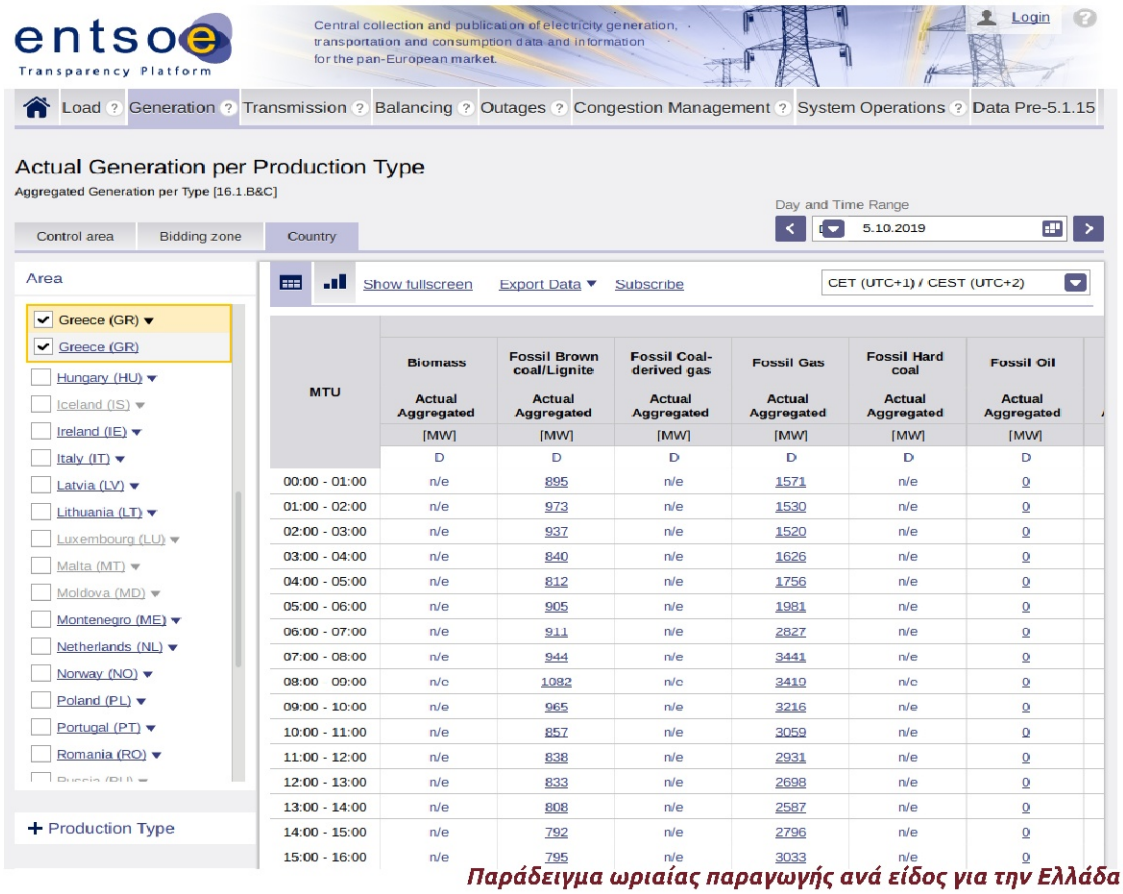
[1] Συνολικότερη περιγραφή των πηγών δεδομένων που επιλέχθηκαν ακολουθεί στην ενότητα ΣΧΕΔΙΑΣΜΟΣ ΜΕ ΒΑΣΗ ΤΑ ΔΕΔΟΜΕΝΑ του κεφαλαίου ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

3. Ημερήσια ωριαία πρόγνωση συνολικής παραγωγής (χωρισμένης σε πρόγνωση του ωριαίου συνόλου παραγωγής από ανανεώσιμες πηγές και από μη ανανεώσιμες πηγές) και κατανάλωσης ενέργειας σε μεγαβατώρες, κάθε χώρας.

- **Απεικόνιση Δεδομένων με Χωρική Αναφορά.**

Μετά τη συλλογή και την κατάλληλη επεξεργασία των δεδομένων[1], αυτά συνδέονται χωρικά, σε επίπεδο βάσης δεδομένων, με τις χώρες στις οποίες αναφέρονται. Για τις χώρες αυτές υπάρχει στη βάση της εφαρμογής η πληροφορία των επίσημων συνόρων τους σε μορφή MULTIPOLYGON[2].

Αυτός ο συνδυασμός χωρικής πληροφορίας και δεδομένων παραγωγής και κατανάλωσης ενέργειας εντός της εφαρμογής, δίνει τη δυνατότητα αναλυτικότερης οπτικοποίησης της συλλεγμένης πληροφορίας, καθώς και σύγκρισής της μεταξύ των χωρών που συμμετέχουν στο πρόγραμμα ENTSO-E.



Παράδειγμα ωριαίας παραγωγής ανά είδος για την Ελλάδα

[1] Αναλυτική περιγραφή της διαδικασίας επεξεργασίας των δεδομένων βρίσκεται στην ενότητα ΣΧΕΔΙΑΣΜΟΣ ΜΕ ΒΑΣΗ ΤΑ ΔΕΔΟΜΕΝΑ του κεφαλαίου ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

[2] Η χωρική οντότητα αυτή αποτελείται από ένα σύνολο πολυγώνων: https://postgis.net/workshops/postgis-intro/geometries.html#collections

ΣΥΝΤΟΜΗ ΠΑΡΟΥΣΙΑΣΗ ΕΡΓΑΛΕΙΩΝ

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν αποκλειστικά εργαλεία ανοικτού κώδικα (open-source) τα οποία αποτελούν επαγγελματικά πρότυπα στον χώρο της Ανάπτυξης Λογισμικού.

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

- **PostgreSQL:**
  Σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων σχεδιασμένο να επιτρέπει την παράλληλη χρήση από πολλούς χρήστες.
  - **PostGIS:**
    Επέκταση της PostgreSQL που προσθέτει υποστήριξη για γεωγραφικά δεδομένα καθώς και μια πληθώρα εργαλείων-συναρτήσεων για την επεξεργασία τους.
- **Redis:**
  Σύστημα αποθήκευσης δομών δεδομένων (strings, hashes, lists κτλ.). Χρησιμοποιείται ως μη σχεσιακή βάση δεδομένων, cache και σαν message broker[1] μεταξύ τμημάτων της εφαρμογής.

BACK-END

Με τον όρο back-end αναφερόμαστε στο επίπεδο του κώδικα που αφορά τις διεργασίες που εκτελούνται στον server.

- **Python:**
  Γενικού σκοπού, υψηλού επιπέδου, διερμηνευόμενη (δεν χρησιμοποιεί compiler), αντικειμενοστραφής γλώσσα προγραμματισμού.
- **Django:**
  Πλαίσιο ανάπτυξης περίπλοκων διαδικτυακών εφαρμογών βασισμένο στη Python, το οποίο απλοποιεί τη διαδικασία ανάπτυξης τους.
  - **Django Rest Framework:**
    Επέκταση του Django που επιτρέπει την ανάπτυξη REST API[2].
  - **Django Webpack Loader:**
    Επέκταση του Django που επιτρέπει στον server της εφαρμογής να εξυπηρετεί και τον κώδικα του front-end[3] σαν στατικό αρχείο.

---

[1] Διανομέας μηνυμάτων: Σύστημα επικοινωνίας τύπου ουράς (queue) στην οποία γράφονται μηνύματα από ένα μέλος της εφαρμογής με σκοπό να διαβιβαστούν σε άλλα μέλη για την εκτέλεση κατάλληλων διεργασιών.

[2] βλ. ενότητα ΥΛΟΠΟΙΗΣΗ BACK-END του κεφαλαίου ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ

[3] βλ. ενότητα FRONT-END του παρόντος κεφαλαίου καθώς και του κεφαλαίου ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ

- **Celery:**
  Γραμμή εργασιών (tasks) που επιτρέπει την ασύγχρονη[1] εκτέλεση αυτών από ρουτίνες που ονομάζονται "εργάτες" και στηρίζεται σε ένα διανομέα μηνυμάτων (σε αυτή την εφαρμογή τη θέση αυτή καλύπτει η βάση Redis που αναφέρεται παραπάνω).

FRONT-END

Με τον όρο front-end αναφερόμαστε στο επίπεδο του κώδικα που αφορά τις διεργασίες που εκτελούνται στον browser.

- **JavaScript**
  Γενικού σκοπού, υψηλού επιπέδου, διερμηνευόμενη, αντικειμενοστραφής γλώσσα προγραμματισμού. Σε συνδυασμό με την HTML και τη CSS αποτελεί τον πυρήνα των διαδραστικών διαδικτυακών σελίδων.

- **Highcharts**
  Βιβλιοθήκη βασισμένη σε Javascript για τη δημιουργία γραφημάτων.

- **React**
  Πλαίσιο ανάπτυξης διαδραστικών διεπαφών (UI) χρήστη-υπολογιστή βασισμένο στη Javascript. Έχει αναπτυχθεί από τη Facebook για της ανάγκες (αρχικά) της πλατφόρμας.

  - **React Material UI**
    Βιβλιοθήκη βασισμένη σε React η οποία επιτρέπει την δημιουργία διεπαφών με βάση τα πρότυπα περί design που εξέδωσε η Google το 2014 και συνολικά ονομάζονται Material Design ή Quantum Paper.

- **Redux**
  Βιβλιοθήκη βασισμένη σε Javascript για την διατήρηση και διακίνηση δεδομένων που ονομάζονται state (κατάσταση) ανάμεσα στα δομικά στοιχεία (components) μιας React εφαρμογής.

- **Leaflet**
  Βιβλιοθήκη βασισμένη σε Javascript για την δημιουργία διαδραστικών χαρτών.

- **Webpack Loader**
  Συνενώνει τα δομικά στοιχεία που απαρτίζουν το front-end μέρος της εφαρμογής (HTML, CSS, Javascript αρχεία) σε στατικά αρχεία με αποτέλεσμα να μειώνουν το χρόνο απόκρισης του server.

[1] Ασύγχρονος Προγραμματισμός: https://en.wikipedia.org/wiki/Asynchrony_(computer_programming)

ΥΠΟΔΟΜΗ

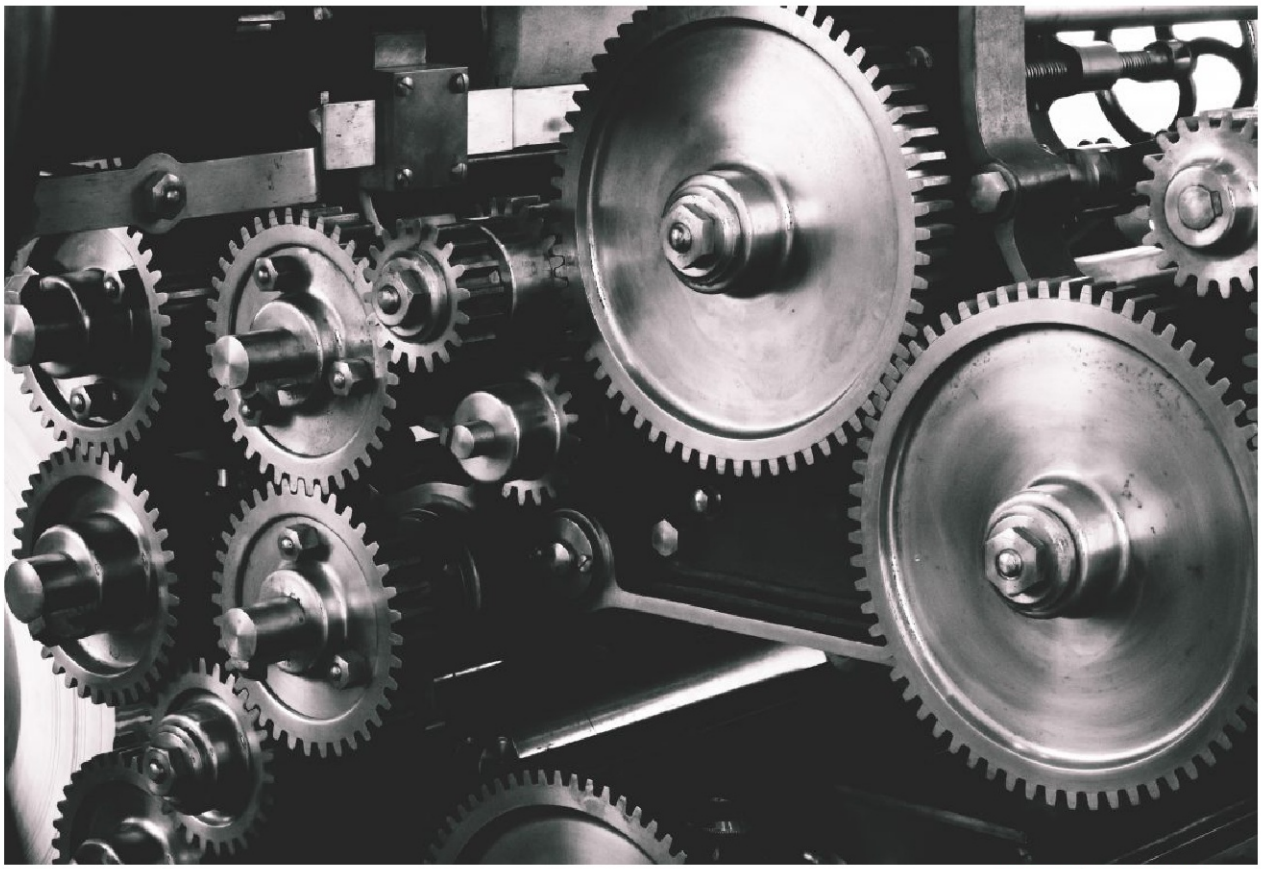
Με τον όρο υποδομή (infrastructure) αναφερόμαστε στο σύνολο των δομικών στοιχείων που επιτρέπουν την εγκατάσταση διαδικτυακών εφαρμογών στον server.

- **Docker**

  Ένα σύνολο υπηρεσιών που επιτρέπουν την εικονική δημιουργία "εικόνων" εφαρμογών μέσω script. Επιτρέπει την ακριβή επανάληψη της διαδικασία εγκατάστασης ανεξαρτήτως λειτουργικού περιβάλλοντος του server που θα φιλοξενήσει την εφαρμογή.

  - **Docker Compose**

    Επιτρέπει την ταυτόχρονη εγκατάσταση εφαρμογών με κοινό και ανεξάρτητο εικονικό δίκτυο δημιουργώντας ουσιαστικά ένα cluster αποκλειστικά για την εφαρμογή.



"Δομικά στοιχεία"

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ

Ο σχεδιασμός της εφαρμογής περιστρέφεται γύρω από την ροή των δεδομένων από την βάση του ENTSO-E, προς το back-end μέρος της εφαρμογής και μέσω του front-end μέρους προς τον χρήστη.

Με γνώμονα αυτό, η ανάπτυξη της εφαρμογής στηρίζεται εν μέρει στο **Data Driven**[1] (σχεδιασμός καθοδηγούμενος από τα δεδομένα και τη μορφή τους) μοντέλο. Είναι όμως ταυτόχρονα και μια διαδικτυακή εφαρμογή η οποία πρέπει να μπορεί να αναπαράγεται (deployed) και να λειτουργεί με τον ίδιο τρόπο σε οποιονδήποτε server. Για το λόγο αυτό, το βασικό μοντέλο πάνω στο οποίο στηρίζεται ο σχεδιασμός της εφαρμογής είναι αυτό του **12 Factor App**[2].

ΣΧΕΔΙΑΣΜΟΣ ΣΤΑ ΠΡΟΤΥΠΑ ΤΟΥ 12 FACTOR APP MODEL

Είναι ένα σύνολο μεθόδων που αναπτύχθηκε από προγραμματιστές της cloud πλατφόρμας Heroku[3] και παρουσιάστηκε από τον Adam Wiggins το 2011. Ο σκοπός του προτύπου αυτού είναι να καταγράψει τις βέλτιστες πρακτικές για την υλοποίηση εφαρμογών με ανθεκτικότητα και φορητότητα[4] όταν αυτές αναπτύσσονται στο διαδίκτυο.

Οι συντελεστές (factors) που ορίζει το πρότυπο ως βέλτιστες πρακτικές είναι 12 (από το οποίο προκύπτει και το όνομα). Για το μέγεθος της τελικής εφαρμογής, στο σχεδιασμό βασικό ρόλο είχαν οι εξής:

1. **Κώδικας (Codebase):** Ο κώδικας της εφαρμογής πρέπει να βρίσκεται αποθηκευμένος σε ένα σύστημα ελέγχου της έκδοσής του (version control system, για αυτή την εφαρμογή χρησιμοποιήθηκε η δημοφιλής πλατφόρμα github[5]). Μόνο αυτός ο κώδικας πρέπει να χρησιμοποιείται για διαφορετικά deployments της εφαρμογής.

2. **Εξαρτήσεις περιβάλλοντος (Dependencies):** Αυτές περιλαμβάνουν βιβλιοθήκες και άλλα εργαλεία που είναι αναγκαία για την λειτουργία της εφαρμογής. Αυτές οι εξαρτήσεις πρέπει να δηλώνονται ρητά (explicitly) και να είναι αυτόνομες σε σχέση με το λειτουργικό σύστημα (OS agnostic).
   Στη παρούσα εφαρμογή, οι εξαρτήσεις του back-end βρίσκονται στα αρχείο *requirements.txt* και οι εξαρτήσεις του front-end στο αρχείο *package.json*

3. **Παράμετροι Συστήματος (Config):** Το σύνολο των παραμέτρων του συστήματος (που δύναται να διαφέρει ανάμεσα σε διαφορετικά deployments της ίδιας εφαρμογής) πρέπει να αποθηκεύεται στο εκάστοτε περιβάλλον στο οποίο θα αναπτυχθεί η εφαρμογή και δεν πρέπει να αποθηκευτεί **ποτέ** στο version control σύστημα.
   Στη παρούσα εφαρμογή υπάρχει ένα αρχείο *env.example* που αποτελεί το υπόδειγμα για τη (χειροκίνητη) δημιουργία του αρχείου *.env* το οποίο θα αποθηκευτεί μόνο στον

[1] βλ. ενότητα ΣΧΕΔΙΑΣΜΟΣ ΜΕ ΒΑΣΗ ΤΑ ΔΕΔΟΜΕΝΑ ΤΟΥ ENTSO-E αυτού του κεφαλαίου.
[2] 12 Factor App: https://12factor.net/
[3] Heroku Homepage: https://www.heroku.com/
[4] Με τον όρο "φορητότητα" μιας εφαρμογής εννοούμε την ευκολία με την οποία μπορούμε να αναπαράγουμε την εφαρμογή σε οποιονδήποτε server.
[5] Github Homepage: https://github.com/

server στον οποίο θα γίνει η ανάπτυξη της εφαρμογής και θα περιέχει τις παραμέτρους του συστήματος για το συγκεκριμένο deploy και μόνο.

4. **Υποστηρικτικές Υπηρεσίες (Backing Services):** Οι υπηρεσίες που χρησιμοποιεί η εφαρμογή (πχ. βάσεις αποθήκευσης δεδομένων όπως η PostgreSQL ή message brookers όπως η Redis) πρέπει να θεωρούνται συνημμένοι πόροι και η εφαρμογή να μην αλλάζει τρόπο προσπέλασης των πόρων αυτών ακόμη και αν αλλάζει ο πάροχος τους. Δηλαδή, πρέπει η εφαρμογή να αξιοποιεί με τον ίδιο τρόπο την βάση δεδομένων, είτε αν πρόκειται για τοπική βάση εντός του server (PosrgreSQL ή mySQL) είτε αν πρόκειται για βάση με πάροχο την Amazon (AWS RDS) ή την Microsoft (Azure RDS). Η αναφορά των "τοποθεσιών" αυτών των υπηρεσιών πρέπει να δηλώνεται στις παραμέτρους του συστήματος για κάθε deploy.

5. **Μέθοδοι (Processes):** Η εφαρμογή πρέπει να εκτελείται ως ένα σύνολο μεθόδων που δεν διατηρεί "μνήμη" της κατάστασης κάθε επιμέρους μεθόδου από προηγούμενες εκτελέσεις. Ό,τι είναι ανάγκη να διατηρηθεί για επόμενες εκτελέσεις πρέπει να φυλάσσεται σε κάποια/ες υποστηρικτική υπηρεσία (συνήθως σε βάση δεδομένων).
   Στη παρούσα εφαρμογή, τόσο το back-end (Django) όσο και το front-end (React) διασφαλίζουν de facto την τήρηση αυτής τη πρακτικής.

6. **Αρχείο Καταγραφής (Logs):** Η πορεία εξέλιξης της εκτέλεσης των επιμέρους μεθόδων πρέπει να καταγράφεται ως μια συνεχής (χρονολογικά) ροή συμβάντων.

7. **Εργασίες Διαχείρισης (Admin Tasks):** Οι διαχειριστικές εργασίες (δημιουργία πινάκων στη βάση, ενημέρωση βάσης ανά συγκεκριμένα χρονικά πλαίσια ή χειροκίνητα, κτλ.) πρέπει να είναι αυτοτελείς και ανεξάρτητες από την κύρια ροή της εφαρμογής. Στη παρούσα εφαρμογή, το back-end διασφαλίζει de-facto τη τήρηση αυτής της πρακτικής, ενώ το front-end δεν περιέχει διαχειριστικές εργασίες.

Οι συντελεστές που ακολουθούν αφορούν τον σχεδιασμό της εφαρμογής στο επίπεδο της Υποδομής και εφαρμόζονται με ημιαυτόματο τρόπο από το σύστημα Docker[1]. Αυτός είναι και ο κυρίαρχος λόγος της επιλογής αυτού του συστήματος για την υλοποίηση της υποδομής.

1. Αυστηρός διαχωρισμός της διαδικασίας μεταμόρφωσης του Codebase σε μια εκτελέσιμη δέσμη (bundle) αλληλένδετων δομικών υλικών από την διαδικασία εκτέλεσής τους.

2. Τα μέρη της εφαρμογής που πρέπει να επικοινωνούν μέσω http είτε εσωτερικά, είτε εξωτερικά, πρέπει να το κάνουν αυτό αποκλειστικά μέσω δεσμευμένων θυρών του δικτύου.

3. Η κλιμάκωση/αποκλιμάκωση της εφαρμογής πρέπει να γίνεται με επαύξηση (κλωνοποίηση)/μείωση (διαγραφή κλώνου) του αριθμού των processes που εκτελούνται παράλληλα.

[1] βλ. Docker στην ενότητα ΥΠΟΔΟΜΗ του κεφαλαίου ΣΥΝΤΟΜΗ ΠΑΡΟΥΣΙΑΣΗ ΕΡΓΑΛΕΙΩΝ

4. Τα εκτελούμενα processes πρέπει να είναι αναλώσιμα. Αυτό σημαίνει να μπορούν να εκκινούνται και να τερματίζονται με ταχύτητα και χωρίς να αφήνουν κατάλοιπα εκτελέσεων στο σύστημα.

5. Το περιβάλλον των δοκιμών (Dev & Staging environments) και το περιβάλλον της παραγωγής (Prod environment) - δηλαδή το περιβάλλον από το οποίο η εφαρμογή θα παρέχεται προς επίσημη κατανάλωση (consumption) - πρέπει να είναι όσο το δυνατόν όμοια.

ΣΧΕΔΙΑΣΜΟΣ ΜΕ ΒΑΣΗ ΤΑ ΔΕΔΟΜΕΝΑ ΤΟΥ ENTSO-E

Όπως αναφέρθηκε στην ενότητα της "Εστίασης της Εργασίας"[1], σκοπός της εφαρμογής είναι να συλλέγει δεδομένα από συγκεκριμένα datasets της πλατφόρμας ENTSO-E τα οποία προσφέρονται μέσω API (Application Programming Interface, δηλαδή ένα σύνολο από URL που επιτρέπουν τη πρόσβαση στα δεδομένα της πλατφόρμας μέσω http) και να τα αποθηκεύει στη βάση δεδομένων για μετέπειτα χρήση.

Τα datasets που επιλέχθηκαν είναι τα εξής:

1. **Installed Generation Capacity Aggregated (document A68, process A33)**

   Αποδίδει το σύνολο των δυνατοτήτων παραγωγής ανά μέρα των υποδομών κάθε χώρας, χωρισμένη ανά είδος παραγωγής (υδροηλεκτρική, ηλιακή, από πετρέλαιο κτλ.). Ενημερώνεται για τις περισσότερες χώρες μια φορά το χρόνο.

2. **Aggregated Generation per Type (document A75, process A16)**

   Αποδίδει τη συνολική ωριαία παραγωγή ανά είδος παραγωγής κάθε χώρας. Ενημερώνεται για τις περισσότερες χώρες ανά μια ώρα, αλλά για κάποιες χώρες (πχ. Ηνωμένο Βασίλειο) ενημερώνεται ανά μισή ώρα.

3. **Aggregated Generation with Day Ahead Forecast (document A71, process A01)**

   Αποδίδει την προβλεπόμενη συνολική ωριαία παραγωγή κάθε χώρας για την επόμενη ημέρα καθώς και το πραγματικό σύνολο της ωριαίας παραγωγής.
   Ενημερώνεται κάποια (μη σταθερή ανά χώρα) στιγμή της ημέρας η πρόγνωση της συνολικής ωριαίας παραγωγής για την επόμενη μέρα και ενημερώνεται ανά ώρα το σύνολο της πραγματικής παραγωγής της τρέχουσας ημέρας. Η πυκνότητα των δεδομένων και εδώ μπορεί να διαφέρει ανά χώρα σε ωριαία και ημίωρη.

[1] βλ. ενότητα ΕΣΤΙΑΣΗ ΤΗΣ ΕΡΓΑΣΙΑΣ του κεφαλαίου ΑΠΑΙΤΗΣΕΙΣ ΧΩΡΙΚΗΣ ΑΠΕΙΚΟΝΙΣΗΣ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα από τη πλατφόρμα αποδίδονται σε μορφή XML[1] όπως στο παράδειγμα που ακολουθεί:

```
<GL_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-6:generationloaddocument:3:0">
        <mRID>21e76d7172cd4fdabae90b9676096dbf</mRID>
        <revisionNumber>1</revisionNumber>
        <type>A75</type>
        <process.processType>A16</process.processType>
        <sender_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</sender_MarketParticipant.mRID>
        <sender_MarketParticipant.marketRole.type>A32</sender_MarketParticipant.marketRole.type>
        <receiver_MarketParticipant.mRID codingScheme="A01">10X1001B1001B61Q</receiver_MarketParticipant.mRID>
        <receiver_MarketParticipant.marketRole.type>A33</receiver_MarketParticipant.marketRole.type>
        <createdDateTime>2016-05-10T12:43:07Z</createdDateTime>
        <time_Period.timeInterval>
                <start>2015-12-31T23:00Z</start>
                <end>2016-12-31T23:00Z</end>
        </time_Period.timeInterval>
        <TimeSeries>
                <mRID>1</mRID>
                <businessType>A01</businessType>
                <objectAggregation>A08</objectAggregation>
                <inBiddingZone_Domain.mRID codingScheme="A01">10YCZ-CEPS-----N</inBiddingZone_Domain.mRID>
                <quantity_Measure_Unit.name>MAW</quantity_Measure_Unit.name>
                <curveType>A01</curveType>
                <MktPSRType>
                        <psrType>B02</psrType>
                </MktPSRType>
                <Period>
                        <timeInterval>
                                <start>2015-12-31T23:00Z</start>
                                <end>2016-01-01T23:00Z</end>
                        </timeInterval>
                        <resolution>PT60M</resolution>
                        <Point>
                                <position>1</position>
                                <quantity>3316</quantity>
                        </Point>
                        <Point>
                                <position>2</position>
                                <quantity>3114</quantity>
                        </Point>
                        <Point>
                                <position>3</position>
                                <quantity>3145</quantity>
```

Εδώ φαίνεται το XML αρχείο ενημέρωσης της ωριαίας πραγματικής παραγωγής ανά είδος παραγωγής του document A75.

[1] Extensible Markup Language (XML) file format: https://en.wikipedia.org/wiki/XML

Για τις ανάγκες λήψης και αποθήκευσης των δεδομένων από τη πλατφόρμα σχεδιάστηκε και υλοποιήθηκε μια διαδικασία τύπου **ETL (Extract - Transform - Load)**[1]:

- **Extract:** Η διαδικασία εξαγωγής των δεδομένων από τη πλατφόρμα υλοποιήθηκε με μια σειρά από scripts που χρησιμοποιούν τη βιβλιοθήκη δημιουργίας και εκτέλεσης διαδικτυακών κλήσεων μέσω http (http requests) της Python, την *requests*[2].
  Τα scripts αυτά εκτελούν 3 λειτουργίες για να επιτύχουν την εξόρυξη των δεδομένων:
  - Συνθέτουν τα κατάλληλα ορίσματα (arguments) που χρειάζονται για την εξαγωγή δεδομένων από τα επιλεγμένα datasets και εκτελούν τις αντίστοιχες http κλήσεις.
  - Χειρίζονται περιπτώσεις σφαλμάτων που μπορούν να προκύψουν κατά την εκτέλεση των κλήσεων, αποτρέποντας την πιθανότητα καταστροφικής αποτυχίας της εφαρμογής κατά την εξόρυξη. Πιθανά σφάλματα είναι:
    - *400 - Bad Request:* Λάθος διαμορφωμένα ορίσματα κατά τη κλήση.
    - *401 - Unauthorized:* Απαγόρευση λήψης δεδομένων για τα παρεχόμενα διαπιστευτήρια χρήστη.
    - *404 - Not Found:* Η διεύθυνση URL δεν υπάρχει.
    - *50x - Server Side Errors:* Ο server δεν αποκρίνεται (501-502-503) ή αντιμετωπίζει σφάλματα κατά την εκτέλεση της εφαρμογής (500).
  - Επιστρέφουν τα δεδομένα για να περάσουν το στάδιο της μετατροπής.

- **Transform:** Τα δεδομένα από τη μορφή XML στην οποία αποδίδονται πρέπει να μετατραπούν σε Python objects για να χρησιμοποιηθούν στη συνέχεια στη φάση του Load και να αποθηκευτούν στη Βάση.
  Για τη διαδικασία του μετασχηματισμού υλοποιήθηκε ένας parser[3] για κάθε dataset με σκοπό να εξάγει τα δεδομένα και να τα φέρει σε μια μορφή κατάλληλη για αποθήκευση στη βάση δεδομένων με βάση το σχεδιασμό της τελευταίας[4].

- **Load:** Στο τέλος των προαναφερθέντων διαδικασιών, τα δεδομένα είναι σε μορφή που μπορεί να χρησιμοποιηθεί για να φορτωθούν τα δεδομένα και να αποθηκευτούν στη βάση δεδομένων της εφαρμογής, στους αντίστοιχους πίνακες που δημιουργήθηκαν για κάθε dataset. Η αποθήκευση γίνεται μέσω μεθόδων που περιέχονται στο Django framework[5].

---

[1] ETL: https://en.wikipedia.org/wiki/Extract,_transform,_load

[2] Requests Library Documentation: https://2.python-requests.org/en/master/

[3] Script το οποίο κάνει συντακτική ανάλυση (parsing) του αρχείου XML και εξάγει τα δεδομένα αυτά που έχουν οριστεί ως σημαντικά

[4] βλ. ενότητα ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ αυτού του κεφαλαίου.

[5] βλ. ενότητα ΥΛΟΠΟΙΗΣΗ BACK-END αυτού του κεφαλαίου.

ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Ο σχεδιασμός έγινε με γνώμονα την σωστή αναπαράσταση και αποθήκευση των δεδομένων που καταναλώνονται από τη πλατφόρμα ENTSO-E σε σχεσιακή βάση δεδομένων, χωρίς περιττές εγγραφές ή πλεονασμό δεδομένων, με έμφαση στην ευκολία ανάκτησης δεδομένων από τη βάση για χρήση εντός της εφαρμογής.

Η τελική μορφή της βάσης είναι Κανονικοποιημένη[1] (Normalized) και πιο συγκεκριμένα είναι στα πλαίσια της Κανονικής Μορφής **Boyce-Codd**[2] (Boyce-Codd Normal Form ή BCNF).

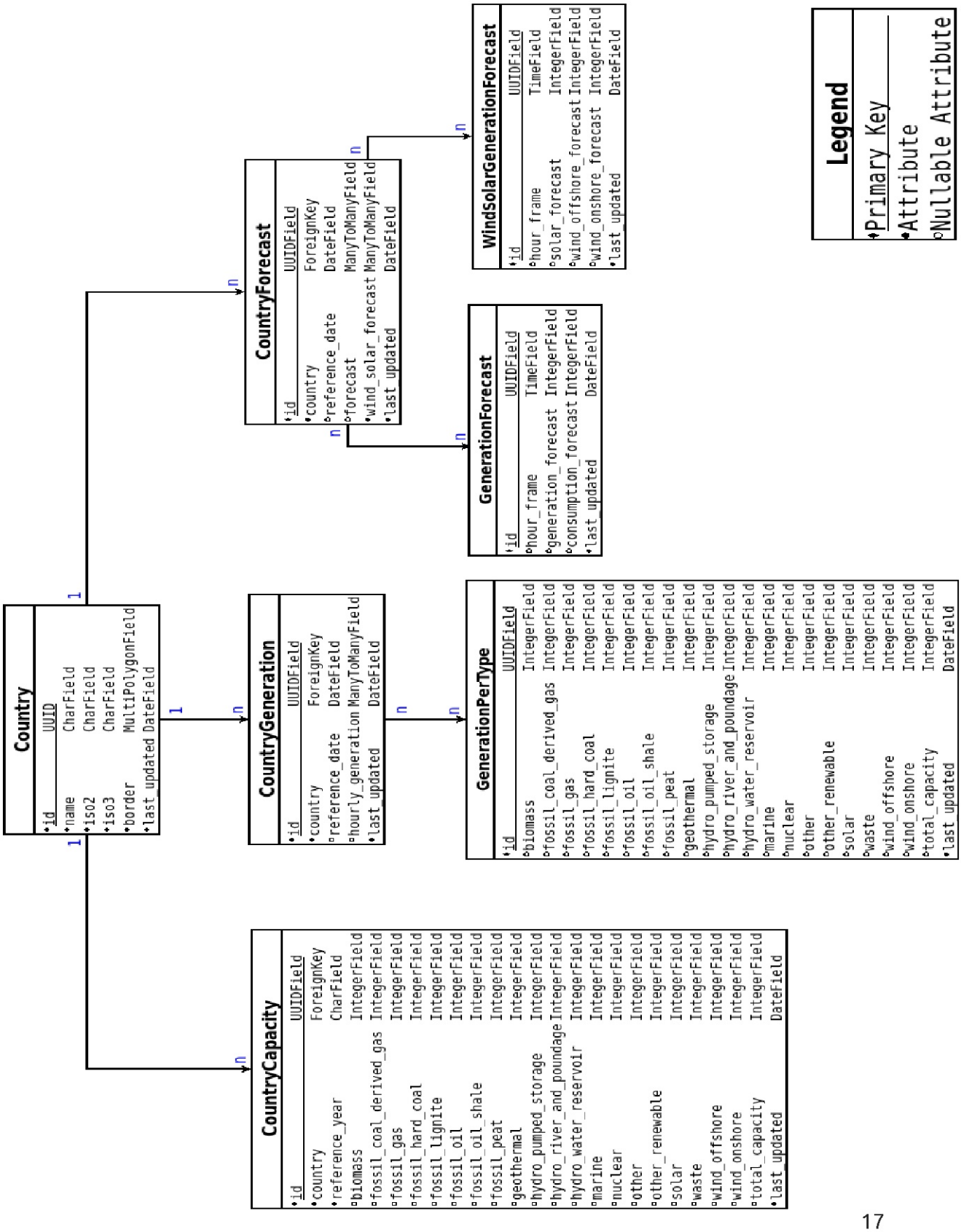
- **Κανονικοποιηση Βασης Δεδομενων**
  Είναι η διαδικασία *σύνθεσης* (εάν πρόκειται για κατασκευή της βάσης εξ αρχής) ή *αποσύνθεσης* (εάν πρόκειται για κανονικοποίηση υπάρχουσας βάσης) μιας σχεσιακής βάσης δεδομένων με βάση μια σειρά από κλιμακούμενους κανόνες που ονομάζονται Κανονικές Φόρμες και κάθε Φόρμα αποτελεί υπερσύνολο της προηγούμενης όσον αφορά τους κανόνες που θέτει.
  Σκοπός της διαδικασίας είναι να μειώσει τις περιττές εγγραφές δεδομένων στη βάση (data redundancy) και να βελτιώσει την ακεραιότητα (integrity) των δεδομένων.

- **Κανονικη Μορφη Boyce - Codd**
  Είναι μια εξέλιξη της 3ης Κανονικής Μορφής (αναφέρεται και ως 3.5NF) και κατά συνέπεια περιέχει τους κανόνες των 1, 2, 3 Κανονικών Φορμών και προσθέτει 1 κανόνα ακόμη:

  - *(1NF)* Όλα τα πεδία των πινάκων πρέπει να είναι ατομικά.
    Εξήγηση: Οι τιμές των πεδίων πρέπει να είναι αδιαίρετες όσον αφορά την πληροφορία που περιέχουν (πχ. τιμές <Country Name> - <ISO> δεν είναι ατομικές γιατί μπορούν να αναλυθούν σε 2 στήλες: <Country Name> και <ISO>
  - *(2NF)* Κανένα χαρακτηριστικό, που δεν αποτελεί μέρος υποψήφιου κλειδιού, δεν προσδιορίζεται από τμήμα του υποψήφιου κλειδιού του πίνακα.
  - *(3NF)* Κανένα χαρακτηριστικό, που δεν αποτελεί μέρος υποψήφιου κλειδιού, δεν προσδιορίζεται μεταβατικά από τμήμα ενός υποψηφίου κλειδιού.
    Εξήγηση: Κανένας πίνακας δεν περιέχει στήλες δεδομένων που συνδέονται μεταξύ τους με μεταβατική σχέση (**α** -> β), (β -> γ) όπου **α** μέρος του υποψήφιου κλειδιού.
  - Τέλος μια Κανονικοποιημένη Βάση είναι σε μορφή BCNF εάν και μόνο εάν για κάθε συναρτησιακή σχέση (α -> β) ισχύει ένα από τα παρακάτω:
    - Το β ⊆ α
    - Το α αποτελεί υποψήφιο κλειδί του συνόλου των δεδομένων (superkey).

[1] Κανονικοποίηση Βάσης Δεδομένων: https://en.wikipedia.org/wiki/Database_normalization
[2] Boyce-Codd Κανονική Μορφή: https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form

Τηρώντας όλους τους κανόνες της BCNF θεωρούμε ότι δεν έχουμε κίνδυνο πλεονασμού δεδομένων και η βάση της εφαρμογής αναπαρίσταται γραφικά ως εξής:



ΥΛΟΠΟΙΗΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Η βάση υλοποιήθηκε με οδηγό τον σχεδιασμό που προαναφέρθηκε και έχει τη δομή που εμφανίζεται στο προηγούμενο σχήμα. Η υλοποίηση έγινε με τη χρήση των εργαλείων του Django για τη δημιουργία μοντέλων βάσεων δεδομένων (Django Database Models)[1] που επιτρέπουν την αναπαράσταση πινάκων σχεσιακών βάσεων δεδομένων (με χρήση αφηρημένων εννοιών - abstractions), σε μορφή αντικειμένων που μπορούν να χρησιμοποιηθούν από την Python[2].

ΑΝΑΛΥΣΗ ΧΡΗΣΗΣ ΠΙΝΑΚΩΝ

- **Πίνακας Country:** Περιέχει τις 41 χώρες που στέλνουν πληροφορίες στη πλατφόρμα ENTSO-E καθώς και τη χωρική πληροφορία των συνόρων κάθε χώρας.
- **Πίνακας CountryCapacity:** Περιέχει την χωρητικότητα σε MW[3] ανά είδος παραγωγής που δύναται να παράξει μια χώρα με βάση τις εγκατεστημένες μονάδες παραγωγής. Συνδέεται με τον πίνακα Country μέσω δευτερεύοντος κλειδιού (ForeignKey) και αποθηκεύει τα δεδομένα από το dataset *Installed Generation Capacity Aggregated.*
- **Πίνακας CountryGeneration:** Πίνακας διασύνδεσης μεταξύ του πίνακα Country (με τον οποίο συνδέεται μέσω ForeignKey) και του πίνακα GenerationPerType. Περιέχει τις ημερομηνίες για τις οποίες υπάρχει πληροφορία ανά χώρα και δημιουργεί μια σχέση ManyToMany[4] με τα δεδομένα της ωριαίας παραγωγής.
    - **Πίνακας GenerationPerType:** Περιέχει την παραγωγή ανά είδος σε MW ανά ώρα της ημέρας. Αποθηκεύει τα δεδομένα από το dataset *Aggregated Generation per Type.*
- **Πίνακας CountryForecast:** Πίνακας διασύνδεσης μεταξύ του πίνακα Country (με τον οποίο συνδέεται μέσω ForeignKey) και των πινάκων GenerationForecast και WindSolarGenerationForecast. Περιέχει τις ημερομηνίες για τις οποίες υπαρχει πληροφορία ανά χώρα και δημιουργεί μια σχέση ManyToMany με τα δεδομένα των προαναφερθέντων πινάκων.
    - **Πίνακας GenerationForecast:** Περιέχει την εκτιμώμενη συνολική παραγωγή και την εκτιμώμενη συνολική κατανάλωση σε MW ανά ώρα της ημέρας.
    - **Πίνακας WindSolarGenerationForecast:** Περιέχει την εκτιμώμενη συνολική παραγωγή σε MW από ηλιακή και αιολική (επίγεια και εγκατεστημένη στη θάλασσα) ενέργεια, ανά ώρα της ημέρας.

  Αποθηκεύουν τα δεδομένα από το dataset *Aggregated Generation with Day Ahead Forecast.*

---

[1] Django Models Documentation: https://docs.djangoproject.com/en/2.2/ref/models/

[2] βλ. ενότητα ΥΛΟΠΟΙΗΣΗ BACK-END του παρόντος κεφαλαίου.

[3] Mega Watt

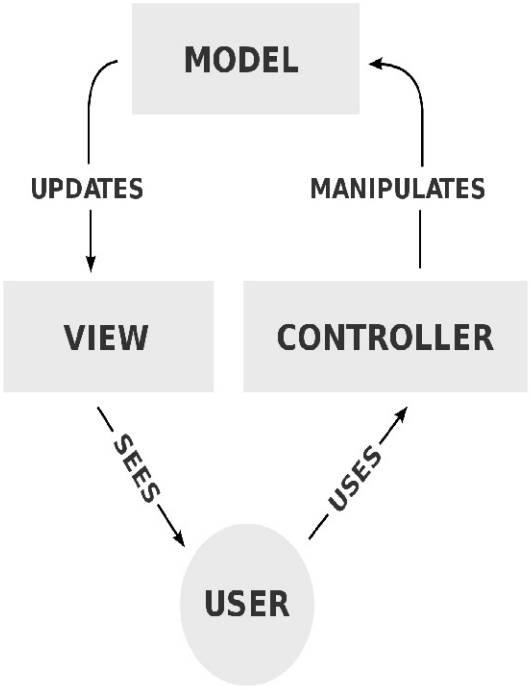
[4] Django ManyToMany Field: Δημιουργεί έναν ενδιάμεσο πίνακα ένωσης (join) για να αντιπροσωπεύσει την σχέση (n <-> n), χωρίς να πλήξει την κανονικοποιημένη μορφή της βάσης.

ΥΛΟΠΟΙΗΣΗ BACK-END

Όπως έχει αναφερθεί στα προηγούμενα κεφάλαια, για την υλοποίηση του back-end μέρους της εφαρμογής έγινε με τη χρήση του Python framework Django. Το πακέτο αυτό έχει δημιουργηθεί με σκοπό την διευκόλυνση δημιουργίας περίπλοκων διαδικτυακών εφαρμογών και λόγο του σχεδιασμού του προσφέρει αυτοματισμούς σε επίπεδο υλοποίησης.

ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Η φιλοσοφία του framework Django είναι βασισμένη στο σύγχρονο μοντέλο ανάπτυξης διαδικτυακών εφαρμογών, το Model - View - Controller (**MVC pattern**) το οποίο αποδομεί την υλοποίηση σε 3 δομικά μέρη και κάθε μέρος περαιώνει αποκλειστικά ένα σαφώς ορισμένο σύνολο εργασιών χωρίς (ιδανικά) να παρεκκλίνει από αυτό.



- **Model:** Αποτελεί μια μοντελοποίηση της δομής των δεδομένων στη βάση. Χρησιμοποιώντας αφηρημένες έννοιες επιτρέπει την αλληλεπίδραση της εφαρμογής με τα δεδομένα, αφαιρώντας την ανάγκη για συγκεκριμένο χειρισμό ανάλογα με τη σχεσιακή βάση δεδομένων που μπορεί να χρησιμοποιηθεί ανά περίπτωση (mySQL, PostgreSQL κτλ.). Τέλος, είναι υπεύθυνο για την επικοινωνία μεταξύ της διεπαφής του χρήστη με την εφαρμογή (User Interface - UI) και της βάσης δεδομένων.
- **View:** Αποτελεί τη λογική της υλοποίησης του UI. Είναι ουσιαστικά το μέρος της εφαρμογής το οποίο είναι ορατό στον χρήστη και με το οποίο μπορεί να αλληλεπιδράσει. Για κάθε μοντέλο της εφαρμογής πρέπει να έχει υλοποιηθεί το αντίστοιχο UI έτσι ώστε να είναι δυνατή η αλληλεπίδραση του χρήστη με αυτό.
- **Controller:** Αποτελεί το λογικό κέντρο της εφαρμογής και ελέγχει τη ροή των δεδομένων από το χρήστη, μέσω του UI στο μοντέλο και το ανάποδο. Δηλαδή επιστρέφει τα κατάλληλα δεδομένα από το μοντέλο ανάλογα με το πιο view έχει επιλέξει ο χρήστης και ανανεώνει τα δεδομένα του μοντέλου ανάλογα με το ποιες ενέργειες λαμβάνει αυτός προς τα δεδομένα (create, update, delete).

Αξίζει να σημειωθεί ότι στο εγχειρίδιο του Django[1] το μοντέλο αυτό αναφέρετε ως Model - Template - View με το Template να παίρνει τη θέση του View και το View να παίρνει τη θέση του Controller, διατηρώντας όμως ακριβώς την ίδια λειτουργία.

[1] Από το FAQ του Django: https://docs.djangoproject.com/en/2.2/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names

ΑΝΑΛΥΣΗ RESTFUL ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Η **Representationnal State Transfer (REST)** αρχιτεκτονική ορίζει ένα σύνολο κανόνων για τη δημιουργία διαδικτυακών εφαρμογών που επιτρέπουν την πρόσβαση και διαχείριση πόρων της εφαρμογής μέσω προκαθορισμένων επιτρεπτών και ομοιόμορφων ενεργειών ανεξαρτήτως του είδους και της μορφής του εκάστοτε πόρου. Η αναπαράσταση των πόρων καθώς και οι ενέργειες που δύναται να τελεστούν επί αυτών εντάσσονται σε αρχεία κειμένου (συνήθως σε JSON ή XML μορφή) και αποστέλλονται από και προς την εφαρμογή μέσω HTTP κλήσεων στα προκαθορισμένα URLs.

Η παρούσα εφαρμογή χρησιμοποιεί RESTful αρχιτεκτονική για να αναπαραστήσει τους πόρους της και αυτό επιτυγχάνεται με τη χρήση της επέκτασης του Django framework που ονομάζεται **Django Rest Framework** (**DRF**). Εκτός από επέκταση των αυτοματισμών που ήδη προσφέρει το Django ώστε να διευκολύνει την ανταλλαγή πόρων με αντιπροσώπευση της μορφής JSON, όπως επιτάσσει η REST αρχιτεκτονική, μετατρέπει το View μέρος του MTV μοντέλου και αναιρεί την ανάγκη ύπαρξης Template, με αποτέλεσμα να αποδίδει για κάθε πόρο της εφαρμογής το σύνολο των URL endpoints που θα χρησιμοποιηθούν από το χρήστη.

ΑΝΑΛΥΣΗ ΑΣΥΓΧΡΟΝΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Ως ασύγχρονος προγραμματισμός αναφέρεται η ύπαρξη ρουτινών που εκτελούνται ανεξάρτητα και παράλληλα της κύριας προγραμματιστικής ροής καθώς και το σύνολο των μεθόδων με τις οποίες γίνεται η διαχείριση αυτών.

Στη παρούσα εφαρμογή γίνεται χρήση ασύγχρονων ρουτινών οι οποίες ενημερώνουν τα datasets ανά τακτά χρονικά διαστήματα αλλά και κατά παραγγελία του χρήστη. Η απόφαση για τη χρήση ασύγχρονων μεθόδων λήφθηκε για να απεμπλέξει την κύρια προγραμματιστική ροή, που είναι υπεύθυνη για τη σωστή λειτουργία του συνόλου της εφαρμογής, από τις χρονοβόρες διαδικασίες της ωριαίας ανανέωσης της βάσης δεδομένων για τα datasets: Actual Generation και Generation Forecast καθώς και της, ανά τρίμηνο, ανανέωσης για το dataset Generation Capacity.

Η διαχείριση των ασύγχρονων μεθόδων είναι δυνατή μέσω της Python βιβλιοθήκης Celery η οποία χρησιμοποιεί τη μη σχεσιακή βάση δεδομένων Redis[1] σαν διαβιβαστή μηνυμάτων για να δημιουργήσει μια ουρά (queue) εργασιών. Από την ουρά αυτή ένας οι περισσότεροι (ανάλογα με τη κλίμακα της εφαρμογής) εργάτες[2] αναλαμβάνουν να εκτελέσουν μια εργασία μέχρι τη περαίωση της, να ενημερώσουν την ουρά και να εκτελέσουν την επόμενη για όσο υπάρχουν εργασίες στην ουρά. Εάν δεν υπάρχουν άλλες καταγεγραμμένες εργασίες, οι εργάτες εισέρχονται σε κατάσταση αδράνειας μέχρι να ενημερωθεί η ουρά.

Τέλος το Celery σε συνδυασμό με το django-celery-beat[3] πακέτο, προσφέρει τη δυνατότητα να ορισθούν επαναλαμβανόμενες εργασίες (πχ. να εκτελούνται κάθε ώρα, μέρα κτλ.) μέσω του πάνελ διαχείρισης του Django.

---

[1] βλ. αναφορά στη Redis της ενότητας ΣΥΝΤΟΜΗ ΠΑΡΟΥΣΙΑΣΗ ΕΡΓΑΛΕΙΩΝ->ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

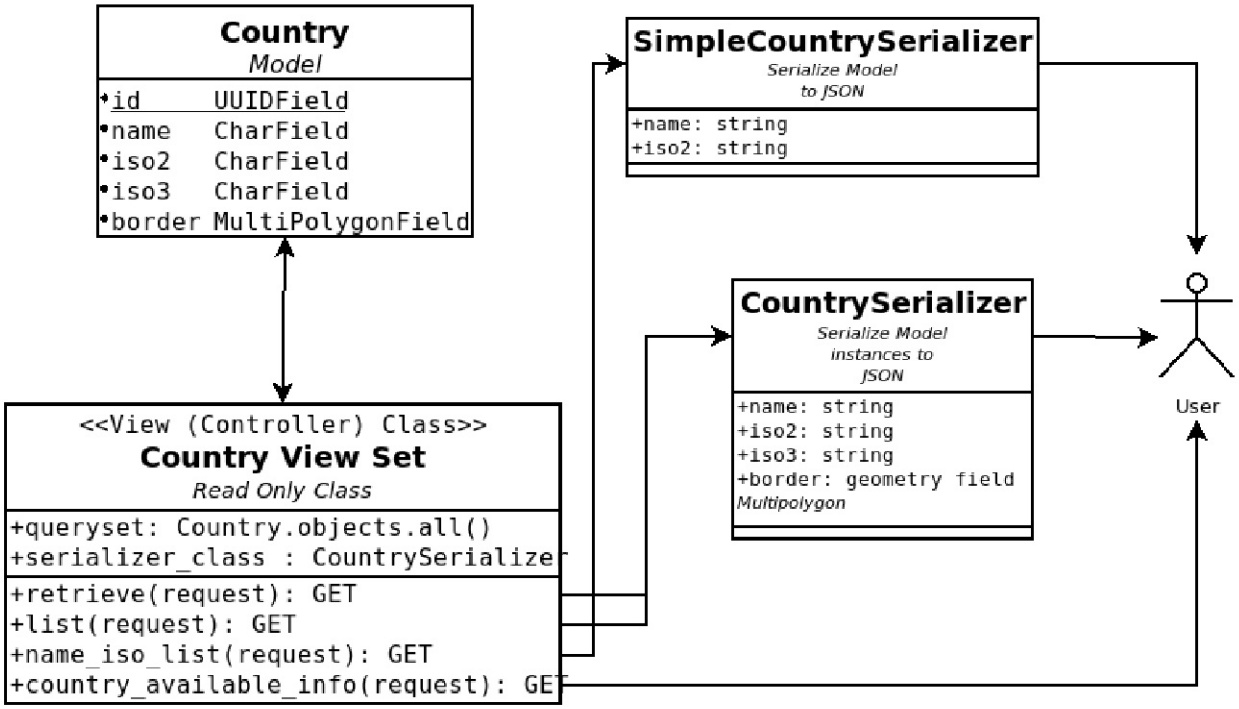
[2] Είναι ένα python script το οποίο εκτελεί μια σειρά ρουτινών ανάλογα με την εργασία που του έχει ανατεθεί.

[3] Django-celery-beat: https://pypi.org/project/django-celery-beat/

ΕΠΙΜΕΡΟΥΣ ΑΝΑΛΥΣΗ ΔΟΜΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Για τις ανάγκες της υλοποίησης και τον καταμερισμό των δεδομένων και της λογικής του χειρισμού τους κατά είδος, δημιουργήθηκαν 4 διαφορετικά modules (δομικά στοιχεία) και ένα ακόμη το οποίο αναπτύχθηκε για τις ανάγκες του front-end και θα αναλυθεί στην επόμενη ενότητα. Σε αυτή την ενότητα θα αναλυθούν διεξοδικά τα 4 modules που αφορούν τα δεδομένα και τη χρήση τους.

1. **Countries Module:** Η υπό-εφαρμογή Countries περιέχει την υλοποίηση του MTV μοντέλου για τα δεδομένα των 41 χωρών που συμμετέχουν στη πλατφόρμα ENTSO-E.



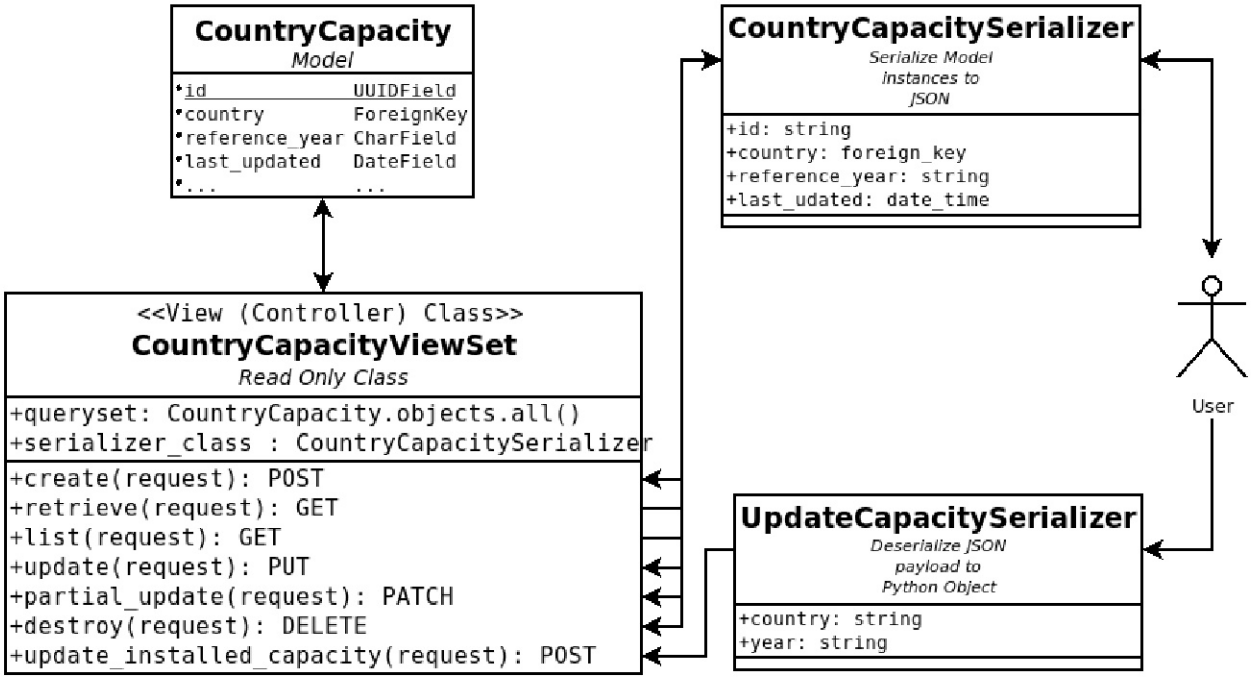
Επειδή οι χώρες είναι το μόνο είδος δεδομένων της εφαρμογής που θεωρείται ότι δεν μεταβάλλεται με τη πάροδο του χρόνου, ο Controller δεν επιτρέπει άλλη αλληλεπίδραση παρά μόνο την παρουσίαση. Είναι με άλλα λόγια Read-Only πληροφορίες αποκλειστικά.

Διαθέσιμα Endpoints:

Endpoint	Μέθοδος	Περιγραφή
api/v1/countries/	GET	Επιστρέφει μια λίστα των χωρών.
api/v1/countries/<country_id>/	GET	Επιστρέφει τη χώρα με id <country_id>
api/v1/countries/name-iso-list/	GET	Επιστρέφει μια λίστα για όλες τις χώρες, της μορφής: *[{label: name, value: iso2}, ...]*
api/v1/countries/available-info/	GET	Επιστρέφει μια ενημέρωση για τις ημερομηνίες (από - έως) για τις οποίες υπάρχουν πληροφορίες ανά dataset στη βάση

Το Countries module δεν έχει δημιουργηθεί με σκοπό να προσπελαύνει σε αυτό ο χρήστης απευθείας για να λαμβάνει πληροφορίες γύρω από τις χώρες, καθώς δεν είναι οι χώρες το σημαντικό μέρος των δεδομένων, αλλά τα υπόλοιπα datasets σε σχέση με αυτές. Ο πραγματικός σκοπός είναι να λειτουργεί ως βάση για την αποθήκευση και διάθεση των βασικών πληροφοριών που αφορούν τις χώρες για τις οποίες έχουμε διαθέσιμα δεδο-μένα.

2. **Country Capacity Module:** Η υπό-εφαρμογή Country Capacity περιέχει την υλοποίηση του MTV μοντέλου για τα δεδομένα του dataset Generation Capacity για κάθε χώρα.



Διαθέσιμα Endpoints:

Endpoint	Μέθοδος	Περιγραφή
api/v1/country-capacity/	GET	Επιστρέφει μια λίστα capacity των χωρών.
api/v1/country-capacity/	POST	Δημιουργεί μια CountryCapacity εγγραφή στη βάση με το κατάλληλο payload.
api/v1/country-capacity/<capacity_id>/	GET	Επιστρέφει το capacity με id <capacity_id>
api/v1/country-capacity/<capacity_id>/	PUT	Ανανεώνει ολόκληρη την εγγραφή του capacity με id <capacity_id>
api/v1/country-capacity/<capacity_id>/	PATCH	Ανανεώνει μερικώς την εγγραφή του capacity με id <capacity_id>
api/v1/country-capacity/<capacity_id>/	DELETE	Διαγράφει το capacity με id <capacity_id>
api/v1/country-capacity/update-installed-capacity/	POST	Κατά παραγγελία εκτέλεση του ασύγχρονου script για την ενημέρωση της βάσης.

Το endpoint *GET api/v1/country-capacity/* επιτρέπει το φιλτράρισμα των δεδομένων με βάση τη τιμή του iso2 της χώρας με την οποία συνδέεται κάθε εγγραφή, καθώς και τη τιμή

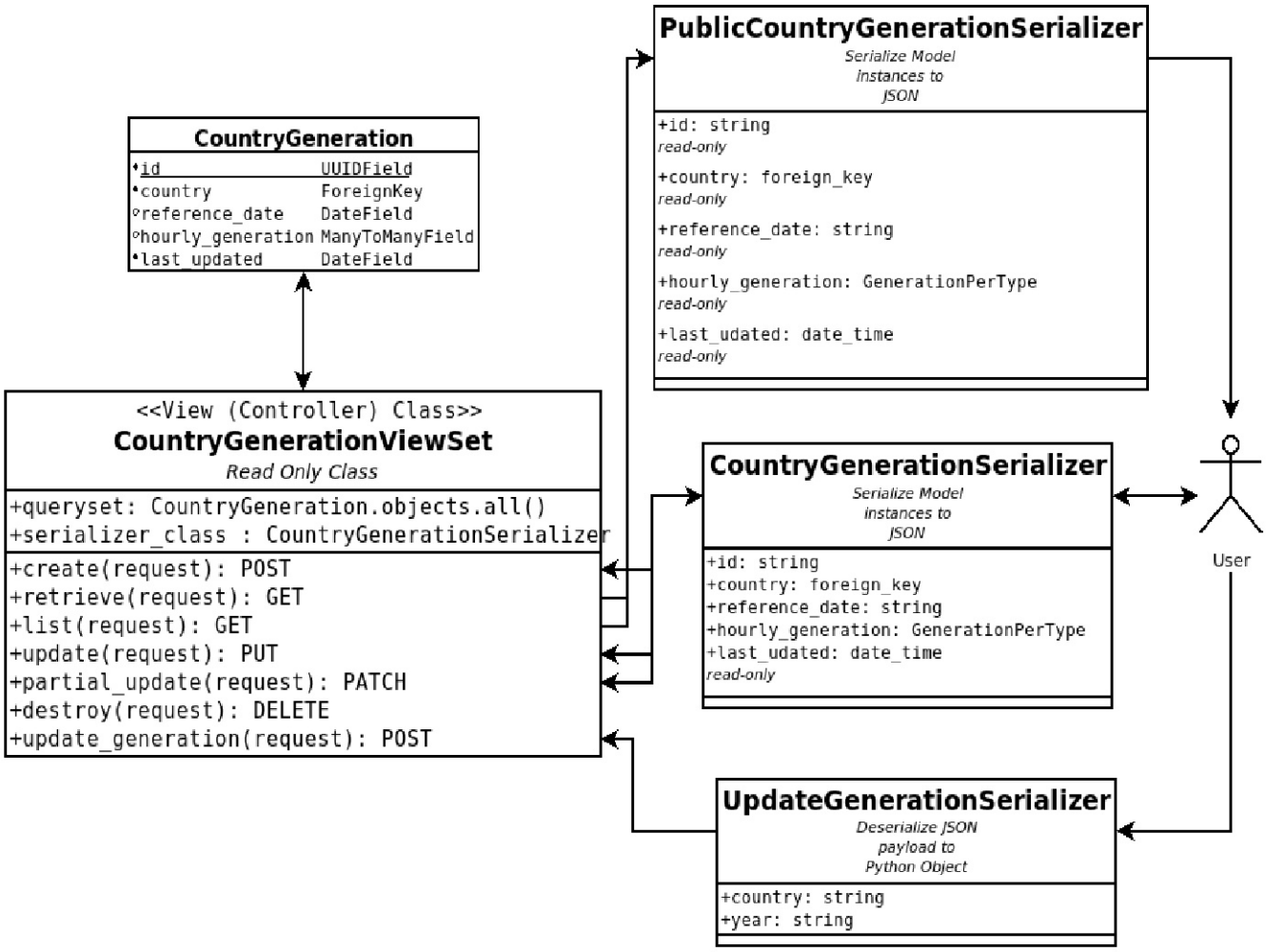
του πεδίου reference_year. Για την γεωμετρία ειδικά (το MultiPolygon) της χώρας με την οποία συνδέεται κάθε εγγραφή, υλοποιήθηκε χωρικό φίλτρο **contains** που επιτρέπει το φιλτράρισμα πληροφορίας με βάση τη χώρα της οποίας τα σύνορα περιέχουν μια ορισμένη γεωμετρία που ορίζει ο χρήστης.

Παραδείγματα φίλτρων:

- GET api/v1/country-capacity/?country_iso2=GR&reference_year=2019
- GET api/v1/country-capacity/?country_contains_geom={
  "type":"Point", "coordinates": [
  -123.26436996459961, 44.564178042345375
  ]
  }

Τέλος στο module αυτό περιέχονται τα ασύγχρονα scripts που ενημερώνουν τις εγγραφές στο μοντέλο CountryCapacity.

3. **Country Generation Module:** Η υπό-εφαρμογή Country Generation περιέχει την υλοποίηση του MTV μοντέλου για τα δεδομένα του dataset Actual Generation για κάθε χώρα.



Διαθέσιμα Endpoints:

Endpoint	Μέθοδος	Περιγραφή
api/v1/country-generation/	GET	Επιστρέφει μια λίστα με δεδομένα ωριαίας παραγωγής ανά χώρα και ανά ημερομηνία.
api/v1/country-generation/	POST	Δημιουργεί μια CountryGeneration εγγραφή στη βάση με το κατάλληλο payload.
api/v1/country-generation/<generation_id>/	GET	Επιστρέφει την εγγραφή generation με id <generation_id>
api/v1/country-generation/<generation_id>/	PUT	Ανανεώνει ολόκληρη την εγγραφή του generation με id <generation_id>
api/v1/country-generation/<generation _id>/	PATCH	Ανανεώνει μερικώς την εγγραφή του generation με id <generation _id>
api/v1/country-generation/<generation _id>/	DELETE	Διαγράφει την εγγραφή generation με id <generation _id>
api/v1/country-generation/update-generation/	POST	Κατά παραγγελία εκτέλεση του ασύγχρονου script για την ενημέρωση της βάσης.

Το endpoint *GET api/v1/country-generation/* επιτρέπει το φιλτράρισμα των δεδομένων με βάση τις τιμές που ακολουθούν:

- **country_contains_geom**: Χωρικό φίλτρο ίδιο σε λειτουργία με το φίλτρο που αναλύεται στη περιγραφή του Country Capacity Module.
- **country_iso2**: Φίλτρο με βάση τη τιμή iso2 κάθε χώρας.
- **reference_date_exact**: Φίλτρο με βάση την ακριβή ημερομηνία παραγωγής.
- **reference_date_after & reference_date_before**: Φίλτρο με βάση το εύρος ημερομηνιών παραγωγής όπως αυτό ορίζεται από το **_after** και **_before**. Σε περίπτωση που δεν οριστεί κάποιο από τα 2 άκρα του εύρους, φιλτράρει από τη πιο παλιά ημερομηνία έως τη πιο καινούργια αντίστοιχα.
- **generation_time_exact**: Φίλτρο με βάση την ακριβή ώρα παραγωγής.
- **generation_time_after & generation_time_before**: Φίλτρο με βάση το εύρος ωρών όπως αυτό ορίζεται από το **_after** και **_before**. Ισχύει για τον καθορισμό των άκρων του εύρους, ότι ισχύει και για το εύρος ημερομηνιών, με τη διαφορά ότι το εύρος ωρών ορίζεται από 00:00 έως 23:00 κάθε μέρας.

Παραδείγματα φίλτρων:

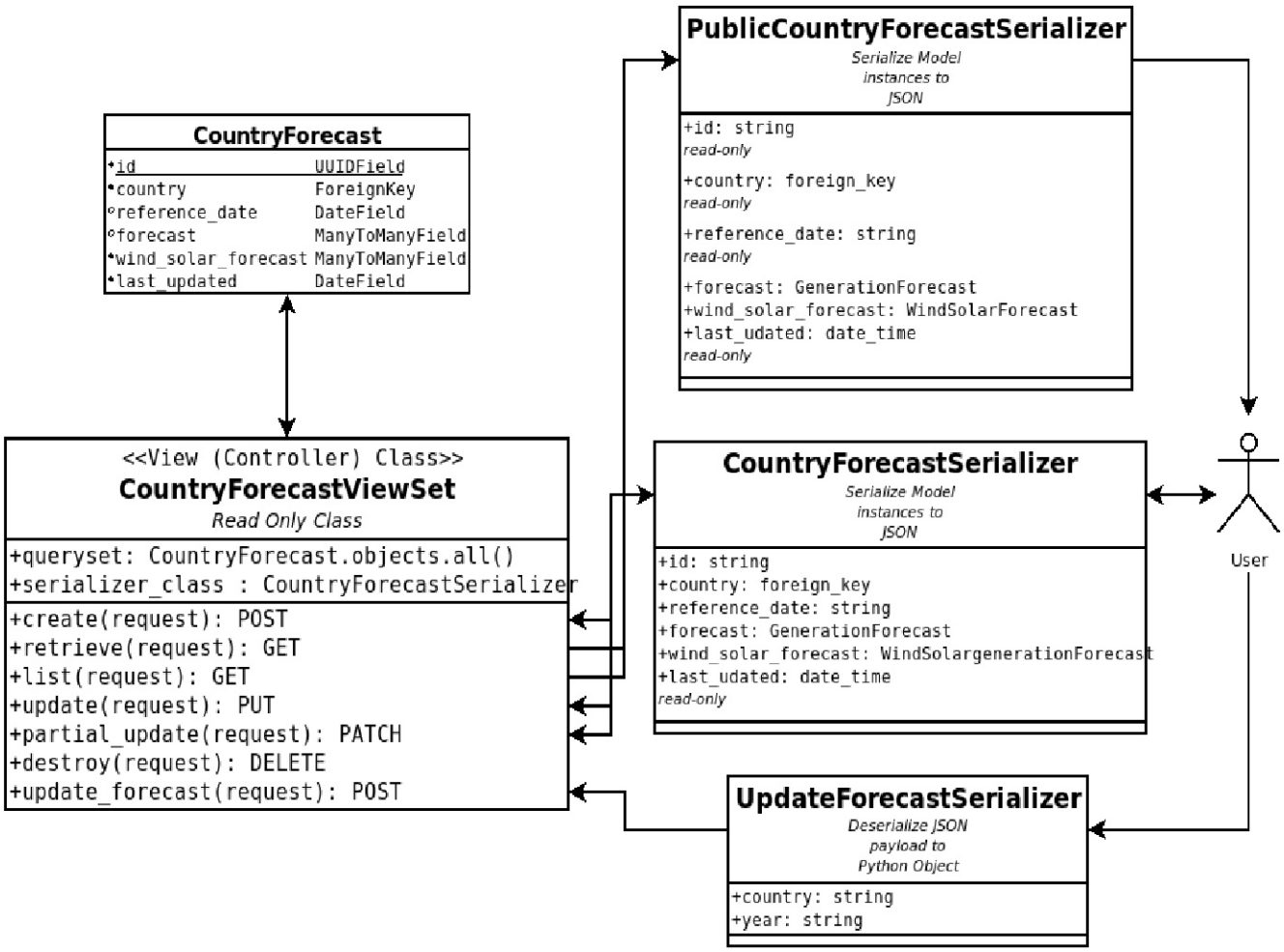
- GET api/v1/country-generation /?country_iso2=GR
  &reference_date_exact=2019-09-10

  Θα επιστρέψει την ωριαία παραγωγή της Ελλάδας για την 10/09/2019.

- GET api/v1/country-generation /?country_iso2=GR
  &generation_time_after=11:00&generation_time_before=14:00
  &reference_date_exact=2019-08-30

  Θα επιστρέψει την ωριαία παραγωγή της Ελλάδας, ανάμεσα στις 11:00 και 14:00 για την 30/08/2019.

4. **Country Forecast Module:** Η υπό-εφαρμογή Country Forecast περιέχει την υλοποίηση του MTV μοντέλου για τα δεδομένα του dataset Generation Forecast για κάθε χώρα.

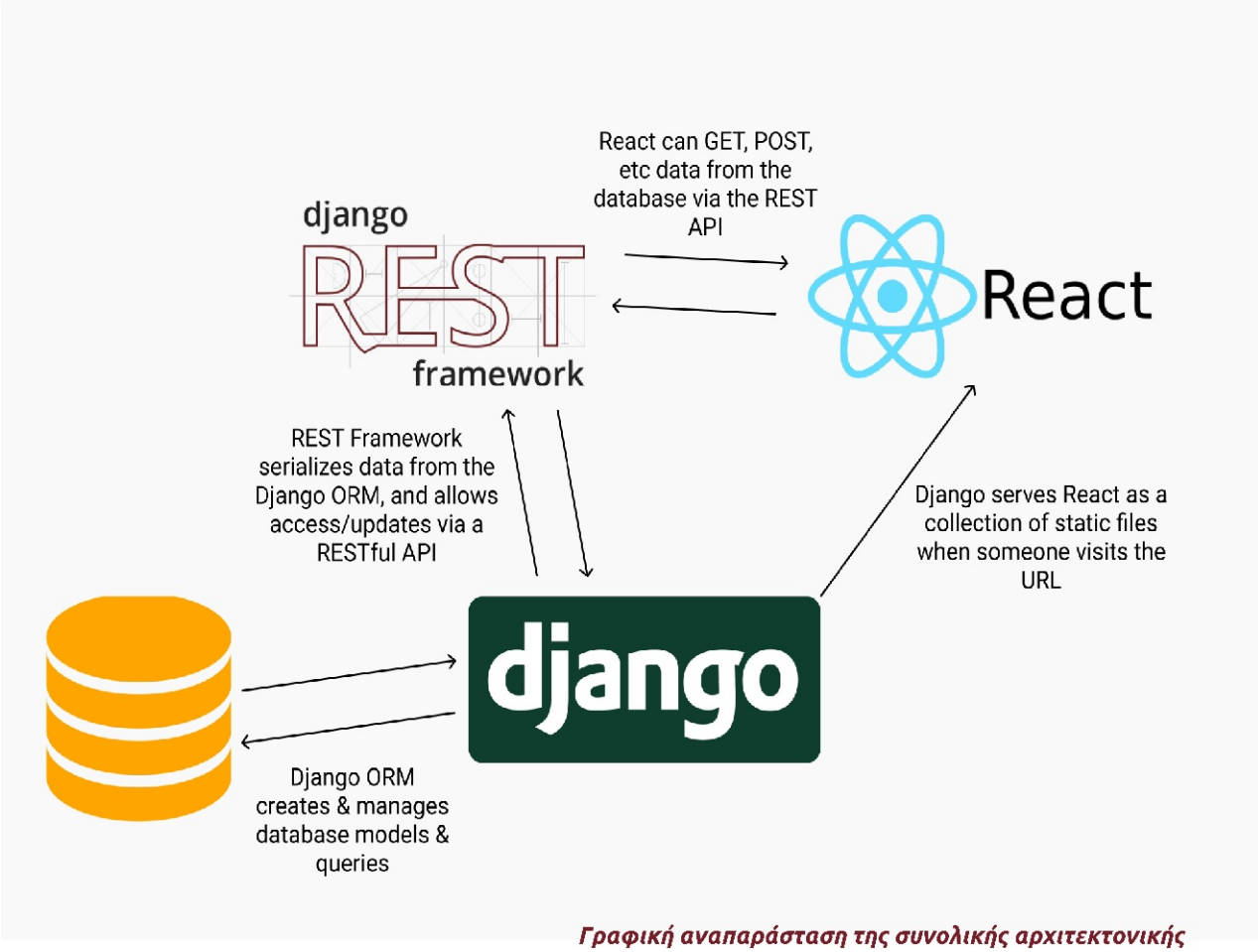


Διαθέσιμα Endpoints:

Endpoint	Μέθοδος	Περιγραφή
api/v1/country-forecast/	GET	Επιστρέφει μια λίστα με δεδομένα ωριαίας πρόγνωσης παραγωγής και κατανάλωσης ανά χώρα και ανά ημερομηνία.
api/v1/country-forecast/	POST	Δημιουργεί μια CountryForecast εγγραφή στη βάση με το κατάλληλο payload.
api/v1/country-forecast/<generation_id>/	GET	Επιστρέφει την εγγραφή forecast με id <forecast _id>
api/v1/country-forecast/<generation_id>/	PUT	Ανανεώνει ολόκληρη την εγγραφή του forecast με id <forecast _id>
api/v1/country-forecast/<generation _id>/	PATCH	Ανανεώνει μερικώς την εγγραφή του forecast με id <forecast _id>
api/v1/country-forecast/<generation _id>/	DELETE	Διαγράφει την εγγραφή forecast με id <forecast _id>
api/v1/country-forecast/ update-forecast /	POST	Κατά παραγγελία εκτέλεση του ασύγχρονου script για την ενημέρωση της βάσης.

Το endpoint *GET api/v1/country-forecast/* επιτρέπει το φιλτράρισμα των δεδομένων με βάση τις τιμές που ακολουθούν:

- **country_contains_geom**: Χωρικό φίλτρο ίδιο σε λειτουργία με το φίλτρο που αναλύεται στη περιγραφή του Country Capacity Module.
- **country_iso2**: Φίλτρο με βάση τη τιμή iso2 κάθε χώρας.
- **reference_date_exact**: Φίλτρο με βάση την ακριβή ημερομηνία παραγωγής.
- **reference_date_after & reference_date_before**: Φίλτρο με βάση το εύρος ημερομηνιών παραγωγής όπως αυτό ορίζεται από το **_after** και **_before**.
- **forecast _time_exact**: Φίλτρο με βάση την ακριβή ώρα πρόγνωσης παραγωγής.
- **forecast _time_after & forecast _time_before**: Φίλτρο με βάση το εύρος ωρών όπως αυτό ορίζεται από το **_after** και **_before**. Φιλτράρει τα ωριαία δεδομένα πρόγνωσης παραγωγής και κατανάλωσης.
- **wind_solar_time_exact**: Φίλτρο με βάση την ακριβή ώρα πρόγνωσης παραγωγής.
- **wind_solar_time_after & wind_solar_time_before**: Φίλτρο με βάση το εύρος ωρών όπως αυτό ορίζεται από το **_after** και **_before**. Φιλτράρει τα ωριαία δεδομένα πρόγνωσης παραγωγής από ηλιακή και αιολικές πηγές.



Γραφική αναπαράσταση της συνολικής αρχιτεκτονικής

ΥΛΟΠΟΙΗΣΗ FRONT-END

Η υλοποίηση λαμβάνει μέρος στο 5° module της εφαρμογής με όνομα "dashboard". Η επιλογή αυτή έγινε με σκοπό την ευκολία διασύνδεσης με το back-end και ειδικότερα για τη διευκόλυνση του "σερβιρίσματος" των επιμέρους στοιχείων που απαρτίζουν την υλοποίηση του front-end (static files, assets, resources etc.).

ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ REACT & REDUX

- **React**

  Η φιλοσοφία της ανάπτυξης εφαρμογής με χρήση React είναι βασισμένη στο διαχωρισμό των δομικών στοιχείων που απαρτίζουν τη συνολική διεπαφή του χρήστη (UI) σε μικρότερα αυτόνομα και επαναχρησιμοποιήσιμα τμήματα που ονομάζονται **components**[1]. Είναι στην ουσία τους συναρτήσεις Javascript οι οποίες λαμβάνουν ως είσοδο ένα σύνολο ορισμάτων που ονομάζονται **props**[1] και συνθέτουν (render) το οπτικό αποτέλεσμα που θα εμφανιστεί στο UI. Σε συνδυασμό με τα props, κάθε component έχει ένα ιδιωτικό και πλήρως ελεγχόμενο σύνολο μεταβλητών που ορίζει την κατάσταση του component και των περιεχομένων τους και ονομάζεται **state**[2]. Κάθε component ανασυνθέτει το οπτικό αποτέλεσμα του εαυτού του σε περίπτωση που λάβει νέα props ή αλλάξει το state του[2].
  Τα components χωρίζονται εννοιολογικά μεταξύ τους σε component-γονείς που περιέχουν component-παιδιά. Η ροή της πληροφορίας μεταξύ των 2 ειδών γίνεται μέσω των props με τις οποίες το component-γονέας μπορεί να περάσει το state του (ή μέρος του state του) καθώς και συναρτήσεις στα component-παιδιά του. Από τη μεριά των component-παιδιών, πληροφορία προς τα component-γονείς μπορεί να περάσει μόνο μέσω συναρτήσεων που περιέχονται στα props που το component-γονέας περνάει στο component-παιδί.

- **Redux**

  Το σύστημα που διέπει το state κάθε component σε μια React εφαρμογή, τείνει να γίνεται απρόβλεπτο ανάλογα με τη πολυπλοκότητα του UI καθώς και από τις επάλληλες ανασυνθέσεις (rerenders) κάθε component. Το σύστημα Redux επιτρέπει τη δημιουργία και διαχείριση ενός προβλέψιμου state το οποίο είναι κοινό ανάμεσα στα επιμέρους components. Η αρχιτεκτονική του Redux μπορεί να αναλυθεί σε 3 διακριτά στοιχεία[3]:

  - **State:** Ένα σύνολο από Read-Only μεταβλητές χρήσιμες για την επιμέρους εκτέλεση της σύνθεσης συγκεκριμένων components.
  - **Actions:** Ένα σύνολο συναρτήσεων-ενεργειών που εκτελούν την μετάλλαξη του state. Το state επιτρέπεται να μεταλλαχθεί **μόνο** μέσω των δηλωμένων actions.
  - **Reducers:** Λαμβάνουν ως είσοδο το προηγούμενο state και ένα action και δημιουργούν το νέο state ανάλογα με το action που επικαλέστηκε την αλλαγή.

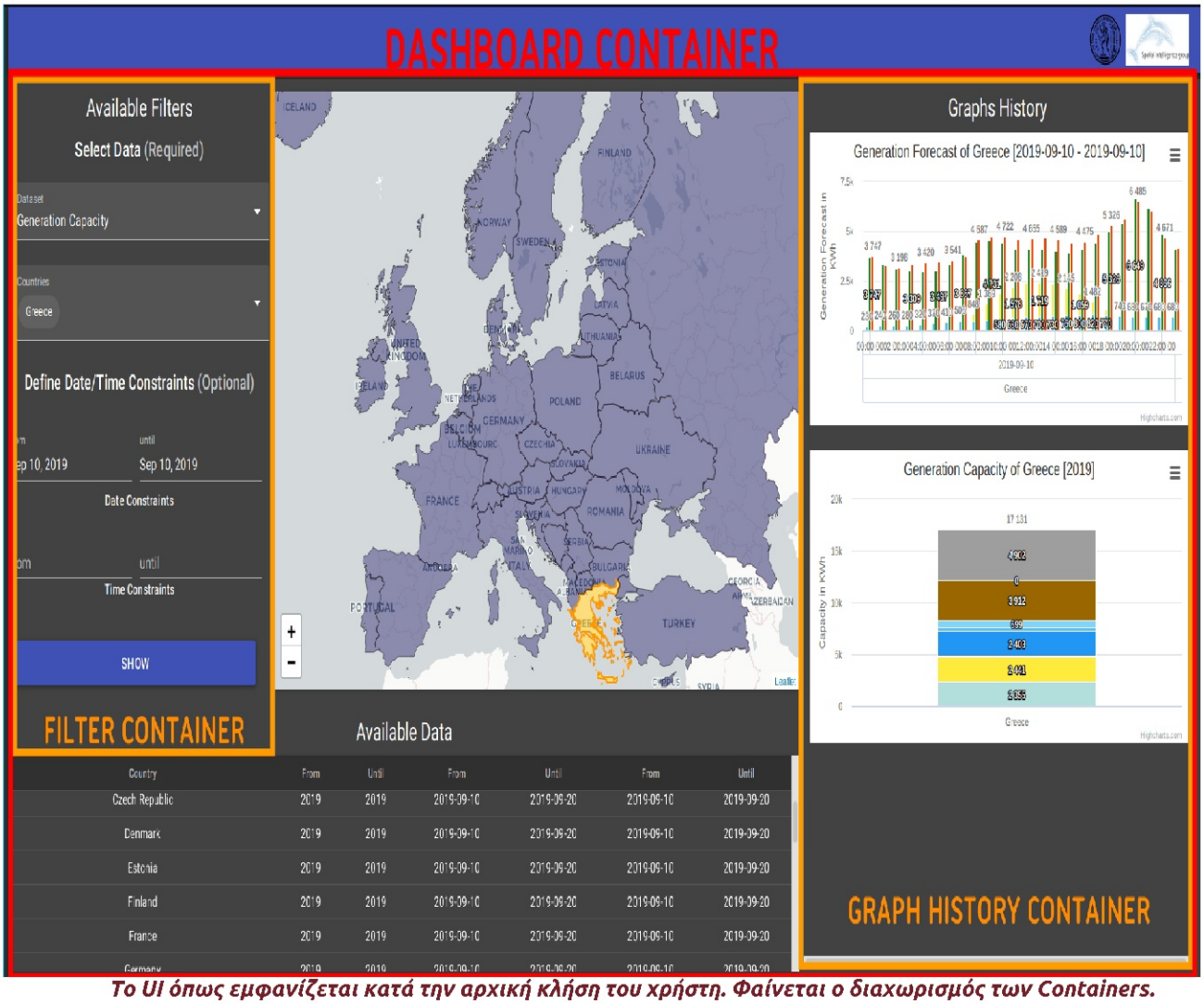
---

[1] Components & Props: https://reactjs.org/docs/components-and-props.html
[2] State & Lifecycle: https://reactjs.org/docs/state-and-lifecycle.html
[3] Αρχές του Redux: https://redux.js.org/introduction/three-principles#three-principles

ΕΠΙΜΕΡΟΥΣ ΑΝΑΛΥΣΗ ΔΟΜΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΤΟΥ UI

Για τις ανάγκες του καταμερισμού των δομικών στοιχείων του UI κατά την αρχιτεκτονική του React & Redux framework, η υλοποίηση χωρίστηκε σε 4 τμήματα. Το πρώτο τμήμα περιέχει την υλοποίηση του Redux Store και τα υπόλοιπα 3 τμήματα χωρίζονται εννοιολογικά σε 3 Component-Containers[1] τα οποία λειτουργούν και σαν "γονείς" για τα components που περιέχουν.

1. **Redux Store:** Περιέχει τα δομικά στοιχεία που στοιχειοθετούν την αρχιτεκτονική του Redux (state, actions, reducers). Στη παρούσα εφαρμογή το state διατηρεί την πληροφορία για το ποιες χώρες έχουν επιλεχθεί μέσω των φίλτρων και του χάρτη έτσι ώστε να επιτρέπει την ταυτόχρονη ανασύνθεση των αντίστοιχων containers (του χάρτη και του Countries Filter) όταν μια χώρα επιλέγεται ή αποεπιλέγεται από το χρήστη.



Το UI όπως εμφανίζεται κατά την αρχική κλήση του χρήστη. Φαίνεται ο διαχωρισμός των Containers.

[1] React Container Αρχιτεκτονική: https://reactpatterns.com/#container-component

2. **Dashboard Container:** Αποτελεί το κεντρικό Container-component της εφαρμογής. Περιέχει το σύνολο των containers που απαρτίζουν το UI και είναι υπεύθυνο για τη ροή των δεδομένων από το back-end προς αυτά, μέσω των παρακάτω διεργασιών:
    - Με την αρχική κλήση του χρήστη προς το UI, εκτελεί ασύγχρονα τις κατάλληλες HTTP κλήσεις προς το back-end για να φορτώσει τα δεδομένα που θα συντεθούν ως επίπεδα στον χάρτη (συγκεκριμένα τα σύνορα των χωρών) καθώς και τα δεδομένα που συμπληρώνουν το πίνακα *Available Info.*
    - Διατηρεί τη κατάσταση των επιλεγμένων χωρών και φίλτρων μέσω συναρτήσεων που τροφοδοτεί στα container-παιδιά μέσω των props.
    - Συνθέτει τα κατάλληλα URL με βάση τις επιλογές του χρήστη και εκτελεί τις αντίστοιχες κλήσεις προς το back-end.
    - Συλλέγει τα δεδομένα όπως επιστρέφουν από την επιτυχή ολοκλήρωση του προηγούμενου βήματος.
    - Συνθέτει τις επιλογές της σύνθεσης των κατάλληλων γραφημάτων και ενημερώνει τα components που θα παράξουν τις γραφικές αναπαραστάσεις των δεδομένων σε γραφήματα (Graph-Components[1]).
    - Περιέχει τη λειτουργικότητα που επιτρέπει στο χρήστη να αποθηκεύει γραφήματα στο Graph History Component και την τροφοδοτεί στα Graph-Components μέσω των props τους.
    - Διατηρεί το state των σωζόμενων γραφημάτων που αποθηκεύονται στο Graph History Component.
3. **Filters Container:** Αποτελεί το κύριο σημείο αλληλεπίδρασης του χρήστη με την εφαρμογή καθώς μέσω των components που ορίζονται εντός του Container μπορεί να επιλέξει μια σειρά από φίλτρα για τα δεδομένα που θα συλλεχθούν και θα προβληθούν. Το Container σε συνδυασμό με τα components που περιλαμβάνει, εκτελεί τις παρακάτω διεργασίες:
    - Με την αρχική κλήση του χρήστη προς το UI, εκτελεί ασύγχρονα τη λήψη των δεδομένων που θα αποτελέσουν το σύνολο των επιλογών για το Countries Filter component.
    - Τροφοδοτεί κάθε component-παιδί με κατάλληλες συναρτήσεις μέσω των οποίων θα ενημερώσουν το state του Dashboard Container με βάση τις επιλογές του χρήστη.
    - Αναλαμβάνει την ενημέρωση του state του Dashboard Container όταν ο χρήστης επιλέξει την τελική καταχώρηση των φίλτρων που επιλέγει (κουμπί "SHOW").
    - Τα επιμέρους component δίνουν τη δυνατότητα επιλογής του dataset και των επιλογών με τις οποίες θα φιλτραριστούν τα δεδομένα πριν συντεθούν σε γράφημα. Οι υπάρχουσες επιλογές επιτρέπουν το φιλτράρισμα με βάση τις χώρες, τις ημερομηνίες και τις ώρες της ημέρας για τις οποίες θα συλλεχθούν δεδομένα.

[1] βλ. ενότητα Graph Components του παρόντος κεφαλαίου.

- Τέλος, χειρίζεται τυχόν λανθασμένες επιλογές του χρήστη και αποτρέπει την λανθασμένη ανανέωση του state. Λάθος επιλογές είναι η μη επιλογή dataset ή η μη επιλογή τουλάχιστον μιας χώρας.

4. **Graph History Container:** Αποτελεί το σημείο στο οποίο αποθηκεύονται ήδη συντεθειμένα γραφήματα κατ' επιλογή του χρήστη. Επιτρέπει στο χρήστη να αποθηκεύει μέχρι 3 γραφήματα και σε περίπτωση που ξεπεράσει τον αριθμό, το component αυτόματα αντικαθιστά το πιο παλιό γράφημα με το νεότερο. Το component εκτελεί τις παρακάτω διεργασίες:
   - Επιτρέπει στο χρήστη να επιλέξει με click ένα από τα σωζόμενα γραφήματα και να το ανάσυνθέσει σε Graph-component.
   - Διατηρεί στο state την τελευταία επιλογή σωζόμενου γραφήματος του χρήστη.

Αξίζει να σημειωθεί ότι τα Containers που αναφέρθηκαν περιέχουν τις πληροφορίες μορφοποίησης (style) του εαυτού τους και κάθε component που περιέχουν.

Graph Components: Είναι components υπεύθυνα για την μετατροπή των δεδομένων από JSON σε Javascript αντικείμενα και για την τελική σύνθεση τους σε γραφήματα ανάλογα με το είδος του dataset. Τα γραφήματα παρουσιάζονται εντός ένος παραθύρου τύπου modal[1] και τα components επιτρέπουν στο χρήστη να αποθηκεύσει (κουμπί "KEEP") ένα γράφημα ή να κλείσει το modal.

[1] Παράθυρο τύπου modal: https://en.wikipedia.org/wiki/Modal_window

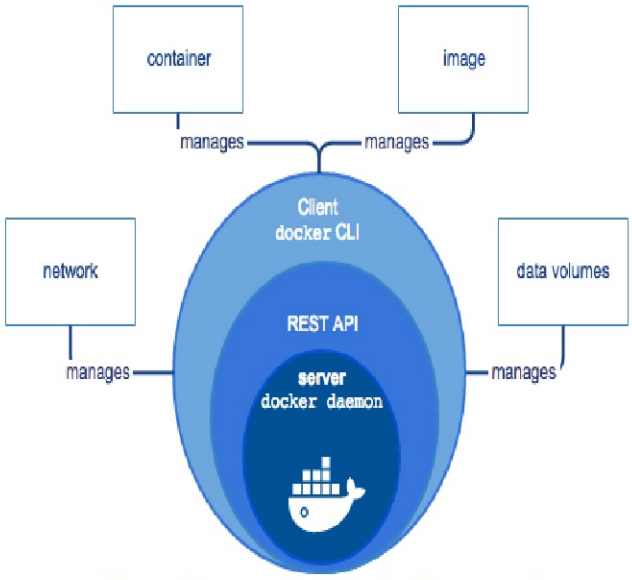
ΥΛΟΠΟΙΗΣΗ ΥΠΟΔΟΜΗΣ

Η υλοποίηση της υποδομής γίνεται με χρήση του συστήματος Docker, το οποίο επιτρέπει τη "συσκευασία" και εκτέλεση εφαρμογών σε μια μορφή απομονωμένου περιβάλλοντος που ονομάζεται container. Τα containers δεν καταναλώνουν πολλούς πόρους συστήματος και αναπτύσσονται κατευθείαν στον πυρήνα του λειτουργικού συστήματος του μηχανήματος στο οποίο αναπτύσσονται, κάνοντας έτσι δυνατή την ταυτόχρονη ανάπτυξη μεγάλου πλήθους containers χωρίς να μειώνουν την απόδοση του μηχανήματος.

ΑΝΑΛΥΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΟΥ DOCKER

Το Docker χρησιμοποιεί το μοντέλο της client-server[1] αρχιτεκτονικής κατά την οποίο η εφαρμογή docker client επιτρέπει την επικοινωνία του χρήστη (ή του συστήματος) με την εφαρμογή docker daemon[2] που διαχειρίζεται στο παρασκήνιο τα δομικά στοιχεία που απαρτίζουν τη Docker αρχιτεκτονική:

- **Images:** Είναι ένα αρχείο που αποτελείται από πολλαπλά επίπεδα πληροφορίας και εντολών, τα οποία χρησιμοποιούνται για την εκτέλεση κώδικα εντός ενός Docker container.
- **Containers:** Αποτελούν το δομικό στοιχείο του συστήματος Docker. Είναι κανονικοποιημένες μονάδες λογισμικού οι οποίες αν αναπτυχθούν με τον ίδιο τρόπο σε οποιοδήποτε μηχάνημα, θα αποδώσει ακριβώς το ίδιο αποτέλεσμα. Μια εφαρμογή ανεπτυγμένη σε container ονομάζεται και docker instance της εφαρμογής.



πηγή: https://vmarena.com/dockrchitecture-and-components/

- **Networks:** Είναι εικονικά δίκτυα με τα οποία συνδέονται τα containers και τους επιτρέπουν να επικοινωνούν μεταξύ τους αλλά και με εξωτερικούς παράγοντες, μέσω σαφώς ορισμένων κανόνων που με κατάλληλη χρήση τους διασφαλίζουν την ασφάλεια των docker instances από εξωγενείς κακόβουλους παράγοντες.
- **Volumes:** Τα δεδομένα κάθε docker instance δεν διατηρούνται σε περίπτωση επανεκκίνησης του container. Οι τόμοι (volumes) δεδομένων επιτρέπουν την διατήρηση δεδομένων εκτός του container τα οποία δεν χάνονται αν δεν διαγραφεί ο τόμος που τα περιέχει. Με αυτό το τρόπο είναι δυνατή η διατήρηση σημαντικών δεδομένων (πχ. εγγραφές σε μια βάση δεδομένων ανεπτυγμένη μέσα σε container) μεταξύ διαφορετικών docker instances της ίδιας εφαρμογής.

---

[1] Client-Server Αρχιτεκτονική: https://en.wikipedia.org/wiki/Client%E2%80%93server_model
[2] Daemon: https://en.wikipedia.org/wiki/Daemon_(computing)

ΑΝΑΛΥΣΗ ΔΙΑΔΙΚΑΣΙΑΣ DOCKERIZATION ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Η συνήθης πρακτική για την υλοποίηση της υποδομής είναι να αναπτύσσεται μια εικόνα βάσης πάνω στην οποία χτίζονται οι προϋποθέσεις για την πλήρη λειτουργία της εφαρμογής. Για τις ανάγκες της παρούσας υλοποίησης πρέπει πρώτα να χτιστεί ο κώδικας που απαρτίζει το front-end και σε δεύτερο χρόνο να χτιστεί ο κώδικας του back-end μέσω του οποίο εκκινείται και ο server της εφαρμογής. Αυτή η διαδικασία ορίζεται ως multi-stage build (χτίσιμο πολλαπλών σταδίων)[1] και επιτρέπει τον καταμερισμό της διαδικασίας σε στάδια τα οποία εκτελούνται σειριακά. Από όλα τα στάδια της εκτέλεσης το τελευταίο θα είναι το ενεργό και αυτό που θα καθορίσει την ανάπτυξη του container.

Το Dockerfile (Dockerfile.production) με το οποίο χτίζεται το container της εφαρμογής είναι το εξής:

```
1   FROM node:12.2.0-alpine AS build
2   MAINTAINER John Moutafis <jg2moutafis@gmail.com>
3
4   RUN mkdir -p /usr/src/app
5   WORKDIR /usr/src/app
6
7   COPY .babelrc .
8   COPY package.json .
9   COPY package-lock.json .
10  COPY webpack.base.config.js .
11  COPY webpack.prod.config.js .
12
13  RUN npm install
14  COPY ./modules/dashboard/static/ ./modules/dashboard/static/
15  RUN npm run build-production
16
17  FROM intelligems/geodjango:0.1 AS production
18  MAINTAINER John Moutafis <jg2moutafis@gmail.com>
19
20  ENV PYTHONUNBUFFERED=1
21
22  COPY requirements.txt /usr/src/app/
23  RUN pip install -r requirements.txt
24  ENV GDAL_SKIP=DODS
25
26  COPY ./ /usr/src/app/
27  COPY --from=build /usr/src/app/staticfiles/assets/dist ./staticfiles/assets/dist
28  COPY --from=build /usr/src/app/webpack-stats-prod.json ./webpack-stats-prod.json
29
30  EXPOSE 8000
31
32  CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

[1] Multi-Stage Build: https://docs.docker.com/develop/develop-images/multistage-build/

- Το πρώτο στάδιο (γραμμές 1 έως 15) αναλαμβάνει το χτίσιμο του front-end κώδικα σαν στατικό αρχείο το οποίο τοποθετείται στο τελικό στάδιο (συγκεκριμένα στις γραμμές 27 και 28) για να μπορεί να διαμοιραστεί μέσω του server.
- Το δεύτερο στάδιο (γραμμές 16 έως 32) αναλαμβάνει το χτίσιμο του back-end κώδικα λαμβάνει τη τελική μορφή του front-end όπως χτίστηκε από το πρώτο στάδιο και εκκινεί τον Django Server.

Ό,τι εγκαταστάθηκε κατά το πρώτο στάδιο απεγκαθίσταται κατά την εκκίνηση της ανάπτυξης του τελικού σταδίου.

ΑΝΑΛΥΣΗ ΑΝΑΠΤΥΞΗΣ ΣΥΝΟΛΙΚΗΣ ΥΠΟΔΟΜΗΣ

Όπως έχει προαναφερθεί, για την ανάπτυξη της πλήρους λειτουργικότητας της εφαρμογής είναι αναγκαία η ύπαρξη των παρακάτω στο μηχάνημα στο οποίο θα αναπτυχθεί ο server:

- Βάση δεδομένων (PostgreSQL με Postgis)
- Message Broker (Redis server)
- Το container της εφαρμογής (ανεπτυγμένο με το Dockerfile.production της προηγούμενης ενότητας)
- Celery container για την εκτέλεση των ασύγχρονων διεργασιών της εφαρμογής.
- Celery Beat container για την εκτέλεση των χρονικά προγραμματισμένων διεργασιών της εφαρμογής.

Με γνώμονα την αρχιτεκτονική του Docker τα παραπάνω πρέπει να αναπτυχθούν σε ξεχωριστά και αυτόνομα containers τα οποία θα επικοινωνούν μεταξύ τους στο ίδιο δίκτυο.

Η υλοποίηση ενός τέτοιου συνόλου από συνεργαζόμενα containers ορίζεται ως multi-part Docker application και επιτυγχάνεται με τη χρήση του εργαλείου docker-compose[1].

Το εργαλείο αυτό επιτρέπει τον ορισμό των επιμέρους δομικών στοιχείων για την ανάπτυξη μιας πολυεπίπεδης Docker εφαρμογής μέσω ενός δομημένου αρχείου τύπου yaml το οποίο ονομάζεται docker-compose.yml.

Στο αρχείο αυτό δηλώνεται το πλήρες "οικοσύστημα" της εφαρμογής το οποίο αποτελείται από το σύνολο των δομικών στοιχείων, των δικτύων και των τόμων που θα χρησιμοποιηθούν από κάθε container.

Ακολουθεί το docker-compose.yml της παρούσης εφαρμογής και ανάλυση του:

[1] docker-compose: https://docs.docker.com/compose/

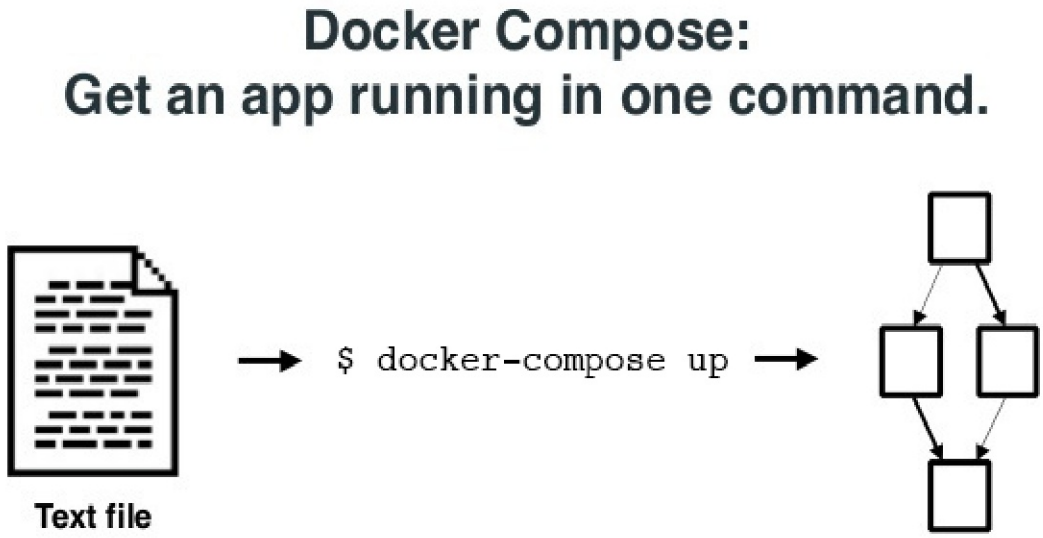
```yaml
version: '3'

services:

  postgis:
    image: kartoza/postgis:9.6-2.4
    volumes:
      - power_map_data:/var/lib/postgresql
    env_file:
      - .env
    environment:
      ALLOW_IP_RANGE: 0.0.0.0/0

  redis:
    image: redis:4

  web:
    build:
      context: .
      dockerfile: Dockerfile.production
    image: power_map
    env_file:
      - .env
    ports:
      - 80:8000
    depends_on:
      - postgis
      - redis

  worker:
    image: power_map
    command: celery -A power_map worker -l info
    env_file:
      - .env
    depends_on:
      - postgis
      - redis
      - web

  beat:
    image: power_map
    command: celery -A power_map beat -l info --scheduler django_celery_beat.schedulers:DatabaseScheduler
    env_file:
      - .env
    depends_on:
      - postgis
      - redis
      - web
      - worker

volumes:
  power_map_data: {}
```

Σύντομη ανάλυση docker-compose.yml

Τα δομικά στοιχεία που ορίζονται είναι τα εξής:

- **postgis** (γραμμές 5-12)**:** Χτίζει το container της βάσης δεδομένων με βάση την εικόνα (docker image) kartoza/postgis:9.6-2.4[1], το οποίο συνδέεται με τον τόμο (volume) power_map_data για τη διατήρηση των δεδομένων της βάσης.
- **redis** (γραμμές 14-15)**:** Χτίζει το container του Message Broker με βάση την εικόνα redis:4[2]
- **web** (γραμμές 17-28)**:** Χτίζει το container της εφαρμογής με βάση το περιβάλλον που ορίζεται από το αρχείο Dockerfile.production που προαναφέρθηκε. Δεσμεύει τη θύρα 80 του συστήματος με την θύρα 8000 του Django server με αποτέλεσμα όσες κλήσεις εκτελούνται προς τον server στη θύρα 80 να αναδρομολογούνται στον Django server που θα τις διαχειριστεί.
- **worker** (γραμμές 30-38)**:** Χτίζει το container του Celery worker με βάση το περιβάλλον που ορίζεται από το αρχείο Dockerfile.production.
- **beat** (γραμμές 40-49)**:** Χτίζει το container του Celery beat με βάση το περιβάλλον που ορίζεται από το αρχείο Dockerfile.production.

Όλα τα δομικά στοιχεία επικοινωνούν με το default εικονικό δίκτυο που δημιουργείται αυτόματα κατά την εκτέλεση του docker-compose.

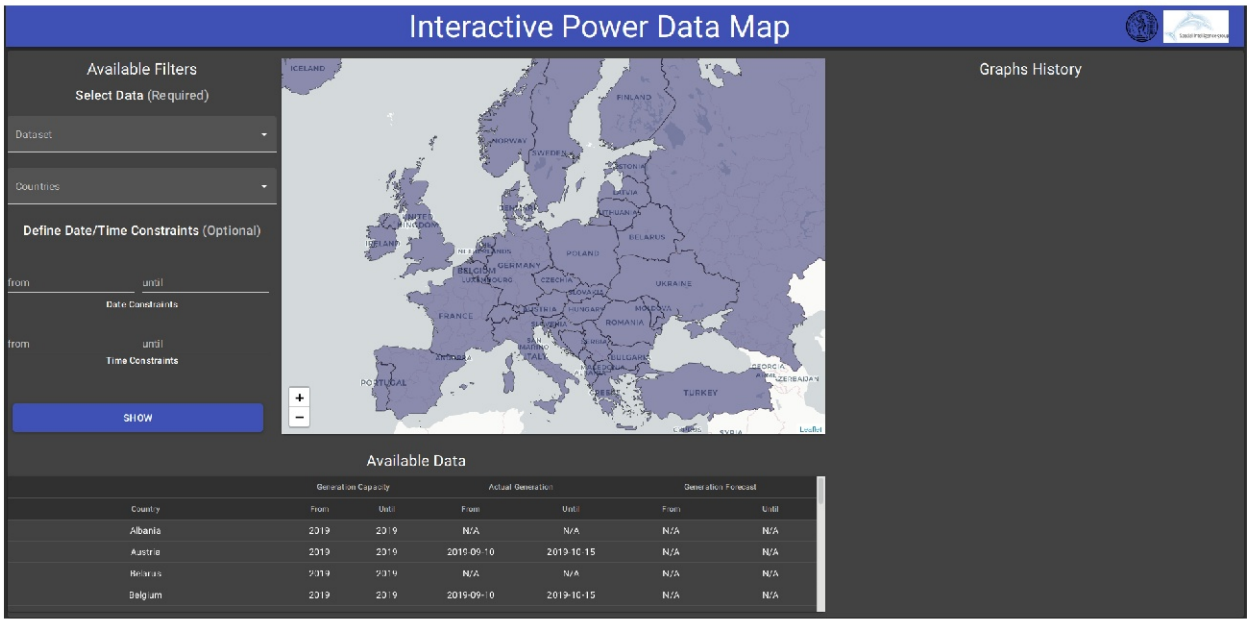


[1] Εικόνα kartoza/postgis: https://hub.docker.com/r/kartoza/postgis/
[2] Εικόνα redis: https://hub.docker.com/_/redis

ΠΑΡΑΔΕΙΓΜΑ ΧΡΗΣΗΣ

Αφού η εφαρμογή αναπτυχθεί στον server[1] είναι έτοιμη προς χρήση. Έστω ότι ένας χρήστης θέλει να συγκρίνει δεδομένα ωριαίας παραγωγής ενέργειας για τις χώρες: Ελβετία και Ελλάδα της ημέρας 15/10/2019, μεταξύ των ωρών 09:00 και 15:00. Ακόμη θέλει να συγκρίνει τις δυνατότητες παραγωγής των εγκαταστάσεων ανά είδος αυτών των χωρών. Τέλος ο χρήστης θέλει να αποθηκεύσει τα γραφήματα που θα παραχθούν.

Η αρχική οθόνη στην οποία εισέρχεται ο χρήστης είναι το dashboard:



1. Στη κεντρική οθόνη της εφαρμογής (dashboard) εμφανίζεται ο πίνακας "Available Data", ο οποίος και περιέχει πληροφορία για τα συλλεγμένα δεδομένα ανά χώρα:

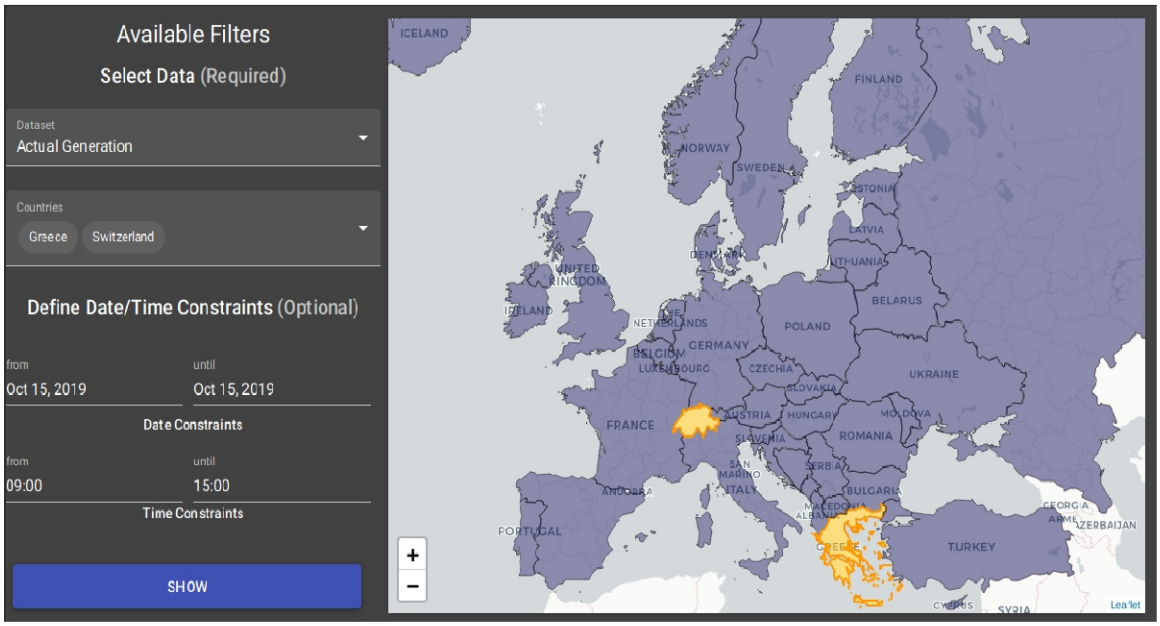
Available Data

	Generation Capacity		Actual Generation		Generation Forecast	
Country	From	Until	From	Until	From	Until
Albania	2019	2019	N/A	N/A	N/A	N/A
Austria	2019	2019	2019-09-10	2019-10-15	N/A	N/A
Belarus	2019	2019	N/A	N/A	N/A	N/A
Belgium	2019	2019	2019-09-10	2019-10-15	N/A	N/A

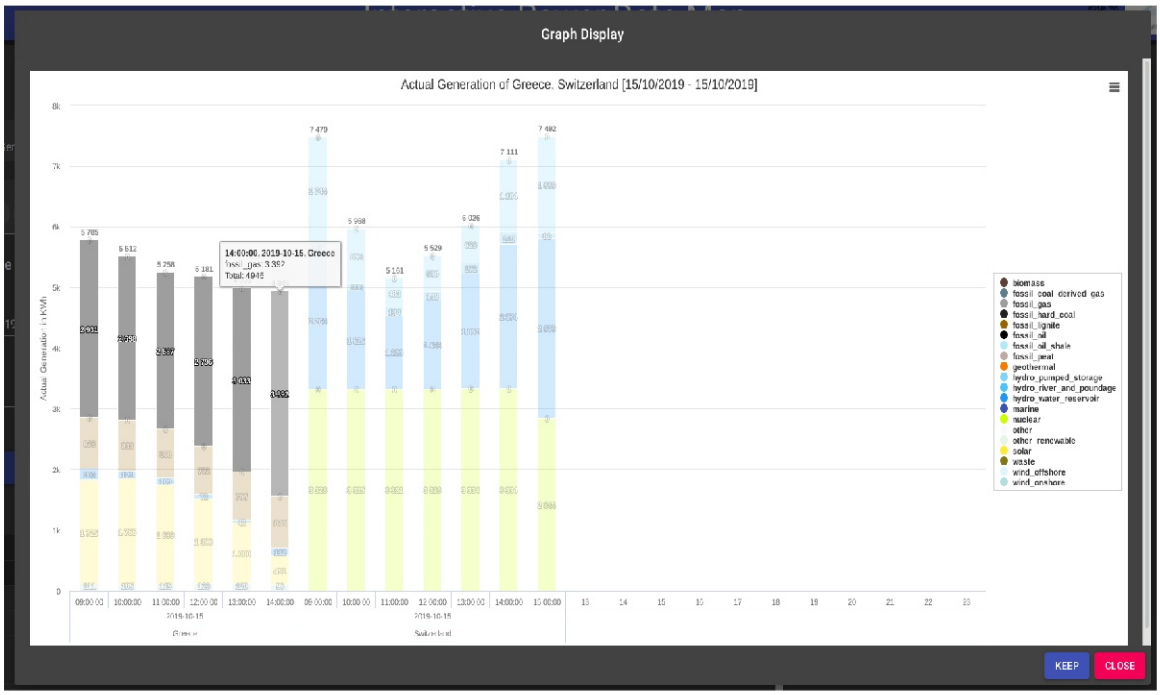
Η πληροφορία του πίνακα αφορά τις χρονικές περιόδους για τις οποίες υπάρχουν δεδομένα ανά dataset.

[1] Οδηγίες εγκατάστασης: βλ. ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ ΣΕ SERVER του παραρτήματος Α'

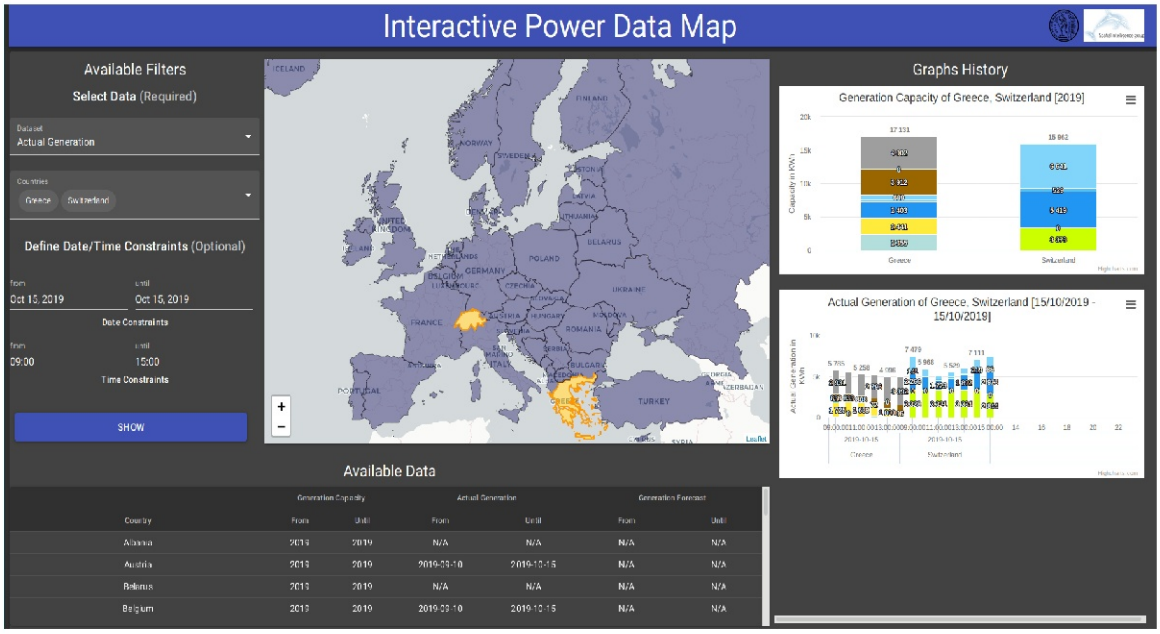
2. Η κεντρική οθόνη της εφαρμογής (dashboard) επιτρέπει την επιλογή των κατάλληλων φίλτρων στο χρήστη για να δημιουργήσει των ερωτημάτων του:



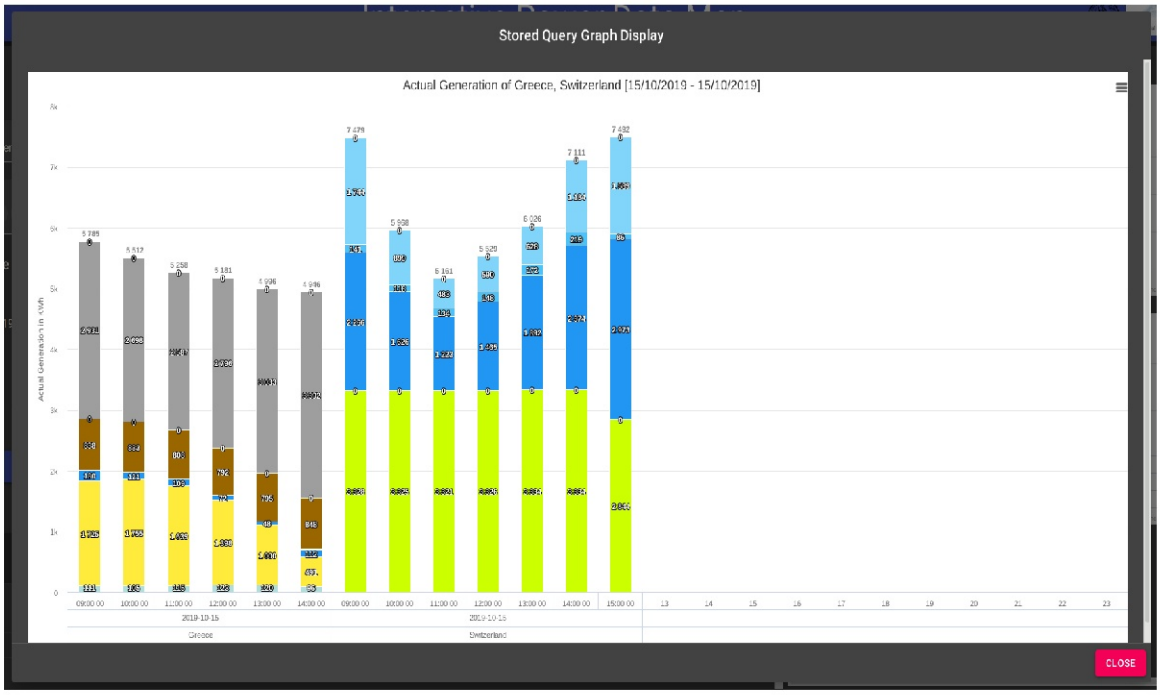
3. Αφού ο χρήστης επιλέξει τα φίλτρα, κάνοντας κλικ στο κουμπί “SHOW” δίνει την εντολή της δημιουργίας του αντίστοιχου διαγράμματος:



4. Από αυτή την οθόνη τύπου modal, επιλέγοντας το κουμπί “KEEP” σώζει το διάγραμμα. Έστω ότι έχει σώσει και τα δύο γραφήματα που τον ενδιαφέρουν. Τότε το αποτέλεσμα είναι:



5. Τέλος, ο χρήστης μπορεί να προσπελάσει τα αποθηκευμένα γραφήματα, κάνοντας κλικ πάνω στην επιφάνεια τους:



ΕΠΙΛΟΓΟΣ

Η υλοποίηση της εφαρμογής επιτυγχάνει τη χωρική σύνδεση των δεδομένων της πλατφόρμας διαφάνειας ENTSO-E, που αφορούν την ελεύθερη αγορά και παραγωγή ηλεκτρικής ενέργειας στην Ευρωπαϊκή Ένωση. Επιπλέον υλοποιεί τη δυνατότητα άμεσης σύγκρισης πληροφορίας από τις διαφορετικές χώρες που συμμετέχουν στο πρόγραμμα.
Ο συνδυασμός των δυνατοτήτων που παρέχει η εφαρμογή επί των δεδομένων ενισχύουν την αξία τους στο τομέα της λήψης αποφάσεων για ζητήματα αγοράς και πώλησης ενέργειας. Επίσης θέτουν τις βάσεις για τον καθοδηγούμενο από δεδομένα (data-driven) καθορισμό των τιμών της μεγαβατώρας ανά παραγωγικές ζώνες, μέσω της πρόγνωσης των αναγκών και της, σε πραγματικό χρόνο, οπτικοποίησης της ανά χώρα παραγωγής. Σε συνθήκες ελεύθερης αγοράς, ο ακριβής και άμεσος ορισμός των αναγκών και του κόστους που χρειάζεται για να καλυφθούν αυτές οι ανάγκες , προσδίδει μεγαλύτερη αξιοπιστία στο σύστημα και κατά συνέπεια επιβεβαιώνει την ανάγκη υλοποίησης στοχευμένων εφαρμογών για την αξιοποίηση της διαθέσιμης πληροφορίας.

Τα δεδομένα που μπορούν να αντληθούν από τη πλατφόρμα ENTSO-E, για τις περισσότερες χώρες είναι επαρκή και ενημερωμένα. Υπάρχουν όμως χώρες για τις οποίες οι πληροφορίες είναι από ελλιπείς έως και ανύπαρκτες (Αλβανία, Λευκορωσία, Κροατία, κ.α.) με αποτέλεσμα να μην είναι δυνατός ο σχηματισμός σαφούς εικόνας για την ενεργειακή τους κατάσταση. Αυτό έχει ως συνέπεια να μη λαμβάνονται υπόψη οι χώρες αυτές σε οποιαδήποτε απόπειρα μελέτης.

Τέλος, όπως έχει αναφερθεί και στην εισαγωγή, τα δεδομένα αφορούν πρωτίστως ζώνες παραγωγής που τις διαχειρίζονται οι TCOs και δευτερευόντως χώρες, καθώς ζώνες παραγωγής μπορούν να εκτείνονται και πέραν των συνόρων μιας μόνο χώρας. Από το πρόγραμμα ENTSO-E δεν είναι δυνατό να αντληθεί η χωρική πληροφορία των ορίων αυτών των ζωνών, παρά μόνο σε μορφή PDF[1] η οποία δεν είναι αξιοποιήσιμη σε επίπεδο υλοποίησης GIS εφαρμογών.

Όσο επεκτείνεται η Εσωτερική Αγορά Ενέργειας στην ΕΕ, τόσο η ανάγκη για πιο ακριβή και μεγάλα σε όγκο δεδομένα θα αυξάνεται. Ταυτόχρονα με αυτή την αύξηση των απαιτήσεων σε πληροφορία, θα αυξάνεται και η ανάγκη δημιουργίας εφαρμογών που θα χρησιμοποιούν τα δεδομένα στο έπακρο τους, ενισχύοντας με το τρόπο αυτό την αξία τους.

[1] Λήψη χαρτών σε μορφή PDF: https://www.entsoe.eu/data/map/downloads/

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΣΩΤΕΡΙΚΗ ΑΓΟΡΑ ΕΝΕΡΓΕΙΑΣ ΣΤΗΝ ΕΥΡΩΠΗ ΚΑΙ ΤΟ ΠΡΟΓΡΑΜΜΑ ENTSO-E

- Thomas-Olivier Léautier & Claude Crampes (2016): Liberalisation of the European electricity markets: a glass half full.
  Ανασύρθηκε στις 12/10/2019 από: https://fsr.eui.eu/liberalisation-european-electricity-markets-glass-half-full/

- ENTSO-E (2015): Electricity Market Transparency.
  Ανασύρθηκε στις 12/10/2019 από: https://www.entsoe.eu/data/transparency-platform/

- ENTSO-E (2015): Who is ENTSO-E?
  Ανασύρθηκε στις 12/10/2019 από: https://www.entsoe.eu/about/inside-entsoe/objectives/

- ENTSO-E (2016): Detailed Data Descriptions.
  Ανασύρθηκε στις 12/10/2019 από:
  https://www.entsoe.eu/fileadmin/user_upload/_library/resources/Transparency/02_MoP%20Ref02%20-%20DDD_V2R5.pdf

- ENTSO-E (2015) Transparency Platform.
  Ανασύρθηκε στις 12/10/2019 από: https://transparency.entsoe.eu/

- ENTSO-E (2015) Static Content.
  Ανασύρθηκε στις 12/10/2019 από:
  https://transparency.entsoe.eu/content/static_content/

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

- PostgreSQL: What is PostgreSQL.
  Ανασύρθηκε στις 05/10/2019 από: https://www.postgresql.org/about/

- PostGIS: Spatial and Geographic objects for PostgreSQL.
  Ανασύρθηκε στις 05/10/2019 από: https://postgis.net/

- Redis: Introduction to Redis.
  Ανασύρθηκε στις 05/10/2019 από: https://redis.io/topics/introduction

- Wikipedia (2019): Database Normalization.
  Ανασύρθηκε στις 05/10/2019 από:
  https://en.wikipedia.org/wiki/Database_normalization

- Wikipedia (2019): Boyce Codd Normal Form.
  Ανασύρθηκε στις 05/10/2019 από: https://en.wikipedia.org/wiki/Boyce-Codd_normal_form

BACK-END

- Adam Wiggins (2017): The Twelve-Factor App.
  Ανασύρθηκε στις 05/10/2019 από: https://www.12factor.net/
- Wikipedia (2019): Representational state transfer.
  Ανασύρθηκε στις 05/10/2019 από:
  https://en.wikipedia.org/wiki/Representational_state_transfer
- Django (2019): The web framework for perfectionists with deadlines.
  Ανασύρθηκε στις 05/10/2019 από: https://www.djangoproject.com/
- Django Rest Framework (2019): Powerful and flexible toolkit for building Web APIs.
  Ανασύρθηκε στις 05/10/2019 από: https://www.django-rest-framework.org/
- Celery (2019): Distributed Task Queue.
  Ανασύρθηκε στις 05/10/2019 από: http://www.celeryproject.org/
- Django Celery Beat (2019): Database-backed Periodic Tasks.
  Ανασύρθηκε στις 05/10/2019 από: https://github.com/celery/django-celery-beat
- Django Webpack Loader (2017): Transparently use Webpack with Django.
  Ανασύρθηκε στις 05/10/2019 από: https://github.com/owais/django-webpack-loader

FRONT-END

- Facebook Inc. (2019): React, a JavaScript library for building user interfaces.
  Ανασύρθηκε στις 05/10/2019 από: https://reactjs.org/
- Dan Abramov (2019): Redux, a predictable state container for JavaScript apps.
  Ανασύρθηκε στις 05/10/2019 από: https://redux.js.org/
- Highcharts (2019): Make your data come alive.
  Ανασύρθηκε στις 05/10/2019 από: https://www.highcharts.com/
- React Material UI (2019): React components for faster and easier web development.
  Ανασύρθηκε στις 05/10/2019 από: https://material-ui.com/
- Vladimir Agafonkin (2019): Leaflet, an open-source JavaScript library for mobile-friendly interactive maps. Ανασύρθηκε στις 05/10/2019 από: https://leafletjs.com/
- Webpack (2019): Bundle your assets.
  Ανασύρθηκε στις 05/10/2019 από: https://webpack.js.org/

ΥΠΟΔΟΜΗ

- Docker Inc. (2019): Modernize your applications, accelerate innovation. Ανασύρθηκε στις 05/10/2019 από: https://www.docker.com/
- Docker Inc. (2019): Overview of Docker Compose. Ανασύρθηκε στις 05/10/2019 από: https://docs.docker.com/compose/
- Docker Inc. (2019): Docker Daemon. Ανασύρθηκε στις 05/10/2019 από: https://docs.docker.com/engine/reference/commandline/dockerd/

ΚΩΔΙΚΑΣ ΕΡΓΑΣΙΑΣ: https://github.com/JohnMoutafis/power_map

ΔΙΕΥΘΥΝΣΗ ΙΣΤΟΤΟΠΟΥ: http://147.102.112.142/

ΠΑΡΑΡΤΗΜΑ Α'

Οι οδηγίες και στις 2 περιπτώσεις αφορούν εγκατάσταση σε σύστημα Linux και προϋποθέτουν βασική γνώση χρήσης terminal. Για τη σωστή λειτουργία της εφαρμογής πρέπει να δημιουργηθεί λογαριασμός στο ENTSO-E και να εκδοθεί ένα κλειδί εισόδου (Web Api Security Token) με αίτηση στο σύστημα.

ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ ΤΟΠΙΚΑ ΓΙΑ ΑΝΑΠΤΥΞΗ

Η τοπική εγκατάσταση προϋποθέτει την δημιουργία λογαριασμού στο github και την εγκατάσταση της Python βιβλιοθήκης VirtualEnv[1] τοπικά στο σύστημα. Ακόμη αναγκαία είναι η εγκατάσταση PostgreSQL και Redis server.

1. Λήψη του κώδικα από το github:

   ```
   $ git clone https://github.com/JohnMoutafis/power_map.git
   ```

   Δημιουργεί ένα φάκελο με όνομα power_map στον οποίο αποθηκεύει τον κώδικα.

2. Κατεύθυνση στον φάκελο της εφαρμογής:

   ```
   $ cd power_map
   ```

3. Δημιουργία εικονικού περιβάλλοντος και εγκατάσταση των αναγκαίων βιβλιοθηκών (υπάρχουν καταγεγραμμένες στο αρχείο requirements_dev.txt) για την ανάπτυξη της εφαρμογής:

   ```
   $ mkvirtualenv power_map —python=python3
   $ workon power_map
   $ pip install -r requirements_dev.txt
   ```

4. Δημιουργία αρχείου .env στο οποίο θα συμπληρωθούν οι εξαρτήσεις περιβάλλοντος (environment variables):

   ```
   $ cp env.example .env
   $ vim .env
   ```

   Η χρήση του vim δεν είναι αναγκαία, μπορεί να χρησιμοποιηθεί οποιοσδήποτε άλλος κειμενογράφος. Μέσω του κειμενογράφου πρέπει να αντικατασταθούν κατάλληλα οι μεταβλητές περιβάλλοντος για το εκάστοτε σύστημα.
   Παράδειγμα αρχείου .env:

   ```
   # Django Setup
   SECRET_KEY=super-secret-key
   DEBUG=True
   ALLOWED_HOSTS=*
   ```

[1] Οδηγίες εγκατάστασης: https://virtualenv.pypa.io/en/latest/installation/

```
ADMINS=admin@mail.com
# Database Setup
POSTGRES_PASS=username
POSTGRES_USER=password
POSTGRES_DB=power_map_db
DATABASE_URL=postgresql://developer:developer@localhost:5432/power_map_db
# Celery Setup
REDIS_URL=redis://localhost:6379
# App Setup
ENTSOE_BASE_URL=https://transparency.entsoe.eu/api
ENTSOE_SECURITY_TOKEN=entsoe-web-api-token
```

5. Αρχικοποίηση της βάσης:

```
$ ./manage.py migrate
$ ./manage.py populate_countries
```

Η δεύτερη εντολή είναι αυτοματισμός που αρχικοποιεί το πίνακα Countries με τα χωρικά (σύνορα) και ποιοτικά (όνομα, κωδικός κτλ.) δεδομένα των χωρών της ΕΕ.

6. Εκκίνηση του Django Server:

```
$ ./manage.py runserver
```

7. Εκκίνηση του Celery Worker:

```
$ celery -A power_map worker -l info
```

8. Εκκίνηση του Celery Beat:

```
$ celery -A power_map beat -l info \
        --scheduler django_celery_beat.schedulers:DatabaseScheduler
```

9. Δημιουργία κεντρικού χρήστη με δικαιώματα διαχειριστή:

```
$ ./manage.py createsuperuser
```

Απαιτεί τη συμπλήρωση των ερωτήσεων που εμφανίζονται στην οθόνη για την επιτυχή δημιουργία του χρήστη.

ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ ΣΕ SERVER

Η εγκατάσταση σε server (ή τοπικά αλλά με τη χρήση Docker) προϋποθέτει την εγκατάσταση Docker[1] και Docker-Compose[2] στο σύστημα και τη λειτουργία του Docker-Daemon.

Τα βήματα 1, 2 και 4 της εγκατάστασης είναι ίδια με αυτά της τοπικής εγκατάσταση που αναλύθηκε παραπάνω και το βήμα 3 **παραλείπεται** καθώς δεν είναι αναγκαίο.

Μετά το 4ο βήμα η εγκατάσταση έχει ως εξής:

5. Κατασκευή της υποδομής:

```
$ docker-compose build
```

6. Εκκίνηση εφαρμογής ως daemon:

```
$ docker-compose up -d
```

7. Για τη δημιουργία κεντρικού χρήστη με δικαιώματα διαχειριστή, απαιτείται πρώτα η σύνδεση στο δομικό στοιχείο **web**[3]:

```
$ docker exec -it web /bin/bash
```

   Στη συνέχεια η διαδικασία είναι ίδια με το βήμα 9 της τοπικής εγκατάστασης.

[1] Οδηγίες εγκατάστασης Docker σε Linux Ubuntu: https://docs.docker.com/install/linux/docker-ce/ubuntu/
[2] Οδηγίες εγκατάστασης Docker-Compose: https://docs.docker.com/compose/install/
[3] βλ. ΣΥΝΤΟΜΗ ΑΝΑΛΥΣΗ docker-compose.yml του κεφαλαίου ΥΛΟΠΟΙΗΣΗ ΥΠΟΔΟΜΗΣ