
**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ**

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα (ΠΛΣ)

# Διπλωματική Εργασία

«Προσομοίωση προγραμμάτων ενεργειακού συμψηφισμού με αποθήκευση»

Γιαννακοσιάν Ανέστης

Επιβλέπων καθηγητής: Καψάλης Βασίλειος

Πάτρα, Ιούνιος 2023

ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

# «Προσομοίωση προγραμμάτων ενεργειακού συμψηφισμού με αποθήκευση»

Γιαννακοσιάν Ανέστης

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Καψάλης Βασίλειος

«Καθηγητής του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου & ΣΕΠ ΕΑΠ»

Συν-Επιβλέπων Καθηγητής:

Παλιόκας Ιωάννης

«Επίκουρος Καθηγητής του Τμήματος Λογιστικής και Χρηματοοικονομικής του Διεθνούς Πανεπιστημίου της Ελλάδος, Πανεπιστημιούπολη Καβάλας»

Πάτρα, Ιούνιος 2023

*«Ευχαριστίες»*

Ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Καψάλη Βασίλειο για την καθοδήγησή του καθ' όλη τη διάρκεια της διπλωματικής μου και την οικογένειά μου για την υπομονή και τη στήριξή της...

# Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η συστηματική τεχνοοικονομική μελέτη όλων των παραμέτρων, που επηρεάζουν το κόστος και το όφελος μίας εγκατάστασης Φωτοβολταϊκού (ΦΒ) συστήματος. Έτσι θα προκύψει ένας αλγόριθμος σχεδίασης υβριδικού ΦΒ συστήματος και ηλεκτρικών συσσωρευτών, με στόχο τη βέλτιστη διαστασιολόγηση του συστήματος και τη μεγιστοποίηση της απόδοσης της επένδυσης. Το αποτέλεσμα της εργασίας θα υλοποιηθεί ως μια διαδικτυακή υπηρεσία υποβοήθησης λήψης αποφάσεων για ΦΒ συστήματα, εντασσόμενα σε προγράμματα ενεργειακού συμψηφισμού με δυνατότητα αποθήκευσης ηλεκτρικής ενέργειας.

**Λέξεις – Κλειδιά**

Ενεργειακός συμψηφισμός, αυτοπαραγωγός, ταυτοχρονισμός, ιδιοκατανάλωση, προσομοίωση, Φωτοβολταϊκά συστήματα, ΦΒ, Django - Python, παραγωγή ηλιακής ενέργειας

# "Simulation of net metering programs with storage"

Giannakosian Anestis

## Abstract

The aim of this diploma thesis is the systematic techno-economic study of all the parameters that affect the cost and benefit of a Photovoltaic (PV) system installation. This will result in an algorithm for the design of a hybrid PV system with battery energy storage (BESS), aiming at the optimal sizing of the system and maximizing the return on investment. The result of this study will be implemented as a web-based decision support service for PV systems in net metering programs with electricity storage capability.

**Keywords**

Net metering, Django - Python, simulation photovoltaic systems, self consumption, production of solar PV energy, PV system

# Περιεχόμενα

# Κατάλογος Εικόνων / Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες & Ακρωνύμια

| | |
|---|---|
| ΦΒ | Φωτοβολταϊκό/ά |
| ΒΔ | Βάση Δεδομένων |
| ΧΤ | Χαμηλή Τάση |
| ΣΕΦ | Σύνδεσμος Εταιριών Φωτοβολταϊκών |
| ΣΕΣ | Σύμβασης Ενεργειακού Συμψηφισμού |
| ΑΠΕ | Ανανεώσιμες Πηγές Ενέργειας |
| ΕΕ | Ευρωπαϊκή Ένωση |
| ΚΠΑ | Καθαρή Παρούσα Αξία |
| ΦΕΚ | Φύλλο Εφημερίδας Κυβερνήσεως |
| PV | Photovoltaic/s |
| PVGIS | Photovoltaic Geographical Information System |
| EVA | Ethylene Vinyl Acetate |
| BESS | Battery Energy Storage System |
| FiT | Feed-in-Tariff |
| NPV | Net Profit Value |
| LCOE | Levelized Cost Of Energy |
| ROI | Return On Investment |
| AROI | Annualized Return On Investment |
| IRR | Internal Rate of Return |
| UX | User Experience |

# 1. Net metering

Στο κεφάλαιο αυτό παρουσιάζονται βασικές πληροφορίες σχετικά με την έννοια του net metering, πως και πότε ξεκίνησε στην Ελλάδα και την πορεία του, το βασικό εξοπλισμό του, όπως και την απήχηση σε άλλες χώρες της Ευρωπαϊκής Κοινότητας. Θα γίνει επιπλέον μία αναφορά στο νέο Πρόγραμμα και πακέτο επιδότησης «Φωτοβολταϊκά στη Στέγη» το οποίο εκκινήθηκε το Μάϊο του 2023 στην Ελλάδα.

Εν μέσω πρωτοφανών υψηλών τιμών ενέργειας, κράτη και καταναλωτές αναζητούν συνεχώς διεξόδους. Το net metering αποτελεί πλέον μία βιώσιμη οικονομικά λύση, ενσωμάτωσης συστημάτων παραγωγής ηλεκτρικής ενέργειας με χρήση ανανεώσιμων πηγών ενέργειας (ΑΠΕ), στο κύριο Δίκτυο διανομής ρεύματος. Με την υιοθέτηση ΑΠΕ, αυτοπαραγωγής και ιδιοκατανάλωσης, δύναται να επέλθει σημαντική μείωση κόστους ηλεκτρικής ενέργειας. Συγκριτικά δε, με μία εγκατάσταση αυτόνομου ΦΒ συστήματος για ανεξάρτητο ενεργειακά οικιακό δίκτυο, το net metering παρουσιάζεται ως μία οικονομικότερη επένδυση, προσβάσιμη για τον μέσο πλέον καταναλωτή.

## 1.1 Τι είναι το net metering

Το net metering είναι γενικά ο συμψηφισμός μεταξύ παραγόμενης και καταναλισκόμενης ενέργειας. Ως ενεργειακός συμψηφισμός ορίζεται: *ο συμψηφισμός της παραγόμενης ηλεκτρικής ενέργειας, από σταθμό ΑΠΕ ή ΣΗΘΥΑ αυτοπαραγωγού με την καταναλισκόμενη ηλεκτρική ενέργεια σε εγκατάσταση του αυτοπαραγωγού, η οποία βρίσκεται στον ίδιο ή όμορο χώρο με το σταθμό ΑΠΕ ή ΣΗΘΥΑ.*[1] Ο αυτοπαραγωγός δύναται να τοποθετήσει σταθμό σε χώρο μακριά από την εγκατάσταση του, *αρκεί να συνδέεται ηλεκτρικά με αποκλειστική γραμμή διασύνδεσης η οποία θα αποτελεί μέρος της εσωτερικής ηλεκτρικής εγκατάστασης*.[2]

Να αναφέρουμε ότι στον ενεργειακό συμψηφισμό η παραγόμενη ενέργεια δεν είναι απαραίτητο να ταυτοχρονίζεται με την καταναλισκόμενη. Ως **ταυτοχρονισμός** ορίζεται: *η κατάσταση κατά την οποία η ηλεκτρική ενέργεια που παράγεται από το Φωτοβολταϊκό καταναλώνεται εκείνη ακριβώς τη χρονική στιγμή στην εγκατάσταση του πελάτη*.[3] Ο ταυτοχρονισμός προκύπτει από το ποσοστό της παραγόμενης ενέργειας που χρησιμοποιείται για να καλύψει απευθείας τις ενεργειακές ανάγκες της εγκατάστασης, σε σχέση με τη συνολική κατανάλωση της εγκατάστασης. Για την ηλεκτρική ενέργεια που

αυτοκαταναλώνεται στην κατοικία με Φ/Β σύστημα, ο αυτοπαραγωγός δε χρεώνεται κόστος προμήθειας ηλεκτρικής ενέργειας, ούτε κόστη χρήσης δικτύου.

Ο ενεργειακός συμψηφισμός διενεργείται, με τελική εκκαθάριση στον πρώτο εκκαθαριστικό λογαριασμό που εκδίδεται με την παρέλευση της **τριετίας**, από την ενεργοποίηση του σταθμού. Επομένως, η πλεονάζουσα ενέργεια στο net metering δεν αποζημιώνεται αλλά πιστώνεται. Η διαδικασία επαναλαμβάνεται ανά τριετία μέχρι τη λήξη της Σύμβασης Ενεργειακού Συμψηφισμού (ΣΕΣ). Όταν επέλθει το τέλος κάθε επόμενης τριετίας ή λήξει η ΣΕΣ, *διενεργείται εκκαθάριση και τυχόν πλεόνασμα εγχυθείσας ενέργειας, δε μεταφέρεται σε επόμενο λογαριασμό, ούτε υφίσταται υποχρέωση για οποιαδήποτε αποζημίωση στον αυτοπαραγωγό για την ενέργεια αυτή*.[2] Ο συμψηφισμός λοιπόν, με βάση τη διαδικασία που αναφέρθηκε είναι ενεργειακός σε kWh, σε αντίθεση με τα προγράμματα Feed-in Tariff όπου είναι λογιστικός. Αυτό έχει ως άμεσο αποτέλεσμα, οι αυξομειώσεις της τιμής της κιλοβατώρας όπως και ο πληθωρισμός να παύουν να επηρεάζουν τον μηνιαίο λογαριασμό του αυτοπαραγωγού, στο βαθμό που επηρεάζεται ένα οικιακό δίκτυο δίχως εγκατάσταση ΦΒ συστήματος net metering.

Η ανάπτυξη ΦΒ συστημάτων από αυτοπαραγωγούς θεσπίστηκε με την Υπουργική Απόφαση ΑΠΕΗΛ/Α/Φ1/οικ.24461, ΦΕΚ Β’ 3583/31.12.2014[4] και αφορά την εγκατάσταση σταθερών ΦΒ συστημάτων για την κάλυψη ιδίων αναγκών από καταναλωτές ηλεκτρικής ενέργειας, με εφαρμογή ενεργειακού συμψηφισμού. Σύμφωνα με το Ν.4414/2016, ΦΕΚ 149Α/9.8.2016[5] *η αυτοπαραγωγή με ενεργειακό συμψηφισμό επεκτάθηκε και σε άλλες τεχνολογίες και συγκεκριμένα στις μικρές ανεμογεννήτριες, σταθμούς βιομάζας/βιοαερίου/βιορευστών, μικρούς υδροηλεκτρικούς σταθμούς και σταθμούς συμπαραγωγής ηλεκτρισμού-θερμότητας (ΣΗΘΥΑ),* ενώ με το Ν.4513/2018, ΦΕΚ 9Α/23.1.2018[6] (που αφορά τις Ενεργειακές Κοινότητες) και την Υπουργική Απόφαση ΥΠΕΝ/ΔΑΠΕΕΚ/15084/382, ΦΕΚ 759Β/5.3.2019[7], *είναι πλέον δυνατή και η εγκατάσταση μονάδων αποθήκευσης σε συνδυασμό με συστήματα αυτοπαραγωγής*.[2]

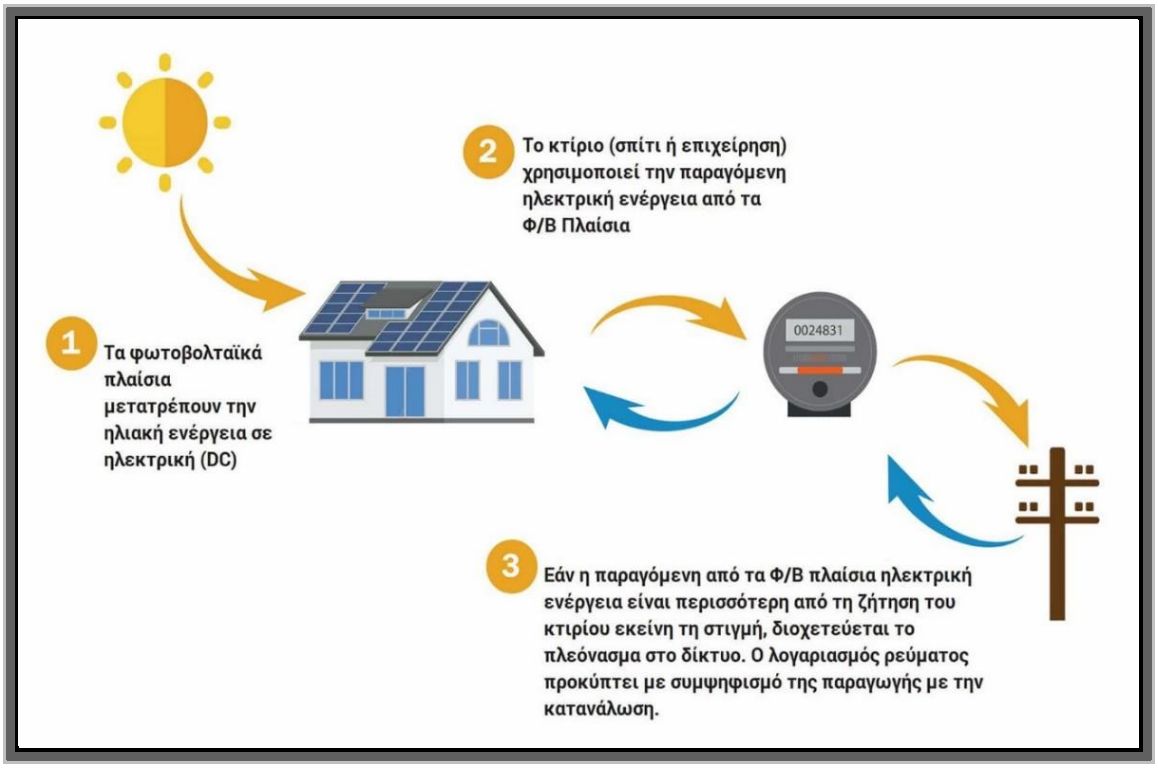
Εδώ ας γίνει μία απλή αναφορά, στον όρο εικονικός ενεργειακός συμψηφισμός (virtual net metering). Εικονικός ενεργειακός συμψηφισμός ορίζεται: *ο συμψηφισμός της παραγόμενης ηλεκτρικής ενέργειας, από σταθμό ΑΠΕ αυτοπαραγωγού, με τη συνολική καταναλισκόμενη ηλεκτρική ενέργεια σε εγκαταστάσεις του αυτοπαραγωγού*,[2] με τη διαφορά ότι τουλάχιστον μία εκ των εγκαταστάσεων αυτών, είτε βρίσκεται σε διαφορετικό χώρο από το σταθμό ΑΠΕ, είτε τροφοδοτείται από διαφορετική παροχή.

## 1.2 Net metering σε PV – BESS συστήματα

Εδώ θα ασχοληθούμε εξ ολοκλήρου με την υλοποίηση του ενεργειακού συμψηφισμού (net metering) σε ΦΒ συστήματα (PV systems) με ή χωρίς τη προσθήκη συμπληρωματικού συστήματος συσσωρευτών (BESS). Για να περιοριστεί το εύρος της υλοποίησης αυτής, θα αναφερόμαστε από εδώ και εμπρός σε οικιακούς καταναλωτές, σε Δίκτυο Χαμηλής Τάσης (ΧΤ) με ισχύ μικρότερη των 25 kVA, οι οποίοι δύνανται να εξυπηρετούνται από οποιαδήποτε πάροχο με τον οποίο έχουν συνάψει σύμβαση. Θα εξαιρεθούν οι καταναλωτές σε Δίκτυα Μέσης Τάσης (ΜΤ) και Υψηλής Τάσης (ΥΤ), καθώς σε αυτές τις δύο κατηγορίες εντάσσονται επιχειρήσεις με συμφωνημένη ισχύ μεγαλύτερη των 250 kVA.

Αναλυτικότερα, ένας αυτοπαραγωγός μπορεί να εγκαταστήσει ΦΒ σύστημα σε επικλινή στέγη, σε ταράτσα ή και σε έκταση επί του εδάφους, σε χώρο ιδιοκτησίας του. Όταν το οικιακό δίκτυο, βρίσκεται σε διαμέρισμα πολυκατοικίας, ο ενδιαφερόμενος αυτοπαραγωγός πρέπει να πάρει έγκριση από τη διαχείριση ή/και τους υπόλοιπους ιδιοκτήτες.

Ξεκινώντας, σημαντική λεπτομέρεια για τα προγράμματα net metering, είναι πως δε μπορεί να υποστηριχθεί η εγκατάσταση ενός αυτόνομου ΦΒ συστήματος (off grid), δηλαδή ο αυτοπαραγωγός θα πρέπει να παραμένει συνδεδεμένος στο Δίκτυο ηλεκτρικής ενέργειας (grid tied ή on grid), έχοντας βέβαια τη δυνατότητα να επιλέξει όποιον πάροχο ηλεκτρικού ρεύματος επιθυμεί. Στην Εικόνα 1-1 της επόμενης σελίδας, φαίνεται η βασική ιδέα και υλοποίηση ενός ΦΒ συστήματος σε στέγη μονοκατοικίας, συνδεδεμένο σε πρόγραμμα net metering.



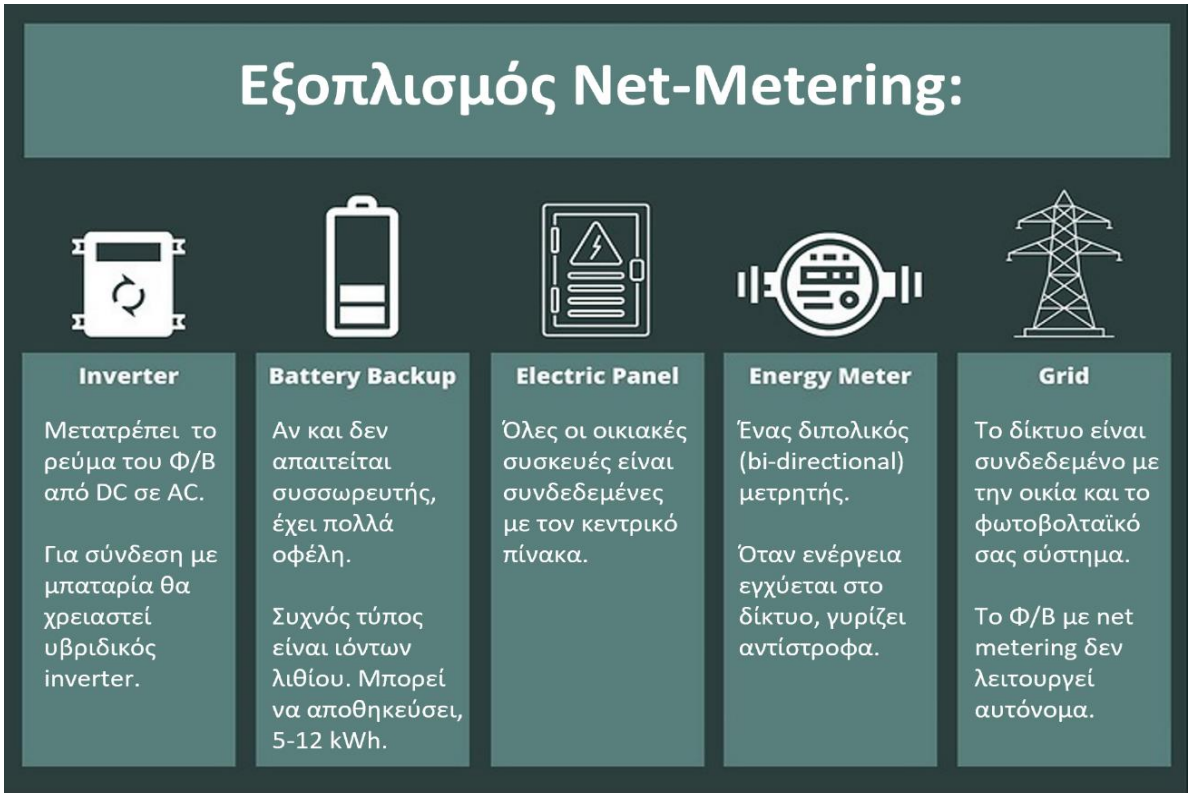

**Εικόνα 1-1 Βασική υλοποίηση net metering**

### 1.2.1 Βασικός εξοπλισμός net metering

Ο βασικός εξοπλισμός, για να θεωρηθεί ολοκληρωμένη μία εγκατάσταση ΦΒ συστήματος, όπως φαίνεται και στην Εικόνα 1-2, αποτελείται από:

- Φωτοβολταϊκά πλαίσια (PV panel)
- Αντιστροφέας (inverter)
- Μετρητικές Διατάξεις & Αμφίδρομος Μετρητής (bi-directional smart meter)
- Σύστημα αποθήκευσης ενέργειας με μπαταρία (BESS)
- Δίκτυο (Grid)

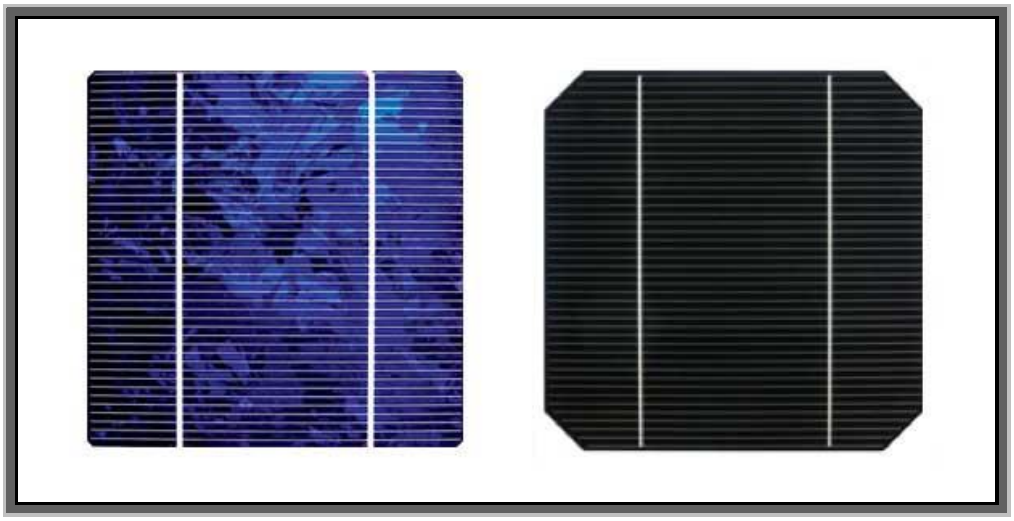
Ίσως, η αναφορά του κεντρικού Δικτύου στο βασικό εξοπλισμό να θεωρηθεί πλεονασμός, αλλά είναι απαραίτητο να γίνει απόλυτα κατανοητό, ότι χωρίς συνδεδεμένη παροχή στο Δίκτυο, net metering δεν υφίσταται.




**Εικόνα 1-2 Εξοπλισμός net metering**

- **Φωτοβολταϊκά πλαίσια (PV panel)**

Τα ΦΒ πλαίσια αποτελούν την «καρδιά» μίας εγκατάστασης. Οι δύο συνήθεις τύποι πάνελ, είναι τα πολυκρυσταλλικά και τα μονοκρυσταλλικά. Η ονομασία προέρχεται από την επεξεργασία του κρυσταλλικού πυριτίου, από το οποίο παράγονται τα μονοκρυσταλλικά και πολυκρυσταλλικά ΦΒ κύτταρα (βλέπε Εικόνα 1-3). Πολλά διατεταγμένα ΦΒ κύτταρα, εγκιβωτίζονται σε προστατευτικό γυαλί (EVA) και πλαίσιο αλουμινίου, για αύξηση αντοχής και προστασίας από περιβαλλοντικούς επιβαρυντικούς παράγοντες π.χ. σκόνη, χαλάζι κτλ., προσδίδοντας εύκολα 200-450 Wp ανά πλαίσιο. Κρίσιμο επιπλέον παράγοντα, αποτελεί ο βαθμός απόδοσης ενός πλαισίου. Στα πολυκρυσταλλικά η απόδοση κυμαίνεται από 16-18%, ενώ στα μονοκρυσταλλικά μπορεί να ξεπεράσει και το 20%. Ένα σύστημα από μονοκρυσταλλικά πάνελ παράγει ετησίως έως και 10% περισσότερη ενέργεια, από ένα σύστημα ίδιου αριθμού πολυκρυσταλλικών πάνελ ίδιων διαστάσεων. Στην αγορά συναντώνται και πάνελ λεπτού υμενίου (thin film), καθώς και υβριδικά πάνελ. Παρόλα αυτά δε θα αναφερθούμε εκτενέστερα στους δύο τελευταίες τύπους πάνελ.

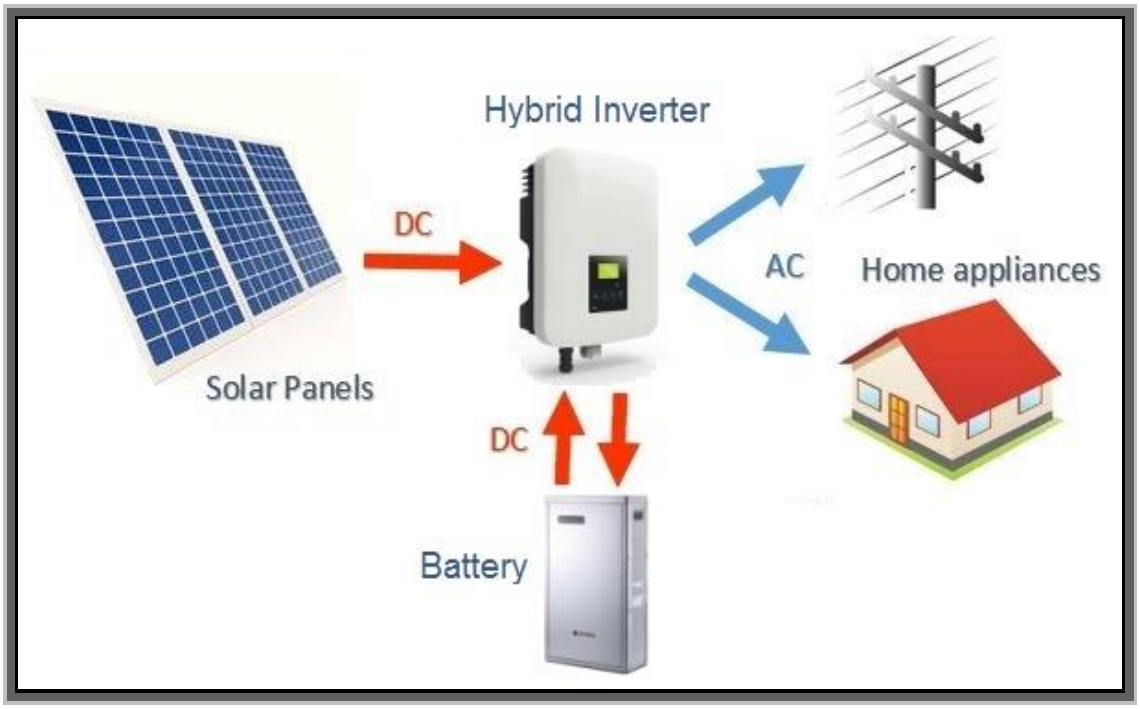


**Εικόνα 1-3 Πολυκρυσταλλικό - μονοκρυσταλλικό ΦΒ κύτταρο**

- **Αντιστροφέας (inverter)**

Ο αντιστροφέας ή μετατροπέας είναι μία ηλεκτρονική συσκευή, η οποία είναι υπεύθυνη να μετατρέψει τη σταθερή τάση (DC) της παραγόμενης από το ΦΒ ενέργειας, σε εναλλασσόμενη (AC), για να μπορέσουν να τροφοδοτηθούν το οικιακό δίκτυο και οι τυπικές συσκευές του. Διακρίνονται σε μονοφασικούς και τριφασικούς, άρα επιλέγονται ανάλογα με τη συμφωνημένη τάση παροχής του αυτοπαραγωγού. Στην αγορά έχουν εισαχθεί τελευταία και οι υβριδικοί αντιστροφείς (hybrid inverters) (βλέπε Εικόνα 1-4), οι οποίοι έχουν εσωτερικά φορτιστή και τη δυνατότητα να συνδεθούν με συσσωρευτές, εφόσον το διασυνδεδεμένο σύστημα net metering συνοδεύεται και από σύστημα αποθήκευσης. Γενικός κανόνας απόδοσης, είναι η ισχύς του αντιστροφέα να είναι ισοδύναμη με την ισχύ του ΦΒ συστήματος. Βέβαια, όσον αφορά την ισχύ ενός υβριδικού αντιστροφέα, δηλαδή για σύνδεση με συσσωρευτές, *δεν δύναται να υπερβαίνει την ονομαστική ισχύ του σταθμού παραγωγής (σε kW), με ανώτατο όριο ισχύος τα 30kVA.*[2]

Μία επιπλέον πληροφορία η οποία μπορεί να αναφερθεί εδώ, είναι πως υπό περιπτώσεις, υπάρχει η δυνατότητα χρήσης τριών (03) μονοφασικών αντιστροφέων σε τριφασική παροχή, έναν για κάθε φάση, ως εναλλακτική πρακτική των επαγγελματιών του χώρου.




**Εικόνα 1-4 Υβριδικός αντιστροφέας**

- **Μετρητικές διατάξεις & αμφίδρομος μετρητής (bi-directional smart meter)**

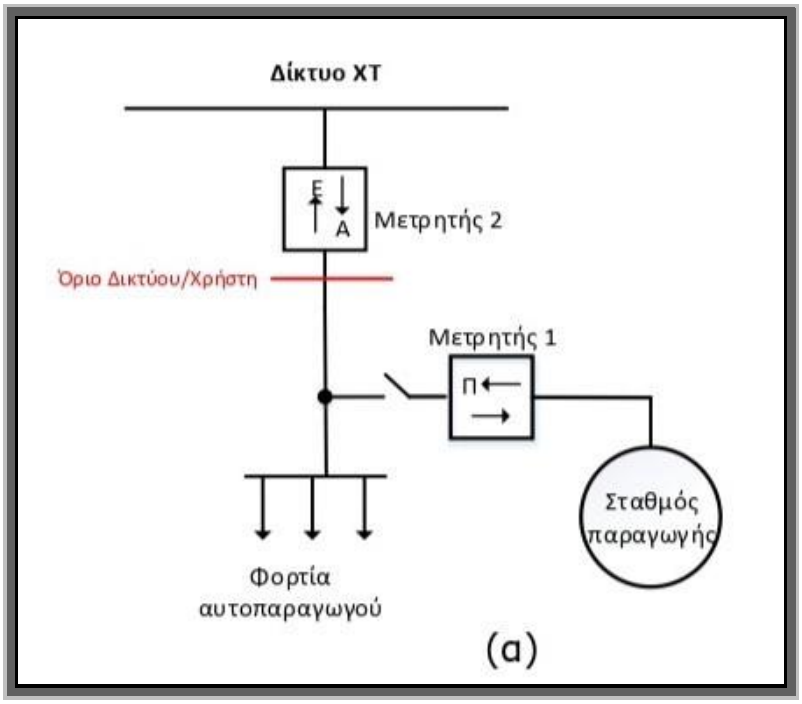
Σύμφωνα με το αναθεωρημένο Τεχνικό Εγχειρίδιο[8] του ΔΕΔΔΗΕ στις 31-08-2023, για την υλοποίηση των συνδέσεων ΦΒ συστημάτων από αυτοπαραγωγούς με ενεργειακό συμψηφισμό στο Δίκτυο ΧΤ απαιτείται: *αφενός η καταγραφή της απορροφώμενης και της εγχεόμενης ενέργειας, από και προς το Δίκτυο, μέσω ενός μετρητή (εφεξής Μετρητής 2) διπλής κατεύθυνσης-καταγραφής (εισερχόμενης και εξερχόμενης ενέργειας) και αφετέρου η καταγραφή της παραγόμενης από το ΦΒ σύστημα ενέργειας, μέσω ενός πιστοποιημένου δεύτερου μετρητή (εφεξής Μετρητής 1).*[8]

Ο Μετρητής 2 εγκαθίσταται στη θέση του υφιστάμενου μετρητή της εγκατάστασης κατανάλωσης (εφόσον ο τελευταίος δεν έχει τη δυνατότητα διπλής κατεύθυνσης - καταγραφής) από τον ΔΕΔΔΗΕ και ανήκει στα πάγια. Ο Μετρητής 1 εγκαθίσταται από τον αυτοπαραγωγό, ο οποίος και τον προμηθεύεται με δικά του έξοδα, με βάση τις υποδείξεις του ΔΕΔΔΗΕ αναφορικά με τους αποδεκτούς τύπους (βλέπε Πίνακα 1-1), ενώ πιστοποιείται προ της τοποθέτησής του στα εργαστήρια του ΔΕΔΔΗΕ. Ο Μετρητής 1 καταγράφει συνεχώς τη συνολική παραγόμενη ενέργεια από το ΦΒ σύστημα, κατά τη διάρκεια της ημέρας. Κατά την ενεργοποίηση της σύνδεσης, το προσωπικό του ΔΕΔΔΗΕ ελέγχει και ρυθμίζει και τους δύο μετρητές και προβαίνει στη σφράγισή τους.

| Εγκεκριμένοι τύποι μετρητών | |
|---|---|
| **Α. Μονοφασικοί μετρητές για ΦΒ ισχύος έως 5 kWp** | 1. Holley Metering Limited τύπου DDSD 285 |
| | 2. Landis & Gyr τύπου ZCF120 |
| | 3. SANXING ELECTRIC τύπου SX1A1-SELS-05 |
| | 4. ELGAMA ELEKTRONIKA τύπου GAMA 100 |
| **Β. Τριφασικοί μετρητές για ΦΒ ισχύος έως 55 kWp** (μέγιστης έντασης 100 Α) | 1. EDMI τύπου ATLAS MK10A WC |
| | 2. ELGAMA ELEKTRONIKA τύπου GAMA300/G3B144 |
| | 3. EMH τύπου LZQJXC |
| | 4. Itron τύπου ACE6000 |
| | 5. Landis & Gyr τύπου ZMD310 |
| | 6. Landis & Gyr τύπου ZMG310 |
| | 7. SANXING ELECTRIC τύπου SX5A2-SELS-04 |
| **Γ. Τριφασικοί μετρητές για ΦΒ ισχύος 55 - 100 kWp** (μέγιστης έντασης 10 Α, σύνδεση μέσω μετασχηματιστών έντασης, με καμπύλη φορτίου και θύρα επικοινωνίας RS485) | 1. EDMI τύπου ATLAS MK10A CT |
| | 2. ELGAMA ELEKTRONIKA τύπου GAMA300/G3B147 |
| | 3. EMH τύπου LZQJXC |
| | 4. Itron τύπου ACE6000 |
| | 5. Itron τύπου SL7000 |
| | 6. Landis & Gyr τύπου ZMD410CT |
| | 7. Landis & Gyr τύπου ZMG410CT |

**Πίνακας 1-1 Εγκεκριμένοι τύποι μετρητών**

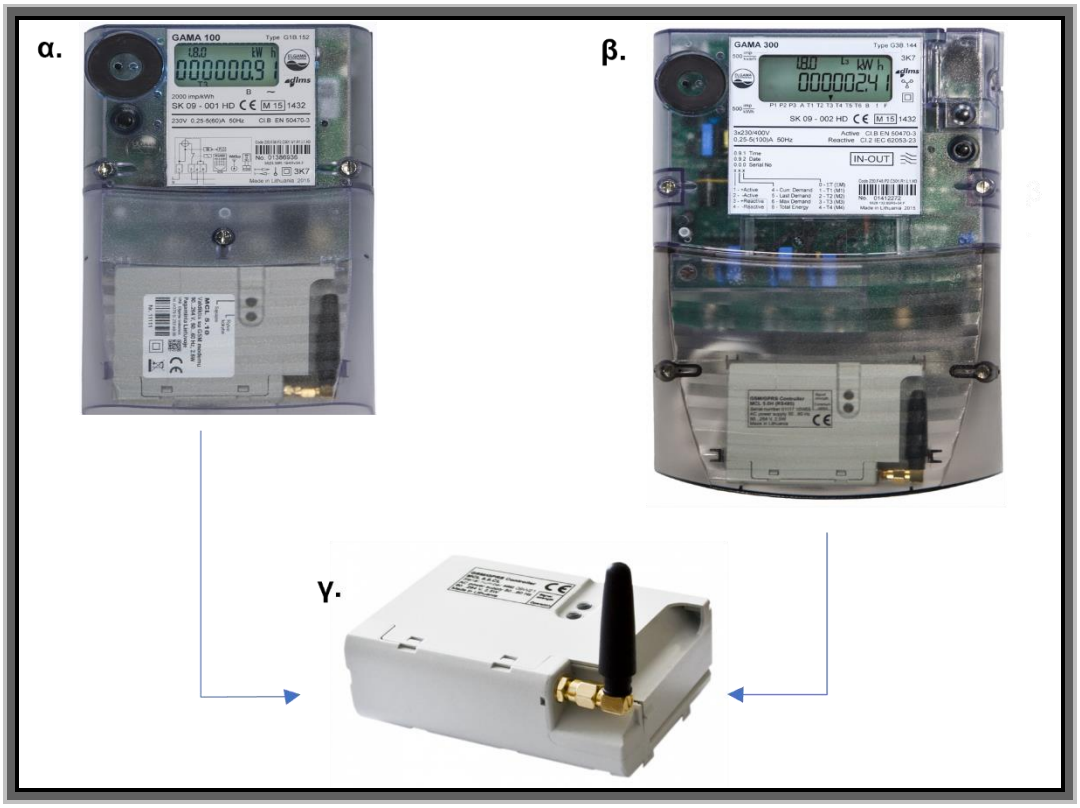
Στο Σχήμα 1-1 φαίνονται ως διάγραμμα οι δύο μετρητές[8], καθώς και τα όρια διαχωρισμού ιδιοκτησίας και ευθύνης μεταξύ Δικτύου και αυτοπαραγωγού, για εγκαταστάσεις που συνδέονται στο Δίκτυο ΧΤ.




**Σχήμα 1-1 Συνδέσεις μετρητικών διατάξεων**

Ο όρος **αμφίδρομος** ή **διπλής κατεύθυνσης**, αναφέρεται στο γεγονός ότι ο Μετρητής 2 μπορεί να μετρήσει τη ροή της ηλεκτρικής ενέργειας και προς τις δύο κατευθύνσεις. Αναλυτικότερα, σε πραγματικό χρόνο (real-time) μετράει την πλεονάζουσα ποσότητα ηλεκτρικής ενέργειας η οποία εγχέεται στο Δίκτυο, καθώς και τη λαμβάνουσα από το Δίκτυο όταν υπάρχουν ανάγκες που δε μπορούν να καλυφθούν, συνήθως κατά τις ώρες μη παραγωγής ενέργειας από το ΦΒ σύστημα. Υπάρχουν κατάλληλοι μετρητές για μονοφασική και τριφασική αντίστοιχα παροχή.

Ένα άλλο σημαντικό χαρακτηριστικό του net metering, είναι ότι ενισχύει την ανάπτυξη τεχνολογιών έξυπνων μετρητών (smart meters). Οι έξυπνοι μετρητές αντικαθιστούν τους συμβατικούς μετρητές ηλεκτρικής ενέργειας και παρέχουν στους καταναλωτές συνεχή πρόσβαση σε ακριβείς πληροφορίες. Οι μετρητές αυτοί, με εγκατάσταση ασύρματου μόντεμ GSM/GPRS, προσφέρουν δυνατότητα εξωτερικής επικοινωνίας, ώστε να γίνεται έλεγχος και εξ αποστάσεως, μέσω τηλεμετρίας ή/και με πρόσβαση μέσω εφαρμογών διαδικτύου. Στην Εικόνα 1-5 φαίνονται δύο μετρητές και ένα modem τηλεμετρίας.



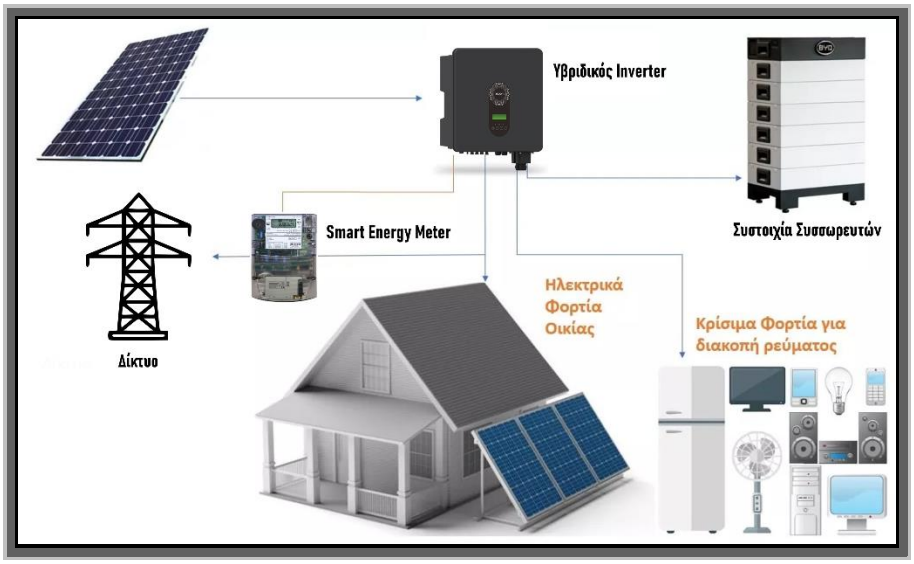

**Εικόνα 1-5 (α.) Μονοφασικός μετρητής, (β.) Τριφασικός μετρητής, (γ). Modem τηλεμετρίας**

- **Σύστημα αποθήκευσης ενέργειας με μπαταρία (BESS)**

Αποτελεί προαιρετικό κομμάτι ενός ΦΒ συστήματος ενεργειακού συμψηφισμού net metering. Με την προσθήκη συσσωρευτών, γίνεται δυνατή η αποθήκευση ενέργειας κατά τη διάρκεια παραγωγής ηλεκτρικής ενέργειας. Κατά προτεραιότητα, ο υβριδικός αντιστροφέας όταν εξασφαλίσει ότι καλύπτονται οι ενεργειακές ανάγκες του δικτύου του αυτοπαραγωγού, αρχίζει να φορτίζει τη συστοιχία και όταν επιτευχθεί πλήρης φόρτιση, η πλεονάζουσα ενέργεια εγχέεται στο Δίκτυο. Κατά τις νυχτερινές (ή συννεφιασμένες-βροχερές) ώρες, πρώτα αποδίδεται η ενέργεια από τη συστοιχία πίσω στο οικιακό δίκτυο, έως το σημείο ασφαλούς αποφόρτισης (ανάλογα τον κατασκευαστή), αυξάνοντας άμεσα την ιδιοκατανάλωση. Στη συνέχεια και εφόσον απαιτηθεί, θα απορροφηθεί επιπλέον ηλεκτρική ενέργεια από το Δίκτυο. **Αύξηση ιδιοκατανάλωσης** λοιπόν, έχουμε όταν το μεγαλύτερο τμήμα της ηλεκτρικής ενέργειας που παράχθηκε από τα ΦΒ, χρησιμοποιήθηκε για να καλύψει τις ενεργειακές ανάγκες της οικίας απευθείας, αλλά και όταν η συστοιχία τροφοδοτεί το δίκτυο με το αποθηκευμένο ρεύμα. Η λειτουργία αυτή μειώνει και το κόστος χρήσης δικτύου, τις γνωστές ρυθμιζόμενες χρεώσεις του λογαριασμού ρεύματος (Δίκτυο Διανομής και Μεταφοράς) ενώ βοηθάει και το Δημόσιο Δίκτυο να είναι πιο ευσταθές.

Σε περίπτωση διακοπής ρεύματος, εάν υπάρχει βοηθητικό σύστημα συσσωρευτών, είναι δυνατόν να δημιουργηθεί μία γραμμή back-up για κάποιες ώρες και φορτία ασφάλειας (βλέπε Εικόνα 1-6), ανάλογα με τα χαρακτηριστικά των συσσωρευτών και του inverter. Τα φορτία αυτά θα μπορούσαν να τροφοδοτήσουν π.χ. κάποιο ψυγείο, το φωτισμό και την τηλεόραση. Σε μερικές περιπτώσεις τριφασικής παροχής μπορεί να δοθεί λίγο μεγαλύτερη αυτονομία, με χρήση συστοιχίας μεγαλύτερης χωρητικότητας, ειδικού πίνακα και σύνδεσης μεταγωγής, χωρίς βέβαια η μεταγωγή να είναι ακαριαία. Πρέπει να περάσουν κάποια δευτερόλεπτα, *μέχρι να δημιουργηθεί το μικροδίκτυο καθώς δεν υπάρχει μετασχηματιστής στον inverter όπως γίνεται σε ένα αυτόνομο ΦΒ*.[9]

Όσον αφορά το σύστημα αποθήκευσης, πρέπει να έχει *κατ' ελάχιστο εγκατεστημένη χωρητικότητα ίση με την εγκατεστημένη ισχύ παραγωγής του φωτοβολταϊκού σταθμού για μία ώρα*.[10] Συνηθέστερος τύπος μπαταριών που χρησιμοποιούνται σε μία συστοιχία είναι λιθίου υψηλής τάσης και η χωρητικότητα αυξάνεται με παραλληλισμό περισσότερων συσσωρευτών. Η συστοιχία συσσωρευτών φορτίζεται αποκλειστικά από την παραγόμενη ενέργεια ΦΒ συστήματος και τροφοδοτεί μόνο τα φορτία του σπιτιού, δίχως να επιτρέπεται να επιστρέφει ενέργεια στο Δίκτυο ή να φορτίζεται από το Δίκτυο.




**Εικόνα 1-6 Φορτία ασφαλείας**

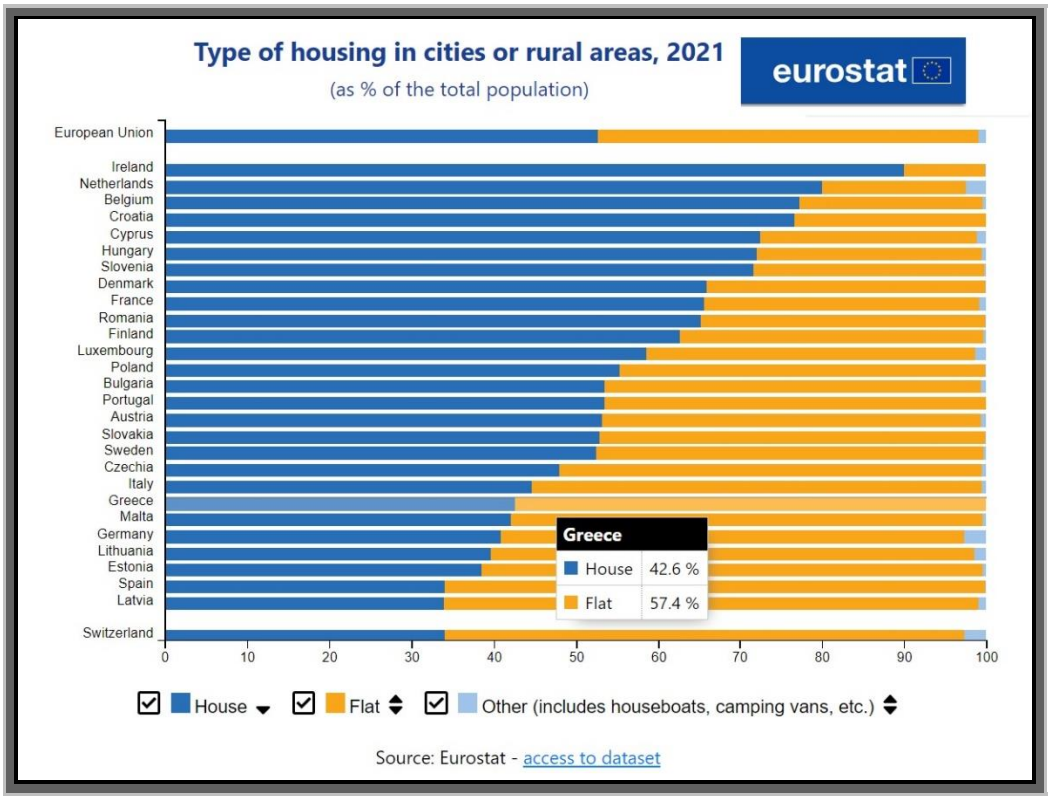
- **Δίκτυο (Grid)**

Όπως αναφέρθηκε και στην αρχή της ενότητας, ο αυτοπαραγωγός με σύστημα net metering, δε μπορεί να αποκοπεί από το Δίκτυο, όπως στα αυτόνομα ΦΒ συστήματα. Άρα λοιπόν το Δίκτυο αποτελεί το «αποθετήριο» ηλεκτρικής ενέργειας του αυτοπαραγωγού, έως ότου απαιτηθεί αυτή η ενέργεια από το οικιακό δίκτυο. Υπενθυμίζουμε ότι αυτή η μορφή πίστωσης για μία συγκεκριμένη ποσότητα ηλεκτρικής ενέργειας, διατηρείται έως την παρέλευση τριετίας, από τη στιγμή της έγχυσης της, άρα και ενσωμάτωσής της στο Δίκτυο. Ύστερα αφαιρείται αυτό το δικαίωμα από τον αυτοπαραγωγό και η εγχυθείσα ενέργεια χάνει την αναδρομική της ισχύ.

## 1.3 Το net metering στην Ελλάδα

Σύμφωνα με μελέτη που πραγματοποίησε το Ίδρυμα Οικονομικών & Βιομηχανικών Ερευνών (ΙΟΒΕ) ως προς την ηλιοθερμική ενέργεια στην Ελλάδα, το 25-30% των νοικοκυριών έχει εγκατεστημένο ηλιακό θερμοσίφωνα, αξιοποιώντας το ηλιακό δυναμικό. Αυτό κατατάσσει την Ελλάδα στην 5η θέση παγκοσμίως και στην 2η θέση πανευρωπαϊκά. Καθυστέρησε όμως αρκετά η δημιουργία κατάλληλων συνθηκών προώθησης της παραγωγής ηλεκτρικής ενέργειας, με χρήση ηλιακής ακτινοβολίας. Τα ΦΒ συστήματα ξεκίνησαν να επιδοτούνται από το Ελληνικό Κράτος το 2006 και με τον νόμο Ν.3468/2006, ΦΕΚ 129Α/29-6-2006[11] «Παραγωγή Ηλεκτρικής Ενέργειας από Ανανεώσιμες Πηγές Ενέργειας», σηματοδοτήθηκε η έναρξη επενδύσεων σε ΦΒ σταθμούς.

Η αυτοπαραγωγή με ενεργειακό συμψηφισμό παρόλα αυτά, εισήχθη νομοθετικά στα τέλη του 2014. Επίσης και σύμφωνα με τα στατιστικά της Eurostat για το έτος 2021, η Ελλάδα έχει μια ιδιαιτερότητα, ότι κοντά το 60% του πληθυσμού διαμένει σε πολυκατοικίες (βλέπε Εικόνα 1-7). Αυτά συνετέλεσαν σε αύξηση δυσκολίας για εγκατάσταση ΦΒ συστημάτων σε κοινόχρηστη ταράτσα, από πλευράς διαθέσιμου χώρου και κατάλληλης συνεννόησης μεταξύ ενοίκων και ιδιοκτητών.




**Εικόνα 1-7 Στατιστικά Eurostat, ποσοστό διαμονής σε διαμερίσματα πολυκατοικίας**

### 1.3.1 Ιστορική αναδρομή net metering

Στη χώρα μας οι όροι και οι προϋποθέσεις για την αυτοπαραγωγή με ενεργειακό συμψηφισμό, καθορίστηκαν με την Υπουργική Απόφαση ΑΠΕΗΛ/Α/Φ1/οικ. 24461/2014, ΦΕΚ 3583/Β/31-12-2014[4] «Εγκατάσταση μονάδων ΑΠΕ από αυτοπαραγωγούς με συμψηφισμό ενέργειας κατ' εφαρμογή του άρθρου 14Α του Ν. 3468/2006». Σύμφωνα με τα στατιστικά στοιχεία του Συνδέσμου Εταιριών Φωτοβολταϊκών (ΣΕΦ) από τον Μάιο του 2015, όταν τέθηκε σε ισχύ η εφαρμογή του προγράμματος net metering, έως και το έτος 2017 είχαν υποβληθεί 1.584 αιτήσεις, όπως φαίνεται και στον Πίνακα 1-2. Από αυτές, οι 1.113 αφορούσαν το Διασυνδεδεμένο Σύστημα, οι 454 τα Μη Διασυνδεδεμένα Νησιά και οι 17 τα συστήματα εικονικού ενεργειακού συμψηφισμού. Η ισχύς των συστημάτων που

είχαν συνδεθεί και λειτούργησαν ανήλθε περίπου στα 12,9 MWp και αντιστοιχούσε στο 35% της συνολικής ισχύος των αιτήσεων (37,05 MWp).[12]

| | Συστήματα ενεργειακού συμψηφισμού | | | | Συστήματα εικονικού ενεργειακού συμψηφισμού | | Σύνολο | |
|---|---|---|---|---|---|---|---|---|
| | Διασυνδεδεμένο Σύστημα | | Μη Διασυνδεδεμένα Νησιά | | | | | |
| | Αριθμός | Ισχύς ($MW_p$) | Αριθμός | Ισχύς ($MW_p$) | Αριθμός | Ισχύς ($MW_p$) | Αριθμός | Ισχύς ($MW_p$) |
| Αιτήσεις | 1.113 | 28,4 | 454 | 8,05 | 17 | 0,6 | 1.584 | 37,05 |
| Προσφορές σύνδεσης/ Συμβάσεις σύνδεσης | 1.009 | 23,1 | 427 | 7,84 | 9 | 0,24 | 1.445 | 31,18 |
| Ενεργοποιήσεις | 692 | 11 | 155 | 1,86 | 2 | 0,04 | 849 | 12,9 |

**Πίνακας 1-2 Συστήματα net metering σε λειτουργία το 2017**

Έως το 2020 από τη συνολικά εγκατεστημένη ισχύ της χώρας, τα ΦΒ συστήματα με ενεργειακό συμψηφισμό ή εικονικό ενεργειακό συμψηφισμό αντιστοιχούσαν μόλις στο 1,36%. Ωστόσο από το 2020 σημειώθηκε ανοδική τάση, καθώς η αγορά αυτοπαραγωγής διπλασιάστηκε σε σχέση με την προηγούμενη χρονιά, παραμένοντας παρόλα αυτά σε επίπεδα σημαντικά χαμηλότερα του δυναμικού της χώρας.

Πιο συγκεκριμένα, εγκαταστάθηκαν 17 MW νέων συστημάτων με ενεργειακό συμψηφισμό ή εικονικό ενεργειακό συμψηφισμό, ανεβάζοντας τη συνολική ισχύ της κατηγορίας αυτής στα 51 MW, με τα εμπορικά συστήματα να καλύπτουν ποσοστό άνω του 90% αυτής της ισχύος και τα οικιακά συστήματα να υπολείπονται σημαντικά. Σημαντικό να αναφερθεί πως το 2020 η αγορά ΦΒ συντήρησε (εν μέσω πανδημίας) 42.200 ισοδύναμες θέσεις πλήρους απασχόλησης (άμεσες, έμμεσες και συνεπαγόμενες).[13]

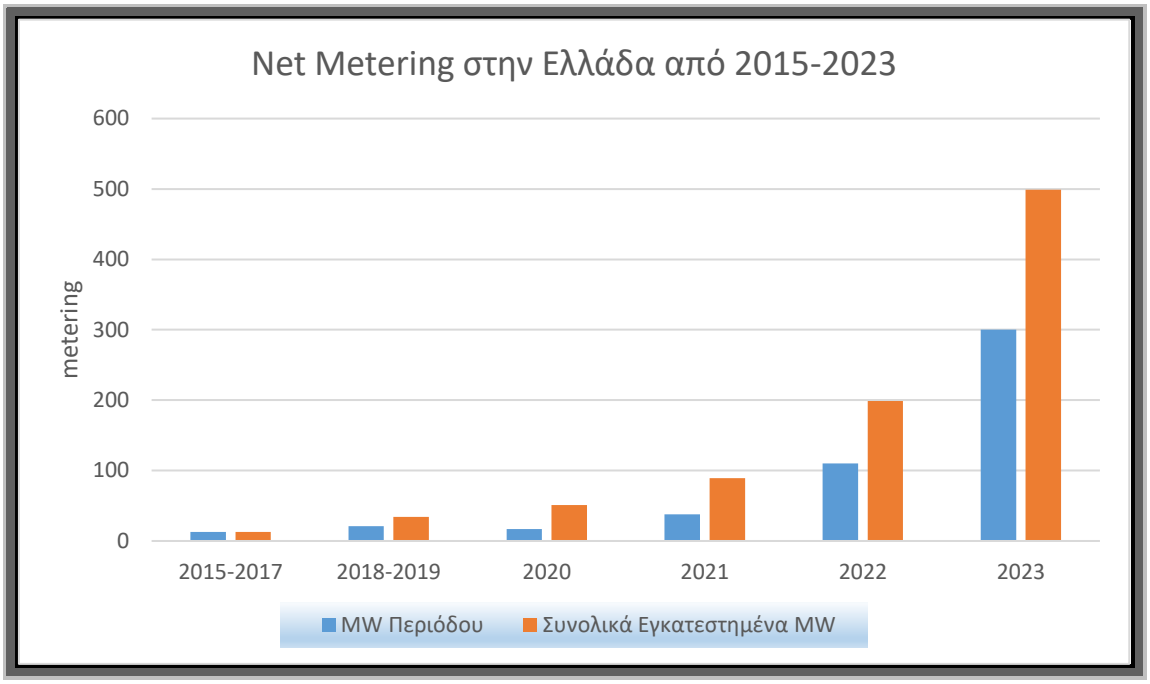
Το 2021, η αγορά των συστημάτων αυτοπαραγωγής υπερδιπλασιάστηκε σε σχέση με την προηγούμενη χρονιά. Πιο συγκεκριμένα, εγκαταστάθηκαν 38 MWp νέων συστημάτων με ενεργειακό συμψηφισμό ή εικονικό ενεργειακό συμψηφισμό, ανεβάζοντας τη συνολική ισχύ της κατηγορίας αυτής στα 89 MWp (βλέπε Πίνακα 1-3).[14]

| Ισχύς net metering 2021 | | |
|---|---|---|
| Νέα εγκατεστημένη ισχύς 2021 | 38 MWp | (98% εμπορικά, 2% οικιακά ως προς την ισχύ) |
| Συνολική ισχύς αυτοπαραγωγής | 89 MWp | (96% εμπορικά, 4% οικιακά ως προς την ισχύ) |

**Πίνακας 1-3 Στοιχεία εγκατεστημένης ισχύος 2021**

Οι εγκαταστάσεις ηλιακών συλλεκτών το 2022 έφτασαν στο ιστορικό υψηλό των 1.362 MW, ξεπερνώντας το προηγούμενο ρεκόρ ισχύος των 838 MW που σημειώθηκε το προηγούμενο έτος. Η περσινή εκτίναξη των συνδέσεων ΦΒ πάνελ στο δίκτυο οδήγησε το χαρτοφυλάκιο των εγκατεστημένων ΦΒ της χώρας από το 2010 σε 5.488 MW. Οι εγκαταστάσεις net metering αυξήθηκαν επίσης σημαντικά το 2022, φθάνοντας σε συνολική δυναμικότητα 110 MW, περίπου τριπλάσια της δυναμικότητας που συγκεντρώθηκε το 2021, σύμφωνα με τα προκαταρκτικά στοιχεία του ΣΕΦ. Η ενίσχυση του net metering το 2022 βέβαια, συνδέθηκε σχεδόν αποκλειστικά με εγκαταστάσεις συστημάτων που έγιναν για εμπορικούς σκοπούς.

Για το 2023, εκτιμάται ότι θα προστεθούν άλλα 300 MW, λόγω και του νέου επιδοτούμενου προγράμματος «Φωτοβολταϊκά στη Στέγη»[15] το οποίο ξεκίνησε στις 02 Μαΐου 2023. Στην περίοδο λοιπόν 2023-2024 υπολογίζεται ότι η συνολική ισχύς προγραμμάτων αυτοπαραγωγής με ενεργειακό συμψηφισμό θα καταφέρει να ξεπεράσει το μισό GW. Στο Σχήμα 1-2 παρουσιάζεται γραφικά, η σταδιακή αύξηση ισχύος από το 2015 έως και σήμερα.




**Σχήμα 1-2 Ισχύς σε MW στην Ελλάδα (2015-2023)**

### 1.3.2 Πρόγραμμα «Φωτοβολταϊκά στη Στέγη» 2023

Στις 02 Μαΐου 2023 υπογράφηκε η Υπουργική Απόφαση Αριθμ. ΥΠΕΝ/ΥΔΕΝ/47129/720, ΦΕΚ 2903Β'/2.5.2023[15] Προκήρυξη του Προγράμματος «Φωτοβολταϊκά στη Στέγη». Σύμφωνα με το Άρθρο 1 αναλύεται το αντικείμενο και ο στόχος του Προγράμματος. Το Πρόγραμμα «Φωτοβολταϊκά στη Στέγη», εφεξής «Πρόγραμμα», επιχορηγεί τα νοικοκυριά για την εγκατάσταση ΦΒ συστημάτων με σύστημα αποθήκευσης και τους αγρότες για την εγκατάσταση ΦΒ συστημάτων με ή χωρίς σύστημα αποθήκευσης για αυτοκατανάλωση με εφαρμογή ενεργειακού συμψηφισμού. Το Πρόγραμμα αφορά σε όλη την Επικράτεια και αναμένεται να έχει σημαντικά οφέλη, τόσο για τους δικαιούχους όσο και συνολικά για την εθνική οικονομία.

Το νέο πρόγραμμα αποτελεί μία σημαντική προτροπή για πολύ κόσμο να επωφεληθεί από τα μεγάλα ποσοστά έκπτωσης και να γίνει αυτοπαραγωγός net metering. Ιδιαίτερη έμφαση δίνεται στην έκπτωση έως και 100% στην αγορά μπαταρίας, ανάλογα με τα εισοδηματικά κριτήρια. Για Μονοφασική Παροχή, το όριο ισχύος ΦΒ σταθμού είναι 5kWp και για Τριφασική Παροχή είναι 10,8kWp. Το σύστημα αποθήκευσης έχει κατ' ελάχιστο εγκατεστημένη χωρητικότητα, ίση με την εγκατεστημένη ισχύ παραγωγής του ΦΒ σταθμού για μία ώρα, με ανώτερο όριο τις 10,8kWh. Στους Πίνακες 1-4 και 1-5, παρουσιάζονται τα ποσοστά επιχορήγησης και τα ανώτατα ποσά σε €, του Προγράμματος.

| Κατηγορία | Ωφελούμενοι | Ποσοστό Επιχορήγησης ΦΒ σταθμού(%) | | Ποσοστό Επιχορήγησης Μπαταρίας(%) | |
|---|---|---|---|---|---|
| | | Ισχύς ≤ 5kWp | 5kWp <Ισχύς≤10,8kWp | Χωρητικότητα ≤5kWh | 5kWh < Χωρητικότητα ≤ 10,8kWh |
| Α | Οικιακές εγκαταστάσεις (Ευάλωτα Νοικοκυριά) | 65% | 60% | 100% | 100% |
| Β | Οικιακές εγκαταστάσεις (Ατομικό Εισόδημα <= 20,000€ ή Οικογενειακό Εισόδημα <= 40.000€) | 35% | 25% | 100% | 100% |
| Γ | Οικιακές εγκαταστάσεις (Ατομικό Εισόδημα > 20,000€ ή Οικογενειακό Εισόδημα > 40.000€) | 30% | 20% | 90% | 90% |

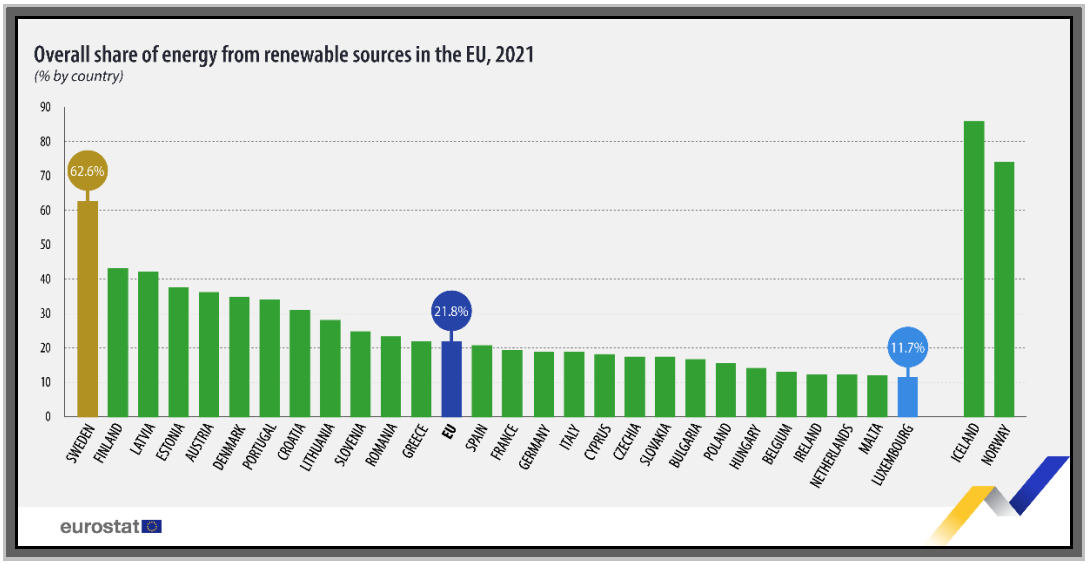
**Πίνακας 1-4 Ποσοστά επιχορηγήσεων Προγράμματος**

| Κατηγορία | Ωφελούμενοι | Ανώτατο ποσό επιχορήγησης ΦΒ σταθμού (€ / kW) Ισχύς <= 5 kWp | Ανώτατο ποσό επιχορήγησης ΦΒ σταθμού (€ / kW) 5kWp<Ισχύς<=10,8kWp | Ανώτατο ποσό Επιχορήγησης Μπαταρίας (€ / kWh) Χωρητικότητα <= 5kWh | Ανώτατο ποσό Επιχορήγησης Μπαταρίας (€ / kWh) 5 kWh < Χωρητικότητα <=10,8 kWh |
|---|---|---|---|---|---|
| **Α** | Οικιακές Εγκαταστάσεις (**Ευάλωτα Νοικοκυριά**) | **(1.200€/kWp)**<br>• 3kWp->3.600€<br>• 4kWp->4.800€<br>• 5kWp->6.000€ | **(830€/kWp)**<br>• 5kWp->4.150€<br>• 6kWp->4.980€<br>• 7kWp->5.810€<br>• 8kWp->6.640€<br>• 9kWp->7.470€<br>• 10kWp->8.300€ | **(890€/kWh)**<br>• 5kWh->4.450€ | **(820€/kWh)**<br>• 5kWh->4.100€<br>• 7.5kWh->6.150€<br>• 10kWh->8.200€ |
| **Β** | Οικιακές Εγκαταστάσεις (**Ατομικό Εισόδημα <=20.000€ ή Οικογενειακό Εισόδημα <= 40.000€**) | **(650€/kWp)**<br>• 3kWp->1.950€<br>• 4kWp->2.600€<br>• 5kWp->3.250€ | **(350€/kWp)**<br>• 5kWp->1.750€<br>• 6kWp->2.100€<br>• 7kWp->2.450€<br>• 8kWp->2.800€<br>• 9kWp->3.150€<br>• 10kWp->3.500€ | **(890€/kWh)**<br>• 5kWh->4.450€ | **(820€/kWh)**<br>• 5kWh->4.100€<br>• 7kWh->6.150€<br>• 10kWh->8.200€ |
| **Γ** | Οικιακές Εγκαταστάσεις (**Ατομικό εισόδημα >20.000€ ή Οικογενειακό Εισόδημα > 40.000€**) | **(560€/kWp)**<br>• 3kWp->1.680€<br>• 4kWp->2.240€<br>• 5kWp->2.800€ | **(280€/kWp)**<br>• 5kWP->1.400€<br>• 6kWp->1.680<br>• 7kWp->1.960€<br>• 8kWp->2.240€<br>• 9kWp->2.520€<br>• 10kWp->2.800€ | **(800€/kWh)**<br>• 5kWh->4.000€ | **(750€/kWh)**<br>• 5kWh->3.750€<br>• 7.5kWh->5.625€<br>• 10kWh->7.500€ |

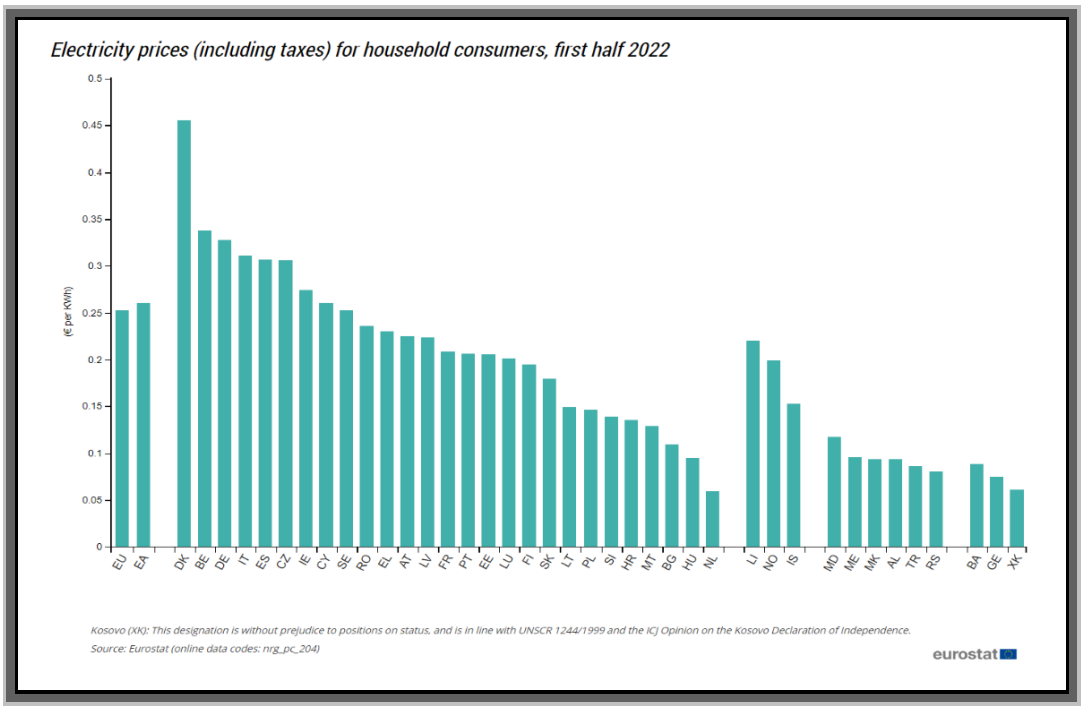
**Πίνακας 1-5 Ανώτατα ποσά επιχορηγήσεων Προγράμματος**

## 1.4 Χρήση ΑΠΕ στην Ευρωπαϊκή Ένωση

Ένας από τους κύριους προβληματισμούς και στόχους του πλαισίου πολιτικής της Ευρωπαϊκής Ένωσης (ΕΕ) για το κλίμα και την ενέργεια, είναι η προώθηση και η υιοθέτηση των ΑΠΕ. Μπορούμε να διακρίνουμε στα δύο Σχήματα 1-3 και 1-4, το συνολικό μερίδιο (overall share) από ΑΠΕ στην ΕΕ για το 2021 και τις αυξημένες τιμές ηλεκτρισμού, για το πρώτο μισό του 2022 από την έρευνα της Eurostat[16]:




**Σχήμα 1-3 Κατανομή ενέργειας από ΑΠΕ στην ΕΕ (2021)**




**Σχήμα 1-4 Τιμές ρεύματος, πρώτο μισό 2022**

Περισσότερα από τα μισά κράτη μέλη της ΕΕ, βρίσκονται κάτω από τον μέσο όρο. Συνολικά, 15 από τα 27 μέλη ανέφεραν μερίδια κάτω από τον μέσο όρο της ΕΕ το 2021 (Βέλγιο, Βουλγαρία, Τσεχία, Γερμανία, Ιρλανδία, Ισπανία, Γαλλία, Ιταλία, Κύπρος, Λουξεμβούργο, Ουγγαρία, Μάλτα, Κάτω Χώρες, Πολωνία και Σλοβακία). Τα χαμηλότερα ποσοστά ΑΠΕ καταγράφηκαν στο Λουξεμβούργο (11,7%), τη Μάλτα (12,2%), τις Κάτω Χώρες (12,3%), την Ιρλανδία (12,5%) και το Βέλγιο (13,0%).

Εκτός από την επίδραση που είχε η άρση των περιορισμών COVID-19 το 2021 στην αύξηση της κατανάλωσης ενέργειας, η οποία μείωσε το μερίδιο των ΑΠΕ (παρά την αύξηση της παραγωγής ενέργειας από ανανεώσιμες πηγές σε απόλυτους αριθμούς σε σύγκριση με το 2020), η αλλαγή της μεθοδολογίας συμβάλλει επίσης στην εξήγηση αυτής της εξέλιξης.

### 1.4.1 Κοινοτικές οδηγίες Ευρωπαϊκού Κοινοβουλίου

Με σχετικές οδηγίες ορίστηκαν οι στόχοι για κάθε χώρα της Ένωσης. Ως ορόσημα, θα γίνει αναφορά σε δύο εκ των οδηγιών:

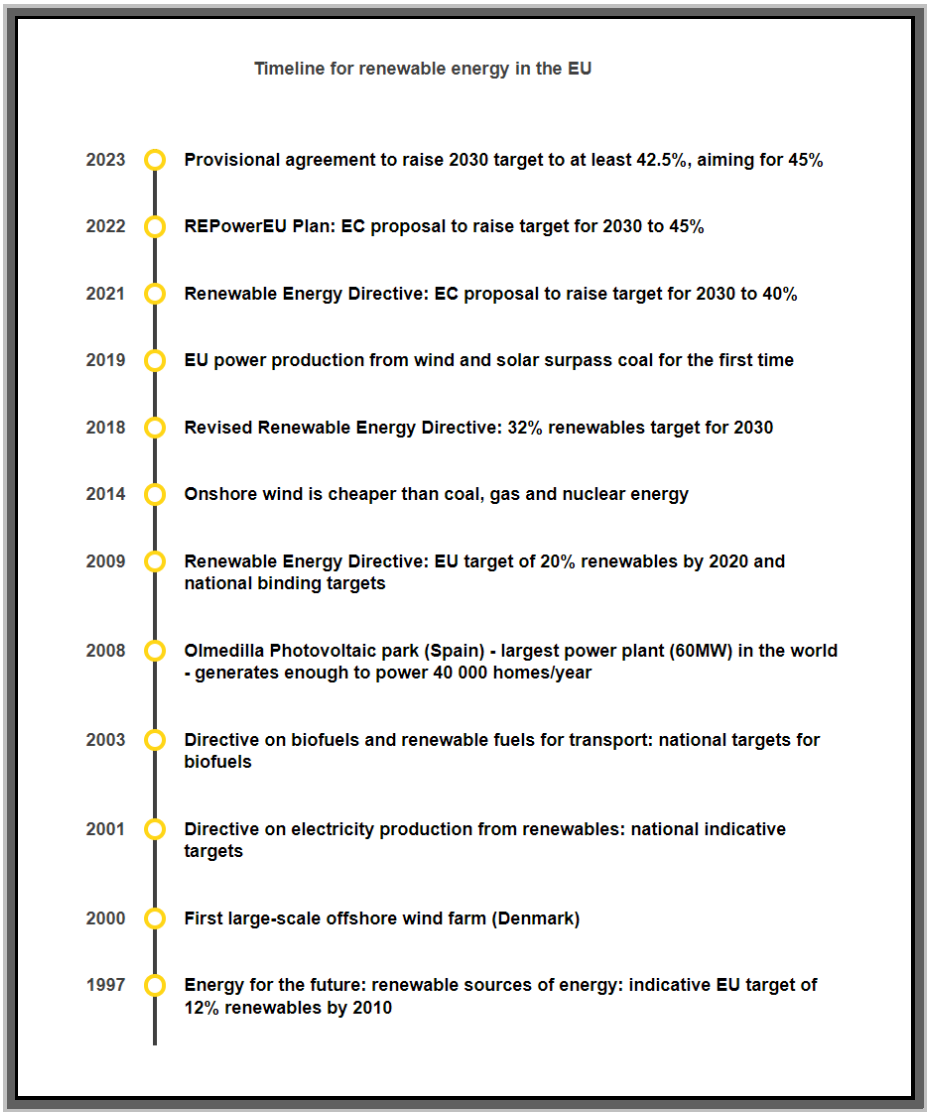
α) «ΟΔΗΓΙΑ 2009/28/ΕΚ»[17] του Ευρωπαϊκού Κοινοβουλίου και του Συμβουλίου της 23ης Απριλίου 2009 σχετικά με την προώθηση της χρήσης ενέργειας από ανανεώσιμες πηγές και την τροποποίηση και τη συνακόλουθη κατάργηση των οδηγιών 2001/77/ΕΚ και 2003/30/ΕΚ. Η οδηγία αναφέρει στο άρθρο 3, παράγραφος 1: *Κάθε κράτος μέλος μεριμνά ώστε το μερίδιο της ενέργειας από ανανεώσιμες πηγές, το οποίο υπολογίζεται σύμφωνα με τα άρθρα 5 έως 11, στην ακαθάριστη τελική κατανάλωση ενέργειας το 2020 να αντιστοιχεί τουλάχιστον στον εθνικό συνολικό στόχο του όσον αφορά το μερίδιο της ενέργειας από ανανεώσιμες πηγές κατά το εν λόγω έτος, όπως αυτό προβλέπεται στην τρίτη στήλη του πίνακα του μέρους Α του παραρτήματος Ι. Αυτοί οι δεσμευτικοί εθνικοί συνολικοί στόχοι είναι σύμμορφοι προς τον στόχο σύμφωνα με τον οποίο το μερίδιο της ενέργειας από ανανεώσιμες πηγές στην ακαθάριστη τελική κατανάλωση ενέργειας της Κοινότητας πρέπει το 2020 να ανέρχεται σε τουλάχιστον 20 %. Για να επιτευχθούν ευκολότερα αυτοί οι στόχοι που ορίζονται σε αυτό το άρθρο, κάθε κράτος μέλος προωθεί και ενθαρρύνει την απόδοση ενέργειας και την εξοικονόμηση ενέργειας*. Η οδηγία 2009/28/ΕΚ καθόρισε τους εθνικούς στόχους για τις ανανεώσιμες πηγές ενέργειας για το 2020 για κάθε χώρα, λαμβάνοντας υπόψη το σημείο εκκίνησης και το συνολικό δυναμικό της για τις ΑΠΕ. Ο Πίνακας 1-6 δίνει μία εικόνα της τότε κατανομής:

| | Μερίδιο ενέργειας από ανανεώσιμες πηγές στην ακαθάριστη τελική κατανάλωση ενέργειας το 2005 ($S_{2005}$) | Στόχος για το μερίδιο ενέργειας από ανανεώσιμες πηγές στην ακαθάριστη τελική κατανάλωση ενέργειας το 2020 ($S_{2020}$) |
|---|---|---|
| Βέλγιο | 2,2 % | 13 % |
| Βουλγαρία | 9,4 % | 16 % |
| Τσεχική Δημοκρατία | 6,1 % | 13 % |
| Δανία | 17,0 % | 30 % |
| Γερμανία | 5,8 % | 18 % |
| Εσθονία | 18,0 % | 25 % |
| Ιρλανδία | 3,1 % | 16 % |
| Ελλάδα | 6,9 % | 18 % |
| Ισπανία | 8,7 % | 20 % |
| Γαλλία | 10,3 % | 23 % |
| Ιταλία | 5,2 % | 17 % |
| Κύπρος | 2,9 % | 13 % |
| Λεττονία | 32,6 % | 40 % |
| Λιθουανία | 15,0 % | 23 % |
| Λουξεμβούργο | 0,9 % | 11 % |
| Ουγγαρία | 4,3 % | 13 % |
| Μάλτα | 0,0 % | 10 % |
| Κάτω Χώρες | 2,4 % | 14 % |
| Αυστρία | 23,3 % | 34 % |
| Πολωνία | 7,2 % | 15 % |
| Πορτογαλία | 20,5 % | 31 % |
| Ρουμανία | 17,8 % | 24 % |
| Σλοβενία | 16,0 % | 25 % |
| Σλοβακική Δημοκρατία | 6,7 % | 14 % |
| Φινλανδία | 28,5 % | 38 % |
| Σουηδία | 39,8 % | 49 % |
| Ηνωμένο Βασίλειο | 1,3 % | 15 % |

**Πίνακας 1-6 Συνολικοί εθνικοί στόχοι έως το 2020**

β) «ΟΔΗΓΙΑ (ΕΕ) 2018/2001»[18] ΤΟΥ ΕΥΡΩΠΑΪΚΟΥ ΚΟΙΝΟΒΟΥΛΙΟΥ ΚΑΙ ΤΟΥ ΣΥΜΒΟΥΛΙΟΥ της 11ης Δεκεμβρίου 2018, για την προώθηση της χρήσης ενέργειας από ανανεώσιμες πηγές (αναδιατύπωση). Η οδηγία αναφέρει στο άρθρο 3, παράγραφος 1: *Τα κράτη μέλη διασφαλίζουν συλλογικά ότι το μερίδιο της ενέργειας από ανανεώσιμες πηγές στην ακαθάριστη τελική κατανάλωση ενέργειας της Ένωσης ανέρχεται το 2030 σε τουλάχιστον 32 %. Η Επιτροπή αξιολογεί τον στόχο αυτόν, με σκοπό να υποβάλει, έως το 2023, νομοθετική πρόταση για την αύξηση του, αν υπάρξουν περαιτέρω σημαντικές μειώσεις κόστους στην παραγωγή ενέργειας από ανανεώσιμες πηγές, ή, όπου χρειάζεται, για την τήρηση των διεθνών δεσμεύσεων της Ένωσης για την απαλλαγή από τις εκπομπές άνθρακα ή όταν σημαντική μείωση στην κατανάλωση ενέργειας στην Ένωση δικαιολογεί τέτοια αύξηση.*

Κατά συνέπεια, η δέσμη μέτρων για την καθαρή ενέργεια και η αναδιατυπωμένη οδηγία για τις ανανεώσιμες πηγές ενέργειας (RED II)[19][20] θέτουν ως συλλογικό στόχο της Ένωσης το 32%, του μεριδίου των ΑΠΕ που πρέπει να επιτευχθεί έως το 2030. Αυτός ο στόχος της ΕΕ δεν μετατρέπεται σε εθνικούς στόχους, αλλά τα κράτη μέλη υποχρεούνται να συμβάλουν στον συλλογικό στόχο. Μια άλλη κομβική πτυχή της σύγχρονης υπερεθνικής ενεργειακής πολιτικής, είναι η φιλοδοξία να ενισχυθούν οι καταναλωτές ενέργειας και να τους ανατεθεί πιο ενεργός ρόλος στις αγορές ηλεκτρικής ενέργειας. Η αναδιατυπωμένη οδηγία για την ηλεκτρική ενέργεια περιέχει κανόνες που αποσκοπούν στην ανάλογη μεταρρύθμιση των αγορών ηλεκτρικής ενέργειας. Στο Σχήμα 1-5 βλέπουμε τα ποσοστά ΑΠΕ στην ΕΕ που προέκυψαν μέσω των ευρωπαϊκών οδηγιών.




**Σχήμα 1-5 Χρονοδιάγραμμα ποσοστών ΑΠΕ στην ΕΕ**

### 1.4.2 Καθεστώτα στήριξης ΦΒ συστημάτων σε Ευρωπαϊκές Χώρες

Για την προώθηση των ΑΠΕ, τα κράτη μέλη θα πρέπει να χρησιμοποιήσουν διάφορα μέσα, συμπεριλαμβανομένων των καθεστώτων στήριξης. Από τα διάφορα καθεστώτα στήριξης των ΑΠΕ, στην ενότητα θα δούμε και τις χώρες που επίλεξαν net metering έναντι Feed-in Tariff (FiT). Τα τιμολόγια τροφοδότησης (Feed-in Tariff - FiT) έχουν διαδραματίσει σημαντικό ρόλο στην ανάπτυξη των ΑΠΕ στην Ευρώπη. Τα τελευταία χρόνια, ορισμένες από τις χώρες της ΕΕ πέρασαν από τα FiT, στον μηχανισμό του net metering.

*Υπάρχει η πεποίθηση, ότι τα συστήματα net metering έχουν ξεφύγει από το δίχτυ του νομοθέτη της Ένωσης. Η δέσμη μέτρων για την καθαρή ενέργεια θέσπισε ένα πιο συνεκτικό και ολοκληρωμένο καθεστώς για τα καθεστώτα στήριξης των ΑΠΕ, για τους αυτοκαταναλωτές και τους ενεργούς πελάτες ανανεώσιμων πηγών ενέργειας, για τις ενεργειακές κοινότητες κ.ο.κ. Παρόλα αυτά, δεν έχουν εισαχθεί ειδικοί κανόνες ή ρητές αναφορές σε συστήματα net metering. Μια τέτοια αγνωστική στάση του νομοθέτη της ΕΕ αφήνει ανοιχτή τη συζήτηση σχετικά με την καταλληλότητα του net metering για την ενίσχυση της παραγωγής από ΑΠΕ και την προώθηση της αυτοκατανάλωσης και της κατανεμημένης παραγωγής.*[21]

*Σε αυτό το πλαίσιο, θα εξεταστεί κατά πόσον το net metering είναι συμβατό με το δίκαιο της ΕΕ. Η έμφαση δίνεται στην ερμηνεία της RED II, αλλά εξετάζονται επίσης οι σχετικές διατάξεις της αναδιατυπωμένης οδηγίας για την ηλεκτρική ενέργεια. Με την εξέταση της θέσης του net metering στο νέο νομικό πλαίσιο, το οποία έχει επικεντρωθεί κυρίως στα καθιερωμένα συστήματα στήριξης των ΑΠΕ, όπως τα σταθερά τιμολόγια ή τα πράσινα πιστοποιητικά, και έχει αφιερώσει ελάχιστη προσοχή στο net metering.*[21]

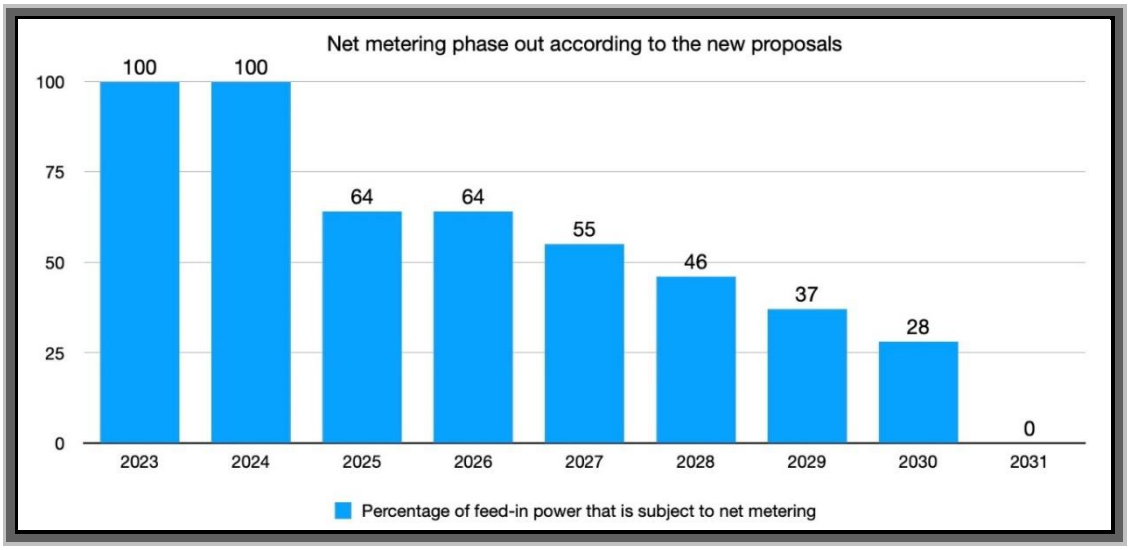
*Επιπλέον, το άρθρο επιδιώκει να παράσχει μια πρωτότυπη συγκριτική νομική ανάλυση των κυπριακών, ελληνικών, ιταλικών και βελγικών (φλαμανδικών) συστημάτων net metering. Τα κράτη αυτά είναι πρωτοπόρα στην ΕΕ στον τομέα του net metering, αλλά οι ομοιότητες και οι διαφορές των καθεστώτων που θεσπίστηκαν δεν έχουν μελετηθεί μέχρι σήμερα. Μία τέτοια μελέτη αναμένεται να καταδείξει πώς διαφέρουν οι δρόμοι που ακολουθούν τα κράτη, αλλά και να υποστηρίξει περαιτέρω την ανάγκη για ένα πιο συντονισμένο υπερεθνικό νομικό πλαίσιο, το οποίο θα διασφαλίσει ότι τα εθνικά καθεστώτα είναι κατάλληλα να εξυπηρετήσουν την υπερεθνική ενεργειακή πολιτική και ότι η ολοκλήρωση της αγοράς ηλεκτρικής ενέργειας δεν θα επιβραδυνθεί.*[21]

*Μέχρι το 2013 στην Ευρώπη, μόνο το Βέλγιο, η Κύπρος, η Δανία, η Ελλάδα, η Ιταλία και οι Ολλανδία έκαναν χρήση καθεστώτων net metering*[22]. Έκτοτε, ακολούθησαν η Φινλανδία, η Ουγγαρία, η Λετονία, η Λιθουανία, η Πορτογαλία, η Σλοβενία και η Ισπανία.

Η Δανία καθιέρωσε το net metering για ιδιωτικά ΦΒ συστήματα στα μέσα του 1998 για μια δοκιμαστική περίοδο τεσσάρων ετών. Το 2002 το σύστημα net metering παρατάθηκε για άλλα τέσσερα χρόνια μέχρι το τέλος του 2006. Το net metering αποδείχθηκε φθηνός, εύκολος στη διαχείριση και αποτελεσματικός τρόπος για την τόνωση της ανάπτυξης των ΦΒ συστημάτων στη Δανία, ωστόσο το σχετικά σύντομο χρονικό περιθώριο της ρύθμισης δεν επέτρεψε μέχρι στιγμής να αξιοποιήσει πλήρως τις δυνατότητές της. Κατά τη διάρκεια των πολιτικών διαπραγματεύσεων το φθινόπωρο του 2005 το net metering για τα ιδιωτικά ΦΒ συστήματα μονιμοποιήθηκε.[23]

Η Ιταλία προσφέρει ένα σύστημα στήριξης, το οποίο συνδυάζει το net metering και ένα καλά κατανεμημένο premium FiT.[24]

Στην Ολλανδία εφαρμόζουν net metering από το 2004. Αρχικά υπήρχε όριο 3000 kWh ετησίως. Αργότερα το όριο αυτό αυξήθηκε σε 5000 kWh. Το όριο καταργήθηκε εντελώς την 1η Ιανουαρίου 2014. Η κυβέρνηση της χώρας επιθυμεί βέβαια να καταργήσει σταδιακά το σύστημα net metering από το 2025 έως το 2031. Το νομοσχέδιο *salderingsregeling* [25], το οποίο ενέκρινε η Βουλή των Αντιπροσώπων, αναφέρει ότι οι ιδιοκτήτες ηλιακών συλλεκτών από το έτος 2025 δεν θα μπορούν πλέον να αφαιρούν από την κατανάλωσή τους την πλήρη ποσότητα της ενέργειας που τροφοδοτείται. Βέβαια θα λάβουν αποζημίωση για την ηλεκτρική ενέργεια που δεν μπορούν να αντισταθμίσουν. Μέχρι το 2027, αυτό είναι τουλάχιστον το 80% του ποσοστού παράδοσης που έχει συμφωνήσει ο ιδιοκτήτης των ηλιακών συλλεκτών με τον προμηθευτή ενέργειας, εξαιρουμένων των φόρων και των εισφορών. Ο παρών κανονισμός τίθεται σε ισχύ μία ημέρα μετά τη δημοσίευση του νόμου στην Εφημερίδα της Κυβερνήσεως. Από το 2027, ο Υπουργός θα καθορίζει την ελάχιστη αποζημίωση που πρέπει να καταβάλλουν οι προμηθευτές ενέργειας στους πελάτες τους κάθε 2 χρόνια, για ρεύμα που δεν μπορούν να συμψηφίσουν. Στο Σχήμα 1-6 παρουσιάζονται τα ποσοστά της σταδιακής κατάργησης.




**Σχήμα 1-6 Σταδιακή κατάργηση net metering στην Ολλανδία**

Η Πολωνία εισήγαγε το 2015 το net metering για ιδιωτικές και εμπορικές ΑΠΕ μέχρι 50 kW. Σύμφωνα με τη νομοθεσία αυτή, η ενέργεια που αποστέλλεται στο δίκτυο πρέπει να χρησιμοποιείται εντός ενός έτους από την τροφοδότηση, διαφορετικά θεωρείται χαμένη. Το ποσό της ενέργειας που εξήχθη και μπορεί να ληφθεί πίσω από τον χρήστη αφαιρείται κατά 20% για εγκαταστάσεις έως 10 kW ή κατά 30% για εγκαταστάσεις έως 50 kW. Η εν λόγω νομοθεσία εγγυάται ότι αυτή η πολιτική net metering θα διατηρηθεί για τουλάχιστον 15 έτη από τη στιγμή της καταγραφής της ΑΠΕ. Η νομοθεσία αυτή σε συνδυασμό με τις κυβερνητικές επιδοτήσεις για τη μικροπαραγωγή, δημιούργησε σημαντική ώθηση στις εγκαταστάσεις ΦΒ συστημάτων στην Πολωνία.[26]

Σε συνδυασμό με το National low carbon strategy, η Γαλλία έχει παρουσιάσει ένα δεκαετές πλάνο από το 2019 έως το 2028, με στόχο την αύξηση παραγωγής ενέργειας από ΑΠΕ. Συγκεκριμένα για τα ΦΒ συστήματα και σύμφωνα με το Multi Annual Energy Plan[27] το σχέδιο για τα εγκατεστημένα συστήματα είναι να ξεπεράσουν τα 20GW μέσα στο 2023, ενώ η δέσμευση για το 2028 είναι να έχουν εγκατασταθεί κάπου ανάμεσα από 35 - 44GW. Προτιμήθηκε το μοντέλο FiT από τη Γαλλική Κυβέρνηση για ενίσχυση του προγράμματος. Σύμφωνα με τον ιστότοπο της Électricité de France, τη μεγαλύτερη παραγωγό ρεύματος της Γαλλίας, η ενέργεια που παράγεται από τους ιδιοκτήτες κατοικιών αγοράζεται σε υψηλότερη τιμή από αυτήν που χρεώνεται στους καταναλωτές. Ως εκ τούτου, ορισμένοι προτείνουν να πωλείται όλη η παραγόμενη ενέργεια και να αγοράζεται πίσω όλη η ενέργεια που απαιτείται σε χαμηλότερη τιμή.

Η Ιρλανδία εφαρμόζει από το 2021 ένα σύστημα net metering, στο πλαίσιο του «Προγράμματος Στήριξης της Μικροπαραγωγής» - "Micro-generation Support Scheme" (MSS)[28], πρόγραμμα παραγωγής ενέργειας από ΑΠΕ. Σύμφωνα με το προτεινόμενο σύστημα, οι μικροπαραγωγοί μπορούν να πωλούν το 30% της πλεονάζουσας ηλεκτρικής ενέργειας που παράγουν και να την εξάγουν πίσω στο δίκτυο. Η τιμή στην οποία θα πωλείται η ηλεκτρική ενέργεια διαμορφώνεται κατά τη διάρκεια της διαδικασίας διαβούλευσης.

Η Πορτογαλία έχει μια πολύ περιορισμένη μορφή net metering, η οποία περιορίζεται σε περιόδους 15 λεπτών, όπου η περίσσεια ενέργεια που εγχέεται στο δίκτυο, δεν αποζημιώνεται όταν υπερβαίνει την κατανάλωση από το δίκτυο σε κάθε περίοδο 15 λεπτών[24]. Μόνο η ενέργεια που εγχέεται μέχρι την ενέργεια που καταναλώνεται εντός της ίδιας περιόδου 15 λεπτών συμψηφίζεται στον τελικό μηνιαίο λογαριασμό. Στην πραγματικότητα, οι παλαιοί αναλογικοί μετρητές ηλεκτρικής ενέργειας που θα επέτρεπαν την πραγματική καθαρή μέτρηση, αντικαθίστανται αμέσως όταν ένας καταναλωτής εγκαθιστά ηλιακή ΦΒ ενέργεια.

Η Σλοβενία εφαρμόζει ετήσιο net metering, από τον Ιανουάριο του 2016 για έως και 11 kW. Σε ένα ημερολογιακό έτος μπορούν να εγκατασταθούν στη χώρα έως 10 MW.[29]

Το 2010, στην Ισπανία, το net metering προτάθηκε από την Asociación de la Industria Fotovoltaica για την προώθηση της ηλεκτρικής ενέργειας από ΑΠΕ, χωρίς να απαιτείται πρόσθετη οικονομική στήριξη. Το net metering για τα ιδιωτικά συστήματα, θα εισαχθεί ως καθεστώς στήριξης και θα καθιερωθεί το 2019, μετά την αποδοχή του βασιλικού διατάγματος 244/2019 από την κυβέρνηση στις 5 Απριλίου. [24]

# 2. Ανάλυση & σχεδίαση εφαρμογής

Στο κεφάλαιο αυτό θα δοθούν ο σκοπός και οι απαιτήσεις της εφαρμογής, το κοινό στο οποίο απευθύνεται και κάποια ακόμα βασικά σημεία της Σχεδίασης. Επίσης θα γίνει σχετική σύγκριση και με ήδη υπάρχοντα εργαλείου αντίστοιχου σκοπού.

## 2.1 Σκοπός της εφαρμογής

Ο κύριος στόχος, είναι η δημιουργία ενός εργαλείου ως διαδικτυακή εφαρμογή, το οποίο θα κατατοπίζει καταναλωτές ηλεκτρικής ενέργειας και πιθανούς επενδυτές, στην αγορά των ΦΒ συστημάτων με net metering (αυτοπαραγωγή με ενεργειακό συμψηφισμό) με προαιρετική χρήση συστήματος αποθήκευσης (BESS).

Η εφαρμογή με τη χρήση ποικίλων παραμέτρων, οι οποίες θα δίνονται από τον χρήστη, θα δημιουργεί ένα βασικό τεχνοοικονομικό μοντέλο, με το οποίο ο χρήστης θα μπορεί να συγκρίνει και να αποφασίσει εάν τελικά η πιθανή επένδυση τον συμφέρει. Με βάση τα ποσά παραγόμενης - καταναλισκόμενης ενέργειας και ιδιοκατανάλωσης, θα υπολογίζονται και 2 από τα κύρια αποτελέσματα της εφαρμογής, δηλαδή το εκτιμώμενο ποσό χρημάτων, το οποίο μπορεί να εξοικονομηθεί σε ετήσια βάση (savings) και το εκτιμώμενο διάστημα απόσβεσης της επένδυσης.

## 2.2 Σχεδίαση της εφαρμογής

Η σχεδίαση ενός λογισμικού, είναι κρίσιμο βήμα, πριν την εκκίνηση της υλοποίησης μίας εφαρμογής. Ο προγραμματισμός βοηθά: α) στο σωστό επιμερισμό εργασιών, σύμφωνα με τις απαιτήσεις που θα οριστούν και β) στην εξοικονόμηση πολύτιμου χρόνου εκ των υστέρων, προβλέποντας πιθανούς κινδύνους.

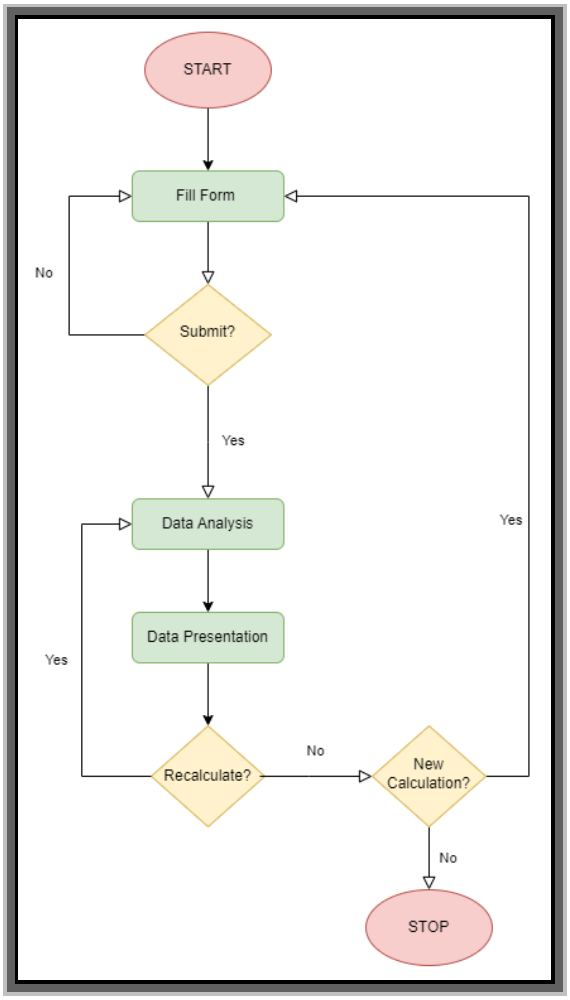
### 2.2.1 Απευθυνόμενο κοινό

Κατά τη φάση της σχεδίασης, η εφαρμογή ορίστηκε να απευθύνεται κυρίως σε καταναλωτές οικιακών δικτύων. Αυτό θα διευκόλυνε την υλοποίηση, καθώς έτσι εισάγονται λιγότερες παράμετροι και απαιτήσεις. Σε πρώτη φάση εξαιρέθηκαν περιπτώσεις μεγάλων επαγγελματικών χώρων, με μεγαλύτερες φυσικά ενεργειακές ανάγκες και απαιτήσεις σε kWp όπως και η μελέτη κοινού ΦΒ συστήματος για χρήση από πολλαπλούς

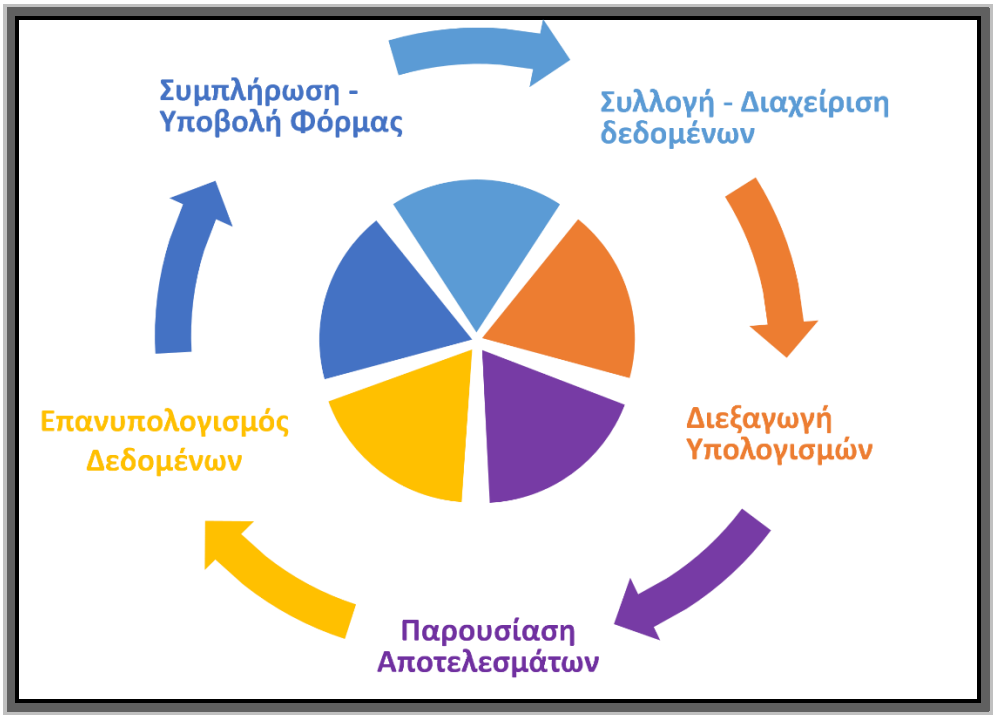
ένοικους πολυκατοικίας, το οποίο προϋποθέτει επιπλέον νομικά ζητήματα, όπως συγκατάθεση ενοίκων και διαχείρισης.

### 2.2.2 Βασική σχεδίαση

Βασικός πυλώνας της σχεδίασης, ήταν η δυνατότητα χειρισμού της εφαρμογής από ένα μέσο χρήστη διαδικτύου και υπολογιστών, έως και έναν επαγγελματία του αντικειμένου ΦΒ σε πρόγραμμα net metering. Οι δύο κύριες οθόνες της εφαρμογής θα είναι: α) μία φόρμα συμπλήρωσης δεδομένων με διαδραστικό χαρακτήρα και όνομα link ***CALCULATOR*** και β) ένα dashboard παρουσίασης των αποτελεσμάτων με όνομα link ***DASHBOARD***, τα οποία θα αναλυθούν στην επόμενη ενότητα. Στο Σχήμα 2-1 που ακολουθεί φαίνεται το διάγραμμα ροής της εφαρμογής και στην Εικόνα 2-1 η γραφική αναπαράσταση της ροής σε βήματα.




**Σχήμα 2-1 Διάγραμμα ροής (Flow chart)**




**Εικόνα 2-1 Ροή εφαρμογής**

Στην Ενότητα 2.4 θα παρατεθούν περαιτέρω πληροφορίες για τη σχεδίαση των οθονών, ***CALCULATOR*** και ***DASHBOARD***, παράλληλα με το τελικό οπτικό τους αποτέλεσμα. Συγκεκριμένα για τη δομή της φόρμας συμπλήρωσης, η φόρμα θα αποτελείται από 6 ομοίου μεγέθους πάνελ, τα οποία θα εμφανίζονται σειριακά, δηλαδή ένα τη φορά έως και την υποβολή της φόρμας. Θα πρέπει να υπάρχει το δικαίωμα κίνησης σε επόμενο ή/και προηγούμενο πάνελ και ο τίτλος κάθε πάνελ να ανταποκρίνεται στο περιεχόμενο του. Επιγραμματικά οι τίτλοι θα έχουν ως εξής:

1. Τοποθεσία
2. Εγκατάσταση
3. Αζιμούθιο – Κλίση
4. Κατανάλωση
5. Ισχύς
6. Αποθήκευση - Επιδότηση

Συνολικά η εφαρμογή θα αποτελείται από 5 οθόνες. Οι υπόλοιπες 3 οθόνες, κατά βάση ενημερωτικού περιεχομένου, θα είναι: α) η αρχική ή κεντρική οθόνη (grounding page) με όνομα link ***HOME***, ως μία πρώτη προσέγγιση σχετικά με το net metering και τα οφέλη του, β) μία οθόνη με νομοθετικά πλαίσια και οδηγίες net metering από τον ΣΕΦ, με όνομα link

***REGULATIONS*** και τέλος γ) μία οθόνη με άλλες πληροφορίες αναφορικά με τον δημιουργό και την εν λόγω εφαρμογή, με όνομα link ***ABOUT***.

Λόγω σχετικά περιορισμένης κλίμακας και έκτασης της εφαρμογής αλλά και ως πρώτης προσπάθειας δημιουργίας μίας εφαρμογής, προτιμήθηκε ο διαδικαστικός προγραμματισμός (Procedural Programming) έναντι του αντικειμενοστρεφή (Object Oriented Programming). Βασικά στοιχεία του διαδικαστικού είναι η δόμηση ενός προγράμματος με επαναλαμβανόμενη χρήση μεθόδων (Function Based views vs Class Based views), οι οποίες επιλύουν αποκλειστικά ένα τμήμα του συνολικού προβλήματος. Είναι κοινώς αποδεκτός τρόπος προσέγγισης για entry level προγραμματισμό, αλλά σε ενδεχόμενη επέκταση της εφαρμογής, θα ήταν ωφέλιμο να γίνει αναδιάταξη σημείων κώδικα σύμφωνα με τις αρχές αντικειμενοστρέφειας (OOP Principles), για βέλτιστη συντήρηση και επαναχρησιμοποίηση κώδικα. Η ανάλυση του κώδικα που υλοποιήθηκε θα γίνει στο Κεφάλαιο 3.

## 2.3 Τεχνικές απαιτήσεις εφαρμογής

Κύριο ζητούμενο είναι η ανάπτυξη λογισμικού, πρόσβαση στο οποίο θα παρέχεται μέσω διαδικτύου. Ύστερα από σχετική μελέτη για το κατάλληλο εργαλείο web development, από πλευράς διακομιστή ή εξυπηρετητή (server side) για τη Back-end υλοποίηση, επιλέχθηκε το framework *Django*[30] γραμμένο σε γλώσσα Python. Από πλευράς χρήστη ή πελάτη (client side), τη Front-end υλοποίηση αποτέλεσε το τρίπτυχο *HTML-CSS-JavaScript*. Δεν υπήρξε πρόβλεψη ανάπτυξης mobile application. Η δημιουργία της εφαρμογής πραγματοποιήθηκε στο *Visual Studio Code*, κοινώς γνωστό και ως VS Code, το οποίο είναι ένα πρόγραμμα επεξεργασίας (editor) πηγαίου κώδικα που δημιουργήθηκε από τη Microsoft με το Electron Framework, για Windows, Linux και macOS.

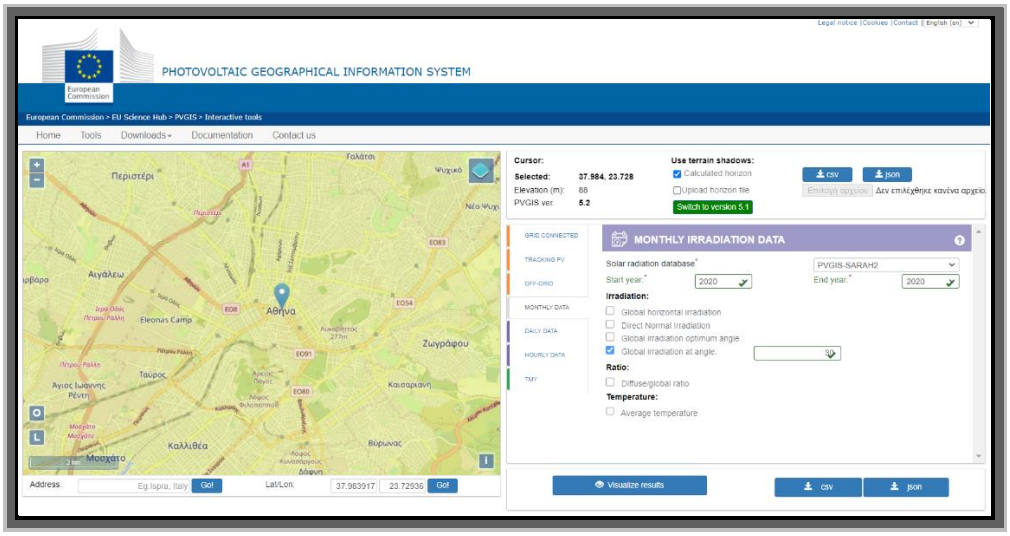
Αναφορικά με τη φόρμα συμπλήρωσης δεδομένων και συγκεκριμένα, στο 1ο πάνελ *Τοποθεσία*, όπου επιλέγεται η περιοχή εγκατάστασης από τον χρήστη, αποφασίστηκε η χρήση διαδραστικού χάρτη, η υλοποίηση του οποίου θα γίνει μέσω *Leaflet maps*[31]. Για την εύρυθμη λειτουργία της εφαρμογής και ορθό υπολογισμό αποτελεσμάτων στο 5ο πάνελ *Ισχύς*, υλοποιήθηκε επικοινωνία με τη ΒΔ του ευρωπαϊκού εργαλείου *PVGIS*[32], για τη λήψη μετρήσιμων δεδομένων ηλιακής ακτινοβολίας του έτους 2020.

### 2.3.1 Django-Python

Το Django[30] είναι ένα υψηλού επιπέδου διαδικτυακό πλαίσιο Python που επιτρέπει την ταχεία ανάπτυξη ασφαλών και συντηρήσιμων ιστότοπων. Χτισμένο από έμπειρους προγραμματιστές, το Django αναλαμβάνει μεγάλο μέρος της ανάπτυξης ιστοσελίδων, ώστε να μπορεί ο προγραμματιστής να επικεντρωθεί στη συγγραφή της εφαρμογής του, παρέχοντας του έτοιμα δομικά κομμάτια κώδικα. Είναι δωρεάν και ανοικτού κώδικα, διαθέτει μια ακμάζουσα και ενεργή κοινότητα, όπως και εξαιρετική τεκμηρίωση. Μπορεί να χρησιμοποιηθεί για την κατασκευή σχεδόν οποιουδήποτε τύπου ιστότοπου, από συστήματα διαχείρισης περιεχομένου, μέχρι κοινωνικά δίκτυα και ειδησεογραφικές ιστοσελίδες. Μπορεί να συνεργαστεί με οποιοδήποτε client side framework και μπορεί να παρέχει περιεχόμενο σε σχεδόν οποιαδήποτε μορφή. Το Django βοηθά τους προγραμματιστές να αποφύγουν πολλά κοινά λάθη ασφαλείας και επιτρέπει την προστασία από πολλές ευπάθειες, συμπεριλαμβανομένου SQL injection, cross-site scripting, cross-site request forgery and clickjacking.

### 2.3.2 PVGIS

Όσον αφορά, τα δεδομένα μηνιαίας ηλιοφάνειας (solar irradiance), για την περιοχή επιλογής του κάθε χρήστη της εφαρμογής, θα εκμεταλλευτούμε τη ΒΔ PVGIS-SARAH2, μέσω του ευρωπαϊκού εργαλείου **PVGIS**[32] (Photovoltaic Geographical Information System). Η σύνδεση της εφαρμογής γίνεται μέσω του PVGIS API, με entry point to URL: 'https://re.jrc.ec.europa.eu/api/v5_2/', για την έκδοση 5.2. Στην Εικόνα 2-2 φαίνεται το ευρωπαϊκό εργαλείο PVGIS.




**Εικόνα 2-2 Ευρωπαϊκό Εργαλείο PVGIS**

### 2.3.3 Leaflet maps

Το Leaflet[31] είναι μία βιβλιοθήκη JavaScript ανοιχτού κώδικα για διαδραστικούς χάρτες. Με μόλις 42 KB JavaScript, διαθέτει όλες τις δυνατότητες χαρτογράφησης που χρειάζονται οι περισσότεροι προγραμματιστές. Το Leaflet έχει σχεδιαστεί με γνώμονα την απλότητα, την απόδοση και τη χρηστικότητα. Λειτουργεί αποτελεσματικά σε όλες τις μεγάλες πλατφόρμες για υπολογιστές και κινητά, μπορεί να επεκταθεί με πολλά πρόσθετα, έχει ένα όμορφο, εύχρηστο και καλά τεκμηριωμένο API και έναν απλό, ευανάγνωστο πηγαίο κώδικα. Η λειτουργία του βασίζεται στη δημιουργία επιπέδων (layers) τα οποία εναποτίθενται το ένα επάνω στο άλλο, με πρώτο layer να είναι ο χάρτης ο ίδιος.

### 2.3.4 Τεχνολογία Git και πλατφόρμα GitHub

Το Git[33] είναι ένα κατανεμημένο σύστημα ελέγχου εκδόσεων (DVCS) που χρησιμοποιείται ευρέως στην ανάπτυξη λογισμικού. Με τη δημιουργία κλαδιών (branches) διευκολύνει την επεξεργασία νέων χαρακτηριστικών ή την αποσφαλμάτωση. Αυτά τα branches μπορούν να συγχωνευθούν (merge) με την κύρια βάση κώδικα (main branch) όταν ολοκληρωθεί μία εργασία. Η χρήση της τεχνολογίας αυτής έγινε μέσω του GitHub, μίας διαδικτυακής πλατφόρμας που παρέχει φιλοξενία σε αποθετήρια Git. Επιπλέον περιλαμβάνει πρόσθετα χαρακτηριστικά και εργαλεία διαχείρισης κώδικα προγραμματισμού.

Για την περίπτωση της συγκεκριμένης ανάπτυξης λογισμικού, χρησιμοποιήθηκε η συγκεκριμένη τεχνολογία, για την παρακολούθηση και τον ασφαλή έλεγχο των αλλαγών στον πηγαίο κώδικα και διευκόλυνση δοκιμών σε έτερο λειτουργικό περιβάλλον και συσκευή. Βέβαια τα συγκεκριμένα εργαλεία παρέχουν και ως δυνατότητα, τη συνεργασία με άλλους προγραμματιστές σε ποικίλα projects ή απλά απομακρυσμένο διαμοιρασμό του περιεχομένου για όποια άλλη χρήση. Αυτό γίνεται εφόσον το αποθετήριο (repository) κώδικα δηλωθεί ως κοινόχρηστο (public).

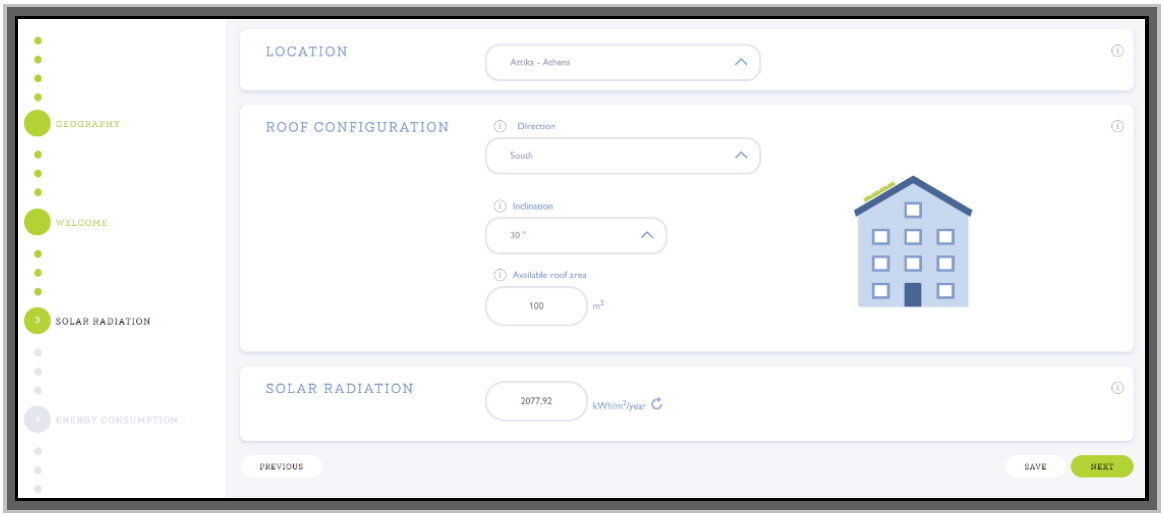
Στην Ανάλυση κώδικα του Κεφαλαίου 3, δίνεται και το link του λογαριασμού (account) και του αποθετηρίου του κώδικα της εφαρμογής, για ελεύθερη περιήγηση και χρήση.

## 2.4 Αντίστοιχες εφαρμογές

Κατά τη φάση της ανάλυσης και σχεδίασης της εφαρμογής, έγινε μία έρευνα για αντίστοιχα εργαλεία στο διαδίκτυο ώστε να αποκομιστούν τα θετικά στοιχεία κάθε εφαρμογής και να προσαρμοστούν σε μία πληρέστερη μορφή. Ένα καλαίσθητο εργαλείο βρίσκεται στην ιστοσελίδα https://value-tool.cloud.si/calculator (βλέπε Εικόνα 2-3).

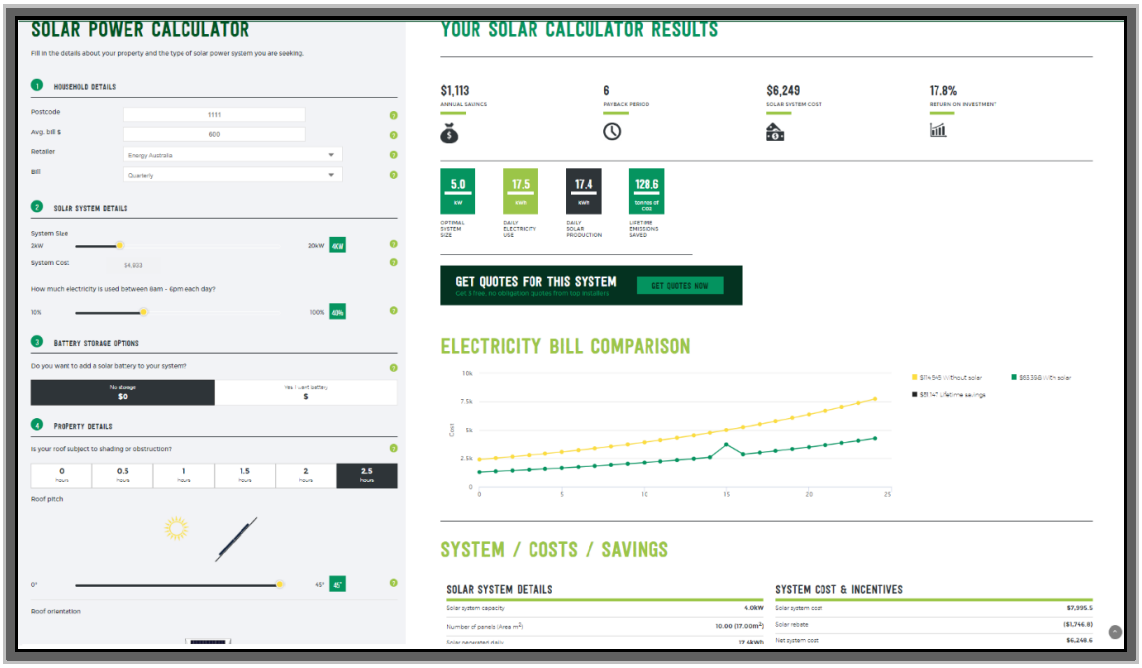

**Εικόνα 2-3 Εργαλείο value-tool, επιλογή χώρας**

Από εδώ αντλήθηκε η ανάγκη για μία πιο εξατομικευμένη επιλογή περιοχής και με βάση αυτό έγινε η υλοποίηση ενός διαδραστικού χάρτη με δυνατότητα κλικ μόνο εντός των ορίων Νομών Ελλάδας. Αυτό έγινε με γνώμονα τη διευκόλυνση του χρήστη να κάνει μία προσομοίωση σύμφωνα με τα ισχύοντα νομοθετικά κριτήρια στην Ελλάδα, περιορίζοντας τη χρήση της εφαρμογής για άλλες χώρες, καθώς σε αυτήν την περίπτωση διαφοροποιούνται και οι βέλτιστες συνθήκες για την παραγωγή ενέργειας. Παρόλα αυτά με τον χάρτη Leaflet που ενσωματώθηκε στην εφαρμογή, παραμένει η διαδραστικότητα και η δυνατότητα επιλογής οποιοδήποτε ζεύγους συντεταγμένων στην Ελληνική επικράτεια.




**Εικόνα 2-4 Εργαλείο value-tool, επιλογή στοιχείων εγκατάστασης**

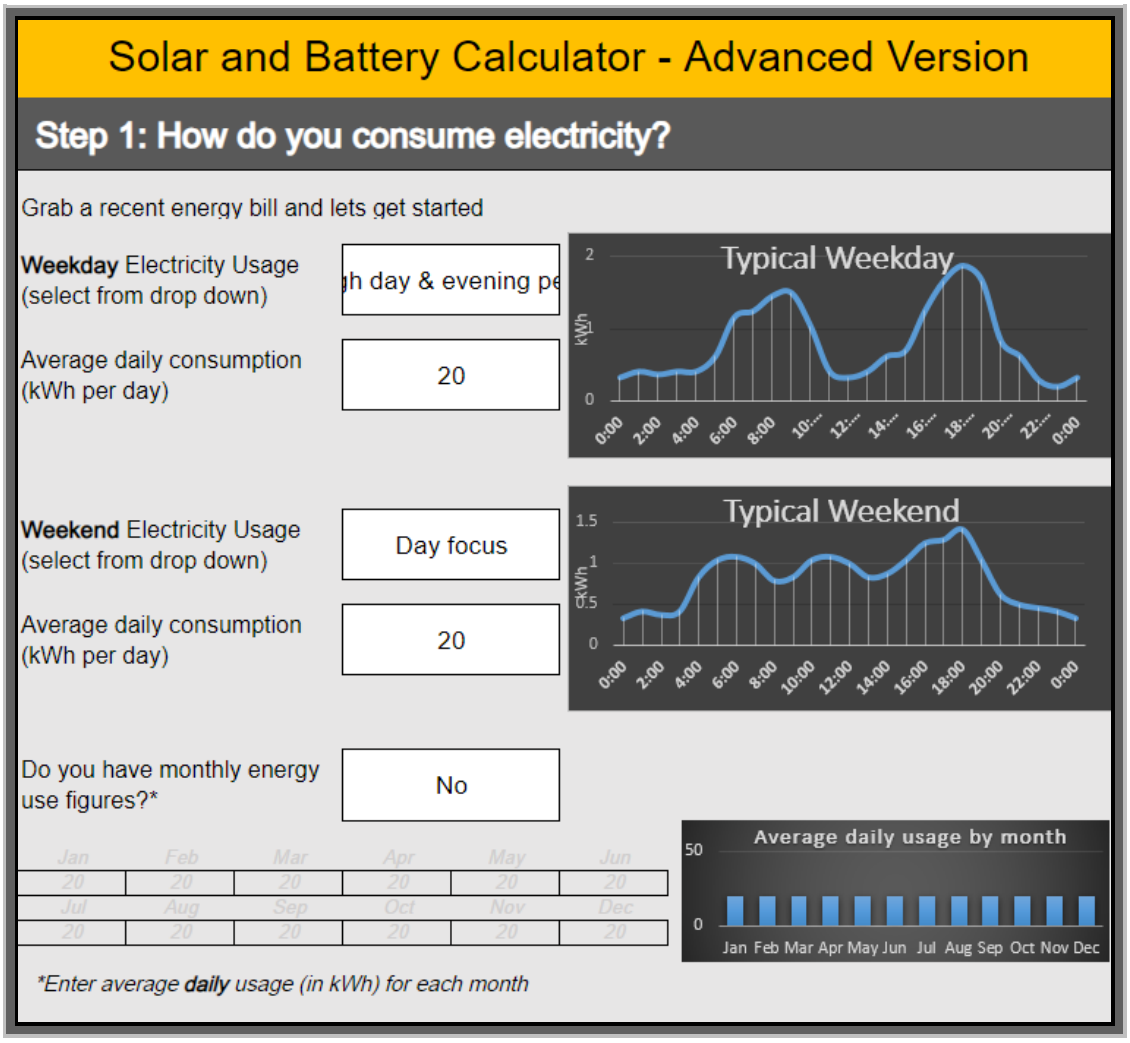
Από την προαναφερθείσα εφαρμογή (βλέπε Εικόνα 2-4) αλλά και εκείνη της ιστοσελίδας https://solarcalculator.com.au/ (βλέπε Εικόνα 2-5) θεωρήθηκε απαραίτητη η προσθήκη επιλογής αζιμούθιου και κλίσης ΦΒ συστήματος, καθώς είναι σημαντικές παράμετροι για την απόδοση και εν τέλει την παραγωγή ηλεκτρικής ενέργειας από την εγκατάσταση.




**Εικόνα 2-5 Εργαλείο solar power calculator**

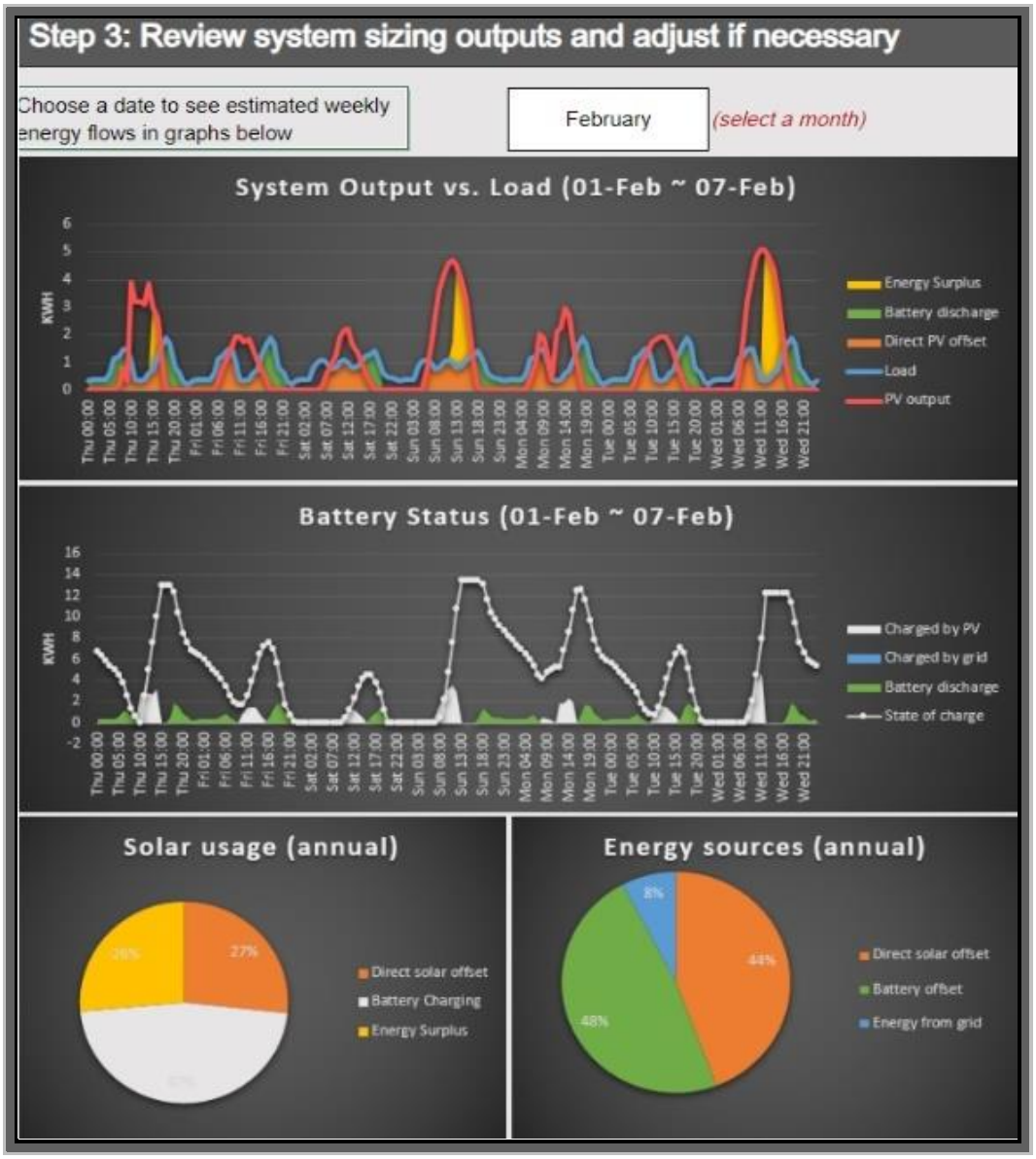
Η εφαρμογή Solar & Battery Calculator της solar choice, στην ιστοσελίδα https://www.solarchoice.net.au/blog/solar-pv-battery-storage-sizing-payback-calculator/ (βλέπε Εικόνα 2-6) επιτρέπει την εξατομίκευση του προφίλ κατανάλωσης του αυτοπαραγωγού. Όπως είπαμε αυτό επηρεάζει άμεσα το ποσοστό ιδιοκατανάλωσης, άρα

είναι σημαντικό να κατέχει μία θέση στην εφαρμογή και εν συνεχεία στον αλγόριθμο υπολογισμού του ετήσιου οφέλους του αυτοπαραγωγού, από την ένταξη του σε πρόγραμμα net metering.




**Εικόνα 2-6 Solar & Battery Calculator**

Επίσης η προσθήκη γραφημάτων με τα οικονομικά μεγέθη (βλέπε Εικόνα 2-7), θεωρήθηκε αποδεκτή προσέγγιση για την οθόνη παρουσίασης, καθώς οπτικοποιεί ιδανικότερα τα αποτελέσματα και προσθέτει επιπλέον αισθητική στην εφαρμογή.

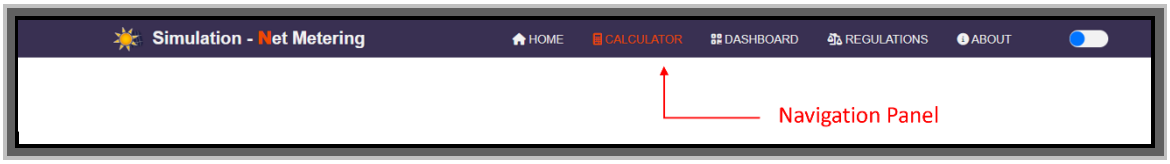



**Εικόνα 2-7 Solar & Battery Calculator, review**

## 2.5 Εμπειρία χρήστη (user experience)

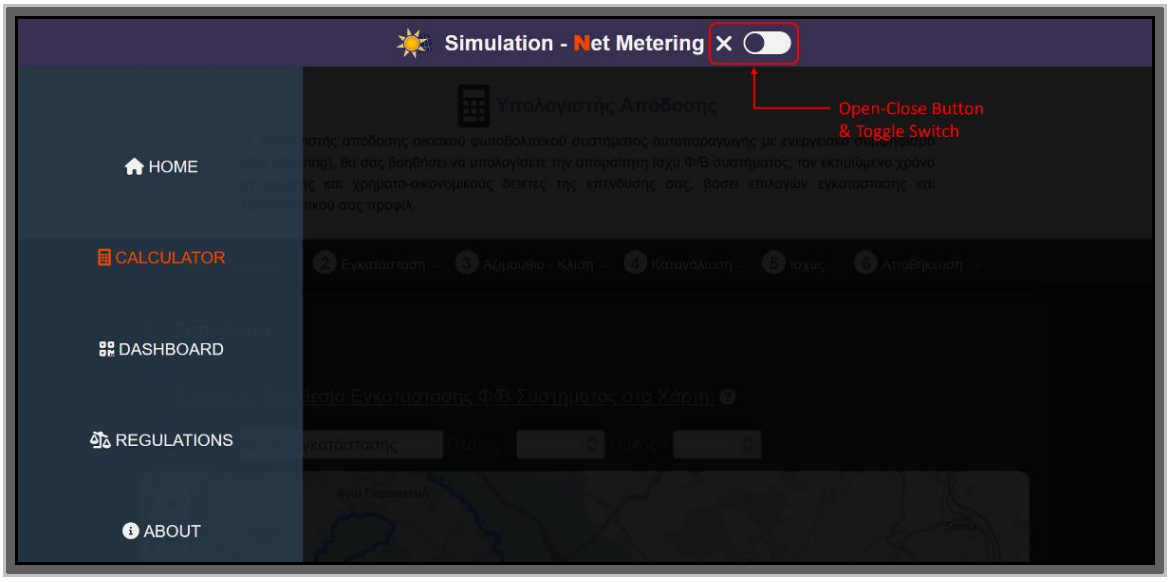
Έπρεπε να διασφαλιστεί η δημιουργία μίας ικανοποιητικής διεπαφής χρήστη, από μία σκοπιά αύξησης της συνολικής εμπειρίας του χρήστη (user experience - user interface design), το οποίο θα διευκολύνει την περιήγηση και γενικά την αλληλεπίδραση με την εφαρμογή. Όπως είπαμε στη φάση της σχεδίασης, ορίστηκε να υπάρχει πρόσβαση σε ενημερωμένες πληροφορίες, σχετικά με το net metering και το υπάρχον νομοθετικό πλαίσιο. Έτσι έπρεπε να σχεδιαστεί ένα κατάλληλο μενού πλοήγησης (Navigation Panel),

το οποίο θα επιτρέπει τη μετάβαση των χρηστών στις διάφορες οθόνες της ιστοσελίδας, όπως φαίνεται στο υλοποιημένο μενού της Εικόνας 2-8.



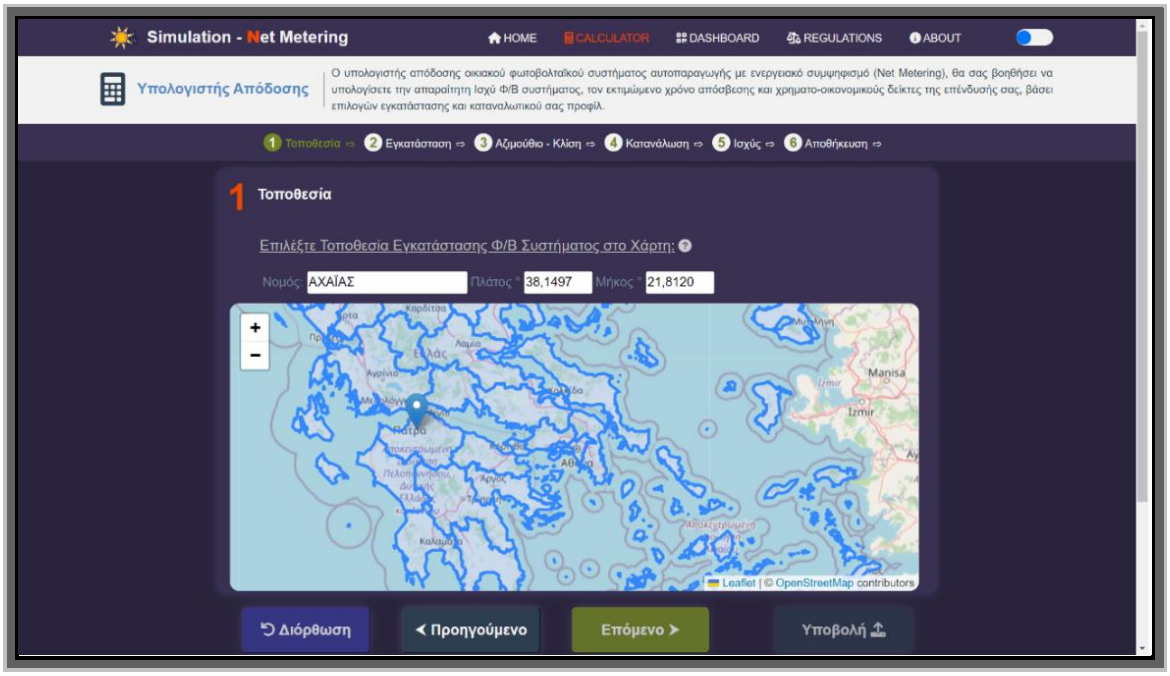

**Εικόνα 2-8 Μενού πλοήγησης**

Όταν θα πραγματοποιείται ζουμ στην οθόνη το στυλ θα προσαρμόζεται και θα παρέχει δυνατότητα χρήσης κουμπιού για εισαγωγή ή/και κλείσιμο πλαϊνής διάταξης μενού πλοήγησης στην οθόνη, όπως φαίνεται στην Εικόνα 2-9.




**Εικόνα 2-9 Μενού πλοήγησης για μικρότερες οθόνες**

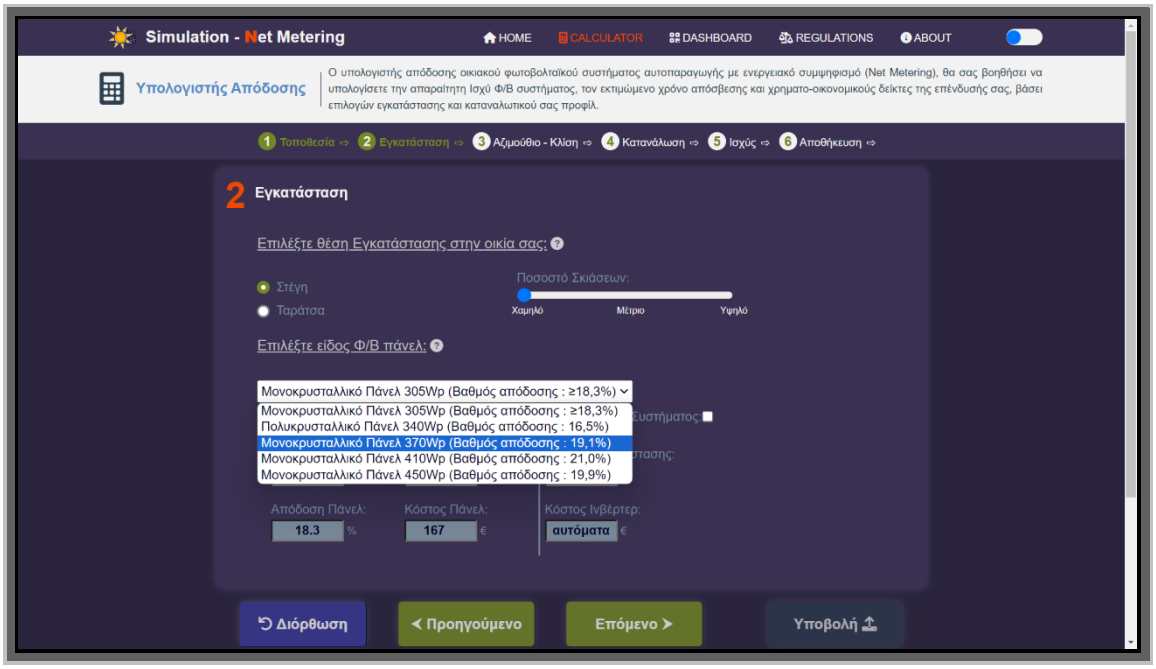
Φόρμες Django και html θα συλλέγουν τα δεδομένα που θα εισάγει ο χρήστης, θα γίνεται ο υπολογισμός και θα παρουσιάζονται τα αναμενόμενα αποτελέσματα. Στο ***CALCULATOR*** θα έχει πρόσβαση κάθε επισκέπτης της ιστοσελίδας. Στις επόμενες Εικόνες παρουσιάζεται οπτικά το κάθε βήμα από τα 6 συνολικά της φόρμας, έως και την τελική υποβολή των δεδομένων επιλογής του χρήστη. Στην Εικόνα 2-10 είναι το πρώτο βήμα. Ο χρήστης μπορεί να επιλέξει εντός του διαδραστικού χάρτη την περιοχή όπου πρόκειται να εγκατασταθεί ή είναι ήδη εγκατεστημένο κάποιο ΦΒ σύστημα. Η υλοποίηση του συγκεκριμένου χάρτη, περιορίζει το εύρος επιλογής μόνο εντός των ορίων Νομών της Ελλάδας, σύμφωνα με τα όρια που δημοσιεύτηκαν στις 16-08-2015 από τον Οργανισμό Κτηματολογίου και Χαρτογραφήσεων Ελλάδας (ΟΚΧΕ)[34].




**Εικόνα 2-10 Βήμα 1ο, Τοποθεσία**

Στα επόμενο πάνελ ο χρήστης της εφαρμογής επιλέγει κάποιες παραμέτρους για την εγκατάσταση του ΦΒ του συστήματος. Οι επιλογές είναι μεταξύ εγκατάστασης σε στέγη ή ταράτσα, με προεπιλεγμένη τη στέγη. Η διαφορά μεταξύ των δύο επιλογών καθορίζει και τα τετραγωνικά μέτρα τα οποία απαιτούνται για το ΦΒ σύστημα, καθώς στη στέγη είναι σαφώς μικρότερη η ανάγκη $m^2$ έναντι μίας ταράτσας, η οποία ως κοινόχρηστος χώρος δύναται να έχει δώμα, κεραίες τηλεόρασης ή δορυφορικά πιάτα, ηλιακούς θερμοσίφωνες κτλ. Επιπλέον μπορεί να αλλάξει και το ποσοστό σκιάσεων, το οποίο μπορεί να επηρεάσει την παραγωγή ηλεκτρικής ενέργειας του ΦΒ συστήματος (βλέπε Εικόνα 2-11).

Στη συνέχεια ο χρήστης καλείται να επιλέξει τα χαρακτηριστικά των ΦΒ πάνελ, μεταξύ αυτών το υλικό κατασκευής (Μονοκρυσταλλικό ή Πολυκρυσταλλικό), βαθμό απόδοσης, εμβαδόν σε $m^2$, ισχύ σε Wp και αξία σε € (βλέπε Εικόνα 2-11). Είναι κρίσιμα δεδομένα ώστε να υπάρχει ακρίβεια κατά τον υπολογισμό της παραγωγής ηλεκτρικής ενέργειας ανά πάνελ, καθώς και τον αριθμό των ικανών πάνελ, ώστε να ελαχιστοποιηθεί ή/και να εκμηδενιστεί η ετήσια κατανάλωση της οικιακής εγκατάστασης.




**Εικόνα 2-11 Βήμα 2ο, Εγκατάσταση**

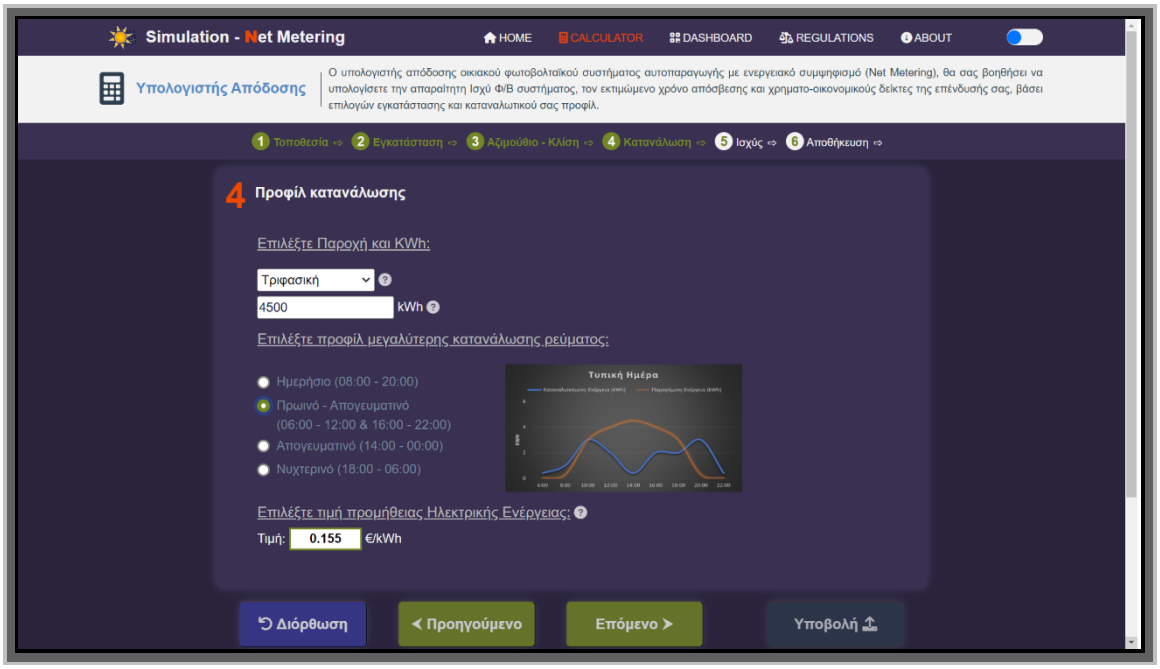
Επιπρόσθετα προβλέφθηκε και η δυνατότητα του χρήστη, να μπορεί να συμπληρώσει προσαρμοσμένες τιμές (customization) για τις παραμέτρους των ΦΒ πάνελ, όπως και για το κόστος εγκατάστασης και το κόστος αντιστροφέα (inverter). Αυτό ενεργοποιείται με επιλογή στα κουτάκια δίπλα στις λεζάντες «Προσαρμοσμένες Παράμετροι» ή «Άλλες Παράμετροι Συστήματος», όπως φαίνεται στην Εικόνα 2-12.


**Εικόνα 2-12 Βήμα 2ο, Εγκατάσταση, προσαρμοσμένες τιμές**

Τα τελευταία δεδομένα που ζητούνται για την εγκατάσταση είναι το αζιμούθιο και η κλίση του ΦΒ συστήματος στο πάνελ 3. Για την περίπτωση που δεν είναι απολύτως γνωστό το αζιμούθιο, έχει προεπιλεχθεί η βέλτιστη επιλογή για την Ελλάδα, η οποία είναι ο Νότιος προσανατολισμός. Σχετικά με την κλίση, ορίστηκε να υπάρχει συρόμενος επιλογέας (slider), με εύρος από $0^o$ έως και $90^o$ και διαδραστική εικόνα η οποία θα στρέφεται ανάλογα με τις επιλεγμένες μοίρες γωνίας. Τα βήματα αυτά φαίνονται και στην Εικόνα 2-13.

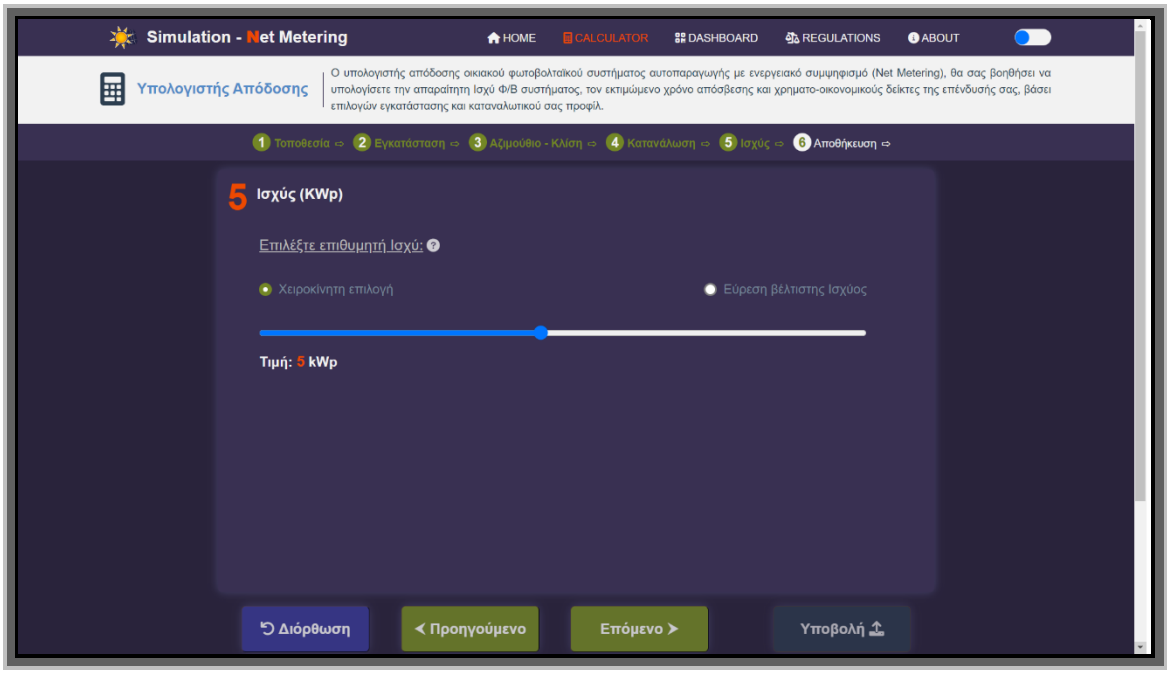

**Εικόνα 2-13 Βήμα 3ο, Αζιμούθιο - Κλίση PV**

Σημαντικά δεδομένα εξίσου είναι τα εξατομικευμένα προφίλ κατανάλωσης του εκάστοτε χρήστη. Για να επιτευχθούν όσο το δυνατόν ακριβέστερα αποτελέσματα, είναι σημαντικό να είναι γνωστή ή να μπορεί να υπολογιστεί κατά προσέγγιση και η ετήσια κατανάλωση της οικιακής εγκατάστασης, καθώς και οι ώρες εντός του 24ώρου όπου παρουσιάζονται οι μεγαλύτερες ανάγκες σε ηλεκτρικό ρεύμα. Αυτό προσαρμόζει την ισχύ και τον αριθμό των πάνελ που απαιτούνται και δίνει το ποσοστό ιδιοκατανάλωσης (self-consumption ratio) που επιτυγχάνεται ή ιδανικά θα έπρεπε να επιτευχθεί, για να επέλθει συνολικά μείωση του λογαριασμού. Ο χρήστης λοιπόν στο πάνελ 4 πρέπει να επιλέξει τη συμφωνημένη παροχή με τον πάροχο ηλεκτρικού ρεύματος, είτε εάν είναι μονοφασική (8-12kVA) είτε τριφασική (>15kVA), το προφίλ κατανάλωσης και την τιμή προμήθειας ρεύματος (βλέπε Εικόνα 2-14). Η επιλογή της παροχής, ενεργοποιεί το πεδίο συμπλήρωσης της ετήσιας κατανάλωσης καθώς και τον επιλογέα της ισχύος του επόμενου βήματος.




**Εικόνα 2-14 Βήμα 4ο, Προφίλ κατανάλωσης**

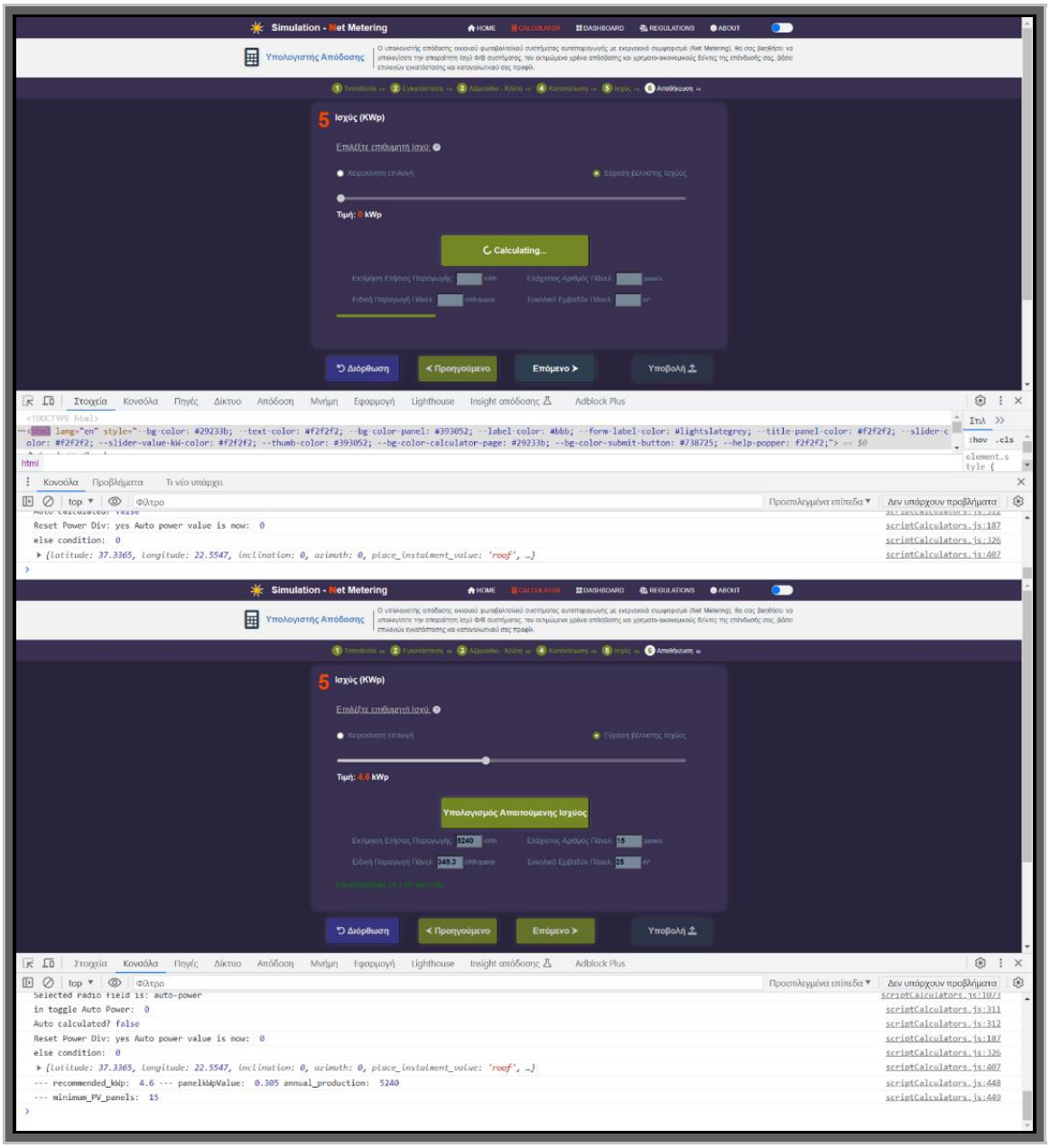
Στο πάνελ 5, δίνεται η δυνατότητα στον χρήστη να συμπληρώσει χειροκίνητα την ισχύ σε kWp του ΦΒ συστήματος που επιθυμεί εφόσον γνωρίζει, με τη «Χειροκίνητη επιλογή» (βλέπε Εικόνα 2-15), διαφορετικά μπορεί να επιλέξει την «Εύρεση βέλτιστης ισχύος» για αυτόματο υπολογισμό.




**Εικόνα 2-15 Βήμα 5ο, Ισχύς (kWp)**

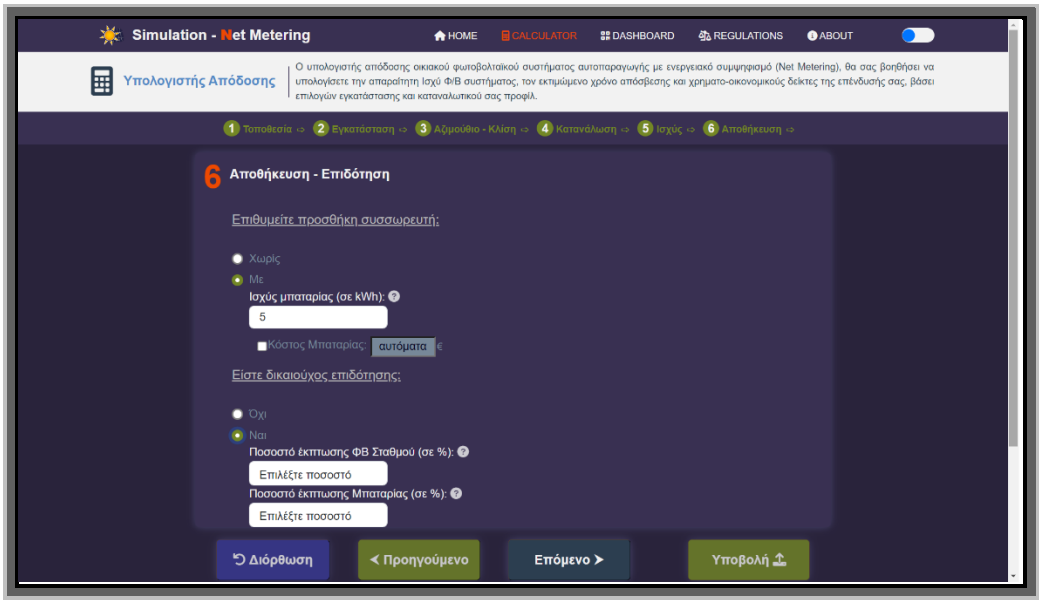
Εάν επιλέξει το δεύτερο radio button εμφανίζεται το κουμπί «Υπολογισμός Απαιτούμενης Ισχύος» και ενεργοποιώντας το, στέλνεται από πλευράς πελάτη (client side) ajax request, στην πλευρά του διακομιστή (server side). Εκεί θα γίνουν οι απαραίτητοι υπολογισμοί, που θα επιστρέψουν την ειδική παραγωγή ανά πάνελ της επιλεγμένης τοποθεσίας, τον αριθμό των ελάχιστων απαιτούμενων πάνελ και την ισχύ του ΦΒ συστήματος, στην προσπάθεια να εκμηδενιστεί η ετήσια κατανάλωση. Επίσης θα επιστραφεί και το συνολικό εμβαδόν, το οποίο απαιτείται για την εν λόγω εγκατάσταση (βλέπε Εικόνα 2-16).

Επίσης στην Εικόνα 2-16, η οποία βρίσκεται στην επόμενη σελίδα, φαίνεται το ασύγχρονο HTTP αίτημα, XMLHttpRequest και η ανταπόκρισή του. Προς επιβεβαίωση της επιτυχίας αποστολής και λήψης των υπολογισμένων τιμών, είχε ρυθμιστεί στη φάση δοκιμών, με εντολές JavaScript να εκτυπώνεται στην κονσόλα του περιηγητή, μαζί με το λεξικό δεδομένων τα οποία έχουν επιλεχθεί έως εκείνη τη στιγμή στη φόρμα και ο κωδικός 200 και τα συμπληρωμένα πεδία τιμών. Όταν θα πατηθεί η επιλογή του Υπολογισμού, το κουμπί θα πρέπει να απενεργοποιείται προσωρινά, ώστε να μην επιτρέπονται επιπλέον ανεπιθύμητα πατήματα (spam) και να εμφανίζεται μία μπάρα, ως δείκτης του βαθμού ολοκλήρωσης του αιτήματος.



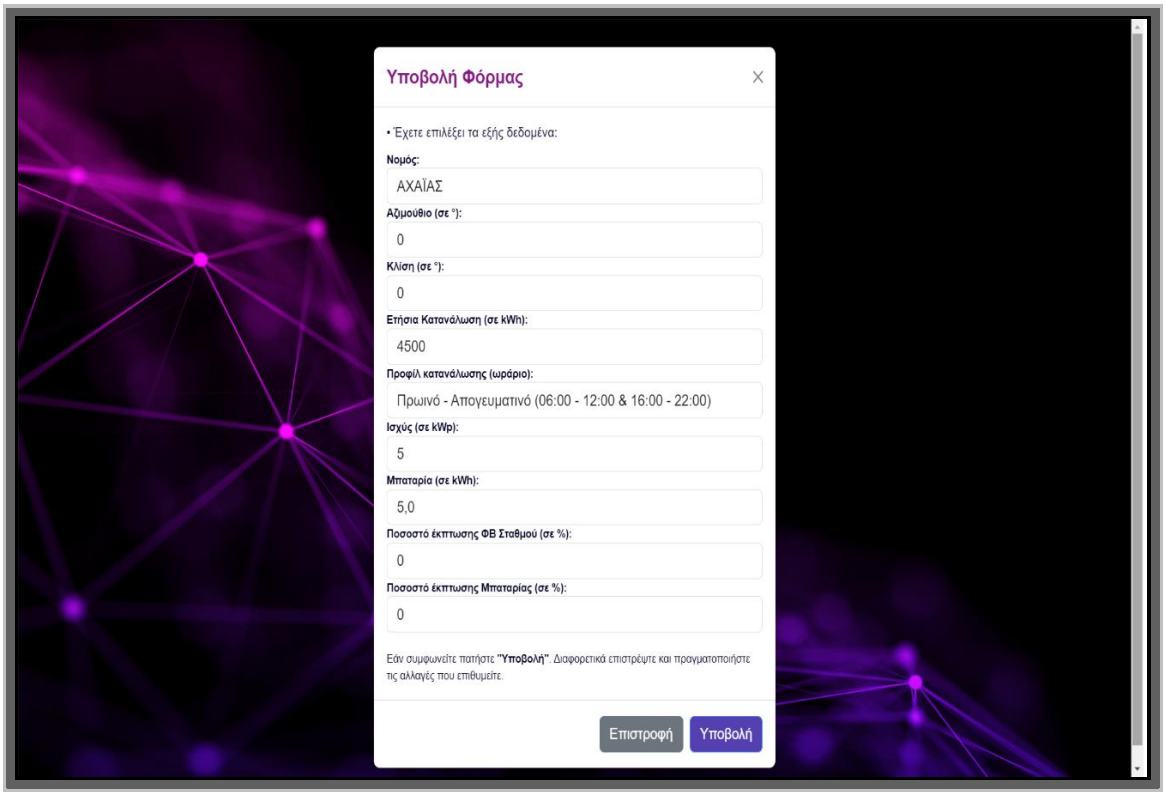

**Εικόνα 2-16 Ajax request - XMLHttpRequest**

Το τελευταίο βήμα της φόρμας, στο πάνελ 6, αποτελείται από τις προαιρετικές επιλογές Αποθήκευσης και Επιδότησης. Εδώ ο χρήστης, μπορεί να επιλέξει εφόσον επιθυμεί προσθήκη συσσωρευτών και τις αντίστοιχες kWh εντός επιτρεπτού εύρους, καθώς και εάν είναι δικαιούχος επιδότησης το αντίστοιχο ποσοστό. Εδώ να υπενθυμίσουμε ότι με το νέο πρόγραμμα «Φωτοβολταϊκά στη Στέγη» και έως το 2025, που θα είναι ενεργό το πρόγραμμα, το ποσοστό επιδότησης για προσθήκη συσσωρευτή-μπαταρίας είναι από 90% έως και 100% ανάλογα τα εισοδηματικά κριτήρια (βλέπε Εικόνα 2-17).



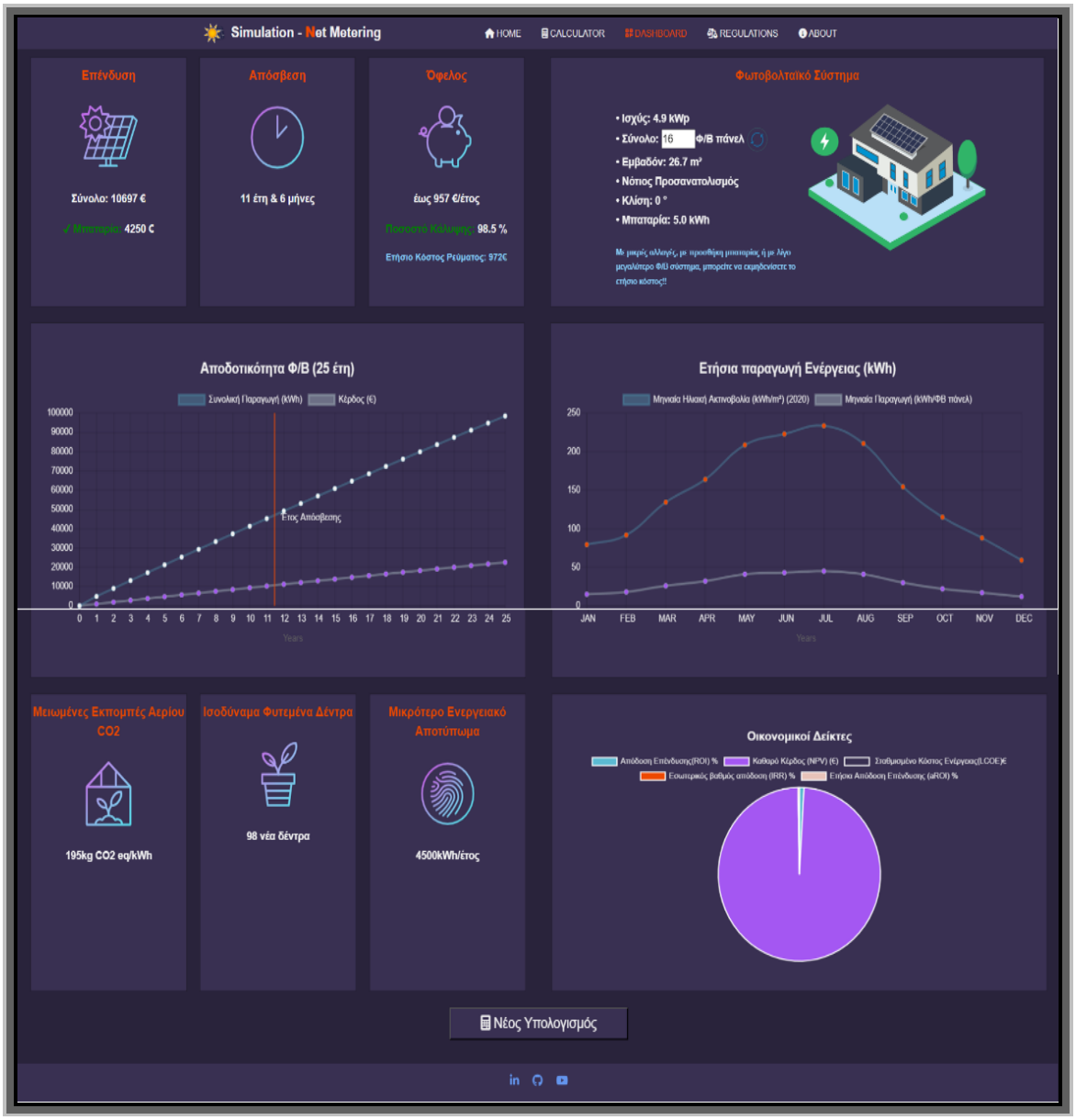

**Εικόνα 2-17 Βήμα 6ο, Αποθήκευση – Επιδότηση**

Όταν ο χρήστης θα φθάνει στο τελευταίο πάνελ θα ενεργοποιείται το κουμπί «Υποβολή» της Φόρμας, κάτι το οποίο φαίνεται εξίσου στην Εικόνα 2-17. Όταν θα επιλέγει την υποβολή ο χρήστης θα εμφανίζεται μία νέα οθόνη (βλέπε Εικόνα 2-18), όπου θα του δίνεται η δυνατότητα να αξιολογήσει τις επιλογές του, ώστε να επιτρέψει στα δεδομένα που καταχώρισε να σταλούν στο Back-end.




**Εικόνα 2-18 Οθόνη, Υποβολή Φόρμας**

Αν επιθυμεί βέβαια ο χρήστης, μπορεί να απορρίψει την ολοκλήρωση της υποβολής και να επιστρέψει πίσω στα βήματα της φόρμας και να αλλάξει όποια δεδομένα επιθυμεί. Στο Back-end λοιπόν, θα γίνονται οι απαραίτητοι υπολογισμοί συνολικού κόστους επένδυσης, κέρδους, περιόδου απόσβεσης σε έτη, με σκοπό να παρουσιαστούν στην οθόνη ***DASHBOARD*** μαζί με τα γραφήματα ηλιοφάνειας, παραγωγής ηλεκτρικής ενέργειας και οικονομικών δεικτών (βλέπε Εικόνα 2-19).




**Εικόνα 2-19 Παρουσίαση αποτελεσμάτων**

## 2.6 Οικονομικοί δείκτες

Για την πληρέστερη τελική εικόνα μίας επένδυσης και γενικά μίας επιχειρηματικής διαδικασίας, γίνεται χρήση και σύγκριση οικονομικών μεγεθών, τα οποία επιτρέπουν την ανάλυση των οικονομικών επιδόσεων και προβλέψεων. Στη συγκεκριμένη εφαρμογή γίνεται χρήση 5 κύριων οικονομικών δεικτών NPV, ROI, Annualized ROI, LCOE, IRR. Ας δούμε κάποιες πληροφορίες για αυτούς τους δείκτες και πως βοηθούν σε μία οικονομική ανάλυση.

### α) Net Present Value (NPV)

Στα ελληνικά μεταφράζεται ως Καθαρή Παρούσα Αξία (ΚΠΑ) χρησιμοποιείται για να προσδιοριστεί εάν μία επένδυση, ένα έργο ή μια επιχείρηση θα είναι κερδοφόρα ή όχι. Ουσιαστικά, η ΚΠΑ μίας επένδυσης είναι το άθροισμα όλων των μελλοντικών ταμειακών ροών κατά τη διάρκεια ζωής της επένδυσης, προεξοφλημένων στην παρούσα αξία. Ο υπολογισμός της ΚΠΑ χρησιμοποιείται συχνά σε έναν προϋπολογισμό, για να βοηθήσει στην απόφαση πώς και πού να διατεθεί ένα κεφάλαιο. Φέρνοντας κάθε επενδυτική επιλογή ή δυνητικό έργο στο ίδιο επίπεδο, πόσο δηλαδή θα αξίζει στο τέλος, είναι ένα χρήσιμο εργαλείο λήψης αποφάσεων.

Βέβαια για την περίπτωση ενός προγράμματος net metering, όπως αναφέραμε και στο Κεφάλαιο 1, επειδή ο συμψηφισμός είναι ενεργειακός, είναι προτιμότερο στη θέση των μελλοντικών ταμειακών ροών να τοποθετήσουμε το συνολικό όφελος κατά προσέγγιση, που θα έχουμε σε βάθος 25ετίας. Ύστερα αφαιρούμε τη συνολική επένδυση. Έτσι η διαφορά σε €, θα αποτελέσει την Καθαρή Παρούσα Αξία της επένδυσης:

$$\textit{Καθαρή Παρούσα Αξία (€)} = \textit{Συνολικό Όφελος} - \textit{Επένδυση} \qquad (1)$$

### β) Return On Investment (ROI)

Στα ελληνικά μεταφράζεται ως απόδοση της επένδυσης ή επιστροφή επί της επένδυσης και είναι ένα μέτρο απόδοσης που χρησιμοποιείται για την αξιολόγηση της κερδοφορίας μιας επένδυσης ή για τη σύγκριση της αποδοτικότητας ενός αριθμού διαφορετικών επενδύσεων. Μία υψηλή επιστροφή επί της επένδυσης σημαίνει πως η επένδυση κερδίζει συγκρινόμενη με το κόστος της. Η επένδυση με τη μεγαλύτερη απόδοση συνήθως παίρνει προτεραιότητα.

Για τον υπολογισμό της επιστροφής επί της επένδυσης, το καθαρό κέρδος διαιρείται με το κόστος της επένδυσης. Το αποτέλεσμα εκφράζεται ως ποσοστό ή αναλογία:

$$\textbf{\textit{Επιστροφή επί της επένδυσης (\%)}} = \left(\frac{\textbf{\textit{Καθαρό Κέρδος}}}{\textbf{\textit{Επένδυση}}}\right) * \textbf{\textit{100}} \qquad (2)$$

Ένας ασφαλής τρόπος μετατροπής της φόρμουλας για να αντιμετωπίσει ιδανικότερα τις συνθήκες ενός προγράμματος net metering, είναι να αντικαταστήσουμε το Καθαρό Κέρδος με την ΚΠΑ και να διαιρέσουμε με τη συνολική επένδυση. Πολλαπλασιάζουμε με 100 για να πάρουμε το ποσοστό %:

$$\textbf{\textit{Επιστροφή επί της επένδυσης (\%)}} = \left(\frac{\textbf{\textit{Καθαρή Παρούσα Αξία}}}{\textbf{\textit{Επένδυση}}}\right) * \textbf{\textit{100}} \qquad (3)$$

### γ) Annualized Return On Investment (AROI)

Στα ελληνικά μεταφράζεται ως ετησιοποιημένη απόδοση. Αυτή η μορφή επιστροφή επί της επένδυσης, λαμβάνει υπόψη τη χρονική διάρκεια κατά την οποία ένας ενδιαφερόμενος έχει την επένδυση. Ακολουθεί ο τρόπος υπολογισμού της ετησιοποιημένης απόδοσης:

$$\textbf{\textit{Ετησιοποιημένη απόδοση (\%)}} = \left[ \left(\frac{\textbf{\textit{Συνολικό Όφελος}}}{\textbf{\textit{Επένδυση}}}\right)^{\frac{1}{25}} - 1 \right] * 100 \qquad (4)$$

Η ύψωση του κλάσματος στη δύναμη του (1 / 25) είναι η σωστή μαθηματική πράξη για τον υπολογισμό του μέσου ετήσιου ρυθμού ανάπτυξης, κατανεμημένο ομοιόμορφα στο χρονικό διάστημα των 25 ετών. Αυτή η εκθετικοποίηση προσαρμόζει τον λόγο της συνολικής αποταμίευσης προς τη συνολική επένδυση, ώστε να ληφθεί υπόψη το αποτέλεσμα του ανατοκισμού για 25 χρόνια. Αφαιρώντας το 1 από τον συντελεστή ανάπτυξης, μετατρέπουμε τη δεκαδική τιμή σε ποσοστό.

### δ) Levelized Cost Of Energy (LCOE)

Στα ελληνικά μεταφράζεται ως ισοσταθμισμένο κόστος ηλεκτρικής ενέργειας και είναι ένας δείκτης, του μέσου καθαρού παρόντος κόστους παραγωγής ηλεκτρικής ενέργειας για μια γεννήτρια ή για ένα ΦΒ σύστημα στην περίπτωση μας, κατά τη διάρκεια της ζωής τους. Χρησιμοποιείται για τον προγραμματισμό επενδύσεων και για τη σύγκριση διαφορετικών μεθόδων παραγωγής ηλεκτρικής ενέργειας σε σταθερή βάση. Ο γενικότερος όρος ισοσταθμισμένο κόστος ενέργειας, μπορεί να περιλαμβάνει το κόστος είτε της ηλεκτρικής ενέργειας είτε της θερμότητας.

Το ισοσταθμισμένο κόστος ηλεκτρικής ενέργειας, αντιπροσωπεύει το μέσο έσοδο ανά μονάδα παραγόμενης ηλεκτρικής ενέργειας που θα απαιτούνταν για την ανάκτηση του κόστους κατασκευής και λειτουργίας μιας μονάδας παραγωγής, κατά τη διάρκεια μιας υποτιθέμενης οικονομικής ζωής και ενός υποτιθέμενου κύκλου λειτουργίας. Το ισοσταθμισμένο κόστος ηλεκτρικής ενέργειας, υπολογίζεται ως ο λόγος μεταξύ όλων των προεξοφλημένων δαπανών κατά τη διάρκεια της ζωής μιας μονάδας παραγωγής ηλεκτρικής ενέργειας, διαιρούμενο με ένα προεξοφλημένο άθροισμα των πραγματικών ποσοτήτων ενέργειας που παραδίδονται/παράγονται. Όσο μικρότερο το ισοσταθμισμένο κόστος, τόσο καλύτερη η επένδυση.

Η φόρμουλα δηλαδή θα είναι:

$$\textbf{\textit{Ισοσταθμισμένο κόστος ηλ. ενέργειας (€/kWh)}} = \left(\frac{\textbf{\textit{Επένδυση}}+\textbf{\textit{Συνολικά Έξοδα}}\ (€)}{\textbf{\textit{Συνολική Παραγωγή}}\ (\textbf{\textit{kWh}})}\right) \qquad (5)$$

### ε) Internal Rate of Return (IRR)

Στα ελληνικά μεταφράζεται ως ο εσωτερικός συντελεστής απόδοσης. Μπορεί να υπολογιστεί όταν η ΚΠΑ όλων των ταμειακών ροών (θετικών και αρνητικών) από την επένδυση θεωρείται ότι είναι ίση με μηδέν. Ισοδύναμα, είναι το προεξοφλητικό επιτόκιο με το οποίο η ΚΠΑ των μελλοντικών ταμειακών ροών είναι ίση με την αρχική επένδυση και είναι επίσης το επιτόκιο με το οποίο η συνολική παρούσα αξία του κόστους (αρνητικές ταμειακές ροές) ισούται με τη συνολική παρούσα αξία των οφελών (θετικές ταμειακές ροές).

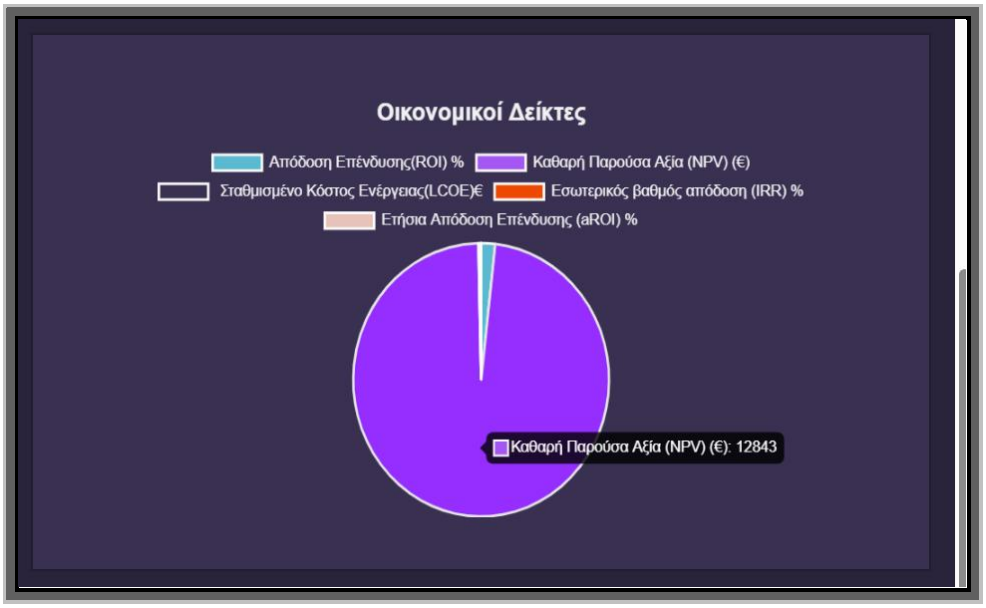
Γίνεται αντιληπτό ότι ο εσωτερικός συντελεστής απόδοσης και η ΚΠΑ είναι δύο δείκτες που δύναται να σχετιστούν άμεσα, προσφέρουν όμως διαφορετικές οπτικές στην ανάλυση επενδύσεων. Είναι επίσης ξεκάθαρο ότι όσο μεγαλύτερος ο συντελεστής απόδοσης, τόσο μεγαλύτερο περιθώριο κέρδους. Ο τυπικός τύπος εύρεσης του συντελεστή απόδοσης είναι:

$$0 = \text{ΚΠΑ} = \sum_{n=0}^{N} \frac{(Ετήσιο\ Όφελος)_n}{(1+IRR)^n} \qquad (6)$$

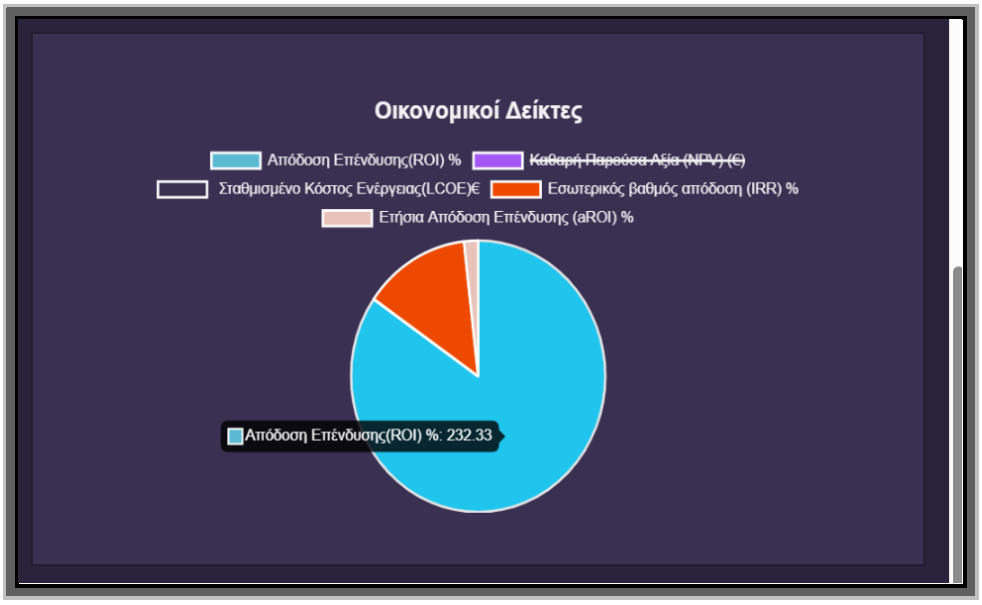
Επομένως καθώς η «ζωή» ενός ΦΒ συστήματος αντιστοιχεί σε περίπου μία 25ετία, θα υπολογίσουμε κατά προσέγγιση το συνολικό όφελος, ή πιο απλά τη μείωση του λογαριασμού που μπορεί να μας προσφέρει ένα τέτοιο σύστημα, μέσω ενός προγράμματος net metering. Για μεγαλύτερη ακρίβεια θα πρέπει να συνυπολογιστεί και ο συντελεστής υποβάθμισης ενός ΦΒ πάνελ, ο οποίος είναι ετήσιος και αντιστοιχεί περίπου στο 0,5%

ανάλογα και με το εγχειρίδιο του κατασκευαστή. Αυτός ο συντελεστής αναφέρεται στην ετήσια μείωση της ισχύος ή της απόδοσης του συστήματος, με αποτέλεσμα λίγο μικρότερη παραγωγή ενέργειας ανά έτος.

Στην Εικόνα 2-20 φαίνονται οι δείκτες εντός του διαγράμματος τύπου πίτα. Επειδή οι δείκτες έχουν μεγάλο εύρος τιμών και δεν είναι ισοδύναμα μεγέθη, μεγαλύτερες τιμές όπως αυτή της ΚΠΑ μπορεί να υπερκαλύψει τους υπόλοιπους δείκτες. Ο χρήστης μπορεί να πατήσει πάνω σε κάποιον δείκτη και να τον διαγράψει προσωρινά, δίνοντας την ευκαιρία στους υπόλοιπους να αναδειχθούν, όπως φαίνεται στην Εικόνα 2-21.




**Εικόνα 2-20 Διάγραμμα οικονομικών δεικτών, ΚΠΑ**




**Εικόνα 2-21 Διάγραμμα οικονομικών δεικτών, επόμενοι δείκτες**

# 3. Ανάλυση κώδικα

Στο κεφάλαιο αυτό θα γίνει μία ανάλυση για τα σημαντικότερα και πιο χαρακτηριστικά κομμάτια του κώδικα, ώστε να ολοκληρωθεί η εφαρμογή *simulation net metering*. Τα σημεία αυτά για τα οποία θα γίνει αναφορά, θα παρατεθούν στο Παράρτημα Β στο τέλος της εργασίας, με τη σειρά την οποία παρουσιάζονται στο κεφάλαιο αυτό. Ο πλήρης κώδικας της εφαρμογής, μπορεί να αναζητηθεί στον κοινόχρηστο φάκελο που βρίσκεται στο link: https://github.com/anegian/Simulation_Net_Metering_Django και συγκεκριμένα στο main branch.

## 3.1 Αρχείο index.html

Είναι το βασικό αρχείο .html της εφαρμογής, πάνω στο οποίο κάνουν extend με χρήση του Django tag *{% extends "index.html" %},* τα υπόλοιπα πέντε (05) *html* αρχεία, *home.html, calculator.html, dashboard.html, regulations.html, about.html*. Επίσης στο <head> βρίσκεται ο κώδικας ο οποίος καλεί τα αρχεία της CSS και τα scripts της JavaScript, υπεύθυνα για το αισθητικό και διαδραστικό(λογικό) κομμάτι της εφαρμογής. Από τα σημαντικά blocks του κώδικα είναι το <header> της εφαρμογής, με το λογότυπο και τον τίτλο, το Navigation bar με τις 5 δυνατές επιλογές πλοήγησης, ως unordered list (ul) και το <footer> στο τέλος με τα στοιχεία του δημιουργού και links από άλλες εφαρμογές, όπου διατηρούνται λογαριασμοί (accounts).

## 3.2 Αρχείο leaflet_map.js

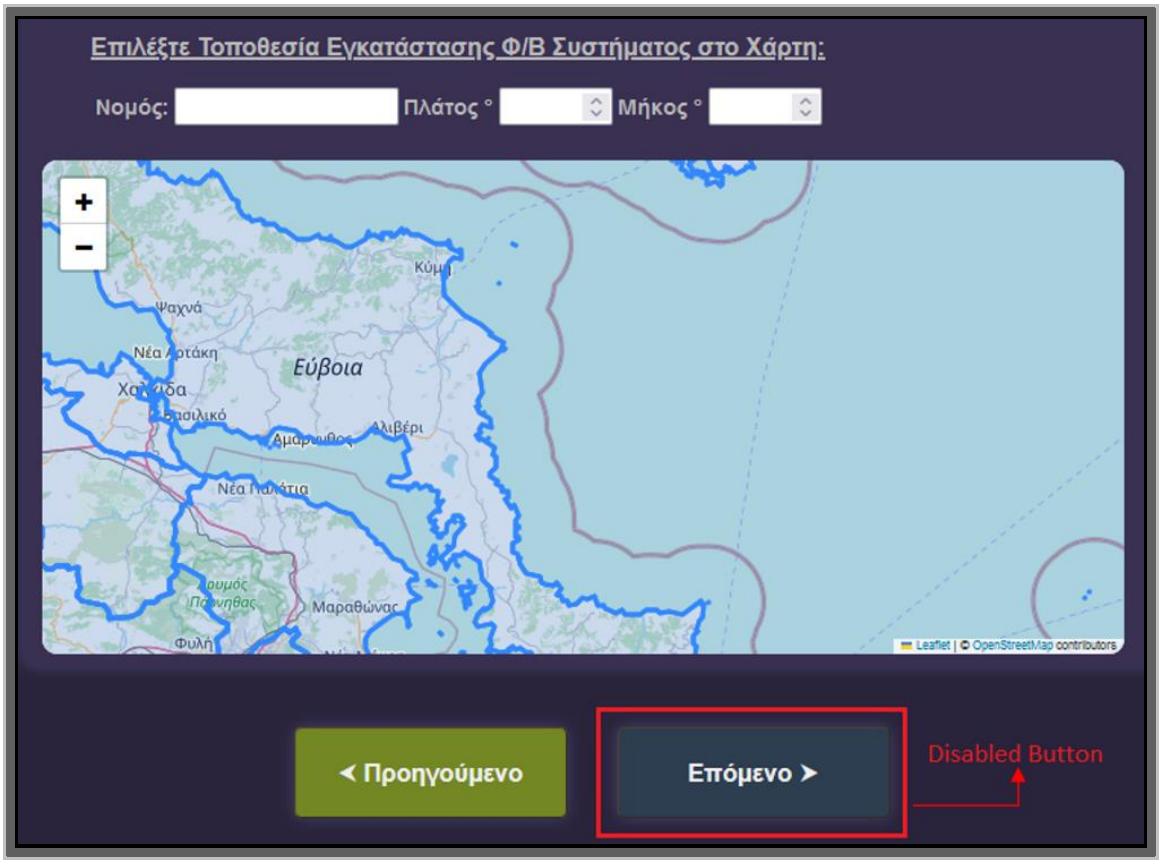
Αυτό το αρχείο κώδικα σε γλώσσα JavaScript, υλοποιεί τη δημιουργία επιπέδων (layers), τα οποία εμφανίζονται επάνω στον χάρτη της open source JavaScript βιβλιοθήκης Leaflet. Το πρώτο βήμα είναι η δημιουργία της μεταβλητής χάρτη με την εντολή:
*let map = L.map('map').setView([37.983917, 23.72936], 7);* η οποία δημιουργεί ένα instance του χάρτη, μέσα στο επιθυμητό  της html με id=*'map'* και συγκεκριμένα κάνει ζουμ με επίπεδο 7, στις συντεταγμένες οι οποίες αντιστοιχούν στην Αθήνα.

Το πρώτο επίπεδο είναι το default layer και σύμφωνα με το documentation της Leaflet με τη χρήση του element L.tileLayer(…), το οποίο εμφανίζει τον contributor. Το δεύτερο και σημαντικότερο επίπεδο για τη λειτουργικότητα της εφαρμογής είναι τα όρια Νομών της

Ελλάδας με μπλε συνεχή γραμμή, σύμφωνα με το αρχείο *Prefectures_boundaries.geojson*[34], το οποίο βρίσκεται αποθηκευμένο τοπικά στον φάκελο της εφαρμογής. Μέσω του Fetch API της JS, η global μέθοδος fetch() αντλεί το περιεχόμενο του αρχείου geojson διαβάζοντας τη διαδρομή *let geojsonPath ="{% static 'Net_Metering/json/Prefectures_ boundaries. geojson' %}"*. Η διαδρομή αυτή ορίζεται ως μεταβλητή στο *calculator.html* σε <script> tag. Τα όρια λοιπόν προστίθενται στον αρχικό χάρτη. Το τρίτο και τελευταίο layer είναι ένας δείκτης (marker), ο οποίος εμφανίζεται στις προεπιλεγμένες (default) συντεταγμένες, όταν φορτώνεται ο χάρτης, αλλά στη συνέχεια μετακινείται επάνω στο σημείο του χάρτη που πατάει ο χρήστης, εφόσον είναι εντός των ορίων των Νομών.

Το κυριότερο κομμάτι της λογικής υλοποιείται στη μέθοδο function *onMapClick(e)*, η οποία καλείται στο τέλος του block της fetch(). Η συγκεκριμένη μέθοδος είναι υπεύθυνη να αλλάξει τις συντεταγμένες και τον δείκτη, ανάλογα με το σημείο στο οποίο θα κλικάρει ο χρήστης. Αυτό που εκτελείται αρχικά, είναι ένας έλεγχος εάν έχει διαβαστεί το αρχείο geojson και είναι σωστός ο τύπος των γεωμετρικών χαρακτηριστικών εντός αυτού. Το συγκεκριμένο αρχείο έχει αντικείμενα με τύπο γεωμετρίας *Multipolygon*, για αυτό μπαίνει στο statement της if. Οπότε σε τελική φάση, γίνεται έλεγχος εάν ο κέρσορας βρίσκεται εντός των σημείων που αντιστοιχούν στα όρια νομών της Ελλάδας. Αν ναι, θεωρούνται έγκυρα. Σε αντίθετη περίπτωση, ο marker αφαιρείται ***marker.remove()*** και οι συντεταγμένες μετατρέπονται σε άδειες συμβολοσειρές (strings).

Σημαντικά κομμάτια του κώδικα είναι οι 3 μέθοδοι ***triggerButtonEnable()*** και ***triggerButtonDisable()*** και ***isPointInsidePolygon(lng, lat, polygon)***. Οι πρώτες δύο καλούνται από το αρχείο *scriptCalculator.js* και ουσιαστικά επεμβαίνουν με το να ενεργοποιούν ή να απενεργοποιούν το κουμπί επιλογής «Επόμενο» (βλέπε Εικόνα 3-1), ως δικλείδα ασφαλείας ότι ο χρήστης έχει επιλέξει σωστό σημείο στον χάρτη και αυτό εμφανίστηκε κανονικά στα αντίστοιχα πεδία. Η τελευταία μέθοδος, κάνοντας χρήση της open source JavaScript βιβλιοθήκης Turf.js, υπολογίζει εάν το σημείο επιλογής του χρήστη βρίσκεται εντός πολυγώνων του geojson, άρα θεωρούνται έγκυρα (valid).




**Εικόνα 3-1 Ανενεργό κουμπί Επόμενο**

## 3.3 Υπολογισμός ηλιακής ακτινοβολίας

Εντός του *views.py* της εφαρμογής δημιουργήθηκε μέθοδος, η οποία επικοινωνεί με τη διαδικτυακή εφαρμογή PVGIS. Η ζεύξη γίνεται μέσω του API entry point. Ως παράμετροι της μεθόδου δίνονται τα ***latitude_value***, ***longitude_value***, ***inclination_value***, ***azimuth_value***. Οι μεταβλητές αυτές αντιστοιχούν στα δεδομένα τα οποία θα επιλέξει ο χρήστης. Το σημείο επιλογής επάνω στον χάρτη (πάνελ 1) θα δώσει στο back-end τις συντεταγμένες latitude - longitude και σε επόμενο βήμα, δηλαδή στο πάνελ 3 θα οριστούν το αζιμούθιο και η κλίση του ΦΒ συστήματος.

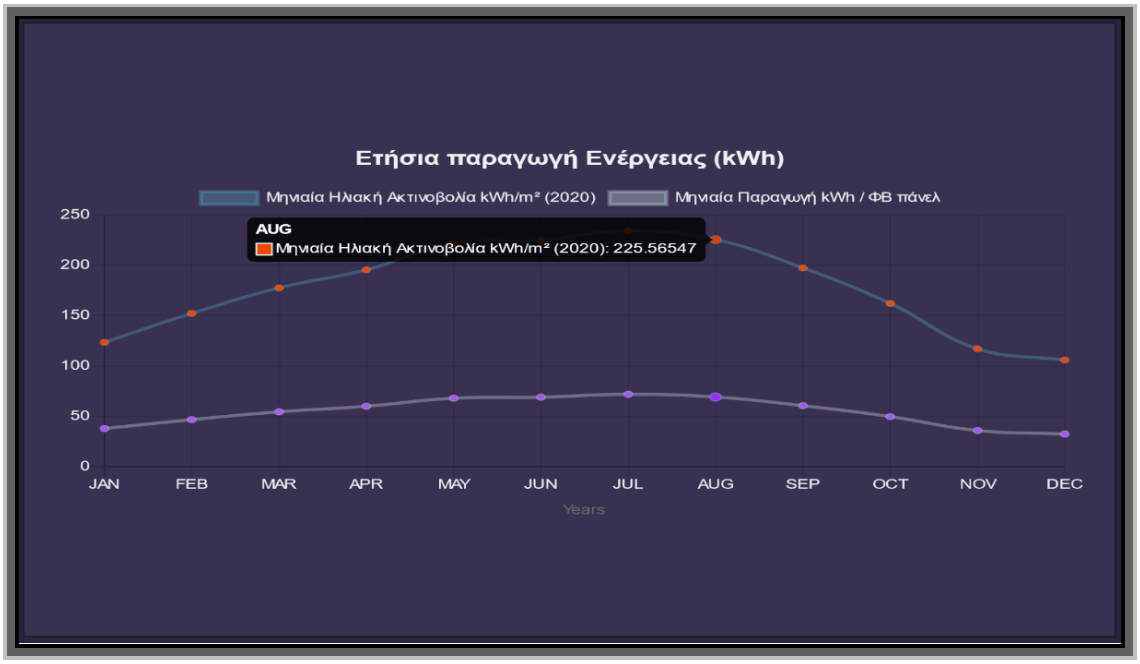
Για να επιτευχθεί η επικοινωνία θα γίνει χρήση της μεθόδου ***.get_pvgis_hourly()*** από τη βιβλιοθήκη εργαλείων της Python ***pvlib.iotools***. Μέσω API θα σταλούν οι 4 αυτές βασικές παράμετροι στο PVGIS και θα γίνει αναζήτηση στη ΒΔ του. Τα δεδομένα ηλιοφάνειας, θα εξαχθούν σε αρχείο μορφής json και αναλυτικά για κάθε ώρα της ημέρας για το επιλεγμένο έτος 2020, το οποίο ορίστηκε στη μέθοδο. Με τη μέθοδο ***resample()*** θα συγκεντρωθούν τα δεδομένα ωριαίας χρονοσειράς *{"time": "20200104:0309", "G(i)": 0.0, "H_sun": 0.0,*

*"T2m": 5.9, "WS10m": 2.9, "Int": 0.0}* και θα μετατραπούν σε μηνιαία δεδομένα. Η επεξήγηση των αρχικών, των δεδομένων δίνεται στον Κώδικα 3-1 παρακάτω. Τέλος τα δεδομένα αυτά θα μετατραπούν σε λίστα και εν συνεχεία σε json αρχείο ***monthly_irradiance_json = json.dumps(monthly_irradiance_list)***, για να το επεξεργαστούν η JavaScript και η Chart.js και να δημιουργηθεί το γράφημα με τη Μηνιαία Ηλιακή Ακτινοβολία.

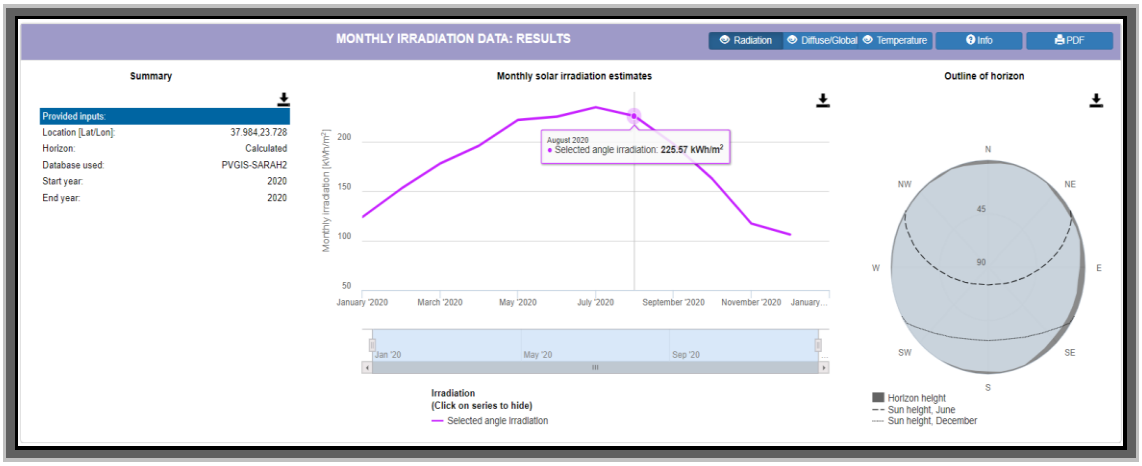
```
# Dictionary mapping PVGIS names to pvlib names
VARIABLE_MAP = {
    'G(h)': 'ghi',
    'Gb(n)': 'dni',
    'Gd(h)': 'dhi',
    'G(i)': 'poa_global',
    'Gb(i)': 'poa_direct',
    'Gd(i)': 'poa_sky_diffuse',
    'Gr(i)': 'poa_ground_diffuse',
    'H_sun': 'solar_elevation',
    'T2m': 'temp_air',
    'RH': 'relative_humidity',
    'SP': 'pressure',
    'WS10m': 'wind_speed',
    'WD10m': 'wind_direction',
}
```

**Κώδικας 3-1 Επεξήγηση λεξικού ονομάτων PVGIS**

Για να γίνει μία επιβεβαίωση ότι η επικοινωνία υλοποιείται ορθά και τα δεδομένα έχουν εξαχθεί σωστά, παρατίθενται οι παρακάτω φωτογραφίες με την ηλιακή ακτινοβολία ανά μήνα για το έτος 2020, για τις συντεταγμένες 37.984, 23.728 οι οποίες αντιστοιχούν στην πλατεία Ομονοίας στην Αθήνα. Ενδεικτικά για τον μήνα Αύγουστο και στα δύο γραφήματα η ακτινοβολία είναι 225,56547 kWh/m$^2$. Στις Εικόνες 3-2 και 3-3 γίνεται αντιληπτό πως οι δύο αυτές τιμές ταυτίζονται.




**Εικόνα 3-2 Γράφημα ετήσιας παραγωγής ενέργειας**

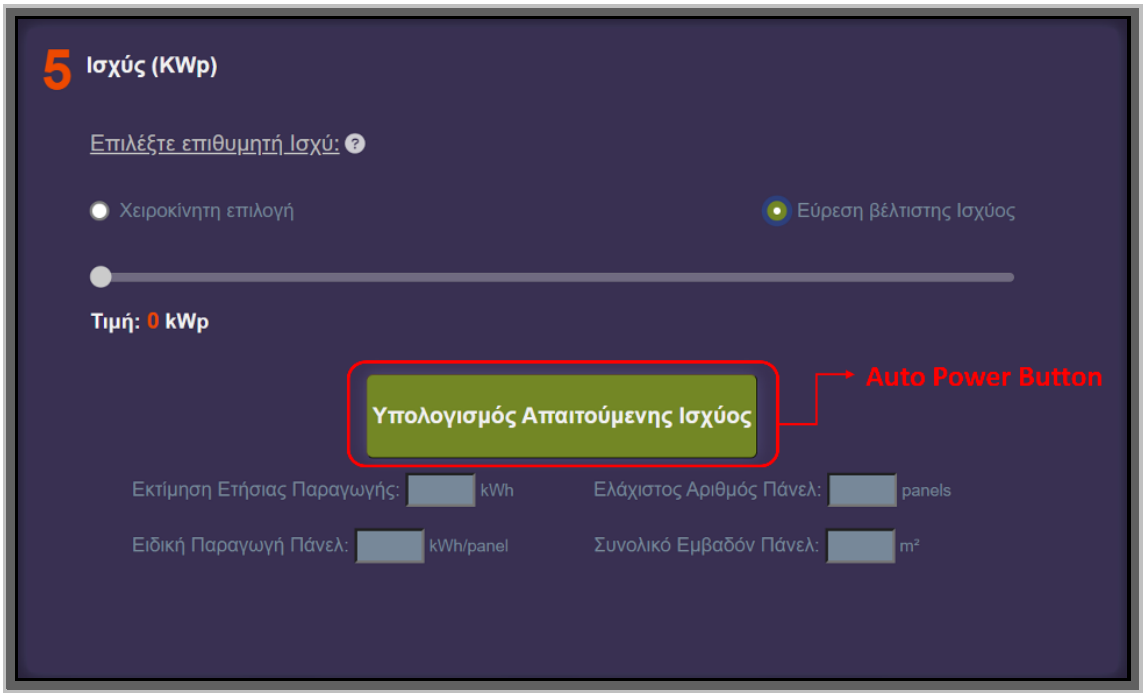



**Εικόνα 3-3 Μηνιαία ηλιακή ακτινοβολία μέσω PVGIS**

## 3.4 Ασύγχρονο HTTP αίτημα

Όταν ο χρήστης πατήσει το κουμπί ***autoPowerButton***, όπως φαίνεται στην Εικόνα 3-4, αποστέλλεται ασύγχρονο HTTP αίτημα στο back-end της εφαρμογής. Υπεύθυνη για την υλοποίηση της αποστολής και της λήψης του αιτήματος είναι η μέθοδος JavaScript ***calculateAutoPower()*** στο αρχείο *scriptCalculator.js*. Η μέθοδος $.ajax() της JavaScript βιβλιοθήκης jQuery, αποστέλλει στο URL, το οποίο αντιστοιχεί στη μέθοδο υπολογισμού ***calculate_power()*** στο *views.py* και έχει οριστεί στο *urls.py* ως *path('ajax/',*

*views.calculate_power, name='calculate_power')*, το λεξικό δεδομένων ***data***. Στο λεξικό αυτό βρίσκονται οι συντεταγμένες, το αζιμούθιο και η κλίση του ΦΒ συστήματος, όπως και επιπλέον παράμετροι (κατανάλωση, παράμετροι πάνελ, ποσοστό σκιάσεων, προφίλ κατανάλωσης). Σε περίπτωση επιτυχίας συμπληρώνονται με τις αντίστοιχες τιμές, τα πεδία κάτω από το *autoPowerButton*. Σε περίπτωση λάθους εκτυπώνεται στην κονσόλα του browser μήνυμα σφάλματος.




**Εικόνα 3-4 Κουμπί, Υπολογισμός Απαιτούμενης Ισχύος**

## 3.5 Υπολογισμός ισχύος

Αναφέραμε νωρίτερα πως όταν ζητηθεί από τον χρήστη να γίνει υπολογισμός βέλτιστης ισχύος, αποστέλλεται ασύγχρονο HTTP αίτημα (ajax request). Η μέθοδος ***calculate_power*** είναι υπεύθυνη να παραλάβει τα δεδομένα που απεστάλησαν με το αίτημα και να κάνει εξαγωγή των τιμών των απαραίτητων μεταβλητών. Υπολογίζει την ηλιακή ακτινοβολία καλώντας τη μέθοδο ***get_solar_data*** στην οποία αναφερθήκαμε προηγουμένως και ύστερα μέσα σε while loop την ετήσια παραγωγή ***annual_production***, ως το γινόμενο του αριθμού πάνελ επί της ειδικής παραγωγής ανά πάνελ, έως ότου ξεπεράσει την ετήσια κατανάλωση. Τέλος δημιουργεί λεξικό με το όνομα ***response_data***, την απόκριση δηλαδή στο ασύγχρονο αίτημα. Σε περίπτωση σφάλματος εμφανίζει *404 Generic Error*.

## 3.6 Ανάκτηση δεδομένων

Κατά το πάτημα του κουμπιού «Υποβολή» στο τελευταίο βήμα της φόρμας συμπλήρωσης, τα δεδομένα της φόρμας αποστέλλονται στο σώμα του αιτήματος HTTP POST. Στη μέθοδο ***calculator_form_fields_handler(request)***, η οποία είναι υπεύθυνη για τη διαχείριση των δεδομένων της φόρμας, ελέγχεται εάν είναι υπαρκτό τέτοιο HTTP αίτημα, μέσα σε if statement με τη μορφή ***request.method == 'POST'***. Επίσης γίνεται επικύρωση (validation) εάν είναι έγκυρη η φόρμα ***PhaseLoad***, υλοποιημένη στο *forms.py*, η οποία είναι υπεύθυνη για την επιλογή της συμφωνημένης παροχής, τριφασικής ή μονοφασικής στην οικιακή εγκατάσταση.

Στο Django framework, το request.POST είναι πρακτικά ένα instance αντικειμένου *QueryDict* και η μέθοδος *.get()*, η οποία στην περίπτωση μας χρησιμοποιήθηκε μέσα στο try-except block, χρησιμοποιείται για την ανάκτηση δεδομένων στην πλευρά του διακομιστή (server-side).

Αναλυτικότερα όταν μια φόρμα υποβάλλεται μέσω POST, έχουμε το αντικείμενο request.POST, το οποίο θυμίζει λεξικό και περιέχει ζεύγη κλειδιών-τιμών των δεδομένων της φόρμας. Η μέθοδος request.POST.get() μας επιτρέπει λοιπόν να ανακτήσουμε την τιμή όλων ή επιλεγμένων παραμέτρων, από το λεξικό. Λαμβάνει το όνομα της παραμέτρου ως όρισμα και επιστρέφει την αντίστοιχη τιμή, εάν υπάρχει. Εάν η παράμετρος δεν υπάρχει, επιστρέφει από προεπιλογή την τιμή None.

Εάν τα δεδομένα έχουν υποβληθεί ορθά και εκχωριστούν χωρίς προβλήματα στις αντίστοιχες μεταβλητές, η μέθοδος ουσιαστικά θα επιστρέψει τη μέθοδο ***dashboard_results()***, την οποία θα μελετήσουμε παρακάτω, με την εντολή *return dashboard_results(request)*. Σε περίπτωση σφάλματος ενημερώνει τον χρήστη και δίνει εντολή επαναφόρτωσης του calculator.html.

## 3.7 Ανάκτηση υπολογισμένων δεδομένων

Εδώ αναλύεται ο κώδικας της μεθόδου ***dashboard_results()*** η οποία είναι υπεύθυνη να «τραβήξει» τα δεδομένα επιλογής του χρήστη, τα οποία συλλέχθηκαν στο session, ύστερα από την υποβολή της φόρμας. Η κύρια εντολή ανάκτησης είναι η request.session.get(). Επίσης στη μέθοδο αυτή δηλώνονται και αρχικοποιούνται όλες οι μεταβλητές, οι οποίες

αποτελούν τα αποτελέσματα των διαφόρων πράξεων που πραγματοποιούνται από τις μεθόδους υπολογισμών που καλεί η ***dashboard_results***. Συνολικά οι μεταβλητές προστίθενται σε λεξικό με όνομα ***context{ }***, το οποίο θα επιστρέψει (return) στο URL που έχει δηλωθεί ως ***result***, με τη γραμμή κώδικα ***return render(request, result, context)***. Σε περίπτωση αποτυχίας θα εμφανιστεί μήνυμα σφάλματος με κωδικό 404. Επίσης όλος ο κώδικας περιέχεται (wrapped) σε try-except block για διαχείριση λαθών ή εξαιρέσεων, με προβολή σφάλματος με κωδικό 404, σε περίπτωση αστοχίας.

## 3.8 Δημιουργία διαγραμμάτων chart.js

Εδώ θα δούμε κάποια κρίσιμα σημεία, του υπεύθυνου κώδικα για τη δημιουργία των τριών γραφημάτων, ύστερα από τη υποβολή της συμπληρωμένης φόρμας. Είναι δύο διαγράμματα τύπου *line* και ένα τύπου *pie*. Τα δεδομένα (data) ***monthIrradianceArray*** και ***monthlyPanelEnergy*** από τα datasets, είναι μεταβλητές οι οποίες έχουν οριστεί στο *dashboard.html* και αντιστοιχούν το μεν πρώτο σε πίνακα μηνιαίας ηλιακής ακτινοβολίας ($kWh/m^2$) για το 2020, για τις συντεταγμένες οι οποίες επιλέχθηκαν και το δεύτερο στη μηνιαία παραγωγή ανά ΦΒ πάνελ σύμφωνα με τις επιλεγμένες παραμέτρους του. Τα δεδομένα ηλιακής ακτινοβολίας, αντλήθηκαν όπως είπαμε από το ευρωπαϊκό εργαλείο PVGIS. Όταν δημιουργηθούν τα διαγράμματα, προστίθενται στον πίνακα ***globalCharts[ ]***. Ο πίνακας αυτός θα καλεστεί σε επόμενο στάδιο, εάν απαιτηθεί επανυπολογισμός των datasets, ώστε να ελέγξουμε αφενός αν περιλαμβάνονται τα δημιουργημένα διαγράμματα, αφετέρου να ανατρέξουμε στα χαρακτηριστικά των διαγραμμάτων ώστε να αλλάξουμε τις τιμές.

Στο τελευταίο γράφημα το οποίο είναι σε μορφή πίτας παρουσιάζονται οικονομικά μεγέθη τα οποία έχουν υπολογιστεί στο back-end σε κατάλληλες μεθόδους. Στο chart.js έχουν δοθεί ως μεταβλητές οι δείκτες *roi*, *npv*, *lcoe*, *irr*, *annual_roi*, οι οποίες ορίστηκαν πρώτα στο *dashboard.html*, για να μπορέσει να γίνει η χρήση τους με JavaScript.

## 3.9 Μέθοδοι υπολογισμού δεδομένων

Το κομμάτι κώδικα της μεθόδου ***calculate_total_investment()***, υπολογίζει τη συνολική επένδυση του εκάστοτε ενδιαφερόμενου χρήστη. Ως κύριο σημείο αναφέρουμε τον έλεγχο

με if statement για τις διαφορετικές περιπτώσεις παροχής ηλεκτρικού ρεύματος, μονοφασικής (single_phase) και τριφασικής (3_phase) και επιθυμίας για προσθήκη συσσωρευτών, από τις οποίες καθορίζονται οι τιμές του αντιστροφέα ή μετατροπέα (inverter) και η αύξηση της επένδυσης λόγω μπαταρίας.

Το κομμάτι κώδικα της μεθόδου ***calculate_PV_energy_produced()***, υπολογίζει τη συνολική ετήσια αλλά και μηνιαία παραγόμενη ενέργεια ανά ΦΒ πάνελ, διατρέχοντας με for loop τη λίστα μηνιαίας ηλιακής ακτινοβολίας. Επίσης εδώ εισαγάγεται και η παράμετρος *special_production_per_panel* (βλέπε Κώδικα 3-2):

```
special_production_per_panel = round(annual_irradiance * panel_area *
panel_efficiency * performance_degradation * shadings_percentage)
```

**Κώδικας 3-2 Υπολογισμός ειδικής παραγωγής ανά πάνελ**

Η παράμετρος αυτή ισούται με το γινόμενο της ηλιακής ακτινοβολίας επί τις επιλεγμένες παραμέτρους εμβαδόν, βαθμό απόδοσης, καθώς και το ποσοστό σκιάσεων. Ο τελευταίος όρος *performance_degradation* είναι μία σταθερά κατά εκτίμηση, βάσει πρόσφατων μελετών, η οποία προσθέτει στη φόρμουλα υπολογισμού μεγαλύτερη ακρίβεια, καθώς συνυπολογίζει ενεργειακές απώλειες από παράγοντες όπως απρόβλεπτες σκιάσεις, σκόνη, διάχυση ακτινοβολίας, αύξηση θερμοκρασίας στην επιφάνεια του πάνελ κτλ. Αυτή η σταθερά ορίστηκε στο 0.75 (75%), οπότε προβλέπει μείωση 25% της αναμενόμενης μηνιαίας παραγωγής ανά ΦΒ πάνελ.

Το κομμάτι κώδικα της μεθόδου ***calculate_annual_savings()***, υπολογίζει το συνολικό ετήσιο όφελος, με την ένταξη σε πρόγραμμα ενεργειακού συμψηφισμού net metering. Περιλαμβάνει αρκετές παραμέτρους οι οποίες υπολογίζονται σε ξεχωριστές μεθόδους. Οπότε για να βρεθεί το συνολικό όφελος, συνυπολογίζονται το συνολικό κόστος κατανάλωσης, ρυθμιζόμενες χρεώσεις για την πλεονάζουσα ενέργεια που πιθανώς εγχύθηκε στο Δίκτυο και απαιτήθηκε εκ των υστέρων (έξοδα διανομής και μεταφοράς ρεύματος, ως παραμένοντες χρεωστικοί παράγοντες), το είδος παροχής και η ιδιοκαταναλισκόμενη ενέργεια. Να επισημάνουμε ξανά τη σημαντικότητα της ιδιοκατανάλωσης ως κριτήριο για αύξηση των ετήσιων κερδών, η οποία προκύπτει από το προφίλ κατανάλωσης του αυτοπαραγωγού. Όταν δηλαδή το ρεύμα που παράγεται από το ΦΒ σύστημα καταναλώνεται την ίδια στιγμή και οι ενεργειακές ανάγκες, σε ώρες μη

παραγωγής, καλύπτονται από μία αποδοτική συστοιχία συσσωρευτών, η ιδιοκατανάλωση μπορεί να αγγίξει το 90-95% με κρίσιμο αντίκτυπο στον λογαριασμό του ρεύματος.

Το κομμάτι κώδικα της μεθόδου ***calculate_payback_period()***, υπολογίζει κατά προσέγγιση την αναμενόμενη περίοδο απόσβεσης της επένδυσης. Αυτό γίνεται με τη χρήση while loop, προσθέτοντας το υπολογισμένο ετήσιο όφελος, συνυπολογίζοντας πάντα και την ετήσια υποβάθμιση παραγωγής ενός ΦΒ συστήματος. Όταν το άθροισμα των ετήσιων οφελών ισοσταθμιστεί ή ξεπεράσει την αρχική επένδυση, θεωρείται πως έχει πραγματοποιηθεί απόσβεση επένδυσης. Για λόγους μεγαλύτερης ακρίβειας και ευελιξίας, έχουν προστεθεί διαφορετικές περιπτώσεις, όπου το ετήσιο όφελος μπορεί να παίρνει από μηδενικές τιμές, έως να καλύπτει το ετήσιο κόστος.

Το κομμάτι κώδικα της μεθόδου ***calculate_total_production_kwh()***, υπολογίζει κατά προσέγγιση την αναμενόμενη συνολική παραγωγή ηλεκτρικής ενέργειας, του εγκατεστημένου ΦΒ συστήματος με το πέρας της 25ετίας. Υπολογίζεται ως συνολική τιμή παραγωγής αλλά και ως πίνακας 25 τιμών, ώστε να υπάρχει και η κάθε ετήσια τιμή, πριν προστεθεί στο τελικό άθροισμα.

## 3.10 Υπολογισμός οικονομικών δεικτών

Δημιουργήθηκαν 4 μέθοδοι υπολογισμού των οικονομικών δεικτών, οι οποίοι εμφανίζονται στην παρουσίαση των αποτελεσμάτων στο γράφημα τύπου πίτα και βασίζονται στους μαθηματικούς τύπους που αναλύθηκαν στην Ενότητα 2.6 του Κεφαλαίου 2.

- **Υπολογισμός NPV**
- **Υπολογισμός ROI & Υπολογισμός AROI**
- **Υπολογισμός LCOE**
- **Υπολογισμός IRR**

Για τον υπολογισμό του εσωτερικού συντελεστή απόδοσης, έγινε import της python βιβλιοθήκης ***numpy_financial***. Από τη βιβλιοθήκη αυτή, έγινε χρήση της ενσωματωμένης μεθόδου .irr() η οποία υπολογίζει τον συντελεστή απόδοσης με παράμετρο μία λίστα με τις ροές εσόδων - εξόδων. Για να είναι σωστό το αποτέλεσμα, με τη χρήση της μεθόδου insert(), έγινε προσθήκη της αρχικής επένδυσης (*initial_investment*) στη λίστα των ετήσιων κερδών (*saving_flows*) στο index 0.

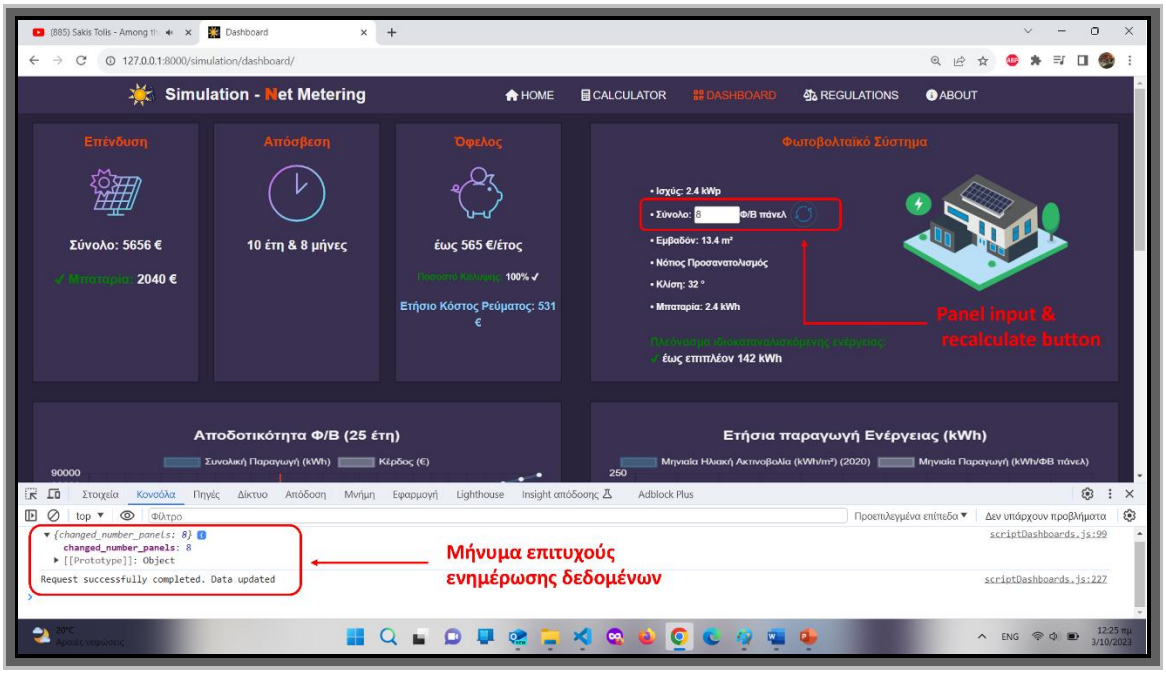
## 3.11 Επανυπολογισμός δεδομένων

Πλέον ο χρήστης έχει μεταφερθεί στην οθόνη παρουσίασης των αποτελεσμάτων, σύμφωνα και με τις κυριότερες μεθόδους που αναδείχτηκαν στις ενότητες 3.7 έως και 3.10. Όπως προαναφέρθηκε ο χρήστης έχει δυνατότητα, στη φάση συμπλήρωσης της φόρμας, να προχωρήσει με χειροκίνητη ή αυτόματη επιλογή ισχύος ΦΒ συστήματος. Εάν σε εκείνη τη φάση δοθεί τιμή ισχύος kWp κατά εκτίμηση του χρήστη (χειροκίνητη), όλα τα αποτελέσματα των υπολογισμών που ακολουθούν, συσχετίζονται άμεσα με αυτή την τιμή. Το ΦΒ σύστημα που θα προκύψει, δύναται ή όχι, να καλύψει το ετήσιο κόστος κατανάλωσης. Υπάρχει πάντα το ενδεχόμενο με αυτόν τον τρόπο, η παραγόμενη ενέργεια να υπερβαίνει ή να μην επαρκεί έναντι της καταναλισκόμενης, σε βαθμό τέτοιο, ώστε η επένδυση να μη μπορεί να θεωρηθεί συμφέρουσα.

Εν αντιθέσει εάν επιλεχθεί προαιρετικός υπολογισμός βέλτιστης ισχύος, η διαδικασία πραγματοποιείται στο back-end, προτού υποβληθεί η φόρμα. Ο αλγόριθμος που υπολογίζει τη βέλτιστη δυνατή ισχύ του ΦΒ συστήματος, βάσει των παραμέτρων, επιδιώκει να προσεγγίσει ή/και να εκμηδενίσει το ετήσιο κόστος κατανάλωσης. Γενικός κανόνας του αλγόριθμου αυτού είναι, πως η ισχύς θεωρείται βέλτιστη, όταν η ετήσια παραγωγή υπερβαίνει την ετήσια κατανάλωση.

Σε κάθε περίπτωση, δίνεται η δυνατότητα στον χρήστη, να αλλάξει την ισχύ του ΦΒ, ώστε να εντοπίσει μεταβολές των οικονομικών μετρικών και δεικτών, χωρίς όμως να χρειάζεται να επαναλάβει συμπλήρωση νέας φόρμας. Αυτό επιτυγχάνεται, δίνοντάς του το δικαίωμα να αυξομειώσει τον αριθμό των πάνελ, τα οποία ορίζουν τελικά και την τιμή kWp της ισχύος του συστήματος, όπως φαίνεται στις Εικόνες 3-5 και 3-6.


**Εικόνα 3-5 Κουμπί επανυπολογισμού δεδομένων**




**Εικόνα 3-6 Πεδίο πάνελ, κουμπί επανυπολογισμού, μήνυμα κονσόλας**

Οι μέθοδοι οι οποίες ευθύνονται για την λειτουργικότητα που μόλις περιγράφηκε, σχεδιάστηκαν με σκοπό να προσθέσουν διαδραστικότητα (interactivity) και δυναμική συμπεριφορά (dynamic) στην οθόνη παρουσίασης. Η υλοποίηση ξεκίνησε με δημιουργία button στο *dashboard.html*, το οποίο ενεργοποιεί τη μέθοδο ***recalculatePvSystemProperties***. Η μέθοδος αυτή της JavaScript, η οποία βρίσκεται στο αρχείο *scriptDashboards.js*, αποστέλλει με ασύγχρονο αίτημα στο back-end, το *csrf Token* για λόγους ασφάλειας μαζί με τον ανανεωμένο αριθμό πάνελ. Αυτό είπαμε γίνεται για να

εκτελεστούν ξανά οι μέθοδοι υπολογισμών, καθώς η αλλαγή του αριθμού πάνελ θα επιφέρει αλλαγές σε όλες τις τιμές των μεταβλητών. Όταν τελειώσουν οι υπολογισμοί, η μέθοδος θα λάβει την απάντηση (response) και θα αντικαταστήσει τα αντίστοιχα πεδία με τις νέες πλέον τιμές.

Από τη πλευρά του back end, η μέθοδος ***recalculate_pv_system_properties*** λαμβάνει το ασύγχρονο αίτημα και πραγματοποιεί τους επανυπολογισμούς, χρησιμοποιώντας τον νέο αριθμό πάνελ. Κατά συνέπεια όλες οι μεταβλητές θα αλλαχθούν και οι νέες τιμές θα σταλούν εκ νέου, στη μέθοδο που αναφέραμε προηγουμένως, προς αντικατάσταση των επιλεγμένων πεδίων. Να αναφέρουμε ότι οι τιμές οι οποίες είχαν νωρίτερα αποθηκευτεί στο Django session, μένουν αναλλοίωτες, οπότε με reset της οθόνης παρουσίασης, δηλαδή του dashboard.html οι τιμές θα επανέλθουν στις αρχικά υπολογισμένες. Άρα οι καινούριες τιμές είναι προσωρινές και δεν αποθηκεύονται κάπου, απλά προβάλλονται στην οθόνη έως ότου τερματιστεί.

## 3.12 Υπόλοιπα αρχεία εφαρμογής

Η εφαρμογή αποτελείται από χιλιάδες γραμμές κώδικα, σε επιπλέον αρχεία python, html, css και scripts, τα οποία είναι υπεύθυνα για την επικοινωνία, την ασφάλεια, το οπτικό κομμάτι, την αισθητική και λοιπά σημεία διαδραστικότητας και ελέγχου. Θεωρήθηκε ωφέλιμο όμως να εξαιρεθούν και να γίνει αναφορά των μεθόδων που συνέβαλαν περισσότερο στο αντικείμενο της διπλωματικής, την προσομοίωση δηλαδή ενός ΦΒ συστήματος σε πρόγραμμα net metering. Πάντα όμως συνυπάρχουν με τον προηγούμενο κώδικα στην πλατφόρμα GitHub και στο link, το οποίο δόθηκε στην αρχή του Κεφαλαίου.

Βέβαια θεωρήθηκε χρήσιμο να γίνει μνεία στα βοηθητικά πλαίσια συμβουλών, τα οποία βρίσκονται σε ποικίλα σημεία της φόρμας συμπλήρωσης, για να καθοδηγούν και να διευκολύνουν τον χρήστη. Είναι στην ουσία κουμπιά (help poppers), στα οποία έχει εφαρμοστεί στυλ με css, ώστε να μοιάζουν με κυκλικά εικονίδια με ένα ερωτηματικό ενδιάμεσα, τα οποία όταν πατηθούν, με κατάλληλες εντολές JavaScript φανερώνουν ένα κείμενο επεξήγησης. Για να κλείσουν τα πλαίσια κειμένου, αρκεί να πατηθεί κάποιο άλλο help popper ή η επιλογή **x** στη δεξιά άνω γωνία του πλαισίου.

Η μορφή τους φαίνεται στον Κώδικα 3-3 και είναι η εξής:

```
<button class="help-popper" type="button" title="help-button">
    <i class="fa fa-question-circle" id="place-info"></i>
</button>

<div class="help-container">
   <span class="help-text" id="place-text">
   &#8226; Επιτρέπετε να επιλέξετε στίγμα, μόνο εντός των γεωγραφικών
συνόρων <br>της Ελλάδας. Οτιδήποτε, εκτός των οριοθετημένων περιοχών με
γραμμή μπλε χρώματος, θεωρείται <strong>μη αποδεκτή τιμή</strong>.<br>
   </span>
   <span class="close-help">&times;</span>
</div>
```

**Κώδικας 3-3 Βοηθητικά πλαίσια συμβουλών σε html**

Στην Εικόνα 3-7 φαίνεται ενεργοποιημένο ένα *help popper*, στο πάνελ 3 όπου ζητείται η συμπλήρωση του αζιμούθιου και της γωνίας κλίσης.


**Εικόνα 3-7 Βοηθητικά πλαίσια συμβουλών και κείμενο**

# 4. Αξιολόγηση & συμπεράσματα

Ο στόχος αυτού του Κεφαλαίου είναι να προσδιοριστεί η αξιοπιστία, η ακρίβεια και η χρηστικότητα της εφαρμογής σε σύγκριση με ήδη υπάρχοντα εργαλεία και εφαρμογές, όπως και να εντοπιστούν πιθανές μελλοντικές βελτιώσεις.

## 4.1 Αξιολόγηση

Το ζητούμενο της αξιολόγησης, είναι να εκτιμηθεί πόσο καλά ευθυγραμμίζονται τα αποτελέσματα της εφαρμογής με εκείνα που παράγονται από άλλα εργαλεία και εφαρμογές προσομοίωσης για ΦΒ συστήματα και να αναδειχθούν πλεονεκτήματα και εξαρτήσεις.

1. **Συγκριτική ανάλυση**

Μέσω δοκιμών, διαπιστώθηκε ότι η εφαρμογή παράγει σταθερά αποτελέσματα που βρίσκονται σε συμφωνία με εκείνα αντίστοιχων εργαλείων. Είναι λοιπόν ένα αρχικό μέτρο σύγκρισης και αξιοπιστίας της εφαρμογής.

Θα πρέπει να καταστεί όμως σαφές, ότι παρότι τα δεδομένα του PVGIS tool διαπιστώνονται να είναι σταθερά, δεν έγινε περαιτέρω έλεγχος της μοντελοποίησης δεδομένων του εργαλείου, για παράδειγμα συγκρίνοντας μοντέλα παρεμφερών εφαρμογών που δραστηριοποιούνται στον χώρο της ηλιακής ακτινοβολίας.

2. **Ασύγχρονα αιτήματα**

Η απρόσκοπτη λειτουργία των ασύγχρονων αιτημάτων, είναι ένα αξιοσημείωτο χαρακτηριστικό αυτής της εφαρμογής. Εξασφαλίζει ότι ο χρήστης δύναται να ανακτά αποτελεσματικά τα δεδομένα και να εκτελεί προσομοιώσεις χωρίς καθυστερήσεις. Αυτή η λειτουργικότητα όχι μόνο βελτιώνει την εμπειρία του χρήστη, αλλά συμβάλλει επίσης στη συνολική αποτελεσματικότητα της εφαρμογής.

3. **Εξάρτηση από PVGIS service**

Αναφέρουμε ξανά ότι απαιτείται επικοινωνία με το ευρωπαϊκό εργαλείο PVGIS, ώστε η εφαρμογή να λειτουργεί ομαλά,. Παρατηρείται λοιπόν μία άμεση εξάρτηση από αυτό. Αν για κάποιο λόγο το ευρωπαϊκό εργαλείο πάψει να υποστηρίζει επικοινωνία μέσω API ή

κάποιο τεχνικό ζήτημα παρουσιαστεί, θα επηρεαστεί και η παρούσα εφαρμογή. Να σημειωθεί τέλος, ότι σύμφωνα με το documentation του PVGIS API, οι κλήσεις API έχουν όριο ρυθμού 30 κλήσεων/δευτερόλεπτο ανά διεύθυνση IP. Εάν γίνει υπέρβαση αυτού του ορίου, η υπηρεσία θα αρνηθεί την κλήση και θα επιστρέψει την κατάσταση HTTP “429 - Too Many Requests”.

### 4. Πραγματοποίηση δοκιμών

Για να διασφαλιστεί λοιπόν η αξιοπιστία και η ορθότητα της εφαρμογής, πραγματοποιήθηκαν εκτεταμένες δοκιμές. Οι δοκιμές αυτές κάλυψαν ένα ευρύ φάσμα σεναρίων, συμπεριλαμβανομένων διαφορετικών επιπέδων και προφίλ κατανάλωσης, παραγωγής ενέργειας και διαφορετικών παραμέτρων ΦΒ πλαισίων και γενικά της εγκατάστασης. Η σχολαστική διαδικασία δοκιμών συνέβαλε στην εξάλειψη της πιθανότητας δυσλειτουργιών και επιβεβαίωσε ότι η εφαρμογή παρέχει σταθερά και ακριβή αποτελέσματα στα διάφορα σενάρια χρήσης. Βέβαια πρέπει να αναφερθεί ότι επειδή η εφαρμογή δε δοκιμάστηκε με επισκεψιμότητα πολλαπλών χρηστών την ίδια στιγμή, δεν υπάρχει ακριβής εικόνα για τη συμπεριφορά της σε ευρεία κλίμακα χρήσης.

### 5. Συμβατότητα με προγράμματα περιήγησης (browsers)

Η προσβασιμότητα του χρήστη αποτελεί κρίσιμο παράγοντα για κάθε εφαρμογή που βρίσκεται στο διαδίκτυο. Η εφαρμογή υποβλήθηκε σε δοκιμές σε σταθερό υπολογιστή και laptop με λειτουργικό σύστημα Windows 10 και 11 και σε τρία κύρια προγράμματα περιήγησης: Google Chrome, Mozilla Firefox και Microsoft Edge. Αυτή η προσέγγιση, δηλαδή δοκιμή πολλαπλών περιηγητών διασφαλίζει ότι ένα ευρύ φάσμα χρηστών μπορεί να έχει πρόσβαση και να χρησιμοποιεί την εφαρμογή χωρίς προβλήματα συμβατότητας. Είναι σημαντικό να συνεχιστεί η παρακολούθηση και η δοκιμή της εφαρμογής σε διαφορετικές εκδόσεις προγραμμάτων περιήγησης ή/και σε λειτουργικό σύστημα και περιηγητές άλλων brands, όπως και άλλες συσκευές κινητά ή tablet, για να διατηρηθεί μια απρόσκοπτη εμπειρία χρήσης.

## 4.2 Μελλοντικές βελτιώσεις

Η αξιολόγηση ανέδειξε τα πλεονεκτήματα της εφαρμογής και τις εξαρτήσεις, οπότε υπάρχουν ορισμένοι τομείς για πιθανή βελτίωση και περαιτέρω διερεύνηση:

**1. Επεκτασιμότητα**

Πέραν της χρήσης της εφαρμογής καθολικά από όλους ως επισκέπτες, μία αρχική βελτίωση θα ήταν να υλοποιηθεί η εγγραφή χρηστών-μελών και ως επέκταση αυτής, η δυνατότητα αποθήκευσης προσομοιώσεων σε αντίστοιχη ΒΔ. Αυτή η προσθήκη θα μπορούσε να προσφέρει τη χρήση των αποτελεσμάτων διαφορετικών προσομοιώσεων ως ιστορικό. Τα διαφορετικά αυτά σενάρια θα λειτουργούσαν ως ένα άμεσο μέτρο σύγκρισης μεταξύ εναλλαγών στις παραμέτρους.

Επόμενη βελτίωση επεκτασιμότητας θα ήταν η διεύρυνση απευθυνόμενου κοινού εφαρμογής, όπως αγρότες και λοιποί επαγγελματίες, καθώς και διαφορετική αντιμετώπιση περιπτώσεων με πολλούς ενοίκους μίας πολυκατοικίας.

**2. Βελτιστοποίηση επιδόσεων**

Να εξεταστεί το ενδεχόμενο βελτιστοποίησης των επιδόσεων της εφαρμογής, ώστε να χειρίζεται αποτελεσματικότερα μεγαλύτερα σύνολα δεδομένων και προσομοιώσεων. Μια πιθανή εξέλιξη του συστήματος θα ήταν, να συμβουλεύει και να ενημερώνει τον χρήστη, περισσότερο στοχευμένα με συγκεκριμένες αλλαγές παραμέτρων οι οποίες θα ανέβαζαν την απόδοση.

Με γνώμονα την εμπειρία του χρήστη (UX), να πραγματοποιούνται βελτιώσεις της διεπαφής χρήστη και συλλογή ανατροφοδότησης ικανοποίησης των χρηστών.

Τακτικές αξιολογήσεις και ενημερώσεις ασφαλείας για την προστασία των δεδομένων των χρηστών και τη διασφάλιση ότι η εφαρμογή παραμένει ασφαλής έναντι πιθανών απειλών.

**3. Βελτίωση στο ζήτημα εξάρτησης από το PVGIS tool**

Ειδικά στην περίπτωση επέκτασης της εφαρμογής, θα πρέπει να αναζητηθούν λύσεις σχετικά με το ζήτημα εξάρτησης από το service του PVGIS tool. Πιθανές λύσεις στο σοβαρό αυτό θέμα θα ήταν:

**α) Εφεδρικό πλάνο αντιμετώπισης προκλήσεων**

Μία πιθανή λύση για αντιμετώπιση σφάλματος κατά την ανάκτηση δεδομένων ηλιακής ακτινοβολίας από το PVGIS, θα ήταν η δημιουργία ενός ενσωματωμένου (integrated) βασικού μοντέλου δεδομένων σε επίπεδο Νομών, σαφώς με κάποιες αποκλίσεις.

**β) Εναλλακτική ανάκτηση δεδομένων**

Μια δεύτερη λύση θα ήταν να υπάρχει δυνατότητα να γίνεται ανάκτηση δεδομένων από άλλο solar data service.

Και οι δύο προτάσεις δύνανται να διαχειριστούν ή/και να αποτρέψουν το σφάλμα 429 του PVGIS tool API.

## 4.3 Συμπεράσματα

Συμπερασματικά, η εφαρμογή προσομοίωσης ΦΒ συστημάτων σε προγράμματα net metering απέδειξε τη χρηστικότητά της, με την αξιοπιστία και την ακρίβειά της. Απέδωσε συγκριτικά με τα υπάρχοντα εργαλεία και πέρασε επιτυχώς τις δοκιμές υπολογισμών αλλά και των τεχνικών ασύγχρονης επικοινωνίας. Οι συνεχείς βελτιώσεις στις επιδόσεις, τη διεπαφή χρήστη, την ασφάλεια, την τεκμηρίωση και την επεκτασιμότητα θα ενισχύσουν περαιτέρω την αξία της εφαρμογής για τους χρήστες της και όλους όσους δείχνουν ενδιαφέρον για τον τομέα των ΑΠΕ και του net metering.

# Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

1. *άρθρο 59 του Ν.4843/2021 - ΦΕΚ 193 Α' 2021, «Ενεργειακός συμψηφισμός - Τροποποίηση της περ. 12 του άρθρου 2 του ν. 3468/2006», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/EE_FEK193A_2021_RES_only.pdf*
2. *«Αυτοπαραγωγή με ενεργειακό συμψηφισμό και εικονικό ενεργειακό συμψηφισμό για ιδιώτες, επιχειρήσεις και ενεργειακές κοινότητες με ή χωρίς αποθήκευση», Ανακτήθηκε από https://helapco.gr/pdf/HELAPCO_Net_Metering.pdf*
3. *ΔΕΗ "myEnergySolar", Ανακτήθηκε από https://www.dei.gr/el/gia-to-spiti/myenergy/myenergy-solar/*
4. *ΦΕΚ 3583Β/31-12-2014: «Εγκατάσταση μονάδων ΑΠΕ από αυτοπαραγωγούς με συμψηφισμό ενέργειας», Ανακτήθηκε από https://helapco.gr/pdf/FEK_Net_Metering_31Dec2014.pdf*
5. *ΝΟΜΟΣ ΥΠ' ΑΡΙΘΜ. 4414/2016 ΦΕΚ 149/Α/9-8-2016, «Νέο καθεστώς στήριξης των σταθμών παραγωγής ηλεκτρικής ενέργειας από ανανεώσιμες πηγές ενέργειας», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/N_4414_2016.pdf*
6. Ν.4513/2018, ΦΕΚ 9Α/23.1.2018, *«Ενεργειακές Κοινότητες», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/N_4513_2018.pdf*
7. ΥΑ ΥΠΕΝ/ΔΑΠΕΕΚ/15084/382, ΦΕΚ 759Β/5.3.2019, «*Εγκατάσταση σταθμών παραγωγής από αυτοπαραγωγούς με εφαρμογή ενεργειακού συμψηφισμού ή εικονικού ενεργειακού συμψηφισμού», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/FEK759B_2019.pdf*
8. *«Τεχνικό Εγχειρίδιο ΔΕΔΔΗΕ 2023», Ανακτήθηκε από https://deddie.gr/media/37431/τεχνικό-εγχειρίδιο-εσωτ-εγκατάστασης-net-metering-gr107a.pdf*
9. *«Ενεργειακός συμψηφισμός με μπαταρία», Ανακτήθηκε από https://www.mp-energy.gr/category/312/net-metering.html*
10. *«Οδηγός Εφαρμογής Προγράμματος - Φωτοβολταϊκά στη Στέγη», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/-ΟΔΗΓΟΣ-ΦΩΤΟΒΟΛΤΑΪΚΑ-ΣΤΗ-ΣΤΕΓΗ_23.03.2023.pdf*

11. *Ν.3468/2006, ΦΕΚ 129Α/29-6-2006 «Παραγωγή Ηλεκτρικής Ενέργειας από Ανανεώσιμες Πηγές Ενέργειας και Συμπαραγωγή Ηλεκτρισμού και Θερμότητας Υψηλής Απόδοσης και λοιπές διατάξεις», Ανακτήθηκε από https://helapco.gr/pdf/n3468_2006.pdf*
12. *«Στατιστικά στοιχεία εφαρμογής net-metering για το 2017», Ανακτήθηκε από https://helapco.gr/nea/nea-2018/statistika-stichia-efarmogis-net-metering-gia-to-2017/*
13. *«Στατιστικά στοιχεία αγοράς φωτοβολταϊκών για το 2020», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/pv-stats_greece_2020_18May2021.pdf*
14. *«Στατιστικά στοιχεία αγοράς φωτοβολταϊκών για το 2021», Ανακτήθηκε από https://helapco.gr/wp-content/uploads/pv-stats_greece_2021_3May2022.pdf*
15. *ΦΕΚ 2903Β'/2.5.2023: «Προκήρυξη του Προγράμματος Φωτοβολταϊκά στη Στέγη» Ανακτήθηκε από https://pvstegi.gov.gr/content/FEK-2023-Tefxos_20B-02903.pdf*
16. *"Electricity price statistics", Ανακτήθηκε από https://ec.europa.eu/eurostat/Electricity_price_statistics*
17. *«ΟΔΗΓΙΑ 2009/28/ΕΚ» του Ευρωπαϊκού Κοινοβουλίου και του Συμβουλίου της 23ης Απριλίου 2009, Ανακτήθηκε από https://eur-lex.europa.eu/2009:140:0016:0062:EL:PDF*
18. *«ΟΔΗΓΙΑ (ΕΕ) 2018/2001» του Ευρωπαϊκού Κοινοβουλίου και του Συμβουλίου της 11ης Δεκεμβρίου 2018, Ανακτήθηκε από https://eur-lex.europa.eu/legal-content/EL/TXT/PDF/?uri=CELEX:32018L2001&from=LV*
19. *"Renewable Energy Recast to 2030" Ανακτήθηκε από https://joint-research-centre.ec.europa.eu/renewable-energy-recast-2030-red-ii*
20. *"Renewable Energy Directive" Ανακτήθηκε από https://energy.ec.europa.eu/topics/renewable-energy-directive*
21. *Theodoros G. Iliopoulos et al (2020), "The EU's 2030 Climate and Energy Policy Framework: How net metering slips through its net"*
22. *A. Poullikkas et al, "A Review of Net Metering Mechanisms for Electricity Renewable Energy Sources", 2013*
23. *"Denmark PV Technology Status and Prospects", Ανακτήθηκε από https://web.archive.org/web/Denmark.pdf.*
24. *"Net metering", Ανακτήθηκε από https://en.wikipedia.org/wiki/Net_metering*

25. *"Government Plan: phase out the net-metering scheme", Ανακτήθηκε από https://www.rijksoverheid.nl/energie-thuis/plan-salderingsregeling*
26. *"Act of February 20, 2015 on renewable energy sources", Ανακτήθηκε από https://isap.sejm.gov.pl/isap.nsf/*
27. *"Multi Annual Energy Plan", Ανακτήθηκε από https://www.ecologie.gouv.fr/sites/default/files/PPE.pdf*
28. *"Micro-generation", Ανακτήθηκε από https://www.gov.ie/en/publication/micro-generation*
29. *"Regulation on self-supply of electricity from renewable energy sources", Ανακτήθηκε από https://www.uradni-list.si*
30. *"The web framework Django", Ανακτήθηκε από https://www.djangoproject.com/*
31. *"Leaflet: an open-source JavaScript library", Ανακτήθηκε από https://leafletjs.com/*
32. *"Photovoltaic Geographical Information System", Ανακτήθηκε από https://re.jrc.ec.europa.eu/pvg_tools/en/*
33. *"Git --distributed-is-the-new-centralized", Ανακτήθηκε από https://git-scm.com/*
34. *«Όρια Νομών (ΟΚΧΕ)», Ανακτήθηκε από https://geodata.gov.gr/dataset/oria-nomon-okkhe*

# Παράρτημα Α: Βιβλιοθήκες και modules Python

Οι βιβλιοθήκες και το module που χρησιμοποιήθηκαν για τον υπολογισμό δεδομένων ήταν τα εξής:

- Η βιβλιοθήκη numpy (Numerical Python) η οποία χρησιμοποιείται για την εργασία με πίνακες. Διαθέτει επίσης συναρτήσεις για την εργασία στον τομέα της γραμμικής άλγεβρας και του μετασχηματισμού Fourier.
- Η βιβλιοθήκη pvlib.iotools για λειτουργίες ανάκτησης, ανάγνωσης και εγγραφής δεδομένων από πηγές και μορφές αρχείων που σχετίζονται με τη μοντελοποίηση της ηλιακής ενέργειας.
- Η βιβλιοθήκη numpy_financial η οποία είναι μια συλλογή στοιχειωδών χρηματοοικονομικών συναρτήσεων. Οι συναρτήσεις αυτές αντιγράφηκαν σε αυτό το πακέτο από την έκδοση 1.17 του NumPy.
- Το ενσωματωμένο module json το οποίο μπορεί να χρησιμοποιηθεί για την επεξεργασία δεδομένων JSON.

Οι τρεις βιβλιοθήκες εγκαταστάθηκαν με χρήση εντολής pip install.

# Παράρτημα Β: Κώδικας εφαρμογής

```
<!DOCTYPE html>

{% load static %}

<html lang="en">
<head>
    <meta charset="UTF-8">
    {% block title %}
    <title>Homepage</title>
    {% endblock %}

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.mi
n.css" rel="stylesheet"
    integrity="sha384-
4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9"
crossorigin="anonymous">
    <link rel="website icon" type="png" href="{% static
'images/panel_128x107.png'%}">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.2/css/all.min.css"/>
    <link rel="stylesheet" href=" {% static
'Net_Metering/css/styles.css' %}"/>

    {% block css %}
    {% endblock %}

    <script src="{% static 'Net_Metering/javascript/script.js' %}"
defer></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bund
le.min.js"
    integrity="sha384-
HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm"
crossorigin="anonymous"></script>

    {% block script %}
    {% endblock %}

</head>

<body class="{% block body_class %}{% endblock %} ">

        <nav class="main-navbar">

            <div class="main-navbar-right-container">

                <header class="main-header">
                    <div class="header-container-left">
                        <a href="{% url 'home' %}"><img class="sm-image"
src="{% static 'images/panel.png'%}" alt="photo not found"></a>
                        <p class="logo"><a href="{% url 'home' %}">
                          Simulation - <strong style="color:
#ed4901;">N</strong>et Metering</a></p>
                    </div>
```

```
                </header>
            </div >

            <div class="main-navbar-left-container" id="main-navbar-
left-container">
                <input type="checkbox" id="check-navbar">
                <label  for="check-navbar" id="menu-nav-button"
class="demo-class"><i class="fas fa-bars"></i></label>

                <ul class="nav-list" id="nav-list">
                    {% block menu_list %}
                    <li><a class="nav-link active" href="{% url 'home'
%}"><i class="fas fa-home"></i> home</a></li>
                    <li><a href="{% url 'simulation:dashboard' %}"><i
class="fas fa-qrcode"></i> dashboard</a></li>
                    <li><a href="{% url 'simulation:calculator' %}"><i
class="fas fa-calculator"></i> calculator</a></li>
                    <li><a href="{% url 'simulation:regulations' %}"><i
class="fas fa-scale-unbalanced-flip"></i> regulations</a></li>
                    <li><a href="{% url 'simulation:about' %}"><i
class="fas fa-info-circle"></i> about</a></li>
                    {% endblock %}
                </ul>
                   <div class="header-container-right">
                        {% block color_theme %}

                        {% endblock %}

                    </div>
            </div>

        </nav>

        <div id="main-index-content-for-all">
            {% block content %}

            {% endblock %}
        </div>

    <footer class="footer">
       <div class="other-links">
          <a href="https://linkedin.com/in/anestis-giannakosian-
880707220" title="connection to linkedin"><i class="fab fa-linkedin-
in"></i> </a>
          <a href="https://github.com/anegian" title="connection to
github"><i class="fab fa-github"></i></a>
          <a href="https://youtube.com" title="connection to youtube"><i
class="fab fa-youtube"></i></a>
       </div>
          <span class="admin-info">
              <i class="fas fa-copyright"> 2023 Anestis Giannakosian.
All rights reserved</i>
        </span>
  </footer>
</body>
</html>
```

**Κώδικας 1 - Αρχείο index.html (HTML)**

```
// Map
var southWest = L.latLng(30, 15); // SW corner (between Africa and
Balkans)
var northEast = L.latLng(45, 32); // NE corner (between Italy, and
Turkey)
var bounds = L.latLngBounds(southWest, northEast); // LatLngBounds
object
let map = L.map('map').setView([37.983917, 23.72936], 7);
let latitude = document.getElementById('latitude')
let longitude = document.getElementById('longitude')
// Get the region input element
let regionInput = document.getElementById('regionInput');
let geojsonLayer;

// Set the maximum bounds for the map after initialization
map.setMaxBounds(bounds);

// When the map moves --> end of the bounds,returns to center coords
map.on('moveend', function () {
    var center = map.getCenter();

    // Check if the map's center longitude has wrapped around
    if (center.lng < -180 || center.lng > 180) {
      // Re-add your layer or perform any other actions you need
      // Example: map.addLayer(yourLayer);
      console.log('Map has wrapped around');
    }
});

// 1st layer, the map itself
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);

// Creating Marker options
var markerOptions = {
    title: "MyLocation",
    clickable: true,
}
// Creating a marker
let marker = L.marker([37.983917, 23.72936], markerOptions);

// Adding marker to the map
marker.addTo(map);


//Function to add text popup, prefecture identifier layer
function popUp(feature, layer) {
    if (feature.properties) {
        const propertiesToShow = ['ΝΟΜΟΣ', 'ΕΔΡΑ', 'ΠΛΗΘΥΣΜΟΣ'];
        const out = propertiesToShow.map(key => {
            let value = feature.properties[key];
            if (key === 'NAME') {
                value = decodeURIComponent(value);
            }
            return key + ': ' + value;
        });
```

```
        layer.bindPopup(out.join('<br />'));
    }
};

// Function to check if a point is inside a polygon using Turf.js
function isPointInsidePolygon(lng, lat, polygon) {
    const turfPoint = turf.point([lng, lat]);
    const turfPolygon = turf.polygon(polygon);
    return turf.booleanPointInPolygon(turfPoint, turfPolygon);
};
function onMapClick(e) {
    latitude.value = e.latlng.lat.toFixed(4);
    longitude.value = e.latlng.lng.toFixed(4);
    console.log(latitude.value);
    console.log(longitude.value);

    let isPointOutsideBounds = true;

    if (geojsonLayer) {
        geojsonLayer.eachLayer(layer => {
            const geometryType = layer.feature.geometry.type;
            const polygons = layer.feature.geometry.coordinates;

            if (geometryType === 'MultiPolygon') {
                polygons.forEach(polygon => {
                    if ( isPointInsidePolygon(longitude.value,
latitude.value, polygon) ) {
                        console.log(layer.feature.properties.NOMOΣ);
                        regionInput.value =
layer.feature.properties.NOMOΣ;
                        placeSelected.value = regionInput.value;
                        triggerButtonEnable();

                        //layer.openPopup();
                        isPointOutsideBounds = false;
                         // Remove the existing marker from the map
                        if (marker) {
                            marker.remove();
                        }
                        // Create a new marker at the clicked position
                        marker = L.marker([latitude.value,
longitude.value]).addTo(map);
                    }
                });
            }
        });
    };
    if (isPointOutsideBounds) {
        latitude.value = '';
        longitude.value = '';
        regionInput.value = ''; // Clear the region input if the point
is outside polygons
        marker.remove();
        triggerButtonDisable();
    };
};


// 2nd layer reading GeoJSON data
```

```
fetch(geojsonPath)
    .then(response => response.json())
    .then(geoJsonData => {
       geojsonLayer = L.geoJSON(geoJsonData, {
            onEachFeature: popUp }
        );

        geojsonLayer.addTo(map);
        // map.fitBounds(geojsonLayer.getBounds());

        const transparentLayer = L.geoJSON(geoJsonData, {
            style: {
                fillOpacity: 0,
                interactive: true,
                strokeOpacity: 0,
            },
        });

        transparentLayer.addTo(map);

        map.on('click', onMapClick);

    });
```

**Κώδικας 2 – Αρχείο leaflet_map.js (JavaScript)**

```
def get_solar_data(latitude_value, longitude_value, inclination_value,
azimuth_value):

    poa_data_2020, meta, input =
pvlib.iotools.get_pvgis_hourly(latitude=latitude_value,
longitude=longitude_value, start=2020, end=2020, raddatabase= 'PVGIS-
SARAH2', components=True, surface_tilt=inclination_value,
surface_azimuth =azimuth_value, outputformat='json', usehorizon=True,
userhorizon=None, pvcalculation=False, peakpower=None,
pvtechchoice='crystSi', mountingplace ='free', loss=0, trackingtype=0,
optimal_surface_tilt=False, optimalangles =False,
url='https://re.jrc.ec.europa.eu/api/v5_2/', map_variables=True,
timeout=30)

    poa_data_2020['poa_diffuse'] = poa_data_2020['poa_sky_diffuse'] +
poa_data_2020['poa_ground_diffuse']
    poa_data_2020['poa_global'] = poa_data_2020['poa_direct'] +
poa_data_2020['poa_diffuse']

    # Convert the index (time) to a separate column named 'time'
    poa_data_2020['time'] = poa_data_2020.index.strftime('%Y-%m-%d-
%H:%M:%S')

    # Resample the data to monthly frequency and calculate the sum
    monthly_irradiance =
round(poa_data_2020['poa_global'].resample('M').sum())

      # Calculate the annual solar irradiance
    annual_irradiance = round(monthly_irradiance.sum())

    # Convert the monthly irradiance Series to an array, annual as a sum
and divide them by 1000 to convert from Wh/m2 to kWh/m2
    monthly_irradiance_array = np.array(monthly_irradiance / 1000)
```

```
    annual_irradiance = float(annual_irradiance / 1000)

    # Convert the NumPy array to a nested Python list
    monthly_irradiance_list = monthly_irradiance_array.tolist()

    # Convert the list to JSON format
    monthly_irradiance_json = json.dumps(monthly_irradiance_list)

    return monthly_irradiance_json, annual_irradiance,
monthly_irradiance_list
```

**Κώδικας 3 - Μέθοδος get_solar_data (python)**

```
function calculateAutoPower() {
  // Record the start time
  const startTime = new Date().getTime();

  // Show the progress bar at the start of the request
  $('#progressBar').css('width', '0%'); // reset progress bar
  $('#progressBar').show();
  // Get the latitude, longitude, azimuth, and tilt values from the form
  const latitudeValue = parseFloat(latitudeInput.value);
  const longitudeValue = parseFloat(longitudeInput.value);
  const azimuthValue = parseFloat(azimuthInput.value);
  const tiltValue = parseFloat(tiltInput.value);
  let panelWpValue = parseFloat(panelWpInput.value);
  let shadingInputValue = parseInt(shadingSliderValue);
  let profileValue = profileConsumptionValue;
  // from Wp to kWp
  let panelKWpValue = panelWpValue / 1000;
  const panelAreaValue = parseFloat(panelAreaInput.value);
  let panelEfficiencyValue =
parseFloat(panelEfficiencyInput.value).toFixed(1);
  //  from xx % to 0,xx %
  panelEfficiencyValue /= 100;
  const annualKWhValue = parseInt(annual_Kwh_input.value);
  const placeInstalmentValue = selectedPlaceInstalmentValue;
  let sliderMaxValue = parseFloat(slider.max);

  // Validate the parsed values
  if (isNaN(panelKWpValue) || isNaN(shadingInputValue) ||
isNaN(panelAreaValue) ) {
      throw new Error("Παρακαλώ ελέγξτε τις παραμέτρους ΦΒ Πάνελ.");
  }

  const csrfToken =
document.querySelector('[name=csrfmiddlewaretoken]').value;
  const url = '/simulation/ajax/' ;  // Make sure this matches the URL
pattern in your Django project's URLs

  // Create the data object
  let data = {
      // place's info
      latitude: latitudeValue,
      longitude: longitudeValue,
      inclination: tiltValue,
      azimuth: azimuthValue,
      place_instalment_value: placeInstalmentValue,
```

```
      shading_value: shadingInputValue,
      // panel's info
      panel_area: panelAreaValue,
      panel_efficiency: panelEfficiencyValue,
      panel_kWp_value: panelKWpValue,
      // consumption
      annual_Kwh_value: annualKWhValue,
      consumption_profile: profileValue,
      //slider power
      slider_max_value: sliderMaxValue,
  };

    console.log(data);

  const jsonData = JSON.stringify(data);

  // Make the AJAX request
  ajaxRequest = $.ajax({
    url: url,
    type: 'POST',
    data: jsonData,
    contentType: 'application/json', //Set the request content type to
JSON
    dataType: 'json', // Expect JSON response from the server
    headers: {
      'X-CSRFToken': csrfToken // Include the CSRF token (Django Safety)
    },

    xhr: function() {
      let xhr = new window.XMLHttpRequest();
      // Initialize the progress percentage
      let currentPercentage = 0;
      // Track progress events to update the progress bar
      xhr.upload.addEventListener("progress", function(event) {
        // Calculate the current percentage of completion
        currentPercentage = (event.loaded/ event.total) * 100;
          // Update the progress bar using a CSS animation
        $('#progressBar').css('animation', 'none');// Disable any
animation
        $('#progressBar').css('animation', 'fill-progress-bar 2s linear
forwards');
        // Set the width of the progress bar
        $('#progressBar').css('width', currentPercentage + '%');

      }, false);
      return xhr;
    },

    success: function(response) {
      // Handle the response
      let specialProduction = response.special_production_per_panel;
      let recommended_kWp = response.recommended_kWp;
      let minimum_PV_panels = response.minimum_PV_panels;
      let totalArea = response.total_panel_area;
      let annualProduction = response.annual_production;
      console.log("--- recommended_kWp: ", recommended_kWp, "---",
'panelkWpValue: ', panelKWpValue, 'annual_production: ',
annualProduction );
      console.log("---", 'minimum_PV_panels: ',minimum_PV_panels);
```

```
      // Update the input field with the calculated power
      $('#placeProduction').val(specialProduction);
      $('#minimumPanels').val(minimum_PV_panels);
      $('#totalArea').val(totalArea);
      $('#annualProduction').val(annualProduction);

      slider.value = recommended_kWp;
      slider_hidden_input.value = recommended_kWp;
      PV_kW_output.innerHTML = slider.value;
      battery_capacity_kwh.min = slider.value;
      battery_capacity_kwh.value = slider.value;
      autoCalculatedPower = true;
      autoCalculatedPowerNumber = recommended_kWp;
      enablePanelButton(nextButton);

      // Hide the progress bar on success
      $('#progressBar').hide();
      // Display a completion message with the actual time taken
      const finishTime =  new Date().getTime();
      const elapsedTimeSeconds = ((finishTime - startTime) /
1000).toFixed(2);
      $('#completionMessage').text(`Ολοκληρώθηκε σε
${elapsedTimeSeconds} seconds`);
      $('#completionMessage').show(); // Show the completion message

      autoPowerButton.disabled = false;
      isAutoCalculatingPower = false;
      autoPowerButton.innerHTML = 'Υπολογισμός Απαιτούμενης Ισχύος';

    },
    error: function(xhr, status, errorThrown) {
      ajaxRequest.abort();

      if (status === 'abort') {
        // Request was intentionally aborted
        console.log('Request aborted');
      } else {
        // Handle other errors
        console.error('Error:', errorThrown);
      }
      // Hide the progress bar on error
      $('#progressBar').hide();
    }
  });
}
```

**Κώδικας 4 - Μέθοδος calculateAutoPower (JavaScript)**

```
# Ajax Request Calculation,  url = simulation/ajax/
def calculate_power(request):
    # Initial assignment
    total_PV_life_years= 25
    annual_production = 0
    minimum_PV_panels = 0

    try:
        if request.method == 'POST':
            # Retrieve values from the JSON data
            data = json.loads(request.body)
```

```
            # Extract the values from the data object
            latitude_value = data.get('latitude')
            longitude_value = data.get('longitude')
            inclination_value = data.get('inclination')
            # from now on pvlib has an 180 offset for azimuth aspect
            azimuth_value = data.get('azimuth')
            # pvlib tools subtract 180 deg from the given azimuth number
            azimuth_value += 180
            place_instalment_value = data.get('place_instalment_value')
            shading_value = data.get('shading_value') # values 1 to 3
            # Calculate the percentage of degradation due to shadings
            shadings_percentage =
calculate_shade_percentage(shading_value)
            # Panel parameters
            panel_area = data.get('panel_area')
            panel_efficiency = data.get('panel_efficiency')
            panel_kWp_value = data.get('panel_kWp_value') # in kWp
            # Consumption profiles
            energy_consumption = data.get('annual_Kwh_value')
            consumption_profile_value = data.get('consumption_profile')
            slider_max_value = data.get('slider_max_value')


# Call the get_solar_data function that uses PVGIS API and retrieve
results
            monthly_irradiance_json, annual_irradiance,
monthly_irradiance_list = get_solar_data(latitude_value,
longitude_value, inclination_value, azimuth_value)
            # Store the monthly_irradiance_list in the session
            request.session['monthly_irradiance_json'] =
monthly_irradiance_json
            request.session['annual_irradiance'] = annual_irradiance
            request.session['monthly_irradiance_list'] =
monthly_irradiance_list

            # CALCULATIONS
            # Initial calculations
            special_production_per_panel = round(annual_irradiance *
panel_area * panel_efficiency * performance_degradation *
shadings_percentage)
            total_consumption = energy_consumption * total_PV_life_years
            minimum_PV_panels = round(energy_consumption /
special_production_per_panel)
            recommended_kWp  = round(minimum_PV_panels *
panel_kWp_value, 1)
            annual_production  = round(minimum_PV_panels *
special_production_per_panel)

            # If the production is degradated enough enlarge the PV
system
            loop_passes = 0
            while annual_production <= energy_consumption:
                loop_passes +=1
                minimum_PV_panels +=1
                annual_production  = round(minimum_PV_panels *
special_production_per_panel)
                print("IN WHILE LOOP: ADDED 1 more PV panel")
```

```
            recommended_kWp  = round(minimum_PV_panels * panel_kWp_value,
1)

        # check if recommended kWp is greater than house power load max
value
            if recommended_kWp >= slider_max_value:
                recommended_kWp = slider_max_value
                minimum_PV_panels = round(recommended_kWp /
panel_kWp_value)
                annual_production = round(minimum_PV_panels *
special_production_per_panel)

            if place_instalment_value == 'roof':
                total_panel_area = round(minimum_PV_panels * panel_area,
1)
            else:
                # needs more space for terrace instead of roof
        total_panel_area = round(minimum_PV_panels * panel_area  * 1.5,
1)

            _, monthly_panel_energy_produced_json,
monthly_panel_energy_produced_list =
calculate_PV_energy_produced(monthly_irradiance_list, annual_irradiance,
special_production_per_panel, minimum_PV_panels, total_panel_area)

            request.session['monthly_panel_energy_produced_list'] =
monthly_panel_energy_produced_list
            request.session['monthly_panel_energy_produced_json'] =
monthly_panel_energy_produced_json
            request.session['minimum_PV_panels'] = minimum_PV_panels
            request.session['annual_production'] = annual_production
            request.session['special_production_per_panel'] =
special_production_per_panel
            request.session['total_panel_area'] = total_panel_area

            response_data = {
                'special_production_per_panel': round
(special_production_per_panel),
                'recommended_kWp': recommended_kWp,
                'minimum_PV_panels': minimum_PV_panels,
                'total_panel_area': total_panel_area,
                'annual_production': round(annual_production),
                'monthly_panel_energy_produced_list':
monthly_panel_energy_produced_list,
                'monthly_panel_energy_produced_json':
monthly_panel_energy_produced_json,
            }

            return JsonResponse(response_data)
    except Http404: # not use bare except
        return Http404("404 Generic Error")
```

**Κώδικας 5 - Μέθοδος calculate_power (python)**

```
def calculator_form_fields_handler(request): # templates/calculator.html
    if request.method == 'POST':
        form_phase_load = PhaseLoad(request.POST)
```

```
        if form_phase_load.is_valid(): # form_district.is_valid() and
            try:
                # initialization of variables
                latitude_coords = request.POST.get('latitude')
                longitude_coords = request.POST.get('longitude')
                place_of_installment = request.POST.get('installation')
                shadings_slider_value =
request.POST.get('shadings_slider')
                # from now on pvlib has an 180 offset for azimuth aspect
                azimuth_value = float(request.POST.get('azimuth'))
                azimuth_value += 180
                inclination_PV = int(request.POST.get('inclination'))
                userPower_profile =
request.POST.get('profile_consumption')
                energy_cost = float(request.POST.get('price_kwh'))
                annual_consumption =
request.POST.get('annual_consumption')
                panel_kWp = float(request.POST.get('panel_kWp'))
                # from Wp to kWp
                panel_kWp /= 1000
                panel_efficiency =
float(request.POST.get('panel_efficiency'))
                # from xx % to 0,xx %
                panel_efficiency = round(panel_efficiency / 100, 3)
                panel_area = request.POST.get('panel_area')
                panel_cost = request.POST.get('panel_cost')
                inverter_cost = request.POST.get('inverter_cost')
                installation_cost =
request.POST.get('installation_cost')
                phase_load = request.POST.get('select_phase')

                if phase_load == "single_phase":
                    phase_loadkVA = 8
                    slider_max_value = 5
                else:
                    phase_loadkVA = 15
                    slider_max_value = 10.8

                power_kWp_method = request.POST.get('power_kWp_method')
                PV_kWp = request.POST.get('myRangeSliderHidden')
                has_storage = request.POST.get('storage')
                battery_capacity_kwh =
request.POST.get('battery_capacity_kwh')
                battery_cost = request.POST.get('battery_cost')
                noDiscountRadio = request.POST.get('discount')

                if request.POST.get('battery_cost') is None or
battery_cost == 'NaN' or battery_cost == 'αυτόματα' :
                    battery_cost = 0
                else:
                    battery_cost = int(battery_cost)

                if noDiscountRadio == 'no' or
request.POST.get('discount_percent') is None and
request.POST.get('discount_percent_battery') is None:
                    # The "no" radio button is selected
                    discount_PV = 0
                    discount_battery = 0
                elif request.POST.get('discount_percent') is None:
```

```
                    discount_PV = 0
                    discount_battery =
int(request.POST.get('discount_percent_battery'))
                elif request.POST.get('discount_percent_battery') is
None:
                    discount_battery = 0
                    discount_PV =
int(request.POST.get('discount_percent'))
                else:
                    # The "yes" discount radio button is selected
                    discount_PV =
int(request.POST.get('discount_percent'))
                    discount_battery =
int(request.POST.get('discount_percent_battery'))

                # print the variables to check
                now = datetime.now()

                if has_storage == 'with_storage':
                    print(f"Battery kWh value: {battery_capacity_kwh}")
                else:
                    battery_capacity_kwh = 0
                    print('No storage selected')
            except KeyError:
                # Handle the case where an invalid key is provided
                return HttpResponse('Invalid request parameters')

            # Storing Data in Session
            request.session['latitude_coords'] = latitude_coords
            request.session['longitude_coords'] = longitude_coords
            request.session['place_of_installment'] =
place_of_installment
            request.session['shadings_slider_value'] =
shadings_slider_value
            request.session['inclination_PV'] = inclination_PV
            # adding the pvlib azimuth offset
            request.session['azimuth_value'] = azimuth_value
            request.session['userPower_profile'] = userPower_profile
            request.session['annual_consumption'] = annual_consumption
            request.session['PV_kWp'] = PV_kWp
            request.session['has_storage'] = has_storage
            request.session['battery_capacity_kwh'] =
battery_capacity_kwh
            request.session['phase_loadkVA'] = phase_loadkVA
            request.session['phase_load'] = phase_load
            request.session['energy_cost'] = energy_cost
            request.session['panel_kWp'] = panel_kWp
            request.session['panel_efficiency'] = panel_efficiency
            request.session['panel_cost'] = panel_cost
            request.session['panel_area'] = panel_area
            request.session['inverter_cost'] = inverter_cost
            request.session['installation_cost'] = installation_cost
            request.session['discount_PV'] = discount_PV
            request.session['discount_battery'] = discount_battery
            request.session['power_kWp_method'] = power_kWp_method
            request.session['slider_max_value'] = slider_max_value
            request.session['battery_cost'] = battery_cost

            return dashboard_results(request)
```

```
        else:
            # Form is not valid, handle the error or display a message
            error_message = "Please correct the errors in the form."
            return render(request, 'simulation/calculator.html',
context={'form_phase_load': form_phase_load, 'error_message':
error_message})
    else:
        form_phase_load = PhaseLoad()
    return render(request, 'simulation/calculator.html',
context={'form_phase_load': form_phase_load})
```

**Κώδικας 6 - Μέθοδος calculator_form_fields_handler (python)**

```
def dashboard_results(request):    # simulation/templates/dashboard.html
    # Retrieving Data from Session
  if 'annual_consumption' and 'PV_kWp' in request.session:
    latitude_coords = float(request.session.get('latitude_coords'))
    longitude_coords = float(request.session.get('longitude_coords'))
    place_of_installment = request.session.get('place_of_installment')
    inclination_PV = float(request.session.get('inclination_PV'))
    shadings_slider_value=int(request.session.get('shadings_slider_value
'))
     # azimuth type is already float
    azimuth_value = request.session.get('azimuth_value')
    userPower_profile = request.session.get('userPower_profile')
    phase_load = request.session.get('phase_load')
    phase_loadkVA = int(request.session.get('phase_loadkVA'))
    energy_cost = float(request.session.get('energy_cost'))
    annual_consumption = int(request.session.get('annual_consumption'))
     # panel_kWp & panel_efficiency are already float
    panel_kWp = request.session.get('panel_kWp')
    panel_efficiency = request.session.get('panel_efficiency')
    panel_cost = int(request.session.get('panel_cost'))
    panel_area = float(request.session.get('panel_area'))
    inverter_cost = int(request.session.get('inverter_cost'))
    installation_cost = int(request.session.get('installation_cost'))
    power_kWp_method = request.session.get('power_kWp_method')
    PV_kWp = float(request.session.get('PV_kWp'))
    slider_max_value = request.session.get('slider_max_value')
    has_storage = request.session.get('has_storage')
    battery_capacity_kwh=float(request.session.get('battery_capacity_kwh
'))
    battery_cost = request.session.get('battery_cost')
    discount_PV = request.session.get('discount_PV')
    discount_battery = request.session.get('discount_battery')
     shadings_percentage=
calculate_shade_percentage(shadings_slider_value)
     request.session['shadings_percentage'] = shadings_percentage

     # important check if the power is auto calculated
     if power_kWp_method == 'auto-power':
          try:
                monthly_irradiance_json = request.session.get
('monthly_irradiance_json')
                annual_irradiance = request.session.get
('annual_irradiance')
                monthly_irradiance_list = request.session.get
('monthly_irradiance_list')
```

```
                monthly_panel_energy_produced_list = request.session.get
('monthly_panel_energy_produced_list')
                monthly_panel_energy_produced_json = request.session.get
('monthly_panel_energy_produced_json')
                number_of_panels_required = request.session.get
('minimum_PV_panels')
                annual_PV_energy_produced = request.session.get
('annual_production')
                special_production_per_panel = request.session.get
('special_production_per_panel')
                total_panel_area =
request.session.get('total_panel_area')

          except KeyError as e:
                return HttpResponse(f"Error while retrieving session
data: {str(e)}")
          except ValueError as e:
                return HttpResponse(f"Error while converting data:
{str(e)}")
     else:
            # PV kWp was manually given -> get data from PVGIS
            monthly_irradiance_json, annual_irradiance,
monthly_irradiance_list = get_solar_data (latitude_coords,
longitude_coords, inclination_PV, azimuth_value)
            request.session['monthly_irradiance_list'] =
monthly_irradiance_list
            request.session['annual_irradiance'] = annual_irradiance
            special_production_per_panel = round(annual_irradiance *
panel_area * panel_efficiency * performance_degradation *
shadings_percentage )
            number_of_panels_required = round(PV_kWp / panel_kWp )
            request.session['minimum_PV_panels'] =
number_of_panels_required
            total_panel_area = round(number_of_panels_required *
panel_area, 1)
            request.session['total_panel_area'] = total_panel_area
            PV_kWp = round(number_of_panels_required * panel_kWp,1)
            annual_PV_energy_produced,
monthly_panel_energy_produced_json, monthly_panel_energy_produced_list =
calculate_PV_energy_produced (monthly_irradiance_list,
annual_irradiance, special_production_per_panel,
number_of_panels_required, total_panel_area)
            request.session['monthly_panel_energy_produced_list'] =
monthly_panel_energy_produced_list

        # Calculate the rest variables
        self_consumption_ratio = calculate_self_consumption_ratio
(userPower_profile, annual_PV_energy_produced, has_storage,
battery_capacity_kwh, annual_consumption)
    total_investment,inverter_cost,battery_cost=
calculate_total_investment (PV_kWp, phase_load, has_storage,
battery_capacity_kwh, battery_cost, panel_cost, discount_PV,
discount_battery, number_of_panels_required, inverter_cost,
installation_cost)
        consumption_total_charges = calculate_consumption_total_charges
(annual_consumption, phase_loadkVA, energy_cost)
        request.session['consumption_total_charges'] =
consumption_total_charges
```

```
        self_consumed_energy, potential_self_consumed_energy,
exported_energy =
calculate_self_consumed_energy(annual_PV_energy_produced,
annual_consumption, self_consumption_ratio)

        total_avoided_charges = calculate_total_avoided_charges
(annual_consumption, annual_PV_energy_produced, self_consumed_energy,
potential_self_consumed_energy, phase_loadkVA, energy_cost,
consumption_total_charges, exported_energy)
        profitPercent, total_savings_potential, potential_kwh =
calculate_annual_savings (annual_consumption, annual_PV_energy_produced,
self_consumed_energy, potential_self_consumed_energy,
consumption_total_charges, total_avoided_charges, phase_loadkVA,
energy_cost)
        total_savings, total_savings_array = calculate_total_savings
(total_savings_potential)
        total_savings_array_json = json.dumps(total_savings_array)
        total_production_kwh_array, total_production_kwh =
calculate_total_production_kwh (annual_PV_energy_produced,
shadings_percentage)
        total_production_kwh_array_json = json.dumps
(total_production_kwh_array)
        payback_period, payback_year_float = calculate_payback_period
(total_investment, total_savings_potential,consumption_total_charges
        net_present_value = calculate_npv(total_investment,
total_savings)
        maintenance_cost = calculate_maintenance_cost(total_investment)
        lcoe = calculate_lcoe(total_investment, maintenance_cost ,
total_production_kwh)
        roi, annualized_roi = calculate_roi(net_present_value,
total_investment, total_savings)
        irr = calculate_irr(total_investment, total_savings_array)
        average_CO2 = round(calculate_CO2_emissions_reduced
(annual_PV_energy_produced))
        trees_planted = round(calculate_equivalent_trees_planted
(annual_PV_energy_produced))

        request.session['self_consumption_ratio'] =
self_consumption_ratio
        request.session['total_investment'] = total_investment
        request.session['inverter_cost'] = inverter_cost
        request.session['net_present_value'] = net_present_value
        request.session['maintenance_cost'] = maintenance_cost
        request.session['lcoe'] = lcoe
        request.session['roi'] = roi
        request.session['annualized_roi'] = annualized_roi
        request.session['irr'] = irr
        request.session['average_CO2'] = average_CO2
        request.session['trees_planted'] = trees_planted
        azimuth_text = transform_azimuth_text(int(azimuth_value))

        # dictionary with rendered variables
        context = {
            # form values
            'latitude_coords': latitude_coords,
            'longitude_coords': longitude_coords,
            'place_of_installment': place_of_installment,
            'panel_kWp': panel_kWp,
            'panel_efficiency': panel_efficiency,
```

```
            'panel_area': panel_area,
            'panel_cost': panel_cost,
            'azimuth_value': azimuth_value,
            'inclination_PV': int(inclination_PV),
            'azimuth_text': azimuth_text,
            'userPower_profile': userPower_profile,
            'phase_load': phase_load,
            'phase_loadkVA': phase_loadkVA,
            'annual_consumption': annual_consumption,
            'PV_kWp': PV_kWp,
            'slider_max_value': slider_max_value,
            'has_storage': has_storage,
            'battery_capacity_kwh': battery_capacity_kwh,
            'discount_PV': discount_PV,
            'discount_battery': discount_battery,
            'battery_cost': battery_cost,

            # calculated values
            'total_investment': total_investment,
            'number_of_panels_required': number_of_panels_required,
            'total_panel_area': total_panel_area,
            'payback_period': payback_period,
            'payback_year_float': payback_year_float,
            'profitPercent': profitPercent,
            'consumption_total_charges': consumption_total_charges,
            'total_avoided_charges': total_avoided_charges,
            'total_savings_potential': round(total_savings_potential),
            'potential_kwh': potential_kwh,
            'has_storage': has_storage,
            'total_savings': total_savings,
            'total_savings_array': total_savings_array,
            'total_savings_array_json': total_savings_array_json,
            'total_production_kwh': total_production_kwh,
            'total_production_kwh_array': total_production_kwh_array,
            'total_production_kwh_array_json':
total_production_kwh_array_json,
            'monthly_panel_energy_produced_list':
monthly_panel_energy_produced_list,
            'monthly_irradiance_json': monthly_irradiance_json,
            'annual_irradiance': annual_irradiance,
            'annual_PV_energy_produced': annual_PV_energy_produced,
            'monthly_panel_energy_produced_json':
monthly_panel_energy_produced_json,

            # economic models values
            'net_present_value': net_present_value,
            'lcoe': lcoe,
            'roi': roi,
            'annualized_roi': annualized_roi,
            'irr': irr,
            'average_CO2': average_CO2,
            'trees_planted': trees_planted,
        }
        try:
            result = 'simulation/dashboard.html'
            now = datetime.now()
            return render(request, result, context)
        except Http404:
            return Http404("404 Generic Error")
```

```
    else:
        try:
            result = 'simulation/dashboard_empty.html'
            return render(request, result)
        except Http404:      # not use bare except
            return Http404("404 Generic Error")
```

**Κώδικας 7 - Μέθοδος dashboard_results (python)**

```
// DASHBOARD CHART 1
const monthValues =
['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC'
];


let myChart1 = new Chart("myChart1", {
  type: "line",
  data: {
    labels: monthValues,
    datasets: [ {
      label: "Μηνιαία Ηλιακή Ακτινοβολία (kWh/m²) (2020)",
      data: monthIrradianceArray,
      backgroundColor: "#ed4901",
      borderColor: "rgba(91, 186, 210,0.3)",
      fill: false
    }, {
      label: "Μηνιαία Παραγωγή (kWh/ΦΒ πάνελ)",
      data: monthlyPanelEnergy,
      backgroundColor: "#A457F2",
      borderColor: "rgba(186, 204, 209,0.4)",
      fill: false
    }]
  },
  options: {
      title: {
      display: true,
      fontColor: "#f2f2f2",
      text: "Ετήσια παραγωγή Ενέργειας (kWh)",
      fontSize: 18
      },
      scales: {
      yAxes: [{ticks: { fontColor: "#f2f2f2" }}],
      xAxes: [{ticks: { fontColor: "#f2f2f2" },
               scaleLabel: { display: true,labelString: "Years"}}],
    },
    legend: {
      labels: {
        fontColor: "#f2f2f2"
      }
    },
  }
});

// Store the chart instance in the globalCharts object
globalCharts["myChart1"] = myChart1;

// DASHBOARD CHART 2
```

```
const pv_lifetime =
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25];

let myChart2 = new Chart("myChart2", {
  type: "line",
  data: {
    labels: pv_lifetime,
    datasets: [ {
      label: "Συνολική Παραγωγή (kWh)",
      data: totalProductionArray,
      backgroundColor: "#f2f2f2",
      borderColor: "rgba(91, 186, 210,0.3)",
      fill: false
    }, {
      label: "Κέρδος (€)",
      data: totalSavingsArray,
      backgroundColor: "#A457F2",
      borderColor: "rgba(186, 204, 209,0.4)",
      fill: false
    }]
  },
  options: {
      title: {
      display: true,
      fontColor: "#f2f2f2",
      text: "Αποδοτικότητα ΦΒ (25 έτη)",
      fontSize: 18
      },
      scales: {
      yAxes: [{ticks: { fontColor: "#f2f2f2" }}],
      xAxes: [{ticks: { fontColor: "#f2f2f2" },
               scaleLabel: { display: true,labelString: "Years"}}],
    },
    legend: {
      labels: {
        fontColor: "#f2f2f2"
      }
    },
    plugins: [],
    animation: {
      onComplete: function () {
        const chart = this.chart;
        const ctx = chart.ctx;
        const xAxis = chart.scales["x-axis-0"];
        const yAxis = chart.scales["y-axis-0"];
        const xValue = xAxis.getPixelForValue(paybackYear);

        ctx.save();
        ctx.strokeStyle = "#e34c0cbf"; // Customize line color
        ctx.lineWidth = 2; // Customize line width
        ctx.beginPath();
        ctx.moveTo(xValue, yAxis.top);
        ctx.lineto(xValue, yAxis.bottom);
        ctx.stroke();

        // Draw a label next to the line
        ctx.fillStyle = "#f2f2f2"; // Customize label color
        ctx.font = "12px Arial"; // Customize label font
        ctx.textAlign = "left";
```

```
        ctx.textBaseline = "middle";
        const label = "Έτος Απόσβεσης"; // Your label text
        const labelX = xValue + 10; // Adjust the X position of the
label
        const labelY = (yAxis.top + yAxis.bottom) / 2 + 10; // Center
the label vertically and move it down by 10 pixels
        ctx.fillText(label, labelX, labelY);

        ctx.restore();
      },
    },
  }
});

// Store the chart instance in the globalCharts object
globalCharts["myChart2"] = myChart2;

// DASHBOARD CHART 3
var zValues = ["Απόδοση Επένδυσης(ROI) %", "Καθαρό Κέρδος (NPV) (€)", "
Σταθμισμένο Κόστος Ενέργειας(LCOE)€", "Εσωτερικός βαθμός απόδοση (IRR)
%", "Ετήσια Απόδοση Επένδυσης (aROI) %"];
var rValues = [roi, npv, lcoe, irr, annual_roi];
var barColors = [
  "#5BBAD2",
  "#A457F2",
  "#393052",
  "#ed4901",
  "#e8c3b9"
];

let myChart3 = new Chart("myChart3", {
  type: "pie",
  data: {
    labels: zValues,
    datasets: [{
      backgroundColor: barColors,
      data: rValues
    }]
  },
  options: {
    title: {
      display: true,
      fontColor: "#f2f2f2",
      text: "Οικονομικοί Δείκτες",
      fontSize: 18
    },
    legend: {
      labels: {
        fontColor: "#f2f2f2"
      }
    },
  }
});

// Store the chart instance in the globalCharts object
globalCharts["myChart3"] = myChart3;
```

**Κώδικας 8 - chart.js (JavaScript)**

```
def calculate_total_investment(PV_kWp, phase_load, has_storage,
battery_capacity_kwh, battery_cost, panel_cost, discount_PV,
discount_battery, number_of_panels_required, inverter_cost,
installation_cost):
    average_battery_cost_per_kW = 850 # average in €
    electric_materials = 100 # average cost in €
    initial_inverter_cost = inverter_cost

    # Cost of PV system is total panels * cost
    panel_bases_cost = 90 * number_of_panels_required # bases for the
panels on roof or terrace, average cost per base
    Pv_panels_cost = round((number_of_panels_required * panel_cost) +
panel_bases_cost)

    if has_storage == "with_storage" and battery_cost == 0:
        battery_cost = round(battery_capacity_kwh *
average_battery_cost_per_kW)
    elif has_storage == "with_storage" and battery_cost > 0:
        battery_cost = battery_cost
    else:
        battery_cost = 0

    if phase_load == 'single_phase':
        energy_meter = 120
        if PV_kWp >= 2.0 and PV_kWp <= 5.0:
            inverter_cost_auto = 350 * PV_kWp
        else:
            inverter_cost_auto = 650
    else:
        energy_meter = 250
        if PV_kWp >= 1.0 and PV_kWp <= 6.1:
            inverter_cost_auto = PV_kWp * (650 - (PV_kWp * 50) )
        elif PV_kWp > 6.1 and PV_kWp <= 10.8:
            inverter_cost_auto = PV_kWp * (250 + (20 * (10.8 - PV_kWp) )
)
        else:
            inverter_cost_auto = 650

    if initial_inverter_cost == 0:
        inverter_cost = round(inverter_cost_auto)
    else:
        inverter_cost = inverter_cost


    # PV system cost without battery
    Pv_system_cost = Pv_panels_cost + installation_cost + inverter_cost
+ electric_materials + energy_meter

    # If client/user is eligible to discounts
    if discount_PV > 0:
        Pv_system_cost -= round(Pv_system_cost * (discount_PV / 100))
    if discount_battery > 0:
        battery_cost -= round(battery_cost * (discount_battery / 100))

    total_investment = round(Pv_system_cost + battery_cost)

    return total_investment, inverter_cost, battery_cost
```

**Κώδικας 9 - Μέθοδος calculate_total_investment (python)**

```
def calculate_PV_energy_produced(monthly_irradiance_list,
annual_irradiance, special_production_per_panel,
number_of_panels_required, total_panel_area):
    monthly_panel_energy_produced_list = []
    annual_PV_energy_produced = round( number_of_panels_required *
special_production_per_panel)

    for irradiance_month in monthly_irradiance_list:
        monthly_energy_produced = round(irradiance_month /
(total_panel_area))
        monthly_panel_energy_produced_list.append(monthly_energy_produce
d)

    monthly_panel_energy_produced_json =
json.dumps(monthly_panel_energy_produced_list)

    return annual_PV_energy_produced,
monthly_panel_energy_produced_json, monthly_panel_energy_produced_list
```

**Κώδικας 10 - Μέθοδος calculate_PV_energy_produced (python)**

```
def calculate_annual_savings(annual_consumption,
annual_PV_energy_produced,
self_consumed_energy,potential_self_consumed_energy,consumption_total_ch
arges, total_avoided_charges, phase_loadkVA, energy_cost):

    if total_avoided_charges == 0:
        total_avoided_charges = 0
        total_savings_potential = 0
        potential_kwh = 0
    else:
        # Again self_consumed_energy >= annual_consumption, is valid
when the potential self consumed is greater than consumption
        if  annual_PV_energy_produced > annual_consumption and
self_consumed_energy >= annual_consumption:
            exported_energy_to_grid = potential_self_consumed_energy -
annual_consumption
            total_savings_potential = round(total_avoided_charges)
            potential_kwh = round(exported_energy_to_grid)
        elif annual_PV_energy_produced > annual_consumption and
self_consumed_energy < annual_consumption:
            exported_energy_to_grid = annual_PV_energy_produced -
self_consumed_energy
            total_savings_potential = round(total_avoided_charges)
            potential_kwh = round(exported_energy_to_grid)
        else: # annual_PV_energy_produced < annual_consumption
            exported_energy_to_grid = annual_PV_energy_produced -
self_consumed_energy
            total_savings_potential = round(total_avoided_charges)
            potential_kwh = round(exported_energy_to_grid)

    if total_savings_potential <= 0:
        profitPercent = 0
    else:
        profitPercent =  min(round(total_savings_potential /
consumption_total_charges * 100, 1),100)

    return profitPercent, total_savings_potential, potential_kwh
```

**Κώδικας 11 - Μέθοδος calculate_annual_savings (python)**

```
def calculate_payback_period(total_investment, total_savings_potential,
consumption_total_charges):
    if total_savings_potential == 0:
        payback_period = 0
        payback_year_float = 0
    else:
        years_to_overcome_investment = 0
        total_savings = []

        # Calculate years and months independently from that point
        if total_savings_potential > (consumption_total_charges *
(1+annual_degradation_production) ):
            while sum(total_savings) <= total_investment:
                savings = consumption_total_charges
                total_savings.append( savings )
                years_to_overcome_investment += 1

            subtraction = sum(total_savings) - total_investment
            monthly_savings = total_savings
[years_to_overcome_investment - 2] /12
            years = years_to_overcome_investment - 1
            months = round(12 - (subtraction/monthly_savings) )
            payback_year_float  = round(years + (months / 13),2)

            if months == 1:
                payback_period = f"{years} έτη & {months} μήνας"
            else:
                payback_period = f"{years} έτη & {months} μήνες"
        else:
            total_savings_potential = total_savings_potential

            # Find the point where savings exceed investment
            while sum(total_savings) <= total_investment:
                try:
                    result = total_savings_potential / (1+
(annual_degradation_production ** years_to_overcome_investment) )
                    # Check if the result is very small (close to zero)
                    if abs(result) < 1e-1:  # adjust threshold
                        payback_period = "0"
                        payback_year_float = 0
                        break  # Break the loop if the result ~0
                    total_savings.append( result )
                    years_to_overcome_investment += 1

                except OverflowError:
                    # Handle the overflow error here, error message
                    payback_period = "Calculation error: Result too
large"
                    payback_year_float = 0

                subtraction = sum(total_savings) - total_investment
                monthly_savings = total_savings
[years_to_overcome_investment - 2] /12
                years = years_to_overcome_investment - 1
                months = round(12 - (subtraction/monthly_savings) )
                payback_year_float = round(years + (months / 13),2)

                if months == 1:
                    payback_period = f"{years} έτη & {months} μήνας"
```

```
                else:
                    payback_period = f"{years} έτη & {months} μήνες"

    return payback_period, payback_year_float
```

**Κώδικας 12 - Μέθοδος calculate_payback_period (python)**

```
def calculate_total_production_kwh(annual_PV_energy_produced,
shadings_percentage):
    # 1st year the total production is the annual_PV_energy_produced
    total_production_kwh = annual_PV_energy_produced
    total_production_kwh_array =[0, total_production_kwh]

    for i in range(1, 25):
        total_production_kwh += annual_PV_energy_produced *
shadings_percentage / (1 + (annual_degradation_production * i))
        total_production_kwh_array.append(round(total_production_kwh))


    return total_production_kwh_array, round(total_production_kwh
```

**Κώδικας 13 - Μέθοδος calculate_total_production_kwh (python)**

```
def calculate_npv(total_investment, total_savings):
    # Calculate the net present value without cash flow
    # based on the total savings and the total investment
    # Return the result
    if total_savings == 0:
        return 0
    else:
        return total_savings - total_investment
```

**Κώδικας 14 - Μέθοδος calculate_npv (python)**

```
def calculate_roi(net_present_value, total_investment, total_savings):
    # Calculate the return on investment
    # Return the result
    if total_savings == 0:
        roi = 0
        annualized_roi = 0
        return roi, annualized_roi
    else:
        roi = round(net_present_value / total_investment * 100, 2)
        annualized_roi = round(((total_savings / total_investment) **
(1/25) -1) *100, 2)

        return roi, annualized_roi
```

**Κώδικας 15 - Μέθοδος calculate_roi (python)**

```
def calculate_lcoe(total_investment, maintenance_cost,
total_production_kwh):
    # Calculate the levelized cost of electricity
    # Return the result
    if total_production_kwh == 0:
        return 0
    else:
```

```
        lcoe = round(( total_investment + maintenance_cost ) /
total_production_kwh, 3)

        return lcoe
```

**Κώδικας 16 - Μέθοδος calculate_lcoe (python)**

```
def calculate_irr(total_investment, total_savings_array):
    # Calculate the return on investment
    # Return the result
    initial_investment = -total_investment
    saving_flows = total_savings_array.copy()  # Create a copy of the
                                                 total_savings_array

    saving_flows.insert(0, initial_investment)  # Insert the initial
                                                  investment at index 0

    irr = round(npf.irr(saving_flows) * 100, 2)

    return irr
```

**Κώδικας 17 - Μέθοδος calculate_irr (python)**

```
function recalculatePvSystemProperties(){
    // Record the start time
    const startTime = new Date().getTime();

    //   Handle the modified input
    numberPanelsDashboardValue = parseInt(numberPanelsDashboard.value);
    // Validate the parsed values
    if (isNaN(numberPanelsDashboardValue)) {
        throw new Error("Παρακαλώ ελέγξτε τον αριθμό ΦΒ Πάνελ που
εισαγάγατε.");
    }
    const csrfToken =
document.querySelector('input[name="csrfmiddlewaretoken"]').value;
    const url = '/simulation/recalculation/' ;  // Make sure this
matches the URL pattern in your Django project's URLs

    // Create the data object
    let data = {
        // number of panels modified
        changed_number_panels: numberPanelsDashboardValue,
    };

    console.log(data);
    const jsonData = JSON.stringify(data);

  // Make the AJAX request
  ajaxRequest = $.ajax({
    url: url,
    type: 'POST',
    data: jsonData,
    contentType: 'application/json', // Set the request content type to
JSON
    dataType: 'json', // Expect JSON response from the server
    headers: {
      'X-CSRFToken': csrfToken // Include the CSRF token in the request
headers
```

```
      // When the AJAX request starts (e.g., in the beforeSend callback)
    },
    beforeSend: function() {
      // Start the rotation animation when the request begins
      $("#recalculate_button img").addClass("rotate");
    },

    success: function(response) {
      setTimeout(function () {

      // Handle the response
      let recalculatedPV = response.recalculated_pv;
      let totalInvestment = response.total_investment;
      let changed_number_panels = response.changed_number_panels;
      let newPanelArea = response.new_panel_area;
      let annualSavings = response.annual_savings;
      let profitPercent = response.profitPercent;
      let totalSavingsPotential = response.total_savings_potential;
      let annual_PV_energy_produced =
response.annual_PV_energy_produced;
      let newMonthlyPanelEnergyProducedList =
response.monthly_panel_energy_produced_list;
      let annual_consumption =  response.annual_consumption;
      let pv_kwp_max_value = response.pv_kwp_max_value;
      let potentialKwh = response.potential_kwh;
      let totalSavings = response.total_savings;
      let paybackPeriod = response.payback_period;
      let newPaybackYear = response.payback_year_float;
      // for chart2
      let newTotalProductionArray =
response.new_total_production_kwh_array;
      let newTotalSavingsArray = response.total_savings_array;
      let consumptionTotalCharges = response.consumption_total_charges;
      let netPresentValue = response.net_present_value;
      let returnOnInvest = response.new_roi;
      let levelisedCostEnergy = response.new_lcoe;
      let internalRateReturn = response.new_irr;
      let annualInternalRateReturn = response.new_annualized_roi;
      let averageCO2 = response.new_average_CO2;
      let treesPlanted = response.new_trees_planted;

      // Update the input field with the calculated power
      $('#number_panels_dashboard').val(changed_number_panels);
      // Update the PV_kWp tag
      $('#PV_kWp').text(recalculatedPV);
      // Update the total investment tag
      $('#total_investment_tag').text(totalInvestment);
      $('#total_panel_area').text(newPanelArea);
      $('#profit_tag').text(totalSavingsPotential);
      $('#profitPercent_tag').text(profitPercent);
      $('#CO2_tag').text(averageCO2);
      $('#trees_tag').text(treesPlanted);

      updateChart2Datasets(newTotalSavingsArray,
newTotalProductionArray, newPaybackYear,
newMonthlyPanelEnergyProducedList);
      updateChartProduction('myChart1',
newMonthlyPanelEnergyProducedList);
```

```
      updateChartSavings('myChart2', newTotalSavingsArray,
newTotalProductionArray, newPaybackYear);
      updateEconomicMetrics(netPresentValue, returnOnInvest,
levelisedCostEnergy, internalRateReturn, annualInternalRateReturn);
      updateChartMetrics('myChart3', netPresentValue, returnOnInvest,
levelisedCostEnergy, internalRateReturn, annualInternalRateReturn);

      if (potentialKwh == 0 && profitPercent >= 100) {
        $("#potential_savings").html('<p style="color: lightslategrey;
font-size: 12px; font-weight: 600;"><span style="color: #738725; font-
size: 12px; font-weight: 600;">Εκμηδενίσατε το ετήσιο κόστος</span>,
όμως δεν υπάρχει διαθέσιμο <br>πλεόνασμα ιδιοκαταναλισκόμενης
ενέργειας');
      } else if (totalSavingsPotential > 0 && profitPercent >= 100) {
        $("#potential_savings").html('<p style="color: #f2f2f2; font-
size: 12px; font-weight: 600;"><span style="color: green; font-size:
12px; font-weight: 600;">Πλεόνασμα ιδιοκαταναλισκόμενης
ενέργειας:<br>✓</span> έως επιπλέον <span id="potentialKwhValue">' +
potentialKwh + '</span> kWh<br>');
      } else if (annual_consumption > annual_PV_energy_produced &&
profitPercent <= 85 && recalculatedPV <= pv_kwp_max_value) {
        $("#potential_savings").html('<span style="color: red; font-
size: 12px; font-weight: 600;">Προσοχή! </span><span style="color:
lightslategray; font-size: 12px; font-weight: 600;">Απαιτείται
μεγαλύτερο ΦΒ <br>σύστημα για κάλυψη ετήσιας κατανάλωσης.</span>');
      } else if ((profitPercent > 85 && profitPercent < 100 &&
recalculatedPV < pv_kwp_max_value) || (recalculatedPV <
pv_kwp_max_value)) {
        $("#potential_savings").html('<span style="color: lightskyblue;
font-size: 12px; font-weight: 600;">Με μικρές αλλαγές, με προσθήκη
μπαταρίας ή με λίγο <br>μεγαλύτερο ΦΒ σύστημα, μπορείτε να εκμηδενίσετε
το<br>ετήσιο κόστος!!</span>');
      } else if ((recalculatedPV == pv_kwp_max_value || recalculatedPV
== pv_kwp_max_value) && annual_consumption > annual_PV_energy_produced)
{
        $("#potential_savings").html('<span style="color:
lightslategrey; font-size: 12px; font-weight: 600;">Η κατανάλωσή σας
υπερβαίνει την παραγόμενη ενέργεια <br>και το ΦΒ σύστημα έχει τη μέγιστη
τιμή kWp. Δοκιμάστε <br>αλλαγές σε κλίση, αζιμούθιο ή
σκιάσεις.</span>');
      } else if (potentialKwh == 0) {
        $("#potential_savings").html('<span style="color:
lightslategrey; font-size: 12px; font-weight: 600;">Περιθώριο
κέρδους:<br></span> Δεν υπάρχει διαθέσιμο πλεόνασμα ιδιοκαταναλισκόμενης
ενέργειας');
      };

      if (profitPercent == 100) {
        $("#profitPercentLabel").css("color", "green");
        $("#profitPercent_tag").text(profitPercent || '0');
        $("#profitPercent_tag").append("% ✓");
        $("#annualCostLabel").css({
          "color": "lightskyblue",
          "font-size": "12px"
        });
        $("#annualCostLabel").text("Ετήσιο Κόστος Ρεύματος: " +
(consumptionTotalCharges || '0') + " €");
      } else if (profitPercent > 85 && profitPercent < 100) {
        $("#profitPercentLabel").css("color", "#738725");
```

```
        $("#profitPercent_tag").text(profitPercent || '0');
        $("#profitPercent_tag").append(" %");
        $("#annualCostLabel").css({
          "color": "lightskyblue",
          "font-size": "12px"
        });
        $("#annualCostLabel").text("Ετήσιο Κόστος Ρεύματος: " +
(consumptionTotalCharges || '0') + " €");
      } else {
        $("#profitPercentLabel").css("color", "grey");
        $("#profitPercent_tag").text(profitPercent || '0');
        $("#profitPercent_tag").append(" %");
        $("#annualCostLabel").css({
          "color": "lightskyblue",
          "font-size": "12px"
        });
        $("#annualCostLabel").text("Ετήσιο Κόστος Ρεύματος: " +
(consumptionTotalCharges || '0') + " €");
      }

      if (paybackPeriod == 0 || paybackPeriod =="0") {
        $("#paybackPeriod_tag").css("color", "lightslategrey");
        $('#paybackPeriod_tag').html('&#8734; <br> Η περίοδος απόσβεσης
υπερβαίνει τα όρια <br>υπολογισμού.');
      }else if (newPaybackYear >= 100){
        $("#paybackPeriod_tag").css("color", "lightslategrey");
        $('#paybackPeriod_tag').html('&#8734; <br> Η περίοδος απόσβασης
υπερβαίνει τα 100 έτη');
      }else {
        $("#paybackPeriod_tag").css("color", "#f2f2f2");
        $('#paybackPeriod_tag').text(paybackPeriod || '0');
      }

      // When the AJAX request completes (success or error callback)
      $("#recalculate_button img").removeClass("rotate");
      // Handle your AJAX success here
        // ...
      }, 1000); // 1-second delay

      console.log("Request successfully completed. Data updated")
    },
    error: function(xhr, status, errorThrown) {
      ajaxRequest.abort();
      // When the AJAX request completes (e.g., in the success or error
callback)
      $("##recalculate_button img").removeClass("rotate");
    }
  });
};
```

**Κώδικας 18 - Μέθοδος recalculatePvSystemProperties (JavaScript)**

```
def recalculate_pv_system_properties(request):
    annual_consumption = int(request.session.get('annual_consumption'))
    monthly_irradiance_list =
request.session.get('monthly_irradiance_list')
    annual_irradiance = request.session.get('annual_irradiance')
    self_consumption_ratio =
request.session.get('self_consumption_ratio')
    phase_loadkVA = int(request.session.get('phase_loadkVA'))
```

```
    energy_cost = float(request.session.get('energy_cost'))
    consumption_total_charges =
request.session.get('consumption_total_charges')
    shadings_percentage = request.session.get('shadings_percentage')
    total_panel_area = float(request.session.get('total_panel_area'))
    net_present_value = request.session.get('net_present_value')

    try:
        if request.method == 'POST':
            # Retrieve values from the JSON data and the session
            data = json.loads(request.body)

            changed_number_panels =
int(data.get('changed_number_panels'))
            panel_kWp_value = request.session.get('panel_kWp')
            panel_area = float(request.session.get('panel_area'))
            panel_cost = int(request.session.get('panel_cost'))
            special_production_per_panel =
request.session.get('special_production_per_panel')
            total_investment = request.session.get('total_investment')
            previous_pv_panels =
request.session.get('minimum_PV_panels')
            phase_load = request.session.get('phase_load')
            pv_kwp_max_value = 5 if phase_load == "single_phase" else
10.8

            if changed_number_panels > previous_pv_panels:
                number_panels_difference = changed_number_panels -
previous_pv_panels
                added_panels_bases_cost = (number_panels_difference *
90)
                new_total_investment = total_investment +
(number_panels_difference * panel_cost) + added_panels_bases_cost
                new_panel_area = round(total_panel_area +
(number_panels_difference*panel_area),1)
            else:
                number_panels_difference = previous_pv_panels -
changed_number_panels
                added_panels_bases_cost = (number_panels_difference *
90)
                new_total_investment = total_investment -
(number_panels_difference * panel_cost) - added_panels_bases_cost
                new_panel_area = round(total_panel_area -
(number_panels_difference*panel_area),1)

            recalculated_pv = round(changed_number_panels *
panel_kWp_value, 1)
            # Use a while loop to limit recalculated_pv
            while recalculated_pv > pv_kwp_max_value:
                changed_number_panels -= 1
                recalculated_pv = round(changed_number_panels *
panel_kWp_value, 1)

            request.session['PV_kWp'] = recalculated_pv

        annual_PV_energy_produced, monthly_panel_energy_produced_json,
monthly_panel_energy_produced_list = calculate_PV_energy_produced
(monthly_irradiance_list, annual_irradiance,
special_production_per_panel, changed_number_panels, new_panel_area)
```

```
        self_consumed_energy, potential_self_consumed_energy,
exported_energy = calculate_self_consumed_energy
(annual_PV_energy_produced, annual_consumption, self_consumption_ratio)
        total_avoided_charges = calculate_total_avoided_charges
(annual_consumption, annual_PV_energy_produced, self_consumed_energy,
potential_self_consumed_energy, phase_loadkVA, energy_cost,
consumption_total_charges, exported_energy)
        annual_savings = calculate_annual_savings(annual_consumption,
annual_PV_energy_produced, self_consumed_energy,
potential_self_consumed_energy,consumption_total_charges,
total_avoided_charges, phase_loadkVA, energy_cost)
        profitPercent, total_savings_potential, potential_kwh =
calculate_annual_savings (annual_consumption, annual_PV_energy_produced,
self_consumed_energy, potential_self_consumed_energy,
consumption_total_charges, total_avoided_charges, phase_loadkVA,
energy_cost)
        total_savings, total_savings_array = calculate_total_savings
(total_savings_potential)
        payback_period, payback_year_float = calculate_payback_period
(new_total_investment, total_savings_potential,
consumption_total_charges)
        new_total_production_kwh_array, new_total_production_kwh =
calculate_total_production_kwh (annual_PV_energy_produced,
shadings_percentage)
        net_present_value = calculate_npv(new_total_investment,
total_savings)
        new_maintenance_cost = calculate_maintenance_cost
(new_total_investment)
        new_lcoe = calculate_lcoe(new_total_investment,
new_maintenance_cost , new_total_production_kwh)
        new_roi, new_annualized_roi = calculate_roi(net_present_value,
new_total_investment, total_savings)
        new_irr = calculate_irr(new_total_investment,
total_savings_array)
        new_average_CO2 = round(calculate_CO2_emissions_reduced
(annual_PV_energy_produced))
        new_trees_planted = round(calculate_equivalent_trees_planted
(annual_PV_energy_produced))

        response_data = {
            'recalculated_pv': recalculated_pv,
            'changed_number_panels': changed_number_panels,
            'total_investment': new_total_investment,
            'new_panel_area': new_panel_area,
            'annual_savings': annual_savings,
            'profitPercent': profitPercent,
            'total_savings_potential': total_savings_potential,
            'potential_kwh': potential_kwh,
            'total_savings': total_savings,
            'total_savings_array': total_savings_array,
            'payback_period': payback_period,
            'payback_year_float': payback_year_float,
            'new_total_production_kwh_array':
new_total_production_kwh_array,
            'annual_PV_energy_produced': annual_PV_energy_produced,
            'monthly_panel_energy_produced_list':
monthly_panel_energy_produced_list,
            'annual_consumption': annual_consumption,
            'pv_kwp_max_value': pv_kwp_max_value,
```

```
            'consumption_total_charges': consumption_total_charges,
            'net_present_value': net_present_value,
            'new_lcoe': new_lcoe,
            'new_roi': new_roi,
            'new_annualized_roi': new_annualized_roi,
            'new_irr': new_irr,
            'new_average_CO2': new_average_CO2,
            'new_trees_planted': new_trees_planted,
        }
         return JsonResponse(response_data)
    except Http404:      # not use bare except
        return Http404("404 Generic Error")
```

**Κώδικας 19 - Μέθοδος recalculate_pv_system_properties (python)**

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.