ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων & Συστημάτων
Αποφάσεων

Ανάπτυξη και Εφαρμογή Ψηφιακού Διδύμου (Digital Twin) για Παρακολούθηση και Βελτιστοποίηση Κτηριακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Πιττάκης

Επιβλέπων: Ευάγγελος Μαρινάκης
Επίκουρος Καθηγητής, Ε.Μ.Π

Αθήνα, Σεπτέμβριος, 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων & Συστημάτων
Αποφάσεων

Ανάπτυξη και Εφαρμογή Ψηφιακού Διδύμου (Digital Twin) για Παρακολούθηση και Βελτιστοποίηση Κτηριακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Πιττάκης

Επιβλέπων: Ευάγγελος Μαρινάκης
Επίκουρος Καθηγητής, Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17η Οκτωβρίου 2025

.....	.....	.....
Ευάγγελος Μαρινάκης Επίκουρος Καθηγητής, Ε.Μ.Π	Ευάγγελος Σπηλιώτης Επίκουρος Καθηγητής, Ε.Μ.Π	Δημήτριος Ασκούνης Καθηγητής, Ε.Μ.Π

Αθήνα, Σεπτέμβριος, 2025

. . . . . .

Γεώργιος Πιττάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εργασία αυτή παρουσιάζει τον σχεδιασμό και την ανάπτυξη μίας ολοκληρωμένης πλατφόρμας *Digital Twin* για την παρακολούθηση, ανάλυση και οπτικοποίηση δεδομένων αισθητήρων σε κτίρια. Το σύστημα βασίζεται σε αρχιτεκτονική τριών επιπέδων (sensors–backend–frontend) και αξιοποιεί τεχνολογίες όπως MQTT για τη συλλογή δεδομένων, PostgreSQL για την αποθήκευση, FastAPI για την υλοποίηση RESTful υπηρεσιών και React/Three.js για το γραφικό περιβάλλον χρήστη και την τρισδιάστατη απεικόνιση.

Η πλατφόρμα επιτρέπει την απεικόνιση μετρήσεων σε πραγματικό χρόνο (ενέργεια, θερμική άνεση, ποιότητα αέρα, κατάσταση θερμαντικών σωμάτων), την ανίχνευση ανωμαλιών (phase imbalance, Isolation Forest), καθώς και την πρόβλεψη κατανάλωσης μέσω LSTM μοντέλου. Επιπλέον, υλοποιεί τον υπολογισμό PMV/PPD σύμφωνα με το πρότυπο ISO 7730:2005, ενσωματώνοντας δεδομένα καιρού για πιο ρεαλιστικές παραμέτρους. Ένα πρόσθετο χαρακτηριστικό είναι ο AI Βοηθός, ο οποίος μέσω του ChatGPT API απαντά σε ερωτήσεις σχετικές με την έννοια του Ψηφιακού Διδύμου και τους αισθητήρες, παρέχοντας υποστηρικτική γνώση χωρίς πρόσβαση σε δεδομένα χρηστών.

Η εργασία περιγράφει επίσης τα αντίστοιχα διαγράμματα χρήσης, κλάσεων και αλληλεπίδρασης, τα οποία συμβάλλουν στη σαφή κατανόηση της αρχιτεκτονικής και των ροών δεδομένων. Η λύση που παρουσιάζεται είναι επεκτάσιμη, ασφαλής (JWT authentication), και προσφέρει τη δυνατότητα περαιτέρω αξιοποίησης σε περιβάλλοντα έξυπνων κτιρίων και ενεργειακής αποδοτικότητας.

Λέξεις-κλειδιά: Ψηφιακό Δίδυμο, Αισθητήρες, Κτηριακά Συστήματα, Παρακολούθηση, Βελτιστοποίηση, Ενεργειακή Αποδοτικότητα, Έξυπνα Κτήρια, Διαχείριση Ενέργειας, MQTT, LSTM, PMV, PPD, AI Βοηθός, Ανομαλίες

Abstract

This thesis presents the design and implementation of a comprehensive Digital Twin platform for monitoring, analyzing, and visualizing building sensor data. The system follows a three-tier architecture (sensors–backend–frontend) and employs modern technologies such as MQTT for data ingestion, PostgreSQL for storage, FastAPI for RESTful services, and React/Three.js for the user interface and 3D visualization.

The platform enables real-time visualization of sensor data (energy, thermal comfort, air quality, heater status), anomaly detection (phase imbalance, Isolation Forest), and consumption forecasting using an LSTM model. Furthermore, it implements the calculation of PMV/PPD according to ISO 7730:2005, incorporating weather data to derive realistic comfort parameters. An additional feature is the AI Assistant, which leverages the ChatGPT API to answer questions related to Digital Twin concepts and sensors, providing explanatory support without accessing user data.

The work also includes the corresponding use case, class, and interaction diagrams, which help clarify the system's architecture and data flows. The proposed solution is extensible, secure (JWT authentication), and can be further applied in smart building and energy efficiency scenarios.

KeyWords: Digital Twin, Building Systems, Monitoring, Optimization, Energy Efficiency, Smart Buildings, Energy Management, Sensors, FastAPI, MQTT, LSTM, PMV, PPD, AI Assistant, Anomalies

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον Καθηγητή του Εθνικού Μετσόβιου Πολυτεχνείου, κύριο Ευάγγελο Μαρινάκη, για την ανάθεση της παρούσας διπλωματικής εργασίας, αλλά και για την άψογη συνεργασία που είχαμε καθ' όλη τη διάρκεια της εκπόνησής της.

Επίσης, οφείλω ένα μεγάλο ευχαριστώ στους κυρίους Ιωάννη Παπία και Φίλιππο Σερέπα, οι οποίοι αποτέλεσαν βασικό πυλώνα για την ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Ασφαλώς, δεν θα μπορούσα να παραλείψω από αυτές τις ευχαριστίες τόσο τους γονείς μου, που με στηρίζουν από την πρώτη μέρα για να καταφέρω να εκπληρώσω τους στόχους μου, όσο και τους δικούς μου ανθρώπους και τους φίλους μου, που με στήριξαν καθ' όλη τη διάρκεια των σπουδών μου και συνέβαλαν σημαντικά σε αυτό το ταξίδι.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

Ένας από τους σημαντικότερους παράγοντες που επηρεάζουν το περιβάλλον αλλά και την οικονομία στον τομέα των κτηριακών εγκαταστάσεων είναι η ενεργειακή κατανάλωση. Τα κτηριακά συστήματα έχουν μεγάλο ποσοστό ευθύνης για την συνολική κατανάλωση ενέργειας και εκπομπών διοξειδίου του άνθρακα[3, 4]. Το γεγονός αυτό καθιστά αναγκαία την εύρεση λύσεων ώστε να μειωθεί αυτή η σπατάλη ενέργειας και να βελτιστοποιηθεί η απόδοσή τους. Στα πλαίσια της μετάβασης σε έξυπνες πόλεις και κτηριακές εγκαταστάσεις, η παρακολούθηση, διαχείριση και βελτιστοποίηση των κτηρίων έχουν κρίσιμη σημασία[5, 6, 7].

Τα τελευταία χρόνια η ανάπτυξη του Διαδικτύου των πραγμάτων (IoT) έχει επιφέρει σημαντικές αλλαγές στον τρόπο με τον οποίο αλληλεπιδρούμε και παρακολουθούμε τα κτηριακά συστήματα[8]. Μέσω αισθητήρων και μετρητικών συστημάτων, μπορούν να συλλέγονται δεδομένα σε πραγματικό χρόνο για ποικίλες παραμέτρους όπως η θερμοκρασία, η υγρασία, η κατανάλωση ηλεκτρικής ενέργειας ή η ποιότητα του αέρα[9]. Οι πληροφορίες αυτές προσφέρουν μία πιο ολοκληρωμένη και σαφή εικόνα για τη συμπεριφορά ενός κτηρίου, δημιουργώντας τις απαραίτητες προϋποθέσεις για καλύτερη διαχείριση των πόρων αλλά και για ακριβείς παρεμβάσεις[10, 11, 12].

Η ανάλυση των δεδομένων αυτών δεν περιορίζεται μόνο στην ενεργειακή κατανάλωση, αλλά επεκτείνεται και στην αξιολόγηση των συνθηκών άνεσης των χρηστών. Οι δείκτες θερμικής άνεσης επιτρέπουν την ποσοτική εκτίμηση του βαθμού ικανοποίησης των ατόμων που βρίσκονται στο κτήριο αξιοποιόντας τις συνθήκες του εσωτερικού περιβάλλοντος [13, 14]. Η ενσωμάτωση τέτοιων υπολογιστικών μοντέλων σε ένα ενιαίο σύστημα παρακολούθησης μας πρόσφέρει καλύτερη ισορροπία μεταξύ ενεργειακής αποδοτικότητας και άνεσης.

Παρόλα αυτά, τα δεδομένα που παράγονται από τα σύγχρονα κτηριακά συστήματα επιφέρουν νέες [15, 16]. Η επεξεργασία και η κατανόηση των δεδομένων αυτών απαιτούν ολοκληρωμένες πλατφόρμες που να μπορούν να συνδυάζουν την ανάλυση των μετρήσεων των αισθητήρων για να ανιχνεύσουν προβλήματα και ανωμαλίες[17, 18, 19] αλλά και να υποστηρίζουν εργαλεία τρισδιάστατης αναπαράστασης. Έτσι, γεννιέται η ανάγκη για μία προσέγγιση η οποία θα μπορεί να υποστηρίξει τη σύνδεση ανάμεσα στο φυσικό κτήριο και την ψηφιακή του αναπαράσταση.

Εδώ έρχεται το Ψηφιακό Δίδυμο (Digital Twin). Το Ψηφιακό Δίδυμο αποτελεί μια δυναμική, ψηφιακή αναπαράσταση ενός πραγματικού κτηρίου, η οποία έχει τη δυνατότητα να ενημερώνεται συνεχώς με δεδομένα που συλλέγονται σε πραγματικό χρόνο[20]. Μέσω αυτής της σύνδεσης, είναι εφικτή η προσομοίωση του κτηρίου, η βελτιστοποίηση της λειτουργείας του, η ανίχνευση προβλημάτων και ανωμαλιών και η πρόβλεψη μελλοντικών καταστάσεων όπως είναι η ανάγκη για συντήρηση ή η κατανάλωση ενέργειας.

1.1 Σκοπός και Αντικείμενο της Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και εφαρμογή ενός *Ψηφιακού Διδύμου* (*Digital Twin*) για την παρακολούθηση, προσομοίωση και βελτιστοποίηση κτηριακών συστημάτων. Το Ψηφιακό Δίδυμο είναι η δυναμική ψηφιακή αναπαράσταση ενός φυσικού συστήματος. Η αναπαράσταση αυτή ενημερώνεται και ανανεώνεται συνεχώς σε πραγματικό χρόνο, με δεδομένα που συλλέγονται από αισθητήρες *IoT*.

Το αντικείμενο της εργασίας είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος το οποίο θα είναι σε θέση να συλλέγει και να επεξεργάζεται δεδομένα από αισθητήρες οι οποίοι είναι τοποθετημένοι σε κάποιο πραγματικό κτήριο. Πέρα από αυτό θα ενσωματώνει τα δεδομένα αυτά στο τρισδιάστατο μοντέλο του κτηρίου για σκοπούς απεικόνισης και προσομοίωσης. Ακόμα,

θα είναι σε θέση να αναλύει τη λειτουργία των κτηριακών συστημάτων και να εξάγει δείκτες απόδοσης. Τέλος, θα διευκολύνει τη βελτιστοποίηση της λειτουργίας του κτηρίου ανιχνεύοντας προβλήματα σε θέματα όπως είναι η εξοικονόμηση ενέργειας και η πρόβλεψη συντήρησης (predictive maintenance).

Η εργασία εστιάζει στην εφαρμογή του Ψηφιακού Διδύμου σε πραγματικές συνθήκες, καθιστώντας το εργαλείο χρήσιμο για την πρόβλεψη και αποφυγή προβλημάτων και τη βελτίωση της αποδοτικότητας των κτηριακών εγκαταστάσεων.

1.2 Δομή της Εργασίας

Η εργασία είναι οργανωμένη σε κεφάλαια με στόχο την επίτευξη της συνοχής στον τρόπο που παρουσιάζονται το θεωρητικό υπόβαθρο, ο σχεδιασμός και η υλοποίηση του συστήματος, καθώς και τα αποτελέσματα που προέκυψαν. Συγκεκριμένα:

- Στο **Κεφάλαιο 1** (Εισαγωγή) αναλύεται ο σκοπός και το αντικείμενο της εργασίας και παρουσιάζεται η δομή του κειμένου.
- Στο **Κεφάλαιο 2** (Θεωρητικό Υπόβαθρο) γίνεται ανασκόπηση στη βιβλιογραφία. Παρουσιάζονται οι έννοιες του Digital Twin, BIM, IFC και τα εργαλεία απεικόνισης (Blender). Επιπλέον, αναλύονται οι μέθοδοι ανίχνευσης ανωμαλιών σε χρονοσειρές (Isolation Forest), η αρχιτεκτονική LSTM και ο ορισμός του AI βοηθού.
- Στο **Κεφάλαιο 3** περιγράφεται η διαδικασία προετοιμασίας του IFC Μοντέλου και η εισαγωγή αισθητήρων στο Blender, με ολόκληρη τη διαδικασία από τη δημιουργία του project, την εισαγωγή τρισδιάστατου μοντέλου, μέχρι και την τοποθέτηση αισθητήρων.
- Στο **Κεφάλαιο 4** παρουσιάζεται η οπτικοποίηση των δεδομένων των αισθητήρων σε πραγματικό χρόνο στο Blender. Από τη λήψη και αντιστοίχιση δεδομένων αισθητήρων στα τρισδιάστατα αντικείμενα μέχρι την ενσωμάτωση UI overlays και εξαγωγή σε .glb για χρήση στη δικτυακή πλατφόρμα.
- Στο **Κεφάλαιο 5** περιγράφεται η σχεδίαση και υλοποίηση της διαδικτυακής πλατφόρμας. Αναλύονται οι τεχνολογίες (βάση δεδομένων, backend, frontend, containerization), παρουσιάζεται το σχήμα της βάσης δεδομένων, ο μηχανισμός λήψης δεδομένων από MQTT, ο σχεδιασμός του API και του frontend μαζί με την προσθήκη του τρισδιάστατου μοντέλου.
- Στο **Κεφάλαιο 6** παρουσιάζονται οι μέθοδοι ανάλυσης δεδομένων, προβλέψεων και ανίχνευσης ανωμαλιών. Περιγράφεται η μεθοδολογία για τον υπολογισμό των PMV και PPD, η ανίχνευση ανωμαλιών στους αισθητήρες ενέργειας, ο αλγόριθμος μηχανικής μάθησης Isolation Forest, η εκπαίδευση και ανάπτυξη του μοντέλου LSTM καθώς και ο τρόπος που εμφανίζονται και χρησιμοποιούνται στη διαδικτυακή πλατφόρμα.
- Στο **Κεφάλαιο 7** περιγράφεται η Ενσωμάτωση και Υλοποίηση του AI Βοηθού. Παρουσιάζεται η αρχιτεκτονική, η υλοποίηση στο backend και frontend, οι πολιτικές φιλτραρίσματος και ασφάλειας.
- Στο **Κεφάλαιο 8** παρουσιάζεται η αρχιτεκτονική του συστήματος μέσω διαγραμμάτων UML (Use Case Diagram, Class Diagram), αναλύονται οι σχέσεις, οι οντότητες και οι αλληλεπιδράσεις μεταξύ των στοιχείων του συστήματος.

- Στο **Κεφάλαιο 9** παρουσιάζονται τα αποτελέσματα της πλατφόρμας, με μετρήσεις σε πραγματικό χρόνο, αξιολογήσεις των αλγορίθμων και της απόδοσης του AI βοηθού.
- Στο **Κεφάλαιο 10** παραθέτονται τα συμπεράσματα της εργασίας.
- Στο **Κεφάλαιο 11** προτείνονται μελλοντικές κατευθύνσεις, με αναφορά σε περιορισμούς, δυνατότητες βελτίωσης και επεκτάσεις.

2 Θεωρητικό Υπόβαθρο

2.1 Ψηφιακό Δίδυμο (Digital Twin)

2.1.1 Ιστορική Αναδρομή

Η έννοια του Ψηφιακού Διδύμου (Digital Twin) πρωτοεμφανίστηκε στις αρχές της δεκαετίας του 2000 ως εξέλιξη προηγούμενων πρακτικών προσομοίωσης και εικονικής μοντελοποίησης. Ο όρος αποδίδεται στον Michael Grieves, ο οποίος το 2002 παρουσίασε την ιδέα στο Πανεπιστήμιο του Μίσιγκαν, στο πλαίσιο του μαθήματος Product Lifecycle Management (PLM)[21].

Η βασική ιδέα ήταν η ύπαρξη ενός εικονικού μοντέλο το οποίο θα συνοδεύει το φυσικό αντικείμενο σε όλο τον κύκλο ζωής του, επιτρέποντας συνεχή παρακολούθηση, ανάλυση και βελτιστοποίηση.

Η έννοια εξελίχθηκε σημαντικά όταν η NASA αξιοποίησε την ιδέα το 2010 για την προσομοίωση και διάγνωση συστημάτων διαστημικών σκαφών[22]. Ο John Vickers της NASA περιέγραψε το Ψηφιακό Δίδυμο ως μια «ολοκληρωμένη πολυφυσική, πολυκλιμακική, πιθανοτική προσομοίωση ενός συστήματος που χρησιμοποιεί τα καλύτερα διαθέσιμα φυσικά μοντέλα, ενημερώσεις αισθητήρων κ.λπ., για να αντικατοπτρίζει τη ζωή του αντίστοιχου δίδυμου.»[23].

Η πρόοδος και η ανάπτυξη στις τεχνολογίες αισθητήρων, Internet of Things (IoT), υπολογιστικού νέφους (Cloud Computing) και Τεχνητής Νοημοσύνης (AI) από το 2015 και μετά, έδωσε τη δυνατότητα στο Ψηφιακό Δίδυμο να μπορεί να εφαρμοστεί σε πολλούς τομείς της βιομηχανίας, όπως η αεροναυπηγική, η αυτοκινητοβιομηχανία, η ενέργεια και οι «έξυπνες πόλεις και κτήρια»[24].

2.1.2 Αρχή Λειτουργίας

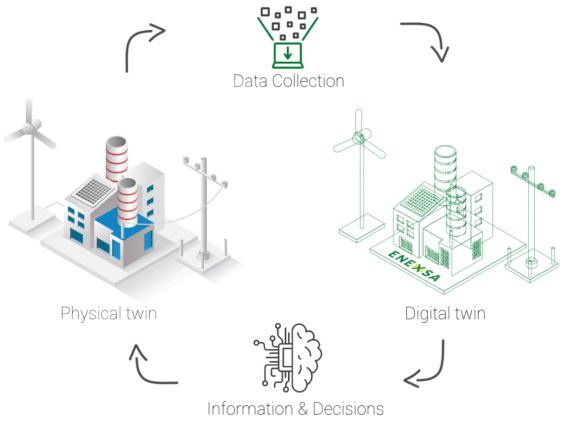
Το Ψηφιακό Δίδυμο (Digital Twin) είναι μια εικονική αναπαράσταση ενός πραγματικού φυσικού αντικειμένου, διαδικασίας, συστήματος ή ακόμη και προσώπου, η οποία σχεδιάζεται ώστε να αντικατοπτρίζει με ακρίβεια το φυσικό αντικείμενο σε όλο τον κύκλο ζωής του[21]. Χρησιμοποιεί δεδομένα σε πραγματικό χρόνο, για την προσομοίωση, παρακολούθηση και βελτιστοποίηση της απόδοσης του φυσικού αντικειμένου. Σε αντίθεση με τα στατικά μοντέλα ή τις μεμονωμένες προσομοιώσεις, το Ψηφιακό Δίδυμο υποστηρίζει αμφίδρομη αλληλεπίδραση, δηλαδή δεν λαμβάνει μόνο δεδομένα από το φυσικό αντικείμενο αλλά μπορεί και να επιδρά πίσω στο σύστημα, επηρεάζοντας τη λειτουργία του.

Ένα από τα κύρια χαρακτηριστικά του Ψηφιακού Διδύμου είναι ο βρόχος ανατροφοδότησης σε πραγματικό χρόνο (*real-time feedback loop*). Οι πληροφορίες που έρχονται από τους αισθητήρες και τα συστήματα συλλογής δεδομένων, μεταφέρονται στο μοντέλο, το οποίο αναλύει τα δεδομένα και τα προσομοιώνει. Τα αποτελέσματα μπορούν να εφαρμοστούν στο πραγματικό σύστημα για τη βελτίωση της λειτουργίας του, τη μείωση του κόστους και την πρόληψη πιθανών προβλημάτων[24].

Όπως φαίνεται στο Σχήμα 1, δεδομένα από τον φυσικό κόσμο (π.χ. αισθητήρες, IoT συσκευές, δεδομένα περιβάλλοντος) συλλέγονται και μεταφέρονται στο Ψηφιακό Δίδυμο. Στη συνέχεια, το μοντέλο πραγματοποιεί προσομοιώσεις και ανάλυση των δεδομένων παρέχοντας αποτελέσματα που μπορούν να χρησιμοποιηθούν για την πρόβλεψη, τη βελτιστοποίηση και τη λήψη αποφάσεων στο πραγματικό σύστημα.

Στον τομέα των κτηρίων, το Ψηφιακό Δίδυμο μπορεί να χρησιμοποιηθεί για την παρακολούθηση της κατανάλωσης ενέργειας, τη βελτίωση της θερμικής απόδοσης, την πρόβλεψη βλαβών σε συστήματα του κτηρίου και την πρόβλεψη συντήρηση για προληπτικούς σκοπούς.

Μέσω τεχνικών τεχνητής νοημοσύνης και προγνωστικής ανάλυσης, το Ψηφιακό Δίδυμο μπορεί να εκτιμήσει μελλοντικές καταστάσεις, επιτρέποντας την έγκαιρη λήψη αποφάσεων.



Σχήμα 1: Αναπαράσταση ενώς κτηρίου με Ψηφιακό Δίδυμο [1].

2.2 BIM

Το Building Information Modeling (BIM) είναι μια διαδικασία που βασίζεται στη δημιουργία και διαχείριση ψηφιακών αναπαραστάσεων των φυσικών και λειτουργικών χαρακτηριστικών ενός κτηρίου ή μίας υποδομής. Το BIM δεν είναι απλώς ένα τρισδιάστατο μοντέλο (*3D model*), αλλά ένα έξυπνο μοντέλο δεδομένων που ενσωματώνει γεωμετρικές πληροφορίες, υλικά, τεχνικές προδιαγραφές και δεδομένα λειτουργίας του έργου σε όλο τον κύκλο ζωής του.

Η φιλοσοφία του BIM στοχεύει στην καλύτερη συνεργασία μεταξύ αρχιτεκτόνων, μηχανικών, κατασκευαστών και διαχειριστών εγκαταστάσεων, προσφέροντας ένα κοινό περιβάλλον εργασίας και ανταλλαγής δεδομένων. Μέσω του BIM, οι εμπλεκόμενοι φορείς μπορούν να εντοπίσουν πιθανά προβλήματα πριν την κατασκευή, να βελτιώσουν το σχεδιασμό και να μειώσουν το κόστος και τον χρόνο ολοκλήρωσης του έργου.

Η εφαρμογή του BIM σε συνδυασμό με τεχνολογίες όπως οι αισθητήρες IoT και τα εργαλεία προσομοίωσης, μπορεί να αποτελέσει τη βάση για την ανάπτυξη ενός πλήρους Ψηφιακού Διδύμου, μετατρέποντας το στατικό μοντέλο σε δυναμική πλατφόρμα παρακολούθησης και βελτιστοποίησης.

2.3 IFC

Το Industry Foundation Classes (IFC) είναι ένα ανοιχτό και τυποποιημένο σχήμα δεδομένων που χρησιμοποιείται για την ανταλλαγή και διαχείριση πληροφοριών κτηριακών έργων και υποδομών. Το IFC αναπτύχθηκε από τον οργανισμό buildingSMART International με σκοπό τη διαλειτουργικότητα (ικανότητα πληροφοριακών συστημάτων με διαφορετικά λειτουργικά συστήματα να συνδέονται και να ανταλλάσσουν πληροφορίες) (*interoperability*) μεταξύ διαφορετικών λογισμικών που χρησιμοποιούνται στον σχεδιασμό, την κατασκευή και τη διαχείριση κτηρίων.

Το IFC βασίζεται στη γλώσσα περιγραφής δεδομένων *EXPRESS* (ISO 10303-11[25]) και ορίζει μια πλούσια βιβλιοθήκη από αντικειμένων που αναπαριστούν τα δομικά στοιχεία ενός κτιρίου, τις γεωμετρικές τους ιδιότητες, τα υλικά, καθώς και τις σχέσεις μεταξύ τους. Μέσω του IFC επιτυγχάνεται η ανταλλαγή πληροφοριών μεταξύ λογισμικών BIM χωρίς απώλειες δεδομένων ή εξάρτηση από συγκεκριμένο κατασκευαστή λογισμικού.

Η χρήση του IFC είναι κρίσιμη για την υλοποίηση έργων που απαιτούν συνεργασία μεταξύ πολλών φορέων και διαφορετικών τεχνολογικών εργαλείων. Παρέχει υψηλό επίπεδο διαλειτουργικότητας, επιτρέποντας την ανταλλαγή δεδομένων μεταξύ διαφορετικών BIM πλατφορμών. Επιπλέον, διασφαλίζει την ακεραιότητα των πληροφοριών κατά την αλλαγή λογισμικών, εμποδίζοντας την απώλεια κρίσιμων δεδομένων. Τέλος, η ανεξαρτησία της αποθήκευσης των δεδομένων από κάποιο συγκεκριμένο λογισμικό καθιστά επιτυγχάνει τη μακροχρόνια αρχειοθέτηση και επαναχρησιμοποίησή των δεδομένων αυτών.

Στο πλαίσιο του Ψηφιακού Διδύμου, το IFC μπορεί να λειτουργήσει ως κοινή γλώσσα δεδομένων που συνδέει το μοντέλο BIM με τα συστήματα παρακολούθησης, τους αισθητήρες IoT και τις πλατφόρμες ανάλυσης, κάνοντας πιο εύκολη την ανταλλαγή και ενημέρωση δεδομένων σε πραγματικό χρόνο.

2.4 Εργαλεία Απεικόνισης

2.4.1 Blender

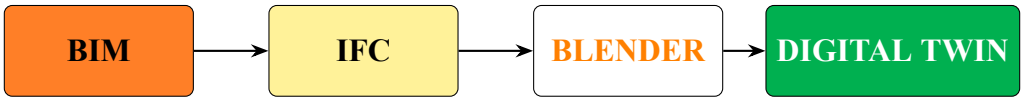
Το Blender είναι ένα ελεύθερο και ανοιχτού κώδικα λογισμικό τρισδιάστατης γραφικής απεικόνισης, σχεδιασμένο για τη δημιουργία και επεξεργασία τρισδιάστατων μοντέλων, κινούμενων εικόνων, προσομοιώσεων και οπτικών εφέ[26]. Υποστηρίζει ένα πλήρες σύνολο εργαλείων που καλύπτουν όλη τη διαδικασία παραγωγής τρισδιάστατου περιεχομένου, συμπεριλαμβανομένης της μοντελοποίησης, της δημιουργίας υλικών, του φωτισμού, της ρύθμισης καμερών, της κίνησης αντικειμένων και της τελικής απόδοσης (*rendering*).

Η προτίμησή του από το κοινό οφείλεται στην ευελιξία του και στην ενεργή κοινότητα χρηστών και προγραμματιστών που συμβάλλουν καθημερινά στην εξέλιξή του. Το Blender χρησιμοποιείται ευρέως τόσο σε δημιουργικά επαγγέλματα (κινηματογράφος, παιχνίδια, εικονική πραγματικότητα) όσο και σε επιστημονικές εφαρμογές που απαιτούν τρισδιάστατη απεικόνιση, προσομοίωση και ανάλυση δεδομένων.

2.4.2 Σύνδεση Blender με BIM και Ψηφιακό Δίδυμο

Στο πλαίσιο του Building Information Modeling (BIM), το Blender μπορεί να λειτουργήσει ως εργαλείο για οπτικοποίηση και παρουσίαση δεδομένων κτιριακών έργων. Μέσω κατάλληλων προσθέτων (*add-ons*), όπως το BlenderBIM, είναι δυνατή η εισαγωγή και επεξεργασία αρχείων IFC. Αυτό επιτρέπει την αξιοποίηση των γεωμετρικών και περιγραφικών δεδομένων που περιέχονται στα BIM μοντέλα για τη δημιουργία ρεαλιστικών αναπαραστάσεων, animations και προσομοιώσεων.

Στη δημιουργία Ψηφιακών Διδύμων, το Blender μπορεί να χρησιμοποιηθεί για τη σύνδεση των τρισδιάστατων μοντέλων με δυναμικά δεδομένα που προέρχονται από αισθητήρες και αισθηρήρες IoT. Μέσω εργαλείων scripting σε Python, είναι δυνατή η εισαγωγή και απεικόνιση δεδομένων σε πραγματικό χρόνο, επιτρέποντας την αλληλεπίδραση του οπτικού τρισδιάστατου μοντέλου με τις αλλαγές που συμβαίνουν στο φυσικό αντικείμενο. Με αυτόν τον τρόπο, το Blender μπορεί να αποτελέσει έναν κρίσιμο σύνδεσμο μεταξύ του BIM μοντέλου και του λειτουργικού Ψηφιακού Διδύμου.



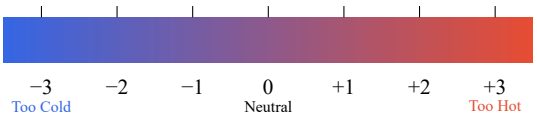
Σχήμα 2: Διαδικασία μετάβασης από BIM σε Ψηφιακό Δίδυμο μέσω IFC και Blender

2.5 Μοντέλα Άνεσης

2.5.1 PMV & PPD

Οι δείκτες Predicted Mean Vote (PMV) και Predicted Percentage of Dissatisfied (PPD) χρησιμοποιούνται για την εκτίμηση της θερμικής άνεσης σε εσωτερικούς χώρους. Αναπτύχθηκαν από τον Povl Ole Fanger[27], ο οποίος υπέθεσε ότι η ανθρώπινη θερμική άνεση βασίζεται στη θερμοκρασία του δέρματος και στην έκκριση ιδρώτα, κάποιος μπορεί να θεωρηθεί «άνετος» μόνο εάν αυτοί οι δύο παράγοντες είναι ισορροπημένοι μέσα σε ένα στενό αποδεκτό φάσμα.

Ο δείκτης PMV εκφράζει τη μέση τιμή της ψήφου μίας μεγάλης ομάδας ατόμων σε μια κλίμακα επτά βαθμών.



Σχήμα 3: Κλίμακα PMV

Ο υπολογισμός του PMV βασίζεται σε έξι παραμέτρους: τη θερμοκρασία αέρα, τη μέση ακτινοβολούμενη θερμοκρασία, τη σχετική υγρασία, την ταχύτητα του αέρα, τον μεταβολικό ρυθμό (*metabolic rate*) και τη θερμική αντίσταση του ρουχισμού (*clo value*).

Ο δείκτης PPD εκτιμά το ποσοστό των ατόμων που πιθανόν να είναι δυσαρεστημένοι από τις συνθήκες θερμικού περιβάλλοντος, ακόμα κι αν το PMV είναι κοντά στο μηδέν. Το PPD δίνεται ως συνάρτηση του PMV και εκφράζεται ως ποσοστό (%). Η ελάχιστη θεωρητική τιμή του PPD είναι περίπου 5%, δηλαδή ακόμα και υπό ιδανικές συνθήκες, ένα μικρό ποσοστό ανθρώπων θα παραμένει δυσαρεστημένο.

Σύμφωνα με το πρότυπο ISO 7730[28], οι αποδεκτές τιμές για PMV και PPD εξαρτώνται από το επίπεδο άνεσης που επιδιώκεται:

- **Κατηγορία A (νέα ή απαιτητικά κτίρια):** PMV μεταξύ -0.2 και $+0.2$, PPD $\leq 6\%$
- **Κατηγορία B (τυπικά νέα κτίρια):** PMV μεταξύ -0.5 και $+0.5$, PPD $\leq 10\%$
- **Κατηγορία C (παλαιότερα κτίρια):** PMV μεταξύ -0.7 και $+0.7$, PPD $\leq 15\%$

Οι δείκτες PMV και PPD είναι κρίσιμοι στη ρύθμιση συστημάτων και στη βελτιστοποίηση ενεργειακής κατανάλωσης, καθώς επιτρέπουν την προσαρμογή του εσωτερικού περιβάλλοντος στις ανάγκες των χρηστών. Στο πλαίσιο του Ψηφιακού Διδύμου, μπορούν να υπολογίζονται δυναμικά με χρήση αισθητήρων και μοντέλων, υποστηρίζοντας την έξυπνη διαχείριση θερμικής άνεσης σε πραγματικό χρόνο.

2.6 Τεχνητή Νοημοσύνη

2.6.1 Ανίχνευση Ανωμαλιών σε χρονοσειρές (Isolation Forest)

Η ανίχνευση ανωμαλιών σε χρονοσειρές αποτελεί βασικό εργαλείο για την παρακολούθηση και τη διάγνωση συστημάτων, ιδιαίτερα στο πλαίσιο ενός Ψηφιακού Διδύμου, όπου η συνεχής ροή δεδομένων από αισθητήρες καθιστά απαραίτητο τον εντοπισμό αποκλίσεων των αισθητήρων από την κανονική συμπεριφορά τους. Ο αλγόριθμος Isolation Forest είναι μια από τις πιο αποδοτικές μεθόδους για την ανίχνευση ανωμαλιών.

Ο αλγόριθμος αυτός δημιουργεί ένα σύνολο από τυχαία δέντρα (isolation trees) στα οποία κάθε διαχωρισμός επιλέγεται τυχαία ως προς το χαρακτηριστικό και το όριο τιμής. Η βασική

ιδιότητα που αξιοποιεί είναι ότι οι ανωμαλίες εμφανίζονται σε σύντομα μονοπάτια στα δέντρα, μιάς και απαιτούν λιγότερους διαχωρισμούς για να απομονωθούν.

Το μήκος μονοπατιού $h(x)$ μιας παρατήρησης x ορίζεται ως ο αριθμός των βημάτων που απαιτούνται για να φτάσει σε φύλλο του δέντρου. Ο αναμενόμενος μέσος όρος μήκους μονοπατιού για κανονικά δεδομένα είναι μεγαλύτερος, ενώ για ανωμαλίες μικρότερος.

Ο βαθμός ανωμαλίας (anomaly score) για κάθε παρατήρηση υπολογίζεται ως εξής:

$$(x, n) = 2^{\frac{-E[h(x)]}{c(n)}}$$

όπου:

- $E[h(x)]$ είναι η μέση τιμή του μήκους μονοπατιού για την παρατήρηση x στα δέντρα,
- $c(n)$ είναι η αναμενόμενη τιμή του μήκους μονοπατιού για δείγμα μεγέθους n, και προσεγγίζεται από τη σχέση:

$$c(n) = \begin{cases} 2H(n-1) - \frac{2(n-1)}{n}, & n > 2 \\ 1, & n = 2 \\ 0, & n \leq 1 \end{cases}$$

με $H(i)$ να είναι ο αρμονικός αριθμός $H(i) = \ln(i) + \gamma$ (και $\gamma \approx 0.5772$, σταθερά Euler-Mascheroni).

Οι τιμές του $s(x, n)$ κυμαίνονται μεταξύ 0 και 1. Τιμές κοντά στο 1 υποδεικνύουν μεγάλη πιθανότητα ανωμαλίας.

Η μέθοδος είναι κατάλληλη για μεγάλα σύνολα δεδομένων λόγω της γραμμικής της συμπεριφοράς.

Για την αξιολόγηση της απόδοσης ενός μοντέλου ανίχνευσης ανωμαλιών, χρησιμοποιούνται μετρικές όπως:

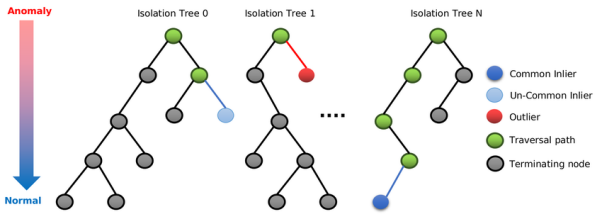
- **Precision** — το ποσοστό των ανωμαλιών που ανιχνεύτηκαν και είναι πραγματικά ανωμαλίες.
- **Recall** — το ποσοστό των πραγματικών ανωμαλιών που εντοπίστηκαν.
- **F1-score** — ο αρμονικός μέσος του Precision και Recall.
- **ROC AUC** — η περιοχή κάτω από την καμπύλη ROC.

Η επιθυμητή απόδοση εξαρτάται από την εφαρμογή. Σε περιβάλλον Ψηφιακού Διδύμου μας ενδιαφέρει κυρίως η γρήγορη, έγκυρη και ελαφριά σε πόρους ανίχνευση, συχνά σε περιβάλλον χωρίς εποπτεία (unsupervised).

2.6.2 LSTM

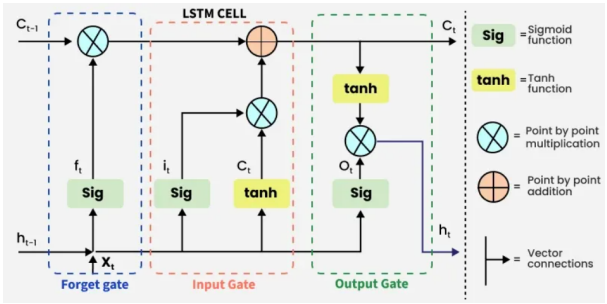
Τα Long Short-Term Memory (LSTM) δίκτυα είναι μια ειδική κατηγορία αναδρομικών νευρωνικών δικτύων (*Recurrent Neural Networks – RNNs*), σχεδιασμένα ώστε να μπορούν να μαθαίνουν και να διατηρούν πληροφορία σε ακολουθιακά δεδομένα μεγάλης διάρκειας. Σε αντίθεση με τα παραδοσιακά RNNs, τα LSTM επιλύουν το πρόβλημα της ”εξασθένισης του σήματος” (*vanishing gradient problem*) χάρη στη χρήση εσωτερικών μηχανισμών μνήμης και πυλών ελέγχου (*gates*).

Κάθε μονάδα LSTM περιλαμβάνει τρεις βασικές πύλες:



Σχήμα 4: Απεικόνιση της λογικής του Isolation Forest. Τα ανώμαλα σημεία απομονώνονται με λιγότερους διαχωρισμούς (σύντομα μονοπάτια) σε σχέση με τα κανονικά σημεία.

- **Forget gate**, που αποφασίζει ποια πληροφορία από την προηγούμενη κατάσταση θα διαγραφεί.
- **Input gate**, που ελέγχει ποια νέα πληροφορία θα προστεθεί στη μνήμη.
- **Output gate**, που καθορίζει ποια πληροφορία θα παραχθεί στην έξοδο και θα περάσει στο επόμενο βήμα.



Σχήμα 5: Μοντέλο LSTM.

Η αρχιτεκτονική αυτή επιτρέπει στα LSTM μοντέλα να ”θυμούνται” κρίσιμες πληροφορίες από προηγούμενες χρονικές στιγμές, καθιστώντας τα ιδανικά για την πρόβλεψη χρονοσειρών, όπως μετρήσεις αισθητήρων, θερμοκρασίες, καταναλώσεις ενέργειας και άλλες δυναμικές μεταβλητές στο πλαίσιο ενός Ψηφιακού Διδύμου.

Η εκπαίδευση των LSTM απαιτεί εποπτευμένα δεδομένα (labelled data) και συνήθως πραγματοποιείται με αλγορίθμους βελτιστοποίησης όπως ο *Adam*. Τα μοντέλα αξιολογούνται με μετρικές όπως η MSE (Mean Squared Error), η MAE (Mean Absolute Error) και R^2 score.

2.6.3 AI Βοηθός

Ο AI Βοηθός σε ένα περιβάλλον Ψηφιακού Διδύμου λειτουργεί ως ένα έξυπνο σύστημα υποστήριξης αποφάσεων, ικανό να αλληλεπιδρά με το ψηφιακό μοντέλο και τον χρήστη και να εξάγει χρήσιμες πληροφορίες για τον χρήστη. Συνδυάζει τεχνικές τεχνητής νοημοσύνης, όπως φυσική γλώσσα (*Natural Language Processing*), μηχανική μάθηση (*Machine Learning*), και λογική ανάλυση δεδομένων για να κατανοήσει αιτήματα, να εντοπίσει προβλήματα, να προτείνει λύσεις και να εξηγήσει καταστάσεις.

Ο AI βοηθός μπορεί να αναλάβει πολλαπλούς ρόλους:

- **Επεξήγηση δεδομένων αισθητήρων**: Ανάλυση εισερχόμενων ροών δεδομένων και αυτόματη αναφορά κρίσιμων μετρήσεων.

- **Ανίχνευση ανωμαλιών ή αποκλίσεων**: Συνδυασμός μαθηματικών μοντέλων (π.χ. Isolation Forest, LSTM) με απλή και κατανοητή εξήγηση στον χρήστη.
- **Πρόβλεψη κατάστασης**: Χρήση εκπαιδευμένων μοντέλων για την πρόβλεψη μελλοντικών μεταβολών και την έγκαιρη πρόταση για πρόληψη.
- **Αλληλεπίδραση μέσω φυσικής γλώσσας**: Δυνατότητα ο χρήστης να δίνει εντολές σε φυσική γλώσσα, οι οποίες μετατρέπονται αυτόματα σε ερωτήματα προς το Ψηφιακό Δίδυμο.

Η ενσωμάτωση ενός AI Βοηθού στο Ψηφιακό Δίδυμο κάνει τη διαχείριση συστημάτων πιο προσιτή και κατανοητή, ειδικά για χρήστες που δεν είναι αρκετά εξοικειωμένοι με τη τεχνολογία. Βελτιώνει τη λειτουργία των υποδομών παρέχοντας πιο έξυπνη ανάλυση, αυτοματοποίηση και καλύτερη λήψη αποφάσεων σε πραγματικό χρόνο.

Στη δική μας περίπτωση, ο AI Βοηθός έχει υλοποιηθεί μέσω της διεπαφής ChatGPT API [29], η οποία επιτρέπει την κατανόηση και επεξεργασία φυσικής γλώσσας. Για λόγους ασφάλειας, ακρίβειας και σχετικότητας, το σύστημα έχει διαμορφωθεί ώστε να φιλτράρει τα ερωτήματα των χρηστών και να απαντά αποκλειστικά σε περιεχόμενο που σχετίζεται με το Ψηφιακό Δίδυμο, τους αισθητήρες, και τα συστήματα παρακολούθησης και πρόβλεψης. Ερωτήσεις που δεν είναι σχετικές με τα παραπάνω επιστρέφουν κατάλληλα μηνύματα καθοδήγησης. Χάρη σε αυτή την προσθήκη, ο AI Βοηθός λειτουργεί ως ένας σύμβουλος, εμπλουτίζοντας την αλληλεπίδραση του χρήστη με το ψηφιακό δίδυμο.

3 Προετοιμασία IFC μοντέλου και Εισαγωγή στο Blender

3.1 Δημιουργία του Blender Project

Πριν από οποιαδήποτε εισαγωγή δεδομένων ή μοντέλων, δημιουργήθηκε ένα νέο *Blender project* το οποίο αποτέλεσε τη βάση για την ενσωμάτωση του τρισδιάστατου περιβάλλοντος. Χρησιμοποιήθηκε η έκδοση "Blender 4.2", η οποία είναι η τελευταία έκδοση που προσφέρει πλήρη υποστήριξη για λειτουργία χωρίς κάρτα γραφικών, καθιστώντας την ιδανική επιλογή για χρήση σε συστήματα χαμηλής επίδοσης που δεν διαθέτουν κάρτα γραφικών. Κατά τη δημιουργία της σκηνής, εφαρμόστηκαν βασικές ρυθμίσεις ώστε να εξασφαλιστεί η σωστή απεικόνιση και συμβατότητα με τα δεδομένα IFC

Εγκαταστάθηκε και ενεργοποιήθηκε το add on "Bonsai"[30], το οποίο έχει σχεδιαστεί για να είναι μια ολοκληρωμένη πλατφόρμα δημιουργίας και επεξεργασίας BIM. Το Bonsai περιλαμβάνει ένα ευρύ φάσμα εργασιών και ροών εργασίας που τα βρίσκουμε αρκετά συχνά σε διάφορα λογισμικά BIM. Μέσα από το Bonsai, είναι δυνατή η προβολή και εξερεύνηση μοντέλων IFC, με πλήρες πρόσβαση σε χώρους, ιδιότητες και σχέσεις. Ακόμα, έχει τη δυνατότητα επεξεργασίας και εξαγωγής χαρακτηριστικών, ιδιοτήτων και μεταδεδομένων (metadata) απευθείας από τα δεδομένα του IFC, καθώς και μετακίνηση, περιστροφή και τροποποίηση της γεωμετρίας των αντικειμένων, ενώ ταυτόχρονα διατηρεί τη σημασιολογία και τη δομή του IFC. Ο χρήστης μπορεί να δημιουργήσει νέα αντικείμενα είτε χρησιμοποιώντας προκαθορισμένα στοιχεία από τη βιβλιοθήκη είτε ορίζοντας καινούργιους παραμετρικούς τύπους. Επιπρόσθετα, υποστηρίζεται η διαχείριση συστημάτων ταξινόμησης, αναφορών εγγράφων και συνδέσμων προς εξωτερικές βιβλιοθήκες. Επιπλέον, το Bonsai διευκολύνει τη δημιουργία δισδιάστατων σχεδίων, όπως κατόψεις, τομές και άλλες όψεις. Το Bonsai αποτελεί την επίσημη BIM επέκταση του Blender και έχει αντικαταστήσει το παλαιότερο BlenderBIM.

Ορίστηκε το σύστημα μονάδων σε μέτρα (metric) με ακρίβεια τουλάχιστον 0.001m, ώστε να ανταποκρίνεται στις γεωμετρικές διαστάσεις των κατασκευών.

Οργανώθηκαν οι συλλογές (*collections*) στο Outliner για ευκολότερη διαχείριση των εισαγόμενων αντικειμένων.

Ρυθμίστηκαν οι οπτικές παράμετροι της σκηνής (grid size, background, φωτισμός) για να υποστηρίζεται καλύτερα η απεικόνιση αισθητήρων και δεδομένων.

Το αρχείο αποθηκεύτηκε ως βάση (template) για την επόμενη φάση, την εισαγωγή του IFC μοντέλου.

3.2 Εισαγωγή του IFC 3D Μοντέλου

Η εισαγωγή του τρισδιάστατου αρχιτεκτονικού μοντέλου πραγματοποιήθηκε μέσω του add on "Bonsai", το οποίο παρέχει πλήρη υποστήριξη για αρχεία IFC. Το αρχείο IFC είχε δημιουργηθεί εκ των προτέρων μέσω εξωτερικού λογισμικού BIM και περιλάμβανε τις βασικές γεωμετρικές πληροφορίες του κτιρίου, όπως τοίχους, δάπεδα, οροφές, ανοίγματα (θύρες και παράθυρα) και λοιπά δομικά στοιχεία.

Μετά την εισαγωγή διατηρήθηκαν τα *property sets* των αντικειμένων για μελλοντική αξιοποίηση μεταδεδομένων, όπως τύπος υλικού και πληροφορίες επιπέδων. Ελέγχθηκε η ακρίβεια του μοντέλου μέσω περιήγησης στον τρισδιάστατο χώρο, ώστε να επιβεβαιωθεί η σωστή τοποθέτηση όλων των στοιχείων. Πραγματοποιήθηκαν στοχευμένες τροποποιήσεις στο μοντέλο μέσα από το Blender, είτε για την αποκατάσταση ελλείψεων (π.χ. απουσία τοίχων ή παραθύρων), είτε για λόγους καλύτερης οργάνωσης και καθαρότητας του σχεδίου.

Τέλος, το μοντέλο ευθυγραμμίστηκε με το κέντρο του συστήματος συντεταγμένων του

Blender (*world origin*), για να γίνει πιό εύκολη η μελλοντική τοποθέτηση αισθητήρων και η συγγραφή των διαφόρων script.

3.3 Τοποθέτηση αισθητήρων στο IFC μοντέλο μέσω Blender

Για την αναπαράσταση των αισθητήρων στο τρισδιάστατο μοντέλο, χρησιμοποιήθηκαν έτοιμα γεωμετρικά αρχεία τύπου `.obj`, τα οποία επιλέχθηκαν από την πλατφόρμα CGTrader[31] με γνώμονα τη μορφολογική ομοιότητά τους με πραγματικούς αισθητήρες.

Αφού εισήχθησαν τα αρχεία στο Blender, δημιουργήθηκαν οι αντίστοιχοι IFC τύποι για κάθε κατηγορία αισθητήρα μέσω της δομής του IfcTypeProduct. Συγκεκριμένα, ορίστηκαν οι τύποι IfcSensorType/Heater, IfcSensorType/AirQuality και IfcSensorType/EnergyMeter. Οι τύποι τοποθετήθηκαν στην αντίστοιχη συλλογή τύπων (collection) και στη συνέχεια τοποθετήθηκαν στο χώρο του μοντέλου ως μεμονωμένα instances.

Κατά την τοποθέτηση, οι αισθητήρες εντάχθηκαν χειροκίνητα στα δωμάτια και τους χώρους όπου προβλεπόταν η λειτουργία τους, σύμφωνα με το αρχιτεκτονικό σχέδιο. Λήφθηκε μέριμνα ώστε κάθε instance να αντιστοιχιστεί στο σωστό IfcBuildingStorey, διατηρώντας έτσι τη σωστή ιεραρχία του μοντέλου IFC. Σε κάθε αντικείμενο προστέθηκαν κάποια αναγνωριστηκά (Name, GlobalId, PredefinedType) ώστε να μπορούν να αναγνωριστούν και να συνδεθούν με τα δεδομένα των αισθητήρων στην επόμενη φάση.

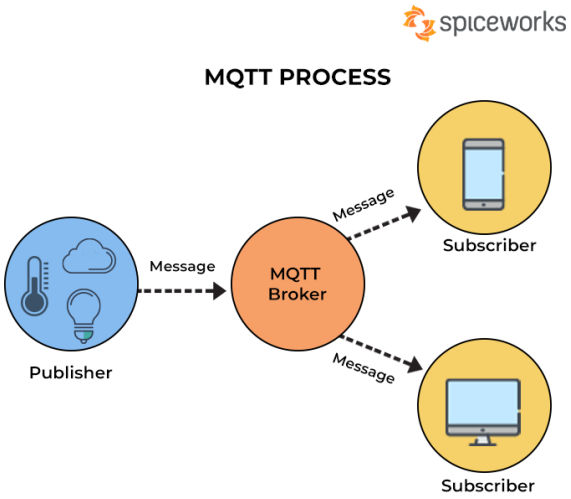
Η τοποθέτηση των αισθητήρων με αυτό τον τρόπο επέτρεψε τόσο την ρεαλιστική απεικόνιση στο περιβάλλον Blender όσο και τη διατήρηση της λογικής IFC δομής, κάνοντας το μοντέλο κατάλληλο για χρήση ως ψηφιακό δίδυμο.

4 Απεικόνιση σε πραγματικό χρόνο στο Blender

4.1 Λήψη δεδομένων

4.1.1 Ορισμός MQTT

Πρίν προχωρήσουμε στη λήψη δεδομένων πρέπει να κατανοήσουμε τι είναι το πρωτόκολλο MQTT. Το Message Queuing Telemetry Transport (MQTT) είναι ένα ελαφρύ πρωτόκολλο ανταλλαγής μηνυμάτων, σχεδιασμένο για επικοινωνία μεταξύ συσκευών με περιορισμένους πόρους και δικτύων χαμηλού εύρους ζώνης[32]. Βασίζεται στο μοντέλο δημοσίευσης/εγγραφής (*publish/subscribe*), στο οποίο οι πελάτες (*clients*) μπορούν να δημοσιεύουν (*publish*) μηνύματα σε συγκεκριμένα θέματα (*topics*) ή να εγγράφονται (*subscribe*) ώστε να λαμβάνουν μηνύματα από αυτά. Η αρχιτεκτονική του περιλαμβάνει τρία βασικά στοιχεία. Τον *client*, που είναι η συσκευή ή εφαρμογή που αποστέλλει ή λαμβάνει μηνύματα, τον *broker*, ο οποίος διαχειρίζεται τη διανομή των μηνυμάτων στους συνδρομητές και το *topic*, που καθορίζει τη θεματική κατηγορία και χρησιμοποιείται για την οργάνωση και δρομολόγηση της πληροφορίας.



Σχήμα 6: Αρχιτεκτονική επικοινωνίας μέσω πρωτοκόλλου MQTT [2].

Το MQTT είναι ιδιαίτερα κατάλληλο για εφαρμογές Internet of Things (IoT), καθώς χαρακτηρίζεται από ελαφριά επικοινωνία με ελάχιστη επιβάρυνση δεδομένων, γεγονός που το καθιστά ιδανικό για συσκευές με περιορισμένο επεξεργαστικό και ενεργειακό απόθεμα. Επίσης παρέχει αξιόπιστη μετάδοση μέσω επιπέδων ποιότητας υπηρεσίας (*QoS*), που επιτρέπουν τον έλεγχο της παράδοσης των μηνυμάτων. Επιπρόσθετα, υποστηρίζει μηχανισμούς ασφαλείας όπως SSL/TLS για κρυπτογραφημένη μεταφορά δεδομένων.

Στο πλαίσιο του *Ψηφιακού Διδύμου*, το MQTT μπορεί να χρησιμοποιηθεί για τη μεταφορά δεδομένων αισθητήρων από τον φυσικό κόσμο προς την ψηφιακή του αναπαράσταση, επιτρέποντας την ενημέρωση του μοντέλου σε πραγματικό χρόνο και την υλοποίηση ενός συνεχούς βρόχου ανατροφοδότησης [33].

4.1.2 Λήψη δεδομένων στο Blender

Για την λήψη δεδομένων σε πραγματικό χρόνο χρησιμοποιήθηκε η λειτουργία scripting που παρέχει το Blender. Τα δεδομένα έρχονται από αισθητήρες μέσω του πρωτοκόλλου MQTT. Η υλοποίηση πραγματοποιήθηκε σε περιβάλλον Python με χρήση της βιβλιοθήκης paho-mqtt, η

οποία επιτρέπει τη δημιουργία MQTT πελάτη (client) που συνδέεται στον broker και εγγράφεται (subscribe) στα κατάλληλα θέματα (topics).

Κάθε αισθητήρας αναγνωρίζεται μέσω του μοναδικού του *Sensor ID* και κατηγοριοποιείται σε έναν από τους τρεις τύπους, *Air Quality*, *Energy Meter* ή *Heater*. Η ένταξη του κάθε αισθητήρα σε μία από τις τρείς κατηγορίες γίνεται βρίσκοντας το *Sensor ID* και τον τύπο που του αντιστοιχεί από μία προϋπάρχουσα λίστα. Με αυτό τον τρόπο διασφαλίζεται η ορθή αντιστοίχιση των μετρήσεων με τον αντίστοιχο αισθητήρα και τον αντίστοιχο τύπο.

Τα δεδομένα λαμβάνονται σε μορφή JSON, όπου το πεδίο `Measurements` περιέχει τις επιμέρους μετρήσεις (π.χ. θερμοκρασία, υγρασία, κατανάλωση ενέργειας). Το script φιλτράρει τα μηνύματα που δεν περιέχουν έγκυρα δεδομένα και σε περίπτωση έγκυρου payload, εξάγει τις τιμές και τις αποδίδει στον αντίστοιχο αισθητήρα.

Στο παράδειγμα 1 φαίνεται ο βασικός μηχανισμός σύνδεσης σε MQTT broker ο διαχωρισμός των αισθητήρων σε κατηγορίες και η επεξεργασία των ακατέργαστων δεδομένων που λαμβάνονται. Η εκτέλεση του script επιστρέφει τις μετρήσεις που αποστέλλονται από τους αισθητήρες, σε μορφή JSON, όπως στο παράδειγμα 2.

Listing 1: Ενδεικτικό Python script για λήψη δεδομένων MQTT

```
# imports
...

# Sensor IDs
AIR_QUALITY_IDS = {...}
ENERGY_METER_IDS = {...}
HEATER_IDS = {...}

# MQTT Connection Setting
BROKER = 'the.brokers.url'
PORT = 1234
TOPIC = '#'
USERNAME = 'username'
PASSWORD = 'password'
client_id = f'mqtt-client-{uuid.uuid4()}'

def categorize_sensor(sensor_id):
    if sensor_id in AIR_QUALITY_IDS:
        return "Air␣Quality"
    elif sensor_id in ENERGY_METER_IDS:
        return "Energy␣Meter"
    elif sensor_id in HEATER_IDS:
        return "Heater"
    return None  # unknown

def on_connect(client, userdata, flags, rc):
    print(f"MQTT␣connected␣(rc={rc})")
    if rc == 0:
        client.subscribe(TOPIC)

def on_message(client, userdata, msg):
    topic = msg.topic
    raw = msg.payload.decode('utf-8', errors='ignore').strip()

    if not raw or not raw.startswith('{'):
        return

    # Parse the data
```

```
    try:
        data = json.loads(raw)
        measurements = data.get("Measurements", [])
        if not measurements or not isinstance(measurements[0], dict):
            return
        value = measurements[0].get("Value")
        if not isinstance(value, dict):
            return

        sensor_id = topic.split("/")[-1]
        category = categorize_sensor(sensor_id)
        if category:
            print(f"\n{category} ({sensor_id}):")
            for k, v in value.items():
                print(f"{k}: {v}")

    except Exception as e:
        print(f"JSON parse error: {e}")

if __name__ == "__main__":
    client = mqtt.Client(client_id=client_id)
    client.username_pw_set(USERNAME, PASSWORD)
    client.on_connect = on_connect
    client.on_message = on_message

    print("Listening for MQTT messages...")
    try:
        client.connect(BROKER, PORT, 60)
        client.loop_forever()
    except KeyboardInterrupt:
        print("\nDisconnected.")
        client.disconnect()
```

Listing 2: Ενδεικτικη απάντηση δεδομένων MQTT

```
Listening for MQTT messages...
MQTT connected (rc=0)

Energy Meter (shellypro3em-08f9e0e5121c):
   a_current: 1.106
   a_voltage: 233.8
   a_act_power: 122.2
   ...

Air Quality (70:ee:50:83:4d:e8):
   temperature: 23.3
   humidity: 58
   Pressure: 1017.3
   AbsolutePressure: 1014.3
   Noise: 32
   CO2: 405
```

Η διαδικασία αυτή επιτρέπει τη συνεχή λήψη και επεξεργασία δεδομένων σε πραγματικό χρόνο. Τα δεδομένα αυτά μπορούν να ενημερώνουν δυναμικά το τρισδιάστατο μοντέλο, υποστηρίζοντας έτσι τη λειτουργία του Ψηφιακού Διδύμου.

4.2 Αντιστοίχιση των δεδομένων των αισθητήρων στα 3D αντικείμενα

Για την αντιστοίχιση των δεδομένων που λαμβάνονται από τους αισθητήρες με τα τρισδιάστατα αντικείμενα του Blender, χρησιμοποιήθηκε ένας μηχανισμός που βασίζεται σε αρχείο διαμόρφωσης ”smart_objects.json” 3. Το αρχείο αυτό αποτελείται από ζεύγη Sensor Name–Sensor ID, επιτρέποντας την αντιστοίχηση του κάθε αισθητήρα με το αντίστοιχο αντικείμενο του τρισδιάστατου μοντέλου.

Κατά την εκτέλεση, το python script φορτώνει το αρχείο διαμόρφωσης και δημιουργεί μια λίστα αντικειμένων που περιέχουν, μεταξύ άλλων, το όνομα του κάθε αισθητήρα στο Blender και το αναμενόμενο MQTT θέμα. Στη συνέχεια, κάθε φορά που λαμβάνεται κάποιο μήνυμα MQTT, γίνεται έλεγχος για το αν το θέμα (topic) του μηνύματος αντιστοιχεί σε κάποιον από τους αισθητήρες που είναι καταγεγραμμένοι στη λίστα αντικειμένων. Στην περίπτωση που βρεθεί κάποια αντιστοιχία, οι τιμές των μετρήσεων που λήφθηκαν αποθηκεύονται σε μια δομή δεδομένων (sensor_values) με κλειδί το όνομα του αισθητήρα που ορίστηκε στο Blender 4.

Με αυτό τον τρόπο, η αντιστοίχιση μεταξύ των Sensor IDs που προέρχονται από τους αισθητήρες και των αντικειμένων στο τρισδιάστατο μοντέλο διατηρείται αυτόματα και δυναμικά, επιτρέποντας έτσι την ορθή απεικόνιση των δεδομένων σε επόμενα στάδια.

Listing 3: Δομή του αρχείου smart_objects.json

```
[
    {
        "name": "Heater_Kitchen_1",
        "type": "ifcsensortype/heater",
        "topic": "the/topic/trv/shellytrv-8cf681a52e2e",
        "host": "mqtt://the.broker.url:1234"
    },
    {
        "name": "AirQuality_Kitchen",
        "type": "ifcsensortype/airquality",
        "topic": "the/topic//iaq/70:ee:50:83:2a:6c",
        "host": "mqtt://the.broker.url:1234"
    }
]
```

Listing 4: Φόρτωση αρχείου διαμόρφωσης και αντιστοίχιση IDs με αντικείμενα Blender

```
# Φόρτωση λίστας αισθητήρων από το smart_objects.json
def load_sensor_config():
    blend_dir = bpy.path.abspath("//")  # directory του .blend αρχείου
    config_path = os.path.join(blend_dir, "smart_objects.json")
    try:
        with open(config_path, "r", encoding="utf-8") as f:
            return json.load(f)
    except Exception as e:
        print(f"Error␣loading␣smart_objects.json:␣{e}")
        return []

# Ενημέρωση δεδομένων για τον σωστό αισθητήρα
for sensor in sensor_config:
    expected_topic = sensor.get("topic", "")
    if expected_topic and topic.startswith(expected_topic):
        name = sensor.get("name", "Unknown")
        with cache_lock:
            sensor_values[name] = value
        break
```

4.3 Επικαλύψεις UI και Αναγνώσεις Δεδομένων στο Blender

Για να προσφέρουμε στον χρήστη τη δυνατότητα να μπορεί να δει τις τελευταίες μετρήσεις των αισθητήρων απευθείας μέσα στο τρισδιάστατο μοντέλο, υλοποιήθηκε ένα UI overlay 5 στο Blender. Το overlay αυτό εμφανίζεται όταν ο χρήστης επιλέξει από το side panel το tab ”Sensor” και ακολούθως ”Show Overlay”. Στη συνέχεια, επιλέγοντας έναν αισθητήρα στο 3D View, εμφανίζονται οι μετρήσεις του σε ένα πλαίσιο πάνω από το σημείο όπου βρίσκεται ο αισθητήρας. Για να γίνει απόκρυψη του πλαισίου, ο χρήστης αρκεί να επιλέξει από το sidepanel το tab ”Sensor” και μετά ”Hide Overlay”.

Η υλοποίηση βασίζεται σε draw_handler του SpaceView3D API του Blender, το οποίο επιτρέπει την σχεδίαση προσαρμοσμένων γραφικών στοιχείων πάνω από την προβολή. Το περιεχόμενο του overlay ενημερώνεται κάθε φορά που λαμβάνονται νέα δεδομένα μέσω MQTT, ενώ η σχεδίαση ανανεώνεται σε πραγματικό χρόνο σε κάθε καρέ (*frame*) της προβολής, εξασφαλίζοντας έτσι ομαλή αλλά και άμεση απεικόνιση των τελευταίων μετρήσεων.

Κατά την εκτέλεση:

1. Γίνεται λήψη της τρέχουσας θέσης του επιλεγμένου αντικειμένου στον 3D χώρο.
2. Η θέση αυτή μετατρέπεται σε 2D συντεταγμένες οθόνης με χρήση της συνάρτησης location_3d_to_region_2d.
3. Σχεδιάζεται ένα πλαίσιο (με χρώμα φόντου και περίγραμμα) και στη συνέχεια προστίθεται το κείμενο με τις μετρήσεις του αισθητήρα.
4. Αν δεν υπάρχουν διαθέσιμα δεδομένα, εμφανίζεται το σχετικό μήνυμα.

Listing 5: Λογική σχεδίασης του overlay στο Blender

```
def draw_callback_px(self, context):
    obj = context.active_object
    if not obj:
        return

    # Μετατροπή 3D θέσης σε 2D
    region = context.region
    rv3d = context.region_data
    coord = obj.location
    vec2d = location_3d_to_region_2d(region, rv3d, coord)
    if not vec2d:
        return

    name = obj.name
    with cache_lock:
        values = sensor_values.get(name)

    font_id = 0
    blf.size(font_id, 14)
    padding = 6
    line_height = 16

    # Κείμενο προς εμφάνιση
    if values:
        lines = [f"{name} Sensor Data:"]
        for k, v in values.items():
            lines.append(f"{k}: {v}")
```

```
    else:
        lines = [f"{name}", "No sensor data"]

    # Υπολογισμός μεγέθους πλαισίου
    width = max(blf.dimensions(font_id, line)[0] for line in lines) +
    ↪ padding * 2
    height = len(lines) * line_height + padding * 2
    x = vec2d.x - width / 2
    y = vec2d.y + 10

    # Σχεδίαση φόντου
    shader = gpu.shader.from_builtin('UNIFORM_COLOR')
    batch = batch_for_shader(shader, 'TRI_FAN', {
        "pos": [(x, y), (x + width, y), (x + width, y + height), (x, y +
    ↪ height)]
    })
    gpu.state.blend_set('ALPHA')
    shader.bind()
    shader.uniform_float("color", (1, 1, 1, 0.85))
    batch.draw(shader)

    # Σχεδίαση περιγράμματος
    border = batch_for_shader(shader, 'LINE_LOOP', {
        "pos": [(x, y), (x + width, y), (x + width, y + height), (x, y +
    ↪ height)]
    })
    shader.uniform_float("color", (0, 0, 0, 1))
    border.draw(shader)
    gpu.state.blend_set('NONE')

    # Σχεδίαση κειμένου
    for i, line in enumerate(lines):
        blf.position(font_id, x + padding, y + padding + i * line_height,
    ↪ 0)
        blf.draw(font_id, line)

# Operator
class ToggleMQTTOverlayOperator(bpy.types.Operator):
    bl_idname = "wm.toggle_mqtt_overlay"
    bl_label = "Toggle MQTT Sensor Overlay"

    def execute(self, context):
        global overlay_active, handler
        if overlay_active:
            if handler:
                bpy.types.SpaceView3D.draw_handler_remove(handler, 'WINDOW'
    ↪ )
            overlay_active = False
        else:
            handler = bpy.types.SpaceView3D.draw_handler_add(
    ↪ draw_callback_px, (self, context), 'WINDOW', 'POST_PIXEL')
            overlay_active = True
        return {'FINISHED'}

# UI Panel
class MQTTOverlayPanel(bpy.types.Panel):
    bl_label = "MQTT Overlay"
    bl_idname = "VIEW3D_PT_mqtt_overlay"
```

```
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "Sensor"

    def draw(self, context):
        layout = self.layout
        layout.operator("wm.toggle_mqtt_overlay", text="Hide␣Overlay" if
    ↪ overlay_active else "Show␣Overlay")
```

Στο Σχήμα 7 παρουσιάζονται τα στάδια για την ενεργοποίηση και χρήσης του UI overlay. Αρχικά το UI overlay ειναι απενεργοποιημένο (Σχήμα 7a). Με την ενεργοποίηση του από τον χρήστης, ο επιλεγμένος αισθητήρας δεν έχει λάβει ακόμα δεδομένα και για τον λόγο αυτό εμφανίζεται το σχετικό μήνυμα(Σχήμα 7b). Τέλος, οταν ο αισθητήρας στείλει δεδομένα, τα λαμβάνει το script και τα εμφανίζει στο UI overlay (Σχήμα 7c). Για να αποκρύψει το UI overlay, ο χρήστης αρκεί να πατήσει το κουμπί ”Hide Overlay”.

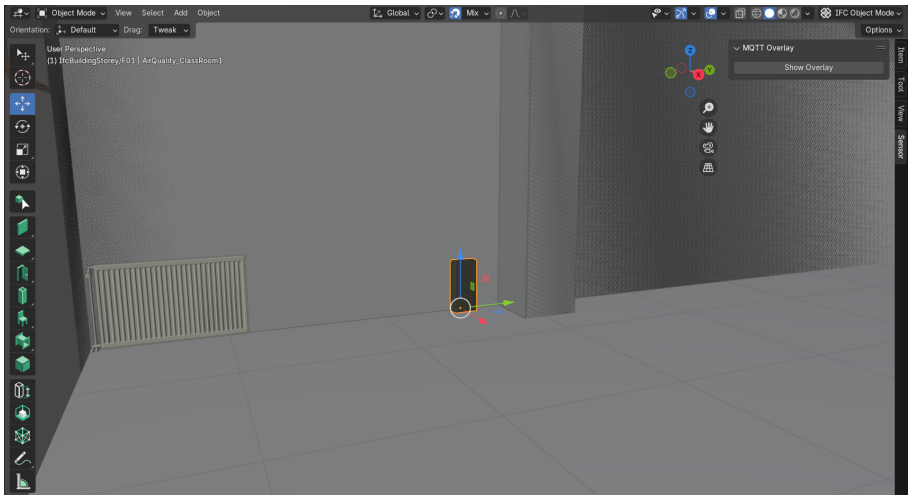
4.4 Εξαγωγή σε glb για Ανάπτυξη Διαδικτυακής Πλατφόρμας

Για την προβολή του τρισδιάστατου μοντέλου στη διαδικτυακή πλατφόρμα, το αρχείο .blend εξήχθη (export) σε μορφή .glb (GL Transmission Format Binary). Η μορφή glb είναι ιδιαίτερα κατάλληλη για χρήση σε διαδικτυακές εφαρμογές καθώς υποστηρίζεται άμεσα από βιβλιοθήκες όπως το three.js [34], επιτρέποντας την αποτελεσματική φόρτωση και αλληλεπίδραση με το μοντέλο στον περιηγητή (browser).

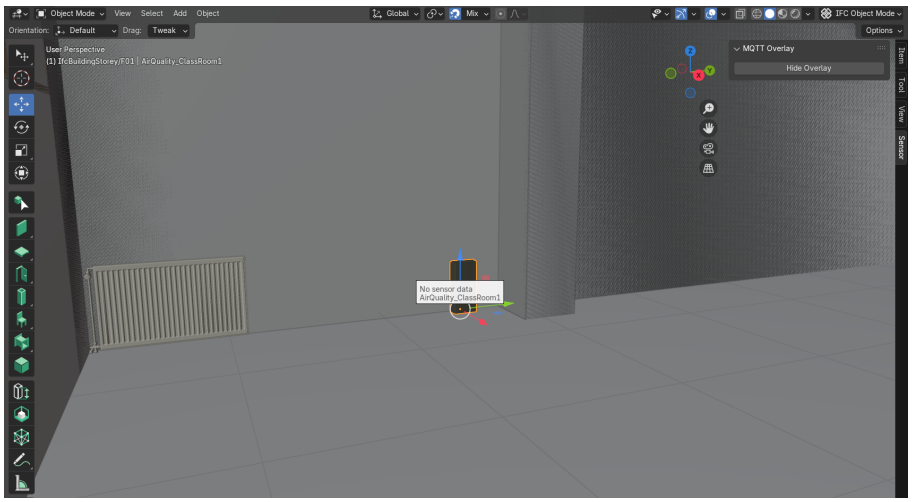
Κατά τη διαδικασία εξαγωγής, δόθηκε έμφαση στη σωστή διατήρηση των γεωμετρικών και υλικών ιδιοτήτων του μοντέλου:

- Έλεγχος και επαλήθευση των UV maps για την ορθή απεικόνιση των υφών.
- Έλεγχος των normals ώστε να εξασφαλιστεί η σωστή κατεύθυνση των επιφανειών και η σωστή απεικόνιση του φωτισμού.
- Συμπερίληψη όλων των απαραίτητων υλικών και textures στο .glb αρχείο, ώστε να μην απαιτούνται εξωτερικά αρχεία κατά τη φόρτωση στο διαδικτυακό περιβάλλον.

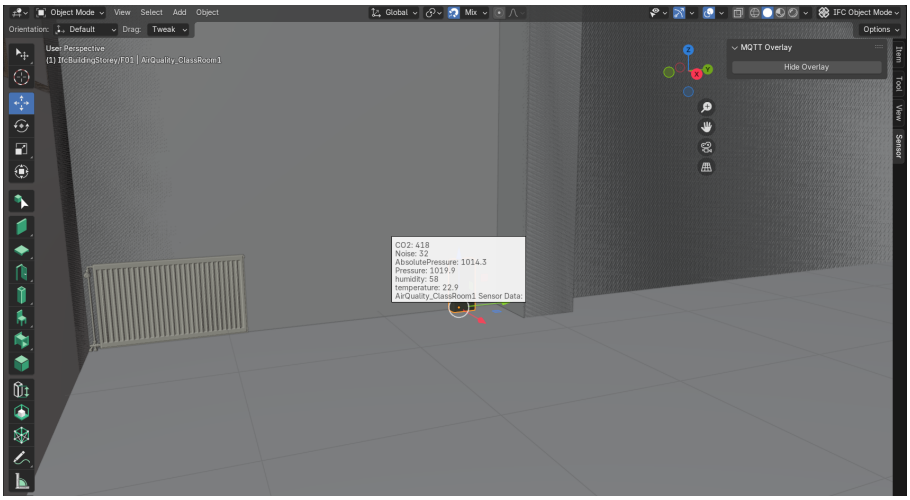
Το παραγόμενο αρχείο .glb χρησιμοποιήθηκε για την τρισδιάστατη απεικόνιση και την αλληλεπίδραση του χρήστη με το Ψηφιακό Δίδυμο.



(a) Hidden Overlay



(b) No Sensor Data



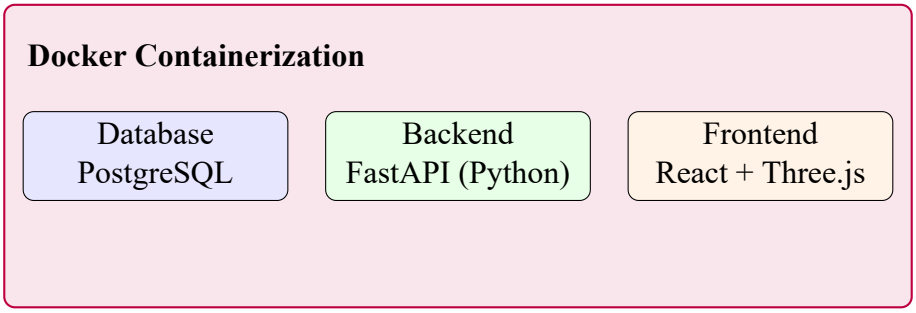
(c) Sensor Data

Σχήμα 7: Διαδοχικά βήματα αλληλεπίδρασης: (a) Απενεργοποιημένο overlay, (b) Ενεργοποιημένο overlay και αισθητήρας χωρίς δεδομένα, (c)Ενεργοποιημένο overlay και αισθητήρας με δεδομένα.

5 Σχεδίαση και υλοποίηση της Διαδικτυακής πλατφόρμας

5.1 Τεχνολογίες

Η ανάπτυξη της διαδικτυακής πλατφόρμας έγινε με χρήση σύγχρονων τεχνολογιών ανοικτού κώδικα (open source) 8. Οι τεχνολογίες αυτές καλύπτουν όλα τα επίπεδα της πλατφόρμας, αποθήκευση δεδομένων (database), backend, frontend. Η επιλογή των τεχνολογιών έγινε με κύριο γνώμονα την ευκολία στην ανάπτυξη της πλατφόρμας, την δυνατότητα για επέκταση και την απόδοση.



Σχήμα 8: Τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της διαδικτυακής πλατφόρμας

5.1.1 Database

Για συστήματος διαχείρισης της βάσης δεδομένων (database) επιλέχθηκε η PostgreSQL. Η PostgreSQL αποτελεί ένα σταθερό σύστημα διαχείρισης βάσεων δεδομένων και υποστηρίζει πολύπλοκες δομές δεδομένων (π.χ. JSON πεδία) οι οποίες επιτρέπουν μια πιο αφηρημένη βάση δεδομένων. Ακόμα παρέχει αξιοπιστία στη διαχείρηση μεγάλου όγκου δεδομένων. Στη βάση δεδομένων αποθηκεύονται όλα τα δεδομένα τα οποία πρέπει να είναι άμεσα και εύκολα προσβάσιμα, χρειάζονται για υπολογισμούς και είναι απαραίτητα για την ομαλή λειτουργία της πλατφόρμας (π.χ οι χρήστες). Μερικά από αυτά είναι οι μετρήσεις των αισθητήρων και τα μεταδεδομένα τους (π.χ. τύπος αισθητήρα, τοποθεσία).

5.1.2 Backend

Η υλοποίηση του Backend έγινε με χρήση του FastAPI. Το FastAPI είναι ένα web framework της Python που παρέχει υψηλές επιδόσεις και πλήρη υποστήριξη για ασύγχρονες κλήσεις (asynchronous requests). Άλλο ένα σημαντικό χαρακτηριστικό του FastAPI είναι ότι παράγει αυτόματα τεκμηρίωση του API (OpenAPI/Swagger) διευκολύνοντας με αυτό τον τρόπο την ανάπτυξη και τη δοκιμή της πλατφόρμας. Το backend είναι υπεύθυνο για την σύνδεση με τον MQTT broker και με όλα τα APIs που χρησιμοποιούνται. Επιπρόσθετα επεξεργάζεται τα δεδομένα των αισθητήρων, τα αποθηκεύει στη βάση δεδομένων και τα παραδίδει στο frontend μέσω μέσω REST API κλήσεων.

5.1.3 Frontend

Για το frontend χρησιμοποιήθηκε η *React*, η οποία προσφέρει modular αρχιτεκτονική και μεγάλη ευελιξία στην ανάπτυξη διαδραστικών διεπαφών. Η τρισδιάστατη απεικόνιση του μοντέλου γίνεται εφικτή με τη χρήση της βιβλιοθήκης Three.js. Η Three.js επιτρέπει την επεξεργασία και παρουσίαση αρχείων glTF/.glb απευθείας στον browser. Υποστηρίζει WebGL το

οποίο επιτρέπει να παρουσιάζονται δισδιάστατα και τρισδιάστατα γραφικά στον browser, χωρίς τη χρήση plugins. Η React συνδυάστηκε με βιβλιοθήκες για την οργάνωση των στοιχείων διεπαφής και την παρουσίαση πινάκων, γραφημάτων και αναλυτικών δεδομένων. Επιπλέον, για τη διαχείριση της πλοήγησης μεταξύ διαφορετικών σελίδων χρησιμοποιήθηκε το React Router.

5.1.4 Containerization

Για κτίσιμο και την εκτέλεση της πλατφόρμας χρησιμοποιήθηκε το Docker Container. Το Docker είναι μία πλατφόρμα ανοικτού κώδικα. Αυτοματοποιεί την ανάπτηξη, την εγκατάσταση και την εκτέλεση των εφαρμογών με χρήση των containers. Τα containers είναι περιβάλλοντα εκτέλεσης που χαρακτηρίζονται από την ελαφρότητα και την φορητότητά τους. Περιέχουν μόνο τα απαραίτητα στοιχεία ώστε να μπορεί να λειτουργήσει μία εφαρμογή και δεν εξαρτόνται από το λειτουργικό σύστημα ή το λογισμικό του περιβάλλοντος που γίνεται η εκτέλεση. Η αρχιτεκτονική του Docker είναι βασισμένη στο μοντέλο client-server. Ο Docker client επικοινωνεί με τον Docker daemon (server), ο οποίος είναι υπεύθυνος για τη δημιουργία, εκτέλεση και διαχείριση των containers. Η επικοινωνία αυτή μπορεί να πραγματοποιηθεί είτε μέσω εργαλείων γραμμής εντολών (CLI), είτε μέσω του REST API που παρέχει το Docker. Με την επιλογή χρήσης του Docker, η πλατφόρμα μπορεί να λειτουργήσει σε οποιοδήποτε περιβάλλον χωρίς προβλήματα ασυμβατότητας. Τέλος, η συντήρηση αλλά και η περαιτέρω ανάπτυξη της πλατφόρμας διευκολύνονται σε αρκετά μεγάλο βαθμό.

5.2 Σχήμα Βάσης Δεδομένων

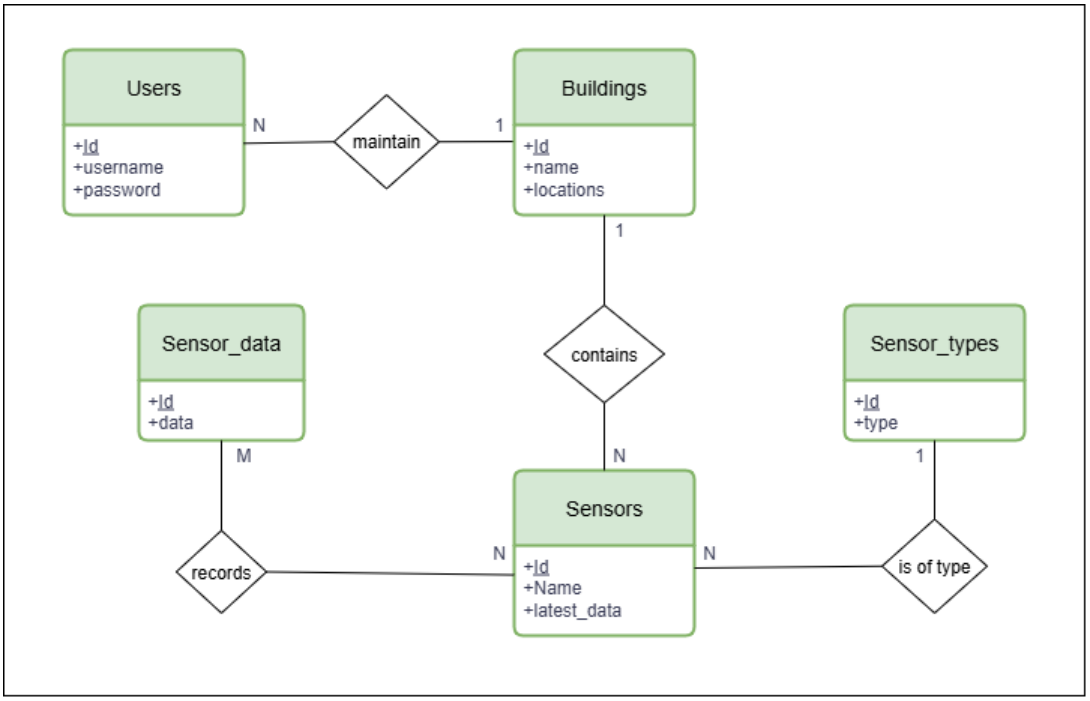
Ο σχεδιασμός της βάσης δεδομένων έγινε με τέτοιο τρόπο ώστε να είναι εφικτή η αποθήκευση και διαχείριση των μετρήσεων των αισθητήρων, των χρηστών και του κτηρίου. Η σχεδίαση και η υλοποίηση ξεκίνησε με τη δημιουργία του εννοιολογικού διαγράμματος (ER Diagram) στο οποίο παρουσιάζονται οι βασικές οντότητες (Users, Buildings, Sensors, Sensor Types, Sensor Data) και οι μεταξύ τους σχέσεις (Σχήμα 9).

Έπειτα, από το εννοιολογικό μοντέλο προχωρήσαμε στο σχεσιακό σχήμα (Relational Scema Diagram), το οποίο αποτυπώνει τους πίνακες, τα πρωτεύοντα κλειδιά και τα ξένα κλειδιά (Σχήμα 10).

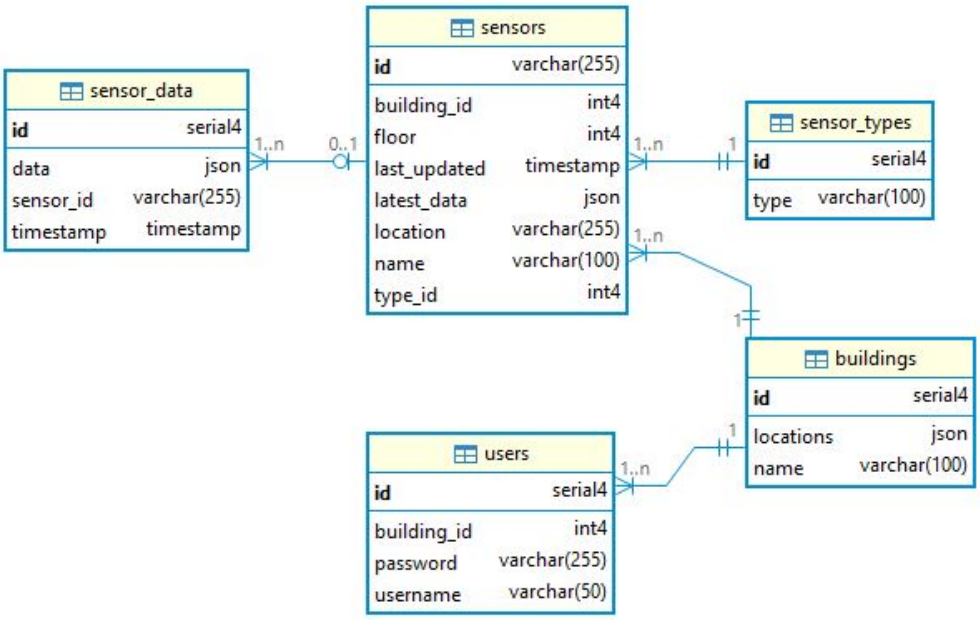
Για τη διευκόλυνση της ανάπτυξης και της αναπαραγωγής του σχήματος, η βάση δεδομένων συνοδεύεται από βοηθητικά αρχεία SQL. Πιο συγκεκριμένα, το αρχείο initialize.sql το οποίο αρχείο περιέχει όλες τις απαραίτητες εντολές (CREATE TABLE, κτλ) και έχει ως σκοπό την αρχικοποίηση του συστήματος με τη δημιουργία μιας «καθαρής» βάσης δεδομένων κατά την εγκατάσταση της πλατφόρμας. Επιπλέον, το αρχείο backup.sql περιλαμβάνει τόσο το σχήμα της βάσης δεδομένων όσο και κάποια δείγματα δεδομένων που συλλέχθηκαν μέσα σε κάποιο χρονικό διάστημα. Χρησιμοποιήθηκε κατά τη διάρκεια της ανάπτυξης της πλατφόρμας ώστε η βάση δεδομένων να περιέχει πάντα πραγματικά δεδομένα απαραίτητα για τις δοκιμές και τους ελέγχους. Τέλος, σε περίπτωση που χρειάζεται να τροποποιηθεί η δομή της βάσης δεδομένων και να προστεθούν ή να αφαιρεθούν στοιχεία, δημιουργείται ένα νέο αρχείο migration.sql το οποίο πρέπει να περιέχει όλες τις αντίστοιχες εντολές (ALTER TABLE, DROP, ADD COLUMN κτλ.) και εκτελείται προκειμένου να ενημερωθεί το υπάρχον σχήμα χωρίς να χαθούν τα δεδομένα.

Το τελικό σχήμα της βάσης δεδομένων περιλαμβάνει τους παρακάτω πίνακες:

- **Buildings** — Πληροφορίες για τα κτήρια.
  - id : SERIAL **NOT NULL** — Μοναδικό αναγνωριστικό κτηρίου. **PK**



Σχήμα 9: Εννοιολογικό μοντέλο (ER Diagram) της βάσης δεδομένων.



Σχήμα 10: Σχεσιακό μοντέλο με πίνακες και σχέσεις (Relational Schema).

  - name : VARCHAR(100) **NOT NULL** — Ονομασία κτηρίου.
  - locations : JSON (DEFAULT {}) — Χώροι/Δωμάτια του κτηρίου.

- **Users** — Χρήστες συσχετισμένοι με συγκεκριμένο κτήριο.
  - id : SERIAL **NOT NULL** — Μοναδικό αναγνωριστικό χρήστη. **PK**
  - username : VARCHAR(50) **NOT NULL UNIQUE** — Όνομα χρήστη (μοναδικό).
  - password : VARCHAR(255) **NOT NULL** — Κωδικός Χρήστη.
  - building_id : INTEGER **NOT NULL** — Αναφορά στο κτήριο του χρήστη.
  - building_id (**FK**) → buildings(id)

- **Sensor Types** — Λεξικό διαθέσιμων τύπων αισθητήρων (π.χ. AirQuality, EnergyMeter, Heater).
  - id : SERIAL **NOT NULL** — Μοναδικό αναγνωριστικό τύπου. **PK**
  - type : VARCHAR(100) **NOT NULL** — Ονομασία τύπου αισθητήρα.

- **Sensors** — Εγγραφές αισθητήρων με βασικά μεταδεδομένα και τη «τελευταία» μέτρηση.
  - id : VARCHAR(255) **NOT NULL** — Αναγνωριστικό αισθητήρα (από MQTT/σύστημα). **PK**
  - name : VARCHAR(100) **NOT NULL** — Φιλικό όνομα στο σύστημα.
  - location : VARCHAR(255) — Περιγραφή θέσης (π.χ. «Room 101»). NOT NULL
  - floor : INTEGER **NOT NULL** (DEFAULT 0) — Όροφος εγκατάστασης.
  - latest_data : JSON (DEFAULT {}) — Τελευταίο στιγμιότυπο μετρήσεων για γρήγορη πρόσβαση. NULL allowed
  - last_updated : TIMESTAMP — Χρόνος τελευταίας ενημέρωσης. NULL allowed
  - building_id : INTEGER **NOT NULL** — Συσχέτιση με κτήριο.
  - type_id : INTEGER **NOT NULL** — Συσχέτιση με τύπο αισθητήρα.
  - building_id (**FK**) → buildings(id)
  - type_id (**FK**) → sensor_types(id)

- **Sensor Data** — Ιστορικό μετρήσεων ανά αισθητήρα.
  - id : SERIAL **NOT NULL** — Μοναδικό αναγνωριστικό εγγραφής ιστορικού. **PK**
  - sensor_id : VARCHAR(255) **NOT NULL** — Αναφορά στον αισθητήρα. ON DELETE CASCADE
  - data : JSON **NOT NULL** — Μετρήσεις (key–value ζεύγη).
  - timestamp : TIMESTAMP (DEFAULT CURRENT_TIMESTAMP) **NULL allowed** — Χρονική σήμανση τελευταίας μέτρησης.
  - sensor_id (**FK**) → sensors(id)

5.3 Λήψη Δεδομένων από MQTT

Η λήψη και αποθήκευση των μετρήσεων υλοποιήθηκε σε Python. Λειτουργεί ως μια υπηρεσία που συνδέεται στον MQTT broker, εγγράφεται στα *topics* και επεξεργάζεται μηνύματα σε πραγματικό χρόνο. Τα στοιχεία του χρήστη και οι απαραίτητες ρυθμίσεις για την σύνδεση (`MQTT_USERNAME`, `MQTT_PASSWORD`, `MQTT_BROKER`, `MQTT_PORT`) περνούν στο σύστημα ως μεταβλητές περιβάλλοντος, ώστε να μη είναι hardcoded στον πηγαίο κώδικα και να διευκολύνεται η ανάπτυξη σε διαφορετικά περιβάλλοντα για διαφορετικούς χρήστες.

5.3.1 Κατηγοριοποίηση των Αισθητήρων

Πριν ξεκινήσει ο listener, αντλούνται από τη βάση τα *Sensor IDs* και ο αντίστοιχος τύπος τους (*AirQuality*, *EnergyMeter*, *Heater*) ώστε να είναι δυνατή η γρήγορη κατηγοριοποίηση των μηνυμάτων όταν ξεκινήσει η διαδικασία παραλαβής μετρήσεων.

Listing 6: Ανάκτηση τύπων αισθητήρων από τη βάση

```
def get_sensor_ids():
    conn = connect_to_db()
    if conn:
        cursor = conn.cursor()
        cursor.execute("SELECT␣sensor.id,␣stypes.type␣FROM␣sensors␣as␣sensor
    ↪ ,␣sensor_types␣as␣stypes␣WHERE␣sensor.type_id␣=␣stypes.id")
        rows = cursor.fetchall()
        cursor.close()
        conn.close()
        for sensor_id, sensor_type in rows:
            if sensor_type == "AirQuality":
                AIR_QUALITY_IDS.add(sensor_id)
            elif sensor_type == "EnergyMeter":
                ENERGY_METER_IDS.add(sensor_id)
            elif sensor_type == "Heater":
                HEATER_IDS.add(sensor_id)
```

5.3.2 Σύνδεση στον MQTT Broker

Ο MQTT πελάτης ξεκινά σε ξεχωριστό daemon thread, συνδέεται στον MQTT Broker και εγγράφεται σε όλα τα θέματα (#), ενώ τα εισερχόμενα μηνύματα δρομολογούνται στην on_message. Η on_message λειτουργεί ως event listener και είναι υπεύθυνη για την επεξεργασία των δεδομένα όταν αυτά σταλούν από τον MQTT Broker και τα λάβει ο mqtt listener.

Listing 7: Σύνδεση σε broker και βρόχος μηνυμάτων

```
def on_connect(client, userdata, flags, rc):
    print(f"MQTT␣Connected␣in␣sensors_realtime␣(rc={rc})")
    client.subscribe("#")

def mqtt_loop():
    try:
        client = mqtt.Client()
        client.username_pw_set(USERNAME, PASSWORD)
        client.on_connect = on_connect
        client.on_message = on_message
        client.connect(BROKER, PORT, 60)
        client.loop_forever()
    except Exception as e:
```

```
        print(f"MQTT connection error: {e}")

def run_sensor_listener():
    get_sensor_ids()
    mqtt_thread = threading.Thread(target=mqtt_loop, daemon=True)
    mqtt_thread.start()
```

5.3.3 Λήψη και Ανάλυση Δεδομένων

Κάθε MQTT μήνυμα αποκωδικοποιείται (JSON), ελέγχεται η δομή του, και εξάγονται το Timestamp και οι Measurements[0].Value (ζεύγη key: value). Το equipment_id προκύπτει από το τελευταίο τμήμα του topic. Αν ο αισθητήρας ανήκει σε γνωστή κατηγορία τότε οι τιμές του αποθηκεύονται στη βάση δεδομένων.

Listing 8: Parsing μηνύματος και αποθήκευση

```
def on_message(client, userdata, msg):
    try:
        payload = msg.payload.decode("utf-8").strip()
        if not payload.startswith("{"):
            return
        data = json.loads(payload)

        timestamp = data.get("Timestamp")
        measurements = data.get("Measurements", [])
        if not timestamp or not measurements:
            return

        values = measurements[0].get("Value")
        if not isinstance(values, dict):
            return

        equipment_id = msg.topic.split("/")[-1]
        category = categorize_device(equipment_id)

        if category:
            add_sensor_data_to_db(equipment_id, timestamp, values)

    except Exception as e:
        print(f"MQTT error: {e}")
```

5.3.4 Αποθήκευση Δεδομένων στη Βάση

Η αποθήκευση στη βάση δεδομένων υλοποιείται με δύο βήματα: (α) ενημέρωση του latest_data στον πίνακα sensors για άμεση πρόσβαση από το frontend και (β) καταγραφή ιστορικού στον πίνακα sensor_data. Για να μην επιβαρύνεται η βάση δεδομένων, διατηρούνται μόνο οι 20 τελευταίες μετρήσεις για κάθε αισθητήρα.

Listing 9: Ενημέρωση latest και ιστορικότητας στη βάση

```
def add_sensor_data_to_db(equipment_id, timestamp, values):
    conn = connect_to_db()
    cursor = conn.cursor()
    try:
        cursor.execute("""
            UPDATE sensors
```

```
            SET latest_data = %s,
                last_updated = %s
            WHERE id = %s
        """, (json.dumps(values), timestamp, equipment_id))

        # Insert into sensor_data history table
        cursor.execute("""
            INSERT INTO sensor_data (sensor_id, data, timestamp)
            VALUES (%s, %s, %s)
        """, (equipment_id, json.dumps(values), timestamp))

        # Keep only the most recent 20 entries per sensor
        cursor.execute("""
            DELETE FROM sensor_data
            WHERE id IN (
                SELECT id FROM sensor_data
                WHERE sensor_id = %s
                ORDER BY timestamp ASC
                OFFSET 20
            )
        """, (equipment_id,))

        conn.commit()
        # print("Sensor JSON data inserted successfully.")
    except Exception as e:
        conn.rollback()
        print("Error inserting sensor data:", e)
    finally:
        cursor.close()
        conn.close()
```

Η παραπάνω ροή επιτρέπει αξιόπιστη λήψη και άμεση διάθεση των δεδομένων προς το frontend (μέσω του latest_data), ενώ παράλληλα διατηρείται ένα σύντομο ιστορικό για δημιουργία γραφημάτων και εκπαίδευση των μοντέλων μηχανικής μάθησης που χρησιμοποιούνται στην πλατφόρμα.

5.4 Σχεδιασμός API

5.4.1 Στόχοι σχεδίασης

Οι στόχοι της σχεδίασης του API είναι η δημιουργία ενός απλού και αποτελεσματικού τρόπου πρόσβασης στα δεδομένα. Δόθηκε ιδιαίτερη έμφαση στην αξιοπιστία, τη λειτουργικότητα και την δυνατότητα επέκτασης. Πιό συγκεκριμένα, το API σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι κατανοητό και εύκολο στη χρήση. Αυτό επιτυγχάνεται με τη σαφή και περιγραφική ονοματολογία των endpoints και με χρήση τυποποιημένων σχημάτων JSON. Επιπρόσθετα, η επεκτασιμότητα επιτυγχάνεται με χρήση versioning και καλά ορισμένων μοντέλων που επιτρέπουν την προσθήκη καινούργιων λειτουργιών. Τέλος, το σύστημα πρέπει να είναι ασφαλές, με χρήση authentication και κατάλληλους μηχανισμούς ελέγχου πρόσβασης για την προστασία των δεδομένων των χρηστών.

5.4.2 Βασικοί πόροι

Οι βασικοί πόροι του API είναι οι κύριες οντότητες του συστήματος και παρέχουν τα απαραίτητα endpoints για την αλληλεπίδραση με τα δεδομένα. Συγκεκριμένα, το API παρέχει

πόρους για τα Buildings, Users, Sensors, Sensor Types και Sensor Data. Κάθε πόρος υλοποιείται με σαφή και προβλέψιμο τρόπο, ώστε να διευκολύνεται η πρόσβαση και η επεξεργασία των δεδομένων από το frontend ή από τρίτες εφαρμογές αν χρειαστεί στο μέλλον. Με αυτόν τον τρόπο εξασφαλίζεται η ομοιομορφία και η επεκτασιμότητα της διαδικτυακής πλατφόρμας. Στον πίνακα 1 φαίνονται τα API endpoints που χρησιμοποιήθηκαν. Πιο αναλυτική περιγραφή των Endpoint είναι η περιγραφή που δημιουργήθηκε αυτόματα από το FastAPI και βρίσκετε στα μονοπάτια /docs και /redoc της διαδικτυακής πλατφόρμας.

Method	**Endpoint**	**Περιγραφή**
POST	/api/v1/login	Έλεγχος ταυτότητας χρήστη και παραχώρηση access token για πρόσβαση στην πλατφόρμα.
GET	/api/v1/healthCheck	Έλεγχος λειτουργικότητας του backend API και επιβεβαίωση ότι η υπηρεσία είναι ενεργή.
GET	/api/v1/pmv	Υπολογισμός του δείκτη θερμικής άνεσης PMV με βάση τα πιο πρόσφατα δεδομένα αισθητήρων.
GET	/api/v1/sensors	Επιστροφή όλων των διαθέσιμων αισθητήρων και των τελευταίων μετρήσεών τους.
GET	/api/v1/getSensorNames	Επιστροφή λίστας με τα ονόματα των αισθητήρων που είναι καταχωρημένοι στη βάση.
GET	/api/v1/getSensorData/{sensor_name}	Επιστροφή μετρήσεων για συγκεκριμένο αισθητήρα με βάση το όνομά του.
GET	/api/v1/sensor/status/{ai_enabled}	Επιστροφή κατάστασης ενός αισθητήρα, με δυνατότητα χρήσης AI για πρόβλεψη.
GET	/api/v1/sensor/history/{sensor_id}	Επιστροφή του ιστορικού των δεδομένων για συγκεκριμένο αισθητήρα.
GET	/api/v1/energy	Υπολογισμός μετρικών και επιστροφή δεδομένων για όλους τους αισθητήρες ενέργειας σε πραγματικό χρόνο.
GET	/api/v1/energy/ai/{sensor_id}	Πρόβλεψη επόμενης τιμής κατανάλωσης για συγκεκριμένο αισθητήρα ενέργειας με χρήση ai.
GET	/api/v1/energy/notifications	Επιστροφή ειδοποιήσεων που σχετίζονται με αισθητήρες ενέργειας.
GET	/api/v1/heater	Τρέχουσες μετρήσεις για αισθητήρες θερμότητας.
GET	/api/v1/getSensorNotifications	Επιστροφή ειδοποιήσεων σχετικών με τη λειτουργία των αισθητήρων.
POST	/api/v1/getAIResponse	Αποστολή ερωτήματος στον AI βοηθό και επιστροφή απάντησης με βάση τα δεδομένα της πλατφόρμας.
GET	/api/v1/aiHealthCheck	Έλεγχος λειτουργικότητας της υπηρεσίας AI και επιβεβαίωση ότι είναι διαθέσιμη.

Πίνακας 1: Endpoints του API

5.4.3 Δομή και Μοντέλα Απόκρισης Δεδομένων

Η τυποποίηση των δεδομένων που επιστρέφονται από τα API endpoints είναι κρίσιμη για την ομαλή χρήση τους από το frontend. Σε πολλές περιπτώσεις υλοποιήθηκαν Pydantic models στο FastAPI, τα οποία όριζαν με σαφήνεια τα πεδία και τους τύπους δεδομένων των αποκρίσεων, προσφέροντας αυστηρό έλεγχο για το αν αυτά είναι έγκυρα. Ωστόσο, υπήρχαν περιπτώσεις όπου τα δεδομένα έρχονταν απευθείας από τον MQTT broker με ήδη καθορισμένη και σταθερή JSON μορφή (π.χ. μετρήσεις αισθητήρων). Σε αυτές τις περιπτώσεις δεν χρειάστηκε να δημιουργηθεί κάποια καινούργια δομή, τα δεδομένα αυτά μεταφέρονταν αυτούσια σε JSON μορφή στο frontend.

5.4.4 Σφάλματα και κωδικοί HTTP

Για τη σωστή επικοινωνία μεταξύ backend και frontend, κάθε endpoint επιστρέφει κατάλληλου κωδικούς κατάστασης (HTTP status codes) ώστε να είναι ξεκάθαρο αν μια κλήση ολοκληρώθηκε επιτυχώς ή απέτυχε. Με τον τρόπο αυτό το frontend μπορεί να ενημερώσει τον χρήστη ή να εκτελέσει fallback μηχανισμούς. Οι κωδικοί που χρησιμοποιήθηκαν συνοψίζονται στον Πίνακα 2.

Κωδικός	**Περιγραφή**
200 OK	Η αίτηση ολοκληρώθηκε επιτυχώς και επιστράφηκαν τα δεδομένα.
400 Bad Request	Η αίτηση περιέχει μη έγκυρα δεδομένα ή λείπουν υποχρεωτικά πεδία.
401 Unauthorized	Αποτυχία ελέγχου ταυτότητας – απαιτείται login.
403 Database Error	Σφάλμα στην επικοινωνία με τη βάση δεδομένων.
404 Not Found	Ο ζητούμενος πόρος δεν βρέθηκε (π.χ. λάθος ID αισθητήρα).
500 Internal Server Error	Γενικό σφάλμα συστήματος.

Πίνακας 2: Κωδικοί HTTP που χρησιμοποιούνται στο API

5.4.5 Ασφάλεια

Η ασφάλεια της πλατφόρμας βασίζεται σε μηχανισμό ελέγχου ταυτότητας με χρήση JSON Web Tokens (JWT). Κάθε χρήστης αποθηκεύεται στη βάση δεδομένων με μοναδικό username, id και password. Η διαδικασία εισόδου (login) πραγματοποιείται μέσω του endpoint /api/v1/login, όπου γίνεται έλεγχος των αναγνωριστικών του χρήστη στη βάση δεδομένων και αν αυτά είναι έγκυρα, δημιουργείται ένα JWT token με διάρκεια ζωής 1 ώρα.

Το JWT περιέχει τα βασικά στοιχεία του χρήστη (sub) και την ημερομηνία λήξης (exp). Όταν δημιουργηθεί το token, αποθηκεύεται στο sessionStorage του browser και στέλνεται σε όλες τις επόμενες κλήσεις API μέσω του Authorization header ως Bearer token. Ο server, σε κάθε αίτηση, επαληθεύει την εγκυρότητα του token και ελέγχει την ημερομηνία λήξης του. Σε περίπτωση που το token έχει λήξει ή δεν είναι έγκυρο, επιστρέφεται ο κωδικός 401 Unauthorized και η πρόσβαση στο endpoint απορρίπτεται. Με αυτόν τον τρόπο διασφαλίζεται ότι η πρόσβαση σε ευαίσθητα δεδομένα, όπως μετρήσεις αισθητήρων ή αναλυτικά στοιχεία κατανάλωσης ενέργειας, είναι δυνατή μόνο από εξουσιοδοτημένους χρήστες.

Επιπλέον, έχει υλοποιηθεί μηχανισμός logout στην πλευρά του client. Η λειτουργία αυτή διαγράφει το αποθηκευμένο JWT από το session storage του browser και οδηγεί τον χρήστη στη σελίδα εισόδου, εξασφαλίζοντας ότι μετά από τη λήξη του χρόνου του token ή με αποσύνδεση του από την πλατφόρμα, δεν θα έχει πλέον πρόσβαση στις υπηρεσίες αν δεν κάνει ξανά login

ωστε να πάρει καινούργιο token. Σε επίπεδο backend, όλα τα endpoints απαιτούν ταυτοποίηση και δηλώνουν εξάρτηση από τη συνάρτηση get_current_user, η οποία ελέγχει την εγκυρότητα του token και εμποδίζει μη εξουσιοδοτημένες κλήσεις.

Με την προσέγγιση αυτή, αποφεύγεται η ανάγκη για αποθήκευση καταστάσεων συνεδρίας στο server κομμάτι, ενώ παράλληλα εξασφαλίζεται ότι οι επικοινωνίες παραμένουν ασφαλείς και ότι μόνο χρήστες με έγκυρα αναγνωριστικά έχουν πρόσβαση στα δεδομένα της πλατφόρμας.

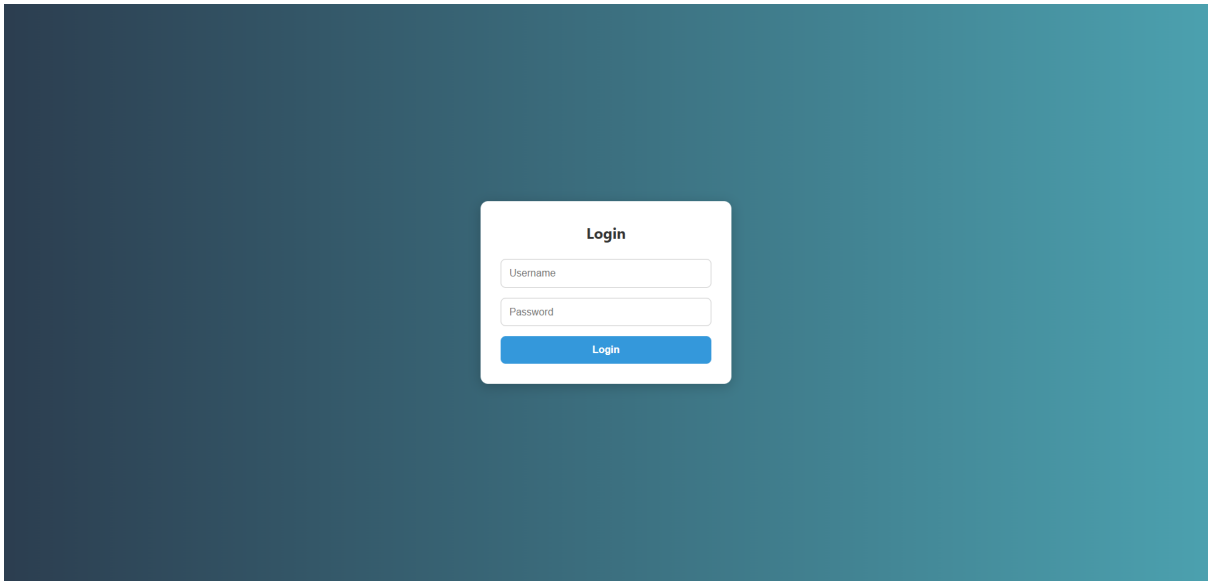
5.5 Frontend: Ενσωμάτωση τρισδιάστατου μοντέλου και στοιχεία UI

Το frontend υλοποιήθηκε με React και οργανώνεται σε επιμέρους σελίδες (routes). Κάθε σελίδα παίρνει δεδομένα από συγκεκριμένα endpoints του backend API και τα παρουσιάζει με κατάλληλα στοιχεία διεπαφής (3D απεικόνιση, πίνακες, γραφήματα, κάρτες με μετρικές) όπως φαίνεται στον πίνακα 3.

5.5.1 Δομή σελίδων και endpoints

Login (/) 11. Σελίδα ελέγχου ταυτότητας χρήστη. Ο χρήστης υποβάλλει τα αναγνωριστικά του και λαμβάνει πίσω ένα JWT token το οποίο αποθηκεύεται στο session storage του browser και είναι απαραίτητο για όλες τις κλήσεις API. Κλήσεις:

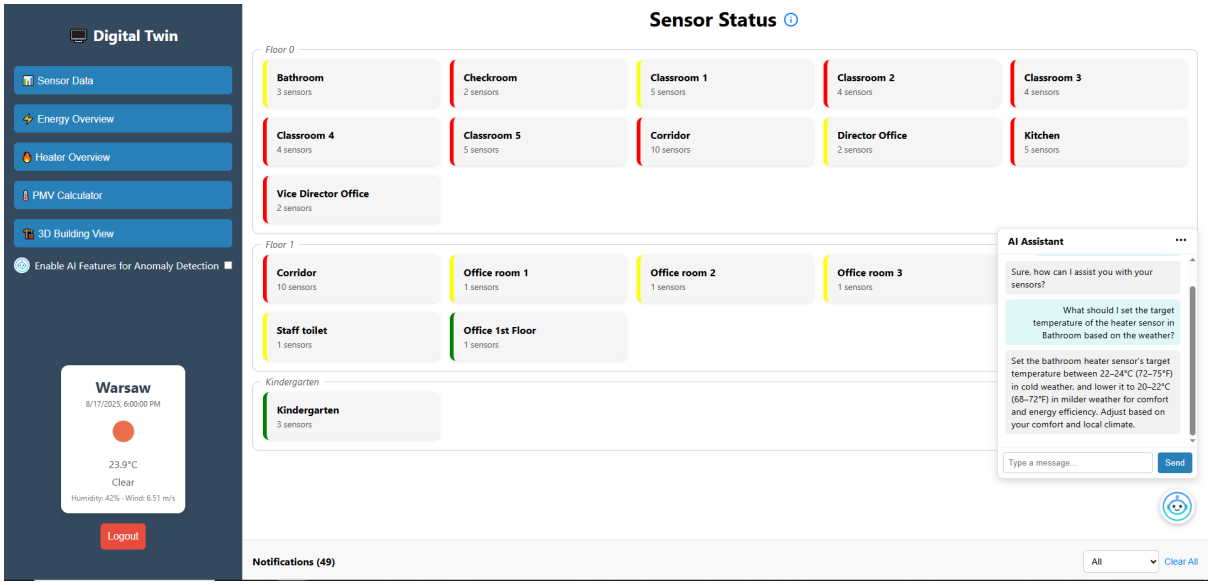
- POST /api/v1/login



Σχήμα 11: Σελίδα εισόδου χρήστη (login).

Dashboard (/dashboard) 12. Σύνοψη κατάστασης συστήματος. Το dashboard αποτελεί το σημείο αναφοράς της διαδυκτιακής πλατφόρμας. Από εδώ ο χρήστης μπορεί να μεταφερθεί σε οποιαδήποτε άλλη σελίδα της πλατφόρμας. Σε αυτή τη σελίδα βρίσκονται οι ειδοποιήσεις για τον κάθε αισθητήρα, η κατάσταση των δωματίων του κτηρίου αλλά και του κάθε αισθητήρα ξεχωριστά. Από το dashboard ο χρήστης μπορεί ακόμα να κάνει χρήση του ai βοηθού και να αποσυνδεθεί (logout) από την πλατφόρμα. Κλήσεις:

- GET /api/v1/pmv (pmv notifications)
- GET /api/v1/getSensorNotifications (heater sensor notifications)
- GET /api/v1/energy/notifications (energy meter notifications)
- GET /api/v1/sensor/status/${aiEnabled}



Σχήμα 12: Κεντρική σελίδα dashboard.

Αισθητήρες (/sensors) 13. Λίστα με όλους τους αισθητήρες του κτηρίου. Εδώ ο χρήστης μπορεί να δει τις τελευταίες μετρήσεις του κάθε αισθητήρα. Η λίστα υποστηρίζει φιλτράρισμα και αναζήτηση καθώς και την επιλογή ”Real-Time Data” που ανανεώνει τις μετρήσεις του κάθε αισθητήρα όταν σταλούν, επιτρέποντας στον χρήστη να παρακολουθεί σε πραγματικό χρόνο τις καινούργιες μετρήσεις που έρχονται στον κάθε αισθητήρα. Κλήσεις:

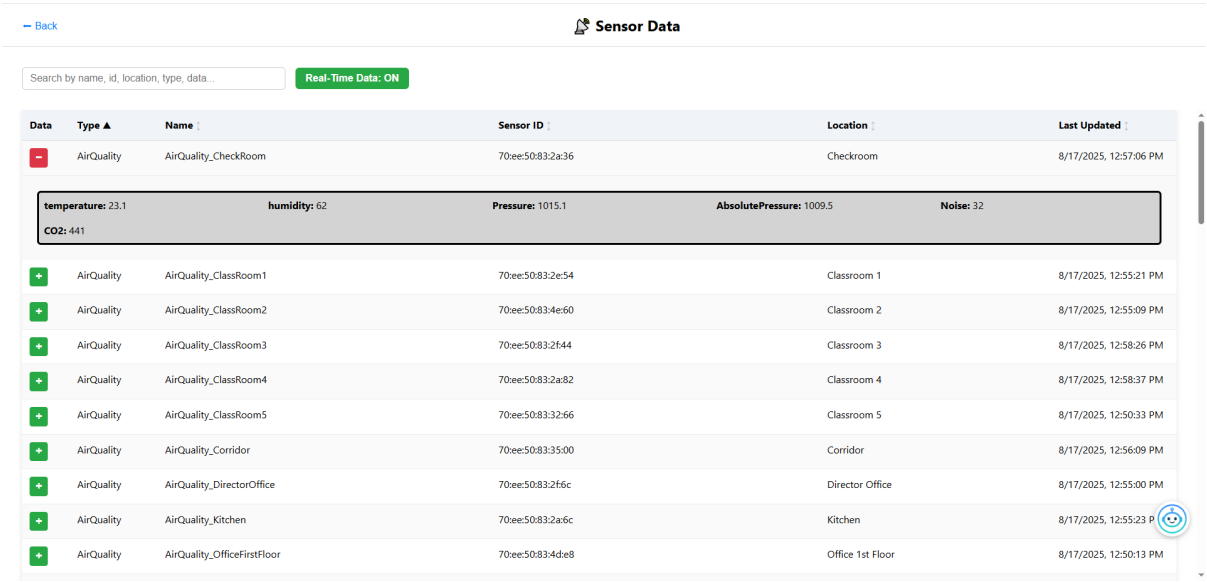
- GET /api/v1/sensors

Ενέργεια (/energy-overview) 14. Συγκεντρωτικά στοιχεία από ενεργειακούς μετρητές. Υπολογίζει κατανάλωση, κόστος, μέγιστη και ελάχιστη τάση, ανισορροπία φάσης, active power και power factor του κάθε αισθητήρα. Επιπρόσθετα κάνει εκτίμηση της επόμενης κατανάλωσης του αισθητήρα με χρήση του μοντέλου LSTM. Κλήσεις:

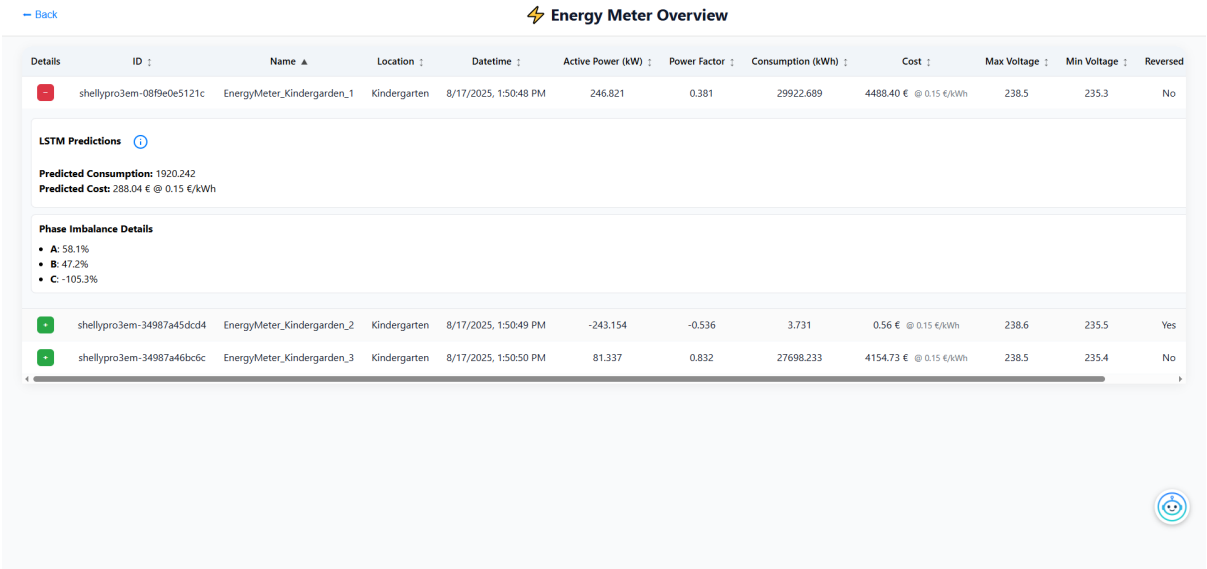
- GET /api/v1/energy
- GET /api/v1/energy/ai/${row_data}

Θέρμανση (/heater-overview) 15. Σε αυτή τη σελίδα παρουσιάζονται σε μορφή λίστας η θέση βαλβίδας, η θερμοκρασία που έχει ρυθμιστεί ο αισθητήρας, η θερμοκρασία του δωματίου, η διαφορά τους και η μπαταρία. Κλήσεις:

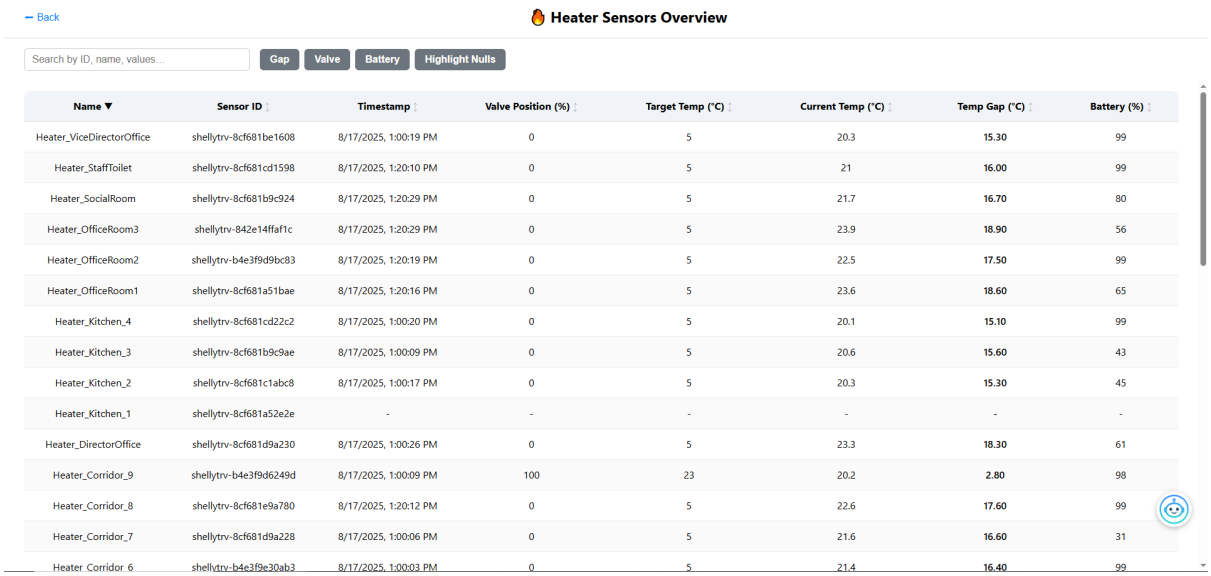
- GET /api/v1/heater



Σχήμα 13: Σελίδα επισκόπησης αισθητήρων (sensors).



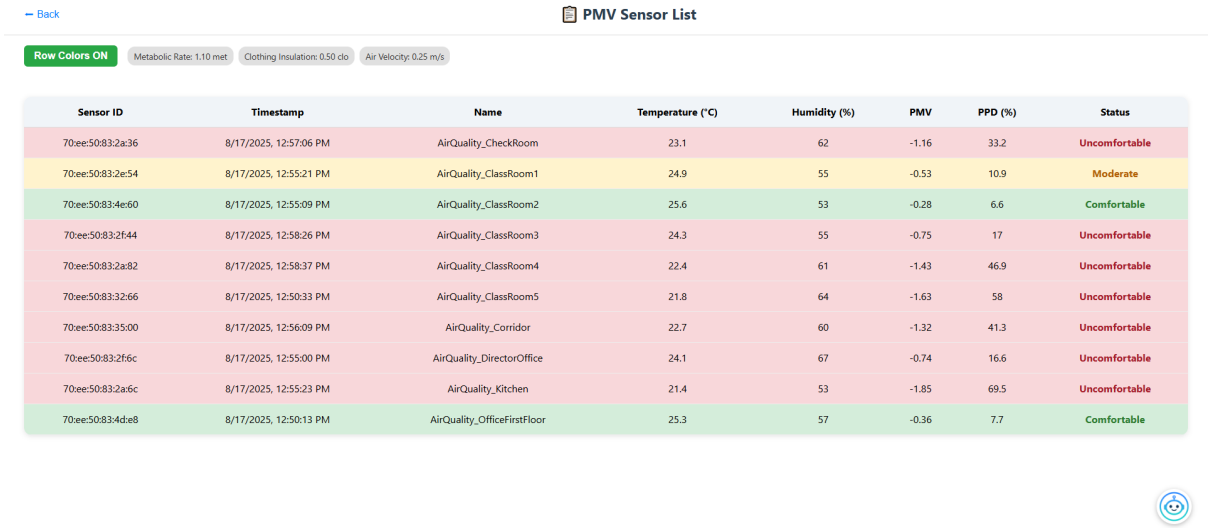
Σχήμα 14: Σελίδα επισκόπησης ενέργειας (energy overview).



Σχήμα 15: Σελίδα παρακολούθησης θέρμανσης (heater overview).

PMV (/pmv) 16. Υπολογισμός και προβολή δείκτη άνεσης PMV/PPD για κάθε αισθητήρα σε μορφή λίστας με χρωματική κωδικοποίηση ανάλογα με το επίπεδο άνεσης. Κλήσεις:

- GET /api/v1/pmv



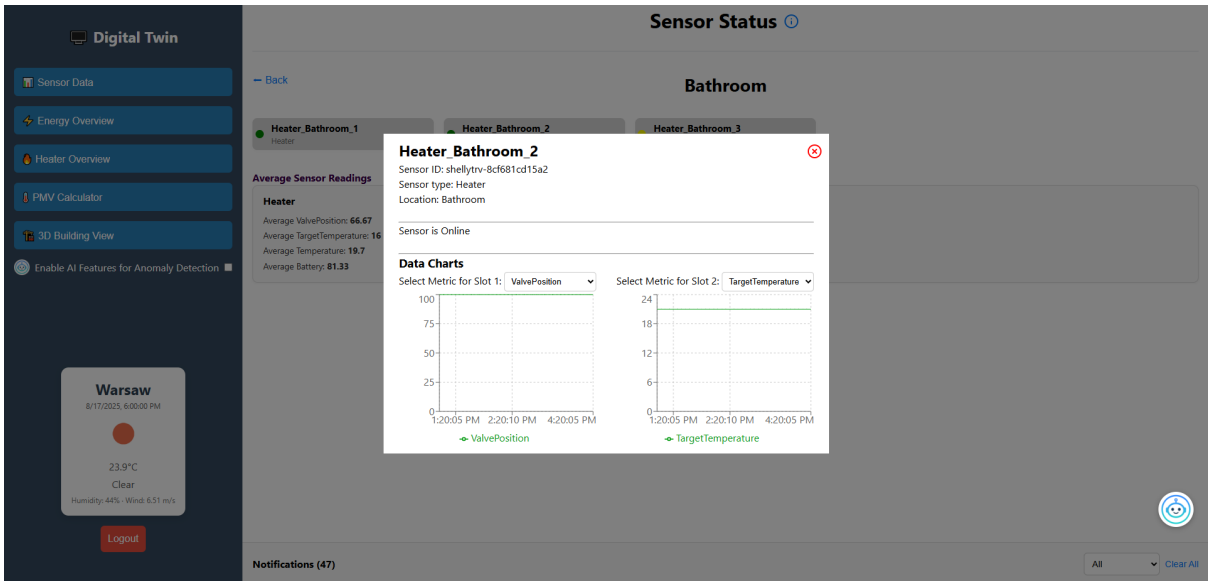
Σχήμα 16: Σελίδα υπολογισμού άνεσης (PMV).

AI Βοηθός (component) 12. Διαλογικό panel ερωτήσεων-απαντήσεων με τον AI βοηθό, που απαντά σε ερωτήσεις σχετικές με αισθητήρες και ψηφιακό δίδυμο. Κλήσεις:

- POST /api/v1/getAIResponse

Γραφική αναπαράσταση ιστορικού (component) 17. Δημιουργεί panel με γραφήματα του ιστορικού μετρήσεων ενός αισθητήρα σε καρτεσιανό σύστημα. Κλήσεις:

- POST /api/v1/sensor/history/${sensorId}



Σχήμα 17: Γραφήματα ιστορικών μετρήσεων.

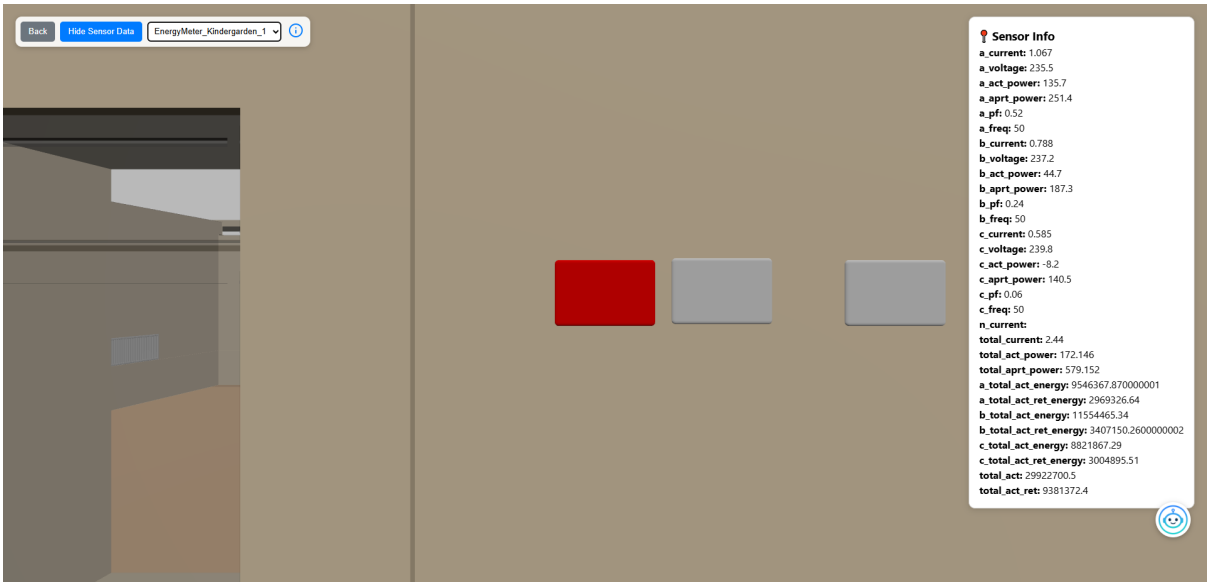
Κατόψεις (/floorplan) 18. Εδώ παρουσιάζονται οι κατόψεις των ορόφων του κτηρίου. Δεν γίνεται κλήση σε κάποιο endpoint επειδή είναι σε μορφή εικόνων αποθηκευμένες στο /public.

Σχήμα 18: Σελίδα κατόψεων ορόφων.

Κτήριο 3D (/building) 19 20. Ενσωμάτωση τρισδιάστατου μοντέλου (.glb) με χρήση three.js. Δυνατότητα επιλογής αισθητήρα από το 3D μοντέλο και εμφάνιση των τελευταίων μετρήσεών του. Κλήσεις:

- GET /api/v1/getSensorNames
- GET /api/v1/getSensorData/${name}

Σχήμα 19: Σελίδα 3D προβολής κτηρίου.



Σχήμα 20: Σελίδα 3D κτηρίου με δεδομένα αισθητήρα.

Σελίδα (route)	**Endpoint**	**Σκοπός χρήσης**
Login (/)	POST /api/v1/login	Έκδοση JWT για προστατευμένες κλήσεις API.
Dashboard (/dashboard)	GET /api/v1/pmv	Ειδοποιήσεις άνεσης (PMV/PPD) και συνοπτικά στοιχεία.
	GET /api/v1/getSensorNotifications	Κεντρική λίστα ειδοποιήσεων αισθητήρων.
	GET /api/v1/energy/notifications	Ειδοποιήσεις/alerts από ενεργειακούς μετρητές.
	GET /api/v1/sensor/status/{aiEnabled}	Έλεγχος κατάστασης/λειτουργιών με ή χωρίς AI.
Αισθητήρες (/sensors)	GET /api/v1/sensors	Λίστα αισθητήρων με τελευταίες μετρήσεις.
	GET /api/v1/getSensorData/{name}	Λεπτομέρειες/τιμές συγκεκριμένου αισθητήρα.
	GET /api/v1/sensor/history/{sensorId}	Ιστορικό μετρήσεων για γραφήματα.
	GET /api/v1/sensor/status/{aiEnabled}	Κατάσταση αισθητήρων (π.χ. online/alerts).
	GET /api/v1/getSensorNotifications	Ειδοποιήσεις για επιλεγμένους αισθητήρες.
Ενέργεια (/energy-overview)	GET /api/v1/energy	Μετρικές κατανάλωσης, τάσης, ισχύος, PF.
	GET /api/v1/energy/ai/{sensorId}	Πρόβλεψη επόμενης κατανάλωσης (LSTM).
Θέρμανση (/heater-overview)	GET /api/v1/heater	Τρέχουσες τιμές θερμοστατικών βαλβίδων.
PMV (/pmv)	GET /api/v1/pmv	Υπολογισμός και οπτικοποίηση PMV/PPD.
Κτήριο 3D (/building)	GET /api/v1/getSensorNames	Αντιστοίχιση αντικειμένων 3D με αισθητήρες.
	GET /api/v1/getSensorData/{name}	Εμφάνιση τελευταίων τιμών στο 3D overlay.
AI Βοηθός (component)	POST /api/v1/getAIResponse	Ερωταποκρίσεις με AI βάσει δεδομένων πλατφόρμας.
Ιστορικό αισθητήρων (component)	GET /api/v1/sensor/history/{sensorId}	Panel γραφημάτων ιστορικών μετρήσεων.

Πίνακας 3: Αντιστοίχιση σελίδων frontend με τα αντίστοιχα API endpoints.

5.5.2 Σημειώσεις υλοποίησης

Τα δεδομένα ανακτώνται με κλήσεις fetch/axios με προσθήκη του JWT στο Authorization header (Bearer). Για τη σελιδοποίηση και φιλτράρισμα ιστορικών δεδομένων χρησιμοποιούνται query παράμετροι στα αντίστοιχα endpoints. Η ενσωμάτωση του .glb πραγματοποιείται με three.js, ενώ το UI ενημερώνεται με περιοδικά αιτήματα ή, όπου απαιτείται, με μηχανισμό near real time.

6 Ανάλυση Δεδομένων, Προβλέψεις και Ανίχνευση Ανωμαλιών

6.1 Μεθοδολογία Υπολογισμού PMV

Ο υπολογισμός του PMV (Predicted Mean Vote) και του PPD (Predicted Percentage of Dissatisfied) υλοποιείται στο backend και είναι προσβάσιμα για το frontend από το endpoint/api/v1/pmv. Γίνεται χρήση της python βιβλιοθήκης pythermalcomfort και πιο συγκεκριμένα της συνάρτησης pmv_ppd_iso, η οποία ακολουθεί το πρότυπο ISO 7730:2005 [28]. Ο υπολογισμός γίνεται με τις τελευταίες μετρήσεις που λαμβάνονται από τους αισθητήρες ποιότητας αέρα (AirQuality sensors). Από τις μετρήσεις αυτές χρησιμοποιούνται η θερμοκρασία και η υγρασία στον αέρα.

6.1.1 Ροή δεδομένων

Η διαδικασία στο backend εκτελείται σε τέσσερα βήματα:

1. Ανάκτηση όλων των αισθητήρων τύπου AirQuality από τη βάση δεδομένων και επιλογή των πιο πρόσφατων μετρήσεων θερμοκρασίας και υγρασίας.
2. Εκτέλεση της συνάρτησης pmv_ppd_iso με παραμέτρους tdb, tr, rh, v_air, met, clo.
3. Στρογγυλοποίηση των αποτελεσμάτων (pmv σε 2 δεκαδικά, ppd σε 1) και ταξινόμηση σε κατηγορίες άνεσης.
4. Δημιουργία λίστας JSON με πεδία: id, name, timestamp, temperature, humidity, pmv, ppd, status.

6.1.2 Παράμετροι μοντέλου

Το endpoint `/api/v1/pmv` δέχεται ως query παραμέτρους:

- **met**: μεταβολικός ρυθμός (default: 1.1),
- **clo**: θερμική μόνωση ρουχισμού (default: 0.5),
- **v_air**: ταχύτητα αέρα (default: 0.1 m/s).

Στο frontend, οι παράμετροι met, clo και v_air υπολογίζονται δυναμικά από εξωτερικά μετεωρολογικά δεδομένα (OpenWeather API [35]). Συγκεκριμένα:

Μεταβολικός ρυθμός (met). Χρησιμοποιείται η σταθερή τιμή `met = 1.1`, που αντιστοιχεί σε *καθιστική εργασία γραφείου με ελαφριά δραστηριότητα* (περίπου 64 W/m²), όπως προτείνεται από το ISO 7730:2005. Δεν γίνεται προσαρμογή με βάση εξωτερικές συνθήκες.

Θερμική μόνωση ρουχισμού (clo). Η παράμετρος `clo` εξαρτάται από την εξωτερική θερμοκρασία που λαμβάνεται από το API και υπολογίζεται με τμηματική συνάρτηση:

$$clo(T_{out}) = \begin{cases} 1.0 & T_{out} \leq 0°C \\ 0.9 & 0°C < T_{out} \leq 10°C \\ 0.8 & 10°C < T_{out} \leq 15°C \\ 0.7 & 15°C < T_{out} \leq 20°C \\ 0.5 & 20°C < T_{out} \leq 26°C \\ 0.36 & T_{out} > 26°C \end{cases}$$

Έτσι, για χαμηλές θερμοκρασίες υιοθετείται μεγαλύτερη θερμική μόνωση ρουχισμού, ενώ σε υψηλές θερμοκρασίες η τιμή αυτή μειώνεται.

Ταχύτητα αέρα (v_air). Το `v_air` προκύπτει από την ταχύτητα εξωτερικού ανέμου (v_{out}) με γραμμική αναγωγή στο εσωτερικό περιβάλλον:

$$v\_air = \min\big(\max(0.05,\ 0.12 \cdot v\_out),\ 0.25\big)$$

δηλαδή η εξωτερική ταχύτητα πολλαπλασιάζεται με συντελεστή 0.12 (υπόθεση για την απόσβεση του αέρα στο εσωτερικό). - Αν η τιμή είναι μικρότερη από 0.05 m/s, τότε ορίζεται στο ελάχιστο όριο (τυπική φυσική ροή σε εσωτερικό χώρο). - Αν ξεπερνά το 0.25 m/s, τότε κόβεται στο μέγιστο όριο (όριο άνεσης σε εσωτερικούς χώρους).

Θερμοκρασία ακτινοβολίας (tr). Για απλοποίηση θεωρούμε `tr = tdb` (θερμοκρασία ακτινοβολίας ίση με θερμοκρασία αέρα), κάτι που συνηθίζεται όταν γίνονται μετρήσεις χωρίς ειδικό αισθητήρα ακτινοβολίας.

6.1.3 Κανόνες κατηγοριοποίησης

Για την παρουσίαση των αποτελεσμάτων στο frontend εφαρμόζονται τα ακόλουθα κατώφλια:

$$\text{status} = \begin{cases} \text{Comfortable}, & |PMV| < 0.5 \\ \text{Moderate}, & 0.5 \leq |PMV| < 0.7 \\ \text{Uncomfortable}, & |PMV| \geq 0.7 \end{cases}$$

Η κατηγοριοποίηση συνοδεύεται από χρωματική επισήμανση στο frontend:

- *Comfortable*: πράσινο χρώμα,
- *Moderate*: κίτρινο χρώμα,
- *Uncomfortable*: κόκκινο χρώμα.

Παράλληλα, το κείμενο της κατάστασης εμφανίζεται με τα αντίστοιχα χρώματα αναλόγως της κατάστασης (πράσινο, κίτρινο, κόκκινο).

6.1.4 Έλεγχοι εγκυρότητας

Για έλεγχο της εγκυρότητας λήφθηκαν τα ακόλουθα μέτρα. Μετρήσεις χωρίς θερμοκρασία ή υγρασία θα αγνοούνται, ενώ αποτελέσματα που επιστρέφουν μη αριθμητικές τιμές (NaN) θα απορρίπτονται. Σε περίπτωση που δεν υπάρχουν διαθέσιμοι αισθητήρες, το endpoint επιστρέφει HTTP status 404.

6.1.5 Σύνοψη

Η μεθοδολογία αυτή είναι βασισμένη στο ISO 7730:2005 [28], προσφέρει ένα τυποποιημένο τρόπο αξιολόγησης της θερμικής άνεσης, συνδυάζοντας μετρήσεις από αισθητήρες, δυναμική παραμετροποίηση των μεταβλητών μέσω API καιρού και απλή και κατανοητή απεικόνιση στο UI.

6.2 Ανίχνευση Ανωμαλιών στους Αισθητήρες Ενέργειας

Η ανάλυση των μετρήσεων των μετρητών ενέργειας πραγματοποιείται στο endpoint /api/v1/energy. Για κάθε αισθητήρα τύπου EnergyMeter λαμβάνονται οι τελευταίες τιμές της ενεργού και φαινόμενης ισχύς ανά φάση (A,B,C) και συνολικά, οι φασικές τάσεις, καθώς και το σύνολο των καταναλώσεων. Έπειτα, υπολογίζονται οι δείκτες ποιότητας και οι ανωμαλίες φάσης με βάση τις ακόλουθες εξισώσεις.

6.2.1 Ροή δεδομένων

1. **Ανάκτηση**: Επιλογή όλων των EnergyMeter από τη βάση και μετατροπή των JSON πεδίων σε float στην SQL ((s.latest_data-»'a_act_power')::float, ...-»'a_voltage κ.ο.κ.).

2. **Υπολογισμοί**: Για κάθε γραμμή (αισθητήρα) γίνεται υπολογισμός των power factor, phase imbalance, voltage stats, reverse flow, missing data και εκτίμηση κόστους.

3. **Συναρμολόγηση**: Δημιουργείται αντικείμενο ανά αισθητήρα με τα μεγέθη που υπολογίστηκαν και τις σημάνσεις ανωμαλιών.

4. **Επιστροφή**: Λεξικό JSON {sensor_id → metrics}.

6.2.2 Ορισμοί μεγεθών και τύποι

Έστω οι ενεργές ισχύς ανά φάση P_A, P_B, P_C, η φαινόμενη συνολική ισχύς S_tot και η ενεργός συνολική ισχύς P_tot. Οι φασικές τάσεις είναι V_A, V_B, V_C. Οι συνολικές ενέργειες (εισερχόμενη/επιστρεφόμενη) είναι E_tot, E_ret.

Power factor (PF).

$$\text{PF} \;=\; \begin{cases} \dfrac{P\_tot}{S\_tot}, & S\_tot > 0 \\ 0, & \text{αλλιώς} \end{cases}$$

Ανισορροπία φάσης (Phase Imbalance). Υπολογίζεται ο μέσος όρος ενεργού ισχύος ανά φάση $\bar{P} = \frac{P_A+P_B+P_C}{3}$ (μόνο όταν $P_A + P_B + P_C > 0$). Το ποσοστό απόκλισης κάθε φάσης είναι:

$$\Delta_A = 100 \cdot \frac{P_A - \bar{P}}{\bar{P}}, \quad \Delta_B = 100 \cdot \frac{P_B - \bar{P}}{\bar{P}}, \quad \Delta_C = 100 \cdot \frac{P_C - \bar{P}}{\bar{P}}.$$

Στον κώδικα τα Δ στρογγυλοποιούνται στο ένα δεκαδικό.

Τάσεις φάσεων.

$$V_{\text{max}} = \max(V_A, V_B, V_C), \quad V_{\text{min}} = \min(V_A, V_B, V_C).$$

Αντιστροφή ροής (Reverse flow).

$$\text{reversed} = \begin{cases} \text{True}, & E\_ret > E\_tot \\ \text{False}, & \text{αλλιώς} \end{cases}$$

Έλλειψη δεδομένων (Missing data).

$$\text{missing} = \bigvee_{x \in \mathcal{R}} (x = \text{None}), \quad \text{όπου } \mathcal{R} = \{P_A, P_B, P_C, P\_tot, S\_tot, V_A, V_B, V_C\}.$$

Εκτίμηση κόστους. Με σταθερή τιμή τ (€/kWh) και κατανάλωση kWh $= E\_tot/1000$:

$$\text{cost} \;=\; \tau \cdot \frac{E\_tot}{1000}.$$

Κατά την υλοποίηση χρησιμοποιείται $\tau = 0.15$ για σκοπούς δοκιμής μιας και η πραγματική τιμή δεν είναι γνωστή.

6.2.3 Κανόνες ανίχνευσης ανωμαλιών

Οι κανόνες εφαρμόζονται στο backend και παράγουν flags/τιμές που χρησιμοποιούνται για ειδοποιήσεις και επισημάνσεις στο UI. Τα κατώφλια είναι ρυθμιζόμενα (οι πιο κάτω τιμές είναι ενδεικτικές για σκοπούς τεκμηρίωσης):

- **Υπερβολική ανισορροπία φάσεων**: $\max(|\Delta_A|, |\Delta_B|, |\Delta_C|) > \theta_{\text{imb}}$, π.χ. $\theta_{\text{imb}} = 20\%$.
- **Χαμηλός συντελεστής ισχύος**: PF $< \theta_{\text{pf}}$, π.χ. $\theta_{\text{pf}} = 0.85$.
- **Ακραίες τιμές τάσης**: $V_{\text{min}} < \theta_{\text{Vmin}}$ ή $V_{\text{max}} > \theta_{\text{Vmax}}$ (π.χ. 207–253 V για ονομαστικό 230 V).
- **Αντιστροφή ροής**: reversed $=$ True.
- **Ελλιπή δεδομένα**: missing $=$ True.

Οι σημάνσεις (flags) δεν εξάγονται ως ξεχωριστά κλειδιά, όμως όλα τα απαιτούμενα μεγέθη (Δ, PF, V_{min}, V_{max}, reversed, missing) παρέχονται ώστε το frontend να αξιολογεί και να χρωματίζει τα αποτελέσματα ή να στέλνει ειδοποιήσεις.

6.2.4 Πολυπλοκότητα και απόδοση

Ανά αισθητήρα η επεξεργασία είναι $O(1)$ (σταθερός αριθμός πράξεων). Συνολικά, για N αισθητήρες, ο χρόνος είναι $O(N)$ με πολύ μικρό κόστος μνήμης (μόνο οι τρέχουσες τιμές). Η απόκριση είναι κατάλληλη για near real–time υπολογισμούς και παρουσίαση.

6.2.5 Μορφή απόκρισης API

Για κάθε αισθητήρα επιστρέφεται ένα αντικείμενο:

```
{
  "id": "...",
  "name": "...",
  "location": "...",
  "type": "EnergyMeter",
  "datetime": "2025-08-16T14:00:00",
  "total_active_power_kw": 1.234,
  "consumption_kwh": 123.456,
  "cost": 18.52,
  "max_voltage": 236.50,
  "min_voltage": 231.20,
  "reversed": false,
  "tariff_id": 0,
  "tariff_rate": 0.15,
  "missing": false,
  "power_factor": 0.93,
  "phase_imbalance_percent": { "A": -12.3, "B": 5.7, "C": 6.6 }
}
```

Το phase_imbalance_percent δίνει τα $\Delta_A, \Delta_B, \Delta_C$ σε ποσοστό (%), ενώ τα missing και reversed λειτουργούν ως δυαδικά flags (boolean).

6.3 Isolation Forest: Εκπαίδευση, Ρύθμιση και Ανάπτυξη

Για την ανίχνευση ανωμαλιών σε διαφορετικούς τύπους αισθητήρων χρησιμοποιείται το μοντέλο Isolation Forest [36]. Η προσέγγιση αυτή είναι προσβάσιμη από το endpoint /api/v1/sensor /status/{ai_enabled}, το οποίο επιστρέφει την κατάσταση κάθε αισθητήρα και όταν το ai_enabled είναι true επιστρέφει και το αποτέλεσμα του Isolation Forest.

6.3.1 Ροή δεδομένων

Η διαδικασία εκτελείται σε τέσσερα στάδια:

1. Ανάκτηση όλων των αισθητήρων από τη βάση δεδομένων με τα πεδία latest_data και τον τύπο (sensor_type).

2. Ομαδοποίηση των αισθητήρων ανά τύπο ώστε να εκπαιδεύεται διαφορετικό μοντέλο Isolation Forest για κάθε κατηγορία αισθητήρα (π.χ. Heater, AirQuality).

3. Εξαγωγή αριθμητικών χαρακτηριστικών από το latest_data και κατασκευή πίνακα χαρακτηριστικών $X \in \mathbb{R}^{N \times d}$.

4. Εκπαίδευση και εφαρμογή Isolation Forest, σε συνδυασμό με κανόνες ανά τύπο αισθητήρα (π.χ. τιμές μπαταρίας ή βαλβίδας για Αισθητήρα Θερμοκρασίας).

6.3.2 Δομή χαρακτηριστικών

Για κάθε αισθητήρα αναζητούνται τα αριθμητικά πεδία του latest_data. Αν ένα χαρακτηριστικό απουσιάζει τότε του παραθέτεται η τιμή 0. Ο πίνακας χαρακτηριστικών (feature matrix) X κατασκευάζεται με στήλες τα χαρακτηριστικά $\{f_1, f_2, \ldots, f_d\}$ και γραμμές τους αισθητήρες. Παράλληλα υπολογίζεται ο μέσος όρος και η τυπική απόκλιση ανά χαρακτηριστικό ώστε να είναι δυνατή η εξαγωγή των ανωμαλιών με Z-score.

6.3.3 Μοντέλο Isolation Forest

Για κάθε κατηγορία αισθητήρων εκπαιδεύεται το μοντέλο IsolationForest με παραμέτρους:

- `n_estimators=100` (αριθμός δέντρων),
- `max_samples` = $\min(N, 256)$ (δείγματα ανά δέντρο),
- `contamination='auto'` (αυτόματη εκτίμηση ποσοστού ανωμαλιών),
- `random_state=42` (αναπαραγωγικότητα).

Το μοντέλο εκπαιδεύεται σε κάθε κλήση του endpoint με τα τρέχοντα δεδομένα, ώστε να αντανακλά τις πιο πρόσφατες μετρήσεις.

6.3.4 Συνδυασμός με Κανόνες

Μαζί με το αποτέλεσμα του Isolation Forest (labels $\{-1, 1\}$, score), εφαρμόζονται και οι παρακάτω έλεγχοι:

- **Έλεγχοι περιορισμών**: Τα πεδία ValvePosition και Battery πρέπει να ανήκουν στο $[0, 100]$.
- **Στατιστικός έλεγχος (z-score)**: Αν $|z| > 3$ για κάποιο χαρακτηριστικό, σημειώνεται ως anomalous feature.
- **Isolation Forest**: Εάν predict = -1, το status χαρακτηρίζεται red. Αν το score είναι αρνητικό, και δεν υπάρχει άλλη ένδειξη, τότε yellow, αλλιώς green.

6.3.5 Κλίμακα Κατάστασης

Το τελικό status κάθε αισθητήρα προκύπτει από συνάρτηση συνδυασμού κανόνων:

$$\text{status} = \begin{cases} \text{green}, & \text{κανένας κανόνας δεν παραβιάζεται} \\ \text{yellow}, & \text{ήπια παραβίαση (score < 0} \\ & \text{ή low battery/valvePosition)} \\ \text{red}, & \text{σοβαρή παραβίαση (Isolation Forest = -1} \\ & \text{ή battery/valvePosition < 0} \\ & \text{ή battery/valvePosition > 100)} \end{cases}$$

6.3.6 Μορφή απόκρισης API

Για κάθε αισθητήρα επιστρέφεται:

```
{
  "id": 101,
  "type": "Heater",
  "status": "red",
  "score": -0.245,
  "anomalous_features": ["Battery", "ValvePosition"]
}
```

Τα anomalous_features ενημερώνουν το χρήστη για το ποια χαρακτηριστικά αποκλίνουν από τις αναμενόμενες τιμές.

6.3.7 Ενσωμάτωση στο UI

Το frontend εμφανίζει τους αισθητήρες ομαδοποιημένους ανά τοποθεσία και όροφο. Το status αποδίδεται με χρωματική κωδικοποίηση (`green`, `yellow`, `red`) στο LED κάθε κάρτας. Επιπλέον, εμφανίζονται οι μέσοι όροι των χαρακτηριστικών ανά τύπο αισθητήρα για κάθε τοποθεσία, ώστε να παρέχεται πληροφορία τόσο ανά αισθητήρα αλλά και ανά δωμάτιο.

6.4 LSTM: Εκπαίδευση, Ρύθμιση και Ανάπτυξη

Για την πρόβλεψη της επόμενης κατανάλωσης (next-step forecasting) στους μετρητές ενέργειας, χρησιμοποιήθηκε το μοντέλο LSTM (Long Short-Term Memory) [37]. Το μοντέλο εκπαιδεύτηκε και αποθηκεύτηκε. Η φόρτωση του μοντέλου γίνεται στο backend φορτώνοντας τα scaler, imputer, feature schema καθώς και τα βάρη του μοντέλου. Τα αποτελέσματα είναι διαθέσιμα για το frontend από το endpoint /api/v1/energy/ai/{sensor_id}.

6.4.1 Αρχιτεκτονική και υπερπαράμετροι

Το μοντέλο ορίζεται ως δίκτυο nn.LSTM με τελική Linear έξοδο:

- **Μήκος ακολουθίας:** `SEQLEN = 20` χρονικά βήματα.
- **Κρυφές μονάδες:** `HIDDEN_SIZE = 256`.
- **Στρώσεις LSTM:** `NUM_LAYERS = 2`.
- **Dropout:** `DROPOUT = 0.2`.

Η είσοδος έχει σχήμα $[B, T, F]$ (batch, χρόνος, χαρακτηριστικά), ενώ η έξοδος είναι $[B, 1]$ και αντιστοιχεί στην προβλεπόμενη κατανάλωση του επόμενου χρονικού βήματος.

6.4.2 Τεχνουργήματα εκπαίδευσης (artifacts)

Για να είναι εφικτό ένα συνεπές αποτέλεσμα, φορτώνονται τα παρακάτω τεχνουργήματα που προέκυψαν κατά τη διάρκεια της εκπαίδευσης:

- scaler.pkl: κανονικοποίηση όλων των χαρακτηριστικών και της κατανάλωσης.
- imputer.pkl: συμπλήρωση ελλιπών τιμών.

- feature_columns.pkl: ακριβής λίστα/διάταξη των χαρακτηριστικών.
- best_model.pth: βάρη του που προέκυψαν από την εκπαίδευση του LSTM.

Γίνεται έλεγχος ύπαρξης των αρχείων. Αν λείπει κάποιο, το μοντέλο τερματίζεται με σφάλμα (RuntimeError).

6.4.3 Προεπεξεργασία και χαρτογράφηση γνωρισμάτων

Το endpoint λαμβάνει τις 20 τελευταίες μετρήσεις κάθε αισθητήρα ενέργειας από τον πίνακα sensor_data και τις μετασχηματίζει σε ένα ενιαίο DataFrame:

1. **Επέκταση API σχήματος:** _expand_db_rows_to_api_df δημιουργεί στήλες όπως Active Power, Power Factor (μέσος φάσεων), Max/Min Voltage, κ.ά..
2. **Χαρτογράφηση σε training schema:** map_incoming_to_training παράγει ακριβώς τα πεδία που χρησιμοποιήθηκαν στο training (datetime, consumption, cost, max_voltage, min_voltage, purpose, reversed, tariff_id, missing).
3. **One-hot & στοίχιση:** one_hot_and_align εφαρμόζει one-hot encoding στις κατηγορικές (purpose, tariff_id) και προσθέτει στήλες που λείπουν ώστε η τελική διάταξη να ταυτίζεται με το feature_columns.
4. **Impute & scale:** impute_then_scale συμπληρώνει κενές τιμές (imputer) και κλιμακώνει με scaler.

Από τον τελικό πίνακα κρατάμε το πιό πρόσφατο παράθυρο $[T{=}20]$ και σχηματίζουμε τον πίνακα $[1, 20, F]$ για τον LSTM.

6.4.4 Inference και αντιστροφή κλίμακας

Η πρόβλεψη $\hat{y}$ παράγεται σε χώρο όπου έγινε κανονικοποίηση (scaled space). Για να ανακτηθεί η πραγματική τιμή (consumption), χρησιμοποιείται η τεχνική προτύπου (_inverse _consumption_with_template) όπου αντικαθίσταται η scaled συνιστώσα consumption στην τελευταία γραμμή χαρακτηριστικών με την $\hat{y}$ και εφαρμόζεται scaler.inverse_transform. Έτσι ώστε να διατηρούνται οι ακριβείς στατιστικές κλίμακας του συνόλου εκπαίδευσης.

6.4.5 Εξαγωγή επόμενου timestamp

Από τα 20 datetime (ιστορικό του κάθε αισθητήρα) υπολογίζεται το επικρατέστερο βήμα (mode των διαφορών). Το next_timestamp ορίζεται ως το τελευταίο datetime συν αυτός ο ρυθμός (cadence) (fallback: 15 min).

6.4.6 Μορφή απόκρισης API

Το endpoint /energy/ai/{sensor_id} επιστρέφει:

```
{
  "id": "shellypro3em-....",
  "next_timestamp": "2025-08-16T13:45:00",
  "predicted_consumption": 1.7823414803
}
```

Αν υπάρξει κάποιο σφάλμα, επιστρέφεται ο κωδικός σφάλματος 404 με το αντίστοιχο μήνυμα (π.χ. «Need at least 20 rows»).

6.4.7 Ενσωμάτωση στο UI

Στη σελίδα «Energy Overview» ο χρήστης επεκτείνει (expand) μία σειρά αισθητήρα, οπότε γίνεται αίτημα στο /energy/ai/{id}. Η πρόβλεψη predicted_consumption εμφανίζεται μαζί με το εκτιμώμενο cost (cost $=$ consumption $\times$ 0.15 €/kWh) και ένα πλαίσιο (modal) που εξηγεί συνοπτικά τη μεθοδολογία LSTM.

6.4.8 Πολιτικές σφαλμάτων και έλεγχοι

- **20 ιστορικά δείγματα:** αν είναι λιγότερα, επιστρέφεται 404 Error.
- **Έλλειψη artifacts:** απουσία scaler/imputer/feature_columns/weights ρίχνει RuntimeError.
- **Ευθυγράμμιση χαρακτηριστικών:** το one_hot_and_align εγγυάται ότι οι στήλες συμφωνούν με το training schema.

6.4.9 Πολυπλοκότητα και απόδοση

Η πολυπλοκότητα είναι $O(T \cdot F \cdot H)$ για τον LSTM (μικρό κόστος για T=20, F δεκάδες, H=256). Το preprocessing (impute, scale, one-hot) είναι γραμμικό. Η καθυστέρηση είναι κατάλληλη για near real-time αλληλεπίδραση ανά αισθητήρα (on demand).

7 Ενσωμάτωση και Υλοποίηση AI Βοηθού

7.1 Συνοπτική αρχιτεκτονική

Ο AI βοηθός λειτουργεί ως ένα bot συνομιλίας που απαντά ερωτήσεις σχετικές με το Digital Twin και τους αισθητήρες. Στην τρέχουσα υλοποίηση δεν έχει πρόσβαση στα δεδομένα της πλατφόρμας (βάση δεδομένων, αισθητήρες, προβλέψεις). Αντίθετα, χρησιμοποιεί το ChatGPT API για να παράγει σύντομες, χρήσιμες απαντήσεις αποκλειστικά πάνω στα παραπάνω θέματα.

Ο χρήστης πληκτρολογεί μία ερώτηση στο chat widget (frontend), η ερώτηση στέλνεται στο backend στο endpoint /api/v1/getAIResponse. Το backend προσθέτει έναν guardrail prompt που περιορίζει το εύρος των ερωτήσεων: «Answer only if it is related to sensors or digital twin; keep it very short» και έπειτα καλεί το ChatGPT API για να πάρει την απάντηση. Η απάντηση επιστρέφεται στο frontend και εμφανίζεται στο chat widget.

7.2 Υλοποίηση AI Βοηθού

7.2.1 Backend

Το backend υποστηρίζει τα παρακάτω endpoints:

- **POST /api/v1/getAIResponse**: δέχεται JSON {message} και καλεί το client.responses .create(...) με model=”gpt-4.1”. Ενσωματώνει προθεματικό prompt που φιλτράρει και περιορίζει τη θεματολογία σε αισθητήρες και ψηφιακό δίδυμο και ζητά πολύ σύντομη απάντηση. Επιστρέφει JSON {response}.
- **GET /api/v1/aiHealthCheck**: απλός έλεγχος συνδεσιμότητας με το API (model=”gpt-3.5-turbo” σε δοκιμαστικό prompt) ώστε να διαγνώσουμε άμεσα ζητήματα δικτύου και σύνδεσης.

Το κλειδί του OpenAI διαβάζεται από την μεταβλητή περιβάλλοντος OPENAI_API_KEY (δεν αποθηκεύονται στον κώδικα). Η πρόσβαση στα endpoints προστατεύεται από JWT (Depends(get_current_user)) όπως όλα τα endpoints, συνεπώς μόνο εξουσιοδοτημένοι χρήστες μπορούν να χρησιμοποιήσουν τον ai βοηθό.

Η υλοποίηση του AI βοηθού στο backend γίνεται με τη χρήση του FastAPI. Παρακάτω φαίνεται ένα απόσπασμα κώδικα που δείχνει πώς γίνεται η λήψη του μηνύματος του χρήστη και η προώθησή του στο ChatGPT API 10:

Listing 10: Endpoint για λήψη AI απαντήσεων

```
@router.post("/getAIResponse")
async def getAIResponse(
    payload: MessageRequest,
    current_user: str = Depends(get_current_user)
):
    try:
        response = client.responses.create(
            model="gpt-4.1",
            input="You␣are␣an␣AI␣assistant.␣Your␣response␣should␣be␣very␣
    ↪ short.␣"
                  "Answer␣the␣following␣question␣only␣if␣it␣is␣related␣to␣
    ↪ sensors␣"
                  "or␣digital␣twin:␣" + payload.message
        )
        return {"response": response.output_text}
```

```
    except Exception as e:
        raise HTTPException(status_code=402, detail=str(e))
```

7.2.2 Πολιτική προτροπών & φιλτράρισμα θεματολογίας

Για να αποφευχθούν παρεκκλίσεις από το θέμα, κάθε ερώτημα συνοδεύεται με κάποιους κανόνες:

1. **Θεματικό φίλτρο:** απαντά μόνο αν η ερώτηση αφορά Digital Twin ή αισθητήρες.
2. **Συντομία:** οι απαντήσεις είναι πολύ σύντομες (direct answer, χωρίς περιττές λεπτομέρειες).
3. **Ουδετερότητα/Ασφάλεια:** αποφυγή ευαίσθητων/προσωπικών δεδομένων (δεν υπάρχει πρόσβαση στα δεδομένα της πλατφόρμας).

Σε μη σχετικές ερωτήσεις, η απάντηση παραμένει ουδέτερη (ανάλογα με το prompt), διατηρώντας τη συνέπεια του ρόλου.

7.2.3 Ασφάλεια & ιδιωτικότητα

- **Χωρίς πρόσβαση σε δεδομένα:** ο βοηθός δεν διαβάζει DB, APIs αισθητήρων ή logs.
- **JWT στο backend:** μόνο συνδεδεμένοι χρήστες μπορούν να καλέσουν /getAIResponse, άρα αποφεύγονται ανώνυμες κλήσεις.
- **Διαμόρφωση κλειδιού:** OPENAI_API_KEY ως μεταβλητή περιβάλλοντος (όχι hardcoded στον κώδικα).

7.2.4 Χειρισμός σφαλμάτων & έλεγχοι υγείας

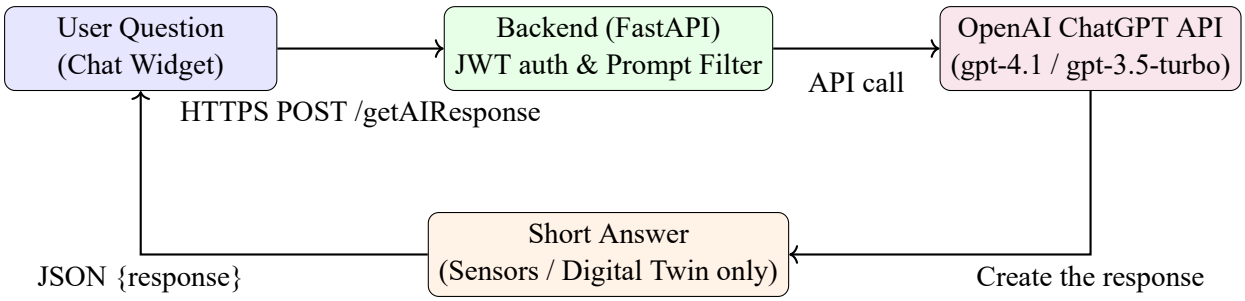
Σε περίπτωση που η κλήση του API αποτύχει (π.χ. σφάλμα δικτύου, λάθος κλειδί), το backend επιστρέφει κωδικό σφάλματος 402 με πληροφορίες για την αιτία σφάλματος. Το /aiHealthCheck παρέχει γρήγορη διάγνωση διαθεσιμότητας/πιστοποίησης προς το ChatGPT API. Δεν είναι προσβάσιμο από το frontend και ο σκοπός του είναι καθαρά για έλεγχο.

7.2.5 Ενσωμάτωση στο frontend

Το chat component είναι διαθέσιμο σε όλες τις σελίδες εκτός της σελίδας login. Κάθε υποβολή αποστέλλεται στο /api/v1/getAIResponse μαζί με το JWT στο Authorization Bearer. Η απάντηση (σύντομο κείμενο) παρουσιάζεται στο panel. Εφόσον δεν υπάρχει πρόσβαση στα δεδομένα, δεν γίνεται εμφάνιση μετρήσεων και διαγραμμάτων. Ο ρόλος είναι καθαρά γνωσιακός και επεξηγηματικός.

7.2.6 Διάγραμμα ροής (AI Assistant)

Το παρακάτω διάγραμμα συνοψίζει τη ροή: User Question→Backend (FastAPI + JWT)→ ChatGPT API→Short Response 21.



Σχήμα 21: Ροή αλληλεπίδρασης AI βοηθού

7.2.7 Όρια και Ασφάλεια Δεδομένων

Σκόπιμα δεν έχει δοθεί πρόσβαση στον AI βοηθό σε πραγματικά δεδομένα (αισθητήρες, ιστορικά, προβλέψεις), καθώς αυτό θα απαιτούσε μηχανισμούς προστασίας από *prompt injection*, data leakage και κακόβουλη χρήση των δυνατοτήτων του μοντέλου. Η ασφαλής ενσωμάτωση μίας τέτοιας λειτουργείας απαιτεί επαγγελματική γνώση στον χώρο της κυβερνοασφάλειας και των μεγάλων γλωσσικών μοντέλων. Σε μελλοντικές εκδόσεις της διαδικτυακής πλατφόρμας, θα μπορούσε να εξεταστεί η χρήση τεχνικών retrieval-augmented generation (RAG) και granular access policies, ώστε να παρέχονται στοχευμένες απαντήσεις χωρίς να τίθεται σε κίνδυνο η ασφάλεια και η ιδιωτικότητα των δεδομένων των χρηστών.

8 Επισκόπηση Αρχιτεκτονικής Συστήματος

8.1 Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram)

Το διάγραμμα περιπτώσεων χρήσης απεικονίζει τις βασικές λειτουργίες που μπορεί να εκτελέσει ο χρήστης μέσα από την πλατφόρμα, καθώς και τις εξωτερικές υπηρεσίες με τις οποίες αλληλεπιδρά το σύστημα (MQTT Broker, Weather AP, ChatGPT API).

Στο σχήμα 22, με τετράγωνο πλαίσιο ορίζεται το όριο συστήματος (Digital Twin web platform) και με τις φιγούρες ορίζονται οι εξωτερικοί δρώντες. Πρωτεύων δρών είναι ο User, ενώ MQTT Broker, Weather API και ChatGPT API είναι δευτερεύοντες δρώντες (εξωτερικά συστήματα με τα οποία αλληλεπιδρά η πλατφόρμα).

8.1.1 Περιπτώσεις χρήσης από τον χρήστη.

Ο χρήστης μπορεί: (α) να συνδεθεί/αποσυνδεθεί (Login/Logout), (β) να δει την κατάσταση των δωματίων και των αισθητήρων, (γ) να πλοηγηθεί σε θεματικές όψεις (Sensor Overview, Heater Overview, Energy Overview, PMV/PPD, 3D Model, Notifications, Weather), (δ) να κάνει ερώτηση στον AI assistant. Οι περιπτώσεις χρήσης είναι ευθυγραμμισμένες με τις σελίδες του frontend και τα endpoints του backend που περιγράφηκαν νωρίτερα.

8.1.2 Σχέσεις include/extend.

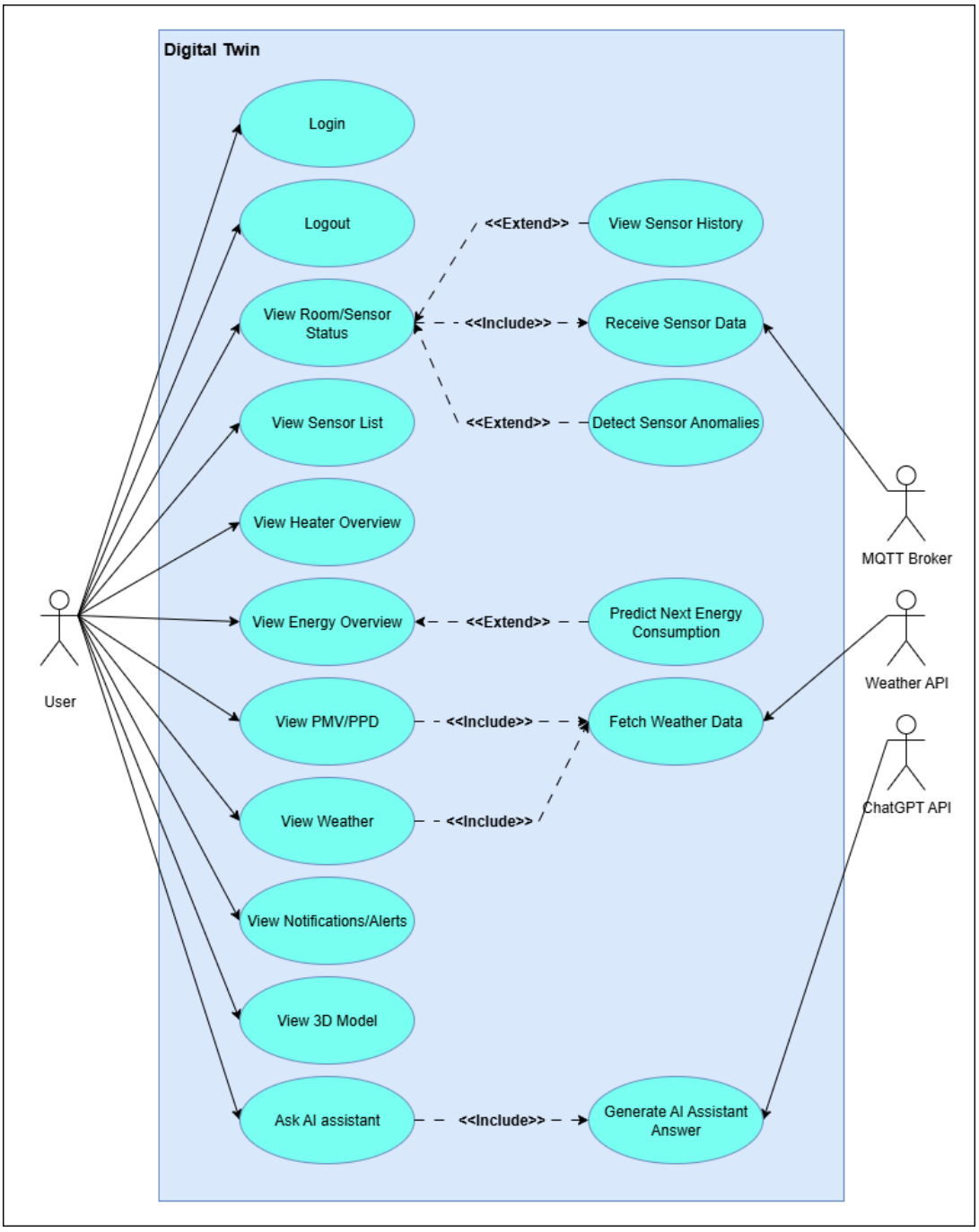
Ορισμένες λειτουργίες συμπεριλαμβάνουν υπο-ροές ή επεκτείνονται από κάποιες προερετηκές δυνατότητες:

- *View Room/Sensor Status* **«include»** *Receive Sensor Data*: η προβολή κατάστασης προϋποθέτει ότι τα τελευταία δεδομένα έχουν ληφθεί ή ενημερωθεί από το MQTT.
- *View Room/Sensor Status* **«extend»** *Detect Sensor Anomalies*: προαιρετική παρουσίαση καταστάσεων μέσω κανόνων και του μοντέλου Isolation Forest.
- *View Energy Overview* **«extend»** *Predict Next Energy Consumption*: ο χρήστης μπορεί να αναπτύξει τη γραμμή και να ζητήσει πρόβλεψη (LSTM) για την κατανάλωση του αισθητήρα.
- *View PMV/PPD* **«include»** *Fetch Weather Data*: η παράμετρος `clo` και `v_air` παράγονται δυναμικά από εξωτερικά μετεωρολογικά δεδομένα.
- *AI assistant* **«include»** *Generate AI Assistant Answer*: προώθηση ερωτήματος στο ChatGPT API, με guardrail prompt.

8.1.3 Αλληλεπιδράσεις με εξωτερικούς δρώντες.

Ο MQTT Broker τροφοδοτεί την πλατφόρμα με μετρήσεις από τους αισθητήρες. Το Weather API παρέχει πληροφορίες για τη θερμοκρασία, άνεμο, υγρασία και ότι άλλο χρειάζεται για παραμετροποίηση των παραμέτρων στον υπολογισμό των PMV και PPD. Το ChatGPT API παράγει σύντομες απαντήσεις σε θεματικές ερωτήσεις (χωρίς πρόσβαση στα δεδομένα της πλατφόρμας).

Με αυτή τη χαρτογράφηση, το διάγραμμα δείχνει τι κάνει ο χρήστης και με ποιους αλληλεπιδρά το σύστημα για να φέρει εις πέρας το κάθε σενάριο.



Σχήμα 22: UML Use Case Diagram

8.2 Διάγραμμα Κλάσεων (Class Diagram)

Το διάγραμμα κλάσεων παρουσιάζει τη δομή του domain model και των βασικών υπηρεσιών στο backend. Ορίζονται οι κλάσεις User, Building, Sensor (και οι υποκλάσεις EnergyMeter, Heater, AirQuality), καθώς και υπηρεσίες όπως AuthenticationService, WeatherService, AIService, SensorService.

Το διάγραμμα αποτυπώνει τις σχέσεις σύνδεσης και κληρονομικότητας, μαζί με τις βασικές μεθόδους κάθε κλάσης.

Το Σχήμα 23 παρουσιάζει το domain model και τις βασικές υπηρεσίες του backend. Η μοντελοποίηση ακολουθεί τις οντότητες της βάσης και τις λειτουργικές ευθύνες των υπηρεσιών.

8.2.1 Οντότητες domain

Building: λογική ομαδοποίηση αισθητήρων και χρηστών. Σχέσεις: Building 1..*—User (πολλοί χρήστες ανήκουν σε ένα κτήριο), Building 1..*—Sensor (πολλοί αισθητήρες ανά κτήριο).

Sensor (αφηρημένη/γενική κλάση): κοινά γνωρίσματα id, name, location, floor, latestData: json, lastUpdated: timestamp και βασικές λειτουργίες: getSensorData(), detectAnomalies(), getHistory(). Η κλάση αυτή αντιστοιχεί στον πίνακα sensors (τρέχουσες μετρήσεις) και στον sensor_data (ιστορικό).

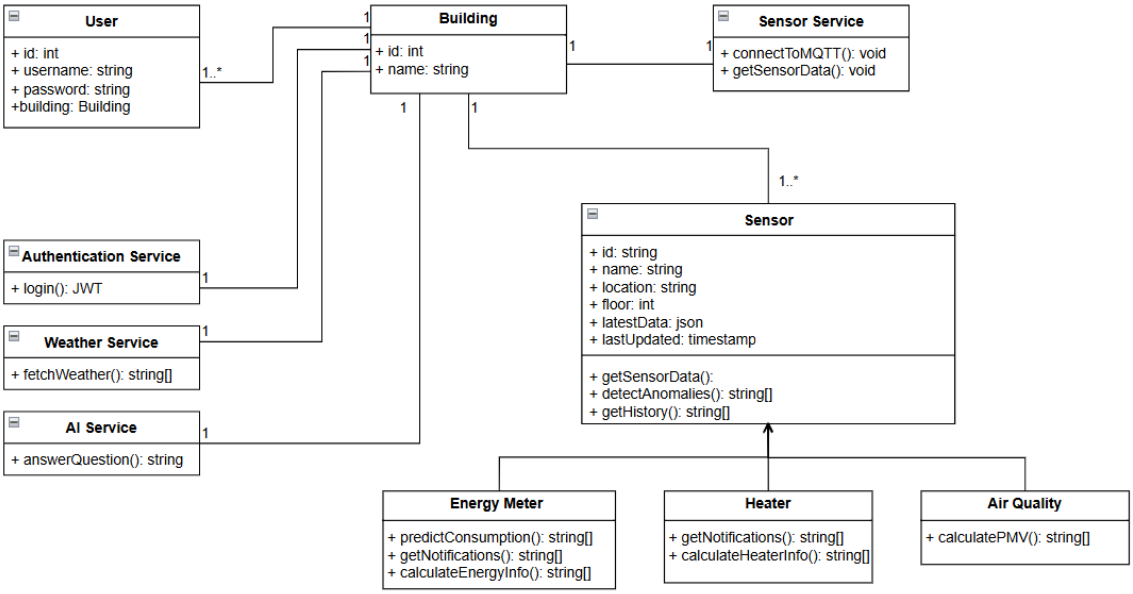
EnergyMeter, **Heater**, **AirQuality**: υποκλάσεις της Sensor με συμπληρωματικές μεθόδους. Το EnergyMeter υλοποιεί predictConsumption() (LSTM), getNotifications() (ανισορροπία φάσεων, κ.α), calculateEnergyInfo((PF, τάσεις, κ.α). Το Heater υλοποιεί getNotifications() (μπαταρία, βαλβίδα) και calculateHeaterInfo(). Το AirQuality υλοποιεί calculatePMV().

8.2.2 Υπηρεσίες (service layer).

SensorService: σύνδεση με MQTT και ανάκτηση/ενημέρωση μετρήσεων.
AuthenticationService: login() και έκδοση JWT.
WeatherService: fetchWeather() για τις παραμέτρους PMV.
AIService: answerQuestion()—κλήση ChatGPT με guardrail prompt.
Οι υπηρεσίες είναι stateless και παραμετροποιούνται μέσω μεταβλητών περιβάλλοντος.

8.2.3 Σχέσεις και σχεδιαστικές επιλογές.

Η κληρονομικότητα από το Sensor μειώνει την επανάληψη κώδικα και επιτρέπει την προσθήκη νέων τύπων αισθητήρων με ελάχιστες αλλαγές.
Με αυτό το διάγραμμα γίνεται σαφής ο καταμερισμός ευθυνών (entities vs services), η ευθυγράμμιση με το σχήμα της βάσης και τα API endpoints, καθώς και ο τρόπος που μπορεί να επεκταθεί το σύστημα (νέοι αισθητήρες ή λειτουργίες) χωρίς να πρέπει να γίνουν αλλαγές στην ήδη υπάρχουσα αρχιτεκτονική.



Σχήμα 23: UML Class Diagram

9 Αποτελέσματα

9.1 Πειραματική Ρύθμιση και Πλατφόρμα Δοκιμών

Για την αξιολόγηση της πλατφόρμας χρησιμοποιήθηκε ένα πραγματικό κτήριο στο οποίο εγκαταστάθηκαν διάφοροι αισθητήρες. Συγκεκριμένα, τοποθετήθηκαν:

- 11 Αισθητήρες ποιότητας αέρα,
- 3 Μετρητές Ενέργειας,
- 41 Αισθητήρες θερμοκρασίας.

Τα δεδομένα μεταδίδονταν σε πραγματικό χρόνο μέσω MQTT broker και αποθηκεύονταν σε βάση δεδομένων (PostgreSQL) για μεταγενέστερη επεξεργασία τους. Το Backend (FastAPI) λειτουργούσε στη θύρα 7781, ενώ το Frontend (React/Three.js) λειτουργούσε στη θύρα 7779. Για την ανάπτυξη και τη διαχείριση του συστήματος χρησιμοποιήθηκαν Docker containers για εξασφάλιση φορητότητας, συμβατότητας με διαφορετικά λειτουργικά συστήματα και ευκολίας κατά την εγκατάστασης. Η μέση συχνότητα δειγματοληψίας ήταν 1 μέτρηση ανά 10 δευτερόλεπτα, ενώ για την πρόβλεψη κατανάλωσης χρησιμοποιήθηκαν παράθυρα 20 βημάτων.

9.2 Μετρήσεις απόδοσης σε πραγματικό χρόνο

Κατά τη διάρκεια των δοκιμών, έγιναν διάφορες μετρήσεις για να μετρηθεί η απόδοση του συστήματος:

- **End-to-end latency (αισθητήρας → UI):** η μέγιστη καθυστέρηση είναι 10 δευτερόλεπτα και εμφανίζεται στη περίπτωση που τα δεδομένα του αισθητήρα έρθουν αμέσος μετά την ανανέωση.
- **Throughput MQTT ingestion:** το σύστημα υποστήριξε περισσότερα από 200 μηνύματα ανά δευτερόλεπτο χωρίς απώλειες.
- **Χρόνος απόκρισης API:** τα περισσότερα endpoints είχαν χρόνο απόκρισης μικρότερο από 200 ms για ένα χρήστη.
- **Απόδοση Frontend:** η οπτικοποίηση του τρισδιάστατου μοντέλου ήταν ομαλή με σταθερό ρυθμό 55–60 FPS χάρη στη χρήση Three.js.

Τα αποτελέσματα δείχνουν ότι η πλατφόρμα μπορεί να χρησιμοποιηθεί για συνεχή παρακολούθηση κτηρίου σε πραγματικό χρόνο, με ελάχιστη καθυστέρηση και χωρίς σημαντικά προβλήματα απόδοσης.

9.3 Ακρίβεια ανίχνευσης ανωμαλιών

Η αξιολόγηση έγινε με πραγματικά δείγματα και πραγματικά σενάρια όπου αυτό ήταν εφικτό.

Phase Imbalance Υπολογίστηκαν συνολικά 40 δείγματα, 20 με πραγματική ανισορροπία φάσης και 20 χωρίς. Το σύστημα:

- **Εντόπισε** 17 από τα 20 δείγματα με ανισορροπία (recall: 85%).
- **Δεν παρήγαγε** ψευδώς θετικά στα 20 δείγματα χωρίς ανισορροπία (specificity: 100%).
- **Συνολική ακρίβεια:** $\frac{17+20}{40} = 92.5\%$.

Isolation Forest Μετά την ενεργοποίηση του αλγορίθμου, εξετάστηκαν 20 αισθητήρες που άλλαξαν κατάσταση. Σε 16/20 περιπτώσεις ο αλγόριθμος εντόπισε outliers που σχετίζονταν με ελλιπή δεδομένα ή δεδομένα με τιμές έξω από το εύρος τιμών τους (ποσοστό επιτυχίας: 80%).

Κανόνες επικύρωσης δεδομένων Οι κανόνες (π.χ. μηδενικές/NaN μετρήσεις) παραμένουν ενεργοί και συμβάλλουν στη μείωση ψευδώς θετικών αποτελεσμάτων, προσθέτοντας αποτελέσματα πάνω στους παραπάνω ανιχνευτές.

9.4 Υπολογισμός Επόμενης Κατανάλωσης (LSTM)

Χρησιμοποιήθηκε LSTM με SEQLEN=96, HIDDEN_SIZE=256, NUM_LAYERS=2 και 7 χαρακτηριστικά εισόδου. Διαστάσεις συνόλου:

- **Input:** $(2857, 96, 7)$, Target: (2857)
- **Train:** $2285 \times 96 \times 7$, Test: $572 \times 96 \times 7$

Εκπαίδευση για 30 εποχές με παρακολούθηση validation loss. Ενδεικτικά αποτελέσματα (Train/Val Loss):

- Εποχές 1–10: σταθερή βελτίωση (π.χ. Epoch 10: Train 0.0927, Val 0.1146).
- Εποχές 11–12: παροδική αύξηση Val Loss (0.3155–0.3202), ένδειξη overfitting που όμως αποτράπηκε.
- Καλύτερες επιδόσεις στις εποχές 17, 19, 29 (ελάχιστο Val Loss: 0.0901 στην Epoch 19).

Συνολικά, το μοντέλο σταθεροποίησε τη γενίκευση σε χαμηλές τιμές σφάλματος (Val Loss ≈ 0.09), με αυξημένες αποκλίσεις σε περιόδους αιχμών, όπως αναμένεται σε δεδομένα με μεγάλη μεταβλητότητα.

9.5 Αποτελεσματικότητα AI Βοηθού

Η αξιολόγηση του AI βοηθού έγινε με python σκριπτ που χρησιμοποιούσε το ίδιο prompt έτσι ώστε να δίνει απαντήσεις μόνο όταν η ερώτηση σχετιζόταν με digital twin ή αισθητήρες. Με βάση τα αποτελέσματα του τεστ:

- **Μέσος χρόνος απόκρισης:** 1.820 s, **διάμεσος:** 1.673 s, **P95:** 3.150 s.
- Ο ai βοηθός, στις περισσότερες περιπτώσεις, απέρριπτε σωστά άσχετες με το θέμα ερωτήσεις και έδινε απάντηση για τις σχετικές.

Από τις δοκιμές φαίνεται ότι ο βοηθός μπορεί στις περισσότερες περιπτώσεις να αναγνωρίσει πότε μια ερώτηση δεν σχετίζεται με το αντικείμενο, δίνει ουσιαστικές και σύντομες απαντήσεις όταν η ερώτηση είναι σχετική με digital twin ή αισθητήρες. Στα δύο παραδείγματα που ακολουθούν παρατηρείται ότι οι ερωτήσει που δεν είναι σχετικές έχουν μικρότερο χρόνο απάντησης.

- **Μη σχετική ερώτηση (σωστή μη-απάντηση):**
  Q62: *Who wrote "Romeo and Juliet"?*
  Response time: 0.731 s
  Απάντηση: *This question is not related to sensors or digital twin.*

- **Σχετική ερώτηση (σωστή απάντηση):**
  Q13: *Ποια είναι τα βασικά στοιχεία αρχιτεκτονικής ενός digital twin πλατφόρμας;*
  Response time: 3.036 s
  Απάντηση (σύνοψη): αισθητήρες για συλλογή δεδομένων, δίκτυα/επικοινωνία, αποθήκευση & επεξεργασία δεδομένων, μοντέλο ψηφιακού διδύμου, διεπαφές απεικόνισης/ελέγχου.

Σε ορισμένες περιπτώσεις το μοντέλο μπορεί να παραπλανηθεί και να απαντήσει σε ερωτήσεις που δεν είναι πραγματικά σχετικές, απλώς επειδή περιέχουν λέξεις-κλειδιά όπως «sensor» ή «digital twin». Αυτό σχετίζεται με γνωστό πρόβλημα prompt injection / keyword baiting (επιρροή της προτροπής από λέξεις-κλειδιά), που οδηγεί σε λανθασμένη ταξινόμηση λόγο συνάφειας.

Συνολικά, οι επιδόσεις είναι θετικές. Βλέπουμε γρήγορη απόκριση (μέσος χρόνος < 2 s) και σωστή διαφοροποίηση μεταξύ σχετικών και μη σχετικών ερωτήσεων, με περιθώριο βελτίωσης απέναντι σε prompt injection / keyword baiting.

10 Συμπεράσματα

Η παρούσα εργασία είχε ως στόχο την ανάπτυξη, υλοποίηση και αξιολόγηση μιας ολοκληρωμένης πλατφόρμας Ψηφιακού Διδύμου για την παρακολούθηση και βελτιστοποίηση κτηριακών εγκαταστάσεων. Το σύστημα συνδυάζει τεχνολογίες συλλογής δεδομένων από IoT αισθητήρες, αποθήκευση και επεξεργασία δεδομένων σε PostgreSQL, παροχή RESTful υπηρεσιών μέσω FastAPI, οπτική διεπαφή χρήστη με React και Three.js και ανάλυση προβλέψεων με μεθόδους μηχανικής μάθησης (Isolation Forest, LSTM). Επιπλέον, ενσωματώθηκε ένας AI βοηθός με χρήση του ChatGPT API, ώστε να παρέχει σύντομες και θεματικά περιορισμένες απαντήσεις γύρω από Ψηφιακό Δίδυμο και τους αισθητήρες.

Τα βασικά συμπεράσματα που προέκυψαν συνοψίζονται ως εξής:

- Η πλατφόρμα απέδειξε ότι η παρακολούθηση κτηριακών συστημάτων σε σχεδόν πραγματικό χρόνο είναι εφικτή, με καθυστέρηση μικρότερη των 2 δευτερολέπτων από τη μέτρηση μέχρι την παρουσίαση στον χρήστη.

- Η ανίχνευση ανωμαλιών με συνδυασμό κανόνων, ελέγχων εγκυρότητας και αλγορίθμων μηχανικής μάθησης έδειξε ικανοποιητική ακρίβεια με περιθώρια βελτίωσης, καθιστώντας δυνατή την έγκαιρη πρόβλεψη προβλημάτων.

- Το μοντέλο LSTM πέτυχε ικανοποιητικά αποτελέσματα στην πρόβλεψη κατανάλωσης ενέργειας, παρέχοντας χρήσιμες ενδείξεις για το επόμενο χρονικό βήμα και τη δυνατότητα εκτίμησης κόστους.

- Ο AI βοηθός λειτούργησε αποτελεσματικά ως υποστηρικτικό εργαλείο,παρόλο που η έλλειψη πρόσβασης σε πραγματικά δεδομένα περιορίζει προς το παρόν την πρακτικότητα του.

- Η χρήση τεχνολογιών όπως Docker containers συνέβαλε στην ευκολία ανάπτυξης, μεταφοράς και χρήσης του συστήματος σε διαφορετικά περιβάλλοντα.

Συνολικά, η εργασία αποδεικνύει ότι η ενσωμάτωση τεχνικών ψηφιακού διδύμου σε κτηριακά συστήματα και η αναπαράστασή τους μέσω μίας διεπαφής χρήστη μπορεί να αποτελέσει ισχυρό εργαλείο για εξοικονόμηση ενέργειας, βελτιστοποίηση λειτουργίας και υποστήριξη αποφάσεων. Η μελλοντική επέκταση με ασφαλή πρόσβαση του AI βοηθού σε δεδομένα, καθώς και η ανάπτυξη της πλατφόρμας ώστε να υποστηρίζει μεγαλύτερα κτηριακά συστήματα, μπορεί να αυξήσει κατά πολύ τη χρησιμότητα της και να την καταστήσει ένα απαραίτητο εργαλείο για τα κτηριακά συστήματα.

11 Μελλοντικές Κατευθύνσεις

11.1 Περιορισμοί & επεκτάσεις

Η εκτίμηση κόστους βασίζεται σε σταθερό τιμολόγιο (τ), ενώ σε πραγματικές συνθήκες απαιτούνται κλιμακωτές χρεώσεις ή χρονικές ζώνες (time-of-use pricing) ανάλογα με την πραγματική τιμή της ενέργειας. Επιπρόσθετα, για την ανίχνευση πιο σύνθετων μοτίβων (π.χ. μεταβατική αστάθεια φάσεων) θα μπορούσε να ενσωματωθεί ανάλυση χρονικών σειρών σε παράθυρα ή η χρήση επιβλεπόμενων μοντέλων (π.χ. LSTM για PF/τάσεις).

11.1.1 Περιορισμοί και επεκτάσεις Isolation Forest

- Το μοντέλο Isolation Forest στην παρούσα υλοποίηση εκπαιδεύεται σε κάθε κλήση με τα πιο πρόσφατα δεδομένα. Για πιο σταθερή συμπεριφορά και ακρίβεια μπορεί να αξιοποιηθεί κάποιο παράθυρο από ιστορικό ώστε να εκπαιδεύεται με περισσότερα δεδομένα.
- Η παράμετρος contamination (μια υπερπαράμετρος που καθορίζει την αναμενόμενη αναλογία ακραίων τιμών στο σύνολο δεδομένων) μπορεί να ρυθμιστεί δυναμικά, με βάση στατιστικές εκτιμήσεις της συχνότητας ανωμαλιών.
- Επέκταση: χρήση ensemble (χρήση περισσότερων από ένα μοντέλο) με άλλες μεθόδους (π.χ. One-Class SVM, Autoencoders) για πιο αξιόπιστη ανίχνευση.

11.1.2 Περιορισμοί και επεκτάσεις LSTM

- **Σταθερό παράθυρο:** η υλοποίηση χρησιμοποιεί παράθυρο μήκους $[T{=}20]$. Μελλοντικά μπορεί να βελτιστοποιηθεί το sequence length.
- **Μονοβηματική πρόβλεψη:** η επέκταση σε multi-step (seq2seq) θα επιτρέψει την εκτίμηση κατανάλωσης για μεγαλύτερα χρονικά περιθώρια.
- **Αβεβαιότητα:** ενσωμάτωση probabilistic forecasts (π.χ. quantiles) για την ποσοτικοποίηση της αβεβαιότητας.
- **Feature drift:** καλά θα ήταν να γίνεται περιοδική επανεκπαίδευση ή online learning ώστε να αντιμετωπίζονται αλλαγές των δεδομένων με την παρόδου του χρόνου και να μένει το μοντέλο up to date.

11.2 Περιορισμοί & μελλοντικές επεκτάσεις AI Βοηθού

- **Τρέχουσα κατάσταση:** ο AI βοηθός δεν έχει πρόσβαση σε πραγματικά δεδομένα και περιορίζεται σε απαντήσεις ερωτήσεων γνώσης (Q&A). Έτσι, δεν μπορεί να απαντήσει σε ερωτήσεις τύπου «ποιος αισθητήρας είναι κόκκινος;».
- **Μελλοντικά:** ελεγχόμενη πρόσβαση σε συγκεκριμένα endpoints (π.χ. read-only) με χρήση retrieval-augmented generation (RAG) και structured outputs, ώστε να δίνει context-specific απαντήσεις με ασφάλεια.
- **Ασφάλεια:** απαιτείται ειδικός σχεδιασμός για την αποφυγή επιθέσεων prompt injection και διαρροής δεδομένων, πιθανώς μέσω εξειδικευμένων policy layers.

11.3 Επέκταση της πλατφόρμας

Εκτός από τις βελτιώσεις των αλγόριθμων, υπάρχουν σημαντικές ιδέες για μελλοντική ανάπτυξης της πλατφόρμας:

- **Υποστήριξη πολλαπλών κτηρίων:** επέκταση της πλατφόρμας ώστε να χειρίζεται δεδομένα από διαφορετικά κτήρια, με συγκεντρωτική ή συγκριτική ανάλυση.
- **Διασύνδεση με BIM:** βαθύτερη σύνδεση με BIM/IFC μοντέλα για αυτοματοποιημένη τοποθέτηση αισθητήρων και enriched 3D visualization.
- **Προηγμένα UI/UX:** χρήση dashboards με διαδραστικά γραφήματα, real-time alerts και notifications (π.χ. push/email).
- **Energy optimization modules:** ενσωμάτωση αλγορίθμων για βελτιστοποίηση καταναλώσεων ή προσομοίωση σεναρίων.
- **Mobile εφαρμογή:** ανάπτυξη lightweight mobile client για άμεση πρόσβαση σε δεδομένα και ειδοποιήσεις.

Συνολικά, η εργασία ανοίγει τον δρόμο για μια πιο ολοκληρωμένη πλατφόρμα Ψηφιακού Διδύμου, η οποία θα μπορεί να λειτουργήσει όχι μόνο ως εργαλείο παρακολούθησης, αλλά και ως υποστηρικτικό σύστημα λήψης αποφάσεων για ενεργειακή αποδοτικότητα και συντήρηση.

Βιβλιογραφία

[1] Enexsa, "Digital twin: From design to operation," in *Enexsa Technical Articles*. Enexsa, 2024, accessed: 2025-08-29. [Online]. Available: https://www.enexsa.com/digital-twin-fdm/

[2] Spiceworks, "What is mqtt? definition, working, and use cases," in *Spiceworks IoT Articles*. Spiceworks, 2024, accessed: 2025-08-29. [Online]. Available: https://www.spiceworks.com/tech/iot/articles/what-is-mqtt/

[3] I. Papias, V. Michalakopoulos, E. Sarmas, and V. Marinakis, "Aggregated flexibility dynamic forecasting of residential energy prosumers for dr programs," in *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2024, pp. 1–9.

[4] V. Michalakopoulos, E. Sarmas, I. Papias, P. Skaloumpakas, V. Marinakis, and H. Doukas, "A machine learning-based framework for clustering residential electricity load profiles to enhance demand response programs," *Applied Energy*, vol. 361, p. 122943, 2024.

[5] I. Papias, V. Michalakopoulos, E. Sarmas, V. Marinakis, G. Antonesi, T. Cioara, and I. Anghel, "A data-driven framework for estimating residential energy flexibility for aggregated demand-side management," *Sustainable Energy, Grids and Networks*, vol. 43, p. 101783, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352467725001651

[6] Wenhui Liu and Yihan Lv and Qian Wang and Bo Sun and Dongchen Han, "A systematic review of the digital twin technology in buildings, landscape and urban environment from 2018 to 2024," *Buildings*, vol. 14, no. 11, p. 3475, 2024.

[7] Muhammad Shahzad and Muhammad Tariq Shafiq and Dean Douglas and Mohamad Kassem, "Digital twins in built environments: An investigation of the characteristics, applications, and challenges," *Journal of Building Engineering*, vol. 57, p. 104858, 2022.

[8] F. Serepas, I. Papias, K. Christakis, N. Dimitropoulos, and V. Marinakis, "Decentralized iot data marketplaces: Enabling peer-to-peer exchange with edge computing," in *2025 IEEE International Workshop on Metrology for Living Environment (MetroLivEnv)*, 2025, pp. 306–311.

[9] F. Serepas, I. Papias, N. Bellos, and V. Marinakis, "A comprehensive approach to real-time and batch processing for energy-efficient iot homes: Leveraging lambda architecture and data lakes," in *2024 15th International Conference on Information, Intelligence, Systems Applications (IISA)*, 2024, pp. 1–6.

[10] Aljawharah A. Alnaser and Mina Maxi and Haytham Elmousalami, "Ai-powered digital twins and internet of things for smart cities and sustainable building environment," *Applied Sciences*, vol. 14, no. 24, p. 12056, 2024.

[11] Yalda Mousavi and Zahra Gharineiat and Armin Agha Karimi and Kevin McDougall and Adriana Rossi, "Digital twin technology in built environment: A review of applications, capabilities and challenges," *Smart Cities*, vol. 7, no. 5, p. 101, 2024.

[12] I. Papias, L. Papapetrou, V. Michalakopoulos, E. Sarmas, V. Marinakis, Z. Mylona, and C. Baslis, "Denoising autoencoder for appliance-specific energy disaggregation in residential settings," in *2025 IEEE International Workshop on Metrology for Living Environment (MetroLivEnv)*, 2025, pp. 384–389.

[13] F. Serepas, I. Papias, K. Christakis, N. Dimitropoulos, and V. Marinakis, "Lightweight embedded iot gateway for smart homes based on an esp32 microcontroller," *Computers*, vol. 14, no. 9, 2025. [Online]. Available: https://www.mdpi.com/2073-431X/14/9/391

[14] V. J. Gan, H. Luo, Y. Tan, M. Deng, and H. H. L. Kwok, "Bim and data-driven predictive analysis of optimum thermal comfort for indoor environment," *Sensors*, vol. 21, no. 13, p. 4401, 2021.

[15] V. Michalakopoulos, I. Papias, E. Sarantinopoulos, E. Sarmas, V. Marinakis, and D. Askounis, "A hyperparameter-space clustering methodology of residential electricity loads," *Applied Soft Computing*, p. 113497, 2025.

[16] V. Michalakopoulos, A. Zakynthinos, E. Sarmas, V. Marinakis, and D. Askounis, "Hybrid short-term wind power forecasting model using theoretical power curves and temporal fusion transformers," *Renewable Energy*, p. 124008, 2025.

[17] Yassine Himeur and Khalida Ghanem and Abdullah Alsalemi and Faycal Bensaali and Abbes Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," *Applied Energy*, vol. 287, p. 116550, 2021.

[18] Zhenghui Mao and Bijun Zhou and Jiaxuan Huang and Dandan Liu and Qiangqiang Yang, "Research on anomaly detection model for power consumption data based on time-series reconstruction," *Energies*, vol. 17, no. 19, p. 4810, 2024.

[19] I. Papias, E. Sarrantinopoulos, V. Michalakopoulos, E. Sarmas, and V. Marinakis, "Flexit: An innovative tool for estimating residential energy flexibility for aggregated & disaggregated demand-side management," SSRN preprint, sep 2025, posted: 18 Sep 2025. [Online]. Available: https://ssrn.com/abstract=5503416

[20] Z. Yang, C. Tang, T. Zhang, Z. Zhang, and D. T. Doan, "Digital twins in construction: Architecture, applications, trends and challenges," *Buildings*, vol. 14, no. 9, p. 2616, aug 2024.

[21] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*. Springer, 2017, pp. 85–113.

[22] J. Vickers, "Nasa digital twin: The next generation," https://www.nasa.gov/sites/default/files/atoms/files/2010_digital_twin.pdf, 2010, accessed: 2025-08-06.

[23] E. H. Glaessgen and D. S. Stargel, "The digital twin paradigm for future nasa and u.s. air force vehicles," NASA/TM-2012-21703, American Institute of Aeronautics and Astronautics, Tech. Rep., 2012, defines Digital Twin as "an integrated multiphysics, multiscale, probabilistic simulation…" (p. 8).

[24] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *Journal of Manufacturing Systems*, vol. 48, pp. 157–169, 2019.

[25] ISO, “Iso 10303-11:2004 – industrial automation systems and integration – product data representation and exchange – part 11: Description methods (express language reference manual),” https://www.iso.org/standard/38047.html, 2004, accessed: 2025-08-06.

[26] Blender Foundation, “Blender – free and open source 3d creation suite,” https://www.blender.org/, 2025, accessed: 2025-08-06.

[27] P. O. Fanger, *Thermal Comfort: Analysis and Applications in Environmental Engineering*. Danish Technical Press, 1970.

[28] International Organization for Standardization, “Iso 7730:2005 – ergonomics of the thermal environment,” https://www.iso.org/standard/39155.html, 2005, accessed: 2025-08-07.

[29] OpenAI, “Chatgpt api,” https://platform.openai.com/docs/guides/chat, 2025, accessed: 2025-08-31.

[30] blender.org, “Bonsai - Blender Add-on for IFC Support,” https://extensions.blender.org/add-ons/bonsai/, 2025, accessed: August 9, 2025.

[31] CGTrader, “Free and premium 3D models,” https://www.cgtrader.com/, 2025, accessed: August 7, 2025.

[32] OASIS Open, “Mqtt version 5.0,” https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, 2019, accessed: 2025-08-06.

[33] R. Light, *Mosquitto: The Lightweight MQTT Broker*. Eclipse Foundation, 2017.

[34] R. Cabello and Contributors, “Three.js: Javascript 3d library,” https://threejs.org/, 2010, accessed: 2025-08-29.

[35] OpenWeather Ltd., “Openweathermap api,” https://home.openweathermap.org/, 2025, accessed: 2025-08-31.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[37] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.