ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

ΑΝΑΠΤΥΞΗ MOBILE ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΕΝΕΡΓΕΙΑΚΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ «ΕΞΥΠΝΟΥΣ» ΜΕΤΡΗΤΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σκαλιδάκη Σταύρου

Επιβλέπων : Χάρης Δούκας
Επίκ. Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

ΑΝΑΠΤΥΞΗ MOBILE ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΕΝΕΡΓΕΙΑΚΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ «ΕΞΥΠΝΟΥΣ» ΜΕΤΡΗΤΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σκαλιδάκη Σταύρου

Επιβλέπων : Χάρης Δούκας

Επίκ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Φεβρουαρίου 2018

……	……	……
Χάρης Δούκας	Ιωάννης Ψαρράς	Δημήτριος Ασκούνης
Επίκ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

.....
Σκαλιδάκης Σταύρος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η διαχείριση ενέργειας και η ανάπτυξη μεθόδων για την επίτευξη της αποτελεί τα τελευταία χρόνια αντικείμενο μελέτης της επιστημονικής κοινότητας. Ένας από τους σημαντικότερους τομείς κατανάλωσης ενέργειας είναι ο κτιριακός τομέας και ως εκ τούτου η μείωση της κατανάλωσης ενέργειας και των εκπομπών διοξειδίου του άνθρακα που προέρχονται από τα κτίρια αποτελεί κύριο στόχο στον χώρο της ενεργειακής και περιβαλλοντικής πολιτικής. Η υπερκατανάλωση της ενέργειας στα κτίρια οφείλεται σε μεγάλο βαθμό στην άγνοια του τελικού καταναλωτή, λόγω ελλιπούς πληροφόρησης, όμως με την σύγχρονη ραγδαία ανάπτυξη της τεχνολογίας και του Internet of Things (IoT) δημιουργήθηκε μια πρωτοφανής παραγωγή ποικίλων δεδομένων που προέρχονται από «έξυπνες» συσκευές. Αυτό με τη σειρά του επιτρέπει πλέον να δημιουργηθούν τεχνολογικές λύσεις, οι οποίες θα είναι ικανές να συλλέξουν και να επεξεργαστούν κατάλληλα τα δεδομένα αυτά, με στόχο την αυτοματοποιημένη και αποδοτική διαχείριση της ενέργειας.

Στο πλαίσιο αυτό, η παρούσα διπλωματική επιχειρεί την ανάπτυξη ενός συστήματος που θα είναι ικανό να διαχειρίζεται αποδοτικά δεδομένα προερχόμενα από «έξυπνους» μετρητές και ταυτόχρονα να προσφέρει αποτελεσματική παρουσίαση αυτών, με στόχο την καλύτερη ενημέρωση του τελικού καταναλωτή, έχοντας ως απώτερο σκοπό τη μείωση της κατανάλωσης ενέργειας και τη βελτίωση των συνθηκών άνεσης ενός κτιρίου. Πιο συγκεκριμένα, η παρούσα εργασία αρχικά περιγράφει τη σχεδίαση και την υλοποίηση ενός συστήματος διαχείρισης δεδομένων προερχόμενα από «έξυπνους» ενεργειακούς μετρητές. Το σύστημα αυτό θα είναι υπεύθυνο για την αποτελεσματική και αποδοτική συλλογή δεδομένων από διαφορετικές πηγές. Ακόμα, θα είναι ικανό να επεξεργαστεί τα παραπάνω δεδομένα και να τα αποθηκεύσει με τέτοιο τρόπο, ώστε να είναι εύκολα διαχειρίσιμα και έτοιμα για ανάλυση από άλλα συστήματα. Ταυτόχρονα, θα αναπτυχθεί μια διεπαφή προγραμματισμού εφαρμογών (API), η οποία είναι υπεύθυνη για την ανάκτηση πληροφοριών και την ασφαλή πρόσβαση στη βάση δεδομένων. Στόχος είναι το σύστημα διαχείρισης δεδομένων να είναι ασφαλές και εύκολα επεκτάσιμο, ώστε να είναι ικανό να δεχτεί δεδομένα από πρόσθετες πηγές. Στη συνέχεια, αναλύεται η σχεδίαση και η ανάπτυξη μιας mobile εφαρμογής, η οποία προσφέρει στον χρήστη εξατομικευμένες πληροφορίες σχετικά με τα ενεργειακά δεδομένα του κτιρίου που τον ενδιαφέρει, καθώς και πρόσθετες λειτουργίες.

Έπειτα από μια ανασκόπηση των βασικών τεχνολογικών εργαλείων και περιβαλλόντων που χρησιμοποιήθηκαν, παρουσιάζεται το κύριο αποτέλεσμα της εργασίας, που είναι μια mobile εφαρμογή, βασισμένη στο λειτουργικό σύστημα Android, ικανή να παρουσιάζει στον τελικό χρήστη τα ενεργειακά δεδομένα που σχετίζονται με τον περιβάλλον εργασίας του. Παρουσιάζονται οι βασικές οθόνες και λειτουργικότητες, μαζί με τις απαραίτητες σχεδιαστικές επιλογές.

Τέλος, αναλύονται τα αποτελέσματα και τα συμπεράσματα τα οποία εξάχθηκαν από την εκπόνηση της διπλωματικής εργασίας, τα πλεονεκτήματα και μειονεκτήματα του συστήματος, καθώς και προτάσεις για μελλοντική επέκταση της εφαρμογής.

Λέξεις Κλειδιά:
Εφαρμογή Android, "έξυπνοι" μετρητές, διαχείριση ενέργειας, ενεργειακά δεδομένα, ενεργειακή παρακολούθηση, σύστημα διαχείρισης δεδομένων

Abstract

To date, energy management has been the subject of study of the scientific community, which made an effort to develop methods for achieving it. One of the most significant types of energy consumption is building-oriented and therefore, the reduction of energy consumption and carbon dioxide emissions from buildings is one of the most important objectives in the field of energy and environmental policies. Overconsumption of energy in buildings results mainly from the ignorance of the consumer regarding the energy data, but with the rapid development of technology and the Internet of Things (IOT), there has been an unprecedented production of various data coming from "smart" devices. This, in turn, allows technological solutions, capable of properly collecting and processing these data, with a view to managing energy automatically and efficiently.

In this context, this diploma thesis attempts to develop a system and a corresponding mobile application, that will be able to efficiently manage data from "smart" meters, at the same time providing an effective presentation and analysis of energy data in order to better inform the final consumer and ultimately reduce the energy consumption and improve the comfort conditions of a building.

More specifically, this paper describes the design and implementation of a data management system. The system will be initially responsible for efficiently collecting data from different sources. It will also be able to process the above data and store it in such a way that is easily manageable and ready for analysis by other systems. Moreover, an application programming interface (API) will be developed, that will be responsible for recovering and safely accessing available information in the database. The intention is for the data management system to be secure, easily expandable and able to accept data from additional sources. Next, a mobile application will be designed and developed, that will provide the user with personalized information about the energy data of the building they are interested in, as well as additional functions.

After a review of the key technology tools used, the main result of the work is presented. It is a mobile application, based on the Android operating system, which is able to present to the final user the energy data related to their work environment. The main screens and the functionality of the application are presented, along with all the necessary design options.

Finally, this paper comprises the results and conclusions drawn by the drafting of the diploma thesis, the system's advantages and disadvantages, as well as proposals for the future expansion of the application.

Keywords:

Android application, mobile, "smart" meters, "smart" buildings, energy management, energy data, energy monitoring, data management system

Πρόλογος

Η παρούσα Διπλωματική Εργασία εκπονήθηκε στο πλαίσιο ολοκλήρωσης του προπτυχιακού κύκλου σπουδών της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Η εργασία ανατέθηκε από το Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης και έχει ως θέμα την ανάπτυξη μιας εφαρμογής σε λειτουργικό σύστημα Android, η οποία θα προσφέρει εξατομικευμένες πληροφορίες στον χρήστη για διάφορα ενεργειακά δεδομένα που σχετίζονται με το κτίριο στο οποίο εργάζεται και άλλες λειτουργίες σχετικά με αυτές τις πληροφορίες.

Στόχος της διπλωματικής εργασίας, είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος συλλογής, επεξεργασίας, αποθήκευσης και παρουσίασης δεδομένων ενεργειακών μεγεθών με το οποίο θα αλληλεπιδρά ο χρήστης μέσω εφαρμογής φορητής συσκευής. Απώτερος σκοπός μέσα από τη συνεχή πληροφόρηση του χρήστη για τα μεγέθη αυτά, είναι η εξοικονόμηση ενέργειας, χρημάτων και η βελτίωση συνθηκών άνεσης στον χώρο εργασίας, τα οποία μελλοντικά και με την προϋπόθεση τη μαζική χρήση τέτοιου τύπου εφαρμογών μπορούν να συμβάλλουν στην περιβαλλοντική διαχείριση.

Πίνακας Περιεχομένων

Πίνακας Εκθεμάτων

Κεφάλαιο 1ο - Εισαγωγή

1.1 Αντικείμενο και Σκοπός

Η εξοικονόμηση ενέργειας και η ανάπτυξη μεθόδων για την αποδοτική χρήση της αποτελεί εδώ και χρόνια αντικείμενο μελέτης πολλών επιστημόνων. Με την πάροδο των χρόνων, ιδιαίτερα στις μεγαλουπόλεις, οι απαιτήσεις για ενέργεια στον οικιακό τομέα, αλλά και σε όλους τους τομείς τις οικονομίας ολοένα και αυξάνονται με υψηλό ρυθμό. Χαρακτηριστικό του φαινομένου αυτού είναι η πρόβλεψη για αύξηση της παγκόσμιας χρήσης ενέργειας κατά σχεδόν 1/3 έως το 2040 [1]. Ταυτόχρονα, για την απρόσκοπτη εξασφάλιση της απαραίτητης ενέργειας οι κύριες πηγές (με μεγάλη διαφορά) είναι το πετρέλαιο, τα ορυκτά καύσιμα και το φυσικό αέριο. Όμως τα αποθέματα αυτών των πηγών είναι περιορισμένα. Ως εκ τούτου, καθίσταται αναγκαία η λήψη σημαντικών μέτρων περιορισμού τουλάχιστον της σπατάλης, έτσι ώστε οι πηγές αυτές να διαρκέσουν περισσότερο έως ότου οι εναλλακτικές πηγές ενέργειας να διαδοθούν περαιτέρω και να επιτευχθεί η απεξάρτηση από τις παραδοσιακές πηγές. Ο περιορισμός αυτός ωστόσο, θα πρέπει να γίνει με γνώμονα την διατήρηση της ομαλής πορείας της οικονομικής δραστηριότητας αλλά και του βιοτικού επιπέδου των ανθρώπων.

Ένας από τα σημαντικότερους τομείς κατανάλωσης ενέργειας είναι ο κτιριακός τομέας, τόσο όσον αφορά τα εμπορικά κτίρια, όσο και τις κατοικίες. Αυτή η τάση παρατηρείται κατά κύριο λόγο στις μεγαλουπόλεις των ανεπτυγμένων χωρών, όπου το 20% με 40% της συνολικής κατανάλωσης ενέργειας πραγματοποιείται σε κτίρια [2, 26]. Ως εκ τούτου, η μείωση της κατανάλωσης ενέργειας και των εκπομπών διοξειδίου του άνθρακα που προέρχονται από τον κτιριακό τομέα αποτελεί έναν από τους σημαντικότερους στόχους στο χώρο της ενεργειακής και περιβαλλοντικής πολιτικής. Σε αυτό το πλαίσιο έχουν εκδοθεί και οι κοινοτικές οδηγίες "Energy Performance of Buildings" (2010/31/EU) [3] και "Energy Efficiency Directive (2012/27/EU) [4]", με στόχο την υιοθέτηση μέτρων για την βελτιστοποίηση της κατανάλωσης ενέργειας που συμβαίνει στα κτίρια.

Η υπερκατανάλωση της ενέργειας στα κτίρια οφείλεται σε μεγάλο βαθμό στην άγνοια του τελικού καταναλωτή, λόγω ελλιπούς πληροφόρησης, όσον αφορά την ενέργεια που καταναλώνεται και πώς αυτή συνδέεται με τη συμπεριφορά του ή με τις διάφορες δραστηριότητές του. Πριν από κάποια χρόνια, η παρακολούθηση και η ανάλυση της κατανάλωσης ήταν ανέφικτη λόγω της αδυναμίας συστηματικής και αυτοματοποιημένης συγκέντρωσης δεδομένων. Με την σύγχρονη ραγδαία ανάπτυξη της τεχνολογίας και του Internet Of Things (IoT) δημιουργήθηκε μια πρωτοφανής παραγωγή ποικίλων δεδομένων προερχόμενες από «έξυπνες» συσκευές. Τα δεδομένα αυτά ωστόσο, από μόνα τους δεν είναι χρήσιμα. Πρέπει να δημιουργηθούν τεχνολογικές λύσεις, οι οποίες θα είναι ικανές να τα συλλέξουν και να τα επεξεργαστούν κατάλληλα, με στόχο την αυτοματοποιημένη και αποδοτική διαχείριση της ενέργειας. Η ενσωμάτωση «έξυπνων» συσκευών στα κτίρια, έχει ως αποτέλεσμα την δημιουργία των λεγόμενων «έξυπνων» κτιρίων, τα οποία είναι σε θέση να παρέχουν πληροφόρηση σχετικά με τα ενεργειακά χαρακτηριστικά τους καθώς και να εκτελούν ευφυείς αυτοματοποιημένες λειτουργίες [5, 6]. Η δημιουργία όμως τέτοιων κτιρίων είναι ακόμα αντικείμενο μελέτης και έρευνας, καθώς υπάρχουν ακόμα αρκετές προκλήσεις, τεχνολογικές και μη για την ενσωμάτωσή τους σε μια «έξυπνη» πόλη [7],

όπως για παράδειγμα η ενσωμάτωση κάτω από ένα κοινό σύστημα πλήθος διαφορετικών τεχνολογιών και συσκευών. Ταυτόχρονα, τα smartphones, tablets, fablets κλπ. έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητας κάτι το οποίο δίνει τη δυνατότητα στους χρήστες να έχουν πρόσβαση σε άμεση επικοινωνία και ενημέρωση. Η επεξεργαστική ισχύς και η μνήμη των συσκευών αυτών έχει φτάσει σε σημείο που μπορούν να παίξουν τον ίδιο και πολλές φορές αποδοτικότερο ρόλο στην διαχείριση και παρουσίαση δεδομένων, προσφέροντας μάλιστα μια ακόμα πιο εξατομικευμένη εμπειρία ανάλογα με τις επιθυμίες του χρήστη. Κατ' επέκταση, η ενημέρωση του καθένα για την κατανάλωση ενέργειας σε ατομικό-οικιακό επίπεδο είναι πλέον εφικτή και μπορεί να αποτελέσει σημαντική συμβολή στην περιβαλλοντική και ενεργειακή διαχείριση. Με λιγοστές κινήσεις, μέσω μιας mobile εφαρμογής, κάποιος είναι δυνατό να πληροφορείται για την ποσότητα ενέργειας που καταναλώνει, για το κόστος της, και την επιβάρυνση που προκαλεί στο περιβάλλον, να παρακολουθεί τα στοιχεία αυτά και τελικά να μειώνει τις σπατάλες έχοντας οικονομικό όφελος χωρίς όμως να επηρεάζονται αρνητικά οι λειτουργικότητες του κτιρίου του. Η χρήση μιας τέτοιας εφαρμογής από μεγάλη μερίδα του πληθυσμού, μπορεί να βοηθήσει σημαντικά στην συνολική εξοικονόμηση ενέργειας, στη μείωση των ρύπων από την παραγωγή της και τελικά να ωφελήσει το περιβάλλον γενικότερα. Μάλιστα, το γεγονός ότι ένα ατομικό ισχυρό κίνητρο για τον καθένα, που είναι η εξοικονόμηση χρημάτων, συμβαδίζει με ένα καθολικό συμφέρον, την ενεργειακή οικονομία του πλανήτη, καθιστά την συγκεκριμένη ιδέα άμεσα εφαρμόσιμη.

Στο πλαίσιο αυτό, η παρούσα διπλωματική επιχειρεί την ανάπτυξη ενός συστήματος και μια σχετικής mobile εφαρμογής, τα οποία θα είναι ικανά να διαχειρίζονται αποδοτικά δεδομένα προερχόμενα από «έξυπνους» μετρητές και ταυτόχρονα να προσφέρουν αποτελεσματική παρουσίαση και ανάλυση των ενεργειακών δεδομένων με στόχο την καλύτερη ενημέρωση του τελικού καταναλωτή ενέργειας και με απώτερο σκοπό τη μείωση της κατανάλωσης ενέργειας καθώς και τη βελτίωση των συνθηκών άνεσης ενός κτιρίου. Πιο συγκεκριμένα, στο πλαίσιο της παρούσας διπλωματικής θα σχεδιαστεί και θα υλοποιηθεί:

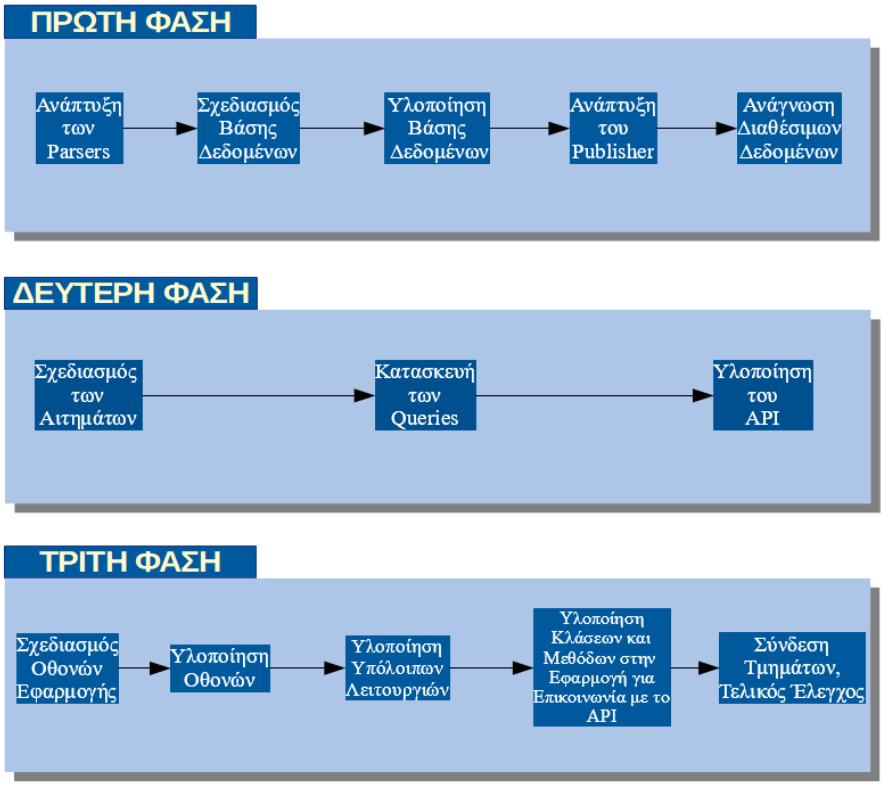
- Ένα σύστημα διαχείρισης δεδομένων προερχόμενα από «έξυπνους» ενεργειακούς μετρητές. Τα σύστημα αυτό θα είναι υπεύθυνο αρχικά για την αποτελεσματική και αποδοτική συλλογή δεδομένων από διαφορετικές πηγές. Ακόμα, θα είναι ικανό να επεξεργαστεί τα παραπάνω δεδομένα και να τα αποθηκεύσει με τέτοιο τρόπο, ώστε να είναι εύκολα διαχειρίσιμα και έτοιμα για ανάλυση από άλλα συστήματα. Ταυτόχρονα, θα αναπτυχθεί μια διεπαφή προγραμματισμού εφαρμογών (API) η οποία είναι υπεύθυνη για την ανάκτηση και την ασφαλή πρόσβαση στις διαθέσιμες πληροφορίες από τη βάση δεδομένων. Στόχος είναι το σύστημα διαχείρισης δεδομένων να είναι ασφαλές και εύκολα επεκτάσιμο, ώστε να είναι ικανό να δεχτεί δεδομένα από πρόσθετες πηγές.
- Μια mobile εφαρμογή, η οποία προσφέρει στον χρήστη εξατομικευμένες πληροφορίες σχετικά με τα ενεργειακά δεδομένα του κτιρίου που τον ενδιαφέρει καθώς και κάποιες πρόσθετες λειτουργίες όπως η πρόβλεψη ορισμένων ενεργειακών μεγεθών σε μελλοντικό χρόνο, η αναζήτηση δεδομένων με βάση διαφορετικά φίλτρα τους και η αποτελεσματική παρουσίαση αυτών στο χρονικό πλαίσιο που ο χρήστης επιθυμεί.

Η ανάπτυξη του παραπάνω συστήματος θα πραγματοποιηθεί με βάση «έξυπνους»

μετρητές και πραγματικά δεδομένα που προέρχονται από τους χώρους ενός εργαστηρίου στο Εθνικό Μετσόβιο Πολυτεχνείο. Στόχος της δημιουργίας του συστήματος, είναι ο χρήστης στο εργαστήριο, με βάση το πόστο του, να μπορεί με τη βοήθεια της mobile εφαρμογής να έχει πρόσβαση στα ενεργειακά δεδομένα του χώρου στον οποίο εργάζεται, να παρακολουθεί την εξέλιξη τους σε βάθος χρόνου, να ελέγχει για πιθανές σπατάλες ενέργειας και κακές συνθήκες εργασίας όπως για παράδειγμα ακραίες τιμές θερμοκρασίας ή υγρασίας ακόμα και να βλέπει προβλέψεις για ορισμένα μεγέθη, οι οποίες μπορούν να τον βοηθήσουν να προετοιμαστεί κατάλληλα, για παράδειγμα σε ακραία καιρικά φαινόμενα. Αξίζει να σημειωθεί πως στόχος είναι η εφαρμογή να είναι επεκτάσιμη, ώστε στο μέλλον να συμπληρωθεί με περαιτέρω υπηρεσίες, κάποιες από τις οποίες θα αναφερθούν στα επόμενα κεφάλαια της διπλωματικής εργασίας.

1.2 Φάσεις Υλοποίησης

Αρχικά, πριν ξεκινήσει η υλοποίηση, συζητήθηκαν σε θεωρητικό επίπεδο τα σενάρια χρήσης του συστήματος που αποτελεί το θέμα της διπλωματικής εργασίας και αποφασίστηκε ποιες είναι οι κύριες λειτουργίες που θα πρέπει να έχει. Στη συνέχεια, η υλοποίηση πραγματοποιήθηκε σε τρεις φάσεις, οι οποίες αφορούσαν το σύστημα διαχείρισης δεδομένων και την κυρίως mobile εφαρμογή, οι οποίες αναλύονται παρακάτω. Η εκπόνηση της διπλωματικής εργασίας πραγματοποιήθηκε μεταξύ Σεπτεμβρίου 2017 και Φεβρουαρίου 2018 και η υλοποίησή της παρουσιάζεται στο παρακάτω σχήμα.



Εικόνα 1.1- Φάσεις υλοποίησης

1η Φάση: Συγκέντρωση, επεξεργασία, αποθήκευση δεδομένων

Η πρώτη φάση της υλοποίησης όπως αναφέρθηκε παραπάνω αποτελείται από την ανάπτυξη αναλυτών (parsers), σε γλώσσα java, οι οποίοι είναι υπεύθυνοι για τη συλλογή των δεδομένων και την επεξεργασία τους ώστε να αποκτηθεί η ζητούμενη πληροφορία. Κατασκευάστηκαν δύο διαφορετικοί αναλυτές. Ο πρώτος είναι υπεύθυνος για τα δεδομένα που προέρχονται από τους αισθητήρες στο εργαστήριο κάθε πέντε λεπτά (σε φυσιολογική κατάσταση λειτουργίας). Ο δεύτερος είναι υπεύθυνος για τη συλλογή πληροφορίας που προέρχεται από εξωτερικό API πρόβλεψης καιρού. Οι δύο parsers αφού τη διαλέξουν, αποθηκεύουν την πληροφορία σε αρχεία σε μορφή (format) json.

Στο επόμενο στάδιο, δημιουργήθηκε βάση δεδομένων SQL, λαμβάνοντας υπόψιν ποιες πληροφορίες πρέπει να αποθηκευτούν ώστε να είναι πλήρης η λειτουργικότητα της αναπαράστασης των χρονοσειρών των ενεργειακών δεδομένων. Αρχικά, πραγματοποιήθηκε ο σχεδιασμός της βάσης σε ER - μοντέλο και στη ύστερα έγινε μετατροπή σε σχεσιακό μοντέλο, το οποίο αποτελεί την κατάλληλη μορφή ώστε να γίνει η εισαγωγή των στοιχείων στο υπολογιστικό σύστημα βάσης δεδομένων.

Στη συνέχεια, κατασκευάστηκε ένας publisher (επίσης με χρήση java) με ρόλο την ανάγνωση της αποθηκευμένης πληροφορίας στα αρχεία που δημιούργησαν οι αναλυτές και την αποθήκευση της στη βάση δεδομένων. Με τον τρόπο αυτό συγκρατείται η πληροφορία σε δομημένη, προσβάσιμη και αξιοποιήσιμη μορφή.

2η Φάση: Διεπαφή προγραμματισμού εφαρμογών για την ανάκτηση δεδομένων

Η δεύτερη φάση της υλοποίησης αποτελείται από τη δημιουργία ενός API που αλληλεπιδρά με την βάση και με τη mobile εφαρμογή . Η εφαρμογή κάνει μια αίτηση (request) στο API για τα δεδομένα που χρειάζεται και το API επικοινωνεί με τη βάση δεδομένων ώστε να αποκτήσει τα ζητούμενα δεδομένα και να τα εξάγει σε μορφή json. Στη συνέχεια η εφαρμογή μπορεί να διαβάσει την πληροφορία αυτή και να την παρουσιάσει. Η διακίνηση της πληροφορίας γίνεται με τρόπο ασφαλή τόσο για το σύστημα όσο και για την πληροφορία καθαυτή.

3η Φάση: Σχεδίαση και ανάπτυξη mobile εφαρμογής

Η τρίτη φάση της υλοποίησης περιλαμβάνει την σχεδίαση και την ανάπτυξη της mobile εφαρμογής σε σύστημα Android. Η εφαρμογή κατασκευάστηκε με στόχο να είναι άμεση και εύκολη στη χρήση. Αποτελείται από τρία κύρια κομμάτια: α) μία κεντρική οθόνη που παρουσιάζει γενικές πληροφορίες σχετικά με τα ενεργειακά δεδομένα κατά την είσοδο του χρήστη στην εφαρμογή, β) μια αναλυτική λίστα όπου παρουσιάζονται γραφικά όλες οι χρονοσειρές από ενεργειακά δεδομένα που αφορούν τον χρήστη στο χρονικό διάστημα και ανάλογα με την κατηγορία που αυτός επιλέγει, γ) αναλυτική παρουσίαση μιας επιλεγμένης χρονοσειράς με περισσότερες λειτουργικότητες όσον αφορά στη γραφική αναπαράσταση καθώς και αναλυτική παρουσίαση των δεδομένων σε μορφή πίνακα. Επίσης κατασκευάστηκαν και άλλες υπηρεσίες στην εφαρμογή που ολοκληρώνουν τη λειτουργικότητα της και περιγράφονται αναλυτικά στη συνέχεια.

1.3 Οργάνωση του Τόμου

Η εργασία χωρίζεται σε έξι κεφάλαια:

- Στο πρώτο κεφάλαιο παρουσιάζεται το πλαίσιο και το αντικείμενο της εργασίας.
- Στο δεύτερο κεφάλαιο αναλύονται κάποιες σύγχρονες εφαρμογές σχετικές με αυτή που αναπτύχθηκε στην παρούσα διπλωματική εργασία. Περιγράφονται οι λειτουργίες τους πολλές από τις οποίες αποτέλεσαν έμπνευση για τις λειτουργικότητες της δικής μας εφαρμογής και τέλος εξηγούνται οι λόγοι που κάνουν το σύστημά της εργασίας να ξεχωρίζει.
- Το τρίτο κεφάλαιο αφορά της τεχνολογίες και τα περιβάλλοντα ανάπτυξης που χρησιμοποιήθηκαν, γίνεται μια σύντομη ανασκόπηση και περιγραφή τους.
- Στο τέταρτο κεφάλαιο περιγράφονται τα αναλυτικά βήματα για τη δημιουργία του συστήματος διαχείρισης δεδομένων. Περιγράφονται οι σχεδιαστικές επιλογές και η υλοποίηση του υποσυστήματος ανάκτησης των δεδομένων, τα βάση δεδομένων, καθώς και του υποσυστήματος υπεύθυνου για την ανάκτηση των πληροφοριών της βάσης.
- Στο πέμπτο κεφάλαιο, περιγράφονται αναλυτικά η σχεδίαση και η υλοποίηση της κύριας mobile εφαρμογής. Περιγράφεται το απαραίτητο λογισμικό που χρησιμοποιήθηκε για να πραγματοποιηθεί η ανάπτυξή της, καθώς και οι δομές δεδομένων που αναπτύχθηκαν. Τέλος, περιγράφονται και όλες οι βασικές σχεδιαστικές επιλογές, ώστε να γίνει η εφαρμογή φιλική για τον χρήστη.
- Στο έκτο κεφάλαιο, παρουσιάζονται παραδείγματα της εφαρμογής σε χρήση και οι λειτουργικότητες αναλυτικά. Τονίζονται οι επιλογές που έγιναν, τα εμπόδια που συναντήθηκαν και με ποιο τρόπο ξεπεράστηκαν καθ' όλη τη διάρκεια ανάπτυξης της εφαρμογής.
- Το έβδομο κεφάλαιο αφορά στα αποτελέσματα και τα βασικά συμπεράσματα που προέκυψαν για το σύστημα παρουσίασης ενεργειακών δεδομένων που κατασκευάστηκε και πραγματοποιούνται προτάσεις για την περαιτέρω εξέλιξη του. Πιο συγκεκριμένα, σχολιάζεται η περίπτωση επέκτασης της εφαρμογής καθώς και πρόσθετες λειτουργικότητες που θα μπορούσαν να εισαχθούν σε μελλοντικό χρόνο, ώστε να θεωρηθεί πιο ολοκληρωμένη και ακόμα πιο αποδοτική σε ότι αφορά στους στόχους της.

Κεφάλαιο 2ο - Σύγχρονες Σχετικές Εφαρμογές

2.1 Σύγχρονες Εφαρμογές

Τα τελευταία χρόνια, οι ανάγκες για παρακολούθηση της κατανάλωσης ενέργειας σε οικιακούς και επαγγελματικούς χώρους έχουν αυξηθεί σε μεγάλο βαθμό με σκοπό τόσο την εξοικονόμηση ενέργειας όσο και την ρύθμιση της ποιότητας των συνθηκών μέσα στα κτίρια. Ως εκ τούτου, παρατηρείται συνεχώς αυξανόμενη χρήση λογισμικού που αφορά στην διαχείριση ενέργειας και άλλων σχετικών μεγεθών. Το γεγονός αυτό σε συνδυασμό με την ραγδαία αύξηση της χρήσης φορητών συσκευών τα τελευταία χρόνια, έχει ως αποτέλεσμα την ανάπτυξη πολλών mobile εφαρμογών που έχουν τον παραπάνω ρόλο. Αυτές περιγράφονται παρακάτω, με μια συνοπτική παρουσίαση των λειτουργιών.

2.1.1 Smartwatt

Η εφαρμογή Smartwatt [41] της εταιρίας παροχής ηλεκτρικού ρεύματος watt+volt παρέχει στο χρήστη τη δυνατότητα να μετατρέψει τον οικιακό ή επαγγελματικό του χώρο σε "έξυπνο", προσθέτοντας "έξυπνες" συσκευές. Οι "έξυπνες" συσκευές επικοινωνούν ασύρματα μέσω του πρωτοκόλλου Zigbee με ένα κεντρικό Gateway, την καρδιά του έξυπνου συστήματος που συνδέεται απευθείας με το router της διαδικτυακής σύνδεσης, με μία πολύ απλή διαδικασία. Μέσω της εφαρμογής Smartwatt, ο χρήστης μπορεί να δώσει εντολές στο Gateway και αυτό με τη σειρά του τις μεταφέρει στις "έξυπνες" συσκευές. Οι συσκευές αυτές είναι ευέλικτες και εύκολες στην εγκατάσταση και υπάρχει δυνατότητα επανατοποθέτησης όπου παρουσιαστεί ανάγκη χωρίς καλώδια και επεμβάσεις στην ηλεκτρολογική εγκατάσταση.

Με την υπηρεσία Smartwatt και την "Έξυπνη" Λάμπα, ρυθμίζονται εύκολα κάθε στιγμή τα φώτα ενός χώρου. Εγκαθιστώντας τους Αισθητήρες Κίνησης και Αισθητήρες Πόρτας/Παραθύρου, ο χρήστης έχει τη δυνατότητα να λαμβάνει ειδοποιήσεις για οποιαδήποτε κίνηση ανιχνευθεί μέσα στο χώρο του. Έτσι, διευκολύνεται η καθημερινότητά του, ενώ παράλληλα διατηρείται η ασφάλεια του χώρου. Με τους Αισθητήρες Διαρροής και Υγρασίας ο χρήστης μπορεί ανά πάσα στιγμή να λαμβάνει ειδοποιήσεις για αύξηση της υγρασίας, ενδεχόμενη πλημμύρα ή υπερχείλιση νερού στο χώρο. Εγκαθιστώντας το "Έξυπνο" Ρελέ στον ηλεκτρολογικό πίνακα αποκτάται η δυνατότητα μέτρησης της ηλεκτρικής κατανάλωσης του χώρου επιτυγχάνοντας μέσω της παρακολούθησης στην μείωση της και τελικώς στην εξοικονόμηση χρημάτων. Με την βοήθεια της "Έξυπνης" Πρίζας γίνεται δυνατός ο έλεγχος οποιαδήποτε ηλεκτρική συσκευή από το κινητό ή άλλη φορητή συσκευή.

Το γεγονός ότι απαιτείται ο συγκεκριμένος εξοπλισμός της εταιρίας προκειμένου να λειτουργήσει η εφαρμογή, αποτελεί σοβαρό περιορισμό στο πλαίσιο της συμβατότητας και επεκτασιμότητας της εφαρμογής. Είναι προφανές ότι όσο μεγαλύτερο είναι το πλήθος των δεδομένων, τόσο ποιο λειτουργική είναι η εφαρμογή. Πιο συγκεκριμένα, δεν μπορεί να γίνει δυνατή η επικοινωνία με αισθητήρες διαφορετικού τύπου ή άλλες υπηρεσίες όπως: web servises, APIs, και άλλα. Επίσης η χρήση των μικρο-υπολογιστικών συστημάτων της εταιρίας , δεν επιτρέπει την ανάπτυξη κώδικα από απλούς χρήστες(open source) με σκοπό την επέκταση της εφαρμογής.

2.1.2 Energy Audit

Η εφαρμογή Energy Audit [42] της εταιρίας Tessera Multimedia A. E. , μέσω εισαγωγής προεπιλεγμένων στοιχείων, προτείνει λύσεις ενεργειακής βελτιστοποίησης της απόδοσης της κατοικίας, υπολογίζει παράλληλα την εξοικονόμηση ενέργειας ετησίως και τις ετήσιες εκπομπές CO2 (kg/m2) και κατατάσσει εκ νέου το ακίνητο σε ενεργειακή κατηγορία, μετά από τις προτεινόμενες παρεμβάσεις. Επιπλέον παρέχει τη δυνατότητα στον χρήστη να αναζητήσει πανελλαδικά τον μηχανικό της αρεσκείας του για πιθανές εργασίες (ενεργειακό έλεγχο και έκδοση πιστοποιητικού ενεργειακής απόδοσης ΠΕΑ, οικολογική μόνωση) μέσα από μία ευρεία βάση δεδομένων, η οποία διαρκώς ανανεώνεται.

Το λογισμικό απευθύνεται στο ευρύ κοινό όπως ιδιοκτήτες ακινήτων, υποψήφιοι ενοικιαστές-αγοραστές, μεσίτες, που επιθυμούν να διενεργήσουν μια προκαταρκτική αξιολόγηση της ενεργειακής συμπεριφοράς του κτιρίου (του οποίου είναι κάτοχοι ή απλώς τους ενδιαφέρει) και να διερευνήσουν ταυτόχρονα τις δυνατότητες εξοικονόμησης ενέργειας, χρήσης εναλλακτικών μορφών ενέργειας και σε κάθε περίπτωση τη μείωση του οικονομικού κόστους, που συνεπάγεται η θέρμανση και ο δροσισμός των κτιρίων. Επίσης είναι χρήσιμο για επαγγελματίες με σχετικό αντικείμενο (ενεργειακοί επιθεωρητές, μηχανικοί, κλπ.). Οι επαγγελματίες αυτοί μπορούν να χρησιμοποιήσουν την εφαρμογή ως ένα εύχρηστο βοηθητικό εργαλείο για τη διευκόλυνση της επί τόπου καταγραφής των απαραίτητων δεδομένων για τη διενέργεια της ενεργειακής επιθεώρησης ενός κτιρίου. Ακόμα αποτελεί εργαλείο για εταιρείες που προσφέρουν τεχνικές λύσεις ενεργειακής αναβάθμισης κτιρίων (κατασκευαστές κουφωμάτων αλουμινίου, εμπόριο συστημάτων θέρμανσης, κλπ.). Οι εταιρείες αυτές μπορούν να χρησιμοποιήσουν την εφαρμογή ως ένα φθηνό και αποτελεσματικό μέσο προβολής των προϊόντων τους και των υπηρεσιών τους. Τεχνολογικοί Φορείς (βιομηχανικές μονάδες, τεχνικές εταιρείες, ερευνητικά ινστιτούτα, πανεπιστημιακά εργαστήρια, κλπ.) ,με τη χρήση της προτεινόμενης εφαρμογής, μπορούν πολύ εύκολα να συγκεντρώσουν έναν σημαντικό πλούτο δεδομένων, πάντα στη βάση του σεβασμού των προσωπικών δεδομένων των χρηστών, που θα επιτρέψουν την κατανόηση και αξιολόγηση των επιλογών και της ενεργειακής συμπεριφοράς των χρηστών, ειδικά στις σημερινές συνθήκες, που οι μεταβολές είναι ραγδαίες (αλλαγή καυσίμου, χρήση εναλλακτικών συστημάτων θέρμανσης).Έτσι τεχνολογικοί φορείς όπως βιομηχανικές μονάδες, τεχνικές εταιρείες, ερευνητικά ινστιτούτα, πανεπιστημιακά εργαστήρια θα μπορούν να αξιοποιήσουν τα στατιστικά δεδομένα από τη χρήση της εφαρμογής για την κατανόηση και αξιολόγηση των επιλογών και της ενεργειακής συμπεριφοράς των χρηστών.

Το μειονέκτημα της εν λόγω εφαρμογής είναι ότι δε βασίζεται σε πραγματικά δεδομένα και το αποτέλεσμα της ενεργειακής μελέτης αποτελεί προσομοίωση.

2.1.3 Sense Home Energy Monitor και Neurio Home Technology

Οι εφαρμογές ελέγχου ροής ρεύματος των εταιριών Sense Lab Inc. [44] και Neurio Technology [43], έχουν ως βασικό εργαλείο ένα σύστημα το οποίο εγκαθίσταται στον

κεντρικό ηλεκτρικό πίνακα του χώρου ενδιαφέροντος. Μέσω της ανάλυσης της ροής και κατανάλωσης ρεύματος, η συσκευή αυτή αναγνωρίζει κάθε μέσο κατανάλωσης ρεύματος και χτίζει βάση δεδομένων μέσω της καθημερινής χρήσης. Αυτή η βάση χρησιμοποιείται ώστε να γίνει αποτύπωση ,στις εκάστοτε εφαρμογές, των δεδομένων σε πραγματικό χρόνο ώστε να υπάρχει έλεγχος κάθε στιγμή. Απαιτείται σύνδεση μέσω wifi στο κεντρικό router όπου και τροφοδοτείται η εφαρμογή.

Οι δυνατότητες του application είναι καταγραφής και ενημέρωσης και όχι παρεμβατικές. Ο χρήστης έχει μέσω της εφαρμογής πρόσβαση σε πραγματικό χρόνο ,αλλά και σε ιστορικό, της κατανάλωσης ρεύματος συνολικά αλλά και συγκεκριμένα μέσω της δυνατότητας διαχωρισμού των συσκευών κατανάλωσης από το υπολογιστικό σύστημα που είναι τοποθετημένο στον ηλεκτρικό πίνακα. Επίσης, ο χρήστης μπορεί να ενεργοποιήσει ειδοποιήσεις για ασυνήθιστη λειτουργία συγκεκριμένης συσκευής ή για παρακολούθηση λειτουργίας. Είναι δυνατή η ενημέρωση του χρήστη οποιαδήποτε στιγμή εφόσον υπάρχει σύνδεση στο ίντερνετ, για παράδειγμα, ο χρήστης μπορεί μέσω της εφαρμογής να ελέγξει αν λειτουργεί το θερμοσίφωνο ενώ βρίσκεται εκτός της οικίας του. Η δημιουργία βάσης δεδομένων και η ανάλυση της, δίνει μια προγνωστική βοήθεια στο χρήστη σχετικά με το μέσο όρο καθημερινής, εβδομαδιαίας ή μηνιαίας κατανάλωσης.

Η εφαρμογή προσφέρει πληροφορίες που αφορούν την κατανάλωση ενέργειας και δεν μελετάει άλλα μεγέθη όπως η θερμοκρασία, η υγρασία, οι ρύποι και άλλα.

2.2 Συμβολή της Διπλωματικής Εργασίας

Οι παραπάνω εφαρμογές αποτελούν πλήρη εμπορικά προϊόντα τα οποία υπό προϋποθέσεις μπορούν να συντελέσουν σημαντικά στην διαχείριση ενέργειας. Παρ' όλα αυτά, αποτελεί πολύ σημαντικό μειονέκτημα στην παροχή δεδομένων η εξάρτηση της κάθε εφαρμογής από τον συγκεκριμένο εξοπλισμό της κάθε εταιρίας, καθώς επίσης και ότι αποτελούν έργα κλειστού κώδικα.

Στην παρούσα διπλωματική εργασία, αναπτύσσεται ένα σύστημα, το οποίο δεν βασίζεται εξ ολοκλήρου σε συγκεκριμένα, εξειδικευμένα μικρο-υπολογιστικά συστήματα που εκτελούν περίπλοκες λειτουργίες. Αντίθετα, δημιουργήθηκαν απλά ανεξάρτητα εργαλεία με όσο το δυνατόν πιο γενικευμένο τρόπο, τα οποία συνεργάζονται μεταξύ τους προκειμένου να συλλέξουν και να παρουσιάσουν ενεργειακά δεδομένα. Τα εργαλεία αυτά είναι ανοιχτά σε όλους (και ως εκ τούτου επεκτάσιμα) και συλλέγουν πληροφορία όχι μόνο από τοπικές συσκευές αλλά και από εξωτερικές υπηρεσίες. Για παράδειγμα, στο αναπτυγμένο σύστημα χρησιμοποιήθηκαν δεδομένα από API πρόβλεψης καιρού τα οποία προσδίδουν στην εφαρμογή μια επιπλέον δυνατότητα, την πρόβλεψη. Συνεπώς, οι δυνατότητες της εφαρμογής δεν περιορίζονται από το hardware.

Η απλότητα της συλλογής, επεξεργασίας και αποθήκευσης των δεδομένων σε μία ενιαία μορφή επιτρέπουν την συνεχή βελτίωση των δυνατοτήτων και επιλογών της. Επίσης, καθίσταται δυνατή η πρόσβαση στα δεδομένα και από άλλα συστήματα καθώς αυτή είναι ομοιόμορφη. Οι αισθητήρες που χρησιμοποιήθηκαν στο παρόν σύστημα εξάγουν δεδομένα σε μια πολύ διαδεδομένη και απλή μορφή (.csv) και δεν

απευθύνονται σε συγκεκριμένο τύπο εφαρμογής ενώ είναι δυνατόν να αντικατασταθούν με σχετικά μειωμένο κόστος, ανά πάσα στιγμή με οποιοδήποτε άλλο αντίστοιχο εξοπλισμό. Οι parsers παράγουν και οι δύο αρχεία σε JSON format, που είναι επίσης διαδεδομένο και εκμεταλλεύσιμο από άλλες υπηρεσίες.

Επίσης, το γεγονός ότι το όλο εγχείρημα πρόκειται για ένα open source έργο με ανοιχτό κώδικα προς όλους το καθιστά επεκτάσιμο και του αποδίδει σημαντικές προοπτικές εξέλιξης, χωρίς να έχει κερδοσκοπικό σκοπό, αλλά στοχεύοντας σε καθολική χρήση για την εξοικονόμηση ενέργειας.

Πρέπει ακόμα να τονιστεί, πως η παροχή πληροφοριών για τα ενεργειακά μεγέθη δεν αποτελεί προσομοίωση, αλλά βασίζεται σε πραγματικές μετρήσεις. Συνεπώς, η πιθανή μελλοντική χρήση προτάσεων από την εφαρμογή για εξοικονόμηση ενέργειας δεν θα βασίζεται σε κάποιο γενικό θεωρητικό πλαίσιο αλλά σε στοχευμένες μετρήσεις.

Τέλος, οι παραπάνω εφαρμογές αποτέλεσαν πηγή έμπνευσης για την δημιουργία του συστήματος που περιγράφεται στην εργασία αυτή, αλλά σε ορισμένες περιπτώσεις παρουσίαζαν είτε έλλειψη λεπτομερούς ιστορικού είτε εστίαζαν αποκλειστικά στην κατανάλωση ενέργειας και όχι στα υπόλοιπα ενεργειακά μεγέθη τα οποία επηρεάζουν τις συνθήκες άνεσης των κτιρίων. Επίσης, κάποιες από τις εμπορικές εφαρμογές δεν είχαν τη δυνατότητα παρουσίασης συγκεκριμένων ενεργειακών μεγεθών για χώρους του κτιρίου. Η εφαρμογή που αναπτύχθηκε στην παρούσα διπλωματική, προσφέρει εξατομικευμένα δεδομένα για κάθε χρήστη με βάση τον χώρο εργασίας του.

Στα παρακάτω κεφάλαια, αναλύεται η υλοποίηση όλων των τμημάτων του συστήματος σε βήματα δίνοντας έμφαση στα σημεία που έγιναν προγραμματιστικές επιλογές προκειμένου να μπορούν να αποκτηθούν δεδομένα από πολλές διαφορετικές πηγές πληροφορίας.

Κεφάλαιο 3ο – Εργαλεία Ανάπτυξης Λογισμικού

Σε αυτό το κεφάλαιο παρουσιάζονται όλα τα εργαλεία ανάπτυξης λογισμικού και τα περιβάλλοντα που χρησιμοποιήθηκαν στη διαδικασία ανάπτυξης και ολοκλήρωσης όλου του συστήματος διαχείρισης δεδομένων και της mobile εφαρμογής.

3.1 Η γλώσσα προγραμματισμού Java

Ιστορικά στοιχεία για τη γλώσσα Java

Στις αρχές του 1991, η Sun αναζητούσε το κατάλληλο εργαλείο για να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού σε μικροσυσκευές (έξυπνες οικιακές συσκευές έως πολύπλοκα συστήματα παραγωγής γραφικών). Τα εργαλεία της εποχής ήταν γλώσσες όπως η C++ και η C. Μετά από διάφορους πειραματισμούς προέκυψε το συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες τους. Ο "πατέρας" της Java, James Gosling, που εργαζόταν εκείνη την εποχή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς κάποιες πειραματικές γλώσσες ως πρότυπα για το νέο εργαλείο που αναζητούσαν στη Sun. Τελικά, μετά από λίγο καιρό κατέληξαν σε μια πρόταση για το επιτελείο της εταιρίας, η οποία ήταν η γλωσσά Oak. Το όνομά της το πήρε από το ομώνυμο δένδρο (βελανιδιά) το οποίο ο Gosling είχε έξω από το γραφείο του και έβλεπε κάθε μέρα.

Η Oak ήταν μία γλώσσα που διατηρούσε μεγάλη συγγένεια με την C++. Παρόλα αυτά είχε πολύ πιο έντονο αντικειμενοστραφή (object oriented) χαρακτήρα σε σχέση με την C++ και χαρακτηριζόταν για την απλότητα της. Σύντομα, οι υπεύθυνοι ανάπτυξης της νέας γλώσσας ανακάλυψαν ότι το όνομα Oak ήταν ήδη κατοχυρωμένο οπότε κατά την διάρκεια μιας εκ των πολλών συναντήσεων σε κάποιο τοπικό καφέ αποφάσισαν να μετονομάσουν το νέο τους δημιούργημα σε Java που εκτός των άλλων ήταν το όνομα της αγαπημένης ποικιλίας καφέ για τους δημιουργούς της. Η επίσημη εμφάνιση της Java, αλλά και του HotJava (πλοηγός με υποστήριξη Java) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η Sun την ανακοίνωσε στο συνέδριο Sun World 1995. Ο πρώτος μεταγλωττιστής (compiler) της ήταν γραμμένος στη γλώσσα C από τον James Gosling. Το 1994, ο A.Van.Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε Java, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν τη Java για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον μία από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής. Στις 13 Νοεμβρίου του 2006, η Java έγινε πλέον μια γλώσσα ανοιχτού κώδικα όσον αφορά το μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK, Java Development Kit).

Στις 27 Απριλίου 2010, η εταιρία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών (πνευματικά δικαιώματα/ πατέντες) που η δεύτερη είχε στην κατοχή της ή δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια. [17,18]

Χαρακτηριστικά γλώσσας Java

Σύμφωνα με την περιγραφή που έχει δοθεί από την εταιρεία Sun για τη γλώσσα προγραμματισμού Java, η τελευταία είναι μια απλή, αντικειμενοστραφής, κατανεμημένη, υψηλής απόδοσης, συμπαγής, ασφαλής, ανεξάρτητης αρχιτεκτονικής, μεταφέρσιμη, υψηλής απόδοσης και ιδιαίτερα δυναμική.

Η Java αποτελεί μια απλή γλώσσα καθώς παρουσιάζει αρκετές ομοιότητες με τις γλώσσες C και C++ οι οποίες είναι ευρέως διαδομένες, διευκολύνοντας έτσι την εκμάθησή της, ενώ ταυτόχρονα περιέχει λίγες προγραμματιστικές δομές και ιδιαίτερα καλά ορισμένη σημασιολογία. Δεδομένου ότι είναι αντικειμενοστραφής γλώσσα εστιάζει στον ορισμό αντικειμένων και των αντίστοιχων λειτουργιών τους. Στη java η έννοια της κλάσης, η οποία περιγράφει μια συλλογή δεδομένων και τις λειτουργίες που αυτά επιδέχονται, διαθέτει ιδιαίτερη σημασία. Κάθε μια από τις κλάσεις προέρχεται από μια άλλη μέσω της κληρονομικότητας μεταξύ των κλάσεων (inheritance), ορίζοντας κατ' αυτό τον τρόπο μια ιεραρχία κλάσεων στη κορυφή της οποίας υφίσταται η αρχική κύρια κλάση που είναι το αντικείμενο (object). Τα αντικείμενα μιας κλάσης χρησιμοποιούνται σε ένα πρόγραμμα Java και δημιουργούνται κατά τη διάρκεια εκτέλεσης του προγράμματος.

Ακόμα, η Java χαρακτηρίζεται ως κατανεμημένη γλώσσα καθώς επιτρέπει την επικοινωνία με αντικείμενα τα οποία βρίσκονται σε απομακρυσμένες θέσεις στο δίκτυο, ενώ ταυτόχρονα επιτρέπει την επικοινωνία με άλλες εφαρμογές μέσω διαδικτυακών συνδέσεων. Εκτός από κατανεμημένη, θεωρείται και ερμηνευόμενη. Αυτό συμβαίνει καθώς ο μεταγλωττιστής της Java δεν παράγει έναν τελικό κώδικα για ένα συγκεκριμένο υπολογιστή. Αντίθετα, παράγει κάτι ενδιάμεσο σε μορφή bytes το οποίο ονομάζεται bytecode. Αυτός ο κώδικας περνά στη συνέχεια από τον διερμηνέα της Java. Αυτή η διαδικασία γίνεται προκειμένου ο κώδικας να έχει τη δυνατότητα να εκτελεστεί σε πολλά διαφορετικά περιβάλλοντα υπολογιστών, εφόσον βέβαια ο εν λόγω διερμηνέας είναι διαθέσιμος σε αυτά.

Η Java θεωρείται ως μια συμπαγής γλώσσα προγραμματισμού. Αυτό συμβαίνει καθώς διαθέτει ένα ισχυρό σύστημα τύπων, το οποίο επιτρέπει εκτενείς ελέγχους κατά τη διάρκεια μετάφρασης των προγραμμάτων. Με τον τρόπο αυτό, η Java συμβάλλει στην ανάπτυξη αξιόπιστου και συμπαγούς λογισμικού. Εκτός των άλλων η Java χαρακτηρίζεται ως ασφαλής γλώσσα. Αυτό συμβαίνει καθώς τα προγράμματα που έχουν αναπτυχθεί σε αυτή, μπορούν να χρησιμοποιηθούν από μεγάλο εύρος χρηστών καθένας από τους οποίους διαθέτει διαφορετικά δικαιώματα πρόσβασης, με αποτέλεσμα να μην είναι δυνατό να υπάρξουν ανεπιθύμητες παρενέργειες στο σύστημα. Το πλεονέκτημα αυτό, προσδίδεται από τη χρήση ενός ενδιάμεσου κώδικα (bytecode verifier) ο οποίος εντοπίζει τυχόν περίεργες ενέργειες οι οποίες είναι δυνατό να επηρεάσουν αρνητικά το περιβάλλον εργασίας του χρήστη.

Συνεχίζοντας, η Java αποτελεί μια γλώσσα ανεξάρτητης αρχιτεκτονικής καθώς διαθέτει έναν ενδιάμεσο κώδικα, ο οποίος δεν αναφέρεται σε ένα συγκεκριμένο τύπο υπολογιστή αλλά αντίθετα μεταφράζεται κατάλληλα με τη βοήθεια του διερμηνέα. Επίσης, εξαιτίας του παραπάνω χαρακτηριστικού της java, δηλαδή της ανεξαρτησίας της αρχιτεκτονικής, τα προγράμματα σε java καθίστανται μεταφέρσιμα καθώς προσαρμόζονται σε διάφορους τύπους υπολογιστών διατηρώντας αμετάβλητους τους αρχικούς τύπους δεδομένων. Αυτό σημαίνει ότι ανεξάρτητα από την αρχιτεκτονική που ακολουθούν όμοια προγράμματα, θα δώσουν ίδια αποτελέσματα για ίδιες αρχικές τιμές των παραμέτρων τους.

Η java αποτελεί μία γλώσσα με αρκετά υψηλή απόδοση. Παρόλα αυτά, δεν είναι δυνατό να φτάσει την απόδοση των γλωσσών προγραμματισμού C και C++ καθώς οι τελευταίες υλοποιούνται απευθείας σε μεταγλωττιστές. Ωστόσο, έχουν κατασκευαστεί μεταγλωττιστές της τελευταίας στιγμής (just-in-time compilers) οι οποίοι εμπεριέχουν διερμηνείς και βελτιώνουν ιδιαίτερα την απόδοση εκτέλεσης των προγραμμάτων Java. Ακόμη, υποστηρίζει πολλαπλά νήματα εκτέλεσης. Αυτό σημαίνει ότι δίνει τη δυνατότητα ταυτόχρονης εκτέλεσης πολλών διεργασιών, γεγονός που είναι ιδιαίτερα αποδοτικό .Έτσι, λοιπόν, φαίνεται ότι η Java αποτελεί μια δυναμική γλώσσα, η οποία έχει σχεδιαστεί προκειμένου να προσαρμόζεται σε ένα δυναμικά εξελισσόμενο περιβάλλον. Αυτό σημαίνει ότι οι κλάσεις που πρέπει να εκτελεστούν στο πρόγραμμα, είναι δυνατό να βρίσκονται σε κάποιο άλλο μέρος του δικτύου και να μεταφερθούν δυναμικά προκειμένου να εκτελεστούν τοπικά. Είναι φανερό, επομένως, ότι οι κλάσεις δεν είναι απαραίτητο να ενσωματωθούν στο πρόγραμμα κατά τη διάρκεια της μετάφρασής τους. Τέλος, η Java υποστηρίζει τη μεταφορά του εκτελέσιμου περιεχομένου σε εφαρμογές πολυμέσων. Αυτό σημαίνει, ότι υπάρχει πλέον η δυνατότητα εκτέλεσης των προγραμμάτων στο περιβάλλον του χρήστη με αποτέλεσμα να αυξάνονται οι δυνατότητες των χρηστών, γεγονός που ενισχύει ακόμα περισσότερο τη δυναμικότητα της γλώσσας.[10,20]

3.2 Το λειτουργικό σύστημα Android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά, αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα tablets, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί και σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ και σε άλλες ηλεκτρονικές συσκευές.

Ιστορικά στοιχεία του λειτουργικού συστήματος Android

Τον Αύγουστο του 2005, η Google εξαγόρασε την Android Inc.(μία εταιρία που ιδρύθηκε στη Καλιφόρνια, τον Οκτώβριο του 2003, για την ανάπτυξη «έξυπνων κινητών συσκευών οι οποίες θα γνωρίζουν κατά τον καλύτερο δυνατό τρόπο την τοποθεσία και τις προτιμήσεις του ιδιοκτήτη τους») Δεν υπάρχουν και πολλά πράγματα που ήταν γνωστά για την Android Inc. εκείνη την εποχή, αλλά πολλοί υπέθεσαν ότι η Google, με την κίνηση αυτή, σχεδιάζει να εισέλθει στην αγορά της κινητής τηλεφωνίας. Η Google ανέπτυξε μια πλατφόρμα κινητών συσκευών που τροφοδοτείται από τον πυρήνα του Linux. Ακόμα προώθησε στην αγορά μια πλατφόρμα για κατασκευαστές και συνεχίζει τη πορεία της με την υπόσχεση να παρέχει ένα ευέλικτο σύστημα, με δυνατότητες αναβάθμισης.

Στις 5 Νοεμβρίου 2007, ιδρύθηκε η Open Handset Alliance (OHA), μια κοινοπραξία 48 τηλεπικοινωνιακών εταιρειών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες στοχεύουν στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Το Android παρουσιάστηκε ως το πρώτο προϊόν της,

μια πλατφόρμα για κινητές συσκευές η οποία κατασκευάστηκε στηριζόμενη στο πυρήνα της Linux version 2.6.25. Σημειώνεται ότι εντός των εταιριών της κοινοπραξίας συμπεριλαμβάνονταν η Google, κατασκευαστές συσκευών όπως HTC, Sony και Samsung, καθώς και άλλοι φορείς , όπως η Sprint Nextel και T-Mobile, και κατασκευαστές chipset, όπως η Qualcomm και Texas Instruments. Η Open Handset Alliance συνεχίζει τη λειτουργία της μέχρι σήμερα και αποτελεί, πλέον, μια κοινοπραξία 84 επιχειρήσεων οι οποίες προωθούν την ανάπτυξη ανοιχτών προτύπων για κινητές συσκευές.

Η ιστορία εκδόσεων του λειτουργικού συστήματος των κινητών ξεκίνησε με την κυκλοφορία του Android beta το Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση ήταν το Android 1.0 που κυκλοφόρησε το Σεπτέμβριο του 2008. Το Android είναι υπό συνεχή ανάπτυξη από την Google και την Open Handset Alliance (OHA), και έχουν γίνει μια σειρά από ενημερώσεις στην λειτουργία του συστήματος από την αρχική κυκλοφορία του.

Από τον Απρίλιο του 2009, οι εκδόσεις του Android έχουν θέμα από την ζαχαροπλαστική στην κωδική ονομασία τους, και κυκλοφόρησαν σε αλφαβητική σειρά, εξαιρουμένων των εκδόσεων 1.0 και 1.1, που δεν τέθηκαν υπό συγκεκριμένα κωδικά ονόματα. Στη συνέχεια, παρατίθεται πίνακας, ο οποίος παρουσιάζει τις διάφορες εκδόσεις του λογισμικού που έχουν κυκλοφορήσει, με τις αντίστοιχες ονομασίες τους και τις ημερομηνίες κυκλοφορίας τους στην αγορά. [27]

Πίνακας 3.1 - Εκδόσεις Λειτουργικού Συστήματος Android

Κωδικό όνομα	**Νούμερο έκδοσης**	**Ημερομηνία αρχικής κυκλοφορίας**	**Επίπεδο API**
N/A	1.0	23 Σεπτεμβρίου 2008	1
	1.1	9 Φεβρουάριου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5–7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9–10
Honeycomb	3.0 – 3.2.6	22 Φεβρουάριου 2011	11–13
IceCream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14–15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16–18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19–20
Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21–22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	7.0 - 7.1.1	22 Αυγούστου 2016	24
Oreo	8.0	21 Αυγούστου 2017	26

Χαρακτηριστικά του λειτουργικού συστήματος Android

Το περιβάλλον ανάδρασης του Android βασίζεται στην άμεση χειραγώγηση, καθώς χρησιμοποιεί ως εντολές εισόδου στο λειτουργικό σύστημα, εντολές αφής, οι οποίες αντιστοιχούν σε πραγματική αλληλεπίδραση με την οθόνη του κινητού τηλεφώνου. Αυτές οι δράσεις είναι κινήσεις όπως το σύρσιμο, πάτημα (στιγμιαίο ή διαρκείας), τσίμπημα (κανονικό και αντίστροφο) προκειμένου να μετακινηθούν και να τοποθετηθούν επί της οθόνης διάφορα αντικείμενα. Κάθε τέτοια κίνηση εντοπίζεται άμεσα από το σύστημα δίνοντας στον χρήστη την εντύπωση ενός πλήρως διαδραστικού ρευστού περιβάλλοντος με πολλές παράλληλες δυνατότητες.

Το hardware της συσκευής, διαθέτει επίσης σε ορισμένες περιπτώσεις επιταχυνσιόμετρα, γυροσκόπια και αισθητήρες εγγύτητας, τα οποία χρησιμοποιούνται από ορισμένες εφαρμογές προκειμένου να ανταποκριθούν στις πρόσθετες ενέργειες του χρήστη, όπως είναι για παράδειγμα η ρύθμιση της οθόνης κατά τον κατακόρυφο και τον οριζόντιο προσανατολισμό, ανάλογα με τον τρόπο που ο χρήστης κρατάει τη συσκευή, ή προσομοιώνοντας τον έλεγχο του τιμονιού σε άλλες εφαρμογές (όπως παιχνίδια κτλ.)

Η αρχική οθόνη των Android, έχει την ίδια λογική με την επιφάνεια εργασίας των υπολογιστών. Πρόκειται για ένα κόμβο ο οποίος περιλαμβάνει την πλοήγηση στο περιβάλλον του Android. Στο homescreen, όπως ονομάζεται, βρίσκονται εικονίδια που με το πάτημά τους εκκινούν τις εφαρμογές που επιθυμεί ο χρήστης να του είναι εύκολα προσβάσιμες. Επίσης, υπάρχουν widgets, τα οποία αποτελούν γραφικά βοηθήματα για γρήγορη ενημέρωση σε θέματα όπως η ώρα, ο καιρός και άλλα. Η αρχική οθόνη μπορεί να αποτελείται από περισσότερες της μίας «σελίδες», οι οποίες είναι δυνατό να εμφανιστούν στον χρήστη με ένα απλό «σύρσιμο» στην οθόνη αφής είτε προς τα μπροστά είτε προς τα πίσω. Ασφαλώς αξίζει να σημειωθεί ότι οι «σελίδες» οι οποίες είναι δυνατό να εμφανιστούν στην αρχική οθόνη του συστήματος, προσαρμόζονται στις προτιμήσεις του χρήστη προκειμένου να ικανοποιηθούν με το βέλτιστο τρόπο οι ανάγκες του από το σύστημα. Επίσης, στο πάνω μέρος της οθόνης, εμφανίζεται μια γραμμή κατάστασης, η οποία περιέχει πληροφορίες σχετικά με τη συσκευή και τη συνδεσιμότητα της. Αυτή η γραμμή κατάστασης μπορεί να "τραβηχτεί" προς τα κάτω για να αποκαλύψει ενημερώσεις (notifications) που αφορούν τον χρήστη από τις εφαρμογές που χρησιμοποιεί, καθώς και έναν μικρό πίνακα ελέγχου για αλλαγές στην ένταση του ήχου, στη φωτεινότητα της οθόνης και σε άλλα στοιχεία της συσκευής.

Είναι προφανές, ότι οι εφαρμογές (applications) αποτελούν ένα σημαντικό τμήμα των Android συσκευών. Οι εφαρμογές αυτές, επεκτείνουν τη λειτουργικότητα των συσκευών και έχουν γραφτεί κατά κύριο λόγο σε γλώσσα προγραμματισμού Java, χρησιμοποιώντας το kit ανάπτυξης λογισμικού Android (SDK). Το SDK αποτελείται από μια πλήρη σειρά εργαλείων ανάπτυξης, εντός των οποίων περιλαμβάνεται ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες λογισμικού, μια συσκευή η οποία εξομοιώνει το περιβάλλον του Android και που βασίζεται στο QEMU, τεκμηρίωση, δείγματα κώδικα και βοηθητικό υλικό, που περιγράφει τον τρόπο λειτουργίας του αναφερόμενου λογισμικού ανάπτυξης. Αρχικά, η Google υποστήριξε το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Eclipse με τη χρήση των Android Development Tools (ADT) plugin ενώ στη συνέχεια, τον Δεκέμβριο 2014 η Google έφερε στη κυκλοφορία το Android Studio, το οποίο είναι βασισμένο στο IntelliJ IDEA, ως κύριο IDE για την ανάπτυξη εφαρμογών Android.

Το Android έχει μια μεγάλη ποικιλία εφαρμογών, οι οποίες μπορούν να αποκτηθούν από τους χρήστες, με ένα απλό κατέβασμα και εγκατάσταση του αρχείου .APK της

εφαρμογής στη συσκευή, είτε με τη λήψη τους χρησιμοποιώντας ένα από τα υπάρχοντα online καταστήματα του διαδικτύου. Το Google Play Store είναι το πρωταρχικό κατάστημα εφαρμογών, το οποίο είναι δυνατό να εγκατασταθεί σε συσκευές Android οι οποίες συμμορφώνονται με τις απαιτήσεις συμβατότητας της Google και άδεια χρήσης του λογισμικού της Google Mobile Services. Το Google Play Store επιτρέπει στους χρήστες να αναζητήσουν, να κατεβάσουν και να ενημερώσουν τις εφαρμογές που δημοσιεύθηκαν από την Google ή ακόμα και από άλλους προγραμματιστές. [27]

3.3 Η γλώσσα Προγραμματισμού Python

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού η οποία δημιουργήθηκε από τον Ολλανδό Γκβίντο βαν Ρόσσουμ (Guido van Rossum) το 1990. Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της και το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ' ότι θα ήταν δυνατόν σε άλλες γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες και την ταχύτητα της εκμάθησης της.

Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

Η Python αναπτύσσεται ως ανοιχτό λογισμικό (open source) και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κώδικας διανέμεται με την άδεια Python Software Foundation License η οποία είναι συμβατή με την GPL. Το όνομα της γλώσσας προέρχεται από την ομάδα Άγγλων κωμικών Μόντυ Πάιθον. [16]

3.4 Περιβάλλοντα και Εργαλεία Ανάπτυξης Εφαρμογών

Η ανάπτυξη των Android εφαρμογών, απαιτεί την προετοιμασία κατάλληλου περιβάλλοντος ανάπτυξής της. Τα εργαλεία που χρησιμοποιούνται για την ανάπτυξη εφαρμογών ονομάζονται Software Development Kit – SDK. Η επιτυχής λειτουργία αυτών των εργαλείων, προϋποθέτει την ύπαρξη δυο άλλων συστημάτων λογισμικού: α) ένα πακέτο ανάπτυξης της Java (Java Development Kit - JDK) και β) ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE- Integrated Development Environment).

Στο σημείο αυτό, αξίζει να σημειωθεί ότι για την ανάπτυξη των parsers, publisher και του API που αλληλεπιδρά με το application και τη βάση έγινε χρήση του IDE, Eclipse. Αντίθετα, για την ανάπτυξη του mobile application έγινε χρήση του IDE Android Studio καθώς προσέφερε ορισμένες λειτουργίες και υπηρεσίες που είχε ως αποτέλεσμα την πολύ γρηγορότερη υλοποίηση της εφαρμογής. [13,15]

3.4.1 Software Development Kit (SDK).

Ένα κιτ ανάπτυξης λογισμικού (SDK) είναι συνήθως ένα σύνολο εργαλείων ανάπτυξης λογισμικού που επιτρέπει τη δημιουργία εφαρμογών για ένα συγκεκριμένο πακέτο λογισμικού, framework λογισμικού, πλατφόρμα υλικού, σύστημα υπολογιστή, κονσόλα παιχνιδιών βίντεο, λειτουργικό σύστημα ή παρόμοια πλατφόρμα ανάπτυξης. Για να εμπλουτιστούν οι εφαρμογές με προηγμένες λειτουργίες, διαφημίσεις, ειδοποιήσεις push και πολλά άλλα, οι περισσότεροι προγραμματιστές εφαρμογών υλοποιούν συγκεκριμένα κιτ ανάπτυξης λογισμικού. Πιο συγκεκριμένα, περιλαμβάνει ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες, έναν εξομοιωτή συσκευής (εικονική συσκευή) που βασίζεται στο QEMU, τεκμηρίωση, δείγματα κώδικα και tutorials. Προκειμένου όμως να αξιοποιηθούν οι δυνατότητες του SDK απαιτείται ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE- Integrated Development Environment), όπως για παράδειγμα το “Android Studio”.[27]

3.4.2 Java Development Kit (JDK)

Το Java Development Kit (JDK) προσφέρει εργαλεία τα οποία χρησιμοποιούνται από ολοκληρωμένα περιβάλλοντα IDEs καθώς και από διάφορα πακέτα ανάπτυξης λογισμικού SDKs, όπως είναι ο μεταγλωττιστής Java. Το JDK παρέχει τη δυνατότητα σε προγράμματα Java να λειτουργήσουν κανονικά στο σύστημα, καθώς περιλαμβάνει ένα περιβάλλον εκτέλεσης Java (Java Runtime Environment – JRE). Το JRE ή διαφορετικά το ιδιωτικό περιβάλλον εκτέλεσης περιλαμβάνει μια εικονική μηχανή (Java Virtual Machine) καθώς και βιβλιοθήκες κλάσεων οι οποίες είναι ιδιαίτερα χρήσιμες στους προγραμματιστές.[9,13]

3.4.3 Περιβάλλον ανάπτυξης Eclipse

Το Eclipse αποτελεί μια εξαιρετικά ισχυρή τεχνολογική πλατφόρμα, μια από τις πιο γνωστές και ευρέως χρησιμοποιούμενες, σχεδιασμένη για την κατασκευή ολοκληρωμένων περιβαλλόντων ανάπτυξης (IDE). Έχει εφαρμοστεί σε μεγάλη ποικιλία εφαρμογών για τη δημιουργία IDEs σε πολλές γλώσσες και για τη δημιουργία εξατομικευμένων IDEs για διάφορα εξειδικευμένα SDKs. Προσφέρει μια βάση εργασίας (workspace) ενώ ταυτόχρονα προσφέρει τη δυνατότητα προσαρμογής του περιβάλλοντός του, σε μεγάλο εύρος εφαρμογών εφόσον διαθέτει ένα σύστημα plug-in με δυνατότητες επέκτασης .

Το Eclipse χρησιμοποιείται ως ένα IDE για τη γραφή, τη δοκιμή και έλεγχο σφαλμάτων του λογισμικού (software debugging), ιδιαίτερα για λογισμικό Java. Υπάρχουν διάφορα παράγωγα IDEs και τα αντίστοιχα SDKs, για τα διάφορα είδη της Java ανάπτυξης λογισμικού τα οποία βασίζονται στο περιβάλλον του Eclipse. Σε αυτή την περίπτωση, προκειμένου να προσαρμοστεί κατάλληλα το περιβάλλον του Eclipse στις ανάγκες ανάπτυξης της εκάστοτε ειδικής εφαρμογής θα πρέπει να προστεθούν τα απαραίτητα plug-ins.

Το Eclipse μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών σε αρκετές γλώσσες προγραμματισμού. Πιο αναλυτικά, με τη βοήθεια διάφορων plug-ins είναι δυνατό να αναπτυχθούν εφαρμογές σε άλλες γλώσσες προγραμματισμού όπως Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Lua, Perl, PHP, Prolog, Python , R, Ruby (συμπεριλαμβανομένων των Ruby on Rails πλαίσιο), Scala, Clojure,

Groovy, Scheme και Erlang. Αξίζει να αναφερθεί ότι υπάρχουν διάφορες μορφές του αναφερόμενου περιβάλλοντος ανάπτυξης Eclipse κάθε μια από τις οποίες, προσαρμόζεται καλύτερα σε ένα μικρότερο εύρος γλωσσών προγραμματισμού. Για παράδειγμα, τα εργαλεία ανάπτυξης Eclipse Java (JDT) χρησιμεύουν περισσότερο για την ανάπτυξη εφαρμογών σε Java και Scala, ενώ το Eclipse CDT για την ανάπτυξη εφαρμογών σε C / C++ και το Eclipse PDT για την γλώσσα προγραμματισμού PHP. [13]

3.4.4 Περιβάλλον ανάπτυξης Android Studio

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο Google I/O από την Google Product Manager, Katherine Chou. Το Android Studio είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0. Το Android Studio ήταν διαθέσιμο σε πρώιμο στάδιο για προεπισκόπηση ξεκινώντας από την έκδοση 0.1 τον Μάιο του 2013, έπειτα ξεκίνησε το δοκιμαστικό στάδιο από την έκδοση 0.8 που βγήκε τον Ιούνιο του 2014. Η πρώτη σταθερή έκδοση βγήκε το Δεκέμβριο του 2014, με την έκδοση 1.0. Βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA, το Android Studio σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux, και αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

Το Android Studio, λειτουργεί όπως ακριβώς και ο προκάτοχος του (το Eclipse σε συνδυασμό με το ADT) , ωστόσο έχει να προσφέρει πολλές περισσότερες δυνατότητες στον χρήστη οι οποίες καθιστούν την δημιουργία μιας mobile εφαρμογής ευκολότερη και γρηγορότερη. Για το λόγο αυτό η Google εστίασε 100% στην ανάπτυξη και τη βελτίωση του Android Studio μετά από ένα σημείο, προτείνοντας το πλέον ως το καταλληλότερο IDE για ανάπτυξη εφαρμογών σε Android. Κάποιες από αυτές τις δυνατότητες είναι: α) η πολύ καλή πρόβλεψη του συστήματος για το τι επιθυμεί ο χρήστης να γράψει κατά την πληκτρολόγηση του κώδικα που ειδικά σε ότι αφορά το κομμάτι του Android ξεπερνάει κάθε άλλη προσπάθεια που έχει γίνει στο παρελθόν. β) η ύπαρξη ενός χειροκίνητου συστήματος με το οποίο ο χρήστης μπορεί να κατασκευάσει εξ ολοκλήρου το γραφικό κομμάτι της εφαρμογής μόνο κάνοντας “drag and drop” τα αντικείμενα που επιθυμεί να εισάγει, χωρίς να χρειαστεί να γράψει ούτε τον ελάχιστο κώδικα. γ) παραμετροποιήσιμα, έτοιμα αντικείμενα για χρήση από τον developer όπως activities, menus, taskbars, views, layouts, buttons κτλ. και πολλά άλλα.[27]

3.4.5 Σύστημα διαχείρισης βάσης δεδομένων PostgreSQL

Το PostgreSQL είναι ένα ισχυρό, ανοιχτού κώδικα σύστημα βάσεων δεδομένων σχεσιακών αντικειμένων. Έχει πάνω από 15 χρόνια ενεργούς ανάπτυξης και μία αρχιτεκτονική που έχει κερδίσει μια ισχυρή φήμη για την αξιοπιστία, την ακεραιότητα των δεδομένων και την ορθότητα της. Εκτελείται σε όλα τα μεγάλα λειτουργικά συστήματα, όπως Linux, UNIX (AIX, BSD, HP-UX, macOS, Solaris) και Windows. Είναι πλήρως συμβατό με το ACID, έχει πλήρη υποστήριξη για ξένα κλειδιά, joins, views, triggers και αποθηκευμένες διαδικασίες (σε πολλές γλώσσες). Περιλαμβάνει τους περισσότερους τύπους δεδομένων SQL:2008, συμπεριλαμβανομένων INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL και TIMESTAMP.

Υποστηρίζει επίσης την αποθήκευση δυαδικών μεγάλων αντικειμένων, συμπεριλαμβανομένων εικόνων, ήχων ή βίντεο. Διαθέτει εγγενείς διεπαφές προγραμματισμού για C / C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, καθώς και εξαιρετικό documentation.

Το σύστημα αυτό χρησιμοποιήθηκε για την υλοποίηση της βάσης δεδομένων, η οποία έχει αποθηκευμένα όλα τα δεδομένα προς παρουσίαση, σε υπολογιστή. Το μεγάλο πλήθος των δεδομένων, η SQL-λογική δόμησης της βάσης,η απλότητα,η ταχύτητα και η αποδοτικότητα στις λειτουργίες του PostgreSQL είναι οι κύριοι λόγοι που έγινε η επιλογή του ανάμεσα σε άλλα. Σε τελική ανάλυση, η εφαρμογή για να λειτουργεί σωστά απαιτεί μεγάλη αποκρισιμότητα και το συγκεκριμένο σύστημα είναι απόλυτα κατάλληλο.[37]

Κεφάλαιο 4ο - Σύστημα Διαχείρισης Δεδομένων

Στο κεφάλαιο αυτό, θα γίνει αναλυτική περιγραφή του συστήματος που είναι υπεύθυνο για τη διαχείριση των δεδομένων που θα τροφοδοτούν την εφαρμογή, από την συλλογή, έως την αποθήκευση και την ανάκτησή τους. Πιο συγκεκριμένα θα περιγραφεί η δημιουργία των αναλυτών (parsers) για τη συλλογή των δεδομένων, του publisher και της βάσης δεδομένων για την αποθήκευση και του API υπεύθυνου για την ανάκτηση των δεδομένων. Θα δοθεί έμφαση στη λειτουργικότητα αυτών των μονάδων, στη σχέση μεταξύ τους και στις προγραμματιστικές επιλογές που έγιναν κατά τη διάρκεια της υλοποίησης τους.

4.1 Συλλογή Δεδομένων

Το σύστημα χρησιμοποιεί δύο κύριες πηγές δεομένων, τους εγκατεστημένους «έξυπνους» μετρητές και την διαδικτυακή υπηρεσία παροχής καιρικών δεδομένων apixu.com. Τα αρχεία με τα δεδομένα από τους «έξυπνους» μετρητές στέλνονται μέσω FTP σε προκαθορισμένες τοποθεσίες, ανά τακτά χρονικά διαστήματα και τα αρχεία αυτά είναι σε θέση να αναγνωστούν για να συλλεχθούν τα δεδομένα. Ταυτόχρονα, τα καιρικά δεδομένα είναι διαθέσιμα μέσω δικτυακής εφαρμογής, από την οποία συλλέγονται και στη συνέχεια επεξεργάζονται. Σε αυτό το πλαίσιο, δημιουργήθηκαν δύο τύποι αναλυτών (parsers) που εξυπηρετούν την συλλογή και επεξεργασία των δεδομένων από αυτές τις δύο διαφορετικές πηγές.

4.1.1 Αναλυτής δεδομένων από τους αισθητήρες

Ο πρώτος αναλυτής που θα περιγραφεί δημιουργήθηκε με σκοπό την ανάγνωση των data που προέρχονται από αρχεία τύπου .csv τα οποία προκύπτουν από αισθητήρες που είναι τοποθετημένοι στον εσωτερικό και εξωτερικό χώρο του εργαστηρίου. Η ανάπτυξη του εν λόγω κώδικα έγινε με σκοπό να αποτελεί έναν αυτοματοποιημένο τρόπο ανάγνωσης ώστε να διαβάζεται με ευκολία όλος ο όγκος των δεδομένων που προκύπτουν από τις μετρήσεις των αισθητήρων κάθε 5 λεπτά. Επίσης, στόχος ήταν ο κώδικας να είναι αρκετά γενικευμένος, ώστε να μπορεί να τρέξει και να διαβάσει δεδομένα και από άλλες πηγές και άλλους τύπους αρχείων. Για το λόγο αυτό γίνεται χρήση ενός configuration file με την παραμετροποίηση του οποίου μπορεί να γίνει εφικτή η ανάγνωση διαφορετικής πληροφορίας από τον χρήστη χωρίς όμως να χρειαστεί να επέμβει καθόλου στον κώδικα. Η λογική του κώδικα και του configuration file περιγράφονται αναλυτικά παρακάτω.

Ανάγνωση στοιχείων του Configuration File

Σε πρώτη φάση, ο αναλυτής δέχεται ως είσοδο το όνομα ενός configuration file (αρχείο με επέκταση .cfg) και διαβάζει την πληροφορία που περιέχεται σε αυτό. Το αρχείο αυτό είναι σε μορφή JSON. Η επιλογή αυτή έγινε καθώς η Java προσφέρει πληθώρα βιβλιοθηκών που είναι σχετικές με τη διαχείριση αντικειμένων JSON (JSON Objects).[39] Στη συγκεκριμένη περίπτωση η βιβλιοθήκη που χρησιμοποιήθηκε είναι η JSON-Simple αφού η χρήση της ήταν αρκετά απλή και δεν ήταν απαραίτητες πιο σύνθετες βιβλιοθήκες που προσφέρουν πολύ περισσότερες λειτουργίες. Από το configuration file γίνεται ανάγνωση αρχικά γενικών πληροφοριών όπως: το μονοπάτι (path) στο οποίο είναι αποθηκευμένα τα data από τους αισθητήρες, το κτίριο το οποίο

αφορούν τα δεδομένα, ο τύπος αρχείων των δεδομένων (.csv ή κάποια άλλη μορφή), η μορφή της ημερομηνίας και της ώρας μέσα στα δεδομένα.[23]

Στη συνέχεια με χρήση του αντικειμένου JSON Array της χρησιμοποιημένης βιβλιοθήκης, διαβάζονται σε μορφή πίνακα όλοι οι τύποι χρονοσειρών και τα χαρακτηριστικά τους για τον εντοπισμό των δεδομένων που τις αφορούν μέσα στα αρχεία των δεδομένων. Τα χαρακτηριστικά αυτά είναι :

- "stream_name": το όνομα της χρονοσειράς,
- "filename": ο τύπος των δεδομένων (που γράφονται πάνω στο αρχείο πχ. Data, Function_ Events,Module_Events),
- "start_row": η σειρά στην οποία ξεκινούν τα δεδομένα (υπάρχουν περιπτώσεις που τα δεδομένα δεν ξεκινούν από την πρώτη σειρά)
- "date_col": η στήλη μέσα στο αρχείο στην οποία βρίσκεται η ημερομηνία,
- "date_row": η σειρά στην οποία βρίσκεται η ημερομηνία (αφορά στα δεδομένα στα οποία η ημερομηνία γράφεται σε μία σειρά και όχι σε όλες. Τίθεται -1 αν το κάθε δεδομένο έχει τη δική του ημερομηνία μέσα στο αρχείο),
- "time_col": η στήλη μέσα στο αρχείο στην οποία βρίσκεται η ώρα,
- "date_time_col": η στήλη μέσα στο αρχείο στην οποία βρίσκεται το date_time (αφορά στα δεδομένα που η ώρα και η ημερομηνία δίνονται σαν μία ενιαία πληροφορία σε μια συγκεκριμένη μορφή. Η μορφή αυτή δίνεται στις γενικές πληροφορίες του Configuration File, με χρήση του πεδίου "date_time_format", και μπορεί να καλύψει περιπτώσεις δεδομένων που χρονικά καθορίζονται με χρήση timestamp ή με κάποια άλλη κωδικοποίηση. Σε περίπτωση που δεν υπάρχουν τέτοιας μορφής δεδομένα τίθεται ως -1) ,
- "value_col": η στήλη μέσα στο αρχείο στην οποία βρίσκεται η τιμή του ενεργειακού δεδομένου που αφορά την χρονοσειρά,
- "id_col": η στήλη μέσα στο αρχείο στην οποία βρίσκεται το αναγνωριστικό όνομα (id) της χρονοσειράς που αφορά την κάθε σειρά-δεδομένο.
- "id_value": το αναγνωριστικό όνομα (id) της χρονοσειράς
- "multiplier": ένας πολλαπλασιαστής που χρησιμοποιήθηκε για την εκτίμηση της κατανάλωσης ενέργειας στο διάστημα των πέντε λεπτών έως ότου καταφτάσουν τα επόμενα δεδομένα από τους αισθητήρες. Συγκεκριμένα, θεωρήθηκε ότι η κατανάλωση μέσα σε αυτό το διάστημα παραμένει σταθερή . Επομένως, για τα 5 λεπτά της ώρας (60 λεπτά) μπορούμε να κάνουμε μια εκτίμηση για την κατανάλωση σε KWh πολλαπλασιάζοντας την κατανάλωση (σε Watt) με έναν πολλαπλασιαστή που παίρνει τιμή multiplier=(5/60)/1000=0.0000833. Με τον τρόπο αυτό έχουμε μια μετατροπή

της κατανάλωσης από Watt σε KWh . Με παρόμοιο τρόπο μπορούμε να κάνουμε και άλλες μετατροπές για άλλα μεγέθη σε άλλες χρονοσειρές. (Στις περιπτώσεις που δεν απαιτείται καμία μετατροπή, ο όρος αυτός τίθεται 1.)

Είναι προφανές ότι με την χρήση όλων των παραπάνω παραμέτρων ο χρήστης μπορεί να εξάγει πληροφορία η οποία είναι αποθηκευμένη σε διάφορες μορφές απλά αλλάζοντας τα κατάλληλα πεδία για το κάθε stream.

Ανάγνωση αρχείων δεδομένων

Όταν όλη η απαραίτητη πληροφορία από το configuration file έχει αποθηκευτεί, ο parser ξεκινάει να διαβάζει ένα ένα τα αρχεία που περιέχουν τα ενεργειακά δεδομένα των αισθητήρων από τον φάκελο "unprocessed" που βρίσκεται στο δοσμένο μονοπάτι (path). Σε περίπτωση που αυτά είναι συμπιεσμένα (.rar, .zip) τα αποσυμπιέζει και τα αποθηκεύει προσωρινά σε ένα φάκελο (temp) έως ότου γίνει η ανάγνωση όλων των δεδομένων. Όταν ολοκληρωθεί η ανάγνωση, τα περιεχόμενα του φακέλου "temp" διαγράφονται και το αναγνωσμένο αρχείο μεταφέρεται από τον φάκελο "unprocessed" στον φάκελο "processed".

Κατά την ανάγνωση, επιλέγονται επαναληπτικά οι χρονοσειρές μία μία και στη συνέχεια γίνεται προσπέλαση της κάθε σειράς μέσα στο αρχείο ανάγνωσης ώστε αρχικά να γίνει γνωστό αν αυτή η σειρά αφορά την επιλεγμένη χρονοσειρά. Ο τρόπος με τον οποίο πραγματοποιείται αυτή η επαλήθευση είναι με τη χρήση των πεδίων "id_col" και "id_value". Το κάθε stream έχει τις δικές τιμές για αυτά τα δύο πεδία. Ως εκ τούτου, σε μία συγκεκριμένη γραμμή γίνεται έλεγχος αν στο αρχείο στην στήλη με αριθμό id_col περιέχεται το id_value του επιλεγμένου stream. Αν αυτό ισχύει τότε ο parser αποθηκεύει σε ένα αντικείμενο JSON Array το δεδομένο με όνομα το όνομα του stream κρατώντας την ημερομηνία, την ώρα και την τιμή του δεδομένου. Η διαδικασία αυτή επαναλαμβάνεται για κάθε γραμμή του αρχείου και για κάθε χρονοσειρά. Όταν ολοκληρωθεί όλα τα JSON Arrays, που το καθένα περιέχει πληροφορίες για διαφορετικό stream, εισάγονται σε ένα JSON Object το οποίο μεταφέρεται και αποθηκεύεται σε αρχείο στον φάκελο "JSONResults". Με τον τρόπο αυτό ο parser διαβάζει και όλα τα υπόλοιπα διαθέσιμα αρχεία αισθητήρων και εξάγει αντίστοιχο αριθμό αρχείων JSON μέσα στα οποία πλέον η πληροφορία είναι αποθηκευμένη σε πολύ πιο αξιοποιήσιμη μορφή.

Στο σημείο αυτό αξίζει να επισημανθεί η δημιουργία μιας μεθόδου με όνομα: "MyFormat" η οποία μπορεί να εξάγει την πληροφορία για τη χρονική στιγμή της μέτρησης ανάλογα από το format της ημερομηνίας και της ώρας. Ο χρήστης αρκεί να επέμβει στο configuration file και να συμπληρώσει κατάλληλα τα πεδία "date_format","time_format", "date_time_format" και ο κώδικας έχει προσαρμοστεί με τέτοιο τρόπο ώστε να βρίσκει και να αποθηκεύει για κάθε είσοδο στο JSON αρχείο την ημερομηνία και την ώρα σε κατάλληλη μορφή.[15,17]

4.1.2 Αναλυτής δεδομένων για το API καιρικών δεδομένων

Ο δεύτερος αναλυτής ακολουθεί την ίδια λογική με τον πρώτο (δηλαδή χρήση configuration file ώστε να μην είναι αναγκαία η επέμβαση στον κώδικα), αλλά αυτή τη φορά διαβάζει την πληροφορία από ένα εξωτερικό API πρόβλεψης καιρικών συνθηκών και εξάγει την πληροφορία που επιθυμεί ο χρήστης. Αναλυτικά η διαδικασία αυτή περιγράφεται παρακάτω.

Ανάγνωση στοιχείων του Configuration File

Αρχικά, ο αναλυτής δέχεται ως είσοδο το όνομα ενός configuration file (αρχείο με επέκταση .cfg) και διαβάζει την πληροφορία που περιέχεται σε αυτό. Το αρχείο αυτό είναι επίσης σε μορφή JSON. Για τους ίδιους λόγους έγινε χρήση της βιβλιοθήκης JSON-Simple για λόγους απλότητας. Από το configuration file γίνεται ανάγνωση αρχικά γενικών πληροφοριών όπως:

- το μονοπάτι (path) στο οποίο θα αποθηκευτεί σε αρχείο η πληροφορία που θα εξάγει το API,
- το κτίριο το οποίο αφορούν τα δεδομένα,
- ο τύπος αρχείων των δεδομένων (στη συγκεκριμένη περίπτωση δίνονται σε μορφή JSON),
- ένα "κλειδί" (api_key) για το API πρόβλεψης καιρού το οποίο χρησιμοποιείται για να γίνει ή αίτηση στο API και να ελέγχεται η ποσότητα αυτών των αιτήσεων. Ο λόγος αυτού του ελέγχου είναι η αποφυγή υπερφόρτωσης του API με επαναλαμβανόμενα request από πολλαπλούς χρήστες, κάτι το οποίο μπορεί να οδηγήσει σε μη φυσιολογική λειτουργία του.
- οι μέρες της πρόβλεψης που επιθυμεί ο χρήστης (days_of_forecast),
- η τοποθεσία για την οποία θέλει να λαμβάνει την πρόβλεψη (place).

Στη συνέχεια με χρήση του αντικειμένου JSON Array της χρησιμοποιημένης βιβλιοθήκης, διαβάζονται σε μορφή πίνακα όλοι οι τύποι χρονοσειρών και τα χαρακτηριστικά τους για τον εντοπισμό των δεδομένων που τις αφορούν μέσα στα αρχεία των δεδομένων. Τα χαρακτηριστικά αυτά είναι :

- "stream_name": το όνομα της χρονοσειράς,
- "stream_type": ο τύπος των δεδομένων που επιστρέφει το API, όπως για παράδειγμα "ημερήσια", "ωριαία" κτλ.

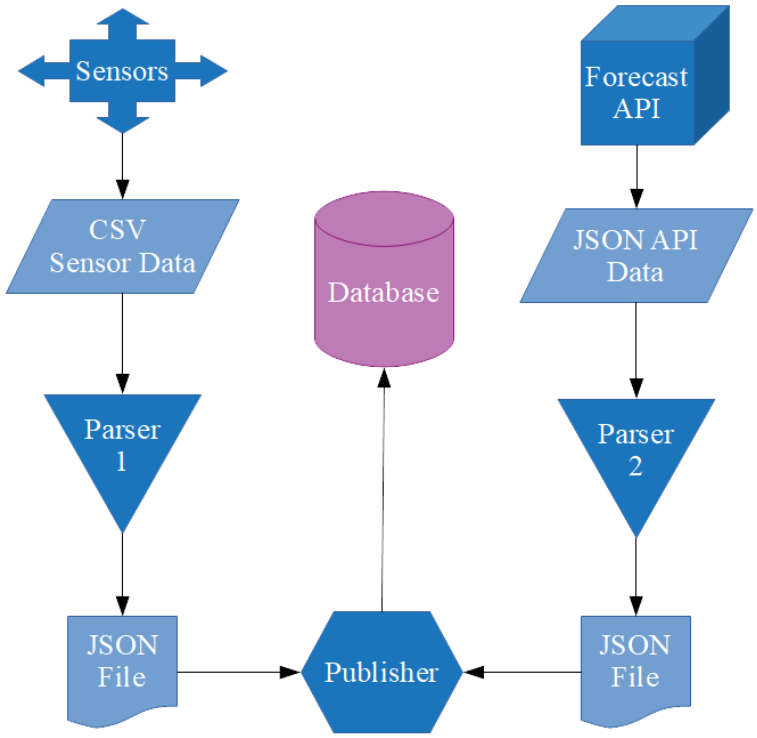
Request στο API πρόβλεψης καιρού

Όταν όλη η απαραίτητη πληροφορία από το configuration file έχει αποθηκευτεί, ο parser κάνει ένα request στο API με χρήση του κλειδιού (api_key), τον αριθμό των ημερών πρόβλεψης και την τοποθεσία που αφορά τον χρήστη (όπως αυτά έχουν δηλωθεί στο configuration file). Το API δέχεται το request και εφόσον γίνει δεκτό (δηλαδή δεν έχει ξεπεραστεί ο επιτρεπτός αριθμός αιτήσεων και οι δοθείσες πληροφορίες είναι σωστές), επιστρέφεται σε μορφή JSON η ζητούμενη πρόβλεψη για όλα τα μεγέθη τα οποία μετράει το API. Στη συνέχεια γίνεται η ανάγνωση αυτής της πληροφορίας.[30]

Κατά την ανάγνωση, επιλέγονται επαναληπτικά οι χρονοσειρές που έχουν δηλωθεί στο configuration file μία μία και στη συνέχεια γίνεται ανάγνωση της πληροφορίας για κάθε

μία από τις μέρες πρόβλεψης (και κάθε ώρα αν αυτό είναι στοιχείο της επιλεγμένης χρονοσειράς). Ο parser αποθηκεύει σε ένα αντικείμενο JSON Array το δεδομένο με όνομα το όνομα του stream κρατώντας την ημερομηνία, την ώρα και την τιμή του δεδομένου. Η διαδικασία αυτή επαναλαμβάνεται για κάθε χρονοσειρά. Όταν ολοκληρωθεί όλα τα JSON Arrays, που το καθένα περιέχει πληροφορίες για διαφορετικό stream, εισάγονται σε ένα JSON Object το οποίο μεταφέρεται και αποθηκεύεται σε αρχείο στον φάκελο "JSONResults". Με τον τρόπο αυτό η πληροφορία είναι αποθηκευμένη σε αξιοποιήσιμη μορφή μπορεί να αναγνωστεί και να δημοσιευτεί στη βάση δεδομένων από τον publisher .

Επίσης, πρέπει να διευκρινιστεί ότι χρησιμοποιήθηκε η βιβλιοθήκη "gson_library" καθώς ήταν απαραίτητη για τη σωστή λειτουργία του API πρόβλεψης καθώς και η βιβλιοθήκη "joda_time". [38] Η δεύτερη είναι η βιβλιοθήκη που αντικατέστησε τις κλάσεις Date και Time της Java και τις "φτωχές" μεθόδους τους που ήταν αρμόδιες για τη διαχείριση τιμών ημερομηνίας και ώρας, καθώς προσέφερε πολύ καλύτερη ακρίβεια, ευκολία στις μετατροπές, διαφορετικά συστήματα ημερολογίων και χρησιμοποιήθηκε καθολικά από το μεγαλύτερο κομμάτι των Developers.



Εικόνα 4.1- Σχεδιάγραμμα ροής πληροφορίας σε Parsers, Publisher και Βάση Δεδομένων

4.2 Βάση Δεδομένων

Σημαντικό κομμάτι δεν είναι μόνο η ανάγνωση των δεδομένων αλλά και η αποθήκευση τους με τέτοιο τρόπο ώστε να είναι εύκολα προσβάσιμα. Η δημιουργία μιας σωστής βάσης δεδομένων και οι λόγοι ύπαρξης συγκεκριμένων δομών μέσα σε αυτή περιγράφονται αναλυτικά στο παρακάτω κεφάλαιο. Αναλύεται ο σχεδιασμός της βάσης

δεδομένων και η μεταφορά της σε υπολογιστικό σύστημα.

4.2.1 Σχεδιαστικές Επιλογές

Βασικός τύπος οντοτήτων για τη συγκεκριμένη εφαρμογή είναι η "χρονοσειρά", αφού η παρουσίαση ενεργειακών μεγεθών σε ορισμένα από τον χρήστη χρονικά διαστήματα αποτελεί την κυριότερη λειτουργικότητα της εφαρμογής. Κάθε χρονοσειρά έχει ένα μοναδικό όνομα, το οποίο είναι το "κλειδί" για τον συγκεκριμένο τύπο οντοτήτων.[11,14] Επίσης, άλλα χαρακτηριστικά της είναι:

- Ένα όνομα με το οποίο θα απεικονίζεται μέσα στην εφαρμογή η χρονοσειρά (η συγκεκριμένη επιλογή έγινε για πιο όμορφο οπτικό αποτέλεσμα καθώς το όνομα-κλειδί ήταν πολύ μεγάλο για να χρησιμοποιηθεί)

- Ο τύπος της χρονοσειράς, για παράδειγμα "ΘΕΡΜΟΚΡΑΣΙΑ", "ΥΓΡΑΣΙΑ", "ΚΑΤΑΝΑΛΩΣΗ" κλπ.

- Το χρώμα που θα χρησιμοποιηθεί για την γραφική παράσταση της.(σε δεκαεξαδικό σύστημα RGB)

- Η μονάδα μέτρησης του μεγέθους.

- Η αθροιστική συνάρτηση του μεγέθους (aggregate function), η οποία είναι πολύ χρήσιμη ώστε να αθροίζονται με σωστό τρόπο τα δεδομένα ανάλογα με τη φύση τους για τα ζητούμενα από τον χρήστη χρονικά διαστήματα. Για παράδειγμα, για τις τιμές τις θερμοκρασίας έχει νόημα να παίρνουμε τον μέσο όρο όταν ψάχνουμε μια ενδεικτική τιμή για ένα χρονικό διάστημα. Αντίθετα, για τις τιμές της κατανάλωσης ενέργειας αθροίζουμε όλες τις επί μέρους καταναλώσεις καθώς επιθυμούμε να γνωρίζουμε την συνολική κατανάλωση.

Κάθε μία συγκεκριμένη χρονοσειρά την θεωρήσαμε διαφορετικό τύπο οντοτήτων για δύο λόγους. Ο πρώτος είναι η εύκολη προσπέλαση και η αποφυγή αναζήτησης σε πολύ μεγάλο (και συνεχώς αυξανόμενο) όγκο δεδομένων που αφορούν τις χρονοσειρές όλες μαζί. Ο δεύτερος είναι η δυνατότητα μελλοντικής προσθήκης παραπάνω χαρακτηριστικών που διαφέρουν από χρονοσειρά σε χρονοσειρά. Τα attributes κάθε χρονοσειράς είναι: η τιμή του μεγέθους, η ημερομηνία και η ώρα. Μάλιστα, ο συνδυασμός ημερομηνίας και ώρας αποτελεί κλειδί για κάθε εγγραφή αφού είναι μοναδικός.

Επίσης, ο χώρος εργασίας αποτελεί διαφορετικό τύπο οντοτήτων, όπως και το κτίριο στο οποίο ανήκει ο κάθε χώρος. Στην παρούσα εφαρμογή δεν χρησιμοποιείται σαν παράμετρος το κτίριο ωστόσο προσφέρει δυνατότητες επέκτασης της εφαρμογής από έναν χώρο, σε ολόκληρο ίδρυμα .

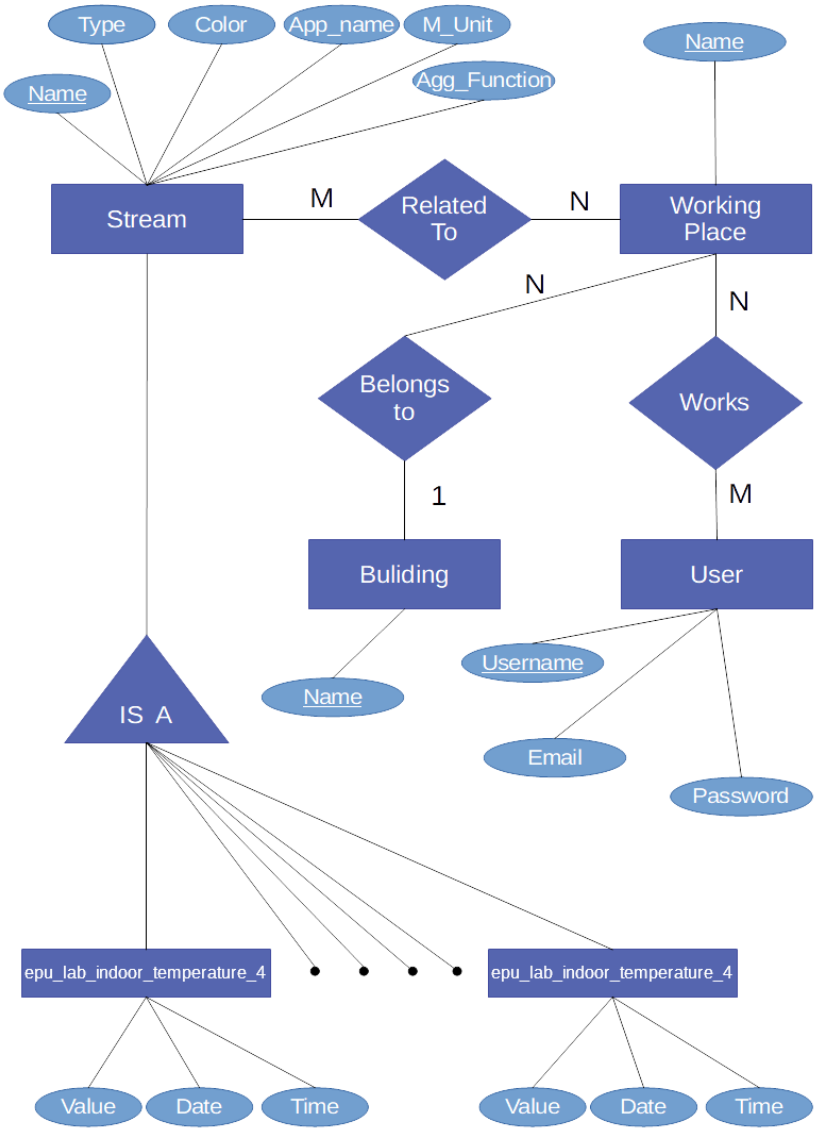
Ο χρήστης είναι ένας ξεχωριστός τύπος οντοτήτων επίσης και έχει ως κλειδί το όνομα χρήστη (username) το οποίο είναι μοναδικό για τον καθένα. Ο συγκεκριμένος τύπος είναι χρήσιμος για την αποθήκευση των προσωπικών στοιχείων του χρήστη και την εξακρίβωση αυτών κατά την είσοδο του στην εφαρμογή.

Υπάρχει προφανής συσχέτιση μεταξύ του χώρου εργασίας και του κτιρίου στο οποίο

ανήκει. Επίσης, υπάρχει συσχέτιση μεταξύ του χρήστη και του χώρου στον οποίο εργάζεται (πληθικότητας M:N). Μόνο ο διαχειριστής μπορεί να βλέπει τις χρονοσειρές για όλους τους χώρους εργασίας και για όλα τα κτίρια. Τέλος, υπάρχει συσχέτιση μεταξύ του χώρου εργασίας και του τύπου της χρονοσειράς (πληθικότητας M:N). Η συγκεκριμένη επιλογή έχει γίνει με σκοπό τον περιορισμό στην παρουσίαση δεδομένων σε κάθε χρήστη ανάλογα με την θέση εργασίας του. Αυτό έχει και περισσότερο νόημα αφού η παρουσίαση πολλαπλών χρονοσειρών που δεν αφορούν τον χρήστη απλά υπερφορτώνουν την εφαρμογή και δεν έχουν καμία ουσία στις επιλογές που μπορεί να κάνει για να εξοικονομήσει, για παράδειγμα, ενέργεια, αφού δεν πρόκειται για τον δικό του χώρο εργασίας. Όσον αφορά στη συμμετοχή, διευκρινίζεται ότι κάθε τύπος οντοτήτων που σχετίζεται με έναν άλλον έχει ολική συμμετοχή.[11,14]

4.2.2 Μοντέλο Οντοτήτων - Συσχετίσεων

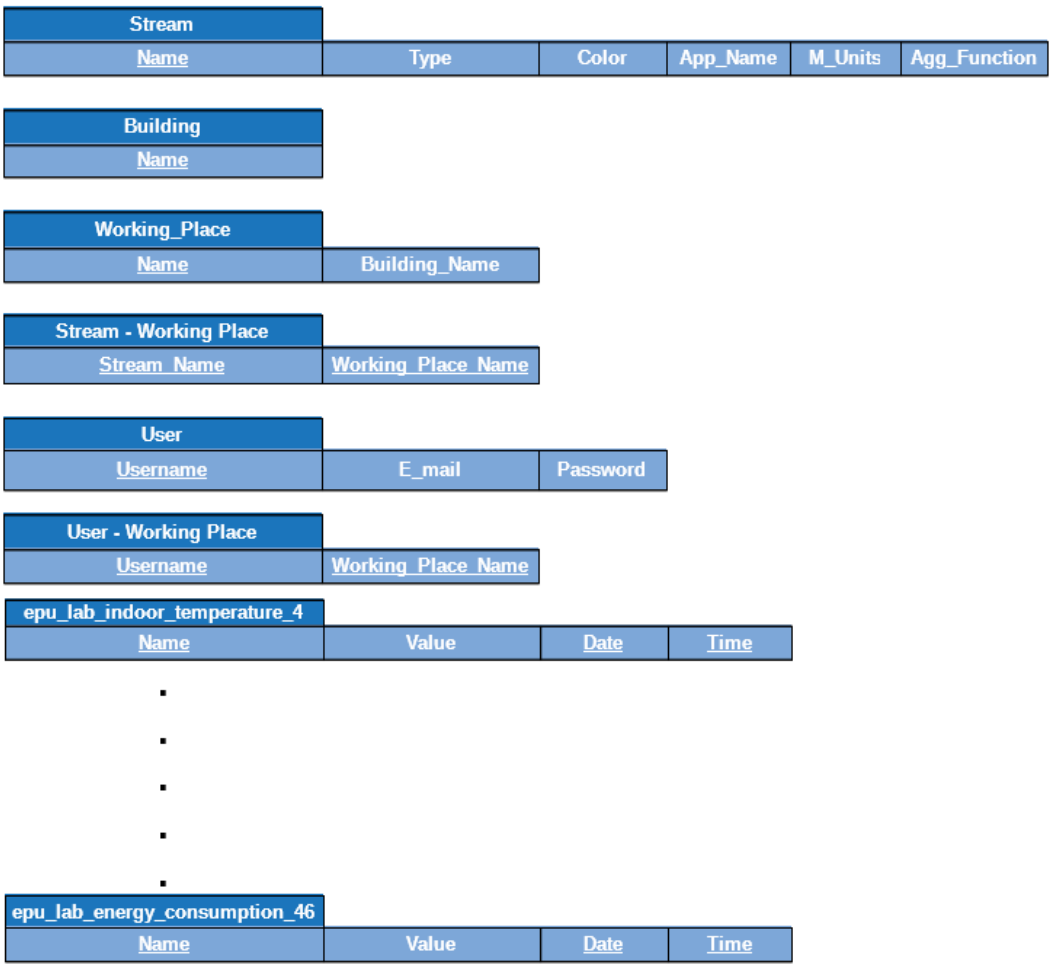
Παρακάτω παρουσιάζεται ο σχεδιασμός του ER-Μοντέλου της βάσης δεδομένων, ως το πρώτο βήμα για την υλοποίηση της.



Εικόνα 4.2- ER- Μοντέλο Βάσης Δεδομένων

4.2.3 Μετατροπή σε Σχεσιακό Μοντέλο

Παρακάτω παρουσιάζεται το σχεσιακό μοντέλο της βάσης δεδομένων, το οποίο αποτελεί το κύριο βήμα για την εισαγωγή της βάσης σε υπολογιστικό σύστημα.



Εικόνα 4.3- Σχεσιακό Μοντέλο Βάσης Δεδομένων

4.2.4 Δημιουργία Βάσης Δεδομένων

Η ολοκλήρωση της δημιουργίας της βάσης δεδομένων συμβαίνει με την εισαγωγή της βάσης στο PostgreSQL.[37] Η διαδικασία αυτή αποτελεί το ευκολότερο κομμάτι, αφού το σχεσιακό μοντέλο είναι η καταλληλότερη μορφή που μπορεί να έχει η βάση σε θεωρητικό επίπεδο ώστε να εισαχθεί σε υπολογιστικό σύστημα.

Αρχικά, γίνεται εκκίνηση του τοπικού server στον οποίο επιθυμούμε να τρέχει η βάση και δημιουργούμε μία νέα βάση. Στην περίπτωση μας την ονομάζουμε “database_energydb”. Το όνομα αυτό καθώς και ο κωδικός που θα εισάγουμε για την ασφάλεια της είναι σημαντικό για το επόμενο στάδιο, όπου ο publisher θα χρειαστεί να έχει πρόσβαση σε αυτή για ανάγνωση και αποθήκευση πληροφοριών.

Στη συνέχεια, δημιουργούνται τόσοι πίνακες (tables), όσοι φαίνονται στο σχεσιακό μοντέλο της βάσης καθένας από τους οποίους έχει τόσες στήλες όσα είναι τα πεδία του κάθε πίνακα. Για κάθε ένα από αυτά τα στοιχεία εισάγεται ο τύπος των δεδομένων του καθενός. Για παράδειγμα, ορίζουμε το πεδίο της ημερομηνίας ως “DATE” , της ώρας ως “TIME WITHOUT TIMEZONE”, και τα υπόλοιπα ως “CHARACTER VARYING” αφού πρόκειται για συμβολοσειρές. (strings).

Στο σημείο αυτό, η βάση δεδομένων είναι έτοιμη για χρήση.

4.3 Αποθήκευση Δεδομένων (Publisher)

Ο publisher είναι στην ουσία ένα αυτόνομο σύστημα, το οποίο κάνει ανάγνωση αρχείων JSON και εισάγει την πληροφορία που διαβάζει στη βάση δεδομένων. Είναι προφανές, ότι πρόκειται για το τελευταίο και σημαντικότερο βήμα για την αποθήκευση της πληροφορίας σε δομημένη μορφή και είναι κοινός για την δημοσίευση των πληροφοριών που προκύπτουν από τους αισθητήρες και από το API της πρόβλεψης. (εκτός από μία επιπλέον λειτουργία που εκτελείται στην περίπτωση αποθήκευσης των δεδομένων που αφορούν σε πρόβλεψη, η οποία εξηγείται παρακάτω αναλυτικά).Ο Publisher προσφέρει και αυτός τη δυνατότητα να χρησιμοποιηθεί σαν τμήμα ενός άλλου συστήματος αφού κατασκευάστηκε με την ίδια λογική με τους δύο Parsers. Δηλαδή, ο χρήστης μπορεί με αλλαγές στο Configuration File να τον παραμετροποιήσει κατάλληλα ώστε να είναι λειτουργικός και σε άλλο σύστημα χωρίς να χρειαστεί να επέμβει καθόλου στον κώδικα.

Ανάγνωση απαραίτητων στοιχείων από το Configuration File

Αρχικά, ο publisher δέχεται ως είσοδο το όνομα ενός configuration file (αρχείο με επέκταση .cfg) και διαβάζει τις απαραίτητες πληροφορίες μέσα από αυτό. Για χάρη της απλότητας, έγινε χρήση του ίδιου configuration file με τον parser των δεδομένων από τους αισθητήρες. Από το configuration file γίνεται ανάγνωση αρχικά γενικών πληροφοριών όπως: το μονοπάτι (path) στο οποίο είναι αποθηκευμένα τα αρχεία JSON που έχουν παράγει οι αναλυτές, το κτίριο το οποίο αφορούν τα δεδομένα, το όνομα της βάσης δεδομένων, ο κωδικός πρόσβασης σε αυτή και το όνομα του server. Στην περίπτωση μας ο server λειτουργεί τοπικά (local) και έχει όνομα: ”jdbc:postgresql://localhost:5432/database_energydb”.[22,23] Με χρήση των τριών τελευταίων στοιχείων γίνεται δυνατή η σύνδεση με τη βάση δεδομένων που έχει αναπτυχθεί και εξηγήθηκε στο προηγούμενο κεφάλαιο.

Επίσης, γίνεται ανάγνωση όλων των χρονοσειρών με τον ίδιο τρόπο που έγινε και στους parsers. Σκοπός μας είναι για κάθε μία χρονοσειρά, να γίνεται έλεγχος στα αρχεία JSON που δημιούργησαν οι parsers, και αν περιέχεται σε αυτά να εισάγονται όλες οι πληροφορίες στη βάση δεδομένων.

Σύνδεση στη Βάση Δεδομένων

Οι πληροφορίες από το configuration file που είναι απαραίτητες για τη λειτουργία του publisher έχουν αποθηκευτεί. Το επόμενο βήμα είναι η σύνδεση στη βάση. Αυτό γίνεται με χρήση της κλάσης “Connection” και “DriverManager” της Java.[37,22,23] Με χρήση του ονόματος του server, του ονόματος της βάσης και του κωδικού πρόσβασης δημιουργείται μια καινούρια σύνδεση με τη βάση δεδομένων μέσα από την οποία ο χρήστης μπορεί να εκτελέσει SQL Queries σε αυτή, να εισάγει νέα στοιχεία ή να διαγράψει στοιχεία από αυτή.

Ανάγνωση πληροφορίας από τα αρχεία JSON και εισαγωγή στη Βάση Δεδομένων

Στη συνέχεια, γίνεται ανάγνωση επαναληπτικά σε όλα τα αρχεία που έχουν δημιουργήσει οι αναλυτές και βρίσκονται στον φάκελο JSONResults. Όταν γίνει η εισαγωγή όλου του περιεχομένου του κάθε αρχείου στη βάση, τότε αυτό μετακινείται στον φάκελο JSONToDatabase.

Κατά την ανάγνωση ενός αρχείου, επιλέγονται όλες οι χρονοσειρές μία μία και σε περίπτωση που το όνομα μίας είναι ίδιο με το όνομα ενός JSONObject που περιέχεται στο αρχείο, όλες οι πληροφορίες που βρίσκονται στο συγκεκριμένο JSONObject εισάγονται στην βάση, στον πίνακα με το αντίστοιχο όνομα χρονοσειράς. Αυτό γίνεται με ένα SQL Query στη βάση το οποίο έχει τη μορφή: "INSERT INTO public."+ stream_name + "(date, time, value) VALUES (to_date('" + date + "', 'DD-MM-YYYY'),'" + time + "','" + value + "');"

Αξίζει να τονίσουμε το γεγονός ότι το σύστημα της PostgreSQL επιτρέπει την δήλωση της μορφής της ημερομηνίας (" DD-MM-YYYY") ώστε να γίνει ορθώς η ανάγνωσή της από το σύστημα. Ο λόγος είναι ότι εμείς εισάγουμε μια συμβολοσειρά η οποία μπορεί να έχει πολλές διαφορετικές ερμηνείες από το σύστημα. Δηλαδή, γίνεται δήλωση της μορφής της ημερομηνίας και με τον τρόπο αυτό η βάση αντιλαμβάνεται σωστά το δοσμένο κείμενο. (δηλαδή την αντιστοιχία ημερών, μηνών και ετών).

Σε αυτό το σημείο θα πρέπει να επισημανθεί το γεγονός ότι σε περίπτωση που ο publisher ξεκινήσει να διαβάζει αρχείο με δεδομένα πρόβλεψης από το εξωτερικό API έχει ρυθμιστεί έτσι ώστε να διαγράφει όλες τις προβλέψεις που βρίσκονται μετά από το χρονικό σημείο στο οποίο καλείται. Με τον τρόπο αυτό αρχικά αποφεύγονται οι διπλές τιμές για την ίδια χρονική στιγμή, που προκύπτουν από δύο διαφορετικές προβλέψεις, ενώ παράλληλα διατηρείται πάντοτε η πιο πρόσφατη πρόβλεψη στη βάση δεδομένων η οποία είναι και η πιο έγκυρη. Η διαδικασία αυτή πραγματοποιείται πριν γίνει εισαγωγή νέων στοιχείων στη βάση με ένα SQL Query στη βάση το οποίο έχει τη μορφή:"DELETE FROM public." + stream_name + " WHERE date>'"+currentdatetime.toLocalDate() +"';" και διαγράφει από αυτή όλα τα δεδομένα στις χρονοσειρές των προβλέψεων που βρίσκονται μετά από την παρούσα χρονική στιγμή.[37]

4.4 Δημιουργία νέων χρονοσειρών

Σε αυτό το σημείο ολοκληρώνεται η διαδικασία ανάγνωσης και αποθήκευσης των ενεργειακών δεδομένων όπως αυτά προκύπτουν από τους αισθητήρες και από εξωτερικά API. Τα δεδομένα αυτά μπορούν να παρουσιαστούν ως έχουν και να εξάγουμε σημαντικά συμπεράσματα για την θερμοκρασία, την υγρασία , την κατανάλωση ενέργειας κλπ. Ωστόσο, η αποθήκευση τους στην βάση δεδομένων δίνει την δυνατότητα να τα χρησιμοποιήσουμε με όποιον τρόπο επιθυμούμε και να παράγουμε νέα δεδομένα και ως αποτέλεσμα νέες χρονοσειρές. Αυτό δίνει πολύ περισσότερες επιλογές και δυνατότητες στους χρήστες αλλά και στους προγραμματιστές και η εφαρμογή αποκτά πραγματικό νόημα μέσα από αυτή τη διαδικασία. Μεγέθη που απασχολούν τον χρήστη όπως το κόστος ή η επιβάρυνση στο περιβάλλον μπορούν με χρήση των υπαρχόντων δεδομένων να υπολογιστούν, κάποιες φορές προσεγγιστικά και άλλες με απόλυτη ακρίβεια. Στο κεφάλαιο αυτό, λοιπόν, αναλύεται η δημιουργία επιπλέον χρονοσειρών με χρήση των δεδομένων που υπάρχουν στη βάση .

Η σκέψη είναι η δημιουργία ενός τμήματος κώδικα το οποίο θα τρέχει κάθε μια ώρα ώστε να κατασκευάζει ωριαία δεδομένα για χρήσιμα μεγέθη. Στο τέλος κάθε ώρας, θα χρησιμοποιούνται τα δεδομένα που προέκυψαν από την προηγούμενη ώρα και με την κατάλληλη επεξεργασία θα προκύπτει ένα αποτέλεσμα το οποίο θα είναι στοιχείο μιας νέας χρονοσειράς. Φυσικά αυτό μπορεί να γίνει και για παραπάνω από μια τεχνητές

χρονοσειρές. Στην εν λόγω εφαρμογή, δημιουργήσαμε τρεις νέες χρονοσειρές. Η πρώτη αφορά στην συνολική κατανάλωση του εργαστηρίου, η δεύτερη στο συνολικό κόστος της χρησιμοποιημένης ενέργειας και η τρίτη στο σύνολο των ρύπων που προέκυψαν για την παραγωγή της.

4.4.1 Χρονοσειρά Συνολικής Κατανάλωσης

Η πρώτη "τεχνητή" χρονοσειρά είναι αυτή της συνολικής κατανάλωσης. Είναι σημαντική καθώς δίνει μία γενική εικόνα για την κατανάλωση ενέργειας όλου του εργαστηρίου και μπορεί να χρησιμοποιηθεί για τον υπολογισμό του κόστους. Για κάθε χρονική στιγμή το άθροισμα τεσσάρων δεδομένων κατανάλωσης (συγκεκριμένα των χρονοσειρών με όνομα: "epu_lab_energy_consumption_11", "epu_lab_energy_consumption_12", "epu_lab_energy_consumption_13" και "epu_lab_energy_consumption_14" αντίστοιχα) αποτελεί τη συνολική κατανάλωση του εργαστηρίου. Άρα κάθε μία ώρα, αθροίζονται για κάθε μία από αυτές της χρονοσειρές τα δεδομένα που αφορούν την περασμένη ώρα και τα τέσσερα αποτελέσματα αθροίζονται μεταξύ τους. Αναλυτικά, διαβάζονται τα ζητούμενα δεδομένα με το κατάλληλο SQL Query που έχει τη μορφή: "SELECT value FROM public."+ stream_name +" WHERE time>='"+ starting_hour +"' AND time<'"+ ending_hour +"' AND date='"+ starting_date +"'" για κάθε χρονοσειρά, αθροίζονται μεταξύ τους και τέλος γίνεται η εισαγωγή του στοιχείου στη βάση με το αντίστοιχο Query. Αυτό δημιουργεί ένα αποτέλεσμα-στιγμιότυπο που είναι δεδομένο της νέας μας χρονοσειράς.

4.4.2 Χρονοσειρά Συνολικού Κόστους

Είναι προφανές ότι ακόμα και οι ίδιες οι τεχνητές χρονοσειρές μπορούν να χρησιμοποιηθούν για την παραγωγή νέων δεδομένων. Παράδειγμα αυτής της περίπτωσης είναι ο υπολογισμός του συνολικού κόστους της ενέργειας που καταναλώνεται. Χρησιμοποιήθηκε απλό μονοφασικό οικιακό τιμολόγιο για την κοστολόγηση της κατανάλωσης για λόγους απλότητας ωστόσο αποτελεί ξεχωριστή συνάρτηση στον κώδικα οπότε μπορεί να αντικατασταθεί με ευκολία με οποιοδήποτε άλλη τιμολογιακή πολιτική. (η ακόμα και με όλες και να γίνεται επιλογή της κατάλληλης μέσα από το configuration file).

Στη συγκεκριμένη περίπτωση, ορίζονται τρεις διαφορετικές χρονικές περίοδοι μέσα στον χρόνο (δηλαδή τρία τετράμηνα) και τίθεται όριο κατανάλωσης σε κάθε μία από αυτές (2000kWh).[40] Αν το όριο αυτό ξεπεραστεί τότε η χρέωση της κάθε KWh αυξάνεται για το υπόλοιπο της περιόδου. (Διευκρινίζεται ότι το συγκεκριμένο σχέδιο τιμολόγησης είχε δρομολογηθεί να εφαρμοστεί στις αρχές του 2018 και για αυτό τον λόγο επιλέχτηκε. Δεν είναι γνωστή ωστόσο ακόμα η συγκεκριμένη ημερομηνία εφαρμογής.) Τελικά, ανάλογα με το ύψος της κατανάλωσης πολλαπλασιάζεται κάθε μία ώρα η συνολική κατανάλωση με το αντίστοιχο κόστος και προκύπτουν τα νέα δεδομένα. Με τον τρόπο αυτό, ο χρήστης μπορεί να είναι ενήμερος ανά πάσα ώρα και στιγμή για το πόσο κοστίζει η ενέργεια που καταναλώνει.

4.4.3 Χρονοσειρά Συνολικών Εκπομπών Καυσαερίων

Επειδή ένας από τους στόχους της εφαρμογής είναι και η περιβαλλοντική διαχείριση,

εύλογο θεωρείται να υπολογίζονται και οι ρύποι που παράγονται κατά την παραγωγή της ενέργειας που έχει καταναλωθεί. Μια προσέγγιση αυτού του μεγέθους είναι εφικτή με τη χρήση μέσου συντελεστή εκπομπών CO2 του συνολικού συστήματος ηλεκτροπαραγωγής της ΔΕΗ ο οποίος πιο πρόσφατα έχει τιμή 1,12 tCO2/MWh .[40] Ο πολλαπλασιασμός της συνολικής κατανάλωσης ενέργειας με τον συντελεστή αυτόν μας δίνει τα νέα ωριαία δεδομένα που αφορούν τις συνολικές εκπομπές ρύπων.

Κλείνοντας, θα πρέπει να διευκρινίσουμε πως η κατασκευή νέων χρονοσειρών με βάση τις υπάρχουσες μπορεί να δώσει πολύ περισσότερες λειτουργικότητες στην εφαρμογή και είναι μια ενέργεια που δεν θα πρέπει να παραμεληθεί. Οι τρεις χρονοσειρές που δημιουργήθηκαν στο παρόν κεφάλαιο αποτελούν ένα μικρό δείγμα και είναι φανερό πως η ύπαρξη περισσότερων αισθητήρων και API μπορεί να προσφέρει ακόμα μεγαλύτερη ποικιλία στην δημιουργία νέων χρονοσειρών.

4.5Ανάκτηση των Δεδομένων

Το τελευταίο στάδιο δημιουργίας του συστήματος διαχείρισης δεδομένων είναι η δημιουργία της διεπαφής για την ανάκτηση των αποθηκευμένων δεδομένων. Σε αυτή την κατεύθυνση, δημιουργήθηκε ένα RESTful API σε γλώσσα Python,[16] με χρήση του Framework “Flask”[35] και αποτελεί τον ενδιάμεσο στην μεταφορά πληροφορίας από τη βάση δεδομένων στην mobile εφαρμογή και αντίστροφα. Στο κεφάλαιο αυτό αναφέρονται οι ιδιότητες του συγκεκριμένου API και περιγράφεται αναλυτικά η διαδικασία διεκπεραίωσης ενός request και απόκτησης πληροφορίας στην εφαρμογή.

Ιδιότητες του RESTful API

Η αρχιτεκτονική REST αρχικά σχεδιάστηκε ώστε να ταιριάζει με το HTTP πρωτόκολλο επικοινωνίας που χρησιμοποιείται στον παγκόσμιο ιστό. Η βασική λογική πίσω από την τεχνολογία αυτή είναι η εκτέλεση ενεργειών μέσω της υπηρεσίας με την χρήση ενός URI. Ο χρήστης εκτελεί ένα request μέσα από ένα URI χρησιμοποιώντας μεθόδους που ορίζονται από το HTTP πρωτόκολλο και αποτέλεσμα αυτού μπορεί να είναι είτε η απόκτηση πληροφορίας είτε η τροποποίηση της πληροφορίας. Στην περίπτωση μας έγινε χρήση δύο βασικών μεθόδων. Η μέθοδος “GET” χρησιμοποιήθηκε για όλα τα requests, τα οποία είχαν ως στόχο την απόκτηση πληροφορίας. Η μέθοδος “POST” χρησιμοποιήθηκε για όλα τα requests, τα οποία είχαν ως στόχο την τροποποίηση της πληροφορίας στην βάση δεδομένων ή την προσθήκη νέας πληροφορίας.

Το RESTful API χαρακτηρίζεται ως “stateless”. Αυτό σημαίνει ότι κάθε αίτημα από τον πελάτη πρέπει να περιέχει όλη την πληροφορία που χρειάζεται ο server για να το φέρει εις πέρας. Με άλλα λόγια, ο server δεν έχει τη δυνατότητα να αποθηκεύσει πληροφορίες που δίνονται από έναν πελάτη σε ένα request και να της χρησιμοποιήσει σε κάποιο επόμενο request.[35]

Ένα ακόμα χαρακτηριστικό του RESTful API που δημιουργήθηκε είναι η συγκεκριμενοποιημένη επιστροφή πληροφορίας. Σε κάθε αίτημα που διεκπεραιώνεται ο πελάτης στέλνει την πληροφορία με χρήση του URI, δηλώνοντας την μέθοδο και αν αναμένει απάντηση αυτή επιστρέφεται σε μορφή JSON για να χρησιμοποιηθεί στην

εφαρμογή. Σε περίπτωση που το αίτημα αποτύχει ή δεν έχει γίνει αίτηση πληροφορίας επιστρέφεται η τιμή "null".

Requests από την Mobile Εφαρμογή

Για κάθε πιθανή αλληλεπίδραση με τη βάση και για την διατήρηση βασικών πληροφοριών οι οποίες απαιτούνται σε παραπάνω από μια οθόνες της εφαρμογής δημιουργήθηκαν οι κλάσεις Database_Class και API_Class. Όταν σε κάποιο activity της εφαρμογής απαιτείται πληροφορία από τη βάση τότε μέσα από το αντικείμενο της κλάσης Database_Class καλούνται οι κατάλληλες μέθοδοι για την απόκτηση της. Οι αναγκαίες τιμές για να υλοποιηθεί το αίτημα συλλέγονται είτε από κάποια εισαγωγή του χρήστη στα Edit Texts και Spinners των οθονών είτε από το ίδιο το αντικείμενο της κλάσης Database_Class.

Όταν όλες γίνουν διαθέσιμες τότε κατασκευάζεται το URI στην κλάση API_Class. Συγκεκριμένα για κάθε διαφορετικό τύπο αιτήματος υπάρχει διαφορετική μέθοδος στην οποία κατασκευάζεται το URI, το οποίο περιέχει την IP και τη θήρα στην οποία τρέχει το API και τέλος όλα τα πεδία που είναι απαραίτητα ώστε να μπορεί το API να έχει πρόσβαση στα κατάλληλα δεδομένα στη βάση.

Στη συνέχεια πραγματοποιείται το αίτημα μέσα από την κλάση Request Class. Δημιουργείται αντικείμενο της κλάσης με όρισμα τη μέθοδο του αιτήματος σύμφωνα με το HTTP πρωτόκολλο, το οποίο ανοίγει τον δίαυλο επικοινωνίας μεταξύ του application και της βάσης. Αν το αίτημα ήταν μεθόδου "GET" τότε αναμένεται να επιστραφεί σε μορφή JSON το αποτέλεσμα και αυτό επιστρέφεται πίσω στην Database Class όπου η κατάλληλη μέθοδος επεξεργάζεται την πληροφορία και την αξιοποιεί με τον ανάλογο τρόπο. Αν το αίτημα ήταν μεθόδου "POST" τότε δεν επιστρέφεται κάποια πληροφορία. Σε κάθε περίπτωση, αν αποτύχει το αίτημα η εφαρμογή διαχειρίζεται την κατάσταση ενημερώνοντας τον χρήστη για την αστοχία.

Όλοι οι τύποι των αιτημάτων, καθώς και η ενέργειες που λαμβάνουν χώρα στο RESTful API όταν αυτά καταφτάνουν σε αυτό, περιγράφονται αναλυτικά στην επόμενη παράγραφο.

Δομή του API

Ο σχεδιασμός του API είναι πολύ απλός.[35] Στο αρχείο της υλοποίησης του υπάρχουν τα εξής:

- Μία εντολή σύνδεσης του API με τη βάση δεδομένων στον κατάλληλο host και στην σωστή θήρα.
- Μία εντολή εκκίνησης της υπηρεσίας Web η οποία ενεργοποιεί το API το οποίο τρέχει διαρκώς αναμένοντας για αιτήματα από χρήστες.
- Δηλώσεις όλων των resources, δηλαδή όλων των τύπων αιτημάτων που μπορεί να δεχτεί το API.
- Τον κώδικα που εκτελείται σε κάθε ένα από τα παραπάνω αιτήματα.

Requests και Queries

Παρακάτω περιγράφονται τα αιτήματα που γίνονται δεκτά από το API και οι μέθοδοι που εκτελούνται αντίστοιχα.

- "http://192.168.1.105:5002/login-authentication/"+username+"/"+password . Το αίτημα με το παραπάνω URI δέχεται το όνομα χρήστη και τον κωδικό που εισάγει ο χρήστης και εκτελεί query στη βάση στον πίνακα users προκειμένου να επιστραφούν όλα τα στοιχεία του χρήστη με τον παραπάνω συνδυασμό, αν αυτός υπάρχει. Αν δεν υπάρχει επιστρέφεται κενό. Το query έχει την μορφή: "SELECT * FROM users WHERE username ='%s' AND password='%s';" % (username,password). Γίνεται χρήση μεθόδου “GET”.

- “http://192.168.1.105:5002/user-working-place/" + username. Το αίτημα με αυτό το URI δέχεται μόνο το όνομα χρήστη και εκτελεί query στη βάση συνδυάζοντας πίνακες ώστε να επιστραφεί το μέρος και το κτίριο εργασίας του χρήστη. Το query έχει τη μορφή: """SELECT working_place, building FROM workingplace_user INNER JOIN working_place ON (workingplace_user.working_place=working_place.name) WHERE "user" ='%s';""" % (username). Γίνεται χρήση μεθόδου “GET”.

- "http://192.168.1.105:5002/signup-authentication/" + username. Το αίτημα με αυτό το URI δέχεται μόνο το όνομα χρήστη και εκτελεί query στη βάση ώστε να επιστραφεί η εγγραφή στη βάση του χρήστη με αυτό το κλειδί. Αν επιστραφεί χρήστης τότε σημαίνει ότι το όνομα χρησιμοποιείται, επομένως η εγγραφή δεύτερου χρήστη με το ίδιο όνομα απαγορεύεται. Το query έχει τη μορφή: "SELECT * FROM users WHERE username ='%s';" % username. Γίνεται χρήση μεθόδου “GET”.

- "http://192.168.1.105:5002/available-workplaces/"+building. Το αίτημα με αυτό το URI δεν έχει κάποιο απαιτούμενο πεδίο. Το API εκτελεί query στη βάση ώστε να επιστραφούν όλα τα κτίρια τα οποία είναι δηλωμένα σε αυτή. Το query έχει τη μορφή: "SELECT * FROM building;". Γίνεται χρήση μεθόδου “GET”.

- "http://192.168.1.105:5002/available-buildings". Το αίτημα με αυτό το URI έχει ως απαιτούμενο πεδίο το όνομα ενός κτιρίου. Το API εκτελεί query στη βάση ώστε να επιστραφούν όλοι οι χώροι εργασίας που συνδέονται με το συγκεκριμένο κτίριο στη βάση. Το query έχει τη μορφή: "SELECT * FROM working_place WHERE building='%s';" % building. Γίνεται χρήση μεθόδου “GET”.

- “http://192.168.1.105:5002/current-temp-hum". Το αίτημα με αυτό το URI δεν έχει κάποιο απαιτούμενο πεδίο. Το API εκτελεί πολλαπλά queries στη βάση ώστε να επιστραφούν οι τελευταίες τιμές όλων των χρονοσειρών που αφορούν την εσωτερική θερμοκρασία, την εξωτερική θερμοκρασία και την υγρασία. Αυτές επιστρέφονται στην εφαρμογή, υπολογίζεται ο μέσος όρος της κάθε

κατηγορίας και όλα εμφανίζονται στην κεντρική οθόνη. (ως ενημερωμένες τιμές της παρούσας θερμοκρασίας και υγρασίας του κτιρίου) Γίνεται χρήση μεθόδου “GET”.

- "http://192.168.1.105:5002/available-streams/"+streamtype+"/"+username”. Το αίτημα αυτό έχει ως απαιτούμενα πεδία το όνομα χρήστη και την κατηγορία των ζητούμενων χρονοσειρών.Έπιστρέφονται όλες οι πληροφορίες για κάθε χρονοσειρά που σχετίζεται με το δοσμένο όνομα χρήστη και ανήκει στην δοσμένη κατηγορία. Σε περίπτωση που ο χρήστης επιθυμεί την επιστροφή χρονοσειρών όλων των κατηγοριών, ως streamtype εισάγει την λέξη “ΟΛΑ”. Το query έχει τη μορφή: "SELECT name, m_unit, aggregate_function, type, color, application_name FROM stream_info INNER JOIN workingplace_stream ON (workingplace_stream.stream_name = stream_info.name) INNER JOIN workingplace_user ON workingplace_user.working_place = workingplace_stream.working_place) WHERE workingplace_user.user='" + username + "' AND stream_info.type='" + streamtype + "' GROUP BY name;". Γίνεται χρήση μεθόδου “GET”.

- "http://192.168.1.105:5002/home-stream-data/"+streamname+"/"+ starting_date + "/" + ending_date + "/" + agg_function. Το URI αυτό σχετίζεται με αίτημα που απαιτεί ως δεδομένα το όνομα μιας χρονοσειράς, το χρονικό διάστημα στο οποίο μας αφορούν τα δεδομένα και την αθροιστική συνάρτηση της συγκεκριμένης χρονοσειράς. Χρησιμοποιείται για την αρχική οθόνη της εφαρμογής προς απόκτηση δεδομένων μιας μεμονωμένης γνωστής χρονοσειράς σε μηνιαίο πλαίσιο .

- "http://192.168.1.105:5002/year-stream-data/" + streamname + "/" + starting_date + "/" + ending_date + "/"+agg_function. Το URI αυτό σχετίζεται με αίτημα που απαιτεί ως δεδομένα το όνομα μιας χρονοσειράς, το χρονικό διάστημα στο οποίο μας αφορούν τα δεδομένα και την αθροιστική συνάρτηση της χρονοσειράς. Χρησιμοποιείται στην οθόνη με τη λίστα των χρονοσειρών όταν ο χρήστης απαιτεί ετήσια δεδομένα. Αυτά συλλέγονται και αθροίζονται με βάση τη δοσμένη αθροιστική συνάρτηση ανά διαστήματα μηνών και επιστρέφονται σε κατάλληλη μορφή για χρήση από το application. Το query έχει τη μορφή: "SELECT extract(month from Month) as month, avg_value FROM (SELECT date_trunc('month', stream.date) AS Month, %s(stream.value :: FLOAT) AS avg_value FROM %s AS stream WHERE date >= '%s' and date<='%s' GROUP BY Month ORDER BY Month) as sq1;" % (agg_function,streamname,starting_date,ending_date). Γίνεται χρήση της μεθόδου “GET”.

- "http://192.168.1.105:5002/month-stream-data/" + streamname + "/" + starting_date + "/" + ending_date + "/" + agg_function. Το URI σχετίζεται με αίτημα ίδιας φύσης με το προηγούμενο αλλά αφορά μηνιαία δεδομένα. Χρησιμοποιείται στην οθόνη με τη λίστα των χρονοσειρών όταν ο χρήστης

απαιτεί μηνιαία δεδομένα. Αυτά συλλέγονται και αθροίζονται με βάση τη δοσμένη αθροιστική συνάρτηση ανά διαστήματα (28-31) ημερών και επιστρέφονται σε κατάλληλη μορφή για χρήση από το application. Το query σε SQL έχει τη μορφή: "SELECT extract(day from Day) as day, avg_value FROM (SELECT date_trunc('day', stream.date) AS Day, %s(stream.value :: FLOAT) AS avg_value FROM %s AS stream WHERE date >= '%s' and date<='%s' GROUP BY Day ORDER BY Day) as sq1;" % (agg_function,streamname,starting_date,ending_date). Γίνεται χρήση της μεθόδου “GET”.

- "http://192.168.1.105:5002/week-stream-data/" + streamname + "/" + starting_date + "/"+ending_date + "/"+agg_function. Το URI σχετίζεται με αίτημα ίδιας φύσης με το προηγούμενο αλλά αφορά εβδομαδιαία δεδομένα. Χρησιμοποιείται στην οθόνη με τη λίστα των χρονοσειρών όταν ο χρήστης απαιτεί εβδομαδιαία δεδομένα. Αυτά συλλέγονται και αθροίζονται με βάση τη δοσμένη αθροιστική συνάρτηση ανά διαστήματα 12 ωρών για 7 ημέρες και επιστρέφονται σε κατάλληλη μορφή για χρήση από το application. Το query σε SQL έχει τη μορφή: "SELECT (extract(day from Day)||'/'||extract(month from Day)||'/'||extract(year from Day)) AS sdate, TwelveHour, avg_value FROM (SELECT date_trunc('day', stream.date) AS Day, extract(hour from stream.time)::INTEGER/12*12 AS TwelveHour, %s(stream.value :: FLOAT) AS avg_value FROM %s AS stream WHERE date >= '%s' and date<='%s' GROUP BY Day, TwelveHour ORDER BY Day, TwelveHour) as sq1;" % (agg_function,streamname,starting_date,ending_date)). Γίνεται χρήση της μεθόδου “GET”.

- "http://192.168.1.105:5002/day-stream-data/" + streamname + "/" + starting_date + "/"+agg_function. Το URI σχετίζεται με αίτημα ίδιας φύσης με το προηγούμενο αλλά αφορά ημερήσια δεδομένα. Χρησιμοποιείται στην οθόνη με τη λίστα των χρονοσειρών όταν ο χρήστης απαιτεί ημερήσια δεδομένα. Αυτά συλλέγονται και αθροίζονται με βάση τη δοσμένη αθροιστική συνάρτηση ανά διαστήματα ωρών για 1 ημέρα και επιστρέφονται σε κατάλληλη μορφή για χρήση από το application. Το query σε SQL έχει τη μορφή: "SELECT to_char(Hour, 'HH24:00') || '-' || to_char(Hour + interval '1 hour', 'HH24:00') AS timeinterval, avg_value FROM (SELECT date_trunc('day', stream.date) AS Day, date_trunc('hour', stream.time) AS Hour, %s(stream.value :: FLOAT) AS avg_value FROM %s AS stream WHERE date = '%s' GROUP BY Day, Hour ORDER BY Day, Hour) as sq1;" % (agg_function,streamname,starting_date)). Γίνεται χρήση της μεθόδου “GET”.

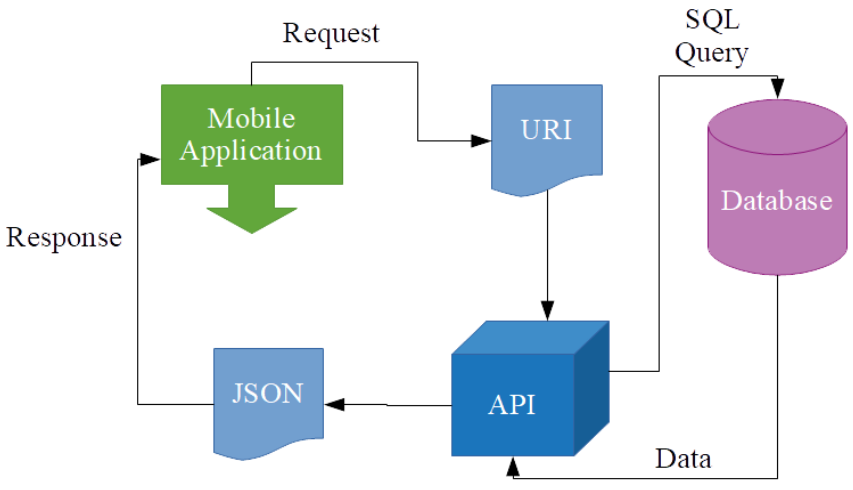
- "http://192.168.1.105:5002/change-password/"+username+"/"+newpassword. Το συγκεκριμένο URI σχετίζεται με αίτημα αλλαγής κωδικού από τον χρήστη. Πιο συγκεκριμένα, δέχεται ως ορίσματα το όνομα χρήστη και τον νέο κωδικό και με το κατάλληλο query της μορφής: "UPDATE users SET password='%s'

WHERE username ='%s';" % (newpassword,username), το API τροποποιεί το πεδίο του κωδικού του χρήστη με το δοσμένο όνομα θέτοντας ως νέα τιμή τη δοσμένη. Στη συγκεκριμένη περίπτωση γίνεται χρήση της μεθόδου “POST”.

- "http://192.168.1.105:5002/sign-up-user/" + username + "/" + password + "/" + email + "/" + name + "/" + lastname + "/"+workplace. Το συγκεκριμένο URI σχετίζεται με αίτημα εγγραφής νέου χρήστη. Πιο συγκεκριμένα, όταν ο χρήστης έχει εισάγει όλα τα στοιχεία στη φόρμα εγγραφής και αφού έχει γίνει η επιβεβαίωση ότι δεν υπάρχει ήδη στην βάση χρήστης με το ίδιο username, γίνεται το request με το παραπάνω URI που απαιτεί ως ορίσματα όλα τα στοιχεία που ο χρήστης εισήγαγε στη φόρμα της οθόνης εγγραφής και το API εκτελεί δύο queries στη βάση. Στο πρώτο εισάγει τον χρήστη και τα στοιχεία του στον πίνακα users: "INSERT INTO users (username, password, email, name, lastname, usertype) VALUES ('%s','%s','%s','%s','%s','ΑΠΛΟΣ');" % (username, password, email, name, lastname). Στο δεύτερο σχετίζει τον χρήστη με τον χώρο εργασίας του στον πίνακα workingplace_user: "INSERT INTO workingplace_user ("user",working_place) VALUES ('%s','%s');". Στη συγκεκριμένη περίπτωση γίνεται επίσης χρήση της μεθόδου “POST”.

Μετατροπή Απάντησης σε JSON

Όταν η εκτέλεση του Query ολοκληρωθεί στην βάση δεδομένων ολοκληρωθεί τότε το αποτέλεσμα που επιστρέφεται τοποθετείται με εύκολο και γρήγορο τρόπο σε μια δομή της Python η οποία ονομάζεται Dictionary.[16] Η συγκεκριμένη δομή είναι πολύ χρήσιμη αφού με μια απλή εντολή με όνομα jsonify() το περιεχόμενο ενός dictionary μετατρέπεται σε μορφή JSON. Τώρα η πληροφορία βρίσκεται πλέον στην κατάλληλη μορφή ώστε να επιστραφεί πίσω στη mobile εφαρμογή.



Εικόνα 4.4- Σχεδιάγραμμα Διαδικασίας Αιτήματος στο API

Κεφάλαιο 5ο - Σχεδίαση και Υλοποίηση Android Εφαρμογής

Στο κεφάλαιο αυτό, θα αναλυθεί η σχεδίαση της εφαρμογής Android η οποία αποτελεί το εργαλείο παρουσίασης των ενεργειακών δεδομένων. Αρχικά, θα περιγραφούν οι επιλογές που έγιναν για τη δημιουργία όλων των οθονών και τα βασικά αντικείμενα που χρησιμοποιήθηκαν ώστε να γίνει η εφαρμογή φιλική για τον χρήστη. Έμφαση θα δοθεί στις βιβλιοθήκες που χρησιμοποιήθηκαν για τις γραφικές παραστάσεις και στις δομές δεδομένων. Το συγκεκριμένο κεφάλαιο αφορά μόνο στην γενική περιγραφή στοιχείων που χρησιμοποιήθηκαν για την υλοποίηση της βάσης του application. Στο επόμενο κεφάλαιο θα εστιάσουμε στις λειτουργικότητες της εφαρμογής και στα σενάρια χρήσης της κάθε οθόνης.

5.1 Activities και Layouts

Η εφαρμογή αποτελείται από επτά διαφορετικές οθόνες, αυτό μεταφράζεται σε επτά διαφορετικά Activities [27] τα οποία περιέχουν τον κώδικα σε java που τρέχει όταν γίνεται χρήση της εκάστοτε οθόνης και σε επτά διαφορετικά layouts[8] τα οποία περιέχουν όλα τα γραφικά αντικείμενα που φορτώνει στην οθόνη κάθε activity κατά την έναρξή του. Αυτές είναι οι εξής: log-in screen, sign-up screen, στοιχεία λογαριασμού χρήστη, γενικές πληροφορίες σχετικά με την εφαρμογή, homescreen, λίστα χρονοσειρών, αναλυτική παρουσίαση επιλεγμένης χρονοσειράς.

Όλα τα activities και όλος ο κώδικας σε java που αφορά στην λειτουργία της εφαρμογής περιέχονται στον φάκελο "java" του Android Project.[27] Όλα τα layouts βρίσκονται μέσα στον ομότιτλο φάκελο ο οποίος είναι τοποθετημένος στον φάκελο "res"του Android Project. Μέσα στον φάκελο "res" υπάρχουν επίσης: ο φάκελος "drawable" που περιέχει ότι εικόνα έχει χρησιμοποιηθεί στην εφαρμογή σε διαφορετικές αναλύσεις ώστε το σύστημα να επιλέγει πάντοτε την καταλληλότερη ανάλογα με την οθόνη στην οποία προβάλλεται η εφαρμογή αλλά και σχήματα και backgrounds που ο χρήστης έχει κατασκευάσει σε αρχεία .xml, ο φάκελος "mipmap" που περιέχει τα εικονίδια της εφαρμογής σε πολλές αναλύσεις για τον ίδιο λόγο, ο φάκελος "assets" στο οποίο αποθηκεύονται προσωρινά αρχεία .txt που χρειάζεται να αναγνώσει η εφαρμογή, ο φάκελος "menu" στον οποίο περιέχεται σε μορφή .xml το menu που κατασκευάσαμε για το Action Bar της εφαρμογής και χρησιμοποιείται σε παραπάνω από μία οθόνες, και ο φάκελος values μέσα στον οποίο περιέχονται σε μορφή .xml τα βασικά χρώματα της εφαρμογής ώστε να μπορούμε να τα τροποποιήσουμε για όλη την εφαρμογή από το αρχείο αυτό, βασικές συμβολοσειρές της εφαρμογής, και τα στυλ και θέματα που χρησιμοποιούμε για διάφορα τμήματα της εφαρμογής.

Επίσης, κάθε activity καθώς και βασικές πληροφορίες που το αφορούν δηλώνεται στο αρχείο AndroidManifest.xml. Σημαντική είναι η δήλωση του activity με το οποίο ξεκινάει η εφαρμογή στην έναρξη. Αν αυτό παραβλεφθεί η εφαρμογή δεν θα μπορέσει να εκτελεστεί.

Κάθε layout που φορτώνεται κατά την έναρξη ενός activity χαρακτηρίζεται από τα στοιχεία που περιέχει και από τον τρόπο που αυτά τα στοιχεία συνδέονται μεταξύ τους και τοποθετούνται στον χώρο. Στην εν λόγω εφαρμογή έγινε κατά βάση η χρήση του

Constraint Layout [27] το οποίο προτείνεται από τους ίδιους τους δημιουργούς ως το πιο αποδοτικό και αποκρίσιμο. Το Constraint Layout βασίζεται στην σύνδεση των αντικειμένων της οθόνης με δεσμούς μεταξύ τους και με το container στο οποίο περιέχονται και την προσαρμογή των δεσμών αυτών ώστε όλα τα στοιχεία να διατηρούν την σωστή τους θέση ανεξάρτητα με το μέγεθος της οθόνης και τον προσανατολισμό της. Σε μικρά τμήματα της εφαρμογής που θεωρήθηκε ευκολότερο έγινε επίσης χρήση των Linear Layout και Frame Layout.

5.2 Χρήση Βασικών Στοιχείων Android

Μέσα στο layout της κάθε οθόνης μπορούν να περιέχονται άλλα layouts ή βασικά στοιχεία που απαρτίζουν το μεγαλύτερο κομμάτι των οθονών όλων των mobile εφαρμογών. [8] Αυτά περιγράφονται συνοπτικά παρακάτω:

Text Views

Text Views είναι οποιοδήποτε στοιχείο περιέχει αποκλειστικά κείμενο στην οθόνη. Το χρησιμοποιούμε σε πολλά σημεία της κάθε οθόνης για προβολή επικεφαλίδων, χαρακτηρισμό προβαλλόμενων γραφικών παραστάσεων κτλ. Κάθε Text View μπορεί να πάρει την τιμή του από την αρχή μέσω του αρχείου .xml στο οποίο δηλώνεται ή μπορεί να αλλάξει και δυναμικά κατά τη διάρκεια της εκτέλεσης .

Buttons

Τα κουμπιά (buttons) είναι το κύριο μέσο αλληλεπίδρασης του χρήστη με την εφαρμογή. Μπορούν να περιέχουν κείμενο αλλά και εικονίδιο. Στην δική μας περίπτωση έγινε η κατασκευή του κουμπιού από την αρχή με χρήση του αντικειμένου shape του Android. Το χρώμα, το σχήμα καθώς και η αντίδραση του κατά το πάτημα ορίζονται όλα στο αρχείο δημιουργίας του το οποίο περιέχεται στον φάκελο “drawables” του Android Project. Κουμπιά έχουν χρησιμοποιηθεί στις περισσότερες οθόνες και στις περισσότερες περιπτώσεις οδηγούν σε μετάβαση σε άλλη οθόνη ή σε αποθήκευση πληροφορίας.

EditText

Το EditText είναι μία φόρμα που συμπληρώνεται από τον χρήστη με κείμενο και η πληροφορία που εισάγεται χρησιμοποιείται από την εφαρμογή. Στην εφαρμογή μας το συγκεκριμένο στοιχείο χρησιμοποιείται στην οθόνη Log-in όπου ο χρήστης εισάγει το όνομα χρήστη και τον κωδικό του καθώς και στις οθόνες Sign-up και στα στοιχεία λογαριασμού του χρήστη. Άξιο αναφοράς είναι το γεγονός ότι μπορεί να δηλωθεί ο τύπος του κειμένου εισαγωγής για εντοπισμό λάθους ή για ασφάλεια. Για παράδειγμα αν αν το κείμενο που εισάγεται πρόκειται για κωδικό εισόδου στο σύστημα τότε δεν είναι ορατό στην οθόνη και εμφανίζεται ως μεγάλες μαύρες κουκίδες.

ImageView

Πρόκειται για στοιχείο που εμφανίζει μία εικόνα της επιλογής του χρήστη. Μπορεί να πάρει τιμή στατικά αλλά και δυναμικά και χρησιμοποιείται στην προβολή του λογότυπου της εφαρμογής στην οθόνη γενικών πληροφοριών σχετικά με την εφαρμογή.

Spinners

Τα Spinners είναι στοιχεία τύπου drop-down list τα οποία είναι χρήσιμα για την επιλογή του χρήστη μιας τιμής από μία λίστα τιμών. Χρησιμοποιούνται στην οθόνη προβολής όλων των χρονοσειρών όπου ο χρήστης έχει την επιλογή να επιλέξει ένα χρονικό διάστημα στο οποίο θέλει να παρατηρήσει τα ενεργειακά δεδομένα. Καθώς τα χρονικά διαστήματα θεωρήθηκαν εξ αρχής προσαρμοσμένα για κάθε περίπτωση, η χρήση των Spinners είναι η καταλληλότερη. Τα Spinners φορτώνονται με πληροφορία δυναμικά κατά την εκτέλεση του προγράμματος και μπορεί να δοθεί μια αρχική τιμή για αυτά. Κάθε στοιχείο που περιλαμβάνεται στο Spinner φορτώνεται από έναν Adapter και μορφή που έχει δίνεται σε ξεχωριστό layout.xml.

ScrollView

Το συγκεκριμένο στοιχείο είναι ένα δοχείο στοιχείων σαν τα παραπάνω. Είναι μια λίστα αντικειμένων και η χρήση της ενδείκνυται σε περιπτώσεις που τα στοιχεία που θέλουμε να προβάλουμε είναι πολλά σε αριθμό και δεν γίνεται να εμφανίζονται όλα ταυτόχρονα στην οθόνη. Στην περίπτωση αυτή ο χρήστης μπορεί να σύρει την οθόνη προς τα κάτω (scroll) και να δει όλα τα στοιχεία που βρίσκονται μέσα στο ScrollView. Δεν συνηθίζεται η χρήση της σε παρουσίαση δεδομένων.

ListView

Το ListView έχει ακριβώς τον ίδιο ρόλο με το προηγούμενο στοιχείο με δύο βασικές διαφορές. Αρχικά, υποστηρίζεται και φορτώνεται με περιεχόμενο από τον αντίστοιχο Adapter. Δεύτερον, το ListView φορτώνει τα στοιχεία "On Demand" και όχι όλα. Αυτό σημαίνει ότι φορτώνει στην μνήμη μόνο τα στοιχεία που είναι ορατά στον χρήστη τη συγκεκριμένη στιγμή και όχι ολόκληρη τη λίστα. Ως αποτέλεσμα, ιδιαίτερα στις περιπτώσεις που το περιεχόμενο της λίστας είναι μεγάλο, η χρήση του ListView θεωρείται πολύ πιο αποδοτική. Στην εφαρμογή που αναπτύσσουμε έγινε χρήση του ListView στην προβολή της λίστας όλων των χρονοσειρών. Δημιουργήθηκε η κλάση "Custom_List_Adapter" η οποία επεκτείνει την κλάση "ArrayAdapter<String>" και περιέχει όλες τις απαραίτητες μεθόδους για την φόρτωση των δεδομένων στη λίστα επαναληπτικά. Η κάθε σειρά-αντικείμενο της λίστας είναι σε μορφή η οποία δηλώνεται σαν ξεχωριστό layout.xml ώστε να υπάρχει ομοιομορφία (περιέχει ένα TextView και ένα διάγραμμα).

GridView

Το GridView είναι ένα δοχείο αντικειμένων το οποίο τα προβάλει σε δύο διαστάσεις σε μορφή πλέγματος και δίνει τη δυνατότητα scrolling. Στη δική μας περίπτωση, τα αντικείμενα φορτώνονται από τον "Custom_Grid_Adapter" ο οποίος επεκτείνει την κλάση "Base_Adapter" και περιέχει τις μεθόδους για την δημιουργία της λίστας.

5.3 Διαγράμματα - MPAndroid Chart

Όπως έχει αναφερθεί και προηγουμένως, η παρουσίαση των δεδομένων αποτελεί τον κύριο στόχο της εφαρμογής. Ως εκ τούτου, θεωρείται απαραίτητη η χρήση κώδικα που μπορεί να δημιουργήσει γραφικές παραστάσεις και διαγράμματα τα οποία είναι φιλικά προς τον χρήστη. Η ελεύθερη βιβλιοθήκη "MPAndroid Chart" [29] έχει ακριβώς αυτόν

τον ρόλο και η επιλογή της έγινε καθώς πρόκειται για την πιο ολοκληρωμένη, εύχρηστη και καλαίσθητη open-source βιβλιοθήκη με πλήρες documentation και ενεργό forum. Παρακάτω, αναφέρονται συνοπτικά τα βασικά στοιχεία αυτής τα οποία χρησιμοποιήθηκαν.

Δήλωση του διαγράμματος στο Layout της οθόνης

Αρχικά το διάγραμμα (LineChart, BarChart, ScatterChart, CandleStickChart, PieChart, BubbleChart , RadarChart) δηλώνεται σε στο αρχείο .xml της οθόνης και στο αντίστοιχο activity μπορεί να χρησιμοποιηθεί αφού συνδεθεί με μια μεταβλητή με χρήση μιας εντολής όπως η "LineChart chart = (LineChart) findViewById(R.id.chart)". Με τον τρόπο αυτό δημιουργείται το στιγμιότυπο του διαγράμματος.

Προσθήκη δεδομένων

Επόμενο βήμα είναι η προσθήκη δεδομένων στο στιγμιότυπο που έχει δημιουργηθεί. Αυτό γίνεται προσθέτοντας τις τιμές και για τους δύο άξονες του κάθε δεδομένου σε μία λίστα αντικειμένων τύπου Entry τα οποία είναι αναγνώσιμα από τη βιβλιοθήκη ώστε να αναπαρασταθούν στους άξονες.

Στη συνέχεια, η κάθε λίστα από Entries που δημιουργούμε αντιστοιχείται σε ένα DataSet κάθε ένα από τα οποία μπορεί να έχει τα δικά του χαρακτηριστικά κατά την απεικόνιση του διαγράμματος.

Τέλος, το κάθε DataSet αντιστοιχείται σε ένα αντικείμενο τύπου Data το οποίο εισάγεται στην γραφική παράσταση με την εντολή: "chart.setData(data)". Αξίζει να σημειωθεί ότι για συγκεκριμένους τύπους διαγραμμάτων είναι δυνατή η απεικόνιση περισσότερων από ένα DataSets.

Μία πολύ απλή γραφική αναπαράσταση κάνοντας τα παραπάνω βήματα μπορεί να παρουσιαστεί τώρα στην οθόνη απλά με χρήση της εντολής: "chart.invalidate()" .

Αλληλεπίδραση με τα διαγράμματα

Η βιβλιοθήκη επιτρέπει την αλληλεπίδραση του χρήστη με το διάγραμμα χρησιμοποιώντας τις βασικές κινήσεις του λογισμικού Android όπως το τσίμπημα για μεγέθυνση και σμίκρυνση (zoom in και zoom out) στο διάγραμμα, το σύρσιμο για τη μετακίνηση της εστίασης σε άλλο σημείο καθώς και το απλό άγγιγμα το οποίο επιτρέπει να τονίζονται οι επιλεγμένες τιμές στο διάγραμμα (highlights).

Επίσης, η κλάση MarkerView επιτρέπει τη δημιουργία ενός pop-up παραθύρου που περιέχει την τιμή ενός στοιχείου κατά την επιλογή του στο διάγραμμα από τον χρήστη. Στο συγκεκριμένο κομμάτι δημιουργήσαμε δική μας κλάση MyMarkerView κάνοντας επέκταση της δοσμένης κλάσης MarkerView και τροποποιήσαμε τις μεθόδους τις κατάλληλα ώστε να εμφανίζεται το παράθυρο με τη μορφή που επιθυμούμε, περιέχοντας το μήνυμα που επιλέξαμε. Η κατασκευή του παραθύρου, τα χρώματα του background και το μέγεθος και χρώμα του κειμένου που περιέχει ορίζεται σε αρχείο .xml στα layouts της εφαρμογής.

Προσαρμογή αξόνων

Φυσικά, σημαντικές αλλαγές είναι δυνατό να γίνουν και στους άξονες. Η ορατότητα

τους, το χρώμα, το πάχος, το μέγιστο και το ελάχιστο και αρκετά άλλα χαρακτηριστικά τους μπορούν να ρυθμιστούν μέσα από μεθόδους της κλάσης του κάθε άξονα. Οι τιμές της κλίμακας του κάθε άξονα καθώς και το πως αυτές είναι ορατές στον χρήστη ρυθμίζονται επίσης.

Σημαντικό στοιχείο είναι η προσαρμογή του άξονα του χρόνου ώστε να δείχνει το κατάλληλο κείμενο. Η βιβλιοθήκη λειτουργεί σωστά μόνο όταν και οι δύο άξονες έχουν ως τιμές αριθμούς. Στις περιπτώσεις που κάποιο μέγεθος δεν μετριέται με ακέραιους η δεκαδικούς γίνεται χρήση της κλάσης IAxisValueFormatter και της μεθόδου getFormattedValue. Τα δεδομένα εισάγονται στη λίστα με τα Entries με ενδεικτικές τιμές ακεραίους και στη συνέχεια αντικαθίστανται οι τιμές στον άξονα με δοσμένο κείμενο από τον χρήστη. Στη δική μας περίπτωση το κείμενο αυτό μπορεί να είναι ώρα, διάστημα μεταξύ δύο ωρών μέσα στη μέρα, το όνομα ενός μήνα, μια ημερομηνία ή το διάστημα μίας εβδομάδας.

Προσαρμογή γραφήματος και Styling και Animation

Η βιβλιοθήκη δίνει παραπάνω επιλογές για την τροποποίηση του γραφήματος ώστε να δείχνει όμορφο στον χρήστη. Αυτές αφορούν στο background, στην τοποθέτηση κειμένου πάνω στο γράφημα, στα χρώματα, στην καμπυλότητα των γραμμών, στο μέγεθος του κάθε στοιχείου που απεικονίζεται, στις σκιές και γενικά σε κάθε τμήμα του κάθε τύπου γραφήματος, δίνοντας πλήθος καλαίσθητων επιλογών και δυνατοτήτων.

Τέλος, δίνεται η δυνατότητα animation στις γραφικές παραστάσεις. Παρόλο που αυτό θα είχε περισσότερο νόημα σε μια realtime εφαρμογή, είναι χρήσιμο και στη δική μας περίπτωση αφού η κατασκευή τους γίνεται με τον επιλεγμένο αλγόριθμο animation στον κατάλληλο χρόνο δίνοντας κίνηση στην εφαρμογή και συνεισφέροντας στη δημιουργία ενός πλήρους οπτικού αποτελέσματος. Το animation ενεργοποιείται με χρήση της εντολής “chart.animateX” (“chart.animateY” ή “chart.animateXY”) αντί της “chart.invalidate”. Η μέθοδος αυτή παίρνει ως ορίσματα τον χρόνο ο οποίος θα χρειαστεί για να ολοκληρωθεί το animation κατά τη δημιουργία της γραφικής παράστασης και τον αλγόριθμο σύμφωνα με τον οποίο θα γίνει το animation (easing functions).

Αναλυτικότερα στοιχεία για τη χρήση της βιβλιοθήκης περιέχονται στο documentation σύνδεσμος για το οποίο αναγράφεται στη βιβλιογραφία.

5.4 Fragments

Το Android δίνει τη δυνατότητα χρήσης των Fragments[27] για τις περιπτώσεις που η δράση του χρήστη πάνω σε μια οθόνη επιθυμούμε να προκαλεί αλλαγή σε ένα μέρος της οθόνης και όχι να ξαναφορτώνεται η οθόνη από την αρχή. Για το λόγο αυτό είναι δυνατή και η δημιουργία μιας οθόνης μέσα στην οποία βρίσκεται ένα Fragment το οποίο μπορεί να πάρει πολλαπλές μορφές (δυναμικά) αντί της δημιουργίας πολλών διαφορετικών οθονών που μεγάλο τους κομμάτι είναι κοινό.

Για κάθε Fragment δημιουργείται το αντίστοιχο layout το οποίο περιέχει όλα τα στοιχεία που θέλουμε να περιέχει. Είναι προφανές ότι το κάθε Fragment θα περιέχει διαφορετικά στοιχεία διαφορετικά δεν υπάρχει ιδιαίτερο νόημα στη χρήση τους. Επίσης για κάθε Fragment δημιουργείται η αντίστοιχη κλάση η οποία επεκτείνει την

κλάση Fragment και περιέχει όλο τον κώδικα που τρέχει κατά την δημιουργία του fragment στην οθόνη και την αλληλεπίδραση με αυτό. Πρέπει να σημειωθεί ότι τα στοιχεία του fragment δεν είναι ορατά απευθείας στο activity μέσα στο οποίο βρίσκεται. Οι επικοινωνία των δύο γίνεται με τη δημιουργία ενός interface μέσα στο fragment το οποίο υλοποιεί το activity στο οποίο περιέχεται το fragment. Αυτός ο δίαυλος επικοινωνίας δημιουργείται με σκοπό το activity να μπορεί να διαβάσει πληροφορία που εισάγεται από τον χρήστη μέσα στο fragment και το ανάποδο.

Στην περίπτωση μας έγινε χρήση fragments σε δύο σημεία της εφαρμογής: στην οθόνη με τα στοιχεία του λογαριασμού του χρήστη και στην οθόνη στην οποία εμφανίζεται η λίστα με όλες της χρονοσειρές. Στο δεύτερο σημείο, έχει κατασκευαστεί ένα fragment για κάθε τύπο χρονικού διαστήματος που ενδιαφέρει τον χρήστη (για παράδειγμα ετήσιο, μηνιαίο, εβδομαδιαίο, ημερήσιο) και το καθένα περιέχει τα κατάλληλα Spinners ώστε αυτός να επιλέξει τη συγκεκριμένη χρονική στιγμή που επιθυμεί. Η λειτουργικότητα περιγράφεται αναλυτικά στο επόμενο κεφάλαιο.

5.5 Δομές Δεδομένων

Τα δεδομένα τα οποία εισάγουμε στα διαγράμματα είναι λογικό να θέλουμε να είναι στην κατάλληλη μορφή ώστε να είναι προσβάσιμα και εύκολα διαχειρίσιμα. Για το λόγο αυτό δημιουργήθηκε η κλάση My_Data η οποία περιέχει δύο πεδία. Το πρώτο (value) είναι η τιμή του μεγέθους στο οποίο αφορά το δεδομένο (και είναι αριθμός κινητής υποδιαστολής float). Το δεύτερο είναι το πεδίο που δείχνει την χρονική στιγμή στην οποία παρουσιάζεται η προαναφερθείσα τιμή του μετρούμενου μεγέθους (και είναι συμβολοσειρά String).

Κάθε δεδομένο είναι αντικείμενο της κλάσης My_Data ενώ το σύνολο των δεδομένων που χρησιμοποιούνται για τη δημιουργία μιας γραφικής παράστασης αποθηκεύεται σε πίνακα My_Data [] . Με τον τρόπο αυτό κάθε set δεδομένων είναι προσπελάσιμο με γρήγορο και εύκολο τρόπο και εισάγεται στην λίστα με τα Entries του γραφήματος ώστε να παρουσιαστεί . Μας είναι ιδιαίτερα χρήσιμη η διατήρηση των δεδομένων και όχι απλά η προβολή τους καθώς αυτά μεταφέρονται από το ένα activity στο άλλο σε ορισμένες περιπτώσεις. Ο τρόπος που γίνεται αυτό βασίζεται στην διεπαφή (interface) Parcelable [20,21] και εξηγείται σε επόμενη παράγραφο.

Κλάση My_Chart

Σε πολλά σημεία της εφαρμογής χρησιμοποιείται η γραφική παράσταση τύπου LineChart της βιβλιοθήκης MPAndroid Chart. Για να μην γίνεται επανάληψη του κώδικα δημιουργίας του συγκεκριμένου τύπου γραφήματος θεωρήθηκε εύλογο να κατασκευαστεί μια κλάση η οποία θα περιέχει όλα τα χρήσιμα στοιχεία που αφορούν ένα γράφημα και τις μεθόδους για την προβολή του στην οθόνη. Η λογική της σχεδίασης είναι ίδια σε κάθε περίπτωση. Αυτά που αλλάζουν κάθε φορά είναι τα δεδομένα που απεικονίζονται και ο τύπος τους, το χρώμα της γραφικής παράστασης, και το όνομα της χρονοσειράς και περνιούνται ως παράμετροι κατά τη δημιουργία του αντικειμένου της κλάσης My_Chart. Το δημιουργημένο αντικείμενο περιέχει όλες τις χρήσιμες πληροφορίες που χρειάζεται και τις διατηρεί ώστε να δίνεται η δυνατότητα επιπλέον λειτουργιών (όπως για παράδειγμα η προβολή της τιμής επιλεγμένου

δεδομένου πάνω στη γραφική παράσταση με χρήση των Markers). Η κλάση My_Chart υλοποιεί επίσης το interface "Parcelable".

Για τις περιπτώσεις που στο ίδιο γράφημα απεικονίζονται δύο διαφορετικά μεγέθη (η παραπάνω από δύο) κατασκευάστηκε αντίστοιχη κλάση με την ίδια σχεδόν λογική, εκτός από μια μικρή τροποποίηση στα DataSets που είναι περισσότερα από ένα και εισάγονται με έναν λίγο διαφορετικό τρόπο στο γράφημα.

Κλάση Database_Class

Η συγκεκριμένη κλάση περιέχει όλες τις μεθόδους που προετοιμάζουν τα δεδομένα για τα request [31] στο API ώστε να αποκτηθούν πληροφορίες που είναι απαραίτητες στη mobile εφαρμογή και να διαμορφωθούν κατάλληλα ώστε να μπορούν να χρησιμοποιηθούν από αυτή. Το αντικείμενο της κλάσης που δημιουργείται κατά την είσοδο του χρήστη στο σύστημα φέρει όλες τις απαραίτητες πληροφορίες για αυτόν οι οποίες είναι διαθέσιμες σε κάθε activity της εφαρμογής με την κλήση των μεθόδων "getters" για κάθε πεδίο.

Οι πληροφορίες αυτές είναι χρήσιμες σε πρώτη φάση στη διαδικασία των requests. Ο τύπος του χρήστη, το όνομα χρήστη και ο χώρος εργασίας είναι σημαντικές πληροφορίες ώστε να αποκτηθούν τα δεδομένα μόνο για τις χρονοσειρές που αφορούν τον χρήστη που είναι συνδεδεμένος αυτή τη τη στιγμή στην εφαρμογή. Επίσης, το αντικείμενο της κλάσης χρησιμοποιείται στην οθόνη των στοιχείων λογαριασμού του χρήστη καθώς εκεί εμφανίζονται όλες οι πληροφορίες του. Είναι προφανές ότι η παρουσία του αντικειμένου σε παραπάνω από ένα activities απαιτεί την δυνατότητα μεταφοράς του και για τον λόγο αυτό η κλάση υλοποιεί το interface "Parcelable".

Με λίγα λόγια οποιαδήποτε αλληλεπίδραση με τη βάση δεδομένων απαιτείται, γίνεται μέσω του αντικειμένου της κλάσης Database_Class. Σε άλλο κεφάλαιο αναλύεται η διαδικασία εκτέλεσης του request. Αυτό που αρκεί να γνωρίζουμε προς το παρόν είναι ότι το API επιστρέφει σε μορφή JSON τα δεδομένα τα οποία διαβάζονται μέσω κατάλληλης μεθόδου του αντικειμένου της κλάσης Database_Class και εξάγεται όλη η πληροφορία που ζητήθηκε.

Κλάση My_Datetime

Η κλάση My_Datetime είναι το εργαλείο που διαχειρίζεται όλα τα θέματα που αφορούν ημερομηνίες και ώρες. Περιέχει μεθόδους που μετατρέπουν ημερομηνίες και ώρες στο κατάλληλο format ή σε συμβολοσειρές (και το αντίστροφο), μεθόδους για τον υπολογισμό των χρονικών διαστημάτων που φορτώνονται στα spinners ανάλογα με την επιλογή του χρήστη για τον τύπο των χρονικών δεδομένων που τον αφορούν, μεθόδους για την εύρεση της παρούσας χρονικής στιγμής και πολλές άλλες που εκτελούν μικρές εργασίες απαιτούμενες σε διάφορα σημεία της εφαρμογής. Το κύριο τμήμα της κλάσης χρησιμοποιεί τη βιβλιοθήκη "JodaTime" για τις βασικές λειτουργίες και τον υπολογισμό ημερομηνιών.

Θα πρέπει να τονίσουμε ότι στην κλάση εμπεριέχονται τέσσερα static πεδία τα οποία δηλώνουν την χρονική στιγμή έναρξης των δεδομένων από την οποία γίνεται και ο διαχωρισμός και η μέτρηση των χρονικών διαστημάτων.

5.6 Επικοινωνία Μεταξύ των Activities

Δεν θα ήταν λειτουργική η εφαρμογή αν το κάθε activity λειτουργούσε ατομικά χωρίς να απαιτεί κάποιας μορφής πληροφορία από κάποιο άλλο. Ουσιαστικά, η τεχνολογία android δίνει την εντύπωση στον χρήστη ότι το περιβάλλον της είναι πλήρως δυναμικό και βασίζεται στην αλληλεπίδραση με τον χρήστη. Πολλές φορές η επιλογή ενός στοιχείου σε ένα application οδηγεί σε ένα νέο activity το οποίο μπορεί να προβάλει αναλυτικά την πληροφορία που αφορά στο επιλεγμένο στοιχείο. Ως εκ τούτου δεν κρίνεται απαραίτητη μόνο η δυνατότητα δημιουργίας ενός activity μέσα από ένα άλλο αλλά και η μεταφορά δεδομένων μεταξύ τους. (κάποιες φορές μάλιστα είναι επιθυμητό η επικοινωνία να είναι αμφίδρομη)

Το λειτουργικό Android επιτρέπει την δημιουργία ενός νέου activity με τις εντολές “Intent myIntent = new Intent(CurrentActivity.this, New_Activity.class);” και “CurrentActivity.this.startActivity(myIntent);”. [27] Με τον τρόπο αυτό μία νέα οθόνη αντικαθιστά την παρούσα. Το λειτουργικό έχει δικό του τρόπο να διαχειρίζεται τα activities που έχουν αντικατασταθεί με χρήση ειδικής μνήμης. Ως εκ τούτου με τη δημιουργία ενός νέου activity δεν καταστρέφεται το παλαιό (killed activity) αλλά φορτώνεται στην κατάλληλη μνήμη ώστε ο χρήστης να μπορεί να επιστρέψει σε αυτή με το κουμπί της οπισθοχώρησης. Εκτός αν επέμβει ο ίδιος ο προγραμματιστής το λειτουργικό διατηρεί τα activities έως ότου θεωρήσει αυτό απαραίτητο και θεωρείται σωστό προγραμματιστικά να μην γίνονται συχνά επεμβάσεις στην διαχείριση αυτή με εξαίρεση τις περιπτώσει όπου δημιουργείται πρόβλημα στον σχεδιασμό και δεν υπάρχει άλλη επιλογή.

Η εντολή “myIntent.putExtra("key", text);” του activity-δημιουργού (όπου “key” είναι το αναγνωριστικό κλειδί του μηνύματος και text το μήνυμα) επιτρέπει την αποστολή μηνύματος στο activity που δημιουργεί, το οποίο το παραλαμβάνει ένα πακέτο με όλα τα μηνύματα που έχουν σταλεί (καθώς μπορεί να είναι περισσότερα από ένα) με την εντολή “Bundle bundle= getIntent().getExtras();” και κάνει ανάγνωση του μηνύματος με την εντολή “String message = bundle.getString("key");” (δηλαδή με τη χρήση του αναγνωριστικού κλειδιού).

Η αποστολή μηνυμάτων που περιγράφεται παραπάνω είναι δυνατή μόνο με συγκεκριμένους τύπους : Strings, primitives, Serializable, Parcelable. Για τον λόγο αυτό κάθε μήνυμα το οποίο δεν ανήκει στα παραπάνω θα πρέπει να μετατραπεί με κάποιο τρόπο σε έναν από τους συμβατούς τύπους.

Το Interface “Parcelable” είναι μια Android υλοποίηση του Java Serializable η οποία επιτρέπει την εγγραφή αντικειμένων σε πακέτο και την επανάκτηση τους ώστε να μπορέσουν να περαστούν ως μήνυμα μεταξύ των δύο activities. Λόγω συγκεκριμένης δομής και τρόπου επεξεργασίας, ένα Parcelable μπορεί να επεξεργαστεί σχετικά γρήγορα, σε σύγκριση με το τυποποιημένο Java Serializable. Οι κλάσεις οι οποίες υλοποιούν την διεπαφή Parcelable έχουν την δυνατότητα μέσω των υλοποιημένων μεθόδων του interface να διαβάσουν τα στοιχεία του αντικειμένου της κλάσης και να τα εισάγουν στο πακέτο. Η ανάκτηση των στοιχείων αυτών γίνεται με άλλες υλοποιημένες μεθόδους της διεπαφής οι οποίες εξάγουν τις πληροφορίες από το πακέτο και επαναδημιουργούν το αντικείμενο στο νέο Activity.

Όπως προαναφέρθηκε οι κλάσεις My_Data, My_Chart, Database_Class υλοποιούν το Interface Parcelable καθώς η μεταφορά τον αντικειμένων τους από ένα activity σε ένα

άλλο θεωρείται απαραίτητη για τη σωστή λειτουργία της εφαρμογής.

5.7 Βασικό Μενού

Με σκοπό την αποσυμφόρηση της οθόνης από μεγάλο πλήθος κουμπιών θεωρήθηκε σκόπιμη η δημιουργία ενός menu στο πάνω και δεξιά μέρος της οθόνης (μέσα στο Action Bar) με επιλογές για μετάβαση στην οθόνη στοιχείων του λογαριασμού χρήστη, στην οθόνη γενικών πληροφοριών της εφαρμογής, στην κεντρική οθόνη και επιλογή εξόδου από την εφαρμογή, δηλαδή αποσύνδεση του χρήστη.

Για μεγαλύτερη ευκολία στην περιήγηση το menu αυτό τοποθετήθηκε σε παραπάνω από μία οθόνες. Για να μην υπάρχει επανάληψη του κώδικα για κάθε χρήση του ίδιου menu δημιουργήθηκε η Base_Activity με το δικό της layout που περιέχει όλα τα στοιχεία που αφορούν στο menu και τις λειτουργίες του. Κάθε activity που χρησιμοποιεί το menu είναι υποκλάση της Base_Activity κληρονομώντας όλα τα χαρακτηριστικά της και στο εκάστοτε layout συμπεριλαμβάνεται (include) το layout της Base_Activity προκειμένου να προβληθεί το menu στην οθόνη.

Μια συγκεκριμένη προγραμματιστική επιλογή που αφορά στο menu ήταν η επιλογή για ανακατεύθυνση στο homescreen. Θα πρέπει να τονίσουμε, ότι δηλώθηκε ρητά στον κώδικα πως όταν ο χρήστης επιλέξει την μετάβαση στην αρχική οθόνη, δεν μπορεί να επιστρέψει με το κουμπί τη οπισθοχώρησης στο προηγούμενο activity καθώς αυτό δημιουργούσε προβλήματα στον σχεδιασμό και στη λειτουργία της εφαρμογής. Έτσι όταν ο χρήστης βρίσκεται στο homescreen και πατήσει το κουμπί της οπισθοχώρησης ανεξάρτητα από πιο activity προήλθε, βγαίνει από την εφαρμογή. [28]

5.8 Σχεδίαση οθονών

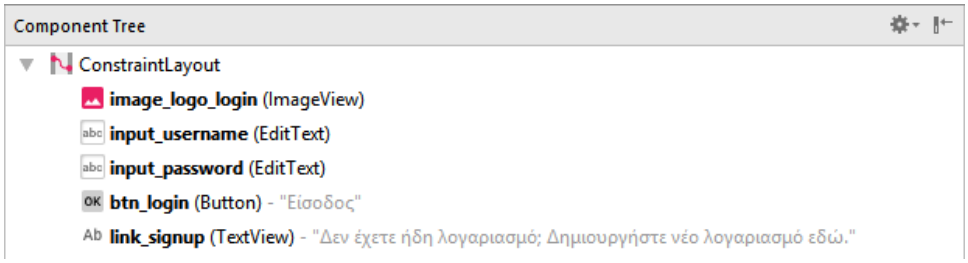
Όταν η εφαρμογή είναι πλέον λειτουργική τελευταίο σημαντικό κομμάτι είναι η προσαρμογή όλων των γραφικών τις τμημάτων τα οποία δεν έχουν κατασκευαστεί από τον ίδιο τον προγραμματιστή ώστε το αποτέλεσμα να είναι καλαίσθητο και φιλικό προς τον χρήστη.

5.8.1 Τοποθέτηση στοιχείων στον χώρο της οθόνης

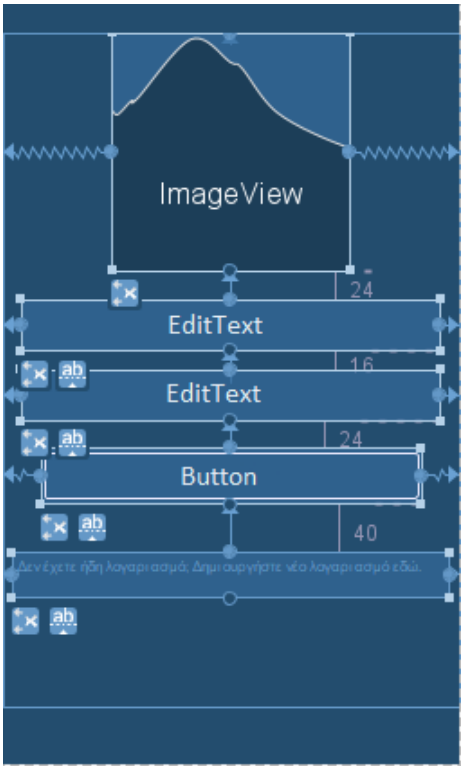
Η διαδικασία αυτή ξεκινάει με την τοποθέτηση όλου του περιεχομένου στην οθόνη σε σημείο κατάλληλο ώστε να είναι εύκολα ορατό στον χρήστη και, εφόσον πρόκειται για στοιχείο αλληλεπίδρασης, εύκολα προσβάσιμο. Αυτό το σημείο παρά την φαινομενική του απλότητα μπορεί να μειώσει σε σημαντικό βαθμό τον χρόνο εξυπηρέτησης του χρήστη από την εφαρμογή και πολλές φορές περιπλέκει τον προγραμματισμό. Για το λόγο αυτό πρέπει να μελετηθεί και κατά τον βασικό σχεδιασμό του application αλλά και για κάθε προσθήκη νέας λειτουργίας.

Στην εν λόγω εφαρμογή, το homescreen προσφέρει τις βασικές πληροφορίες που αφορούν τον χρήστη που πολλές φορές είναι αρκετές ώστε να μην χρειαστεί να ψάξει παραπάνω στοιχεία. Ακόμα και στην περίπτωση που επιθυμεί να αναζητήσει περισσότερες πληροφορίες, οι κινήσεις που απαιτούνται για την εύρεση του έχουν μειωθεί στο ελάχιστο με την τοποθέτηση των στοιχείων σε ορθή θέση και με την προσθήκη λειτουργιών που θα περιγραφούν στο επόμενο κεφάλαιο.

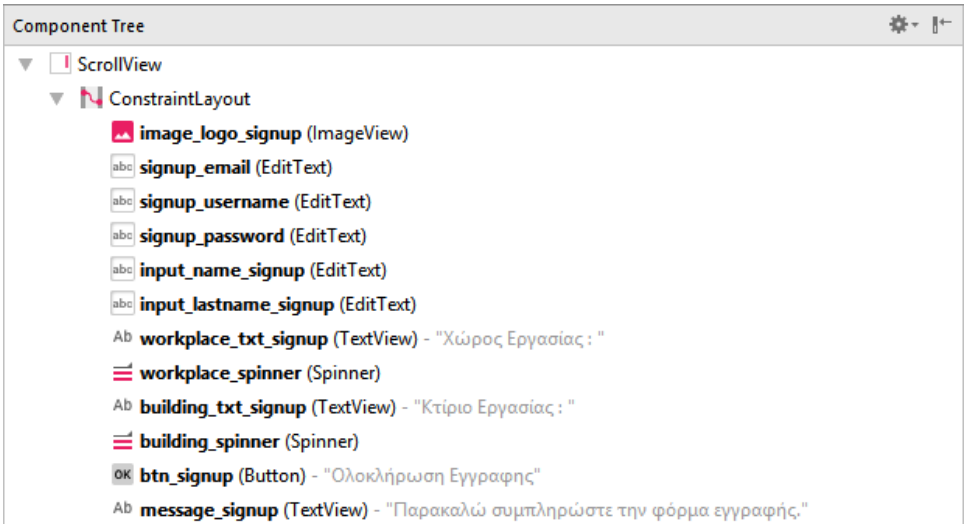
Παρακάτω παρουσιάζονται τα blueprints όλων των οθονών της εφαρμογής που δείχνουν την τοποθέτηση των στοιχείων στην οθόνη και τους δεσμούς μεταξύ αυτών κατά το Constraint Layout. Επίσης, παρουσιάζεται το "component tree" κάθε οθόνης το οποίο είναι η παρουσίαση της διάταξης των στοιχείων μέσα στην οθόνη σε όλα τα επίπεδα.



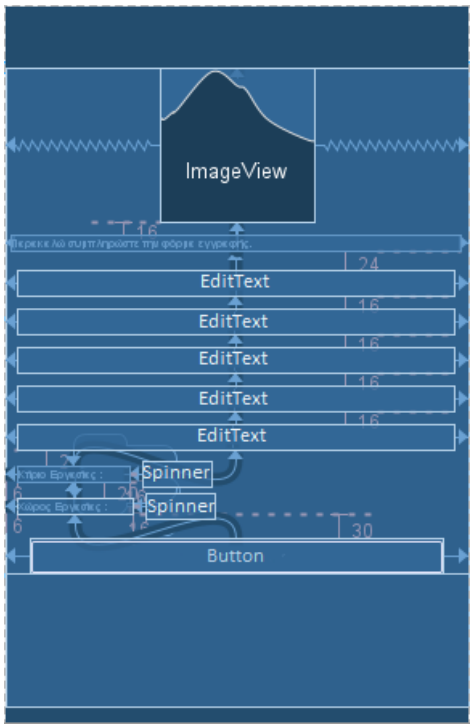
Εικόνα 5.1- Component Tree της Οθόνης Εισόδου (Log-In Screen)



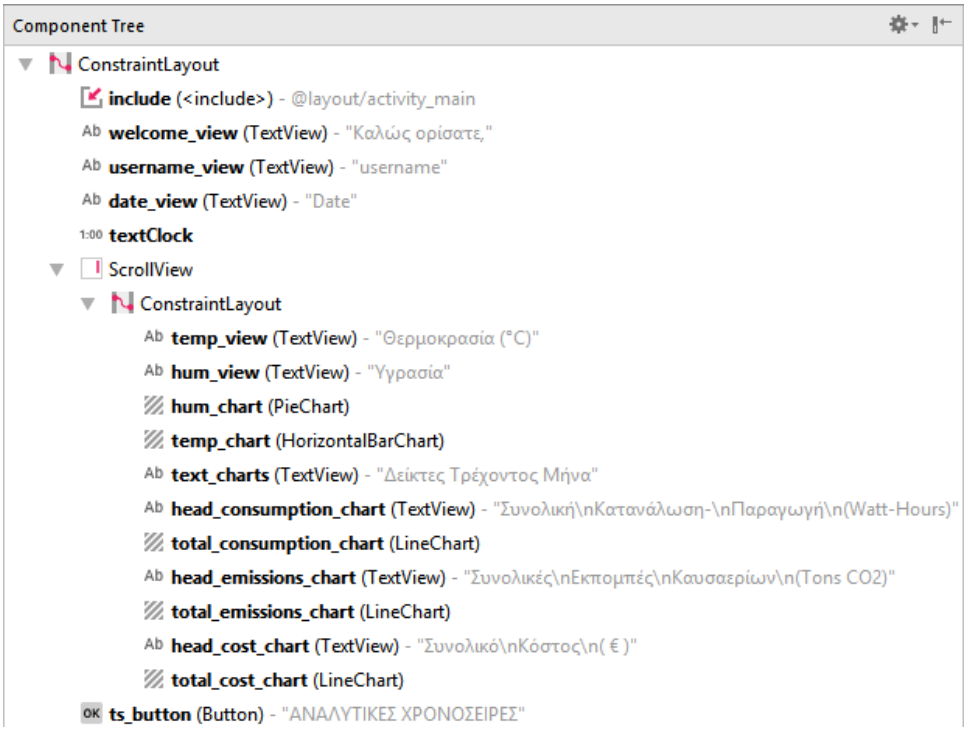
Εικόνα 5.2- Blueprint Οθόνης Εισόδου (Log-In Screen)



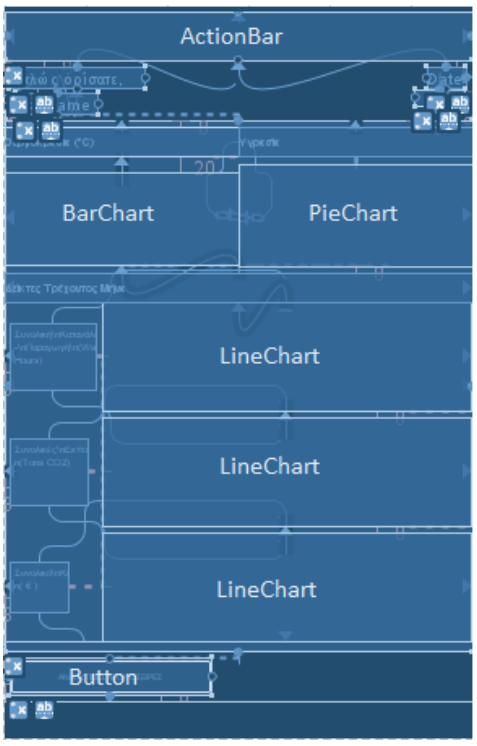
Εικόνα 5.3- Component Tree Οθόνης Εγγραφής Χρήστη (Sign-Up Screen)



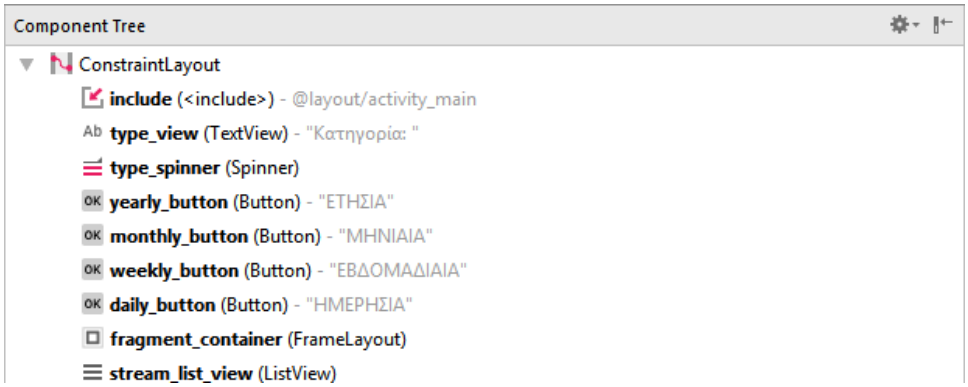
Εικόνα 5.4- Blueprint Οθόνης Εγγραφής Χρήστη (Sign-Up Screen)



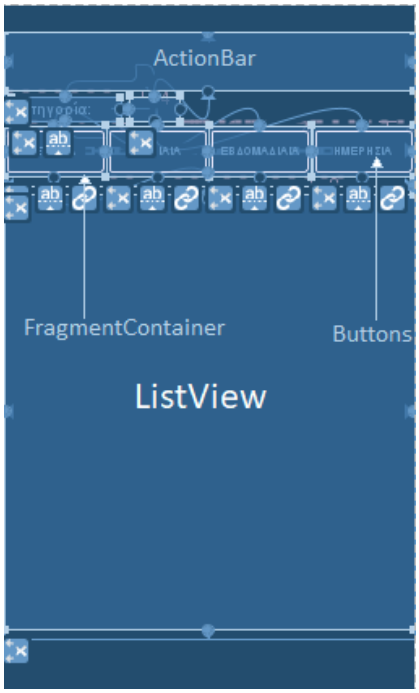
Εικόνα 5.5- Component Tree Κεντρικής Οθόνης (Homescreen)



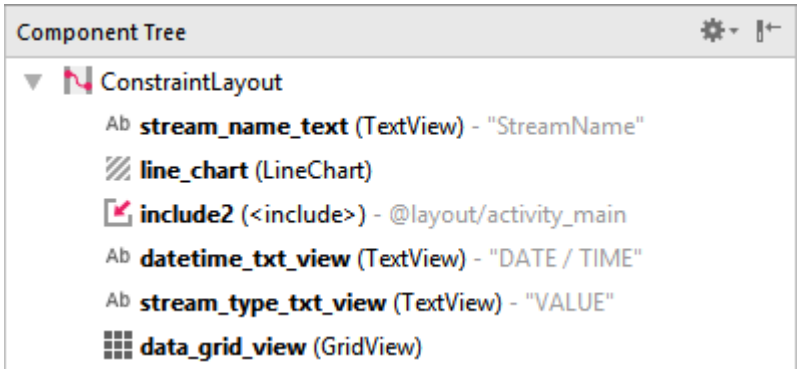
Εικόνα 5.6- Blueprint Κεντρικής Οθόνης (Homescreen)



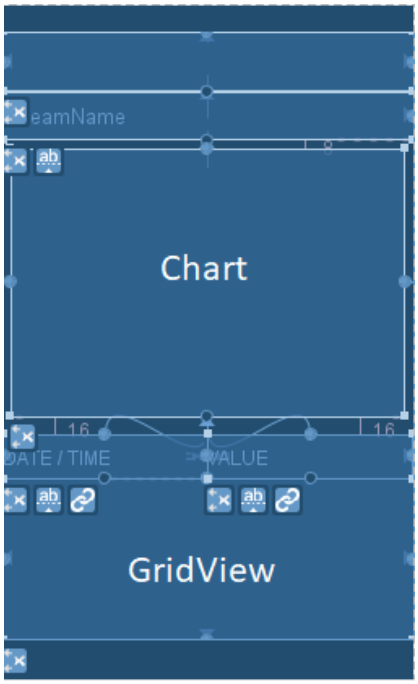
Εικόνα 5.7- Component Tree Οθόνης Λίστας Χρονοσειρών (Stream List Screen)



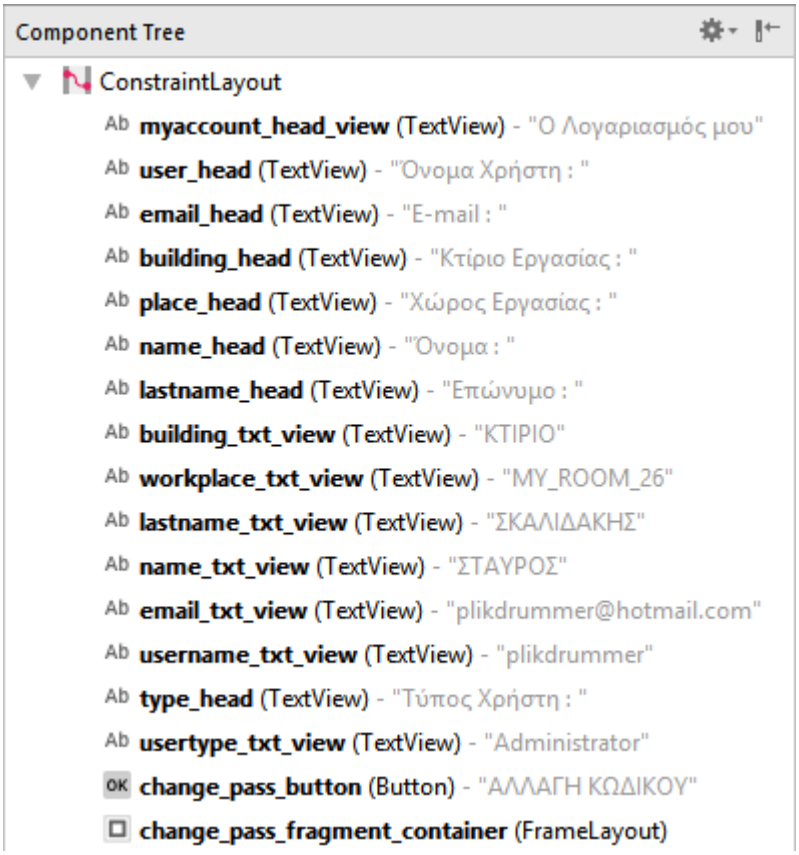
Εικόνα 5.8- Blueprint Οθόνης Λίστας Χρονοσειρών (Stream List Screen)



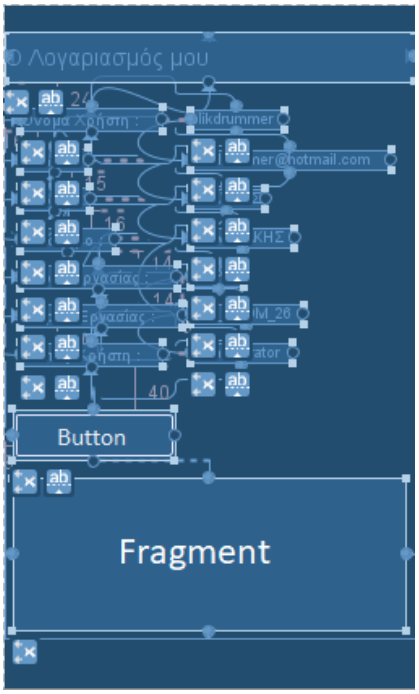
Εικόνα 5.9- Component Tree Οθόνης Αναλυτικής Παρουσίασης Χρονοσειράς (Detailed Stream Screen)



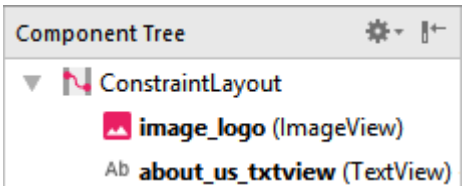
Εικόνα 5.10- Blueprint Οθόνης Αναλυτικής Παρουσίασης Χρονοσειράς (Detailed Stream Screen)



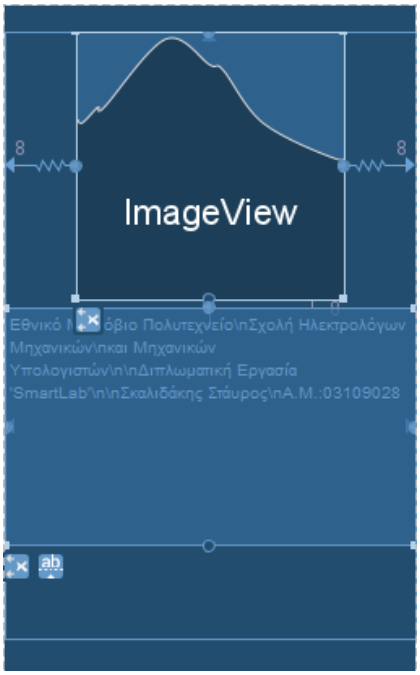
Εικόνα 5.11- Component Tree Οθόνης Πληροφοριών Λογαριασμού (My Account Screen)



Εικόνα 5.12- Blueprint Οθόνης Πληροφοριών Λογαριασμού (My Account Screen)



Εικόνα 5.13- Component Tree Οθόνης Πληροφοριών Σχετικά με την Εφαρμογή (About Us Screen)



Εικόνα 5.14- Blueprint Οθόνης Πληροφοριών Σχετικά με την Εφαρμογή (About Us Screen)

5.8.2 Background και Επιλογή Χρωμάτων

Χρησιμοποιήθηκε εικόνα μπλε απόχρωσης η οποία τίθεται ως background σε κάθε activity με την προσθήκη της κατάλληλης εντολής στο .xml αρχείο που περιγράφει το layout της. Η επιλογή βασίστηκε στο χρώμα του εικονιδίου της εφαρμογής το οποίο κατασκευάστηκε σε online εφαρμογή γραφιστικής με όνομα “DesignApp.io”.

Η επιλογή χρωμάτων για τις γραφικές παραστάσεις των ενεργειακών δεδομένων αποτελεί σημαντικό κομμάτι του οπτικού αποτελέσματος της κάθε οθόνης. Έγινε κατάλληλη επιλογή χρωμάτων ώστε, όσο είναι δυνατόν, αυτά να παραπέμπουν στο ενεργειακό μέγεθος το οποίο απεικονίζουν. Η επιλογή αυτή ωστόσο είχε αρχικά ανεπιθύμητα αποτελέσματα καθώς οι οθόνες παρουσίαζαν πολλά διαφορετικά και αταίριαστα χρώματα μεταξύ τους. Το πρόβλημα ξεπεράστηκε με χρήση συγκεκριμένων αποχρώσεων που ανήκουν σε υπάρχουσα χρωματική παλέτα. Με τον τρόπο αυτό διατηρήθηκε σε σημαντικό βαθμό η λογική συσχέτιση μεταξύ χρώματος και φύσης

ενεργειακού μεγέθους και επιτεύχθηκε ομοιομορφία και ισορροπία χρωμάτων σε κάθε οθόνη.

5.9 Progress Dialogue

Μία από της τελευταίες προσθήκες που έγινε στην εφαρμογή είναι το στοιχείο Progress Dialogue του Android [8] για τις περιπτώσεις που χρειάζεται χρόνος μέχρις ότου ο χρήστης δει κάποια αλλαγή στην οθόνη. Πιο συγκεκριμένα, σε πολλά σημεία της εφαρμογής είναι απαραίτητη η αίτηση πληροφοριών από τη βάση δεδομένων προκειμένου να υπάρχουν τα απαραίτητα στοιχεία για την προβολή της οθόνης. Τότε, δημιουργείται ένα διαφορετικό νήμα (thread) από το UI-Thread (interface thread) στο οποίο τρέχει ο κώδικας που απαιτείται για την απόκτηση της πληροφορίας από τη βάση. Στο UI-Thread εμφανίζεται μια μπάρα φόρτωσης (Progress Dialogue Box) η οποία βρίσκεται εκεί μέχρι να ολοκληρωθεί η διαδικασίας της αίτησης πληροφοριών. Έτσι ο χρήστης είναι ενήμερος ότι πρέπει να αναμένει μέχρι να δει το σωστό περιεχόμενο της οθόνης.

Στο σημείο αυτό πρέπει να τονιστεί ότι τα νήματα μεταξύ τους επικοινωνούν μέσω Handlers. Δηλαδή, η διακοπή της μπάρας φόρτωσης όταν η αίτηση πληροφοριών ολοκληρωθεί, η εμφάνιση στοιχείων στην οθόνη και γενικότερα όλες οι ενέργειες που αφορούν το UI-Thread αλλά εξαρτώνται από την εκτέλεση του νέου δημιουργημένου νήματος γίνονται με αποστολή μηνυμάτων από το ένα νήμα στο άλλο μέσω των Handlers τα οποία αποτελούν και τον μοναδικό τρόπο επικοινωνίας μεταξύ των νημάτων.

Κεφάλαιο 6ο - Παρουσίαση Εφαρμογής

Στο κεφάλαιο αυτό θα γίνει αναλυτική περιγραφή της λειτουργικότητας της εφαρμογής και θα δοθούν σενάρια χρήσης για κάθε οθόνη που περιλαμβάνεται σε αυτή. Θα περιγραφεί με λεπτομέρεια η διαδικασία της χρήσης του application με σκοπό την πληροφόρηση για κάποιο ενεργειακό δεδομένο καθώς και όλες οι υπόλοιπες λειτουργικότητες που συμβάλλουν σε μία ολοκληρωμένη εμπειρία από την πλευρά του χρήστη. Αρχικά, θα περιγράφονται οι οθόνες και το περιεχόμενο τους, θα εξηγείται η λειτουργία τους και οι λόγοι επιλογών του προγραμματιστή και θα αναλύεται σε βήματα κάθε κίνηση την οποία ο χρήστης θα μπορεί να εκτελέσει προκειμένου να επιτύχει το επιθυμητό αποτέλεσμα.

6.1 Οθόνη Log-In (Log-In Screen)

Η οθόνη του Log-in είναι το πρώτο activity του application που ξεκινάει κατά την έναρξη της εφαρμογής (Launching Activity) και σκοπός του είναι η συλλογή βασικών πληροφοριών από τον χρήστη προκειμένου να συνδεθεί στην εφαρμογή.

Στο πάνω μέρος της οθόνης έχει τοποθετηθεί το λογότυπο της εφαρμογής σε επιλεγμένες διαστάσεις (ImageView).Η οθόνη επίσης περιέχει δύο στοιχεία τύπου EditText, ένα για το όνομα χρήστη και ένα για τον κωδικό πρόσβασης, ένα κουμπί το οποίο όταν πατηθεί στέλνει τα εισαχθέντα στοιχεία για επιβεβαίωση και ένα TextView το οποίο παροτρύνει τον χρήστη να δημιουργήσει νέο λογαριασμό σε περίπτωση που δεν έχει ήδη. [34]

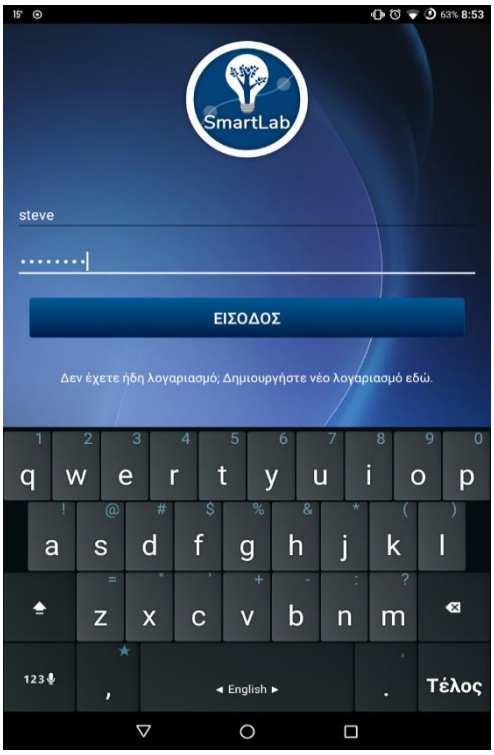
Όταν ο χρήστης πατήσει το εικονίδιο της εφαρμογής στην λίστα των εφαρμογών του η πρώτη οθόνη η οποία προβάλλεται είναι το Log-in screen.

Αν ο χρήστης έχει δημιουργήσει ήδη λογαριασμό στην εφαρμογή τότε εισάγει το όνομα χρήστη του (username) και τον κωδικό χρήστη (password). (Εικόνας 6.1) Στην περίπτωση που κάποιο από τα στοιχεία που εισάγει δεν έχει αποδεκτό μήκος, τότε το πάτημα του κουμπιού δεν στέλνει τα στοιχεία για επιβεβαίωση. Αντίθετα, εμφανίζεται κατάλληλο μήνυμα το οποίο ενημερώνει το χρήστη για το λάθος του.(Εικόνα 6.4) Αν τα εισαχθέντα στοιχεία τηρούν τις απαραίτητες προϋποθέσεις ώστε να θεωρηθούν αποδεκτά (δηλαδή οι δοσμένες συμβολοσειρές του ονόματος χρήστη και του κωδικού έχουν μήκος μεγαλύτερο του μηδενός) τότε το πάτημα του κουμπιού στέλνει τα στοιχεία στο API προκειμένου να γίνει η επιβεβαίωση τους. Κατά τη διάρκεια της επιβεβαίωσης εμφανίζεται μια μπάρα φόρτωσης και το αντίστοιχο μήνυμα.(Εικόνα 6.2) Αν αυτή είναι επιτυχής τότε γίνεται αίτηση στο API και επιστρέφονται στην εφαρμογή όλα τα δεδομένα που αφορούν τον χρήστη καθώς γίνεται είσοδος στην κεντρική σελίδα της εφαρμογής (homescreen). Αν είναι ανεπιτυχής τότε εμφανίζεται μήνυμα αποτυχίας και ο χρήστης θα πρέπει να αλλάξει τα στοιχεία που έχει εισάγει αφού αυτά δεν αντιστοιχούν σε χρήστη που είναι εγγεγραμμένος στο σύστημα (Εικόνα 6.2). Επίσης η είσοδος στην εφαρμογή δεν πραγματοποιείται αν η συσκευή δεν είναι συνδεδεμένη στο διαδίκτυο και εμφανίζεται ανάλογο μήνυμα (Εικόνα 6.2).

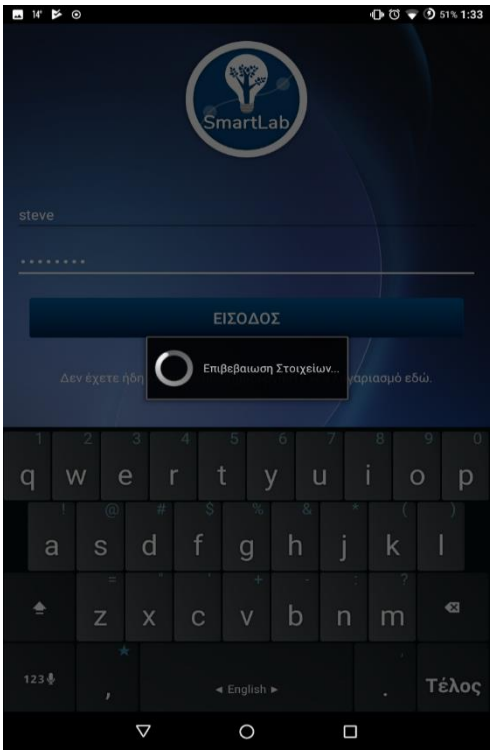
Αν ο χρήστης δεν έχει δημιουργήσει λογαριασμό στο παρελθόν τότε αρκεί να πατήσει πάνω στο TextView που τον παραπέμπει στην δημιουργία νέου λογαριασμού. Αυτό ξεκινάει ένα νέο activity. Πρόκειται για την οθόνη Sign-Up, η οποία περιγράφεται στην

επόμενη παράγραφο.

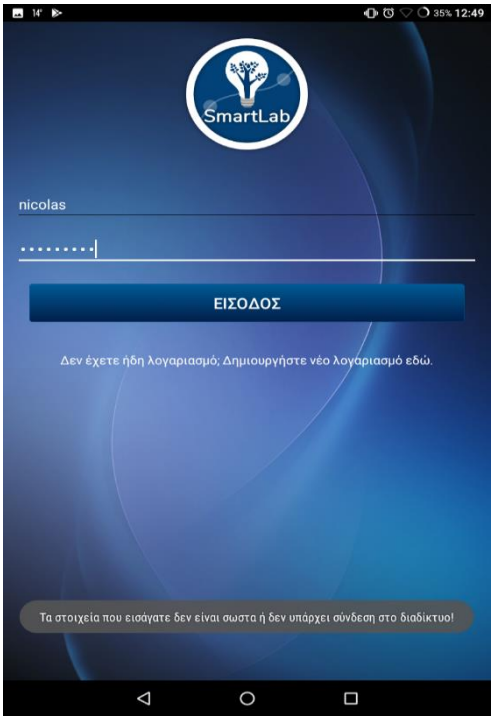
Παρακάτω εμφανίζονται στιγμιότυπα της οθόνης Log-In σε χρήση.



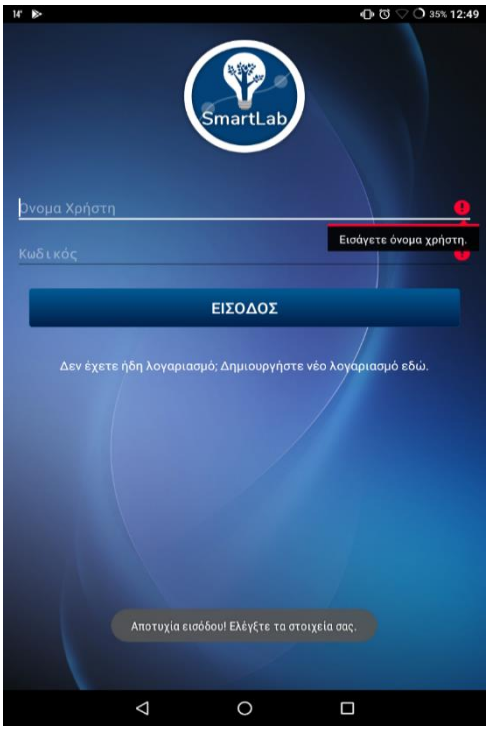
Εικόνα 6.1- Εισαγωγή Κωδικού



Εικόνα 6.2- Επιβεβαίωση Στοιχείων



Εικόνα 6.3- Λανθασμένος Συνδυασμός Username και Password



Εικόνα 6.4- Εισαγωγή Κενών Στοιχείων

6.2 Οθόνη Sign-Up (Sign-Up Screen)

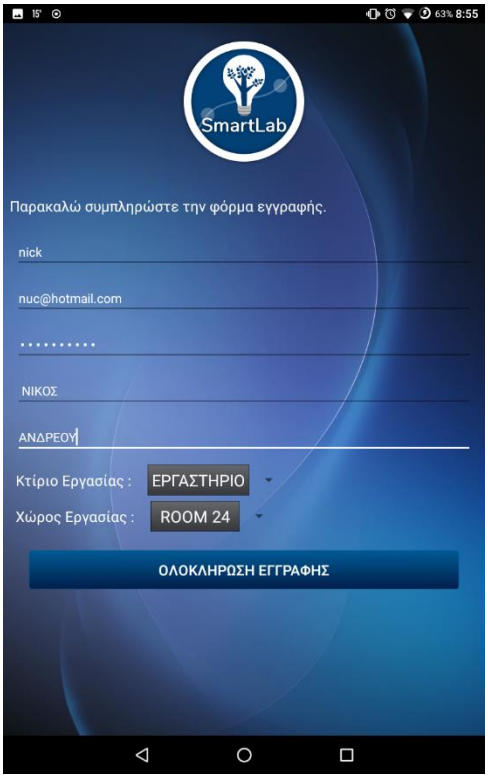
Στην οθόνη αυτή ο χρήστης οδηγείται όταν στην οθόνη του Log-In πατήσει πάνω στο TextView που τον παροτρύνει να δημιουργήσει καινούριο λογαριασμό. Πρόκειται για ξεχωριστό activity.

Στο πάνω μέρος της οθόνης έχει τοποθετηθεί επίσης το λογότυπο της εφαρμογής σε επιλεγμένες διαστάσεις (ImageView). Η οθόνη περιέχει ένα TextView που δίνει οδηγίες στον χρήστη για την εγγραφή του, πέντε στοιχεία τύπου EditText (όνομα χρήστη, E-mail, κωδικός, όνομα, επώνυμο) και ένα Spinner το οποίο φορτώνεται με όλους τους χώρους εργασίας που υπάρχουν. Στο κάτω μέρος της οθόνης έχει τοποθετηθεί κουμπί με το οποίο ολοκληρώνεται η διαδικασία της εγγραφής νέου χρήστη στο σύστημα.

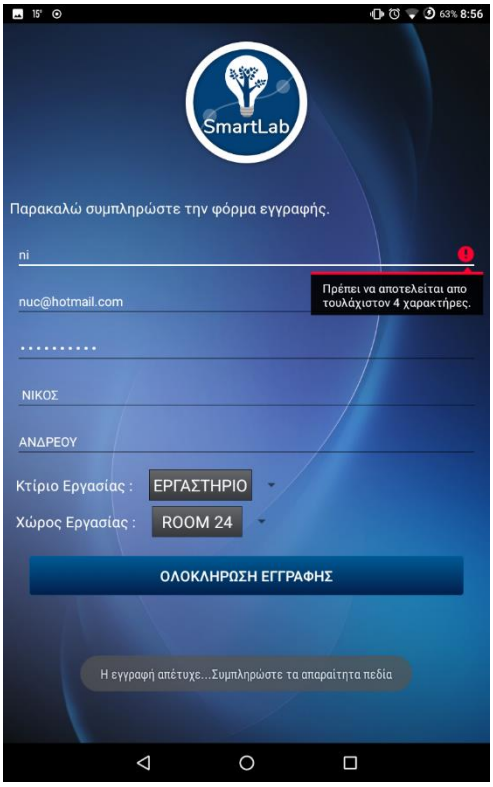
Ο χρήστης ο οποίος επιθυμεί να κάνει έναρξη του δικού του λογαριασμού στην εφαρμογή, συμπληρώνει τα πεδία της φόρμας που προβάλλεται στην οθόνη (Εικόνα 6.5). Με το πάτημα του κουμπιού ολοκλήρωσης εγγραφής, γίνεται ένας πρώτος έλεγχος προκειμένου τα εισαχθέντα στοιχεία να είναι έγκυρα. Αυτό ισχύει όταν το όνομα χρήστη (username) έχει μήκος μεταξύ 4-30 χαρακτήρων, ο κωδικός (password) έχει μήκος μεταξύ 4-10 χαρακτήρων, το όνομα και το επώνυμο (name, lastname) έχει μήκος μικρότερο από 30 χαρακτήρες και το email έχει τη φυσιολογική μορφή ηλεκτρονικής διεύθυνσης (για παράδειγμα user_new@provider.com). Σε αντίθετη περίπτωση, όταν ο χρήστης επιχειρήσει να πατήσει το κουμπί για την ολοκλήρωση της εγγραφής, η διαδικασία της εγγραφής δεν προχωράει στον έλεγχο των πληροφοριών στη βάση δεδομένων αλλά εμφανίζεται κατάλληλο μήνυμα δίπλα από το κάθε λανθασμένο πεδίο (Εικόνα 6.7).

Αν όλα τα στοιχεία που εισάγονται τηρούν τις παραπάνω προϋποθέσεις, τότε γίνεται έλεγχος στη βάση για το όνομα χρήστη. Αυτό θα πρέπει να είναι μοναδικό για τον κάθε χρήστη (αποτελεί όπως προαναφέρθηκε κλειδί της οντότητας "χρήστης") και για το λόγο αυτό δεν μπορεί να εμφανίζεται παραπάνω από μια φορές στη βάση δεδομένων. Αν το όνομα υπάρχει ήδη , τότε εμφανίζεται στην οθόνη το κατάλληλο μήνυμα σε μορφή Toast παροτρύνοντας τον χρήστη να εισάγει διαφορετική τιμή για το πεδίο (Εικόνα 6.8). Αν δεν υπάρχει, εμφανίζεται στην οθόνη μια μπάρα φόρτωσης με το μήνυμα "Δημιουργία Λογαριασμού", (Εικόνα 6.6) όσο γίνεται το request της εφαρμογής στο API για την εγγραφή του νέου χρήστη στο σύστημα. Όταν αυτή ολοκληρωθεί, τότε το activity τερματίζεται, ο χρήστης οδηγείται στην οθόνη Log-In και εμφανίζεται μήνυμα στην οθόνη που επιβεβαιώνει την δημιουργία του λογαριασμού και προτείνει στον χρήστη να εισάγει τα στοιχεία του για να συνδεθεί στο σύστημα.

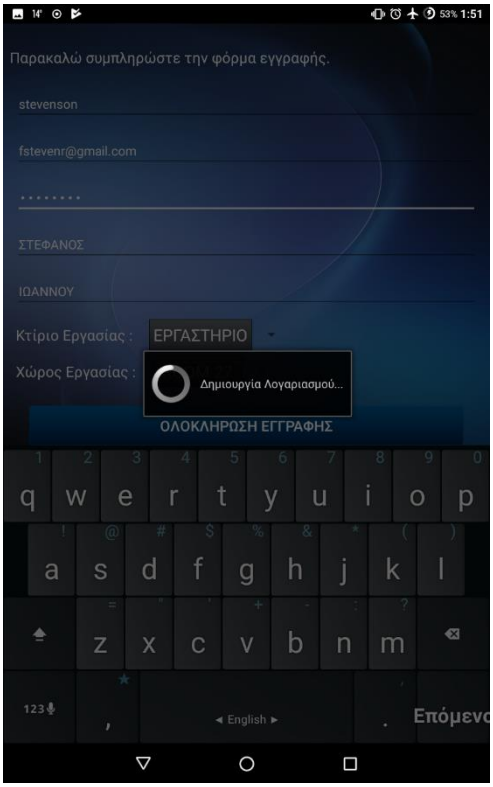
Παρακάτω εμφανίζονται στιγμιότυπα της οθόνης Sign-Up σε χρήση.



Εικόνα 6.5- Φόρμα Εγγραφής



Εικόνα 6.7- Μήνυμα σφάλματος εισαχθέντος στοιχείου



Εικόνα 6.6- Δημιουργία Νέου Λογαριασμού

Εικόνα 6.8- Αποτυχημένη ολοκλήρωση εγγραφής

6.3 Κεντρική Οθόνη (Homescreen)

Ο χρήστης οδηγείται στο Homescreen μετά από επιτυχημένο Log-In. Η οθόνη αυτή αποτελεί την κεντρική οθόνη της εφαρμογής και προβάλει βασικά ενεργειακά δεδομένα που αφορούν τον χρήστη. Κατά την είσοδο της οθόνης εκτελείται το Animation σε κάθε γραφική παράσταση που περιέχεται σε αυτή.

Στο ανώτερο μέρος της οθόνης υπάρχει ένα Action Bar. Στο αριστερό του μέρος αναγράφεται το όνομα της εφαρμογής και στο δεξί του υπάρχει εικονίδιο που όταν πατηθεί εμφανίζει menu τεσσάρων επιλογών. (Home, Ο λογαριασμός μου, Σχετικά με εμάς, Έξοδος). Κάτω από το Action Bar υπάρχει ένα TextView με κείμενο που καλωσορίζει τον χρήστη στην εφαρμογή και αναγράφει το username του. Επίσης, προβάλλεται η παρούσα ημερομηνία και ώρα. Σε κατώτερο σημείο της οθόνης, εμφανίζεται ένα Bar Chart με την εσωτερική και εξωτερική θερμοκρασία καθώς και ένα Pie Chart με την υγρασία. Ως εσωτερική θερμοκρασία εμφανίζεται πάντα η τελευταία τιμή της χρονοσειράς θερμοκρασίας που σχετίζεται με τον χώρο στον οποίο εργάζεται ο χρήστης (και έχει δηλωθεί κατά την εγγραφή του). Η εξωτερική θερμοκρασία και υγρασία προέρχονται από τις τελευταίες τιμές των αντίστοιχων χρονοσειρών. Κάθε στιγμή η ανώτερη από τις δύο θερμοκρασίες σχεδιάζεται με χρώμα κόκκινο και η άλλη με χρώμα μπλε. Η διάταξη των θερμοκρασιών, η μία δίπλα στην άλλη, σε συνδυασμό με τον χρωματισμό της υψηλότερης θερμοκρασίας διευκολύνουν το χρήστη να αποφανθεί για την απόκλιση μεταξύ των δύο και να δράσει ανάλογα με τις συνθήκες άνεσής του. Σε χαμηλότερο σημείο στην οθόνη, προβάλλονται τρεις δείκτες του τρέχοντος μήνα: η συνολική κατανάλωση και παραγωγή ενέργειας σε Watt-Hours, οι συνολικές εκπομπές καυσαερίων σε τόνους CO2 και το συνολικό κόστος της ενέργειας που καταναλώθηκε σε ευρώ. Στη δική μας περίπτωση όπως έχει τονιστεί προηγουμένως, το εργαστήριο δεν παράγει ενέργεια. Παρ' όλα αυτά, έχει δοθεί μία τυχαία αλληλουχία τιμών στην χρονοσειρά αυτή για χάρη της αναπαράστασης της λειτουργίας.

Στο κατώτερο σημείο της οθόνης έχει τοποθετηθεί κουμπί το οποίο μεταφέρει τον χρήστη στην οθόνη με την αναλυτική λίστα όλων των χρονοσειρών.

Ο χρήστης με την είσοδο στη σελίδα μπορεί να παρατηρήσει τη θερμοκρασία, την υγρασία και τους μηνιαίους δείκτες (Εικόνα 6.9). Μάλιστα στους τελευταίους μπορεί να αλληλεπιδράσει πάνω στα γραφήματα. Με ένα πάτημα πάνω σε σημείο που ανήκει σε ένα γράφημα εμφανίζεται (με χρήση Markers) ένα παράθυρο που αναγράφει τις πληροφορίες για το συγκεκριμένο δεδομένο (Εικόνα 6.10).

Επίσης, ο χρήστης μέσα από το Menu, (Εικόνα 6.11) μπορεί να ανανεώσει την οθόνη πιέζοντας την επιλογή "Home" (ουσιαστικά επαναφορτώνεται η οθόνη), να μεταβεί στην οθόνη ΄λογαριασμού του χρήστη πιέζοντας την επιλογή "Ο λογαριασμός μου", να μεταβεί στην οθόνη γενικών πληροφοριών της εφαρμογής επιλέγοντας το "Σχετικά με εμάς" ή να αποχωρήσει από την εφαρμογή με την επιλογή "Έξοδος", η οποία θα τον οδηγήσει στο Log-In Screen.

Τέλος, αν ο χρήστης επιθυμεί να αναζητήσει συγκεκριμένα ενεργειακά δεδομένα διαφόρων χρονοσειρών σε χρονικά διαστήματα της επιλογής του μπορεί να πιέσει το κουμπί "Αναλυτικές Χρονοσειρές" το οποίο θα τον οδηγήσει στην κατάλληλη οθόνη.

Παρακάτω παρουσιάζονται στιγμιότυπα της κεντρικής οθόνης σε χρήση.

Εικόνα 6.9- Κεντρική οθόνη

Εικόνα 6.10- Λειτουργικότητα γραφημάτων

Εικόνα 6.11- Επιλογή Menu

6.4 Οθόνη Λογαριασμού Χρήστη (MyAccount Screen)

Ο χρήστης μεταφέρεται σε αυτή την οθόνη μετά από την επιλογή "Ο Λογαριασμός μου" στο menu του Action Bar. Πρόκειται για οθόνη που προβάλλει όλες τις πληροφορίες που αφορούν στον χρήστη που είναι συνδεδεμένος στην εφαρμογή.

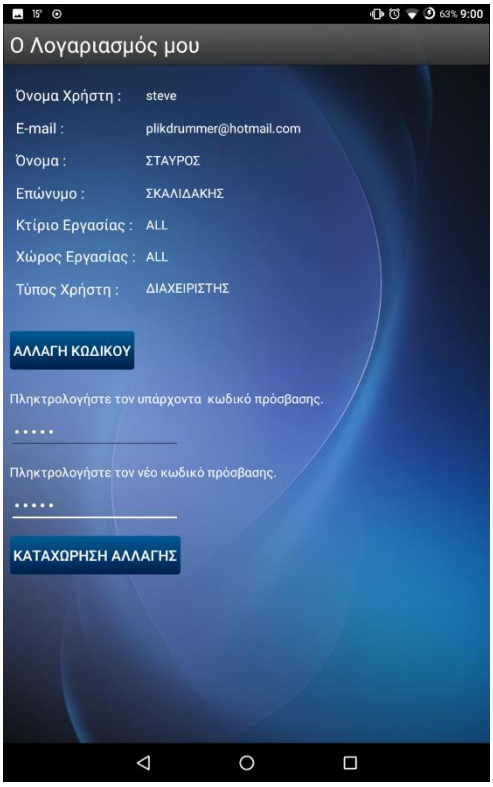
Το ανώτερο μέρος της οθόνης αποτελείται από ένα σύνολο στοιχείων TextView τα οποία εμφανίζουν το όνομα χρήστη, το E-mail, το όνομα, το επώνυμο, τον χώρο εργασίας και τον τύπο του χρήστη. Προκειμένου η εφαρμογή να λειτουργεί σωστά για τον κάθε χρήστη, τα στοιχεία αυτά πρέπει να είναι σωστά.

Στο κάτω μέρος της οθόνης έχει τοποθετηθεί κουμπί αλλαγής κωδικού και ένα στοιχείο τύπου Fragment το οποίο είναι αρχικοποιημένο ως κενό.(και για αυτό τον λόγο δεν είναι ορατό στον χρήστη) Στην πραγματικότητα έχει προκατασκευαστεί συγκεκριμένο περιεχόμενο που θα εμφανιστεί στο σημείο αυτό όταν ο χρήστης πατήσει το κουμπί αλλαγής του κωδικού και αποτελείται από δύο στοιχεία τύπου TextView που περιέχουν μηνύματα οδηγιών για αλλαγή του κωδικού, δύο στοιχεία τύπου EditText (στο ένα εισάγεται ο παλαιός κωδικός και στο άλλο ο νέος) και ένα κουμπί το οποίο καταχωρεί την αλλαγή αυτή.

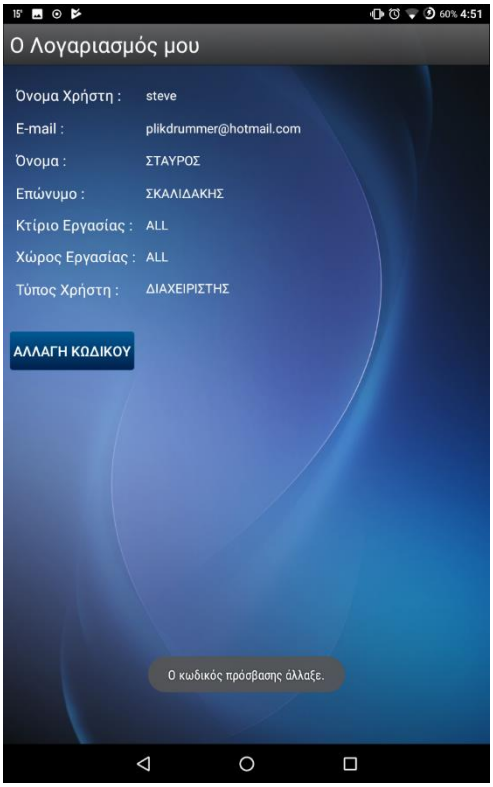
Πέραν της πληροφόρησης για τα στοιχεία του λογαριασμού το χρήστη, (Εικόνα 6.12) η μόνη λειτουργικότητα της εν λόγω οθόνης είναι η αλλαγή του κωδικού. Αν ο χρήστης επιθυμεί να αλλάξει τον κωδικό πρόσβασης του στην εφαρμογή πατάει το αντίστοιχο κουμπί. Εκείνη τη στιγμή, το αόρατο μέχρι στιγμής Fragment που βρίσκεται κάτω από το κουμπί εμφανίζει την φόρμα αλλαγής κωδικού. Ο χρήστης εισάγει τον παλαιό κωδικό και τον νέο κωδικό και πιέζει το κουμπί στο οποίο αναγράφεται "Καταχώρηση Αλλαγής" (Εικόνα 6.13). Ο παλαιός κωδικός πρόσβασης ελέγχεται και αν είναι σωστός γίνεται request στο API για καταχώρηση του νέου κωδικού στην βάση δεδομένων. Όταν αυτή ολοκληρωθεί επιτυχώς το περιεχόμενο του Fragment γίνεται ξανά μη ορατό στον χρήστη και εμφανίζεται μήνυμα το οποίο επιβεβαιώνει την επιτυχή αλλαγή του κωδικού (Εικόνα 6.14). Αν ο έλεγχος του παλαιού κωδικού αποτύχει τότε εμφανίζεται μήνυμα απόρριψης της αλλαγής (Εικόνα 6.15) και προτείνεται να εισαχθεί σωστά ο κωδικός.(ενώ το περιεχόμενο του Fragment παραμένει ορατό στον χρήστη).

Παρακάτω παρουσιάζονται στιγμιότυπα της οθόνης στοιχείων λογαριασμού σε χρήση.

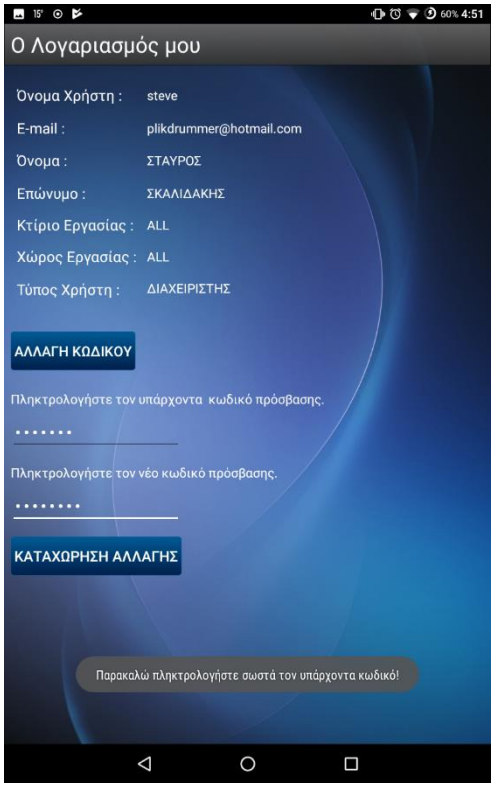
Εικόνα 6.12- Οθόνη πληροφοριών λογαριασμού χρήστη



Εικόνα 6.13- Αλλαγή κωδικού πρόσβασης



Εικόνα 6.14- Επιτυχής αλλαγή κωδικού πρόσβασης



Εικόνα 6.15- Αποτυχημένη αλλαγή κωδικού πρόσβασης

6.5 Οθόνη Πληροφοριών Εφαρμογής (About Us Screen)

Στην οθόνη αυτή μπορεί να μεταβεί ο χρήστης από την επιλογή "Σχετικά με εμάς" στο Menu. Η οθόνη αυτή δεν έχει καμία λειτουργικότητα. Αποτελείται αποκλειστικά από ένα στοιχείο ImageView με το λογότυπο της εφαρμογής και στοιχεία TextView τα οποία περιέχουν γενικές πληροφορίες για την πτυχιακή εργασία και την εφαρμογή.

Εικόνα 6.16- Πληροφορίες Σχετικά με την Εφαρμογή

6.6 Οθόνη Λίστας Χρονοσειρών (StreamList Screen)

Στην οθόνη αυτή οδηγείται ο χρήστης με το πάτημα του κουμπιού στο κάτω αριστερά μέρος της κεντρικής οθόνης της εφαρμογής.

Η οθόνη περιέχει το Action Bar και το menu όπως και το homescreen στο ανώτερο μέρος. Ακριβώς από κάτω έχει τοποθετηθεί Spinner το οποίο έχει φορτωθεί με όλες τις κατηγορίες των χρονοσειρών και λειτουργεί ως φίλτρο με βάση τον τύπο της κάθε χρονοσειράς. (δηλαδή το μέγεθος που αναπαριστά). Πιο κάτω έχουν τοποθετηθεί στη σειρά τέσσερα κουμπιά και ένα Fragment. Το περιεχόμενο του Fragment εξαρτάται κάθε φορά από το ποιο από τα τέσσερα κουμπιά έχει πατηθεί και αποτελείται από στοιχεία τύπου TextView και Spinners με τα οποία ο χρήστης επιλέγει το επιθυμητό χρονικό διάστημα. Το σύστημα αυτό αποτελεί τρόπο αναζήτησης ενεργειακών δεδομένων στον χρόνο.

Στην υπόλοιπη οθόνη εμφανίζονται μέσα σε στοιχείο ListView (σε μορφή λίστας) όλες

οι γραφικές παραστάσεις των χρονοσειρών που ο χρήστης επέλεξε με τη βοήθεια των δύο φίλτρων που αναφέρθηκαν παραπάνω και ο τίτλος της κάθε χρονοσειράς με χρήση TextView.

Ο χρήστης αν δεν επιθυμεί να βρίσκεται στην συγκεκριμένη οθόνη μπορεί να επιστρέψει στο Homescreen είτε με το κουμπί της οπισθοχώρησης είτε μέσα από την επιλογή του Menu. Επίσης μπορεί να μεταβεί στις άλλες δύο οθόνες μέσω του Menu όπως εξηγήθηκε προηγουμένως ή να αποσυνδεθεί τελείως από την εφαρμογή.

Αν επιθυμεί να αναζητήσει συγκεκριμένες τιμές ενός μεγέθους για κάποια χρονική περίοδο τότε εργάζεται ως εξής. Αρχικά, με βάση τον τύπο των χρονοσειρών που τον ενδιαφέρουν επιλέγει την κατηγορία από το αντίστοιχο Spinner. Προς το παρόν οι διαθέσιμες κατηγορίες είναι: "Όλα", "Θερμοκρασία", "Υγρασία", "Κατανάλωση", "Κόστος", "Ρύποι". Το πάτημα κάποιας από τις επιλογές ανανεώνει αυτόματα όλες τις χρονοσειρές που προβάλλονται στη λίστα. Στη συνέχεια, επιλέγεται ο τύπος του χρονικού διαστήματος που αφορά τον χρήστη με το πάτημα ενός από τα τέσσερα κουμπιά.

Αναλυτικότερα, αν ο χρήστης πατήσει το κουμπί "Ετήσια" το Fragment θα του εμφανίσει ένα Spinner φορτωμένο με όλα τα έτη από την έναρξη των δεδομένων (αρχικοποιημένο με την τιμή του παρόντος έτους). Ο ίδιος μπορεί να επιλέξει σε ποιο έτος αφορά η αναζήτηση του και με την επιλογή του ανανεώνεται αυτόματα η λίστα με τις γραφικές παραστάσεις των χρονοσειρών. Το αποτέλεσμα της αναζήτησης στην οθόνη θα αποτελείται από γραφικές παραστάσεις δώδεκα σημείων, ένα σημείο για κάθε μήνα του χρόνου (Εικόνα 2.19).

Αν ο χρήστης πατήσει το κουμπί "Μηνιαία" το Fragment εμφανίζει δύο Spinners. Το πρώτο είναι φορτωμένο με όλα τα έτη από την αρχή των δεδομένων και το δεύτερο με όλους τους μήνες (αρχικοποιημένα στην παρούσα χρονική στιγμή). Ο ίδιος μπορεί να επιλέξει σε ποιο έτος και σε ποιο μήνα αφορά η αναζήτηση του και με την επιλογή του ανανεώνεται αυτόματα η λίστα με τις γραφικές παραστάσεις των χρονοσειρών. Το αποτέλεσμα της αναζήτησης στην οθόνη θα αποτελείται από γραφικές παραστάσεις τόσων σημείων όσες μέρες υπάρχουν μέσα στον επιλεγμένο μήνα του επιλεγμένου έτους (Εικόνα 2.20).

Αν ο χρήστης πατήσει το κουμπί "Εβδομαδιαία" το Fragment εμφανίζει τρία Spinners. Το πρώτο είναι φορτωμένο με όλα τα έτη από την αρχή των δεδομένων, το δεύτερο με όλους τους μήνες και το τρίτο με όλες τις εβδομάδες με σημείο έναρξης την πρώτη Δευτέρα από την αρχή των δεδομένων (αρχικοποιημένα στην παρούσα χρονική στιγμή). Ο ίδιος μπορεί να επιλέξει σε ποιο έτος και σε ποιο μήνα και σε ποια εβδομάδα αφορά η αναζήτηση του και με την επιλογή του ανανεώνεται αυτόματα η λίστα με τις γραφικές παραστάσεις των χρονοσειρών. Το αποτέλεσμα της αναζήτησης στην οθόνη θα αποτελείται από γραφικές παραστάσεις δεκατεσσάρων σημείων. Το κάθε σημείο θα αντιστοιχεί σε διάστημα δώδεκα ωρών μιας μέρας (00:00-12:00 ή 12:00-24:00) και θα απεικονίζονται συνολικά οι επτά μέρες τις εβδομάδας δίνοντας σύνολο δεκατεσσάρων σημείων (Εικόνα 2.21).

Αν ο χρήστης πατήσει το κουμπί "Ημερήσια" το Fragment εμφανίζει τρία Spinners. Το πρώτο είναι φορτωμένο με όλα τα έτη από την αρχή των δεδομένων το δεύτερο με όλους τους μήνες και το τρίτο με ημερομηνίες (όλα αρχικοποιημένα στην παρούσα χρονική στιγμή). Ο χρήστης μπορεί να επιλέξει σε ποιο έτος, σε ποιο μήνα και σε ποια

συγκεκριμένη ημερομηνία αφορά η αναζήτηση του και με την επιλογή του ανανεώνεται αυτόματα η λίστα με τις γραφικές παραστάσεις των χρονοσειρών. Το αποτέλεσμα της αναζήτησης στην οθόνη θα αποτελείται από γραφικές παραστάσεις εικοσιτεσσάρων σημείων, δηλαδή τόσων σημείων όσες ώρες υπάρχουν μέσα σε μία ημέρα (Εικόνα 2.22).

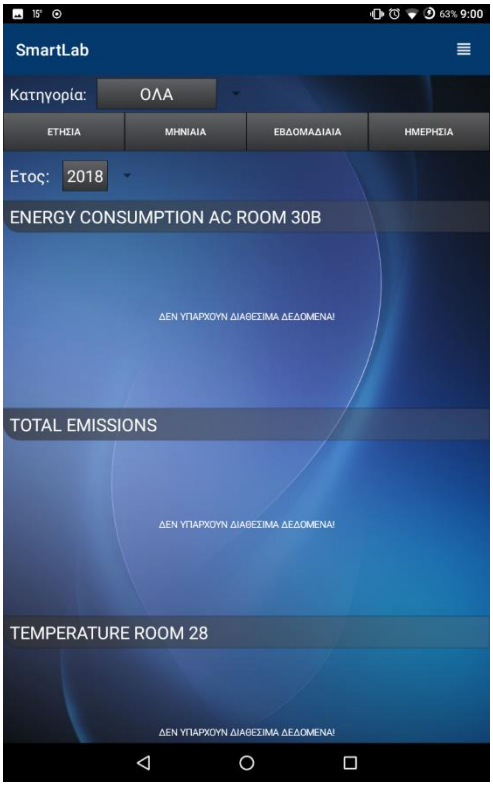
Με λίγα λόγια ο χρήστης με τα δύο παραπάνω φίλτρα συγκεκριμενοποιεί την αναζήτηση του ως προς τον τύπο του ενεργειακού μεγέθους που τον αφορά και τον τύπο του χρονικού διαστήματος, αλλά επίσης δηλώνει και την χρονική στιγμή στην οποία θέλει να αναπαρασταθούν γραφικά τα δεδομένα. Τα παραπάνω βήματα μπορούν να εκτελεστούν με οποιαδήποτε σειρά αλλά σε κάθε αλλαγή που συμβαίνει ανανεώνεται αυτόματα και η λίστα αφού για κάθε αλλαγή γίνεται ένα νέο Request στο API (με διαφορετικό χρονικό διάστημα ενδιαφέροντος είτε με διαφορετική κατηγορία χρονοσειράς) και τα δεδομένα που λαμβάνει η εφαρμογή είναι διαφορετικά. Η φόρτωση των χρονοσειρών μπορεί να διαρκέσει κάποια δευτερόλεπτα σε κάποιες περιπτώσεις. Για τον λόγο αυτό καθ' όλη τη διάρκεια της φόρτωσης των χρονοσειρών εμφανίζεται αντίστοιχο μήνυμα αναμονής (Εικόνα 6.17). Επίσης, κάποιες χρονοσειρές που ζητούνται είναι πιθανό να μην περιέχουν δεδομένα για το ζητούμενο χρονικό διάστημα. Σε αυτές τις περιπτώσεις εμφανίζεται μήνυμα πάνω στο γράφημα που ενημερώνει τον χρήστη για την έλλειψη διαθέσιμων δεδομένων Εικόνα 6.18). Αξίζει να σημειωθεί σε αυτό το σημείο ότι στον χρήστη είναι ορατές μόνο οι χρονοσειρές που σχετίζονται με τον χώρο στον οποίο εργάζεται. Ο διαχειριστής (administrator) είναι ο μόνος χρήστης στον οποίο είναι ορατές όλες οι χρονοσειρές.

Ωστόσο, για λόγους ορθής λειτουργίας και δυνατότητας scrolling της λίστας δεν έχουν δοθεί όλες η λειτουργίες πάνω στο κάθε γράφημα. Αν ο χρήστης επιθυμεί να μελετήσει αναλυτικά κάποιο από τα γραφήματα τότε αρκεί να πατήσει πάνω σε αυτό. Τότε θα μεταβεί στην οθόνη αναλυτικών πληροφοριών της χρονοσειράς που επέλεξε (DetailedStream Screen) η οποία περιγράφεται αναλυτικά στην επόμενη παράγραφο.

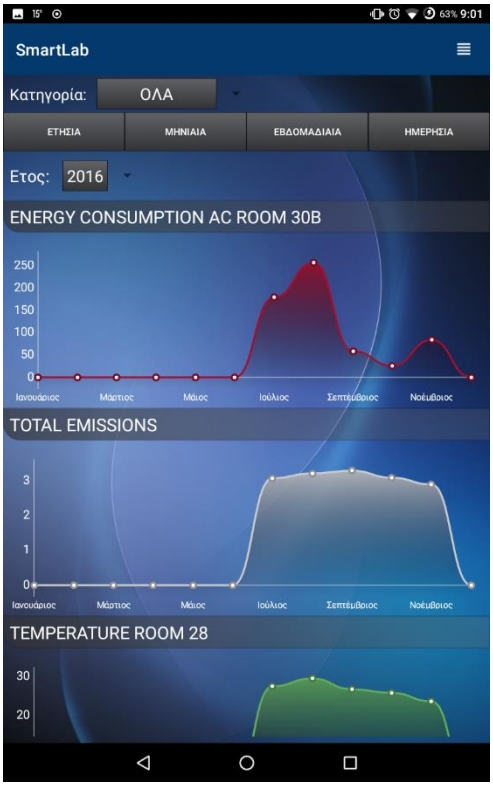
Στο συγκεκριμένο σημείο θα πρέπει να τονίσουμε ότι μία από τις σημαντικότερες δυσκολίες στην υλοποίηση ήταν η κατασκευή το Adaptor ο οποίος διαχειρίζεται την λίστα. Λόγω αρχικής κακής υλοποίησης πολλές φορές δημιουργούνταν πολλαπλά στοιχεία μέσα στη λίστα μετά από ανανέωση της ή scrolling σε σημείο που κάποιες γραφικές παραστάσεις σταματούσαν να είναι ορατές στον χρήστη. Τα εμπόδια αυτά ξεπεράστηκαν με την πλήρη κατανόηση του στοιχείου ListView και των κλάσεων που σχετίζεται μέσα από βιβλιογραφία, βίντεο με αντίστοιχο εκπαιδευτικό υλικό και επανακατασκευή των τμημάτων κώδικα που προκαλούσαν τον πρόβλημα.

Παρακάτω παρουσιάζονται στιγμιότυπα της οθόνης με τη λίστα των χρονοσειρών. Φαίνονται όλα τα σενάρια χρήσης των φίλτρων .

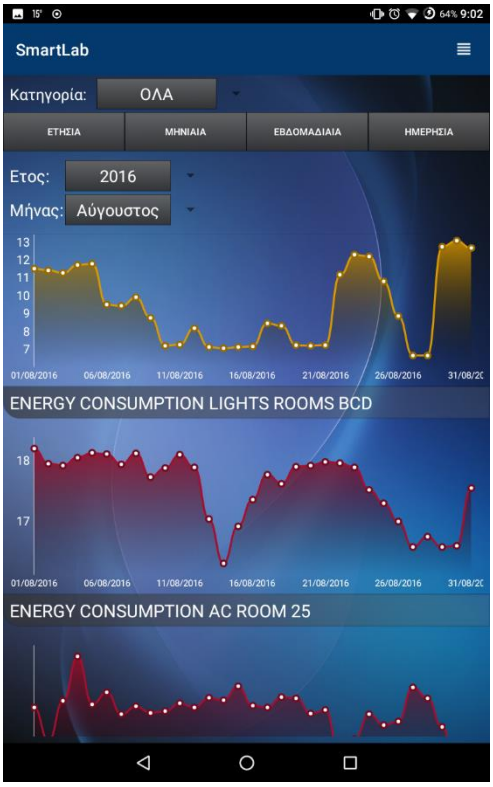
Εικόνα 6.17- Μπάρα φόρτωσης χρονοσειρών



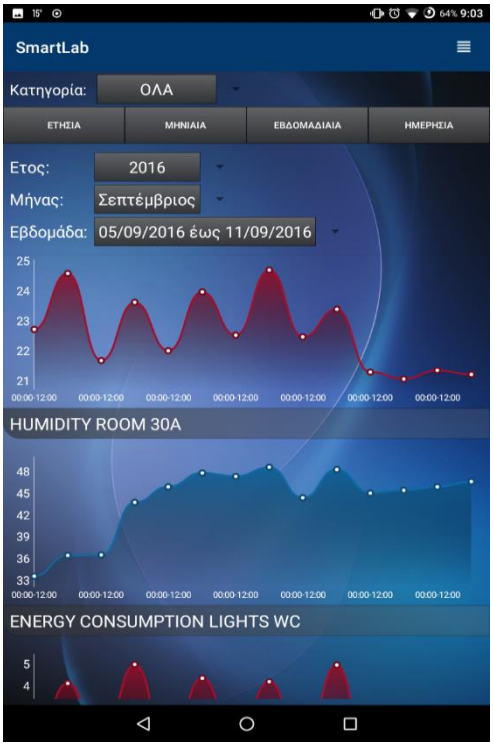
Εικόνα 6.18- Κενές χρονοσειρές



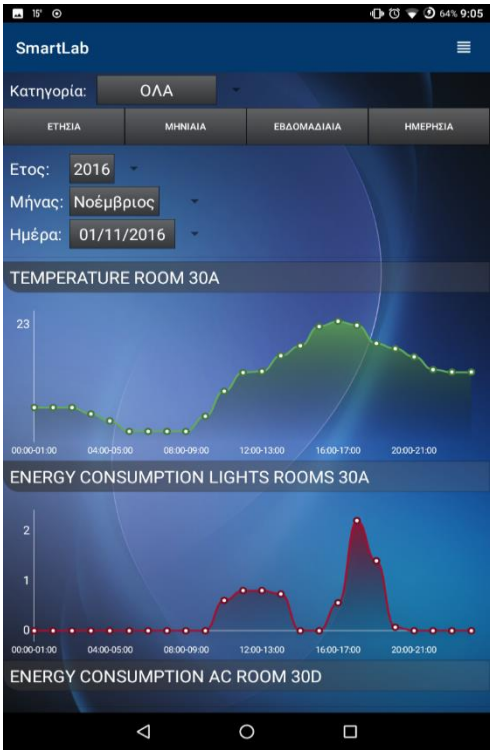
Εικόνα 6.19- Ετήσιες χρονοσειρές



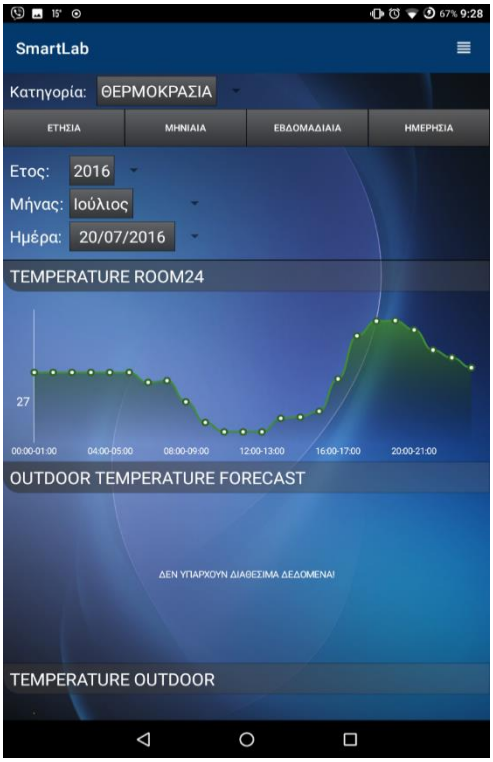
Εικόνα 6.20- Μηνιαίες χρονοσειρές



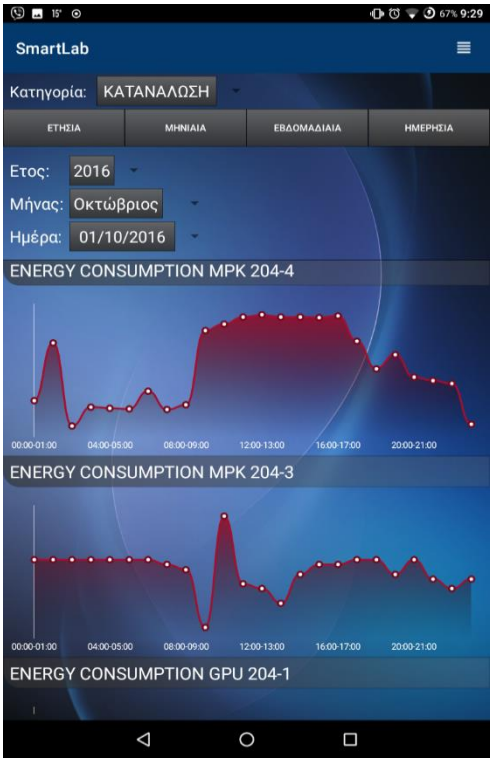
Εικόνα 6.21- Εβδομαδιαίες χρονοσειρές



Εικόνα 6.22- Ημερήσιες χρονοσειρές



Εικόνα 6.23- Επιλογή κατηγορίας "Θερμοκρασία"



Εικόνα 6.24- Επιλογή κατηγορίας "Κατανάλωση"

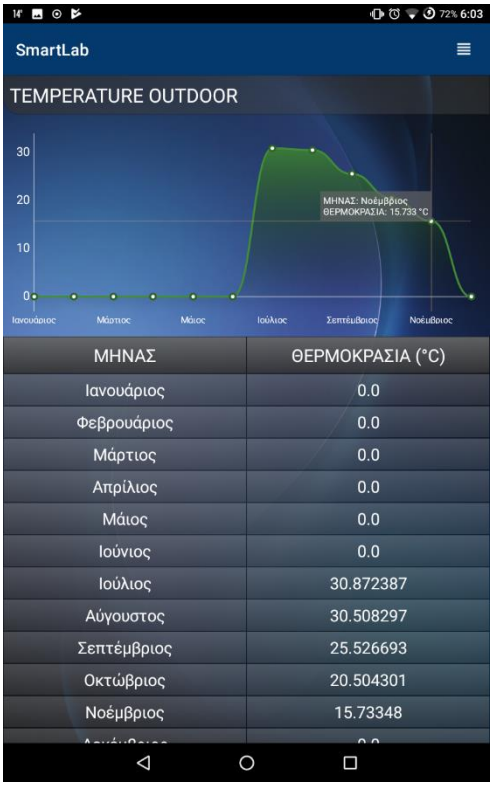
6.7 Οθόνη Αναλυτικής Περιγραφής Χρονοσειράς (DetailedStream Screen)

Στην οθόνη αυτή οδηγείται ο χρήστης με το πάτημα κάποιας γραφικής παράστασης χρονοσειράς που υπάρχει μέσα στην λίστα του StreamList Screen. Το περιεχόμενο της εξαρτάται από το ποια χρονοσειρά επιλέχτηκε από τον χρήστη.

Η οθόνη περιέχει το Action Bar και το menu όπως και το Homescreen και το StreamList Screen στο ανώτερο μέρος. Ακριβώς από κάτω, έχει τοποθετηθεί η γραφική παράσταση (LineChart) της επιλεγμένης χρονοσειράς και ένα στοιχείο TextView που περιέχει το όνομα της χρονοσειράς. Στο υπόλοιπο μέρος της οθόνης έχει τοποθετηθεί στοιχείο GridView το οποίο παίζει τον ρόλο πίνακα τιμών όλων των στοιχείων που απεικονίζονται στο γράφημα.

Το νόημα της συγκεκριμένης οθόνης είναι η δυνατότητα που δίνεται στον χρήστη να μελετήσει αναλυτικά την χρονοσειρά που έχει επιλέξει. Ο χρήστης μπορεί να μεγεθύνει την γραφική παράσταση (Εικόνα 6.30) ώστε να την παρατηρήσει με μεγαλύτερη λεπτομέρεια (με κίνηση τσιμπήματος ή ανάποδου τσιμπήματος για σμίκρυνση), να μετακινηθεί κατά μήκος των αξόνων x και y του γραφήματος (με απλό σύρσιμο) (Εικόνα 6.31) , ή να ελέγξει την ακριβή τιμή ενός στοιχείου πάνω στο γράφημα με ένα απλό πάτημα πάνω σε αυτό (Εικόνα 6.29) . Επίσης, ο πίνακας τιμών όλων των στοιχείων εμφανίζεται στο κάτω μέρος της οθόνης ως εναλλακτικός τρόπος αναπαράστασης. Μάλιστα, αξίζει να σημειωθεί πως με βάση με την επιλογή του τύπου χρονικού διαστήματος και του τύπου ενεργειακών δεδομένων ο πίνακας τροποποιείται ανάλογα ώστε να είναι σαφές κάθε φορά στον χρήστη το τι νόημα έχει ακριβώς ό,τι προβάλλεται στην οθόνη. Συγκεκριμένα, ως επικεφαλίδα στον πίνακα που περιέχει τις χρονικές τιμές εμφανίζεται “Μήνας”, “Ημέρα”, “Ημερομηνία-Ώρες” και “Ώρες” αν ο χρήστης έχει επιλέξει στην προηγούμενη οθόνη δεδομένα “Ετήσια”, “Μηνιαία”, “Εβδομαδιαία” και “Ημερήσια” αντίστοιχα (Εικόνες 6.25, 6.26, 6.27, 6.28) . Ομοίως, ανάλογα με τον τύπο της χρονοσειράς εμφανίζεται ως επικεφαλίδα του πίνακα των μετρήσεων του ενεργειακού μεγέθους, η κατηγορία του μεγέθους και η μονάδα μέτρησης του. Για παράδειγμα, αν ο χρήστης έχει επιλέξει χρονοσειρές θερμοκρασίας, θα εμφανίζεται ως επικεφαλίδα “ Θερμοκρασία °C ”. Τέλος, διευκρινίζεται ότι ο χρήστης μπορεί να σύρει το δάχτυλο του προς τα πάνω στο σημείο της οθόνης που προβάλλεται η λίστα προκειμένου να κάνει scrolling και να δει παραπάνω δεδομένα στον πίνακα.

Παρακάτω παρουσιάζονται στιγμιότυπα της οθόνης της επιλεγμένης χρονοσειράς.

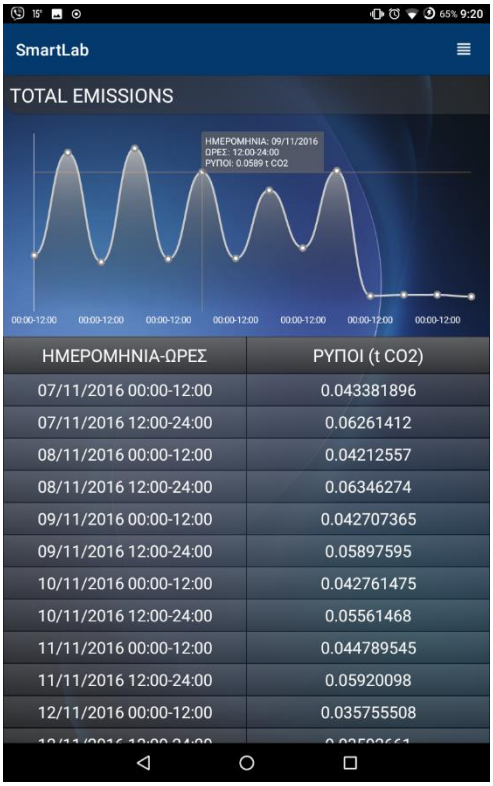


ΜΗΝΑΣ	ΘΕΡΜΟΚΡΑΣΙΑ (°C)
Ιανουάριος	0.0
Φεβρουάριος	0.0
Μάρτιος	0.0
Απρίλιος	0.0
Μάιος	0.0
Ιούνιος	0.0
Ιούλιος	30.872387
Αύγουστος	30.508297
Σεπτέμβριος	25.526693
Οκτώβριος	20.504301
Νοέμβριος	15.73348

Εικόνα 6.25- Αναλυτική παρουσίαση ετήσιας χρονοσειράς

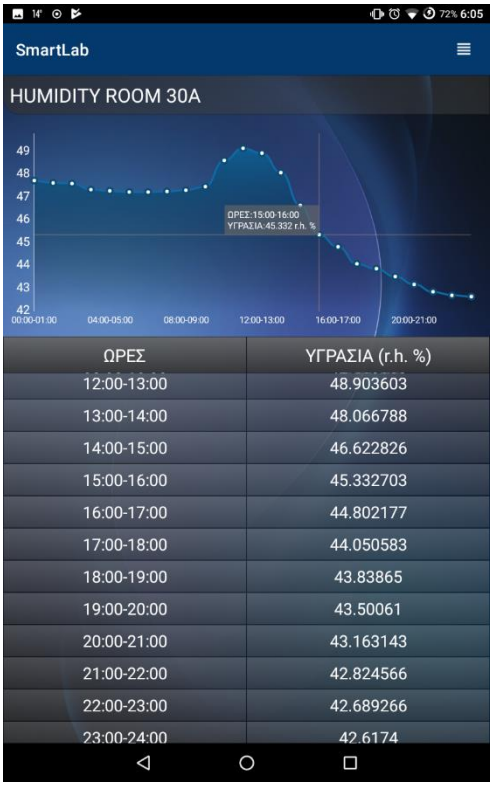
ΗΜΕΡΑ	ΚΑΤΑΝΑΛΩΣΗ (Watt-Hours)
01/08/2016	0.07164633
02/08/2016	0.06871417
03/08/2016	0.07222943
04/08/2016	0.07616952
05/08/2016	0.07193788
06/08/2016	0.07299579
07/08/2016	0.07106323
08/08/2016	0.07173796
09/08/2016	0.07117985
10/08/2016	0.07131313
11/08/2016	0.07201285

Εικόνα 6.26- Αναλυτική παρουσίαση μηνιαίας χρονοσειράς



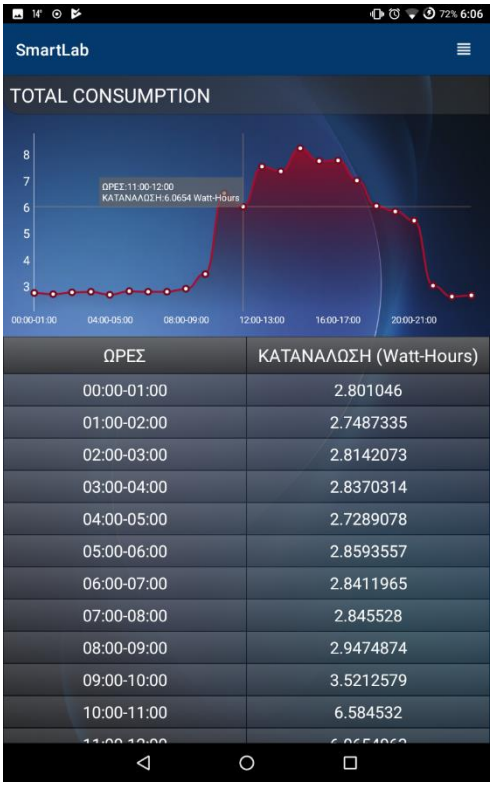
ΗΜΕΡΟΜΗΝΙΑ-ΩΡΕΣ	ΡΥΠΟΙ (t CO2)
07/11/2016 00:00-12:00	0.043381896
07/11/2016 12:00-24:00	0.06261412
08/11/2016 00:00-12:00	0.04212557
08/11/2016 12:00-24:00	0.06346274
09/11/2016 00:00-12:00	0.042707365
09/11/2016 12:00-24:00	0.05897595
10/11/2016 00:00-12:00	0.042761475
10/11/2016 12:00-24:00	0.05561468
11/11/2016 00:00-12:00	0.044789545
11/11/2016 12:00-24:00	0.05920098
12/11/2016 00:00-12:00	0.035755508

Εικόνα 6.27- Αναλυτική παρουσίαση εβδομαδιαίας χρονοσειράς

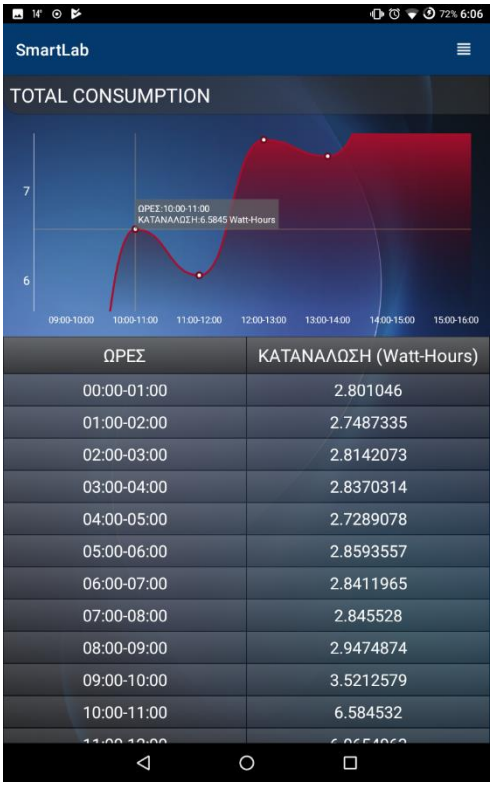


ΩΡΕΣ	ΥΓΡΑΣΙΑ (r.h. %)
12:00-13:00	48.903603
13:00-14:00	48.066788
14:00-15:00	46.622826
15:00-16:00	45.332703
16:00-17:00	44.802177
17:00-18:00	44.050583
18:00-19:00	43.83865
19:00-20:00	43.50061
20:00-21:00	43.163143
21:00-22:00	42.824566
22:00-23:00	42.689266
23:00-24:00	42.6174

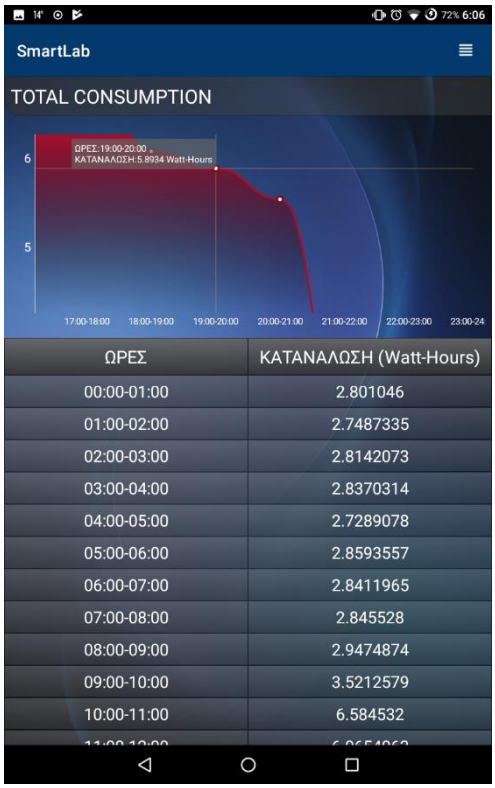
Εικόνα 6.28- Αναλυτική παρουσίαση ημερήσιας χρονοσειράς



Εικόνα 6.29- Πάτημα πάνω στο γράφημα



Εικόνα 6.30- Μεγέθυνση γραφήματος (Zoom In)



Εικόνα 6.31- Μετακίνηση όψης γραφήματος

Κεφάλαιο 7ο - Συμπεράσματα, Περιορισμοί και Προτάσεις Μελλοντικής Εξέλιξης

Στο κεφάλαιο αυτό γίνεται περιγραφή των αποτελεσμάτων και των συμπερασμάτων που εξάχθηκαν από την δημιουργία του συστήματος παρουσίασης ενεργειακών δεδομένων. Συγκεκριμένα, αναλύονται η γενική εικόνα της κατασκευής, τα πλεονεκτήματα και μειονεκτήματα που παρατηρήθηκαν μετά από της χρήση της και τέλος οι προτάσεις που μπορούν να οδηγήσουν στην επέκταση της εφαρμογής, προσθέτοντας της περισσότερες ουσιαστικές λειτουργίες.

7.1 Συμπεράσματα και Αποτελέσματα

Όπως έχει προαναφερθεί, στόχος του συστήματος είναι η συλλογή, η αποθήκευση και η προβολή των στοιχείων ενεργειακών δεδομένων. Απώτερος σκοπός εφαρμογών αυτού του τύπου, είναι η εξοικονόμηση ενέργειας διατηρώντας τις συνθήκες άνεσης του κτιρίου ή ακόμα και βελτιώνοντας τις, η εξοικονόμηση χρημάτων και τελικά η περιβαλλοντική προστασία.

Εφαρμογές και προγράμματα τα οποία έχουν τον παραπάνω στόχο αναπτύσσονται σε μεγάλο βαθμό τα τελευταία χρόνια. Τέτοια παραδείγματα είναι οι εφαρμογές “Smartwatt” της εταιρίας “Watt + Volt”, “Energy Audit” της “Tessera Multimedia”, “Sense Home Energy Monitor” της “Sense Lab Inc.”, “Neurio Home” της “Neurio Technology” και πολλές άλλες. Προτού ξεκινήσουμε την υλοποίηση της δικής μας εφαρμογής παρατηρήσαμε πολλές από τις λειτουργίες των παραπάνω εφαρμογών, την δομή τους και τη χρήση των τεχνολογιών. Αυτό μας παρείχε σημαντικές πληροφορίες και ιδέες για την ανάπτυξη του δικού μας συστήματος όσον αφορά στην συλλογή, στην επεξεργασία, στην αποθήκευση και στην παρουσίαση των δεδομένων.

Οι παραπάνω εφαρμογές λειτουργούν και είναι συμβατές μόνο με χρήση του αντίστοιχου hardware το οποίο πωλείται από τις ίδιες τις εταιρίες. Αυτό μας έδωσε την ιδέα να δημιουργήσουμε το σύστημα με τέτοιο τρόπο ώστε να μπορεί να λαμβάνει πληροφορίες από διαφορετικά μέσα αποφεύγοντας σε μεγάλο βαθμό τον περιορισμό από το hardware. Τα συστήματα των parsers και publishers είναι ανεξάρτητα μεταξύ τους και με την εφαρμογή. Αυτό δίνει την δυνατότητα στους προγραμματιστές να δημιουργούν συνεχώς καινούριο κώδικα συλλογής και αποθήκευσης πληροφοριών αυξάνοντας με πολύ εύκολο τρόπο την ροή των πληροφοριών, αφού κάνουν το σύστημα συμβατό με περισσότερες συσκευές και APIs. Συνεπώς, το πρώτο κομμάτι του συστήματος είναι επεκτάσιμο και μάλιστα με πολύ εύκολο τρόπο. Επίσης, η βάση δεδομένων έχει δομηθεί με τέτοιο τρόπο που καλύπτει πλήρως την εφαρμογή και τις απαιτήσεις τις σε πληροφορία. Φυσικά, μπορούν να προστεθούν νέοι πίνακες σε περίπτωση επέκτασης χωρίς να επηρεαστεί η υπάρχουσα δομή.

Η χρήση της εφαρμογής μας οδήγησε στο συμπέρασμα ότι η παρουσίαση των δεδομένων είναι το πρώτο αλλά το σημαντικότερο βήμα για την εξοικονόμηση ενέργειας. Η εμπειρία των χρηστών που δοκίμασαν την εφαρμογή χαρακτηρίστηκε ως ευχάριστη και ουσιώδης.

Σημαντικό ρόλο για αυτό έπαιξε η δομή της εφαρμογής. Η κεντρική οθόνη περιέχει

γενικές πληροφορίες που πολλές φορές καλύπτουν τον χρήστη και αυτός δεν επιθυμεί να αναζητήσει συγκεκριμένα δεδομένα στο παρελθόν.(ή το μέλλον αν πρόκειται για πρόβλεψη). Ακόμα και αν αυτό κριθεί αναγκαίο, η χρήση των φίλτρων διευκολύνει σημαντικά την αναζήτηση δεδομένων. Η προβολή των στοιχείων μόνο των χρονοσειρών που ενδιαφέρουν τον χρήστη θεωρείται επίσης σημαντικό προτέρημα της εφαρμογής, αφού με τον τρόπο αυτό αποφεύγεται η συσσώρευση μη αξιοποιήσιμης πληροφορίας στην οθόνη. Σε τελική ανάλυση, πρόκειται για εφαρμογή σε προσωπική συσκευή και για τον λόγο αυτό οι πληροφορίες πρέπει να είναι εξατομικευμένες, ώστε η περιήγηση να μην κουράζει τον χρήστη. Οι γραφικές αναπαραστάσεις των ενεργειακών μεγεθών είναι επίσης πολύ σημαντικές, καθώς ο χρήστης αποκτά γρήγορα και εύκολα μια γενική εικόνα για την πορεία των μεγεθών μέσα στον χρόνο. Οι πίνακες με τα αναλυτικά δεδομένα και οι λειτουργίες που προσφέρει η βιβλιοθήκη MPAndroid Chart πάνω στις γραφικές παραστάσεις, μετατρέπουν την μελέτη μιας χρονοσειράς σε ανώδυνη και ταχύτατη. Οι κινήσεις που απαιτούνται από τον χρήστη για την εξαγωγή πληροφοριών για συγκεκριμένο μέγεθος είναι ελάχιστες και γενικότερα η μεταφορά του χρήστη από οθόνη σε οθόνη και η χρήση της εφαρμογής είναι μια πολύ απλή διαδικασία, ακόμα και αν οι γνώσεις του πάνω στην mobile τεχνολογία είναι ελάχιστες. Οπτικά η εφαρμογή είναι πολύ φιλική προς στον χρήστη και η χρήση του animation στα γραφήματα, τα χρώματα, τα λειτουργικά γραφικά (Spinners, Buttons, EditText κτλ.) ολοκληρώνουν την εμπειρία του χρήστη.

Η ύπαρξη του τελευταίου τμήματος του συστήματος, δηλαδή του API, αποδείχθηκε πλήρως λειτουργική και απαραίτητη για πολλούς λόγους. Αρχικά, υπάρχει ασφάλεια ειδικά εφόσον πρόκειται για εφαρμογή που θα χρησιμοποιείται από πολλούς χρήστες. Παράλληλα, μπορούμε να υποθέσουμε ότι αν κάθε χρήστης έκανε τα δικά του Queries στην βάση αυτά δεν θα ήταν τα βέλτιστα και αυτό μπορούσε να μειώσει σημαντικά την απόδοση της βάσης. Ως εκ τούτου, από το γεγονός ότι δεν δίνεται πρόσβαση στη βάση δεδομένων στον οποιοδήποτε, παρά μόνο στο API, συμπεραίνουμε ότι έχει εξασφαλιστεί η ασφάλεια των δεδομένων και η μέγιστη απόδοση. Τέλος, παρατηρήσαμε πως η αλλαγή στη δομή της βάσης ή η ολοκληρωτική αλλαγή πλατφόρμας δεν θα επηρεάζει καθόλου την εφαρμογή και τους χρήστες παρά μόνο το API.

7.2 Πλεονεκτήματα και Μειονεκτήματα της Υλοποίησης

Η χρήση του συστήματος που κατασκευάστηκε είχε σημαντικά πλεονεκτήματα σε σχέση με άλλες παρόμοιες εφαρμογές. Αυτά που εντοπίστηκαν μετά από την ολοκλήρωση και τη χρήση του είναι τα εξής :

- Η εφαρμογή είναι λειτουργική, εύχρηστη, καλαίσθητη και φιλική προς τον χρήστη απαιτώντας ελάχιστο αριθμό κινήσεων για την εκπλήρωση ενός στόχου. Η χρήση της δεν προϋποθέτει γνώσεις πληροφορικής ή οποιασδήποτε τεχνολογίας.

- Διαθέτει ευελιξία καθώς τα κατασκευασμένα φίλτρα επιτρέπουν την προβολή στοιχείων για επιλεγμένα χρονικά διαστήματα, γεγονός που διευκολύνει σε μεγάλο βαθμό την μελέτη των χρονοσειρών.

- Αφού τα δεδομένα ενημερώνονται κάθε λίγα λεπτά, δίνει την εντύπωση μιας real-time αναπαράστασης ενεργειακών μεγεθών.

- Προσφέρει δυνατότητες πρόβλεψης μεγεθών με τη βοήθεια του εξωτερικού API.

- Προωθεί την εξοικονόμηση ενέργειας σε πρώτη φάση δίνοντας τη δυνατότητα στον χρήστη να έχει μια εικόνα για τα δεδομένα που τον ενδιαφέρουν. Η γραφική απεικόνιση τον μεγεθών διαδραματίζει τον κύριο ρόλο σε αυτή τη διαδικασία. Ως εκ τούτου μειώνεται η ενεργειακή σπατάλη και το κόστος της κατανάλωσης ενώ παράλληλα γίνεται συνεχής έλεγχος ώστε να μην υποβαθμίζεται η ποιότητα των συνθηκών άνεσης στο κτίριο. (αυτό μπορεί να γίνεται με παρατήρηση μεγεθών όπως η θερμοκρασία και η υγρασία)

- Προσφέρει αποκλειστικά εξατομικευμένες πληροφορίες στον χρήστη και όχι όγκο δεδομένων τα οποία του είναι αδιάφορα.

- Το κάθε στάδιο του συστήματος είναι ανεξάρτητο. Αυτό σημαίνει ότι η επέκταση ή η αντικατάσταση ενός τμήματος αποτελεί πολύ εύκολη διαδικασία για τους προγραμματιστές.

- Το σύστημα είναι σε κάθε στάδιο επεκτάσιμο. Πιο συγκεκριμένα, στη διαδικασία της συλλογής πληροφοριών, ο χρήστης μπορεί να αποκτήσει δεδομένα από πολλούς τύπους αρχείων είτε αυτά προέρχονται από αισθητήρες είτε από API, απλά τροποποιώντας τα configuration files των parsers. Συνεπώς, με μικρές αλλαγές μπορεί να κάνει το σύστημα συμβατό με διαφορετικούς τύπους αισθητήρων και εξωτερικών APIs επεκτείνοντας σημαντικά τις δυνατότητες αφού η εφαρμογή βασίζεται στα δεδομένα. Επίσης, ο σχεδιασμός της mobile εφαρμογής επιτρέπει την επέκταση των λειτουργιών της κάτι που θα βελτιώσει σημαντικά τη λειτουργικότητα και το νόημα της εφαρμογής. Κάποιες από τις προτάσεις για την επέκταση της εφαρμογής αναφέρονται στην επόμενη παράγραφο.

Εκτός από τα πλεονεκτήματα ωστόσο, η εφαρμογή παρουσιάζει και ορισμένα μειονεκτήματα:

- Η γραφική απεικόνιση στοιχείων της εφαρμογής εξαρτάται αποκλειστικά από τα δεδομένα. Αν για κάποιο λόγο οι αισθητήρες ή το API δεν παρέχουν τα δεδομένα η εφαρμογή ναι μεν είναι λειτουργική αλλά παρουσιάζει μόνο παλαιότερα στοιχεία.

- Η απουσία σύνδεσης στο διαδίκτυο δεν επιτρέπει τη χρήση της εφαρμογής.

- Απουσιάζουν κάποιες λειτουργικότητες οι οποίες θα καθιστούσαν την εφαρμογή ένα πλήρες προϊόν. Κάποιες από αυτές περιγράφονται σε επόμενη παράγραφο.

7.3 Περιορισμοί

Για την ανάπτυξη του συστήματος υπήρξαν ορισμένοι περιορισμοί που δυσχέραιναν την ανάπτυξη του και περιόρισαν σε κάποιο βαθμό το τελικό αποτέλεσμα. Αυτοί οι περιορισμοί περιγράφονται παρακάτω.

Περιορισμένοι τύποι συσκευών

Έναν βασικό περιορισμό αποτελεί το γεγονός ότι οι συσκευές στις οποίες δοκιμάστηκε η λειτουργία της εφαρμογής είναι μόνο τρεις. Αυτό σημαίνει ότι παρόλο που έγινε σημαντική προσπάθεια για την δημιουργία μίας εύχρηστης και μη απαιτητικής σε πόρους εφαρμογής δεν μπορεί κανείς να είναι απόλυτα σίγουρος ότι η εφαρμογή λειτουργεί χωρίς προβλήματα σε όλες τις συσκευές, διότι η μνήμη, ο επεξεργαστής, οι διαστάσεις της οθόνης άλλων συσκευών μπορεί να μην το επιτρέπουν. Επίσης, αν και θεωρητικά η εφαρμογή λειτουργεί σε όλα τα λειτουργικά Android με API Level μεγαλύτερο από 19, δεν μπορεί κανείς να εγγυηθεί σωστή λειτουργία για κάθε έκδοση του λειτουργικού. Συγκεκριμένα, η δοκιμή του σε συσκευή με Android Lollipop εμφάνισε πρόβλημα στο animation των γραφικών παραστάσεων. Ωστόσο, δεν είναι σίγουρο αν το συγκεκριμένο θέμα αφορά στη συσκευή, στο λειτουργικό της ή σε κάποια ατέλεια της βιβλιοθήκης που χρησιμοποιήθηκε για τον σχεδιασμό των γραφικών παραστάσεων.

Περιορισμοί στα δεδομένα

Σημαντικό περιορισμό αποτελεί επίσης το γεγονός ότι τα διαθέσιμα δεδομένα των αισθητήρων είναι πολύ περιορισμένα. Για τον λόγο αυτό δεν ήταν δυνατό να εξεταστεί ο χρόνος απόκρισης απαιτητικών queries στη βάση, όπως για παράδειγμα αυτών που συλλέγουν ετήσια δεδομένα. Εκτός αυτού, λόγω της μη λειτουργίας των αισθητήρων κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας, η έλλειψη προσφάτων δεδομένων και για την ακρίβεια του τελευταίου μήνα, ή της τελευταίας ώρας περιορίζει σημαντικά την λειτουργία της κεντρικής οθόνης η οποία βασίζεται στην προβολή μεγεθών τη δεδομένη στιγμή.

7.4 Προτάσεις εξέλιξης

Το Mobile Application, που αναπτύχθηκε διαθέτει μεν συμπαγή μορφή, αλλά μπορεί να εμπλουτιστεί με νέα στοιχεία τα οποία μπορεί να διευρύνουν τις δυνατότητές του καθώς και να συνεργαστεί με άλλες εφαρμογές ώστε να αποκτήσει ακόμα περισσότερα δεδομένα.

7.4.1 Προτάσεις σχετικά με τα δεδομένα

Όπως έχει τονιστεί αρκετές φορές προηγουμένως, η ύπαρξη δεδομένων είναι ο σημαντικότερος παράγοντας προκειμένου η εφαρμογή να είναι λειτουργική. Το γεγονός αυτό μας οδηγεί στο συμπέρασμα ότι για να επιτύχουμε την εξέλιξη του συστήματος μας θα πρέπει να εστιάσουμε στην αύξηση παροχής πληροφοριών και στην αποθήκευσή τους στη βάση δεδομένων. Αυτό προϋποθέτει χρήση περισσότερων αισθητήρων που μετρούν διαφορετικά μεγέθη και παραμετροποίηση των parsers και publishers ώστε να συλλέγεται και να αποθηκεύεται η πληροφορία αυτή.

Παράλληλα, η αύξηση των δεδομένων μπορεί να επιτρέψει την δημιουργία ακόμα περισσότερων τεχνητών χρονοσειρών όπως αυτές που περιγράφηκαν στο τρίτο κεφάλαιο. Για παράδειγμα, θα μπορούσαμε να δημιουργήσουμε χρονοσειρές που:

- Απεικονίζουν τη συνολική κατανάλωση σε κάθε δωμάτιο ξεχωριστά.
- Δείχνουν την συνολική κατανάλωση ανά κατηγορία. Για παράδειγμα, συνολική κατανάλωση από μονάδες κλιματισμού, από τον φωτισμό, από τη θέρμανση, από ηλεκτρονικές συσκευές και άλλα.
- Να αποτυπώνουν δείκτες ενεργειακής αποδοτικότητας χρησιμοποιώντας τις χρονοσειρές της θερμοκρασίας, της κατανάλωσης ενέργειας ή κάποιου μετρητή καυσίμων. (φυσικού αερίου, πετρελαίου κτλ.)
- Να συνδυάζουν προβλέψεις πολλών API για διάφορα μεγέθη με σκοπό την καλύτερη πρόβλεψη. (αυτό μπορεί να βελτιωθεί ακόμα περισσότερο και με χρήση μοντέλων πρόβλεψης)
- Να προβάλλουν την συνολική παραγωγή ενέργειας αν το κτίριο κάνει χρήση ανεμογεννητριών, φωτοβολταϊκών και άλλων μορφών ενέργειας.
- Ακόμα και χρονοσειρές που αποκλίνουν από το καθαρά ενεργειακό αντικείμενο αλλά σχετίζονται με την περιβαλλοντική διαχείριση και μπορούν να αφορούν στην κατανάλωση νερού και άλλα.

Επίσης, η χρήση άλλου τύπου αισθητήρων, όπως για παράδειγμα αισθητήρες κίνησης, μπορεί να συνδυαστεί με τους υπάρχοντες προκειμένου να παρατηρείται ευκολότερα η ενεργειακή σπατάλη. Ένα παράδειγμα είναι ο φωτισμός ενός χώρου στον οποίο μια συγκεκριμένη χρονική στιγμή δεν βρίσκεται κάποιο άτομο. Σε αυτή την περίπτωση, τα δεδομένα του αισθητήρα κατανάλωσης στα φώτα του δωματίου μπορούν να συνδυαστούν με αυτά του αισθητήρα κίνησης και να προτείνεται κάποια λύση.

Τέλος, μια άλλη πρόταση για βελτίωση του συστήματος είναι η περαιτέρω μείωση του χρονικού διαστήματος μεταξύ των στιγμών που οι αισθητήρες κάνουν τις μετρήσεις. Αν αυτό γίνει αρκετά μικρό, τότε το σύστημα θα προσφέρει real-time αναπαράσταση μεγεθών. Αυτή όμως η επιλογή είναι ριψοκίνδυνη και θα πρέπει να μελετηθεί αναλυτικά πριν την εφαρμογή της, καθώς ο όγκος της πληροφορίας θα αυξηθεί σημαντικά και αυτό μπορεί να οδηγήσει σε μειωμένη απόδοση.

7.4.2 Προτάσεις σχετικά με τις λειτουργίες της εφαρμογής

Όσον αφορά στην εν λόγω εφαρμογή υπάρχουν ορισμένες προσθήκες οι οποίες μπορούν να βελτιώσουν σημαντικά την λειτουργικότητα της. Κάποιες από αυτές είναι οι εξής:

- Η προσθήκη της λειτουργίας ενημερώσεων (notifications) προς τον χρήστη με σκοπό την πληροφόρηση του σε σχέση με κάποια γεγονότα. Αυτό θα μπορούσε να είναι πολύ χρήσιμο σε περιπτώσεις που η κατανάλωση ενέργειας υπερβεί κάποιο όριο που έχει θέσει ο χρήστης ή στην περίπτωση επερχόμενου καύσωνα

ή παγετού. Γενικά, τα notifications μπορούν να διευκολύνουν σημαντικά την πληροφόρηση του χρήστη, αφού μπορούν να συμβαίνουν και για συγκεκριμένα συμβάντα που ενδιαφέρουν τον χρήστη ώστε αυτός να μην χρειάζεται να αναζητά συνέχεια τις πληροφορίες που επιθυμεί χειροκίνητα.

- Εκτός όμως από την απλή πληροφόρηση για συγκεκριμένα γεγονότα μια επιπλέον λειτουργία που θα μπορούσε να έχει η εφαρμογή είναι η δυνατότητα να προτείνει λύσεις για συγκεκριμένα συμβάντα (μέσω των notifications) ή ακόμα και να πραγματοποιεί αυτοματοποιημένο έλεγχο και δημιουργία σεναρίων τύπου (if-this-then-that).

- Η δυνατότητα παραμετροποίησης της κεντρικής οθόνης (homescreen) από τον χρήστη μέσω ενός menu ρυθμίσεων της εφαρμογής είναι ακόμα μια χρήσιμη λειτουργία για την εφαρμογή αφού ο κάθε χρήστης μπορεί να επιθυμεί να βλέπει διαφορετικά αντικείμενα στην αρχική οθόνη.

- Σε πολύ μελλοντικό στάδιο, με τη χρήση επιπλέον τεχνολογίας θα ήταν χρήσιμη η δυνατότητα ελέγχου συσκευών από την φορητή συσκευή ή ακόμα και αυτόματου ελέγχου τους με σκοπό την εξοικονόμηση ενέργειας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. U.S. Energy Information Administration, International Energy Outlook 2017
2. Dane, P.; Steele, J.; Wilkerson, J. WattBot: A Residential Electricity Monitoring and Feedback System. Proceedings of the CHI′09 Extended Abstracts on Human Factors in Computing Systems, Boston, MA, USA, 4–9 April 2009
3. European Commission. Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings (recast). Off. J. Eur. Union 2010, 18, 13–35
4. European Commission. Directive 2012/27/EU of the European Parliament and of the Council of 25 October 2012 on energy efficiency, amending Directives 2009/125/EC and 2010/30/EU and repealing Directives 2004/8/EC and 2006/32/EC Text with EEA relevance
5. Wei, Chuyuan & Li, Yongzhen. (2011). Design of energy consumption monitoring and energy-saving management system of intelligent building based on the Internet of things. 3650-3652. 10.1109/ICECC.2011.6066758.
6. Moreno MV, Úbeda B, Skarmeta AF, Zamora MA, 2014. How can We Tackle Energy Efficiency in IoT Based Smart Buildings? *Sensors (Basel, Switzerland)*. 14(6):9582-9614
7. M. Brenna, M.C. Falvo, F. Foiadelli, L. Martirano, F. Massaro, D. Poli, A. Vaccaro, 2012. Challenges in Energy Systems for the SmartCities of the Future, Energy Conference and Exhibition, IEEE International
8. Burt B., 2012. "Android Application Development All-In-One for Dummies, 8 books in one", Hoboken, NJ: John Wiley &Sons, Inc.
9. Cadenhead R., 2012."Sams Teach Yourself Java in 24 Hour". USA: Sams Publishing.
10. Deitel H. M., Deitel P. J., 2006. "Java Προγραμματισμός". Απόδοση: Χ.Α. Κουτρουμπά. Αθήνα: Μ. Γκιούρδας.
11. R. Elmasri, S. B. Navathe. "Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων" - 6η έκδοση.
12. Pressman R. S., "Τεχνολογία λογισμικού- Μία πρακτική προσέγγιση" - 7η έκδοση.
13. "Εισαγωγή στη γλώσσα προγραμματισμού Java". Σημειώσεις Εργαστηρίου Πολυμέσων, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ΕΜΠ.

14. "Εισαγωγή στις βάσεις δεδομένων". Σημειώσεις Εργαστηρίου Βάσεων Δεδομένων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Πατρών.

15. Cay S. Horstmann "Core Java, Volume I - Fundamentals" Tenth Edition, December 2015.

16. Downey, Allen B. (May 2012). Think Python: How to Think Like a Computer Scientist (Version 1.6.6 ed.)

17. Ken Arnold and James Gosling, *The Java Programming Language*, second ed., Addison-Wesley, 1998.

18. Patrick Chan, *The Java Developers Almanac*, Addison-Wesley, 1998.

19. Peter Coad and Mark Mayfield, *Java Design: Building Better Apps and Applets*, Yourdon Press, 1996.

20. David Flanagan, *Java in a Nutshell*, second ed., O'Reilly, 1996.

21. David Flanagan, *Java Foundation Classes in a Nutshell*, O'Reilly, 1999.

22. Graham Hamilton and Rick Cattell and Maydene Fisher, *JDBC Database Access with Java: A Tutorial and Annotated Reference*, SunSoft Press, 1997.

23. Elliotte Rusty Harold, *Java I/O*, O'Reilly, 1999.

24. Doug Lea, *Concurrent Programming in Java: Design Principles and Patterns*, Addison-Wesley, 1997.

25. George Reese, *Database Programming with JDBC and Java*, O'Reilly, 1997.

ΙΣΤΟΤΟΠΟΙ

26. Energy efficiency at buildings, Available online at: https://ec.europa.eu/energy/en/topics/energy-efficiency/buildings, last accessed: 18/10/2017
27. Android Developers website. Available online at: https://developer.android.com/index.html , last accessed 31/12/2017
28. Stack Overflow website.Available online at: https://stackoverflow.com/ , last accesed 12/11/2017
29. Github website: "MP Android Chart Library Documentation".Available online at: https://github.com/PhilJay/MPAndroidChart/wiki , last accessed 12/10/2017
30. Github website: "API XU Library".Available online at: https://github.com/apixu/apixu-java , last accessed 20/11/2017
31. Medium website: "Android AsyncTask HTTP GET request Tutorial". Available online at: https://medium.com/@JasonCromer/android-asynctask-http-request-tutorial-6b429d833e28 , last accessed 03/01/2018
32. TutorialsPoint website. Available online at: https://www.tutorialspoint.com/android , last accessed 15/10/2017
33. Android Arsenal website. Available online at: https://android-arsenal.com , last accessed 30/09/2017

34. Sourcey website: “Beautiful Android Login and Signup Screens with Material Design”. Available online at: “https://sourcey.com/beautiful-android-login-and-signup-screens-with-material-design/” , last accessed 29/10/2017
35. Miguel Grinberg Blog: “Designing a RESTful API with Python and Flask”. Available online at: https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask , last accessed 13/12/2017
36. Mykong website: “Android Tutorial”. Available online at: http://www.mkyong.com/tutorials/android-tutorial/ , last accessed 08/11/2017
37. PostgeSQL website: “PostgreSQL Documentation”. Available online at: https://www.postgresql.org/docs/ , last accessed 10/12/2017
38. Joda website: “Joda-Time Library Documentation”. Available online at: http://www.joda.org/joda-time/ , last accesed 30/09/2017
39. Clifton Labs website: “JSON-Simple Library". Available online at: https://cliftonlabs.github.io/json-simple/ , last accesed 30/09/2017
40. ΔΕΗ Α.Ε. website: "Οικιακά τιμολόγια". Available online at: https://www.dei.gr/el/oikiakoi-pelates/timologia , last accessed 06/12/2017.
41. Smartwatt website. Available online at: https://www.smartwatt.gr/el , last accessed 10/10/2017.
42. Energy Audit website. Available online at: http://energyaudit.gr/ , last accessed 11/10/2017.
43. Neurio website. Available online at: https://www.neur.io/ , last accessed 12/10/2017.
44. Sense website. Available online at: https://sense.com/ , last accessed 12/10/2017.