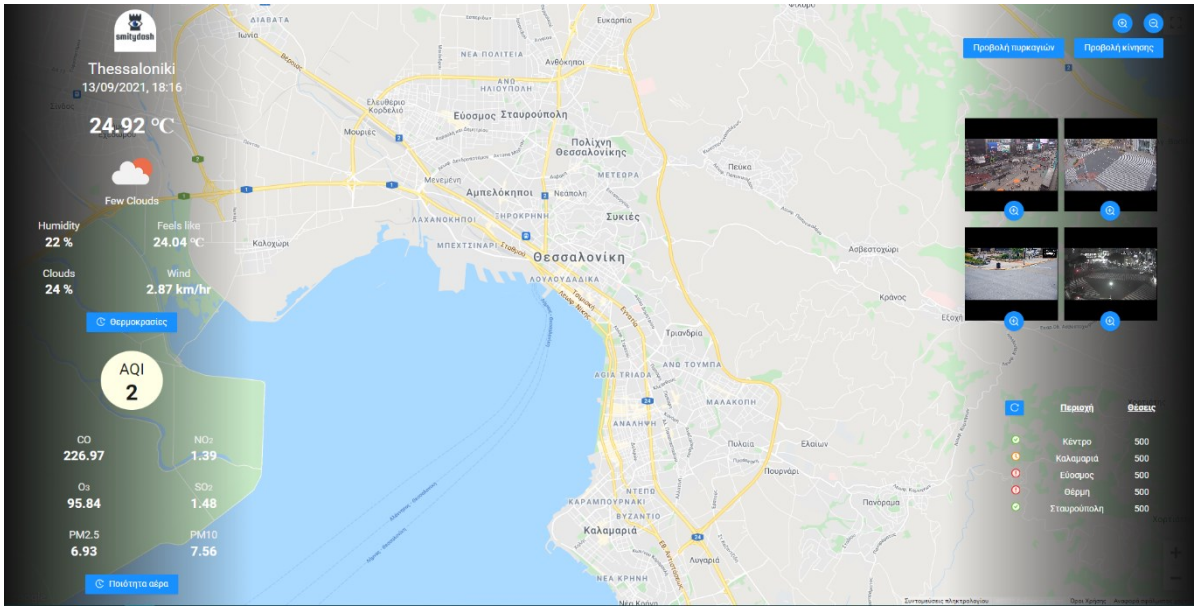
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία διαδικτυακής πλατφόρμας διαχείρισης και απεικόνισης δεδομένων για μια έξυπνη πόλη»



Της φοιτήτριας
Κρίσυ Ντούκα
Αρ. Μητρώου: 174979

Επιβλέπων
Περικλής Χατζημίσιος
Καθηγητής

Σεπτέμβριος 2021

Βεβαιώνω ότι είμαι η συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων των συγγραφέων, εκ μέρους του Τμήματος.

Περίληψη

Στην παρούσα πτυχιακή εργασία, εξετάζεται η έννοια των έξυπνων πόλεων, τα θετικά που προσφέρουν, οι πλατφόρμες που χρησιμοποιούνται ως εργαλεία για τη διαχείριση τους, οι τεχνολογίες που μπορούν να επιλεχθούν για τη δημιουργία μιας τέτοιας πλατφόρμας, όπως και η διαδικασία υλοποίησης της βήμα προς βήμα. Αρχικά, παρέχεται μια εισαγωγή, στην έννοια και τα πέντε βασικά χαρακτηριστικά μιας έξυπνης πόλης. Έπειτα, δίνονται οι λόγοι χρήσης ενός dashboard και παραδείγματα άλλων, τα οποία έχουν την ίδια λειτουργία με αυτό που υλοποιήθηκε σε αυτή την εργασία και αποτέλεσαν πηγή έμπνευσης για τη σχεδίαση του παρόντος dashboard. Στη συνέχεια, αναφέρονται τα χαρακτηριστικά που χρησιμοποιήσαμε και οι λόγοι που επιλέχθηκαν τα συγκεκριμένα. Επίσης, παρουσιάζεται μια ποικιλία από αρκετές τεχνολογίες που είτε απορρίφθηκαν είτε χρησιμοποιήθηκαν. Τέλος, παρατίθενται τα συμπεράσματα μας και κάποιες ιδέες για μελλοντικές επεκτάσεις των χαρακτηριστικών της πλατφόρμας.

Ο στόχος της πτυχιακής εργασίας, είναι να παραθέσει τα πλεονεκτήματα και τις δυνατότητες των έξυπνων πόλεων, που αποτελούν ιδανική λύση για πολλές ανάγκες των ανθρώπων και του περιβάλλοντος. Στα πλαίσια αυτής της εργασίας, μας δόθηκε η ευκαιρία να συνδυάσουμε σχετικά καινούργιες τεχνολογίες, σε δεδομένα τα οποία ανανεώνονται συνεχώς. Με βάση τα παραπάνω, αλλά και όλα όσα αναλύονται στη συνέχεια αυτού του γραπτού, συνεπάγεται πως οι άνθρωποι θα πρέπει να εξοικειωθούν με την εξέλιξη των παραδοσιακών εγκαταστάσεων, τεχνολογιών και υπηρεσιών των κοινωνιών μας και οι επιστήμονες θα πρέπει να συνεχίσουν απτόητοι το έργο τους, ώστε οι πόλεις στις οποίες ζούμε να γίνονται ένα καλύτερο μέρος.

Πρόλογος

Σκοπός μου στην παρούσα εργασία ήταν να κατανοήσω και να παρουσιάσω μέσω μιας πλατφόρμας, όλα τα δεδομένα που είναι θεμέλιοι λίθοι για την αποτελεσματική διαχείριση μιας έξυπνης πόλης του μέλλοντος. Μια έξυπνη πόλη, η οποία διέπεται κυρίως από τις δυνατότητες που της παρέχει το Διαδίκτυο των Πραγμάτων. Δυνατότητες, που σχηματίζουν τις βάσεις για ένα μέλλον που στο επίκεντρο όλων, βρίσκεται η ανάγκη για την καλυτέρευση του ανθρώπινου βίου, όπως και της κατάστασης του πλανήτη μας.

«Development of Web Platform for the Data Management and Representation of a Smart City Dashboard »

«Krisi Duka»

Abstract

This B.Sc. thesis examines, the concept of smart cities, the positives they offer, the platforms used as tools for their management, the technologies that can be selected to develop such a platform, as well as the process of its implementation step by step. First, an introduction to the concept and the five basic characteristics of a smart city is provided. Next, we give the reasons for using a dashboard and examples of others, which have the same function as the one implemented in this B.Sc. thesis and were the source of inspiration for the design of this dashboard. Moving on we noted the features we used and the reasons why they were chosen. It also presents a variety of several technologies that have either been rejected or used. Finally, our conclusions and some ideas for future extensions of the platform features are presented.

The aim of this B.Sc. thesis is to list the advantages and possibilities of smart cities, which are an ideal solution for many needs of people and the environment. As part of this thesis, we were given the opportunity to combine relatively new technologies, in data that is constantly updated. Based on the above, but also everything that is analyzed in the continuation of this dissertation, it implies that people should become familiar with the evolution of traditional facilities, technologies and services of our societies and scientists should continue their work fearlessly, so that the cities in which we live keep on becoming a better place.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον Ανδρέα, που αποτελεί το πολύτιμο στήριγμά μου όλων αυτόν τον καιρό. Επιπλέον, ένα ξεχωριστό ευχαριστώ στην οικογένεια μου, που με έκανε τον άνθρωπο που είμαι σήμερα και που είναι πλάι μου σε κάθε όνειρο μου. Εν κατακλείδι, θα ήθελα να ευχαριστήσω τον κύριο Περικλή Χατζημίσιο (Καθηγητή στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος) για τη στοχευμένη καθοδήγηση και τη διάθεση ώστε αυτή η εργασία να πετύχει όλους τους στόχους της.

Table of contents

Κατάλογος Εικόνων

1 Εισαγωγή

Στη παρουσίαση αυτής της εργασίας, που θα γίνει αναλυτικά στα επόμενα κεφάλαια, θα περιγράψουμε όσον το δυνατόν καλύτερα την ιδέα πίσω από τις έξυπνες πόλεις (Smart Cities) , τα χαρακτηριστικά τους, αλλά και τα εργαλεία για την εύρυθμη λειτουργία και διαχείριση αυτών των πόλεων.

Οι έξυπνες πόλεις είναι περιζήτητες πλέον, όμως για πολλούς μπορεί να παραμένει μία ξένη ιδέα και να τους φαντάζει κάτι πέρα από τις δυνατότητες των υπάρχοντών πόλεων. Μια έξυπνη πόλη, ουσιαστικά είναι μία πόλη με τις ήδη υπάρχουσες εγκαταστάσεις της, στην οποία ενσωματώνονται νέες τεχνολογίες για την εξέλιξη αυτών, όπως και των δυνατοτήτων που προσφέρουν (Εικόνα 1.1) [1] όπως περιγράφονται στο 2ο Κεφάλαιο.

Οι δυνατότητες που δίνουν, είναι στην ουσία όλα τα πλεονεκτήματα της εξέλιξης των χαρακτηριστικών που διαθέτει η κάθε πόλη, ως προς την καθημερινότητα των ανθρώπων και ως προς το περιβάλλον. Τα χαρακτηριστικά ποικίλουν και μπορεί να είναι, η χρήση συστημάτων τηλεϊατρικής, συστήματα εξοικονόμησης νερού και ενέργειας ή η παρακολούθηση της ποιότητας του αέρα. Τα χαρακτηριστικά που επιλέξαμε εμείς να εμφανίσουμε, περιγράφονται στο 3ο Κεφάλαιο.

Για την διαχείριση μίας τέτοιας πόλης απαιτούνται εργαλεία που θα δώσουν την δυνατότητα στον αρμόδιο ή στους αρμόδιους, να έχουν μία εικόνα όλων όσων συμβαίνουν στην πόλη κάθε στιγμή, ώστε να μπορούν να αποτρέψουν δυσάρεστα και επικίνδυνα γεγονότα, όπως μια πυρκαγιά (Εικόνα 1.2). Στην δική μας πλατφόρμα, τα χαρακτηριστικά που εμφανίζονται αφορούν τις πυρκαγιές της πόλης ή μιας πιο ευρείας περιοχής αν το επιθυμούμε, η κίνηση των οχημάτων στην πόλη με κάμερες που μεταδίδουν απευθείας εικόνα, αλλά και από τον κεντρικό χάρτη με κατάλληλες χρωματικές ενδείξεις, η παρακολούθηση της ποιότητας του αέρα όπως και του καιρού και τέλος τον αριθμό των διαθέσιμων θέσεων στάθμευσης που βρίσκονται σε κάθε περιοχή.

Βέβαια, τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την δημιουργία μιας πλατφόρμας με τέτοιο σκοπό, επιλέχθηκαν με κριτήρια την ικανότητα τους να ανταπεξέλθουν στην συνεχή ανανέωση των δεδομένων, στην ευκολία κατανόησης και χρήσης. Οι τεχνολογίες χωρίστηκαν και παρουσιάζονται ανά ενότητες. Δηλαδή, αρχικά περιγράφονται αυτές που χρησιμοποιήθηκαν στο backend και ύστερα, αυτές που χρησιμοποιήθηκαν για το frontend. Επιπλέον, αναφέρονται τα χαρακτηριστικά τεχνολογιών που θα μπορούσαν να χρησιμοποιηθούν για την υλοποίηση, αλλά τελικά δεν επιλέχθηκαν.

Στο 4ο κεφάλαιο, γίνεται πλήρης αναφορά της ολοκληρωτικής συγγραφής κώδικα και υλοποίησης της πλατφόρμας, Και πάλι η περιγραφή των προαναφερθέντων, έχει χωριστεί αρχικά στην υλοποίηση του backend και στην υλοποίηση του frontend.

Στο τελευταίο κεφάλαιο, γίνεται μία σύνοψη όλων όσων μελετήθηκαν και πραγματοποιήθηκαν στην παρούσα πτυχιακή εργασία, καταλήγοντας σε χρήσιμα συμπεράσματα για την χρησιμότητα των έξυπνων πόλεων και των dashboard τους.

Εικόνα 1.1: Η ευφυΐα που προσφέρεται στην πόλη μέσω των χαρακτηριστικών της [2]

Εικόνα 1.2: Οι πολλαπλές δυνατότητες διαχείρισης μέσω των dashboards [3]

2 Έξυπνες Πόλεις

2.1 Εισαγωγή στις έξυπνες πόλεις

Ο σπόρος για την ιδέα μιας έξυπνης πόλης, καλλιεργήθηκε από την ανάγκη και τη βαθύτατη επιθυμία των ανθρώπων να μπορούν να διαχειρίζονται με τον πιο οικονομικό και βέλτιστο τρόπο τις υπερσύγχρονες πόλεις τους. Αρχικά, η έννοια των έξυπνων πόλεων κάνει την πρώτη εμφάνιση της στο σύγγραμμα 'The Technopolis Phenomenon', των Gibson, Kozmetsky και Smilor (1993), τη δεκαετία του '90, οραματίζοντας τη δημιουργία ενός περιβάλλοντος, στο οποίο είναι δυνατή η σύνδεση νέων και διαφόρων τεχνολογιών μαζί τη συνεχή εξέλιξη των αστικών περιοχών [4].

Το παραπάνω όραμα μπορούμε πλέον να πούμε ότι έχει γίνει πραγματικότητα ή γίνονται τα βήματα ώστε να γίνει, αφού οι περισσότερες πόλεις του κόσμου είναι έξυπνες ή τείνουν να γίνουν. Αυτές οι σύγχρονες πόλεις λοιπόν, είναι ουσιαστικά αστικές περιοχές, οι οποίες βασίζονται στην ευφυΐα των τεχνολογιών. Κάνουν πλήρη χρήση αυτών των τεχνολογιών, όπως της τεχνητής νοημοσύνης και της μηχανικής μάθησης, με απώτερο σκοπό να υιοθετήσουν τις δυνατότητές τους. Δυνατότητες, που είναι η αιτία ύπαρξης του επιθέτου "έξυπνη" στη φράση έξυπνη πόλη, εφόσον αποπνέουν την αρχή της αυτόματης υπολογιστικής, όπως για παράδειγμα της μηχανικής σκέψης για αυτο-βελτιστοποίηση, αυτοδιάθεση, αυτοπροστασία και αυτοθεραπεία [5].

Τα χαρακτηριστικά που αποκτά μία έξυπνη πόλη από τις παραπάνω δυνατότητες, είναι πολύτιμα αφού υπάρχει ένας μεγάλος αριθμός απαιτήσεων που τις διέπει, λόγο του τεράστιου όγκου ποικιλόμορφων δεδομένων που προέρχονται από πολλές και διάφορες πηγές. Τα δεδομένα αυτά, προκύπτουν από τη διαδικασία διαχείριση της κατανάλωσης ενέργειας, της ποιότητας του αέρα, της κίνησης των πολιτών στη πόλη, του ελέγχου των πυρκαγιών και των καιρικών συνθηκών.

Κύριος στόχος λοιπόν για τις έξυπνες πόλεις, θα μπορούσαμε να πούμε είναι ο έλεγχος της ραγδαίας αστικής ανάπτυξης μέσα από ένα δίκτυο το οποίο διατηρεί τις απαραίτητες συνδέσεις ανάμεσα σε πολίτες, επιχειρήσεις και κυβερνητικούς φορείς συγκεντρώνοντας πληροφορίες οικονομικού, κοινωνικού και περιβαλλοντικού ενδιαφέροντος (Εικόνα 1.1) [4].

Εικόνα 2.1: Το περιεχόμενο ενός δικτύου μιας έξυπνης πόλης [6]

2.2 Χαρακτηριστικά έξυπνων πόλεων

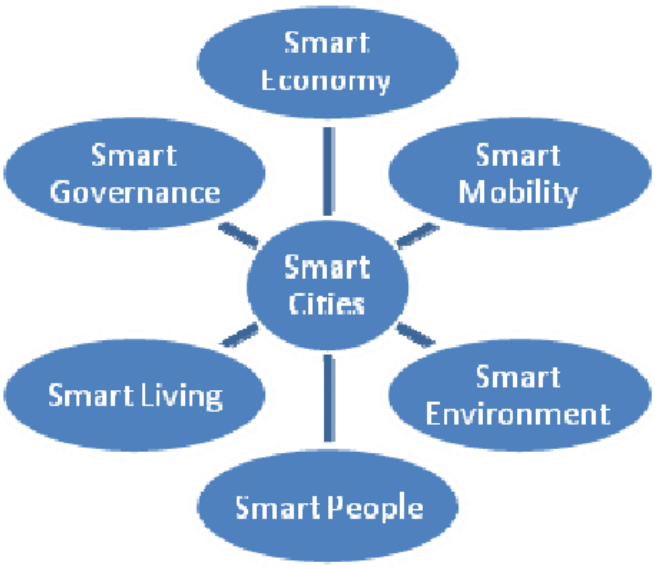
Ο ορισμός της έξυπνης πόλης που γνωρίζουμε σήμερα, έχει τις ρίζες της βαθιά ριζωμένες στο παρελθόν. Τον 19ο αιώνα πρωτοεμφανίστηκε ως η ιδανική πόλη και με το πέρασμα του χρόνου εξελίχθηκε σε έξυπνη πόλη [7]. Τι είναι όμως ουσιαστικά μια έξυπνη πόλη; Είναι μία πόλη, η οποία κάνει χρήση όλων των τεχνολογιών που της δίνονται, ώστε να αναπτυχθεί ένα περιβάλλον, στο οποίο θα προσφέρονται πλήρης και σωστά οργανωμένες υπηρεσίες για να λυθούν όσα προβλήματα μπορεί να δυσκολεύουν την ποιοτική και εύρυθμη ζωή των πολίτων της. Οι υπηρεσίες αυτές δημιουργούνται με βάση τις ανάγκες που υπάρχουν στην πολιτεία, τις οποίες καλούνται να καλύψουν [8].

Οι ανάγκες αυτές έχουν να κάνουν με βασικά συστατικά μιας πόλης, τα οποία στελεχώνουν τις βάσεις της. Καλύπτουν τους τομείς της οικονομίας, της μετακίνησης, της ποιότητας ζωής, της διακυβέρνησης, του περιβάλλοντος και πιο σημαντικό της νοοτροπίας των ανθρώπων (Εικόνα 2.2). Κάθε μία από τις υπηρεσίες αυτές, αποτελεί χαρακτηριστικό των έξυπνων πόλεων και θα αναλυθεί παρακάτω η σημασία εξέλιξης και καλυτέρευσης τους [7].

i. Οικονομία

Στις έξυπνες πόλεις, οι μέθοδοι εξοικονόμησης χρημάτων έχουν εξελιχθεί σε μεγάλο βαθμό λόγω των τεχνολογιών. Οι αρμόδιοι που διαχειρίζονται τις πηγές εσόδων και εξόδων φροντίζουν καθημερινά να βρίσκουν καινούργιους τρόπους ή να εξελίσσουν τους ήδη υπάρχοντες, ώστε με την ενσωμάτωση τους στην λειτουργία της πόλης, να αυξηθούν τα έσοδα ή να ελαττώσουν τα έξοδα της πόλης [7].

Παραδείγματα τέτοιων πόλεων είναι η Στσέτσιν στην Πολωνία, η οποία έχει τοποθετήσει ένα σύστημα έξυπνης διαχείρισης της φωταγώγησης των δρόμων της. Το σύστημα αυτό ανιχνεύει την κίνηση των πολιτών στους δρόμους και ενεργοποιεί τα φώτα, αλλιώς παραμένουν κλειστά (Εικόνα 2.3). Έτσι με την χρήση αυτού του συστήματος έχει μειωθεί η κατανάλωση ηλεκτρικής ενέργειας κατά 50% και κατά επέκταση έχει μειωθεί και το κόστος κατανάλωσης κατά 70% [10].



Εικόνα 2.2: Τα 6 χαρακτηριστικά μιας έξυπνης πόλης [9]

Εικόνα 2.3: Smart lighting [11]

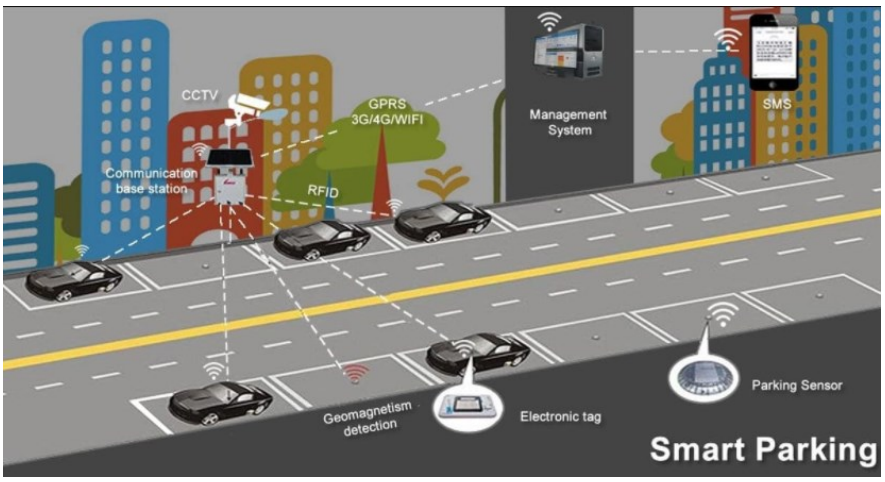
ii. Μετακίνηση

Η μετακίνηση στις πόλεις είναι ένα φλέγον θέμα για τους πολίτες. Η μεγάλη αύξηση πληθυσμού στις πόλεις, οι λίγες θέσεις στάθμευσης και η όχι σωστά οργανωμένη δημόσια μετακίνηση (π.χ. λεωφορεία, μετρό, τραμ) έχει προκαλέσει προβλήματα και καθημερινό στρες στους πολίτες.

Η σωστή οργάνωση και τα τακτικά δρομολόγια των δημόσιων μέσων μεταφοράς, μπορούν να αποτελέσουν ένα πολύ σημαντικό κομμάτι, ώστε οι άνθρωποι να έχουν την δυνατότητα να οργανώσουν με ευελιξία και χωρίς άγχος το καθημερινό τους πρόγραμμα, χωρίς να φοβούνται για καθυστερήσεις.

Επιπλέον, οι λιγοστές θέσεις στάθμευσης συγκριτικά με τα τόσα πολλά οχήματα που κυκλοφορούν στην πόλη απαιτούν άμεση και δραστική λύση. Η λύση είναι τα συστήματα τα οποία παρακολουθούν την διαθεσιμότητα και την εμφανίζουν μέσο κάποιας πλατφόρμας, ώστε οι ενδιαφερόμενοι να μπορούν να δουν τις ελεύθερες θέσεις και να κατευθυνθούν προς αυτές (Εικόνα 2.4).

Η Μπαρτσελόνα (Ισπανία), για παράδειγμα ύστερα από την εφαρμογή του προαναφερόμενου συστήματος για την διαχείριση των θέσεων στάθμευσής, κατάφερε να αυξήσει τα έσοδα που εισπράττει κάθε χρόνο από τέλη, κατά το αστρονομικό ποσό των 50 εκατομμυρίων [10].



Εικόνα 2.4: Σύστημα διαχείρισης θέσεων στάθμευσης [12]

iii. Ποιότητα ζωής

Η καλή και ποιητική ζωή των ανθρώπων έχει γίνει πλέον κατανοητό πως είναι από τα σημαντικά χαρακτηριστικά και απαιτεί συνεχή επιτήρηση και έρευνα για καινούργιους τρόπους και μεθόδους, που θα οδηγήσουν στην διατήρηση και στην αναβάθμιση του ήδη υπάρχοντος επιπέδου της.

Η ζωή των ανθρώπων περικλείεται από διάφορους τομείς όπως της υγείας, της εκπαίδευσης και της επιστήμης. Τομείς που όταν εξελίσσονται και παρέχονται με τον καλύτερο δυνατό τρόπο στους ανθρώπους οδηγούν σε μία κοινωνία, η οποία παρέχει στους κατοίκους την αίσθηση ασφάλειας αφού όλοι με την σωστή διαμόρφωση του χαρακτήρα που θα έχουν αποκτήσει μέσω της εκπαίδευσης, θα έχουν ενστερνιστεί τις αρχές και τους κανόνες της αρμονικής συμβίωσης με τους υπόλοιπους, χωρίς να παραβιάζει δηλαδή κανένας τα δικαιώματα του άλλου [7].

Επιπλέον, με την καθημερινή εξέλιξη της επιστήμης έχει επιτευχθεί ένα επίπεδο υγείας σε πολλές πόλεις, το οποίο καθημερινά αντιμετωπίζει όλο και περισσότερες ασθένειες που ανησυχούν και εμποδίσουν τους ανθρώπους να ζήσουν μια ξέγνοιαστή και ήρεμη ζωή [7].

iv. Διακυβέρνηση

Η διακυβέρνηση και οι άνθρωποι πίσω από αυτήν είναι εξαιρετικά σημαντικά για μία πόλη. Από την διακυβέρνηση εξαρτάται η εύρυθμη λειτουργία μιας κοινωνίας, όπως και οι κανόνες της (Εικόνα 2.6). Για αυτό οι άνθρωποι που έχουν αυτή την εξουσία οφείλουν να είναι ειλικρινής, με σχέδια για ένα καλύτερο μέλλον και αρκετά δραστήριοι για να τα εκπληρώσουν.

Επιπλέον, η διαχείριση της πόλης και οι μέθοδοι εξέλιξης που επιλέγουν πρέπει να είναι καινοτόμες, να ισορροπούν το απόθεμα των χρημάτων όπως και τις υπηρεσίες B2C και B2B. Τέλος, πρέπει να μειώσουν τους λόγους διαμάχης (π.χ. περιουσίες, υπηρεσίες, βιοτικό επίπεδο) μεταξύ των πολιτών και να τους δίνουν εναύσματα για να τους φέρουν πιο κοντά, με αποτέλεσμα να δημιουργήσουν μια κοινωνία που θα λειτουργεί ως μια ενωμένη ομάδα [7].

Εικόνα 2.5: Συστατικά μιας ποιοτικής ζωής στις έξυπνες πόλεις [13]

Εικόνα 2.6: Ευφυής διακυβέρνηση [14]

v. Περιβάλλον

Το περιβάλλον αποτελεί πολύτιμο αγαθό για τους ανθρώπους, αλλά και για όλα τα ζωντανά πλάσματα του πλανήτη. Προσφέρει στέγη, οξυγόνο και τροφή σε όλους μας. Βέβαια μας δίνει και πολλά άλλα, όπως τα δάση που μας προστατεύουν από πλημμύρες και τα ποτάμια που μας δίνουν πόσιμο νερό.

Για να συνεχίσουμε να απολαμβάνουμε όλα αυτά τα προνόμια που μας δίνει το περιβάλλον μας, θα πρέπει και εμείς να το προστατέψουμε και να το εξελίξουμε παράλληλα με τις έξυπνες πόλεις, χωρίς όμως τα παρεμβαίνομε στους κανόνες του.

Για να κάνουμε λοιπόν το περιβάλλον μας πιο έξυπνο αλλά με σεβασμό πάντα προς αυτό, μπορούμε να ελέγχουμε την ποιότητα του αέρα και όπου η μόλυνση ξεπερνάει κάποια επιτρεπτά όρια που έχουν θέσει οι επιστήμονες μετά από έρευνες, να λαμβάνουμε τα κατάλληλα μέσα ώστε να την μειώσουμε (Εικόνα 2.7) [7].

Τέλος, κάποια από τα μέσα αυτά, θα μπορούσε να είναι η μείωση καύσης καυσίμων, η χρήση συστημάτων ΤΠΕ για πιο ευέλικτη διαχείριση των απορριμμάτων της πόλης, χρήση ανανεώσιμων πηγών ενέργειας, ανακύκλωση και σωστή διαχείριση υδάτων με την βοήθεια αυτοματισμών και πληροφοριακών συστημάτων [15].

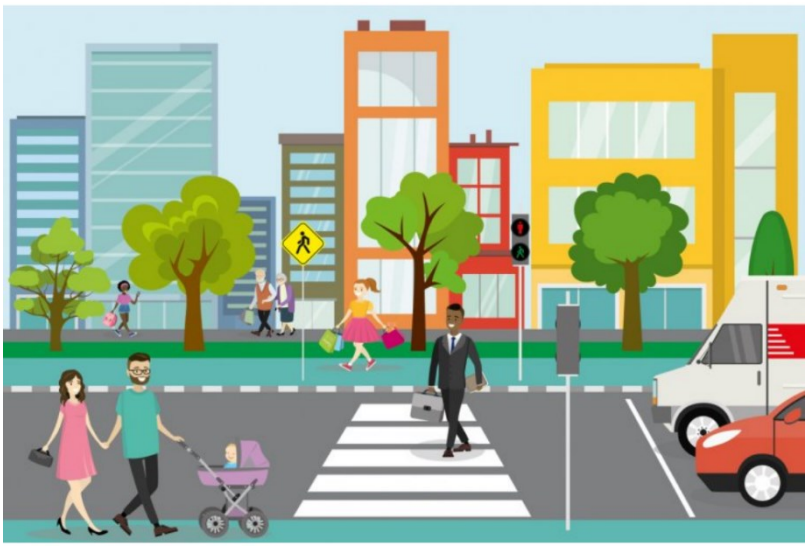
Εικόνα 2.7: Ευφυές περιβάλλον [15]

vi. Νοοτροπία

Τελευταίο και πιο βασικό συστατικό μιας έξυπνης πόλης, είναι η νοοτροπία που διακατέχεις τους πολίτες της. Η νοοτροπία των ανθρώπων που συμβιώνουν σε τέτοιες πόλεις, πρέπει να συμβαδίζει με τους κοινά αποδεκτούς κανόνες και να σέβεται όλους τους άλλους τριγύρω του, όπως και το περιβάλλον στο οποίο ζει.

Οι άνθρωποι αυτοί λοιπόν, πρέπει να είναι ανοιχτόμυαλοι, να δέχονται δηλαδή το διαφορετικό και το καινούργιο χωρίς να το προδικάζουν ή να το απομονώνουν. Οφείλουν να είναι ενεργοί στα κοινά θέματα, δηλαδή να προτάσσουν καινούργιες ιδέες που πιστεύουν θα εξελίξουν την πόλη ή τους ίδιους και τυχόν προβλήματα της κοινωνίας να τα επικοινωνούν στις αρμόδιες αρχές, ώστε να βρίσκεται πιο εύκολα και γρήγορα λύση [7].

Ακόμα, θα πρέπει να είναι πρόθυμοι και σε ετοιμότητα συνέχεια, να εκλαμβάνουν καινούργιες γνώσεις και εμπειρίες για διάφορα θέματα. Τέλος, πρέπει να έχουν απελευθερώσει το μυαλό τους από παλαιές εθνικιστικές απόψεις και να είναι ανοιχτοί, χωρίς ταμπού σε όλους τους ανθρώπους ανεξαιρέτως καταγωγής, χρώματος, θρησκείας ή σεξουαλικότητας [7].



Εικόνα 2.8: Οι πολίτες των έξυπνων πόλεων [16]

2.3 Οι έξυπνες πόλεις του πλανήτη μας

Οι έξυπνες πόλεις του πλανήτη μας, τα τελευταία χρόνια αυξάνονται και πληθαίνουν σε αριθμό συνέχεια. Σε κάθε μία πόλη, προστίθεται κάθε φορά ένα καινούργιο χαρακτηριστικό, το οποίο αυξάνει την ευφυΐα της και κατ' επέκταση τις υπηρεσίες που διαθέτει στους πολίτες. Τα χαρακτηριστικά αυτά, αφορούν τους τομείς της οικονομίας, του περιβάλλοντος, της ποιότητας ζωής, της νοοτροπίας των ανθρώπων, της μετακίνησης, της διακυβέρνησης, και αναλύθηκαν περισσότερο στην προηγούμενη υποενότητα.

2.4 Οι έξυπνες πόλεις της Ελλάδας

Τέτοιες πόλεις στην Ελλάδα αριθμούμε πέντε μέχρι σήμερα. Αυτές οι πόλεις είναι η Λάρισα, τα Τρίκαλα, το Ηράκλειο Κρήτης, η Βέροια και τα Ιωάννινα [17]. Στη συνέχεια θα αναλυθούν τι χαρακτηριστικά διαθέτουν, τα οποία τις κατατάσσουν στην κατηγορία ων έξυπνων πόλεων.

Η Λάρισα διαθέτει συστήματα που αφορούν την υγεία και την πολεοδομία. Πιο συγκεκριμένα, προσφέρει υπηρεσίες τηλεϊατρικής, ένα σύστημα επικοινωνίας με ειδικούς, για όσους πάσχουν από κατάθλιψη ή αλτσχάιμερ και ένα άλλο σύστημα που δίνει την δυνατότητα πρόσβασης μέσω Wi-Fi στην πολεοδομία της πόλης (Εικόνα 2.9) [17].

Οι κάτοικοι των Τρικάλων έχουν πρόσβαση σε υπηρεσίες της τηλεϊατρικής, του οπτικού δακτυλίου, των ευφυή μεταφορών και της αναζήτησης χαμένων κατοικίδιων. Στα μελλοντικά σχέδια της πόλης βρίσκονται συστήματα για τον έλεγχο κυκλοφορίας, της τηλεματικής, των καυσίμων και της έξυπνης διαχείρισης και συλλογής των απορριμμάτων της πόλης (Εικόνα 2.10) [17].

Τα Ιωάννινα παρέχουν υπηρεσίες που διευκολύνουν την καθημερινότητα των πολιτών αλλά και των τουριστών. Αναλυτικότερα, ξεχωριστά από το ασύρματο δίκτυο που έχει, προσφέρει την δυνατότητα οι άνθρωποι να συμπληρώνουν ηλεκτρονικά τις αιτήσεις τους για πιστοποιητικά που επιθυμούν, να ξεναγούνται στην πόλη διαδικτυακά, εφόσον έχουν ψηφιοποιηθεί τα εκθέματα του Δημοτικού Μουσείου και το περιεχόμενο της βιβλιοθήκης του δήμου (Εικόνα 2.11) [17].

Η Βέροια τα τελευταία χρόνια έχει κάνει τα πρώτα βήματα προς την ευφυΐα. Το χαρακτηριστικό που την θέτει έξυπνη πόλη, είναι Δημόσια Κεντρική βιβλιοθήκη του δήμου. Τα μελλοντικά σχέδια της, ώστε να προσθέσει περισσότερα παρόμοια χαρακτηριστικά, είναι οι αύξηση υπηρεσιών σε 10.000 κατοίκους και επαγγελματίες (Εικόνα 2.12) [17].

Τέλος, το Ηράκλειο Κρήτης έχει δώσει βάση κυρίως στις υπηρεσίες που διαθέτει. Δηλαδή, προσφέρει 163 υπηρεσίες για παροχή πληροφοριών, 29 για ηλεκτρονικές συμπληρώσεις αιτήσεων και πληρωμών, ψηφιοποιημένες εφημερίδες και free Wi-Fi στους δρόμους της πόλης (Εικόνα 2.13) [17].

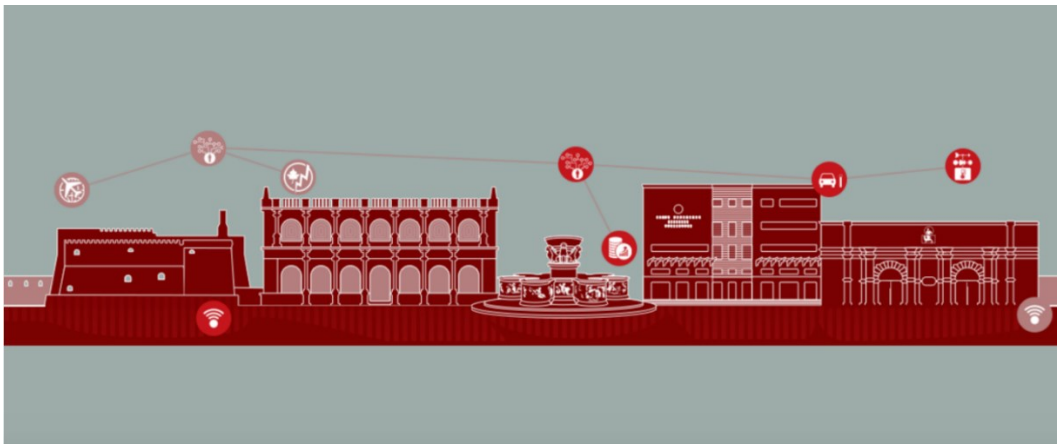
Από τα χαρακτηριστικά αυτών των πέντε πόλεων, γίνεται κατανοητό πως έχει γίνει ένα καλό ξεκίνημα προς την κατεύθυνση των ευφυή πόλεων, όμως έχουμε αρκετό περιθώριο βελτίωσης των πόλεων μας και κατ' επέκταση της καθημερινότητας μας.

Εικόνα 2.9: Σχέδιο έξυπνης πόλης – Λάρισα [18]

Εικόνα 2.10: Τα χαρακτηριστικά της έξυπνης πόλης – Τρίκαλα [19]

Εικόνα 2.11: Σχέδιο έξυπνης πόλης – Ιωάννινα [20]

Εικόνα 2.12: Σχέδιο έξυπνης πόλης - Βέροια [21]



Εικόνα 2.13: Σχέδιο έξυπνης πόλης – Ηράκλειο [22]

2.4.1 Οι έξυπνες πόλεις παγκοσμίως

Παγκοσμίως, οι έξυπνες πόλεις αριθμούνται 109 [23]. Παρακάτω, θα παρουσιαστούν τα χαρακτηριστικά κάποιων κορυφαίων σε ευφυΐα πόλεων του πλανήτη, οι οποίες είναι το Άμστερνταμ, το Χονγκ Κόνγκ, το Τορόντο, το Σεούλ, η Σιγκαπούρη, η Ρέικιαβικ, το Τόκυο, το Παρίσι, το Λονδίνο και η Νέα Υόρκη [24].

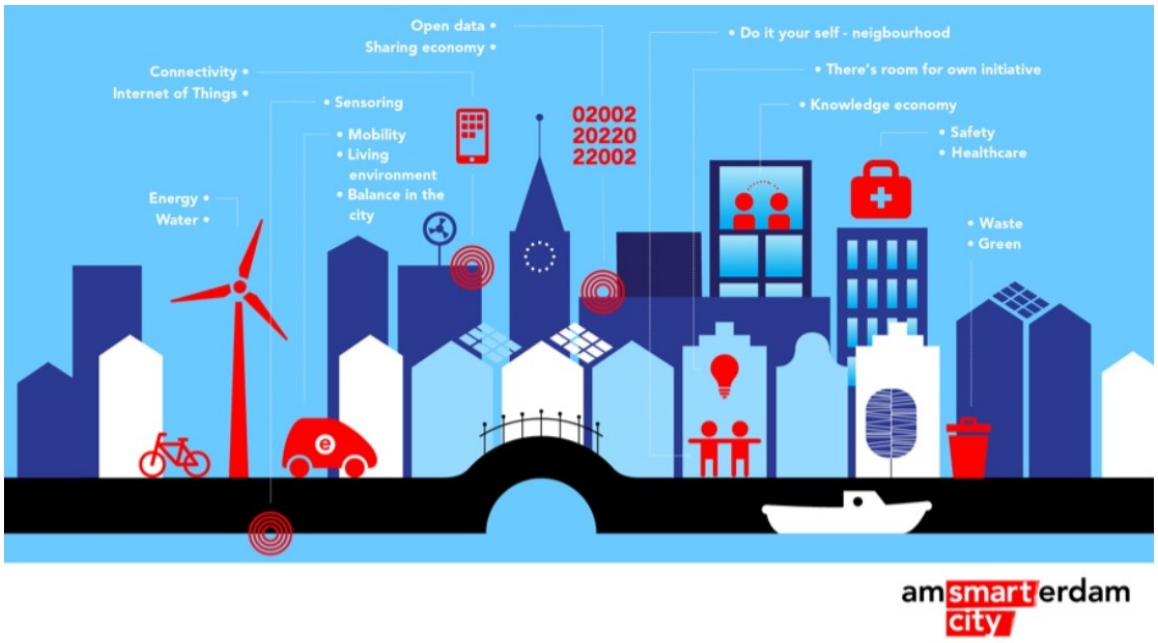
Το Άμστερνταμ, έχει αξιολογηθεί με βάση τα στοιχεία που παραθέτει ο IESCE Cities in Motion Index, στην 10η θέση των καλύτερων έξυπνων πόλεων παγκοσμίως και 4η στη Δυτική Ευρώπη. Έχει καταφέρει να φτάσει τόσο ψηλά στην λίστα των αξιολογήσεων, εφόσον διαθέτει υψηλή τεχνολογία για την οποία βραβεύτηκε ως η 3η καλύτερη πόλη, για τις προβολές της σε διεθνές επίπεδο και για το πολεοδομικό της σχεδιασμό (Εικόνα 2.14) [24].

Η Σεούλ έχει ορισθεί, ως η 7η πιο έξυπνη πόλη σε όλο τον κόσμο. Μέχρι το 2020 η διακυβέρνηση της πόλης, είχε ανακοινώσει πως θα έχουν εγκατασταθεί σε όλη την πρωτεύουσα 50.000 IoT συσκευές, οι οποίες θα συλλέγουν δεδομένα και πληροφορίες για την σκόνη, την κίνηση στους δρόμους της πόλης και για όλα τα θέματα που απασχολούν τις ζωές των πολιτών. Φέτος τα σχέδια της πόλης για την επέκταση της ευφυΐας, αφορούν την υπηρεσία για τα ελεύθερες δημόσιες θέσεις στάθμευσης (Εικόνα 2.15) [24].

Η Ρέικιαβικ είναι η πρωτεύουσα της Ισλανδίας και κατατάσσεται στην 5η θέση των πιο έξυπνων πόλεων. Στην διαδικασία εξέλιξης της πόλης τους η διακυβέρνηση εμψύχωσε μέσο του φόρουμ Better Reykjavik, τους πολίτες να εκφράσουν και να παρουσιάσουν τις ιδέες τους για μελλοντικές υπηρεσίες που θα θέλανε. Επιπλέον, η διακυβέρνηση προώθησε πριν λίγο καιρό μία εφαρμογή, η οποία ενημερώνει τον χρήστη για την δημόσια συγκοινωνία της πόλης (π.χ. αστικά λεωφορεία) και τις διαδρομές που θα μπορούν να ακολουθήσουν για να φτάσουν στον προορισμό τους. Η πόλη αυτή, βραβεύτηκε για πολλούς λόγους (ένας από αυτούς είναι ο προηγούμενος που αναφέρθηκε), κορυφαία στην κατηγορία της περιβαντολογικής πρωτοβουλίας (Εικόνα 2.16) [24].

Προτελευταία στην λίστα των 10 κορυφαίων πόλεων παγκοσμίως, είναι το Λονδίνο βάση του IESE. Οι περισσότεροι πολίτες που κατοικούν και ζουν στο Λονδίνο προέρχονται από ποικίλες χώρες του κόσμου και αυτός είναι ένας λόγος για τον οποίο αναγνωρίστηκε ως η κορυφαία πόλη στον ανθρωπιστικό τομέα. Επιπλέον, ευφυής χαρακτηριστικά της πόλης αυτής αφορούν τις μεταφορές, την διεθνή προσέγγιση και αναγνωρισιμότητα, την οικονομία, τη διακυβέρνηση, την τεχνολογία και τον πολεοδομικό σχεδιασμό της (Εικόνα 2.17) [24].

Τέλος, στην κορυφή της λίστας βρίσκεται η Νέα Υόρκη. Μια από τις μεγαλύτερες πόλεις του κόσμου, η οποία αριθμεί περισσότερους από 8,5 εκατομμύρια κατοίκους. Στα πλαίσια του σχεδίου έξυπνης πόλης, το τμήμα προστασίας του περιβάλλοντος αναπτύσσει ένα σύστημα το οποίο καταγράφει αυτόματα την κατανάλωση του νερού ημερησίως. Το σύστημα αυτό είναι απαραίτητο στην διαχείριση μια τέτοιας πόλης, λόγο του υπερπληθυσμού της, αφού καθημερινά χρησιμοποιείται 1 δισεκατομμύριο γαλόνια νερό. Επιπλέον, ένα ακόμα καινούργιο χαρακτηριστικό της πόλης είναι η έξυπνη διαχείριση και συλλογή των απορριμμάτων της πόλης (Εικόνα 2.18) [24].



Εικόνα 2.14: Σχέδιο έξυπνης πόλης – Άμστερνταμ [25]

Εικόνα 2.15: Έξυπνη πόλη – Σεούλ [26]

Εικόνα 2.16 : Σχέδιο έξυπνης πόλης – Ρέικιαβικ [27]

Εικόνα 2.17 : Σχέδιο έξυπνης πόλης – Λονδίνο [28]

Εικόνα 2.18: Σχέδιο έξυπνης πόλης – Νέα Υόρκη [29]

3 Χαρακτηριστικά των dashboards

3.1 Τι είναι ένα dashboard και ποια είναι τα χαρακτηριστικά του

Την επίσημη εμφάνιση του έκανε ο όρος "Dashboard", την δεκαετία του '90 στο Oxford λεξιλόγιο, με τον ορισμό "screen giving a graphical summary of various types of information, typically used to give an overview of (part of) a business organization." [30], σε ελεύθερη μετάφραση σημαίνει, μία οθόνη στην οποία απεικονίζονται διαφόρων τύπων πληροφορίες, για να δώσουν μια σφαιρική εικόνα ή την μερική εικόνα ενός οργανισμού.

Ένα dashboard θα μπορούσε να χαρακτηριστεί και ως ένα εργαλείο, το οποίο κύρια λειτουργία του είναι η παρουσίαση διαφόρων πληροφοριών, το οποίο και κατ' επέκταση δίνει τη δυνατότητα αλληλεπίδρασης του χρήστη με αυτές, ώστε να μπορούν να παρθούν αποφάσεις για όλα τα θέματα μιας πόλης, μιας εταιρίας ή οτιδήποτε άλλο διαχειρίζεται μέσω αυτού του εργαλείου [31].

Στην παρούσα υποενότητα συλλέχθηκαν και θα παρουσιαστούν τα πιο σημαντικά χαρακτηριστικά που χρησιμοποιούν, διάφορα ανά τον κόσμο dashboard. Τα χαρακτηριστικά τους, είναι όλα όσα απεικονίζονται στην οθόνη όταν προβάλλεται το κάθε dashboard, αλλά και όλες οι δυνατότητες αλληλεπίδρασης στα δεδομένα (π.χ. διαγράμματα).

Βασικός σκοπός μέσω των dashboard λοιπόν, είναι τα δεδομένα που λαμβάνουν, να τα μετατρέπουν σε πληροφορία έπειτα από τις κατάλληλες επεξεργασίες που απαιτούνται (π.χ. μορφοποίηση και τοποθέτηση σε διαγράμματα ή πίνακες κτλ.), οι χρήστες να εισπράττουν γνώση από αυτά, ώστε να μπορούν αργότερα να λαμβάνουν τις σωστές αποφάσεις (Εικόνα 3.1) [32].

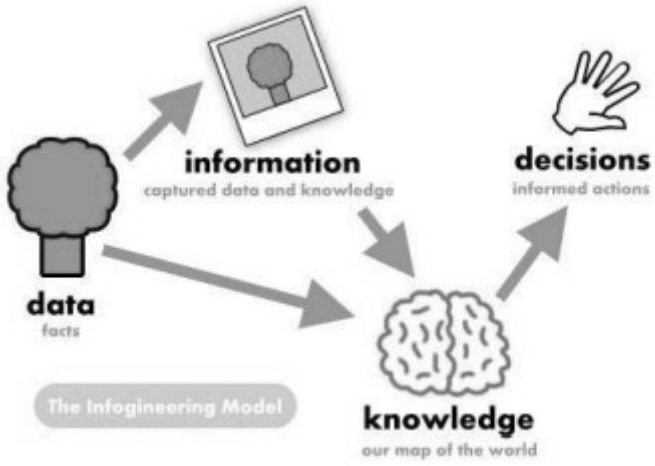
Τα χαρακτηριστικά, δηλαδή τα δεδομένα που παρουσιάζονται γενικά στα περισσότερα dashboard αφορούν διάφορους τύπους του χάρτη της πόλης ή μιας πιο ευρύτερης περιοχής, την ποιότητα του αέρα, τον καιρό (θερμοκρασία, υγρασία κτλ.), την κίνηση στην πόλη, την κατάσταση που επικρατή στα μέσα μαζικής μεταφοράς, την υγεία της κοινωνίας (π.χ. κρούσματα κορονοϊού) και την ενεργειακή κατάσταση της πόλης (π.χ. ποσότητα ενέργειας που παράγεται από αιολικά πάρκα) [33].

Στην Εικόνα 3.2, απεικονίζεται ένα dashboard με κάποια από τα προαναφερόμενα χαρακτηριστικά. Περιέχει τα πιο σημαντικά χαρακτηριστικά για την καλή διαχείριση μιας έξυπνης πόλης και αποτελεί ένα μοντέλο για την ανάπτυξη μελλοντικών dashboard διαχείρισης.

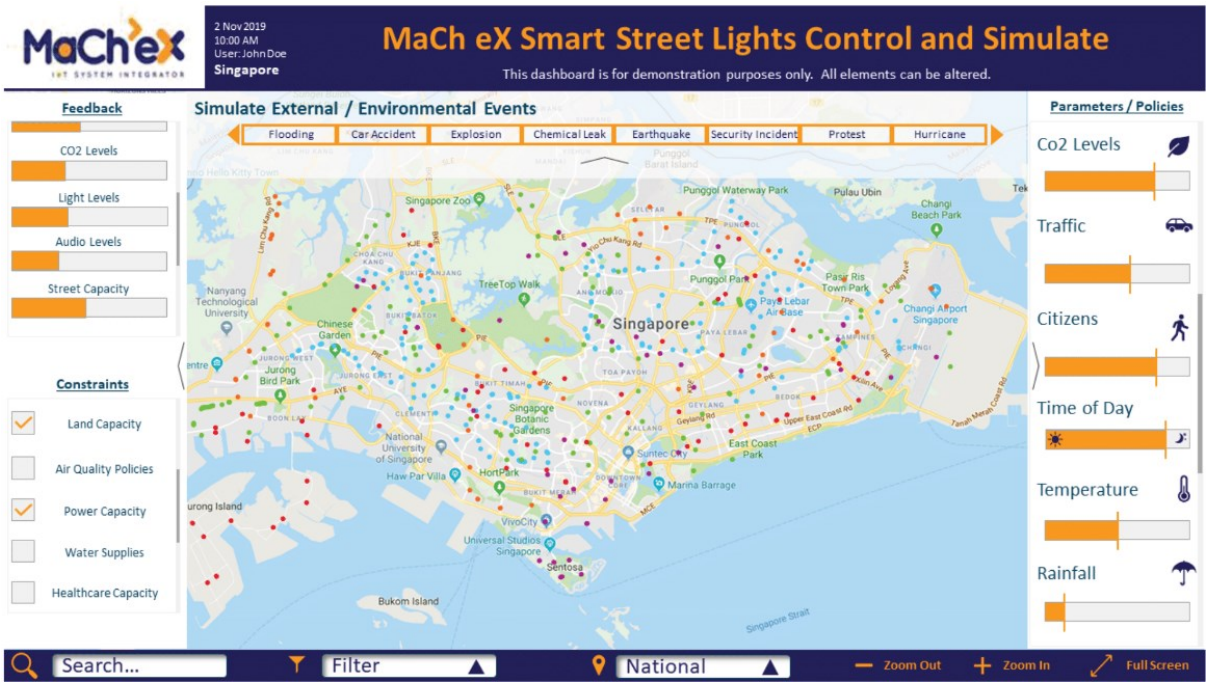
Παραδείγματα τέτοιων dashboard που έχουν υλοποιηθεί και χρησιμοποιούν τα παραπάνω χαρακτηριστικά, είναι του Amsterdam (Εικόνα 3.3), των Τρικάλων (Εικόνα 3.4) και του Ηρακλείου, Κρήτης (Εικόνα 3.5).

Τέλος, υπάρχουν και πλατφόρμες όπως η Km4City (Εικόνα 3.6), η οποία μία από τις υπηρεσίες που προσφέρει είναι να δίνει έμπνευση σε όσους επιθυμούν να δημιουργήσουν το δικό τους dashboard, προβάλλοντας διάφορες μορφές και είδη. Για παράδειγμα πλατφόρμες με κύριο θέμα κάποια βασικά χαρακτηριστικά της πόλης (Εικόνα 3.7), τις ενεργειακές πηγές της πόλης (Εικόνα 3.8) ή το περιβάλλον (Εικόνα 3.9). Τα δεδομένα που χρησιμοποιούνται περισσότερο σε αυτά τα μοντέλα, είναι επί των πλείστων, όσα αναφέραμε και θα αναλύσουμε εκτενέστερα στις παρακάτω υποενότητες.

Παρακάτω, θα γίνει πλήρης παρουσίαση των χαρακτηριστικών που χρησιμοποιήθηκαν στην συγκεκριμένη εργασία. Κατόπιν έρευνας, καταλήξαμε στα συγκεκριμένα χαρακτηριστικά, διότι δίνουν την δυνατότητα στον διαχειριστή μιας τέτοιας πλατφόρμας, να δημιουργήσει μια ολοκληρωμένη άποψη για την πόλη.

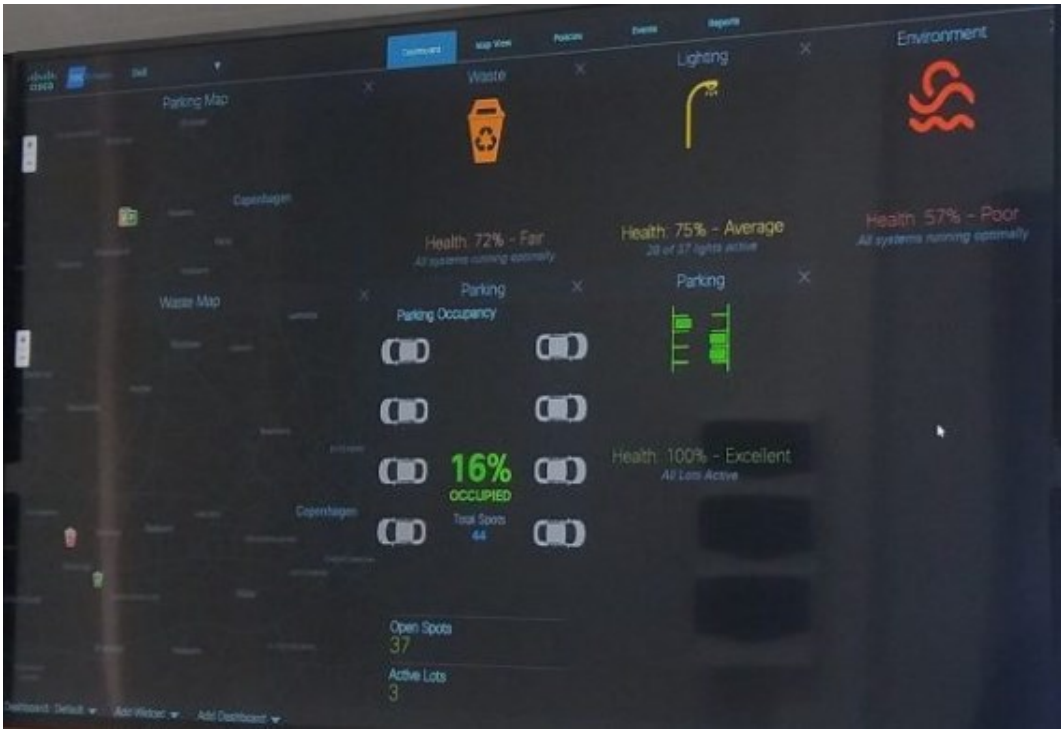


Εικόνα 3.1: Δεδομένα, πληροφορία, γνώση και απόφαση [32]



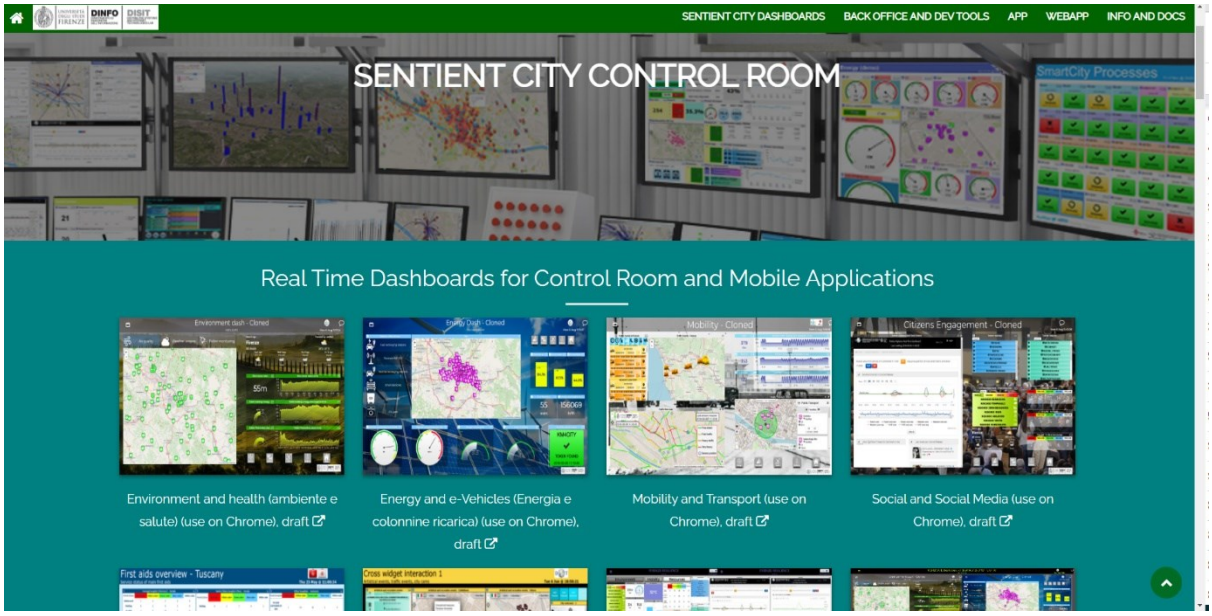
Εικόνα 3.2: Προτεινόμενο μοντέλο dashboard [30]

Εικόνα 3.3: Το dashboard διαχείρισης του Άμστερνταμ [34]

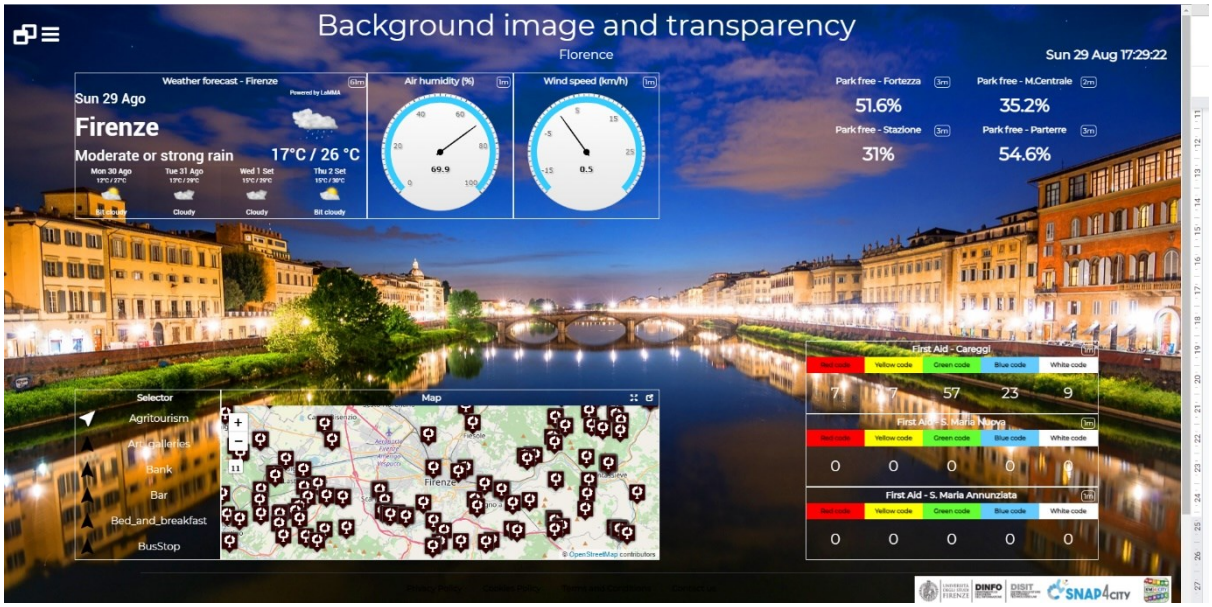


Εικόνα 3.4: Το dashboard διαχείρισης των Τρικάλων [35]

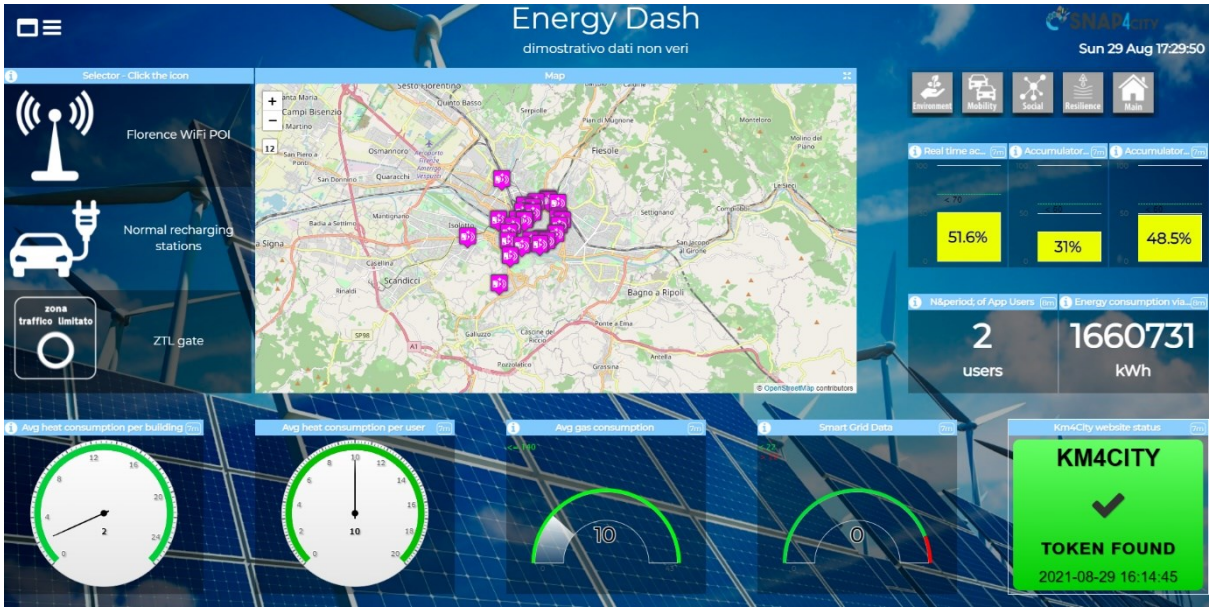
Εικόνα 3.5: Το dashboard διαχείρισης του Ηρακλείου, Κρήτης [36]



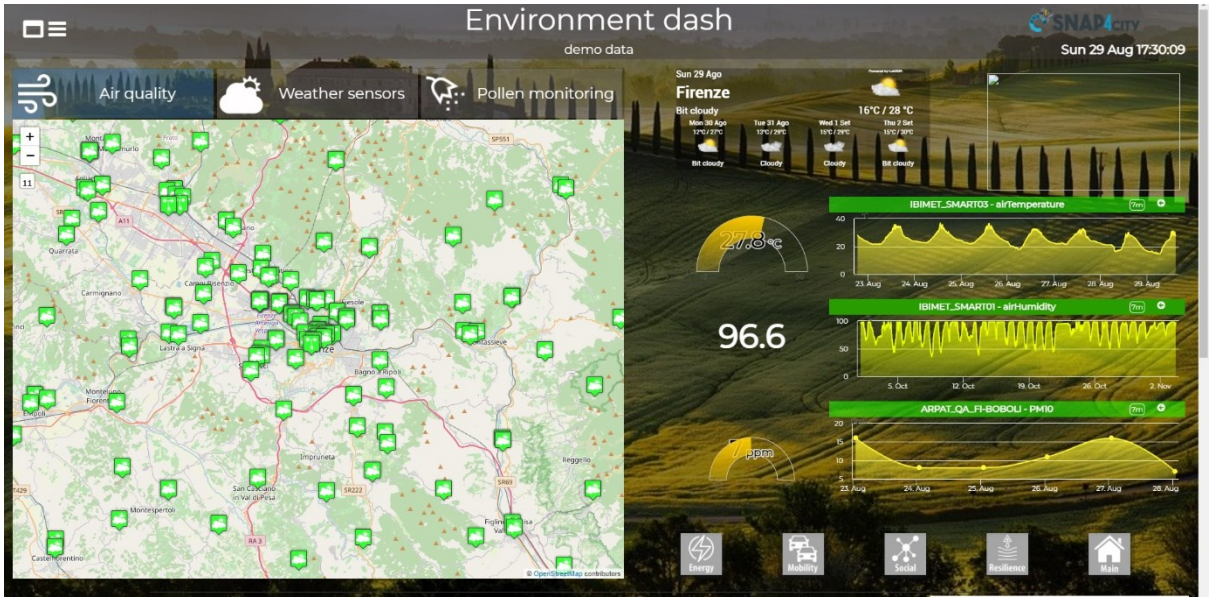
Εικόνα 3.6: Η πλατφόρμα Km4City [37]



Εικόνα 3.7: Προτεινόμενο dashboard από την Km4City [37]



Εικόνα 3.8: Προτεινόμενο dashboard από την Km4City [37]



Εικόνα 3.9: Προτεινόμενο dashboard από την Km4City [37]

3.1.1 Καιρός

Ο καιρός είναι υπεύθυνος για τον έλεγχο της διαχείρισης του νερού από περιοχή σε περιοχή, γεγονός που το θέτει πολύ σημαντικό για την επιβίωση των ζωντανών οργανισμών στη γη. Τι είναι όμως ο καιρός κι έχει μια τόσο ισχυρή ικανότητα στον έλεγχο της διαχείρισης του νερού στον πλανήτη μας;

Ο καιρός είναι η κατάσταση της ατμόσφαιρας για ένα συγκεκριμένο χρόνο και μια περιοχή, σε σχέση πάντα με την εποχή (Εικόνα 3.11), το κλίμα, τη θερμότητα, το χιόνι, τη βροχή, τον ήλιο, τη συννεφιά και την ξηρότητα. Κάποιοι από τους κυριότερους τύπους του καιρού είναι ο ηλιόλουστος, ο συννεφιασμένος, ο βροχερός, ο χιονισμένος, ο νεφελώδης και ο μερικώς νεφελώδης (Εικόνα 3.10) [38].

Επιπλέον, είναι ουσιώδες να κατανοήσουμε πως ο καιρός είναι κάτι διαφορετικό από το κλίμα, αλλά όχι ανεξάρτητο του. Το κλίμα επιγραμματικά είναι οι καιρικές συνθήκες που επικρατούν κατά μέσο όρο σε μια περιοχή για περίπου 30 χρόνια [39].

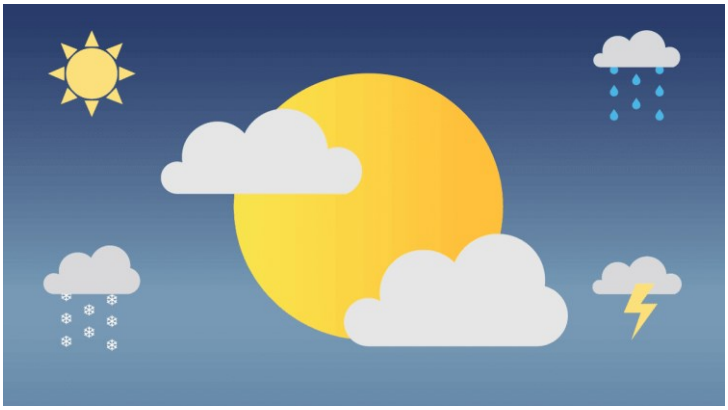
Επομένως, είναι ένα στοιχείο που προσδίδει πολλά πλεονεκτήματα όταν ενσωματωθεί σε πλατφόρμες διαχείρισης πόλεων, εφόσον δίνει την δυνατότητα παρακολούθησης της εξέλιξης των καιρικών συνθηκών για μια συγκεκριμένη περιοχή και την ενημέρωση για αποφυγή επικίνδυνων καταστάσεων (π.χ. αύξηση της θερμοκρασίας από 40˚C έως 50˚C/καύσωνας, χαλάζι κτλ.) [44].

3.1.2 Ποιότητα Αέρα

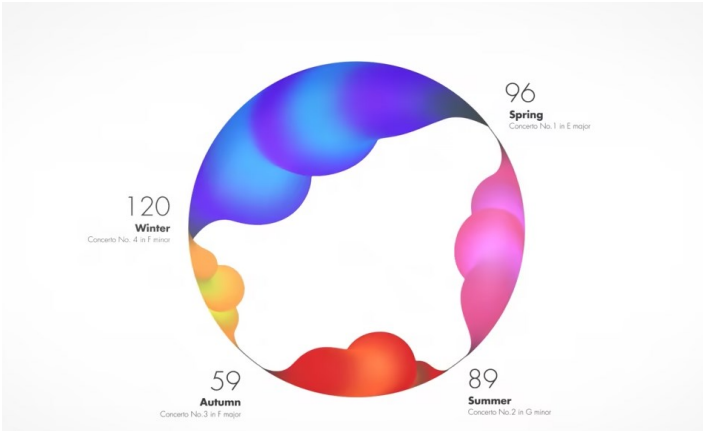
3.1.3 Εισαγωγή

Η ρύπανση της ατμόσφαιρας, είναι ένα φλέγον περιβαλλοντικό θέμα για τους ειδικούς, αλλά και για τους απλούς πολίτες όλων των χωρών του πλανήτη μας, λόγω της μεγάλης τα τελευταία χρόνια, εκβιομηχάνισης. Η ρύπανση της ατμόσφαιρας έχει άμεση επαφή με την ποιότητα του αέρα που αναπνέουμε καθημερινά και κατ' επέκταση με τις επιπτώσεις στην υγεία των ανθρώπων.

Η ανάγκη λοιπόν, για παρακολούθηση της ποιότητας του αέρα σε πραγματικό χρόνο την εποχή που διανύουμε, ώστε να αποφευχθούν σοβαρές και επιβλαβείς επιπτώσεις για την υγεία των ανθρώπων, οδηγεί στη καθημερινή χρήση dashboard, που δίνουν την δυνατότητα παρακολούθησης του AQI αλλά και άλλων δεικτών ποιότητας αέρα.



Εικόνα 3.10: Οι πέντε βασικοί τύποι του καιρού [42]



Εικόνα 3.11: Οι τέσσερις εποχές του χρόνου και η διάρκεια της καθεμιάς [43]

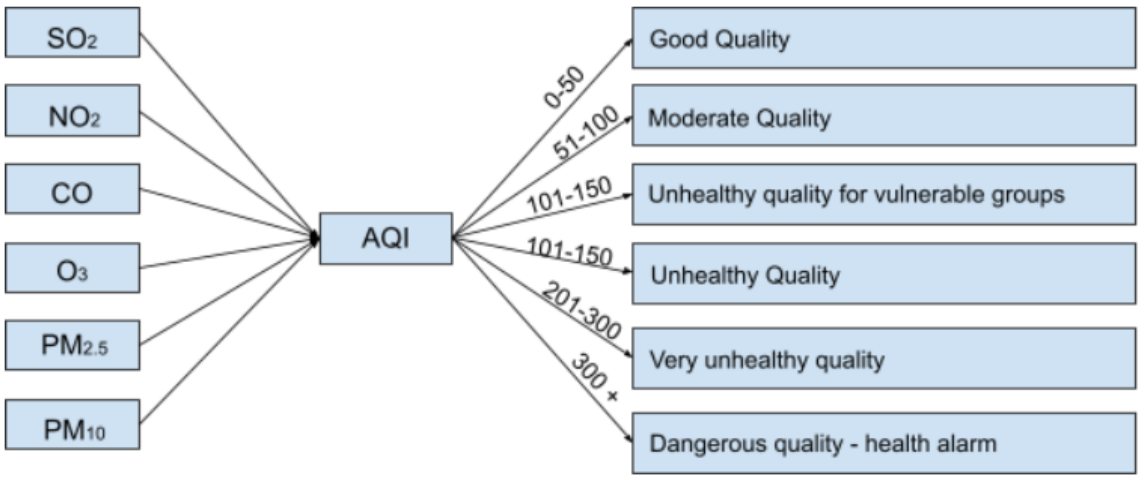
3.1.4 Δείκτες ποιότητας ατμοσφαιρικού αέρα

Πριν κάποια χρόνια, το 1976 δημιουργήθηκε ο δείκτης ρύπων PSI, ο οποίος είναι υπεύθυνος για τη ποιότητα του αέρα. Πιο συγκεκριμένα, αξιολογεί την ποιότητα του αέρα σε μια κλίμακα από το 0-500, σύμφωνα με τα 100 εθνικά πρότυπα ποιότητας του ατμοσφαιρικού αέρα (NAAQS). Η ημερήσια του τιμή υπολογίζεται από την υψηλότερη τιμή ρύπου από όλους που θα καταγράψει. Οι ρύποι που ελέγχει είναι οι σωματιδιακό υλικό (PM_{10}), όζον (O_3), διοξείδιο του θείου (SO_2), μονοξείδιο του άνθρακα (CO) και διοξείδιο του αζώτου (NO_2) [45].

Το βασικό του μειονέκτημα όμως είναι ότι δεν παραθέτει μια έκθεση με όλους τους ρύπους, γεγονός που είναι επίφοβο, διότι αρκετοί επικίνδυνοι ρύποι που δημιουργούν έντονα και σοβαρά προβλήματα υγείας παραλείπονται.

Λόγω του παραπάνω μειονεκτήματος ο δείκτης PSI μετατράπηκε στο δείκτη AQI, μία τιμή χωρίς μονάδα μέτρησης που μπορεί να μετρηθεί ανά μια ώρα έως και χρόνο. Τέθηκε σε εφαρμογή το 1999 από το USEPA. Οι αλλαγές που έγιναν, αφορούν την πρόσθεση ενός καινούργιου ρύπου και την εφαρμογή χρονικού πλαισίου 8 ωρών στο οποίο θα μετράτε η μέση συγκέντρωση O_3. Ο καινούργιος ρύπος είναι ο $PM_{2.5}$, που αφορά τα πολύ μικρά σωματίδια (μεγέθους μικρότερο από 2.5 mm) ο οποίος όμως κάνει δύσκολη έως και αδύνατη την χρήση του AQI δείκτη σε πολλές χώρες λόγω των ακριβών εγκαταστάσεων και μηχανημάτων που απαιτεί για την παρακολούθηση του [46].

Οι δείκτες ποιότητας του αέρα δεν περιορίζονται μόνο σε αυτούς τους δύο που προαναφέρθηκαν. Έχουν δημιουργηθεί διάφοροι, όμως οι δύο που επικράτησαν είναι αυτός των Ηνωμένων Πολιτειών της Αμερικής (US AQI) και της Κίνας (CN AQI). Οι δείκτες αυτοί προσδιορίζονται από την μέτρηση του διοξειδίου του θείου (SO_2), διοξειδίου του αζώτου (NO_2), μονοξειδίου του άνθρακα (CO), αιωρούμενων σωματιδίων $PM_{2.5}$ και PM_{10}. Παρακάτω απεικονίζεται το Σχήμα 3.1, το οποίο βοηθάει στην κατανόηση της σημασίας κάθε πιθανής τιμής του μέσου όρου του AQI [45].



Σχήμα 3.1: Πιθανές τιμές του μέσου όρου του AQI και η σημασία τους [45]

3.1.5 Αισθητήρες μέτρησης ατμοσφαιρικών ρύπων

Για την παρακολούθηση της ποιότητας του αέρα υπάρχει μία αρκετά μεγάλη ποικιλία αισθητήρων που μπορεί να χρησιμοποιηθεί. Σε αρκετές επιστημονικές υλοποιήσεις χρησιμοποιούνται αισθητήρες όπως οι MQ, οι οποίοι είναι χαμηλή σε κόστος και η ακρίβεια τους κυμαίνεται σε αποδεκτά πλαίσια.

Για παράδειγμα, οι αισθητήρες MQ-135 χρησιμοποιούνται κατά βάση για την παρακολούθηση της ποιότητας του αέρα, αλλά και οι MQ-2 αισθητήρες είναι εξίσου καλοί, εφόσον και οι δύο ανιχνεύουν αρκετά αέρια, που μέσα σε αυτά βρίσκονται όλα ή μερικά από τα κύρια 6 αέρια που θεωρούνται πιο βασικά για αυτές τις μετρήσεις. Επιπλέον, οι MQ-135 είναι ιδανικοί για εφαρμογές με μικρό κεφάλαιο λόγω του κόστους τους και τέλος, μπορούν να χρησιμοποιηθούν από απλούς μικροελεγκτές γιατί λειτουργούν στα 5V [45].

3.1.6 Χάρτης

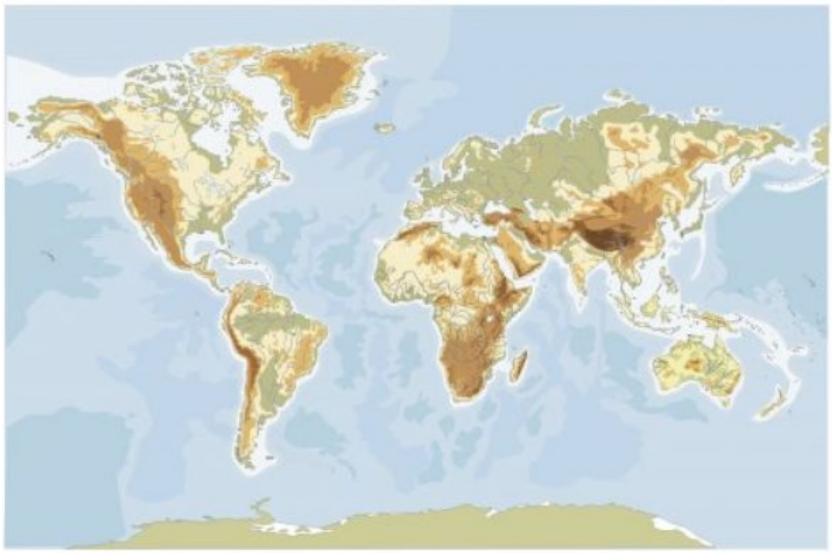
Οι χάρτες είναι ένα πολύ χρήσιμο εργαλείο στην ζωή ενός ανθρώπου, διότι περιέχουν αρκετές πληροφορίες, οι οποίες είναι οι πιο σημαντικές για τα απεικονιζόμενα στοιχεία. Στο χάρτη, ανάλογα πάντα με τον τύπο του, μπορεί να προβάλλονται με διάφορα χρώματα και σύμβολα όπως γραμμές, τα βουνά, οι θάλασσες, το μέγεθος χωρών, οι δρόμοι των πόλεων και άλλα πολλά [47].

Επιπλέον, γίνεται πολύ εύκολη και ευχάριστη η διαδικασία κατανόησης του μεγέθους της περιοχής που βρισκόμαστε ή της χώρας ή και ολόκληρου του πλανήτη μας λόγω της γραφικής απεικόνισης του περιβάλλοντος. Ακόμα, γίνεται λιγότερο χρονοβόρα και περίπλοκη, η διαδικασία εύρεσης της διαδρομής που θα ακολουθήσουμε για να πάμε από το σημείο Α για παράδειγμα, στο σημείο Β [48] [49].

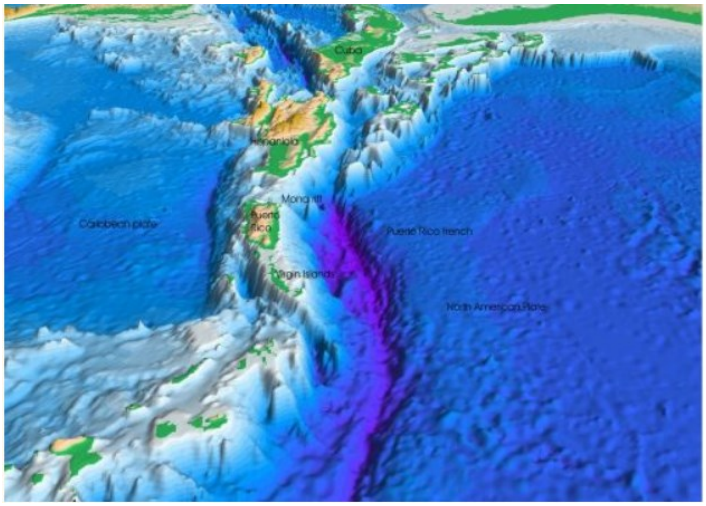
Παρακάτω, στις Εικόνες 3.12 - 3.16, παρουσιάζονται κάποιοι από τους πιο ιδιαίτερους και πιο πολύ χρησιμοποιημένους χάρτες.

Επιπλέον, στους διαδικτυακούς χάρτες, οι οποίοι απεικονίζουν μια πόλη (δρόμους, πολυκατοικίες κτλ.) δίνεται η δυνατότητα προβολής της κίνησης της πόλης και των πυρκαγιών που εξελίσσονται και σε ποια περιοχή (Εικόνα 3.17).

Εν κατακλείδι, όλα τα παραπάνω, θέτουν τον χάρτη ως ένα ουσιώδες και πολύτιμο στοιχείο για την πολύπλευρη πληροφόρηση του διαχειριστή ενός dashboard μιας έξυπνης πόλης.



Εικόνα 3.12: Φυσικός χάρτης [50]

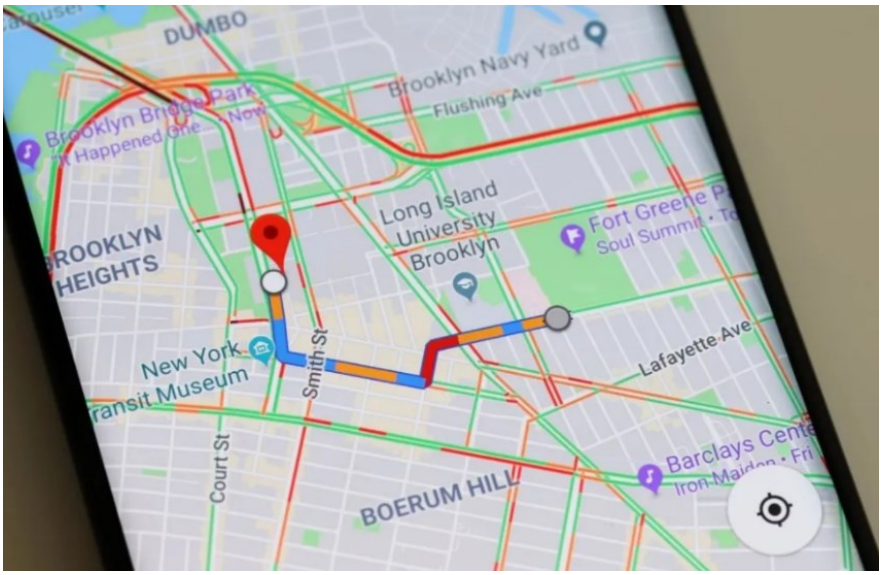


Εικόνα 3.13: Βυθομετρικός χάρτης [50]

Εικόνα 3.14: Γεωλογικός χάρτης [50]

Εικόνα 3.15: Τοπογραφικός χάρτης [50]

Εικόνα 3.16: Πολιτικός χάρτης [50]



Εικόνα 3.17: Διαδικτυακός χάρτης, που εμφανίζει την κίνηση στους δρόμους της πόλης και την διαδρομή που πρέπει να ακολουθήσουμε από το σημείο Α για το σημείο Β [51]

3.1.7 Πυρκαγιά

Η ιστορία έχει δείξει ότι οποιαδήποτε πυρκαγιά είναι ένα καταστροφικό και επικίνδυνο γεγονός, διότι ζώα και άνθρωποι χάνουν τις ζωές τους, λεηλατεί περιουσίες ανθρώπων, δάση και πολλά άλλα, ανάλογα με την κατηγορία της.

Η αρχική διάκριση των πυρκαγιών είναι σε αστικές (πόλεις, σπίτια), σε βιομηχανικές περιοχές, σε δασικές, σε οχήματα, σε πλεούμενα και σε αεροδρόμια. Το μέγεθος της εξαρτάται από τη θερμοκρασία που εκλύεται και την έκταση χώρου όπου καταλαμβάνει. Δηλαδή, οι μεγάλες πυρκαγιές έχουν έκταση περισσότερο από 100 τετραγωνικά μέτρα και θερμοκρασία 1000 με 1200 βαθμούς Κελσίου. Οι μεσαίες, έχουν έκταση λιγότερο από 100 τετραγωνικά μέτρα και η θερμοκρασία τους υπολογίζεται περίπου στους 800 με 1000 βαθμούς Κελσίου. Τέλος, οι μικρές, είναι αυτές που καταλαμβάνουν μικρή σε έκταση γη και η θερμοκρασία τους δεν ξεπερνά τους 800 βαθμούς Κελσίου [52].

Κάθε πολίτης οφείλει να είναι προσεκτικός και υπεύθυνος όσον αφορά το θέμα των πυρκαγιών ιδιαίτερα την καλοκαιρινή περίοδο σε χώρες όπως η Ελλάδα, όπου οι θερμοκρασίες κατά τη διάρκεια αυτών των μηνών είναι αρκετά υψηλές. Επιπλέον, έχοντας γνώση των κατάλληλων μέσων για την αντιμετώπιση

οποιασδήποτε φωτιάς (Εικόνα 3.18), μπορεί να μειωθεί σημαντικά ο κίνδυνος για πυρκαγιές που προκαλούνται από ανθρώπους ή από άλλα αίτια.

Δυστυχώς, το φετινό καλοκαίρι στην Ελλάδα (2021) υπήρξαν αρκετές φωτιές, που έκαψαν πολλά στρέμματα δασικών περιοχών, σπίτια και χωράφια. Οι περισσότερες από αυτές τις πυρκαγιές, εξελίχθηκαν τον Αύγουστο και κατέστρεψαν περισσότερα από 1.200.000 στρέμματα συνολικά [54]. Οι περιοχές που υπέστησαν αυτές τις τεράστιες καταστροφές καταγράφηκαν μέσω της πλατφόρμας, εκείνη την χρονική περίοδο και πιο συγκεκριμένα τις ημερομηνίες 7, 11, 18 και 23 Αυγούστου. Οι εικόνες που καταγράφηκαν απεικονίζονται παρακάτω, αντίστοιχα για κάθε ημερομηνία(Εικόνα 3.19 - Εικόνα 3.22).

Τέλος, πέρα από τη συνεχή προσοχή που πρέπει να δείχνουν οι πολίτες, σε επιφυλακή πρέπει να είναι πάντα και οι αρμόδιες αρχές για την άμεση αντιμετώπιση της φωτιάς αλλά και της ενημέρωσης των κατοίκων. Γι' αυτό τον λόγο, δίνεται η δυνατότητα μέσω της πλατφόρμας να υπάρχει ζωντανή αναμετάδοση των πυρκαγιών που βρίσκονται σε εξέλιξη ή που μόλις έχουν ξεκινήσει.

Water Extinguisher	A
Powder Extinguisher	A B C ⚡
Foam Extinguisher	A B
CO2 Extinguisher	B ⚡
Wet Chemical Extinguisher	A F
Fire Blanket	

Εικόνα 3.18: Ο κατάλληλος πυροσβεστήρας για κάθε είδους φωτιά [53]

Εικόνα 3.19: Οι πυρκαγιές που καταγράφηκαν 07/08/2021

Εικόνα 3.20: Οι πυρκαγιές που καταγράφηκαν 11/08/2021

Εικόνα 3.21: Οι πυρκαγιές που καταγράφηκαν 18/08/2021

Εικόνα 3.22: Οι πυρκαγιές που καταγράφηκαν 23/08/2021

3.1.8 Στάθμευση

Η αύξηση του παγκόσμιου πληθυσμού έχει οδηγήσει στη δημιουργία νέων πόλεων ή στην αύξηση του αριθμού των ήδη υπαρχόντων, εφόσον υπάρχει μεγαλύτερη ανάγκη για εύρεση στέγης και δουλειάς. Αυτό το φαινόμενο είναι γνωστό ως αστικοποίηση.

Η αστικοποίηση έχει πολλά θετικά χαρακτηριστικά, ένα από αυτά είναι ότι φέρνει τους ανθρώπους πιο κοντά και έτσι να κοινωνικοποιούνται. Όμως, έχει και αρκετά μειονεκτήματα. Ένα από αυτά, είναι ότι αυξάνεται η κίνηση των πολιτών με διάφορα οχήματα στους δρόμους της πόλης κι έτσι δεν υπάρχουν τόσες θέσεις στάθμευσης ώστε να τους εξυπηρετήσουν.

Επομένως, δίνοντας στον διαχειριστή μιας τέτοιας πλατφόρμας, την δυνατότητα να επιβλέπει μέσω Wi-Fi καθόλη την διάρκεια της ημέρας, τα μέρη της πόλης, που κυκλοφορούν περισσότερο οι πολίτες και αναζητούν θέσεις στάθμευσης (Εικόνα 3.23) μπορεί να αναγνωρίσει τις ώρες αιχμής, τις περιοχές με τον μεγαλύτερο συνωστισμό και να προβεί στις ανάλογες ενέργειες [55].

Οι ενέργειες αυτές, ξεκινούν από την επικοινωνία τους στους αρμόδιους, αυτοί με την σειρά τους κάνουν σχέδια, τα οποία γίνονται έργα με σκοπό μια πιο λειτουργική πόλη που σέβεται τη φύση και στόχος της είναι η καλυτέρευση της καθημερινής κίνησης των ανθρώπων με τα μέσα τους.

Εικόνα 3.23: Smart parking με την χρήση Wi-Fi [56]

4 Εργαλεία και υλοποίηση

4.1 Εργαλεία και τεχνολογίες για έξυπνες πόλεις

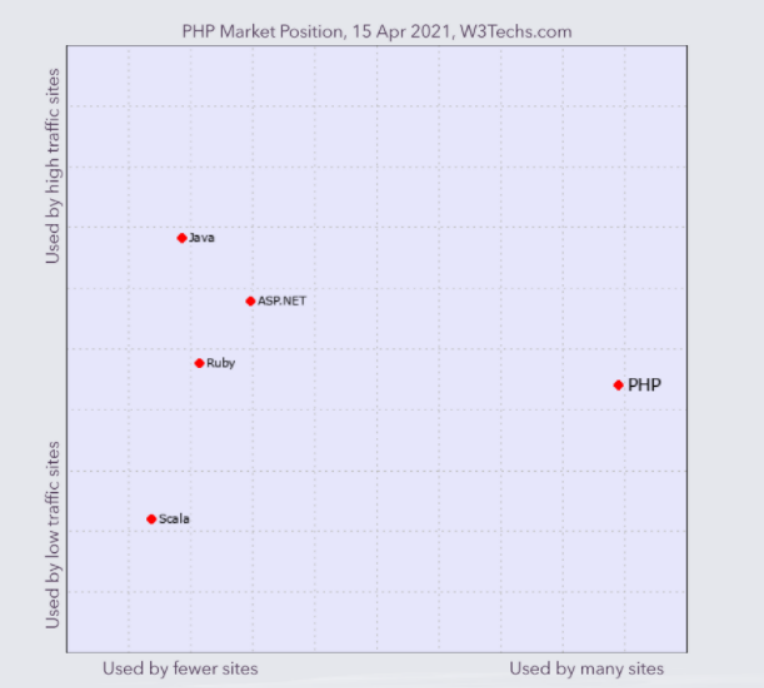
Σε αυτή την υποενότητα θα παρουσιαστούν οι τεχνολογίες που θα μπορούσαν να χρησιμοποιηθούν στην υλοποίηση αντίστοιχων dashboard για έξυπνες πόλεις, όπως και οι δυνατότητες αυτών. Οι τεχνολογίες αυτές χωρίζονται στις backend (PHP, MySQL, Angular) και τις frontend (Python, Java).

4.1.1 PHP

Η PHP (Hypertext Preprocessor), είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται κατά κόρον για την ανάπτυξη διαδικτυακών ιστοσελίδων από την πλευρά του server. Επιπλέον, είναι ανοιχτού λογισμικού και χρησιμοποιείται κυρίως για τη δημιουργία δυναμικών και διαδραστικών ιστοσελίδων.

Μία πολύ σημαντική δυνατότητά της είναι ότι μπορεί να ενσωματωθεί στην HTML, γεγονός που προσθέτει μεγάλη ευκολία στη διαχείριση όπως και στη διαδικασία κατανόησης της. Κάποια άλλα χαρακτηριστικά της, είναι ότι μπορεί να χρησιμοποιηθεί από οποιοδήποτε λειτουργικό σύστημα (Linux, Windows κτλ.), συνδέεται με ευκολία με διάφορες βάσεις δεδομένων και ο χρόνος ανταπόκρισης της είναι αρκετά χαμηλός ειδικά στις τελευταίες εκδόσεις της. Παρόλο τις τόσες πολλές δυνατότητες της, τα τελευταία χρόνια έχει πέσει στις προτιμήσεις των χρηστών του Stack Overflow, διότι από τη 5η θέση που βρισκόταν το 2017, το 2020 βρέθηκε στην 8η (Εικόνα 4.1).

Η σημαντικότητα της και η ευχρηστία της στους χρήστες απεικονίζεται από την προτίμησή της από μεγάλες εταιρείες και οργανισμούς, όπως η Facebook και το Wordpress Foundation αντίστοιχα, που την χρησιμοποιούν για την ανάπτυξη των εφαρμογών τους. Πέρα από αυτές, το 80% περίπου όλων των υπαρχόντων ιστοσελίδων του διαδικτύου τη χρησιμοποιούν [57].



Εικόνα 4.1: Κατάταξη της PHP στην αγορά εργασίας για το 2021 [57]

4.1.2 Python

Η Python είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού, η οποία δημιουργήθηκε από τον Guido Van Rossum, στις αρχές της δεκαετίας του ”80. Με βάση τα στατιστικά της “the TIOBE index”, η Python βρίσκεται τρίτη (3η), στην κατάταξη των πιο δημοφιλών γλωσσών του προγραμματισμού παγκοσμίως.

Ιδιαίτερη σημασία έχει να καταγράψουμε κάποια από τα πιο σημαντικά χαρακτηριστικά της Python, ώστε να καταλάβουμε τους λόγους δημοτικότητάς της. Αρχικά, είναι πολύ εύκολη στην κατανόηση και στη χρήση από κάποιον που τώρα μαθαίνει αυτή τη γλώσσα, διότι η σύνταξη του κώδικά της είναι απλή (π.χ. δε χρειάζονται η τοποθέτηση ερωτηματικού στο τέλος κάθε εντολής). Επιπλέον, η κοινότητα (ομάδες προγραμματιστών) που τη χρησιμοποιεί είναι αρκετά μεγάλη, οπότε υπάρχει αρκετή υποστήριξη μεταξύ των ατόμων για αλγοριθμικά θέματα [58].

Τέλος, οι προγραμματιστές που διαλέγουν την Python, έχουν πολλές επιλογές για ανάπτυξη εφαρμογών στις οποίες μπορούν να τη χρησιμοποιήσουν (διαδικτυακές εφαρμογές, υπολογιστικά μηχανάκια κτλ) [33]. Η ποικιλομορφία χρήσεων της Python, αντικατοπτρίζεται και από το ευρύ φάσμα των εταιριών που τη χρησιμοποιούν. Κάποιες από αυτές τις εταιρείες είναι Netflix, Facebook, IBM, NASA, Pixar, Bank of America και Spotify [60].

4.1.3 Java

Η Java είναι μία αντικειμενοστρεφής γλώσσα προγραμματισμού, η οποία πρωτοεμφανίστηκε το 1995 από την Sun Microsystems [61]. Αποτελεί αγαπημένη επιλογή για πολλούς προγραμματιστές και αυτό απεικονίζεται στην παρακάτω λίστα (Εικόνα 4.2), στην οποία κατακτά την πρώτη θέση από το 2004.

Έχει τόσο υψηλή απήχηση, διότι είναι αντικειμενοστρεφής κι έτσι είναι πιο ευέλικτη η δημιουργία, η επαναχρησιμοποίηση και η χρήση των αντικειμένων. Είναι πιο εύκολη στην κατανόηση και στη χρήση από άλλες αντίστοιχα ισχυρές γλώσσες προγραμματισμού, όπως η C και η C++ [62]. Τέλος, έχει μια τεράστια ποικιλία εφαρμογών που μπορούν να αναπτυχθούν με αυτήν. Για παράδειγμα ανάπτυξη εφαρμογών για κινητές και σταθερές συσκευές, κατανεμημένες εφαρμογές, διαδικτυακές, βασισμένες στο cloud και για παιχνίδια [63].

Εν κατακλείδι, πολλές εταιρείες, οι οποίες υπολογίζονται γύρω στις 9.807 παγκοσμίως, χρησιμοποιούν την Java για όλες τις δυνατότητες της που προαναφέρθηκαν. Ανάμεσα σε αυτές βρίσκονται μεγάλα ονόματα εταιρειών όπως οι Uber, Airbnb, Google, Netflix, Pinterest, Instagram, Spotify και Amazon [64].

Feb 2019	Feb 2018	Programming Language	Ratings	Change
1	1	Java	15.876%	+0.89%
2	2	C	12.424%	+0.57%
3	4	Python	7.574%	+2.41%
4	3	C++	7.444%	+1.72%
5	6	Visual Basic .NET	7.095%	+3.02%
6	8	JavaScript	2.848%	-0.32%
7	5	C#	2.846%	-1.61%
8	7	PHP	2.271%	-1.15%
9	11	SQL	1.900%	-0.46%
10	20	Objective-C	1.447%	+0.32%

Εικόνα 4.2: Η κατάταξη των 10 πρώτων γλωσσών προγραμματισμού για το 2018-2019 [62]

4.1.4 Angular

Η Angular είναι μια ανοιχτού λογισμικού διαδικτυακή πλατφόρμα για την ανάπτυξη διεπαφών. Κυκλοφόρησε επίσημα το 2010 από τους μηχανικούς της Google και βασίζεται κυρίως στη JavaScript [65]. Χρησιμοποιείται κυρίως για τη δημιουργία διεπαφών και για εφαρμογές SPA (Single Page Application). Προτιμάτε πολύ από τους προγραμματιστές, διότι ο κώδικας ανάπτυξης εφαρμογών για διάφορες συσκευές ή για το διαδίκτυο μπορεί να ξαναχρησιμοποιηθεί ανεξαιρέτως συσκευής. Επιπλέον, παρέχει πολλά και εύχρηστα εργαλεία για πιο εύκολη συγγραφή κώδικα. Είναι αρκετά οργανωμένη κι έτσι γίνεται πιο γρήγορη και αποδοτική η ενσωμάτωση καινούργιων μελών σε ομάδες που δημιουργούν και διαχειρίζονται μεγάλα έργα. Ένα ακόμα πολύ σημαντικό χαρακτηριστικό είναι ότι παρέχει ασφάλεια, εφόσον με την Angular ο προγραμματιστής μπορεί να κάνει χρήση κάποιων βημάτων που υπάρχουν για να εξασφαλίσει ασφάλεια και αυθεντικοποίηση στα δεδομένα του [66] [67].

Τέλος, η δύναμη αυτής της πλατφόρμας και οι απεριόριστες δυνατότητες που δίνει, αντικατοπτρίζονται απο τη μεγάλη κοινότητα της, που μέσα σε αυτή βρίσκονται αρκετές ομάδες προγραμματιστών που εργάζονται σε μεγάλες εταιρίες που όλοι τις γνωρίζουμε. Κάποιες από αυτές τις εταιρείες είναι οι YouTube, Netflix, PayPal, The Guardian και LinkedIn [67].

4.1.5 MySQL

Η MySQL είναι ένα ανοιχτού λογισμικού σύστημα διαχείρισης βάσεων δεδομένων. Ανήκει στην Oracle και υπάρχει δωρεάν αλλά και επί πληρωμής έκδοση της. Η επί πληρωμής έκδοση της, προσφέρει μεγαλύτερο εύρος υπηρεσιών υποστήριξης, προς τους χρήστες της [68].

Κάποια από τα χαρακτηριστικά της MySQL είναι ότι χρησιμοποιείται για σχεσιακές βάσεις δεδομένων, είναι αρκετά γρήγορη, μπορεί να λειτουργήσει σε διάφορες συσκευές (σταθερούς υπολογιστές, server κτλ.) και μπορεί να συνδεθούν σε αυτή οι χρήστες της, ανεξαιρέτως πρωτοκόλλου που χρησιμοποιούν [69] [70].

Τέλος, αξίζει να αναφερθούν οι εταιρείες που τη χρησιμοποιούν, ώστε να γίνει κατανοητό πόσο εύχρηστες είναι οι δυνατότητές της στη διαχείριση μεγάλου όγκου δεδομένων. Οι εταιρείες αυτές υπολογίζονται να είναι 5.286 συνολικά. Οι πιο γνωστές απ' όλες αυτές είναι οι Uber, Airbnb, Netflix, Pinterest, Spotify, Amazon, Twitter και Udemy [71].

4.2 Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν στην υλοποίηση

Παρακάτω θα γίνει ανάλυση των τεχνολογιών αλλά και των εργαλείων που χρησιμοποιήθηκαν, ώστε να υλοποιηθεί η συγκεκριμένη εργασία. Και σε αυτό το κεφάλαιο έγινε διαχωρισμός των τεχνολογιών σε backend (NodeJS, ExpressJS, MongoDB,) και frontend (React).

4.2.1 NodeJS

Η NodeJS είναι ένα εργαλείο ανοιχτού λογισμικού ανεπτυγμένο από τον Ryan Dahl και χρησιμοποιείται στο Google Engine Ω8. Χρησιμοποιείται ως ένα server-side περιβάλλον ανάπτυξης και για διαδικτυακές εφαρμογές γραμμένες σε JavaScript [72].

Κάποια από τα χαρακτηριστικά της, είναι ότι διαθέτει πολλές βιβλιοθήκες που καλύπτουν τις περισσότερες ανάγκες των χρηστών και μπορεί να χρησιμοποιηθεί για την ανάπτυξη του frontend αλλά και του backend, με τη χρήση μόνο της JavaScript και για τις δύο περιπτώσεις. Χρησιμοποιείται κυρίως

σε εφαρμογές οι οποίες διαχειρίζονται δεδομένα σε πραγματικό χρόνο, που διαθέτουν μία μόνο σελίδα και τέλος για εφαρμογές που τα APIs τους χειρίζονται τύπου JSON δεδομένα [73].

Βάση αυτών, η NodeJS αποτελεί την ιδανική επιλογή για την ανάπτυξη του backend και της διεπαφής της πλατφόρμας μας, εφόσον έχει μια μόνο σελίδα η οποία είναι και η κεντρική και διαχειρίζεται δεδομένα σε πραγματικό χρόνο που η μορφή τους είναι τύπου JSON [74].

4.2.2 ExpressJS

Το ExpressJS είναι ένα ανοιχτού λογισμικού διαδικτυακό backend framework, που δίνει την ευκαιρία στους προγραμματιστές, σε συνδυασμό με την NodeJS, να αναπτύσσουν εξ ολοκλήρου διαδικτυακές εφαρμογές και APIs.

Επιπλέον, είναι ένα από τα βασικά components και ακρωνύμια της MEAN stack, της οποίας κάθε γράμμα αντιπροσωπεύει και άλλη λειτουργία. Το Μ είναι για την MongoDB βάση δεδομένων, το Ε για το ExpressJS, το Α για τη δημιουργία διεπαφών με Angular και το Ν για τη NodeJS [75].

Είναι ένα πολύτιμο εργαλείο λόγω των δυνατοτήτων που προσφέρει. Κάποια από τα κυριότερα χαρακτηριστικά του, είναι η ευκολία χρήσης και κατανόησής του, αφού απαιτεί από τους χρήστες του, να γνωρίζουν μόνο JavaScript. Επιπλέον, προσφέρει μεγάλη ευκολία όσον αφορά τη διαδικασία του routing για αιτήματα που προέρχονται από τον client, διότι χωρίς αυτό, θα έπρεπε κάθε φορά να ξοδεύουμε χρόνο γράφοντας κώδικα για κάθε ξεχωριστό component του routing.

Οπότε έχοντας ερευνήσει τα χαρακτηριστικά του ExpressJS, όπως την ευκολία, την ευελιξία, την απλότητα και την αποτελεσματικότητα, σε συνδυασμό με την NodeJS, πήραμε την απόφαση να χρησιμοποιήσουμε αυτό τελικά στην εργασία μας ώστε να διαχειριστούμε καλύτερα τα API αιτήματα και την MongoDB [76].

4.2.3 React

Η React είναι μια ανοιχτού λογισμικού βιβλιοθήκη JavaScript, πολυσυζητημένη στην εποχή μας για την ανάπτυξη διεπαφών και πιο συγκεκριμένα, για εφαρμογές με μια μόνο σελίδα. Είναι γνωστή, επειδή προσφέρει ποικίλες δυνατότητες στους προγραμματιστές, αλλά και σε οποιονδήποτε άλλο θελήσει να την χρησιμοποιήσει [77].

Κάποιες από τις δυνατότητές της είναι η ευκολία για να τη μάθει κάποιος, αφού χρειάζονται μόνο βασικές γνώσεις της HTML, CSS και JavaScript. Με την βοήθεια της μπορούν να δημιουργηθούν πέρα από Web εφαρμογές, και Android και iOS εφαρμογές. Επιπλέον, είναι πολύ εύκολο να στηθεί μια καινούργια εφαρμογή με τη χρήση της, αλλά και να διορθωθούν προβλήματα του κώδικα, εφόσον κάθε μέρος της διεπαφής αντιμετωπίζεται ως μια απλή μέθοδος. Τέλος, μια από τις πιο σημαντικές δυνατότητές της, που την κάνει να διαφέρει από πολλές, είναι πως θα ενημερώσει αποτελεσματικά τα σωστά στοιχεία της εφαρμογής όταν αυτά αλλάξουν, χωρίς να χρειάζεται να παρέμβουμε εμείς και να ανανεώσουμε τη σελίδα [78] [79].

Στη συγκεκριμένη πλατφόρμα χρησιμοποιήθηκε η React, διότι η ανάγκη για άμεση ενημέρωση και εμφάνιση των καινούργιων δεδομένων, ήταν βασική προτεραιότητα και χαρακτηριστικό της πλατφόρμας μας. Επιπλέον, χρησιμοποιήθηκε λόγω της ευκολίας να τη μάθουμε πλήρως και σωστά από την αρχή σε μικρό χρονικό διάστημα και λόγω του πολύ καλά δομημένου και τεκμηριωμένου documentation που αποτέλεσε πολύτιμη βοήθεια κατά την υλοποίηση αυτής της εργασίας.

4.2.4 MongoDB

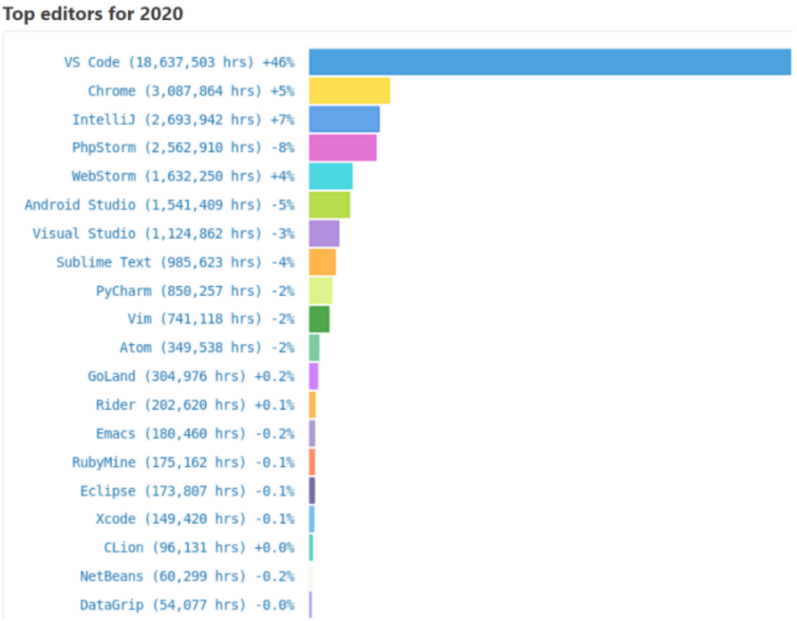
Η MongoDB είναι μία από τις πιο δημοφιλής NoSQL ανοιχτού λογισμικού βάσεις δεδομένων της εποχής μας. Είναι τόσο γνωστή και περιζήτητη, διότι διαφέρει αρκετά από τις γνωστές και συνηθισμένες σχεσιακές βάσης δεδομένων που έχουμε χρησιμοποιήσει μέχρι σήμερα [80]. Ένας βασικός λόγος που διαφέρει η MongoDB, είναι ότι αποθηκεύει και δομεί τα δεδομένα της, σε τύπου JSON αρχεία, τα οποία δεν είναι μια γλώσσα για να εκτελούμε ερωτήματα, αλλά είναι ένας τρόπος για να γίνονται κατανοητά αυτά τα δεδομένα από τα προγράμματα.

Κάποια χαρακτηριστικά των μη σχεσιακών βάσεων που αποτελούν και εξίσου σημαντικές διαφορές με τις σχεσιακές, είναι ότι η MongoDB είναι ένα πολύ ελαφρύ λογισμικό, που δε χρειάζεται ισχυρό εξοπλισμό για να την υποστηρίξει, είναι μια βάση δεδομένων της οποίας οι απαντήσεις στα ερωτήματά μας, επιστρέφονται πολύ πιο γρήγορα και τέλος δεν υπάρχει πλεονασμός αλλά ευελιξία, αφού αποθηκεύονται και χρησιμοποιούνται μόνο τα δεδομένα που θέλουμε, λόγο της ξεχωριστής δομής που μπορούμε να έχουμε σε κάθε εγγραφή αν το επιθυμούμε [81].

Στην παρούσα εργασία χρησιμοποιήσαμε το MongoDB Atlas, μια cloud υπηρεσία για δωρεάν hosting MongoDB βάσεων δεδομένων. Η συγκεκριμένη μη σχεσιακή βάση δεδομένων επιλέχθηκε λόγω όλων των χαρακτηριστικών της που προαναφέρθηκαν παραπάνω. Η υπηρεσία MongoDB Atlas χρησιμοποιήθηκε γιατί εγγυάται διαθεσιμότητα, επεκτασιμότητα και συμμόρφωση με τα πιο απαιτητικά πρότυπα ασφαλείας δεδομένων και απορρήτου, όπως αναφέρεται και στην επίσημη ιστοσελίδα της MongoDB [82].

4.2.5 Visual Studio Code

Το Visual Studio Code γνωστό και ως VS Code, είναι ένα ανοιχτού λογισμικού πρόγραμμα επεξεργασίας κώδικα [83]. Είναι από τα κορυφαία στη λίστα προτίμησης των προγραμματιστών, βάση των στατιστικών που δημοσίευσε το WakaTime, στα οποία παρουσιάζονται οι ώρες χρήσεις του συγκεκριμένου προγράμματος σε σύγκριση με άλλα, για το 2020 (Εικόνα 4.3). Το WakaTime, είναι και αυτό ένα ανοιχτού λογισμικού εργαλείο (open source plugin), το οποίο υπολογίζει το χρόνο που ξοδεύουν οι χρήστες καθώς προβαίνουν σε διάφορες ενέργειες μέσω του προγράμματος.



Εικόνα 4.3: Κατάταξη των εργαλείων για το 2020 [84]

Η προτίμηση των προγραμματιστών καθρεφτίζεται και μέσω του GitHub, στο οποίο αθροίζει 78.4 χιλιάδες αστέρια και 72 χιλιάδες ακόλουθους.

Η προτίμηση αυτή είναι δικαιολογημένη, εφόσον το VS Code δίνει δυνατότητες όπως, η συγγραφή κώδικα σε πάρα πολλές γλώσσες προγραμματισμού (π.χ Java, Python, C++), η συνεργασία πολλών ατόμων εξ' αποστάσεως και σε πραγματικό χρόνο, προτεινόμενες συμπληρώσεις του κώδικα που γράφεται εκείνη τη στιγμή με σκοπό την αποφυγή λαθών, βοήθεια για άμεση και εύκολη αναγνώριση λέξεων κλειδιών κάθε προγραμματιστικής γλώσσας, σύγκριση εκδόσεων του κώδικα μέσω διαγραμμάτων και χρήση μέσα από το ίδιο το VS Code των Jupyter notebooks για απεικόνιση και διάδραση με τα διαγράμματα, τα γραφήματα, τις μεταβλητές και τα δεδομένα [85].

Η απόφασή μου για να χρησιμοποιήσω το VS Code για τη δημιουργία της πλατφόρμας, πάρθηκε όταν αντιλήφθηκα ότι οι δυνατότητές του, όπως η επισήμανση λέξεων κλειδιών με διάφορα χρώματα και οι προτεινόμενες συμπληρώσεις του κώδικα είναι χαρακτηριστικά που προσάπτουν στην διαδικασία του προγραμματισμού, μια αίσθηση ευκολίας, συνεργασίας και βοήθειας στην υλοποίηση. Τέλος, το κομμάτι που προσωπικά με βοήθησε περισσότερο στην ανάπτυξη της πλατφόρμας, ήταν η δυνατότητα σύγκρισης της παλαιότερης έκδοσης των αρχείων του κώδικα με την τωρινή τους μορφή.

4.2.6 Postman

Το Postman είναι ένα πρόγραμμα ανοιχτού λογισμικού, το οποίο χρησιμοποιείται για αυτοματοποιημένο και μη testing, στην ανάπτυξη λογισμικού, τη συγγραφή APIs documentation και σε αρκετά άλλα. Είναι ένα εργαλείο αρκετά δημοφιλές για την ευχρηστία και την αμεσότητα του [86] [87].

Υπάρχει η δωρεάν και η επί πληρωμή έκδοση του. Η δωρεάν έκδοση, παρέχει ένα πακέτο με πληθώρα χαρακτηριστικών που είναι τα πιο χρήσιμα για τις δοκιμές του κώδικα που θέλει να κάνει κάθε προγραμματιστής. Η επί πληρωμή έκδοση του, προσφέρει μια ποικιλία περισσότερων και πιο εξειδικευμένων δυνατοτήτων, όπως για παράδειγμα χρήση από πολλαπλούς χρήστες και Single Sign On (SSO) αυθεντικοποίηση, χαρακτηριστικά που το καθιστούν ως το τέλειο εργαλείο για κάθε είδους εργασία με APIs.

Έχει διευκολύνει όλους τους χρήστες του, απαλλάσσοντας τους από την διαδικασία υλοποίησης κώδικα για το πλαίσιο, το οποίο θα ήταν το ενδιάμεσο στρώμα, ώστε να στέλνονται και να έρχονται αιτήματα στο Postman [88] [89].

Εν κατακλείδι, η επιλογή μου για την χρήση του Postman έγινε λόγω της άνετης και άμεσης διαδικασίας χρήσης, της εύκολης σε κατανόησης διεπαφής και τέλος ένας από τους πιο βασικούς λόγους, είναι ότι υποστηρίζει ποικιλόμορφα σχήματα, δυνατότητα που στην συγκεκριμένη εργασία ήταν πολύτιμη διότι τα δεδομένα διέφεραν μορφολογικά μεταξύ τους και έχρηζαν ξεχωριστής διαχείρισης.

4.2.7 Google Maps

Το Google Maps είναι μια δωρεάν εφαρμογή της Alphabet για τη χαρτογράφηση περιοχών και την πλοήγηση των χρηστών της, σε αυτές. Η εφαρμογή αυτή, προσφέρεται για desktop, κινητά αλλά και σε διαδικτυακή μορφή χωρίς την ανάγκη δηλαδή να γίνει εγκατάσταση της εφαρμογής σε κάποια συσκευή.

Επιπλέον, η προβολή των χαρτών μπορεί να γίνει σε οποιαδήποτε διαστάσεις επιθυμεί ο χρήστης, για παράδειγμα 2D και 3D. Επιπροσθέτως, δίνει την δυνατότητα προβολής της κίνησης στην περιοχή που επιθυμούμε, της πλήρης καθοδήγησης μέχρι τον προορισμό μας και άλλα πολλά [90].

Όλες οι προαναφερόμενες δυνατότητες, είναι προσβάσιμες και μέσω του API της, έτσι ώστε οι προγραμματιστές να μπορούν να επιλέξουν αυτές που επιθυμούν για χρήση στις εφαρμογές που αναπτύσσουν.

Στην παρούσα εργασία, το συγκεκριμένο εύχρηστο και χρήσιμο εργαλείο, χρησιμοποιήθηκε για τις ανάγκες προβολής του χάρτη όπως και της κίνησης στους δρόμους.

Παρακάτω, αναφέρουμε την βασική διαδικασία που χρειάζεται να ακολουθήσει κάποιος όταν και αν θελήσει να χρησιμοποιήσει το Google Maps API σε κάποια δική του εφαρμογή [91].

1. Δημιουργία Google account
2. Σύνδεση στο Google Cloud Platform Console
3. Δημιουργία ενός καινούργιου project
4. Ενεργοποίηση του Google Maps Javascript API
5. Από την παραπάνω ενέργειά μας, δημιουργείται ένα μοναδικό API Key, το οποίο χρησιμοποιείται στην εφαρμογή για αυθεντικοποίηση

4.3 Υλοποίηση backend

4.3.1 Εισαγωγή

Στη συνέχεια, θα γίνει λεπτομερής ανάλυση της υλοποίησης της πλατφόρμας, το οποίο αφορά το backend, ανά ενότητες. Θα παρουσιαστεί η μεθοδολογία και ο τρόπος σκέψης πίσω από κάθε βήμα της υλοποίησης.

4.3.2 Server.js

Το αρχείο server.js, είναι το αρχείο από το οποίο ξεκίνησε η συγγραφή του κώδικα της εργασίας, αφού αποτελεί το “κλειδί” για την ενεργοποίηση των λειτουργιών του backend.

Αρχικά, στις πρώτες πέντε γραμμές είναι όλες οι βιβλιοθήκες οι οποίες χρειαζόμαστε για την καλύτερη διαχείριση των HTTP αιτημάτων, την ορθότερη χρήση των μεταβλητών περιβάλλοντος και σύνδεση με τη βάση δεδομένων αντίστοιχα.

Στη γραμμή 8 φορτώνονται οι μεταβλητές περιβάλλοντος στο πρόγραμμα. Ύστερα, στη γραμμή 11 γίνεται η σύνδεση με τη βάση δεδομένων και στη 13 δηλώνουμε στο app να χρησιμοποιεί τη μέθοδο express.json() έτσι ώστε να μπορεί να διαχειρίζεται η εφαρμογή μας τα εισερχόμενα δεδομένα από τα αιτήματα του frontend. Έπειτα, στη γραμμή 15 δηλώνουμε στο app να κάνει χρήση της μεθόδου cors(), στις γραμμές 18 και 19 ορίζουμε πως όλα τα αιτήματα που έρχονται στο “/” να προωθούνται στη μεταβλητή router, δηλαδή στο αρχείο router.js για να διαχειριστεί τη δρομολόγηση (routing) των αιτημάτων. Τέλος στις γραμμές 27 μέχρι 29 δηλώνουμε την πόρτα στην οποία “ακούει” ο server για αιτήματα και τον ξεκινάμε (Κώδικας 4.1).

```
JS server.js ●
JS server.js > ...
1   const express = require("express");
2   const cors = require("cors");
3   const app = express();
4   const dotenv = require("dotenv");
5   const connectDB = require("./db");
6
7   // Load env vars
8   dotenv.config({ path: "./config/config.env" });
9
10  // Connect to database
11  connectDB();
12  // Body parser
13  app.use(express.json());
14  // Enable CORS
15  app.use(cors());
16
17  // send app to main router
18  const router = require("./router");
19  app.use("/", router);
20
21  // run Server
22  const port = process.env.PORT;
23  app.listen(port, () => {
24      console.log("Server running on port " + port);
25  });
26
27  module.exports = app;
```

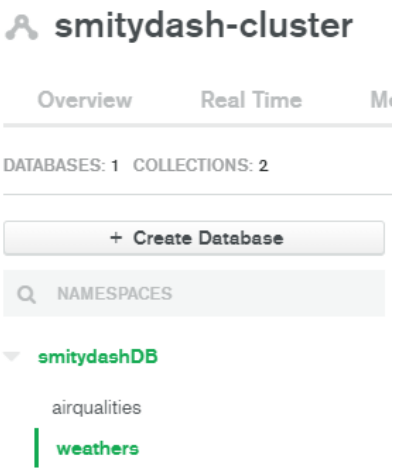
Κώδικας 4.1: Βασικές ρυθμίσεις της πλατφόρμας για την σύνδεση με τον server

4.3.3 DB.js

Η βάση δεδομένων για μια εφαρμογή είναι ζωτικής σημασίας εφόσον εκεί βρίσκεται αποθηκευμένη όλη η πληροφορία της εφαρμογής, και στην εποχή που ζούμε η πληροφορία θεωρείται χρυσός.

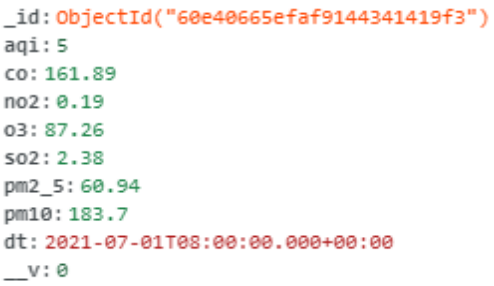
Στη συγκεκριμένη πλατφόρμα, γίνεται χρήση της NoSQL βάση δεδομένων MongoDB στο cloud, λόγω των δυνατοτήτων της στη διαχείριση ποικιλόμορφων δεδομένων.

Για να την ενσωματώσουμε στην εργασία, θα χρειαστούμε έναν λογαριασμό στην υπηρεσία MongoDB Atlas. Έπειτα, θα πρέπει να δημιουργήσουμε ένα cluster, να το ρυθμίσουμε με τις κατάλληλες επιλογές, όπως τη δυνατότητα προβολής και χρήσης των δεδομένων μόνο από συγκεκριμένες IP διευθύνσεις για λόγους ασφαλείας και να φτιάξουμε τα δύο collections που θα μας χρειαστούν για να αποθηκεύουμε τα δεδομένα της ποιότητας του αέρα και του καιρού (Εικόνα 4.4).

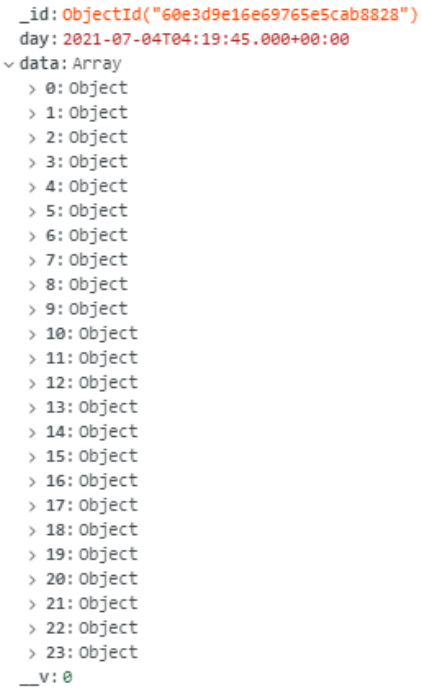


Εικόνα 4.4: Οι δύο συλλογές που περιέχονται στο cluster

Η Εικόνα 4.5 και Εικόνα 4.6 είναι δείγματα για τη μορφή την οποία έχει το περιεχόμενο κάθε εγγραφής, για τις συλλογές airqualities και weathers αντίστοιχα.



Εικόνα 4.5: Οι στήλες κάθε εγγραφής της συλλογής airqualities



Εικόνα 4.6: Οι στήλες κάθε εγγραφής της weathers

Στον επόμενο Κώδικα 4.2 απεικονίζεται η βιβλιοθήκη mongoose, η οποία απαιτείται για την σύνδεση της MongoDB με τη NodeJS.

Στη συνέχεια, η dotenv είναι μια βιβλιοθήκη που χρησιμοποιείται για να διαβάζει τις μεταβλητές περιβάλλοντος. Σκοπός της, είναι να διευκολύνει τον χρήστη στη συγγραφή του κώδικα, αφού αρκεί να δηλώσουμε μία φορά την κάθε μεταβλητή (π.χ. μια μεταβλητή που το περιεχόμενό της είναι ένας σύνδεσμος) στο αρχείο .env και ύστερα χρησιμοποιείται ελεύθερα μόνο με την αναφορά του ονόματος της. Στη γραμμή 4 ξεκινάμε να χρησιμοποιούμε αυτή τη βιβλιοθήκη.

Στις γραμμές 6-15 γίνεται η σύνδεση με τη NoSQL βάση δεδομένων. Στη γραμμή 7 του κώδικα δίνεται το process.env, το οποίο είναι ο τρόπος πρόσβασης στο αρχείο .env και κατ' επέκταση στα δεδομένα, όπως η MONGO_URI που φαίνεται στον κώδικα, η οποία είναι μεταβλητή περιβάλλοντος, περιέχει τον σύνδεσμο για το cluster μας και έχει δηλωθεί στο αρχείο process.env .

Τέλος, λόγω της ασύγχρονης επικοινωνίας πρώτα ολοκληρώνεται η σύνδεση και μετά εμφανίζεται το κατάλληλο μήνυμα για την επιτυχή επιβεβαίωση της διαδικασίας.

```
db.js
db.js > ...
1   const mongoose = require("mongoose");
2   const dotenv = require("dotenv");
3
4   dotenv.config();
5
6   const connectDB = async () => {
7       const conn = await mongoose.connect(process.env.MONGO_URI, {
8           useNewUrlParser: true,
9           useCreateIndex: true,
10          useFindAndModify: false,
11          useUnifiedTopology: true,
12      });
13
14      console.log("MongoDB Connected");
15  };
16
17  module.exports = connectDB;
```

Κώδικας 4.2: Σύνδεση με την NoSQL βάση δεδομένων

4.3.4 Router.js

Το router.js είναι από τα πιο σημαντικά αρχεία της πλατφόρμας. Μέσα σε αυτό το αρχείο, δηλώνουμε κάθε HTTP request μέθοδο (π.χ. GET, POST, PATCH) που θα χρειαστούμε για τη διαχείριση των δεδομένων της πλατφόρμας και για την περιήγηση μας στην εφαρμογή [92].

Στο δικό μας αρχείο, εμφανίζονται έξι τέτοιες περιπτώσεις. Στις γραμμές 4 εώς και 6 (Κώδικας 4.3) εμφανίζονται τύπου GET αιτήματα για τον καιρό. Αναλυτικότερα, στη γραμμή 4 γίνεται ένα GET αίτημα για τον τωρινό καιρό στη πόλη μας, το οποίο διαχειρίζεται το διαχειρίζεται η getCurrentWeatherData() μέθοδος, στη γραμμή 5 έχουμε τον ίδιο τύπο αιτήματος, αλλά για τα ιστορικά δεδομένα που είναι αποθηκευμένα στη NoSQL βάση δεδομένων μας και διαχειρίζονται από τη μέθοδο getWeatherDataFromDb(). Τέλος, στη γραμμή 6 ζητάμε τα καινούργια για τη βάση μας, ιστορικά δεδομένα τα οποία ελέγχονται από τη μέθοδο getRemoteWeatherDataSaveLocal().

Στη γραμμή 8 έως και 10 εμφανίζονται τα αντίστοιχα αιτήματα, αλλά αυτή την φορά για την ποιότητα του αέρα τα οποία ελέγχονται από τις μεθόδους getCurrentAirQualityData(),getAirQualityDataFromDb() και getRemoteAirQualityDataSaveLocal() το καθένα αντίστοιχα.

```
router.js
router.js > ...
1   const router = require("express").Router();
2   const apiHandler = require("./api.controller");
3
4   router.get("/weather/current", apiHandler.getCurrentWeatherData);
5   router.get("/weather/history", apiHandler.getWeatherDataFromDb);
6   router.get("/weather/newData", apiHandler.getRemoteWeatherDataSaveLocal);
7
8   router.get("/airquality/current", apiHandler.getCurrentAirQualityData);
9   router.get("/airquality/history", apiHandler.getAirQualityDataFromDb);
10  router.get("/airquality/newData", apiHandler.getRemoteAirQualityDataSaveLocal);
11
12  module.exports = router;
```

Κώδικας 4.3: Όλα τα πιθανά API calls της πλατφόρμας και οι μέθοδοι διαχείρισης τους

4.3.5 API.controller.js

Στην προηγούμενη υπό ενότητα έγινε χρήση έξι μεθόδων, οι οποίες διαχειρίζονται τις απαντήσεις των τύπου GET αιτημάτων μας.

Όλες αυτές οι μέθοδοι είναι ασύγχρονες και ο λόγος είναι ότι θέλουμε ο υπόλοιπος κώδικας να περιμένει να ολοκληρωθεί η μέθοδος που καλείται κάθε φορά, να μας φέρει τα καινούργια δεδομένα και ύστερα να τα εμφανίσουμε ή να τα αποθηκεύσουμε [93].

```
JS api.controller.js ●
JS api.controller.js > [@] <unknown>
1   const axios = require("axios");
2   const timestamp = require("unix-timestamp");
3   const Weather = require("./models/weather.model");
4   const AirQuality = require("./models/airQuality.model");
5   const Parking = require("./models/parking.model");
6
7   const secondsInADay = 86400;
8
9   module.exports = {
10  >   async getCurrentWeatherData(req, res) {···
24      },
25  >   async getWeatherDataFromDb(req, res) {···
52      },
53  >   async getRemoteWeatherDataSaveLocal(req, res) {···
114     },
115 >   async getCurrentAirQualityData(req, res) {···
137     },
138 >   async getAirQualityDataFromDb(req, res) {···
145     },
146 >   async getRemoteAirQualityDataSaveLocal(req, res) {···
185     },
```

Κώδικας 4.4: Όλες οι μέθοδοι διαχείρισης των API calls

```
async getCurrentWeatherData(req, res) {
  await axios
    .get(
      `https://api.openweathermap.org/data/2.5/weather?lat=40.63&lon=20.95&appid=${process.env.WEATHER_API_KEY}&units=metric`
    )
    .then(function (response) {
      res.status(200).json({
        success: true,
        data: response.data,
      });
    })
    .catch(function (error) {
      return res.status(500).json({ success: false, data: error });
    });
},
```

Κώδικας 4.5: Υλοποίηση της μεθόδου getCurrentWeatherData

Στον Κώδικα 4.5 φαίνεται η μέθοδος getCurrentWeatherData(), η οποία δέχεται το GET αίτημα για τα τωρινά καιρικά δεδομένα της πόλης μας. Εάν ο server μας επιστρέψει status “200”, τότε τα καινούργια δεδομένα αποθηκεύονται, αλλιώς επιστρέφονται τα κατάλληλα μηνύματα. λάθους.

Στον Κώδικα 4.6 απεικονίζεται η υλοποίηση της μεθόδου getWeatherDataFromDb(). Αρχικά, στη μεταβλητή αποθηκεύονται ταξινομημένες οι μέρες και για κάθε μέρα υπολογίζεται η μέση ωριαία θερμοκρασία και υγρασία. Τα επεξεργασμένα αυτά δεδομένα στο τέλος αποθηκεύονται στη βάση.

```
async getCurrentWeatherData(req, res) {···
},
async getWeatherDataFromDb(req, res) {
  let avgDailyWeatherData = [];
  const weather_data = await Weather.find({}).sort("day");

  weather_data.forEach((day) => {
    let avgTemp = 0;
    let avgHumidity = 0;

    day.data.forEach((hourlyData) => {
      avgTemp = hourlyData.temp + avgTemp;
      avgHumidity = hourlyData.humidity + avgHumidity;
    });
    avgTemp = Math.round(avgTemp / 24);
    avgHumidity = Math.round(avgHumidity / 24);

    avgDailyWeatherData.push({
      _id: day._id,
      day: day.day,
      avgTemp: avgTemp,
      avgHumidity: avgHumidity,
    });
  });

  res.status(200).json({
    success: true,
    data: avgDailyWeatherData,
  });
},
```

Κώδικας 4.6: Υλοποίηση της μεθόδου getWeatherDataFromDb

```
53      async getRemoteWeatherDataSaveLocal(req, res) {
54          const currentDayUnixTime = Math.round(timestamp.now());
55          const daysToGetDataFor = [
56              currentDayUnixTime - 5 * secondsInADay,
57              currentDayUnixTime - 4 * secondsInADay,
58              currentDayUnixTime - 3 * secondsInADay,
59              currentDayUnixTime - 2 * secondsInADay,
60              currentDayUnixTime - 1 * secondsInADay,
61          ];
62
63          await daysToGetDataFor.forEach((day) => {
64              axios
65                  .get(
66                      `https://api.openweathermap.org/data/2.5/onecall/timemachine?lat=40.63&lon=20.95&dt=${day}&appid=${process.env.WEATHER_API_KEY}&units=metric`
67                  )
68                  .then(function (response) {
69                      let aDaysData = [];
70                      let weatherDataForADay = [];
71                      let dateOfDay = new Date(day * 1000);
72
73                      for (const hourlyData of response.data.hourly) {
74                          let {
75                              dt,
76                              temp,
77                              feels_like,
78                              humidity,
79                              clouds,
80                              wind_speed,
81                              weather,
82                          } = hourlyData;
```

Κώδικας 4.7: Υλοποίηση της μεθόδου getRemoteWeatherDataSaveLocal

Η μέθοδος getRemoteWeatherDataSaveLocal() δέχεται τα καιρικά δεδομένα για τις τελευταίες πέντε ημέρες από το GET ερώτημα (Κώδικας 4.7) και για κάθε μια μέρα από αυτές αποθηκεύονται τα ωριαία αλλά και τα ημερήσια δεδομένα στη βάση. Η μέθοδος αυτή, έγινε με σκοπό να καλείται κάθε πέντε μέρες χειροκίνητα από εμάς, μέσω του Postman και με την χρήση του API call (localhost:5000/weather/newData), ώστε να μπορέσουμε να έχουμε μεγαλύτερο εύρος ιστορικότητας των καιρικών δεδομένων. Σε περίπτωση αποτυχίας ή επιτυχίας της παραπάνω διαδικασίας, εμφανίζεται το κατάλληλο μήνυμα (Κώδικας 4.8).

```
                        aDaysData.push({
                            dt,
                            temp,
                            feels_like,
                            humidity,
                            clouds,
                            wind_speed,
                            weather,
                        });
                    }

                    weatherDataForADay.push({
                        day: dateOfDay,
                        data: aDaysData,
                    });

                    Weather.create(weatherDataForADay, function (err, weatherData) {
                        if (err) res.status(500).json({ success: false, data: error });
                    });
                })
                .catch(function (error) {
                    return res.status(500).json({ success: false, data: error });
                });
        });
        res
            .status(200)
            .json({ success: true, data: "weather data save succesfully" });
    },
```

Κώδικας 4.8: Συνέχεια υλοποίησης της μεθόδου getRemoteWeatherDataSaveLocal

```
async getCurrentAirQualityData(req, res) {
  await axios
    .get(
      `http://api.openweathermap.org/data/2.5/air_pollution?lat=40.63&lon=20.95&appid=${process.env.WEATHER_API_KEY}`
    )
    .then(function (response) {
      let filteredData = {
        aqi: response.data.list[0].main.aqi,
        co: response.data.list[0].components.co,
        no2: response.data.list[0].components.no2,
        o3: response.data.list[0].components.o3,
        so2: response.data.list[0].components.so2,
        pm2_5: response.data.list[0].components.pm2_5,
        pm10: response.data.list[0].components.pm10,
        dt: new Date(response.data.list[0].dt * 1000),
      };

      res.status(200).json({ success: true, data: filteredData });
    })
    .catch(function (error) {
      return res.status(500).json({ success: false, data: error });
    });
},
```

Κώδικας 4.9: Υλοποίηση της μεθόδου getCurrentAirQualityData

Επόμενη μέθοδος είναι η getCurrentAirQualityData(), η οποία δέχεται τα τωρινά δεδομένα για την ποιότητα του αέρα και τα εμφανίζει στην πλατφόρμα. Ανάλογα με το status που θα επιστρέψει ο server, θα εμφανιστεί και το κατάλληλο μήνυμα (Κώδικας 4.9).

Παρακάτω, στην μέθοδο getAirQualityDataFromDb(), δεχόμαστε τα δεδομένα για την ποιότητα του αέρα από το GET αίτημα και αν η απάντηση που πάρουμε από το server είναι status "200" τότε τα καινούργια δεδομένα αποθηκεύονται αλλιώς εμφανίζουμε κατάλληλο μήνυμα λάθους (Κώδικας 4.10).

```
135     async getAirQualityDataFromDb(req, res) {
136         const airQuality_data = await AirQuality.find({}).sort("dt");
137
138         const finalAggregatedData = dataAggregation(airQuality_data);
139
140         res.status(200).json({
141             success: true,
142             data: finalAggregatedData,
143         });
144     },
```

Κώδικας 4.10: Υλοποίηση της μεθόδου getAirQualityDataFromDb

```
async getRemoteAirQualityDataSaveLocal(req, res) {
  const currentDayUnixTime = Math.round(timestamp.now());
  const startDateToGetData = currentDayUnixTime - 5 * secondsInADay;
  const endDateToGetData = currentDayUnixTime - 1 * secondsInADay;

  await axios
    .get(
      `http://api.openweathermap.org/data/2.5/air_pollution/history?lat=40.63&lon=
      20.95&start=${startDateToGetData}&end=${endDateToGetData}&appid=${process.env.WEATHER_API_KEY}`
    )
    .then(function (response) {
      let airQualityDataPerHour = [];

      for (const hourlyData of response.data.list) {
        let { co, no2, o3, so2, pm2_5, pm10 } = hourlyData.components;

        airQualityDataPerHour.push({
          aqi: hourlyData.main.aqi,
          co,
          no2,
          o3,
          so2,
          pm2_5,
          pm10,
          dt: new Date(hourlyData.dt * 1000),
        });
      }
```

Κώδικας 4.11: Υλοποίηση της μεθόδου getRemoteAirQualityDataSaveLocal

Η μέθοδος getRemoteAirQualityDataSaveLocal() παίρνει τα δεδομένα της ποιότητας του αέρα για τις τελευταίες πέντε ημέρες από το GET ερώτημα (Κώδικας 4.11) και για κάθε μια μέρα από αυτές, αποθηκεύονται τα ωριαία δεδομένα στη βάση. Η μέθοδος αυτή, όπως και η getRemoteWeatherDataSaveLocal() που προαναφέρθηκε, έγινε με σκοπό να καλείται κάθε πέντε μέρες χειροκίνητα από εμάς, μέσω του Postman και με την χρήση του API call (llocalhost:5000/airquality/newData), ώστε να μπορέσουμε να έχουμε μεγαλύτερο εύρος ιστορικότητας των δεδομένων της ποιότητας του αέρα. Σε περίπτωση αποτυχίας ή επιτυχίας της παραπάνω διαδικασίας, εμφανίζεται το κατάλληλο μήνυμα (Κώδικας 4.12).

```
  // save air quality data into database
  AirQuality.create(
    airQualityDataPerHour,
    function (err, airQualityData) {
      if (err) res.status(500).json({ success: false, data: error });
      else res.status(200).json({ success: true, data: airQualityData });
    }
  );
})
.catch(function (error) {
  return res.status(500).json({ success: false, data: error });
});
```

Κώδικας 4.12: Συνέχεια υλοποίησης της μεθόδου getRemoteAirQualityDataSaveLocal

```
186   const dataAggregation = (arrayToAggregate) => {
187       let finalAggregatedArray = [];
188
189       for (let i = 0; i < arrayToAggregate.length; i++) {
190           if (i !== arrayToAggregate.length - 1) {
191               let formatedDateA = moment(arrayToAggregate[i].dt).format("DD MMM YYYY");
192               let formatedDateB = moment(arrayToAggregate[i + 1].dt).format(
193                   "DD MMM YYYY"
194               );
195
196               if (formatedDateA !== formatedDateB) {
197                   finalAggregatedArray.push(arrayToAggregate[i]);
198               }
199           }
200       }
201       return finalAggregatedArray;
202   };
```

Κώδικας 4.13: Μέθοδος μορφοποίησης των ημερομηνιών

Τέλος, ο Κώδικας 4.13 απεικονίζει την μέθοδο dataAggregation(), η οποία δέχεται ως παράμετρο έναν πίνακα με ημερομηνίες και επιστρέφει έναν τελικό πίνακα με ημερομηνίες στην επιθυμητή μορφή. Πιο συγκεκριμένα, μορφοποιεί κατάλληλα τις ημερομηνίες που βρίσκονται στον πίνακα arrayToAggregate(). Ο πίνακας αυτός, γεμίζει με ημερομηνίες που δέχεται από το GET αίτημα μας και για κάθε γραμμή αυτού επεξεργαζόμαστε την μορφή της ημερομηνίας με την βοήθεια της βιβλιοθήκης moment κι αν τα δεδομένα όντως έχουν αλλάξει, τα αποθηκεύουμε στον πίνακα που επιστρέφει.

4.4 Υλοποίηση frontend

4.4.1 Εισαγωγή

Τα επόμενα βήματα, για την υλοποίηση της πλατφόρμας αφορούν το frontend. Οπότε, θα γίνει παρουσίαση, αρχικά του αποτελέσματος που εμφανίζεται στην οθόνη για κάθε κομμάτι της πλατφόρμας και ύστερα η υλοποίηση σε κώδικα του καθενός.

4.4.2 App.js

Πρώτο βήμα, είναι το αρχείο app.js, το οποίο αποτελεί κορμό για την εργασία, όπως και για κάθε εργασία που χρησιμοποιεί την React. Παρακάτω, στις γραμμές 14 έως 16 γίνονται οι δηλώσεις μεταβλητών, η μέθοδος διαχείρισης της κάθε μιας από αυτές και η αρχική τους κατάσταση. Στη συνέχεια, θέτουμε μια ελάχιστη καθυστέρηση στη φόρτωση της πλατφόρμας, ώστε να γίνει δυνατή η

προβολή του εικονιδίου φόρτωσης και εφόσον τελειώσει αυτή η καθυστέρηση και η μεταβλητή loading αλλάξει state εμφανίζεται ο χάρτης (Κώδικας 4.14), οι πίνακες με τον καιρό, την ποιότητα του αέρα (Κώδικας 4.15), οι κάμερες και ο πίνακας με τις θέσεις στάθμευσης (Κώδικας 4.16).

```
const App = () => {
    const [showTraffic, setShowTraffic] = useState(false);
    const [showFire, setShowFire] = useState(false);
    const [loading, setLoading] = useState(true);

    setTimeout(() => {
        setLoading(false);
    }, 1500);

    return (
        <>
            {loading ? (
                <div
                    style={{
                        display: "flex",
                        flexDirection: "column",
                        alignItems: "center",
                        paddingTop: "10%",
                    }}
                >
                    <Logo height={""} logoWidth={"50%"} />
                    <Loader />
                </div>
            ) : (
                <Row style={{ height: "100vh" }}>
                    <Map showFire={showFire} showTraffic={showTraffic} />
```

Κώδικας 4.14: Προβολή του loader & logo και εμφάνιση του χάρτη

```
                <Col
                    span={5}
                    style={{
                        backgroundImage:
                            "linear-gradient(to right, rgb(20, 20, 20), transparent)",
                        height: "100%",
                    }}
                >
                    <Logo height={"35px"} logoWidth={"18%"} />
                    <div
                        style={{
                            display: "flex",
                            flexDirection: "column",
                            justifyContent: "space-evenly",
                            paddingLeft: "1em",
                        }}
                    >
                        <Weather />
                        <AirQuality />
                    </div>
                </Col>
```

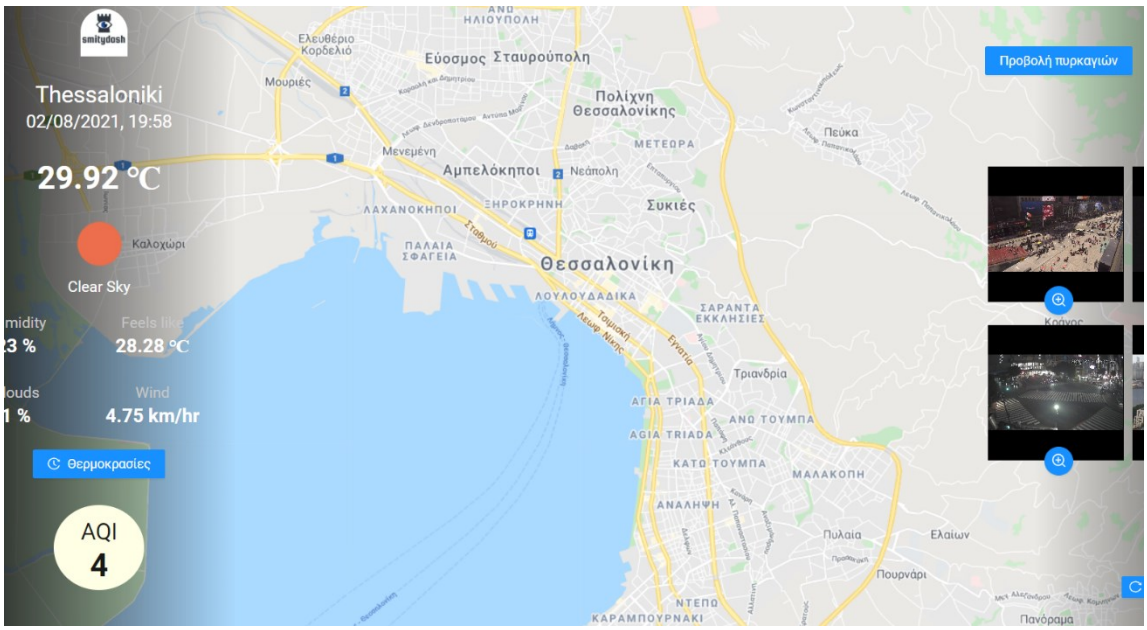
Κώδικας 4.15: Προβολή των πινάκων καιρού και ποιότητας αέρα

```
61                      <Col span={14}>{/* <Map /> */}</Col>
62                      <Col
63                          span={5}
64                          style={{
65                              backgroundImage:
66                                  "linear-gradient(to left, rgb(20, 20, 20), transparent)",
67                              display: "flex",
68                              flexDirection: "column",
69                              justifyContent: "space-evenly",
70                              textAlign: "right",
71                              paddingRight: "1em",
72                              height: "100%",
73                          }}
74                      >
75                          <Cameras
76                              showTraffic={showTraffic}
77                              setShowTraffic={setShowTraffic}
78                              showFire={showFire}
79                              setShowFire={setShowFire}
80                          />
81                          <Parking />
82                      </Col>
83                  </Row>
84              )}
85          </>
86      );
87  };
88
89  export default App;
```

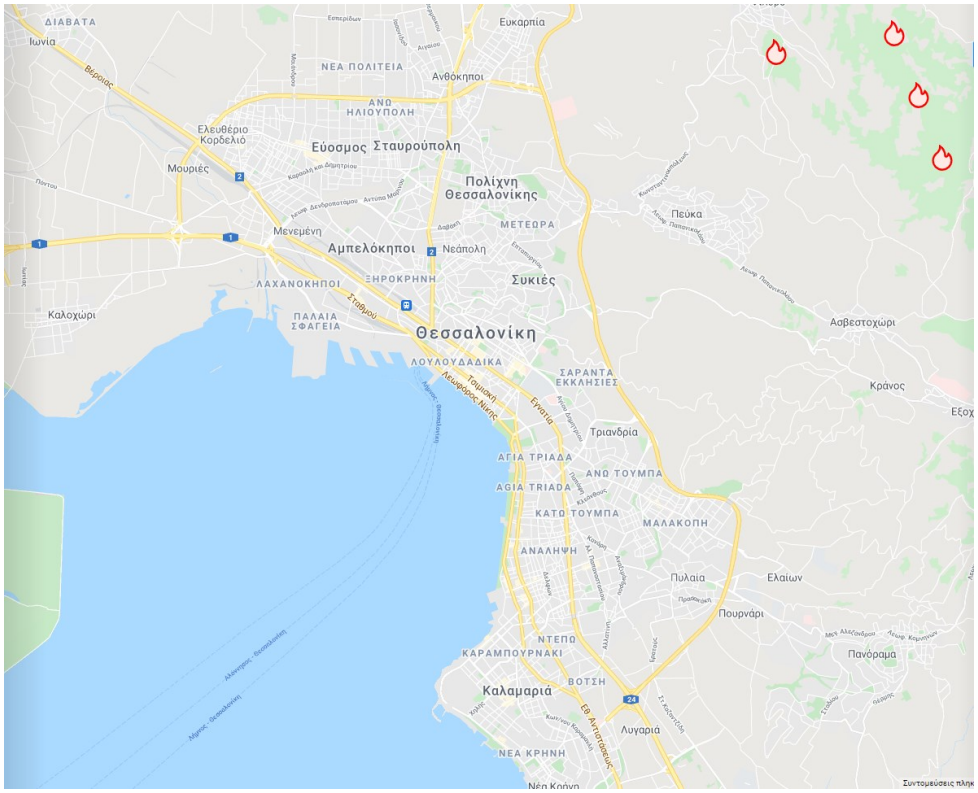
Κώδικας 4.16: Προβολή των πινάκων που περιέχουν τις κάμερες και τα δεδομένα για την στάθμευση

4.4.3 Χάρτης

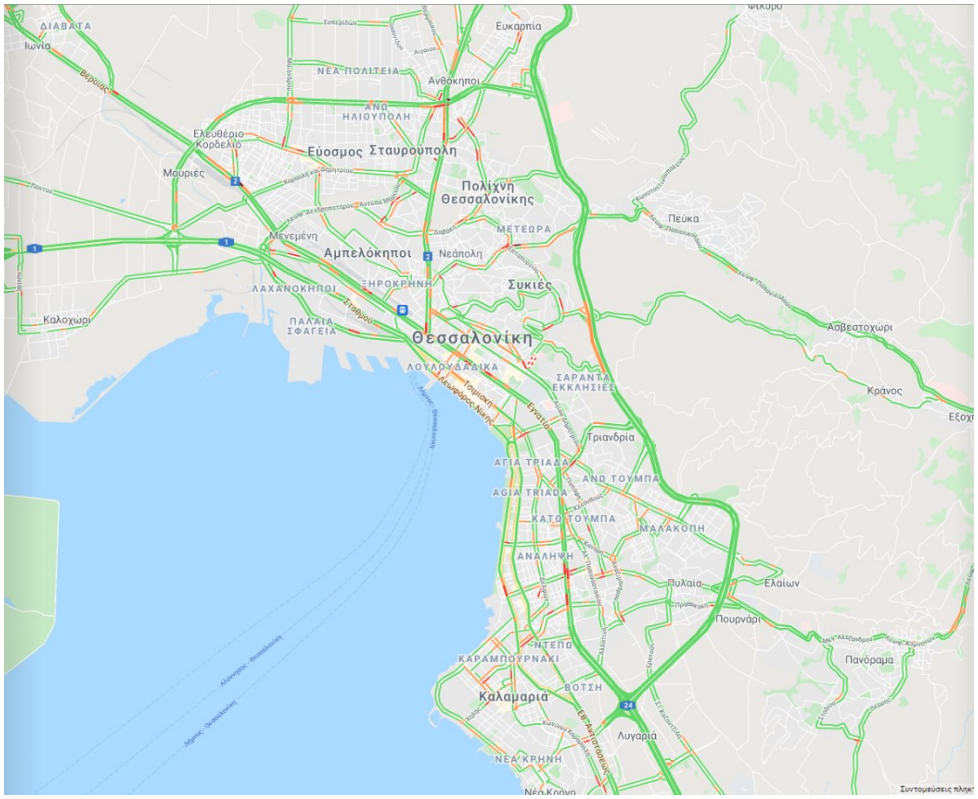
Ο χάρτης της πόλης έχει τοποθετηθεί στο κέντρο της πλατφόρμας και παρέχει αρκετές δυνατότητες στο χρήστη (Εικόνα 4.7). Οι δύο πιο βασικές είναι: η προβολή πυρκαγιών (Εικόνα 4.8) και η προβολή κίνησης των πολιτών με οχήματα στους δρόμους της πόλης (Εικόνα 4.9). Στη συνέχεια της ενότητας γίνεται η ανάλυση της υλοποίησης τους.



Εικόνα 4.7: Προβολή του χάρτη



Εικόνα 4.8: Προβολή του χάρτη με εμφάνιση των πυρκαγιών της πόλης



Εικόνα 4.9: Προβολή του χάρτη με εμφάνιση της κίνησης της πόλης

Στον Κώδικα 4.17 δίνονται οι συντεταγμένες των πυρκαγιών σε εξέλιξη, τις οποίες παίρνουμε φιλτράροντας κατάλληλα τα δεδομένα που μας δίνονται μέσω του API. Ύστερα τα εμφανίζουμε στο χάρτη.

Επιπλέον, έχουν δοθεί από εμάς, πέντε ψευδής συντεταγμένες για τις ανάγκες παρουσίασης του συγκεκριμένου χαρακτηριστικού της πλατφόρμας για τη Θεσσαλονίκη, διότι την περίοδο εκπόνησης της εργασίας δεν υπήρξαν φωτιές στην πόλη μας.

Βέβαια, η δυνατότητα προβολής αληθινών σε εξέλιξη πυρκαγιών μπορεί να γίνει εάν μεγεθύνουμε το οπτικό μας πεδίο στο χάρτη (π.χ. να βλέπουμε όλη την Ευρώπη), πατώντας το κουμπί με τον θετικό μεγεθυντικό φακό (Εικόνα 4.11), έτσι ώστε να υπάρχουν περισσότερες πιθανότητες να εντοπίσουμε μια πυρκαγιά και να εμφανιστεί στο χάρτη στο κατάλληλο σημείο με το κατάλληλο εικονίδιο (Εικόνα 4.10).

Παρακάτω, ο κώδικας 4.18 είναι υπεύθυνος για την εμφάνιση του χάρτη. Η προβολή της κίνησης γραμμές 23 έως 30 δημιουργείται ως ένα επιπρόσθετο στρώμα πάνω στον χάρτη, που εφαρμόζεται όταν πατηθεί το αντίστοιχο κουμπί (Εικόνα 4.9), αλλιώς εμφανίζεται απλός ο χάρτης (Εικόνα 4.7). Στις γραμμές 31 και 40 δηλώνεται ότι σε περίπτωση που πατηθεί το κουμπί για την προβολή πυρκαγιών, σε οποιαδήποτε κατάσταση κι αν είναι ο χάρτης, να εμφανιστούν τα εικονίδια φωτιάς που δηλώνουν τα σημεία που υπάρχει πυρκαγιά σε εξέλιξη (Εικόνα 4.8).

```
4  export default function Map({
5      center,
6      fireData,
7      mapZoom,
8      showTraffic,
9      showFire,
10 }) {
11     const markers = fireData.map((ev, index) => {
12         return (
13             <LocationMarker
14                 key={index}
15                 lat={ev.geometry[0].coordinates[1]}
16                 lng={ev.geometry[0].coordinates[0]}
17             />
18         );
19     });
```

Κώδικας 4.17: Δήλωση μεταβλητών για τις συντεταγμένες των πυρκαγιών και εκχώρηση τιμών σε αυτές

Εικόνα 4.10: Πυρκαγιές σε εξέλιξη στην Ευρώπη

Εικόνα 4.11: Οι φακοί για μεγέθυνση και σμίκρυνση του χάρτη

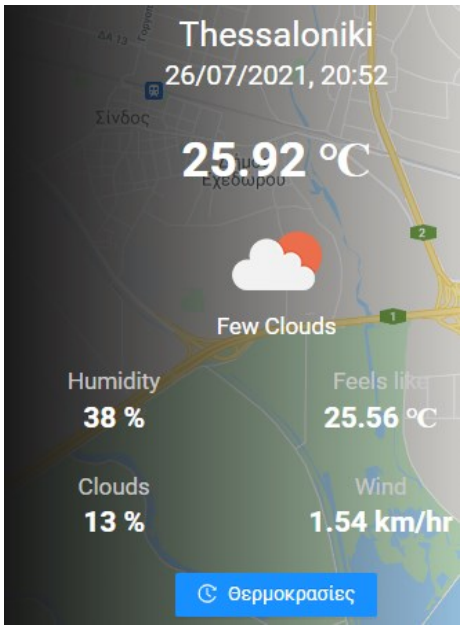
```
21    return (
22        <>
23            {showTraffic ? (
24                <GoogleMapReact
25                    bootstrapURLKeys={{ key: "AIzaSyDvViqPhBvFHMpEvS5eBm78Zh1H3wszz30" }}
26                    layerTypes={["TrafficLayer"]}
27                    defaultCenter={center}
28                    zoom={mapZoom}
29                    style={mapStyle}
30                >
31                    {showFire ? markers : ""}
32                </GoogleMapReact>
33            ) : (
34                <GoogleMapReact
35                    bootstrapURLKeys={{ key: "AIzaSyDvViqPhBvFHMpEvS5eBm78Zh1H3wszz30" }}
36                    defaultCenter={center}
37                    zoom={mapZoom}
38                    style={mapStyle}
39                >
40                    {showFire ? markers : ""}
41                </GoogleMapReact>
42            )}
43        </>
44    );
45    }
```

Κώδικας 4.18: Προβολή του χάρτη με ή χωρίς την εμφάνιση της κίνησης ή των πυρκαγιών

4.4.4 Καιρός

Επόμενο βήμα στη λίστα των υλοποιήσεων για τη δημιουργία της πλατφόρμας είναι η εμφάνιση των καιρικών συνθηκών της Θεσσαλονίκης (Εικόνα 4.12). Επιλέχθηκαν οι πέντε δείκτες (θερμοκρασία, υγρασία, συννεφιά, αίσθηση θερμοκρασίας και αέρας) ως οι πιο χρήσιμοι για την πληροφόρηση του διαχειριστή.

Στην React υπάρχει η δυνατότητα να ορίζουμε ποια μέθοδος θα διαχειρίζεται ποια μεταβλητή [94]. Για γίνει αυτό, αρχικά πρέπει να δηλωθεί το όνομα της μεταβλητής και το όνομα της μεθόδου μέσα σε αγκύλες. Τέλος, δίνεται και η αρχική κατάσταση της μεταβλητής (π.χ. false) (Κώδικας 4.19).



Εικόνα 4.12: Ο πίνακας των καιρικών δεδομένων

```
13  export default function Weather() {
14      const [isModalVisible, setIsModalVisible] = useState(false);
15      const [weatherData, setweatherData] = useState([]);
16      const [loading, setLoading] = useState(true);
17      const [buttonLoading, setButtonLoading] = useState(false);
18      const [chartData, setChartData] = useState({});
19      const [datesButtonName, setDatesButtonName] = useState("Όλα");
20      const [allDataLabels, setAllDataLabels] = useState([]);
21      const [allAvgTempData, setAllAvgTempData] = useState([]);
22      const [allAvgHumidityData, setAllAvgHumidityData] = useState([]);
```

Κώδικας 4.19: Δήλωση μεταβλητών με τις αρχικές τους καταστάσεις και οι μέθοδοι διαχείρισης τους

```
58      async function getWeatherData() {
59          await axios
60              .get("http://localhost:5000/weather/current")
61              .then(function (response) {
62                  let currentDateFromUnixTimestamp = new Date(
63                      response.data.data.dt * 1000
64                  );
65
66                  let currentDate = new Intl.DateTimeFormat("en-GB", {
67                      year: "numeric",
68                      month: "2-digit",
69                      day: "2-digit",
70                      hour: "2-digit",
71                      minute: "2-digit",
72                  }).format(currentDateFromUnixTimestamp);
73
74                  setweatherData({
75                      dt: currentDate,
76                      temp: response.data.data.main.temp,
77                      feels_like: response.data.data.main.feels_like,
78                      humidity: response.data.data.main.humidity,
79                      clouds: response.data.data.clouds.all,
80                      wind_speed: response.data.data.wind.speed,
81                      weatherDescription: response.data.data.weather[0].description,
82                      weatherIcon: response.data.data.weather[0].icon,
83                  });
84
85                  setLoading(false);
86              });
87      }
```

Κώδικας 4.20: Υλοποίηση της μεθόδου επεξεργασίας και εμφάνισης των τωρινών καιρικών δεδομένων

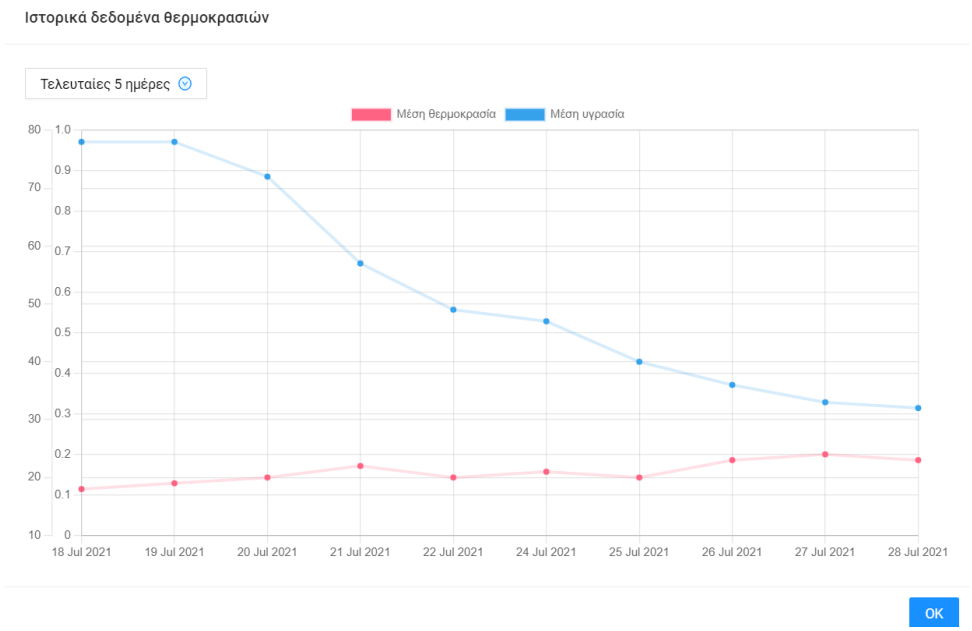
Στον Κώδικα 4.20, υλοποιείται η μέθοδος getWeatherData(), η οποία δέχεται την απάντηση του αιτήματός μας για τα τωρινά καιρικά δεδομένα και τα αποθηκεύει στη μεταβλητή currentDateFromUnixTimestamp. Ύστερα, μεταβάλουμε τη μορφή του περιεχομένου της μεταβλητής στην επιθυμητή, με τη χρήση της μεθόδου format(), τα αποθηκεύουμε και τα εμφανίζουμε στην πλατφόρμα.

Στον Κώδικα 4.21 φαίνεται η υλοποίηση της μεθόδου getHistoricalWeatherData(), η οποία δέχεται τα ιστορικά δεδομένα ως απάντηση του αιτήματός μας, τα μορφοποιεί και τα αποθηκεύει στη βάση. Έπειτα, καλεί τις μεθόδους setAllDataLabels(), setAllAvgTempData() και setAllAvgHumidityData() για να μπορούν όλα αυτά τα δεδομένα να χρησιμοποιηθούν στην συνέχεια στα διαγράμματα.

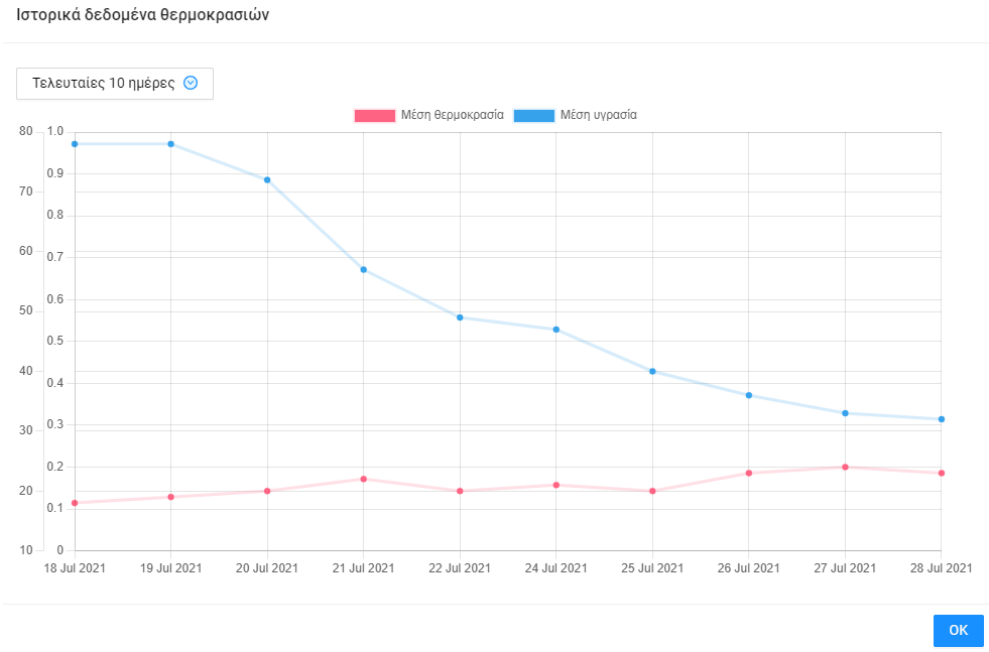
```
async function getHistoricalWeatherData() {
    await axios.get("/weather/history").then(function (response) {
        let dataLabels = [];
        let avgTempData = [];
        let avgHumidityData = [];

        for (let i = 0; i < response.data.data.length; i++) {
            let formatedDate = moment(response.data.data[i].day).format(
                "DD MMM YYYY"
            );

            dataLabels.push(formatedDate);
            avgTempData.push(response.data.data[i].avgTemp);
            avgHumidityData.push(response.data.data[i].avgHumidity);
        }

        setAllDataLabels(dataLabels);
        setAllAvgTempData(avgTempData);
        setAllAvgHumidityData(avgHumidityData);

        setChartData({
            labels: dataLabels,
            datasets: [
                {
                    label: "Μέση θερμοκρασία",
                    data: avgTempData,
                    fill: false,
                    backgroundColor: "rgb(255, 99, 132)",
                    borderColor: "rgba(255, 99, 132, 0.2)",
                    yAxisID: "y-axis",
                },
                {
                    label: "Μέση υγρασία",
                    data: avgHumidityData,
                    fill: false,
                    backgroundColor: "rgb(54, 162, 235)",
                    borderColor: "rgba(54, 162, 235, 0.2)",
                    yAxisID: "y-axis",
                },
            ],
        });
    });
}
```

Κώδικας 4.21: Υλοποίηση της μεθόδου επεξεργασίας των ιστορικών καιρικών δεδομένων και τοποθέτηση τους στο διάγραμμα

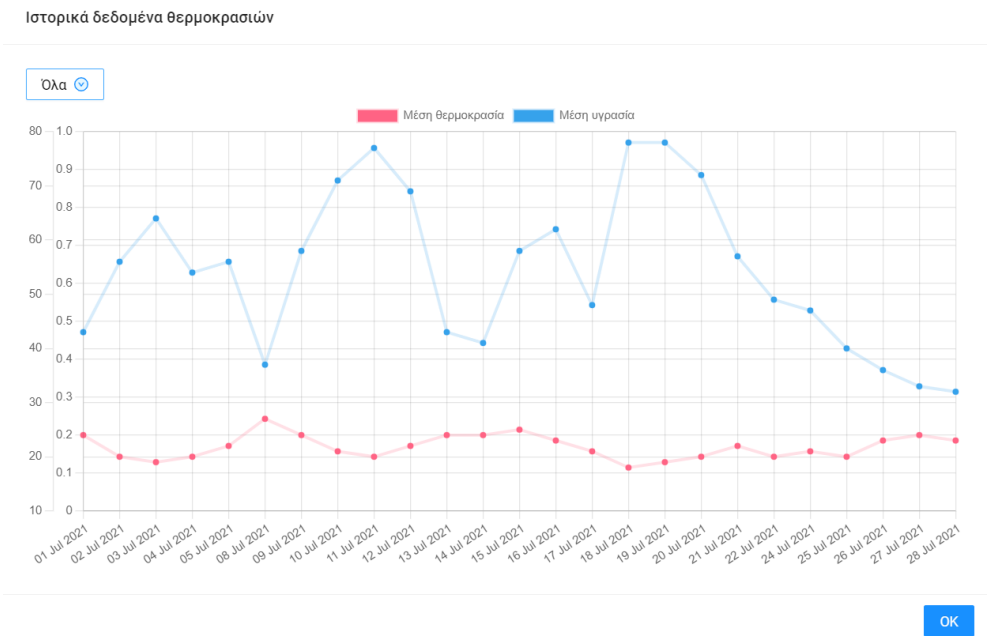
Στην Εικόνα 4.13, Εικόνα 4.14 και Εικόνα 4.15, φαίνονται τα διαγράμματα της πλατφόρμας με τα καιρικά δεδομένα ανά κάποιες ημέρες. Η πρώτη επιλογή αντιστοιχεί στην προβολή δεδομένων των τελευταίων 5 ημερών, η δεύτερη επιλογή για τις τελευταίες 10 μέρες και η τελευταία για την προβολή όλων των δεδομένων της βάσης στο γράφημα. Οι κάθετοι άξονες συμβολίζουν τη μέση θερμοκρασία και τη μέση υγρασία αντίστοιχα, ενώ ο οριζόντιος άξονας τις ημερομηνίες. Τέλος, κάθε κουκίδα αντιστοιχεί στη μέση θερμοκρασία ή μέση υγρασία της συγκεκριμένης ημέρας.



Εικόνα 4.13: Διάγραμμα προβολής των καιρικών δεδομένων της βάσης για τις τελευταίες 5 ημέρες



Εικόνα 4.14: Διάγραμμα προβολής των καιρικών δεδομένων της βάσης για τις τελευταίες 10 ημέρες



Εικόνα 4.15: Διάγραμμα προβολής όλων των καιρικών δεδομένων της βάσης

```
function handleMenuClick(e) {
    let filteredDataLabels = [];
    let filteredAvgTempData = [];
    let filteredAvgHumidityData = [];

    if (e.key === "1") {
        setDatesButtonName("Τελευταίες 5 ημέρες");
        for (let i = 0; i < allDataLabels.length; i++) {
            if (allDataLabels.length - i <= 5) {
                filteredDataLabels.push(allDataLabels[i]);
                filteredAvgTempData.push(allAvgTempData[i]);
                filteredAvgHumidityData.push(allAvgHumidityData[i]);
            }
        }
        setChartData({
            labels: filteredDataLabels,
            datasets: [
                {
                    label: "Μέση θερμοκρασία",
                    data: filteredAvgTempData,
                    fill: false,
                    backgroundColor: "rgb(255, 99, 132)",
                    borderColor: "rgba(255, 99, 132, 0.2)",
                    yAxisID: "y-axis",
                },
                {
                    label: "Μέση υγρασία",
                    data: filteredAvgHumidityData,
                    fill: false,
                    backgroundColor: "rgb(54, 162, 235)",
                    borderColor: "rgba(54, 162, 235, 0.2)",
                    yAxisID: "y-axis",
                },
            ],
        });
    } else if (e.key === "2") {
        setDatesButtonName("Τελευταίες 10 ημέρες");
        for (let i = 0; i < allDataLabels.length; i++) {
            if (allDataLabels.length - i <= 10) {
                filteredDataLabels.push(allDataLabels[i]);
                filteredAvgTempData.push(allAvgTempData[i]);
                filteredAvgHumidityData.push(allAvgHumidityData[i]);
            }
        }
```

Κώδικας 4.22: Η μέθοδος διαχείρισης του μενού επιλογών για τα προβαλλόμενα δεδομένα στο διάγραμμα ιστορικών καιρικών δεδομένων

Στους Κώδικα 4.22 και Κώδικα 4.23 υλοποιείται η διαχείριση των δεδομένων των παραπάνω διαγραμμάτων. Στην αρχή, φιλτράρεται ο πίνακας με τα καιρικά δεδομένα, το μέγεθος του οποίου εξαρτάται από την επιλογή του χρήστη. Η πρώτη επιλογή όπως προαναφέρθηκε, αντιστοιχεί στην προβολή δεδομένων των τελευταίων 5 ημερών, η δεύτερη επιλογή για τις τελευταίες 10 μέρες και η τελευταία για την προβολή όλων των δεδομένων της βάσης στο γράφημα. Έπειτα, αποθηκεύεται η μέση θερμοκρασία, μέση υγρασία και οι ημερομηνίες σε προσωρινούς πίνακες που χρησιμοποιούνται παρακάτω για την εμφάνιση των διαγραμμάτων.

```
177             setChartData({
178                 labels: filteredDataLabels,
179                 datasets: [
180                     {
181                         label: "Μέση θερμοκρασία",
182                         data: filteredAvgTempData,
183                         fill: false,
184                         backgroundColor: "rgb(255, 99, 132)",
185                         borderColor: "rgba(255, 99, 132, 0.2)",
186                         yAxisID: "y-axis",
187                     },
188                     {
189                         label: "Μέση υγρασία",
190                         data: filteredAvgHumidityData,
191                         fill: false,
192                         backgroundColor: "rgb(54, 162, 235)",
193                         borderColor: "rgba(54, 162, 235, 0.2)",
194                         yAxisID: "y-axis",
195                     },
196                 ],
197             });
198         } else if (e.key === "3") {
199             setDatesButtonName("Όλα");
200             setChartData({
201                 labels: allDataLabels,
202                 datasets: [
203                     {
204                         label: "Μέση θερμοκρασία",
205                         data: allAvgTempData,
206                         fill: false,
207                         backgroundColor: "rgb(255, 99, 132)",
208                         borderColor: "rgba(255, 99, 132, 0.2)",
209                         yAxisID: "y-axis",
210                     },
211                     {
212                         label: "Μέση υγρασία",
213                         data: allAvgHumidityData,
214                         fill: false,
215                         backgroundColor: "rgb(54, 162, 235)",
216                         borderColor: "rgba(54, 162, 235, 0.2)",
217                         yAxisID: "y-axis",
218                     },
219                 ],
220             });
221         }
222     }
```

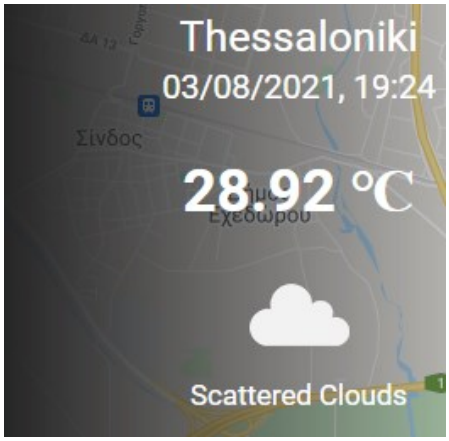
Κώδικας 4.23: Η συνέχεια της μεθόδου διαχείρισης του μενού επιλογών για τα προβαλλόμενα δεδομένα στο διάγραμμα ιστορικών καιρικών δεδομένων

```
224     const dropdownMenu = (
225         <Menu onClick={handleMenuClick}>
226             <Menu.Item key="1">
227                 <span style={{ color: "black" }}>Τελευταίες 5 ημέρες</span>
228             </Menu.Item>
229             <Menu.Item key="2">
230                 <span style={{ color: "black" }}>Τελευταίες 10 ημέρες</span>
231             </Menu.Item>
232             <Menu.Item key="3">
233                 <span style={{ color: "black" }}>Όλα</span>
234             </Menu.Item>
235         </Menu>
236     );
```

Κώδικας 4.24: Το μενού επιλογών για την εμφάνιση των διαγραμμάτων

Στον Κώδικα 4.24, φαίνεται η δήλωση και η μορφοποίηση του μενού επιλογών για το φιλτράρισμα των δεδομένων που προβάλλονται στα διαγράμματα.

Στη συνέχεια ο Κώδικας 4.25 είναι η υλοποίηση της Εικόνας 4.16, όπου γίνεται η προβολή ενός μέρους του πίνακα των καιρικών δεδομένων στην πλατφόρμα. Πιο συγκεκριμένα, στον πίνακα εμφανίζεται το όνομα της πόλης, η ημερομηνία και ώρα, η θερμοκρασία, μια μικρή περιγραφή του καιρού κι ένα εικονίδιο που να αντιπροσωπεύει αυτόν τον καιρό.



Εικόνα 4.16: Το πρώτο μέρος του πίνακα των καιρικών δεδομένων

```
238     return (
239         <div>
240             {!loading && (
241                 <div style={{ marginTop: "30px" }}>
242                     <div className="main-temp-data">
243                         <p className="city">Thessaloniki</p>
244                         <p className="date">{weatherData.dt}</p>
245                         <p className="main-temp">{weatherData.temp} &#8451;</p>
246                         <p className="main-description">
247                             <img
248                                 src={`https://openweathermap.org/img/wn/${weatherData.weatherIcon}@2x.png`}
249                                 alt="Weather icon"
250                             />
251                             <span className="weather-description">
252                                 {weatherData.weatherDescription}
253                             </span>
254                         </p>
255                     </div>
```

Κώδικας 4.25: Προβολή ενός μέρους του πίνακα των καιρικών δεδομένων στην πλατφόρμα

Στον Κώδικα 4.26 στις γραμμές 277 έως 288 παρουσιάζεται ένα δείγμα της δομής του πίνακα για τα καιρικά δεδομένα.

Έπειτα, στις γραμμές 292 έως 298 γίνεται χρήση του κουμπιού και ορίζεται, πως με το πάτημά του, θα εμφανίζεται το παράθυρο με τα ιστορικά δεδομένα των θερμοκρασιών. Στις γραμμές 306 έως 320 δηλώνεται και η διαχείριση των υπολοίπων λειτουργιών του πίνακα, όπως το κλείσιμο του παραθύρου και το μενού επιλογών για το φιλτράρισμα των δεδομένων.

Στην γραμμή 313 ορίζεται πως η ενεργοποίηση του μενού επιλογών που βρίσκεται στο παράθυρο των διαγραμμάτων γίνεται όταν πατηθεί από το χρήστη. Στις γραμμές 314 έως 318 γίνεται η μορφοποίηση του και στη γραμμή 319, δηλώνεται πως τα αποθηκευμένα στον πίνακα chartData δεδομένα θα εμφανιστούν στα διαγράμματα.

```
                                <div className="info-area">
                                    <p className="secondary-info">Clouds</p>
                                    <p className="main-info">{weatherData.clouds} %</p>
                                </div>
                            </td>
                            <td>
                                <div className="info-area">
                                    <p className="secondary-info">Wind</p>
                                    <p className="main-info">{weatherData.wind_speed} km/hr</p>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td colSpan="2">
                                {buttonLoading ? (
                                    <Button type="primary" onClick={showModal}>
                                        <LoadingOutlined />
                                    </Button>
                                ) : (
                                    <Button type="primary" onClick={showModal}>
                                        <HistoryOutlined /> Θερμοκρασίες
                                    </Button>
                                )}
                            </td>
                        </tr>
                    </tbody>
                </table>

                <Modal
                    title="Ιστορικά δεδομένα θερμοκρασιών"
                    visible={isModalVisible}
                    onOk={handleOk}
                    onCancel={handleCancel}
                    width={1000}
                    cancelButtonProps={{ style: { display: "none" } }}
                >
                    <Dropdown overlay={dropdownMenu} trigger={["click"]}>
                        <Button>
                            <span style={{ color: "black" }}>{datesButtonName}</span>{" "}
                            <DownCircleTwoTone twoToneColor="" />
                        </Button>
                    </Dropdown>
                    <Line data={chartData} options={options} />
                </Modal>
            </div>
        )}
```

Κώδικας 4.26: Μέρος της δομής του πίνακα για τα καιρικά δεδομένα και η διαχείριση του κουμπιού εμφάνισης των ιστορικών καιρικών δεδομένων

```
    async function showModal() {
        setButtonLoading(true);
        await getHistoricalWeatherData();
        setIsModalVisible(true);
        setButtonLoading(false);
    }

    const handleOk = () => {
        setIsModalVisible(false);
    };

    const handleCancel = () => {
        setIsModalVisible(false);
    };

    useEffect(() => {
        getWeatherData();
    }, []);
```

Κώδικας 4.27: Οι μέθοδοι που καλούνται ανάλογα με την επιλογή του χρήστη στο παράθυρο προβολής των ιστορικών καιρικών δεδομένων

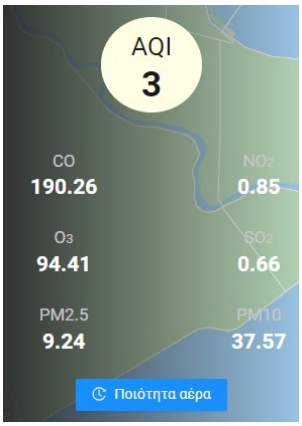
Στον Κώδικα 4.27 φαίνονται οι κλήσεις των μεθόδων που γίνονται αναλόγως την κάθε κίνηση του χρήστη στο παράθυρο της ιστορικότητας των δεδομένων του καιρού.

4.4.5 Ποιότητα Αέρα

Στην Εικόνα 4.17 φαίνεται ο πίνακας της ποιότητας του αέρα της πόλης. Στον πίνακα προβάλλεται το CO (μονοξείδιο του άνθρακα), O_3 (όζον), $PM_{2.5}$ (εισπνεύσιμα σωματίδια με διάμετρο εως 2.5 μm), NO_2 (διοξείδιο του αζώτου), SO_2 (διοξείδιο του θείου) και το PM_{10} (εισπνεύσιμα σωματίδια με διάμετρο έως 10 mm), που είναι τα πιο σημαντικά στοιχεία κατά την έρευνά μας. Η υλοποίηση του βρίσκεται στον Κώδικα 4.26.

Στις επόμενες γραμμές κώδικα φαίνεται η ασύγχρονη μέθοδος getAirQualityData(), η οποία εφόσον δεχθεί ως απάντηση τα δεδομένα για την κατάσταση των τωρινών καιρικών συνθηκών από το αίτημα, κάνει χρήση της μεθόδου setAirQualityData() για να τα εμφανίζει στη σελίδα μας. (Κώδικας 4.28).

Στην συνέχεια είναι η μέθοδος getHistoricalAirQualityData(), η οποία δέχεται τα ιστορικά δεδομένα (ποιότητα αέρα και ημερομηνία) από το αίτημά μας, τα μετατρέπει στην επιθυμητή μορφή με τη βοήθεια της format() μεθόδου και τέλος αποθηκεύει τις ημερομηνίες στον πίνακα dataLabels και τις μετρήσεις στον πίνακα aqiData (Κώδικας 4.29).



Εικόνα 4.17: Ο πίνακας της ποιότητας του αέρα

```
45  async function getAirQualityData() {
46    await axios.get("/airquality/current").then(function (response) {
47      setAirQualityData({
48        dt: response.data.data.dt,
49        aqi: response.data.data.aqi,
50        co: response.data.data.co,
51        no2: response.data.data.no2,
52        o3: response.data.data.o3,
53        pm2_5: response.data.data.pm2_5,
54        pm10: response.data.data.pm10,
55        so2: response.data.data.so2,
56      });
57    });
58  }
```

Κώδικας 4.28: Αποθήκευση της τωρινής ποιότητας του αέρα

```
async function getHistoricalAirQualityData() {
  let dataLabels = [];
  let aqiData = [];
  await axios.get("/airquality/history").then(function (response) {
    for (let i = 0; i < response.data.data.length; i++) {
      let stringToDate = new Date(response.data.data[i].dt);
      let changeDateFormat = new Intl.DateTimeFormat("en-GB", {
        year: "numeric",
        month: "2-digit",
        day: "2-digit",
        hour: "2-digit",
        minute: "2-digit",
      }).format(stringToDate);

      dataLabels.push(changeDateFormat);
      aqiData.push(response.data.data[i].aqi);
    }
```

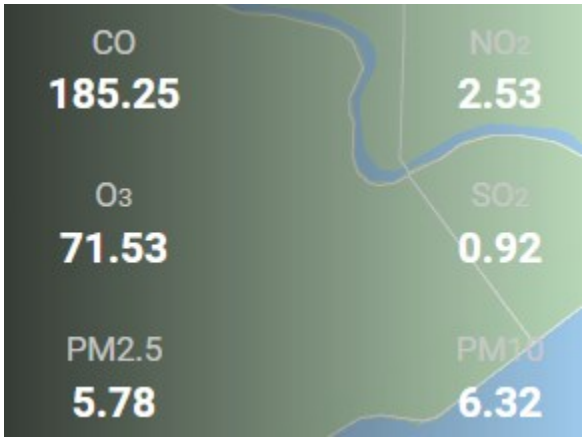
Κώδικας 4.29: Επεξεργασία και αποθήκευση ιστορικών δεδομένων

```
<table style={{ width: "100%", textAlign: "center" }}>
  <tbody>
    <tr>
      <td>
        <div className="airqu-info-area">
          <p className="airqu-secondary-info">CO</p>
          <p className="airqu-main-info">{airQualityData.co}</p>
        </div>
      </td>
      <td>
        <div className="airqu-info-area">
          <p className="airqu-secondary-info">
            NO<span className="airQuality_metrics_pointer">2</span>
          </p>
          <p className="airqu-main-info">{airQualityData.no2}</p>
        </div>
      </td>
    </tr>
    <tr>
      <td>
        <div className="airqu-info-area">
          <p className="airqu-secondary-info">
            O<span className="airQuality_metrics_pointer">3</span>
          </p>
          <p className="airqu-main-info">{airQualityData.o3}</p>
        </div>
      </td>
```

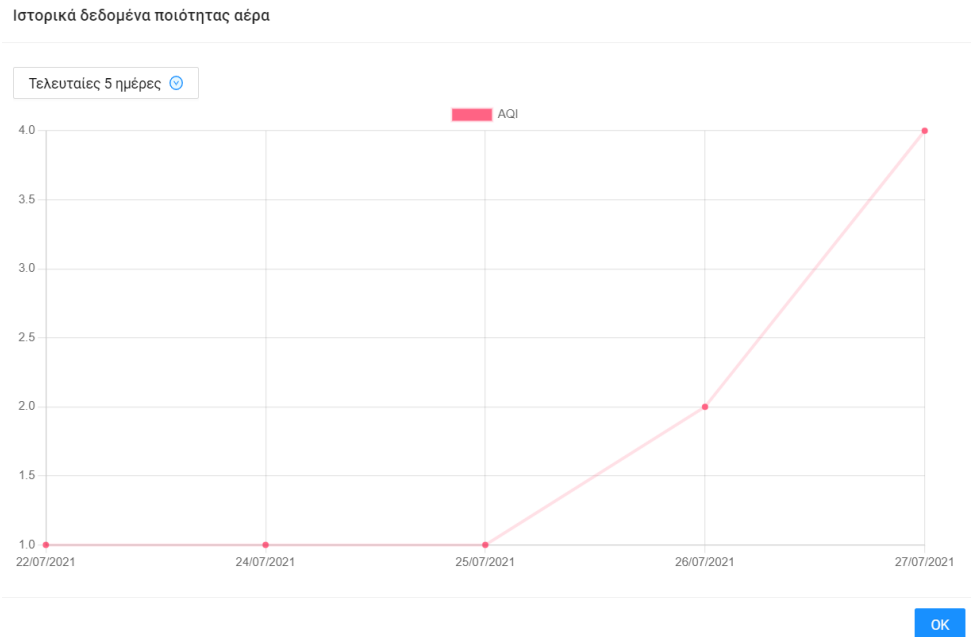
Κώδικας 4.30: Μέρος του πίνακα με τους δείκτες μέτρησης της ποιότητας του αέρα

Στον Κώδικα 4.30 προβάλλεται ένα μέρος της δομής του πίνακα με τους δείκτες μέτρησης της ποιότητας του αέρα. Αναλυτικότερα, για την εμφάνιση των μετρήσεων γίνεται χρήση του πίνακα airQualityData (γραμμές 109,117 και 127), στον οποίο έχουν αποθηκευτεί όλα τα τωρινά δεδομένα της ποιότητας του αέρα και ζητάμε να εμφανιστούν σε αυτόν οι επιθυμητές μετρήσεις.

Η ίδια λογική ακολουθήθηκε και για τις υπόλοιπες μετρήσεις που εμφανίζονται στον πίνακα (Εικόνα 4.18).



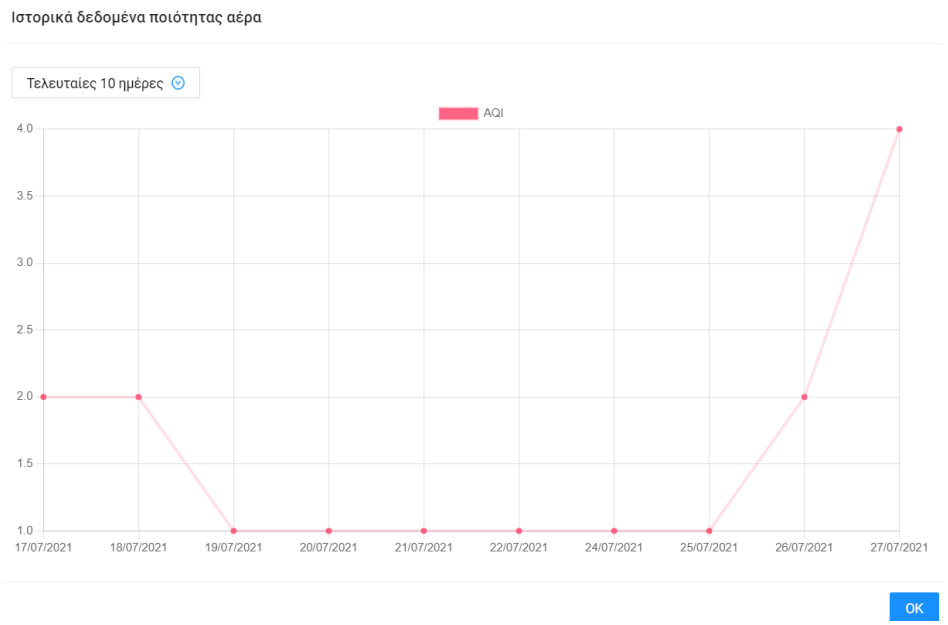
Εικόνα 4.18: Η τελική μορφή του πίνακα με τους δείκτες μέτρησης της ποιότητας του αέρα



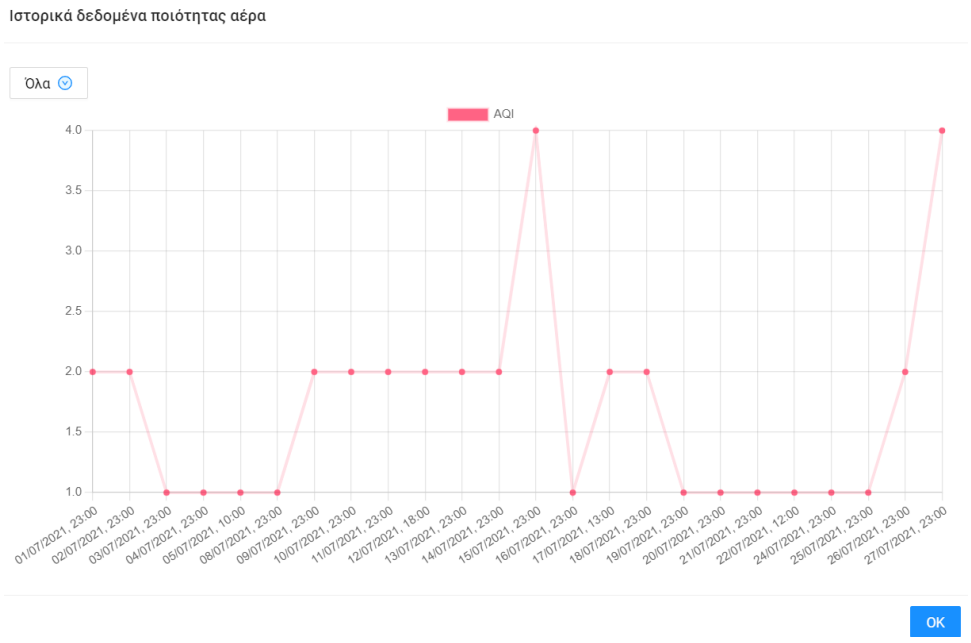
Εικόνα 4.19: Διάγραμμα προβολής των καιρικών δεδομένων της βάσης για τις τελευταίες 5 ημέρες

Στην Εικόνα 4.19, Εικόνα 4.20 και Εικόνα 4.21 φαίνονται τα διαγράμματα της πλατφόρμας με τα δεδομένα της ποιότητας του αέρα ανά κάποιες ημέρες. Η πρώτη επιλογή αντιστοιχεί στην προβολή δεδομένων των τελευταίων 5 ημερών, η δεύτερη επιλογή για τις τελευταίες 10 μέρες και η τελευταία για την προβολή όλων των δεδομένων που βρίσκονται στη βάση, στο γράφημα. Ο κάθετος άξονας συμβολίζει την τιμή του δείκτη AQI, ενώ ο οριζόντιος άξονας τις ημερομηνίες. Τέλος, κάθε κουκίδα αντιστοιχεί στην τιμή του δείκτη AQI για τη συγκεκριμένη ημέρα.

Ο Κώδικας 4.29 γράφτηκε για τη δημιουργία του κουμπιού, το οποίο όταν πατηθεί εμφανίζεται το διάγραμμα των ιστορικών δεδομένων για την ποιότητα του καιρού (Εικόνα 4.17).



Εικόνα 4.20: Διάγραμμα προβολής των καιρικών δεδομένων της βάσης για τις τελευταίες 10 ημέρες



Εικόνα 4.21: Διάγραμμα προβολής όλων των δεδομένων της ποιότητας του αέρα

Στον Κώδικα 4.31, 4.32 και 4.33 υλοποιείται η διαχείριση των δεδομένων των παραπάνω διαγραμμάτων. Στην αρχή, φιλτράρεται ο πίνακας με τα δεδομένα της ποιότητας του αέρα, το μέγεθος του οποίου εξαρτάται από την επιλογή του χρήστη. Η πρώτη επιλογή όπως αναφέρθηκε και σε προηγούμενη ενότητα, αντιστοιχεί στην προβολή δεδομένων των τελευταίων 5 ημερών, η δεύτερη επιλογή για τις τελευταίες 10 μέρες και η τελευταία για την προβολή όλων των δεδομένων της βάσης στο γράφημα. Έπειτα, αποθηκεύεται ο δείκτης AQI και οι ημερομηνίες σε προσωρινούς πίνακες που χρησιμοποιούνται παρακάτω για την εμφάνιση των διαγραμμάτων.

```
107     function handleMenuClick(e) {
108         let filteredDataLabels = [];
109         let filteredAqiData = [];
110 
111         if (e.key === "1") {
112             setDatesButtonName("Τελευταίες 5 ημέρες");
113 
114             for (let i = 0; i < allDataLabels.length; i++) {
115                 if (allDataLabels.length - i <= 5) {
116                     const splitDate = allDataLabels[i].split(",");
117                     const datePart = splitDate[0];
118 
119                     filteredDataLabels.push(datePart);
120                     filteredAqiData.push(allAqiData[i]);
121                 }
122             }
123 
124             setChartData({
125                 labels: filteredDataLabels,
126                 datasets: [
127                     {
128                         label: "AQI",
129                         data: filteredAqiData,
130                         fill: false,
131                         backgroundColor: "rgb(255, 99, 132)",
132                         borderColor: "rgba(255, 99, 132, 0.2)",
133                     },
134                 ],
135             });
```

Κώδικας 4.31: Η μέθοδος διαχείρισης του μενού επιλογών για τα προβαλλόμενα δεδομένα στο διάγραμμα ιστορικών δεδομένων ποιότητας αέρα

```
136         } else if (e.key === "2") {
137             setDatesButtonName("Τελευταίες 10 ημέρες");
138 
139             for (let i = 0; i < allDataLabels.length; i++) {
140                 if (allDataLabels.length - i <= 10) {
141                     const splitDate = allDataLabels[i].split(",");
142                     const datePart = splitDate[0];
143 
144                     filteredDataLabels.push(datePart);
145                     filteredAqiData.push(allAqiData[i]);
146                 }
147             }
148 
149             setChartData({
150                 labels: filteredDataLabels,
151                 datasets: [
152                     {
153                         label: "AQI",
154                         data: filteredAqiData,
155                         fill: false,
156                         backgroundColor: "rgb(255, 99, 132)",
157                         borderColor: "rgba(255, 99, 132, 0.2)",
158                     },
159                 ],
160             });
```

Κώδικας 4.32: Συνέχεια της μεθόδου διαχείρισης του μενού επιλογών για τα προβαλλόμενα δεδομένα στο διάγραμμα ιστορικών δεδομένων ποιότητας αέρα

Επίσης στον παρακάτω Κώδικα 4.33 στις γραμμές 179 έως 190 φαίνεται η δήλωση και η μορφοποίηση του μενού επιλογών για το φιλτράρισμα των δεδομένων που προβάλλονται στα διαγράμματα για τα δεδομένα της πόλης.

```
161         } else if (e.key === "3") {
162             setDatesButtonName("Όλα");
163 
164             setChartData({
165                 labels: allDataLabels,
166                 datasets: [
167                     {
168                         label: "AQI",
169                         data: allAqiData,
170                         fill: false,
171                         backgroundColor: "rgb(255, 99, 132)",
172                         borderColor: "rgba(255, 99, 132, 0.2)",
173                     },
174                 ],
175             });
176         }
177     }
178 
179     const dropdownMenu = (
180         <Menu onClick={handleMenuClick}>
181             <Menu.Item key="1">
182                 <span style={{ color: "black" }}>Τελευταίες 5 ημέρες</span>
183             </Menu.Item>
184             <Menu.Item key="2">
185                 <span style={{ color: "black" }}>Τελευταίες 10 ημέρες</span>
186             </Menu.Item>
187             <Menu.Item key="3">
188                 <span style={{ color: "black" }}>Όλα</span>
189             </Menu.Item>
190         </Menu>
191     );
```

Κώδικας 4.33: Συνέχεια της μεθόδου διαχείρισης του μενού επιλογών για τα προβαλλόμενα δεδομένα στο διάγραμμα ιστορικών δεδομένων ποιότητας αέρα

Στον Κώδικα 4.34 στις γραμμές 239 έως 251 παρουσιάζεται ένα δείγμα της δομής του πίνακα για τα καιρικά δεδομένα.

Έπειτα, στις γραμμές 254 έως 261 γίνεται χρήση του κουμπιού και ορίζεται, πως με το πάτημά του, θα εμφανίζονται τα ιστορικά δεδομένα της ποιότητας του αέρα. Στις γραμμές 268 έως 274 δηλώνεται και η διαχείριση των υπολοίπων λειτουργιών του πίνακα όπως το κλείσιμο του παραθύρου και το μενού επιλογών για το φιλτράρισμα των δεδομένων.

Στη γραμμή 276 ορίζεται πως η ενεργοποίηση του μενού επιλογών που βρίσκεται στο παράθυρο των διαγραμμάτων γίνεται όταν πατηθεί από το χρήστη. Στις γραμμές 277 έως 281 γίνεται η μορφοποίηση του και στη γραμμή 282 δηλώνεται πως τα δεδομένα που θα εμφανιστούν στο διάγραμμα όσα είναι αποθηκευμένα στον πίνακα chartData.

```
                    <tr>
                        <td>
                            <div className="airqu-info-area">
                                <p className="airqu-secondary-info">PM2.5</p>
                                <p className="airqu-main-info">{airQualityData.pm2_5}</p>
                            </div>
                        </td>
                        <td>
                            <div className="airqu-info-area">
                                <p className="airqu-secondary-info">PM10</p>
                                <p className="airqu-main-info">{airQualityData.pm10}</p>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td colSpan="2">
                        {buttonLoading ? (
                            <Button type="primary" onClick={showModal}>
                                <LoadingOutlined />
                            </Button>
                        ) : (
                            <Button type="primary" onClick={showModal}>
                                <HistoryOutlined /> Ποιότητα αέρα
                            </Button>
                        )}
                        </td>
                    </tr>
                </tbody>
            </table>

            <Modal
                title="Ιστορικά δεδομένα ποιότητας αέρα"
                visible={isModalVisible}
                onOk={handleOk}
                onCancel={handleCancel}
                width={1000}
                cancelButtonProps={{ style: { display: "none" } }}
            >
                <Dropdown overlay={dropdownMenu} trigger={["click"]}>
                    <Button>
                        <span style={{ color: "black" }}>{datesButtonName}</span>{" "}
                        <DownCircleTwoTone twoToneColor="" />
                    </Button>
                </Dropdown>
                <Line data={chartData} options={options} />
            </Modal>
        </div>
```

Κώδικας 4.34: Χρήση του κουμπιού προβολής των ιστορικών δεδομένων της ποιότητας του αέρα και διαχείριση των λειτουργιών του

```
const options = {
  scales: {
    yAxes: [
      {
        ticks: {
          beginAtZero: true,
        },
      },
    ],
  },
};

async function showModal() {
  await getHistoricalAirQualityData();
  setIsModalVisible(true);
}

const handleOk = () => {
  setIsModalVisible(false);
};

const handleCancel = () => {
  setIsModalVisible(false);
};

useEffect(() => {
  getAirQualityData();

  setLoading(false);
}, []);
```

Κώδικας 4.35: Οι μέθοδοι που καλούνται ανάλογα με την επιλογή του χρήστη στο παράθυρο προβολής των ιστορικών δεδομένων της ποιότητας του αέρα

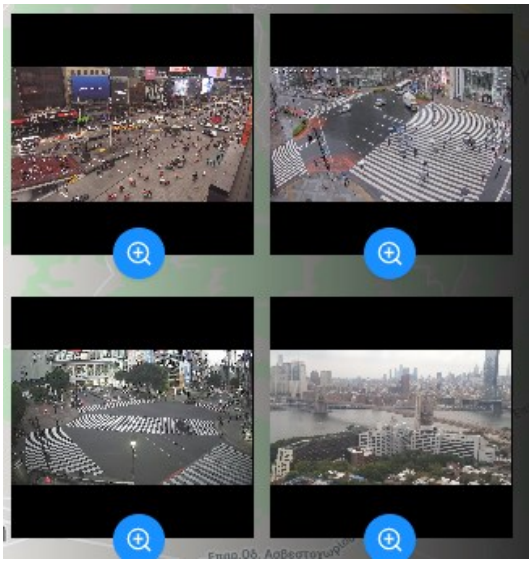
Τέλος, στον Κώδικα 4.35 φαίνονται οι κλήσεις των μεθόδων που γίνονται αναλόγως την κάθε κίνηση του χρήστη στο παράθυρο της ιστορικότητας των δεδομένων της ποιότητας του αέρα.

4.4.6 Κάμερες

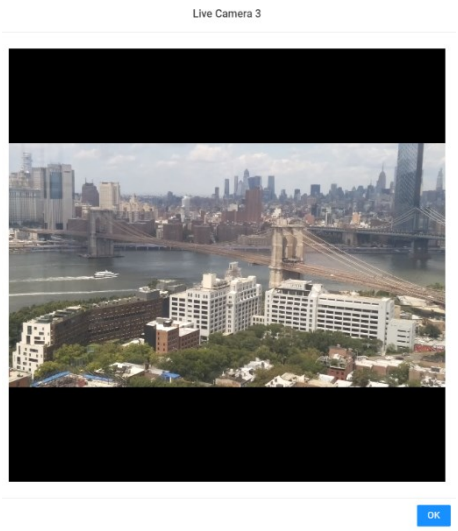
Οι κάμερες της πόλης δίνουν την δυνατότητα στον διαχειριστή να επιβλέπει την ροή της κίνησης των πολιτών. Δυστυχώς, η Θεσσαλονίκη δε διαθέτει ελεύθερες για χρήση κάμερες, οπότε για το λόγο αυτό, στην παρούσα εργασία χρησιμοποιήθηκαν απευθείας μετάδοσης βίντεο από το YouTube, άλλων πόλεων του κόσμου, που φαίνονται στον παρακάτω πίνακα της πλατφόρμας (Εικόνα 4.22).

Επιπλέον, υπάρχει η δυνατότητα όταν πατηθεί το μπλε κουμπί που βρίσκεται κεντρικά στο κάτω μέρος κάθε κάμερας με τον μεγεθυντικό φακό, να ανοίγει ένα καινούργιο παράθυρο για καλύτερη προβολή του βίντεο της κάμερας (Εικόνα 4.23).

Παρακάτω παρουσιάζεται η υλοποίηση όσων αναφέρθηκαν παραπάνω. Αρχικά, στον παρακάτω Κώδικα 4.36 φαίνεται μια από τις προαναφερόμενες δυνατότητες της React, που ορίζουμε ποια μέθοδος θα διαχειρίζεται ποια μεταβλητή. Για να γίνει αυτό, αρχικά πρέπει να δηλωθεί το όνομα της μεταβλητής και το όνομα της μεθόδου με την οποία αλλάζουμε την τιμή της μεταβλητής μέσα σε αγκύλες. Τέλος δίνεται και η αρχική κατάσταση της μεταβλητής (π.χ. false).



Εικόνα 4.22: Ο πίνακας που περιέχει τις κάμερες της πόλης



Εικόνα 4.23: Ένα από τα παράθυρα μεγέθυνσης της προβολής του βίντεο

```
6  export default function Cameras({
7      mapZoom,
8      setMapZoom,
9      showTraffic,
10     setShowTraffic,
11     showFire,
12     setShowFire,
13 }) {
14     const [modalTitle, setModalTitle] = useState("");
15     const [modalVideoID, setModalVideoID] = useState("");
16     const [isModalVisible, setIsModalVisible] = useState(false);
```

Κώδικας 4.36: Δήλωση μεταβλητών με τις αρχικές τους καταστάσεις και οι μέθοδοι διαχείρισης τους

```
18     const mapZoomIn = () => {
19         setMapZoom(mapZoom + 1);
20     };
21     const mapZoomOut = () => {
22         setMapZoom(mapZoom - 1);
23     };
24     const showTrafficHandler = () => {
25         setShowTraffic(!showTraffic);
26     };
27
28     const showFireHandler = () => {
29         setShowFire(!showFire);
30     };
31
32     const showCameraModal = (modalTitle, videoID) => {
33         setModalTitle(modalTitle);
34         setModalVideoID(videoID);
35         setIsModalVisible(true);
36     };
37
38     const handleOk = () => {
39         setIsModalVisible(false);
40     };
41
42     const handleCancel = () => {
43         setIsModalVisible(false);
44     };
```

Κώδικας 4.37: Οι μέθοδοι που καλούνται ανάλογα με την κάθε κίνηση του χρήστη

Στη συνέχεια, φαίνονται οι μέθοδοι που διαχειρίζονται κάποιες λειτουργίες του χάρτη και των παραθύρων που προβάλλουν σε μεγέθυνση τα βίντεο των καμερών. Οι γραμμές 18 έως 23 είναι υπεύθυνες για τη μεγέθυνση και τη σμίκρυνση του χάρτη. Οι γραμμές 23 έως 30 διαχειρίζονται την προβολή κίνησης και των πυρκαγιών αντίστοιχα, οι γραμμές 32 έως 36 θέτουν εμφανές το βίντεο της κάμερας, τον τίτλο της και το id της και τέλος, οι γραμμές 38 έως 44 διαχειρίζονται τη λειτουργία ανοίγματος και κλεισίματος του παραθύρου που προβάλλουν σε μεγέθυνση τα βίντεο των καμερών (Κώδικας 4.37).

Στον Κώδικα 4.38 και Κώδικα 4.39 φαίνεται η μορφοποίηση των κουμπιών που καλούν τις παραπάνω μεθόδους για τη μεγέθυνση του χάρτη, τη σμίκρυνση του, την εμφάνιση της κίνησης και την εμφάνιση των πυρκαγιών.

```
46      return (
47          <div style={{ marginTop: "2px", textAlign: "center" }}>
48              <Button
49                  type="primary"
50                  shape="circle"
51                  icon={<ZoomInOutlined />}
52                  onClick={() => {
53                      mapZoomIn();
54                  }}
55                  style={{
56                      position: "absolute",
57                      right: 100,
58                      top: 15,
59                  }}
60              />
61              <Button
62                  type="primary"
63                  shape="circle"
64                  icon={<ZoomOutOutlined />}
65                  onClick={() => {
66                      mapZoomOut();
67                  }}
68                  style={{
69                      position: "absolute",
70                      right: 50,
71                      top: 15,
72                  }}
73              />
```

Κώδικας 4.38: Μορφοποίηση των κουμπιών για την μεγέθυνση και την σμίκρυνση του χάρτη

```
74              <Button
75                  type="primary"
76                  onClick={showTrafficHandler}
77                  style={{
78                      position: "absolute",
79                      right: 50,
80                      top: 55,
81                  }}
82              >
83                  {showTraffic ? "Απόκρυψη κίνησης" : "Προβολή κίνησης"}
84              </Button>
85              <Button
86                  type="primary"
87                  onClick={showFireHandler}
88                  style={{
89                      position: "absolute",
90                      right: 210,
91                      top: 55,
92                  }}
93              >
94                  {showFire ? "Απόκρυψη πυρκαγιών" : "Προβολή πυρκαγιών"}
95              </Button>
```

Κώδικας 4.39: Μορφοποίηση των κουμπιών για την προβολή της κίνησης και των πυρκαγιών στον χάρτη

Στην συνέχεια, φαίνεται ένα μέρος του πίνακα που περιέχει τις κάμερες τις πόλεις. Σε αυτές τις γραμμές κώδικα αρχικά δηλώνεται το κουμπί το οποίο όταν πατηθεί, ανοίγει μια μεγαλύτερη εικόνα του βίντεο για καλύτερη προβολή. Επιπλέον, παρακάτω δίνεται η διεύθυνση από την οποία πήραμε το βίντεο και γίνεται η μορφοποίηση του τρόπου εμφάνισης του, όπως η σίγαση του κ.α. (Κώδικας 4.40). Η ίδια λογική υιοθετήθηκε και για την υλοποίηση των υπόλοιπων καμερών που εμφανίζονται στην πλατφόρμα.

```
<table>
    <tbody>
        <tr style={{ height: "200px" }}>
            <td style={{ paddingRight: "10px" }}>
                <Button
                    type="primary"
                    shape="circle"
                    icon={<ZoomInOutlined />}
                    onClick={() => {
                        showCameraModal("Live Camera 1", "AdUw5RdyZxI");
                    }}
                    style={{
                        position: "absolute",
                        right: 275,
                        top: 317,
                        zIndex: 10,
                    }}
                />
                <ReactPlayer
                    url="https://www.youtube.com/watch?v=AdUw5RdyZxI"
                    width="150px"
                    height="150px"
                    playing={true}
                    muted={true}
                    style={{
                        position: "absolute",
                        right: 220,
                        top: 184,
                    }}
                />
            </td>
```

Κώδικας 4.40: Μέρος του πίνακα που περιέχει τις κάμερες της πόλης

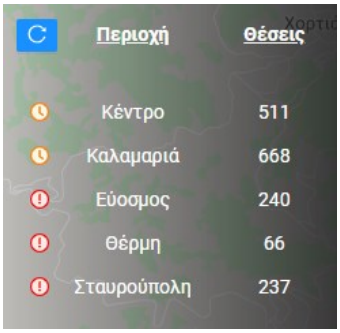
```
<Modal
    title={modalTitle}
    visible={isModalVisible}
    onOk={handleOk}
    onCancel={handleCancel}
    width={1000}
    cancelButtonProps={{ style: { display: "none" } }}
    style={{ display: "flex", textAlign: "center", alignItems: "center" }}
>
    <ReactPlayer
        url={`https://www.youtube.com/watch?v=${modalVideoID}`}
        width="650px"
        height="650px"
        playing={true}
        muted={true}
    />
</Modal>
```

Κώδικας 4.41:Διαχείριση και μορφοποίηση του παραθύρου μεγαλύτερης προβολής των βίντεο

Τέλος, στον Κώδικα 4.41 γίνεται χρήση των μεθόδων handleOk() και handleCancel() που διαχειρίζονται το κλείσιμο του παραθύρου μεγαλύτερης προβολής του βίντεο και μορφοποίηση του πίνακα με αυτά τα βίντεο που εμφανίζονται στην πλατφόρμα.

4.4.7 Πίνακας θέσεων στάθμευσης

Για την ενημέρωση του διαχειριστή δίνεται η δυνατότητα επίβλεψης της κίνησης των αμαξιών και κατ' επέκταση, των πολιτών στις θέσεις στάθμευσης της πόλης. Δυστυχώς, και σε αυτή την περίπτωση δε διατίθενται ανοιχτά δεδομένα της Θεσσαλονίκης για χρήση, οπότε δημιουργήθηκαν ψευδή δεδομένα που παρουσιάζονται στον πίνακα της Εικόνας 4.24.



Εικόνα 4.24: Ο πίνακας με τα δεδομένα θέσεων στάθμευσης όπως φαίνεται στη πλατφόρμα

```
let initialParkingData = [
    {
        id: 0,
        area: "Κέντρο",
        spots: 500,
        icon: <CheckCircleTwoTone twoToneColor="#52c41a" />,
    },
    {
        id: 1,
        area: "Καλαμαριά",
        spots: 500,
        icon: <ClockCircleTwoTone twoToneColor="#ff8000" />,
    },
    {
        id: 2,
        area: "Εύοσμος",
        spots: 500,
        icon: <ExclamationCircleTwoTone twoToneColor="#ff0000" />,
    },
    {
        id: 3,
        area: "Θέρμη",
        spots: 500,
        icon: <ExclamationCircleTwoTone twoToneColor="#ff0000" />,
    },
    {
        id: 4,
        area: "Σταυρούπολη",
        spots: 500,
        icon: <CheckCircleTwoTone twoToneColor="#52c41a" />,
    },
];
```

Εικόνα 4.42 : Δήλωση των περιοχών με θέσεις στάθμευσης της πόλης

Τα ψευδή δεδομένα έχουν τα χαρακτηριστικά του Κώδικα 4.42. Αναλυτικότερα, για κάθε περιοχή, αποθηκεύεται μέσα στον πίνακα initialParkingData ένας μοναδικός δείκτης ονόματι id, το όνομα της περιοχής, οι συνολικές θέσεις που διαθέτει και το εικονίδιο που αλλάζει ανάλογα με την πληρότητα της περιοχής.

Επόμενο βήμα για την υλοποίηση αυτής της ενότητας, είναι ο έλεγχος που γίνεται για την διαθεσιμότητα των ελεύθερων θέσεων στάθμευσης (Κώδικας 4.43). Για τον έλεγχο αυτό, αλλά και για την επιλογή μιας τυχαίας περιοχής μέσω του δείκτη id, χρησιμοποιούνται οι ακέραιες μεταβλητές randomSpots και parkingId, αντίστοιχα, που δημιουργούνται μέσω της μεθόδου random().

Έπειτα, ελέγχουμε τι αριθμός είναι η μεταβλητή randomSpots και ανάλογα με αυτό τον αριθμό εκτελούμε τον αντίστοιχο κώδικα. Ο κώδικας αυτός, είναι ίδιος για κάθε περίπτωση της μεταβλητής randomSpots, το μόνο που διαφέρει είναι το εικονίδιο που αποθηκεύεται κάθε φορά. Στην υλοποίηση του κώδικα ουσιαστικά αναζητείται το τυχαίο id που παράχθηκε από τη μέθοδο Random() στον πίνακα parking και εφόσον βρεθεί ανανεώνουμε τα αποθηκευμένα δεδομένα της συγκεκριμένης περιοχής.

```
Parking.js
client > src > components > Parking.js > Parking
85    if (randomSpots < 400) {
86      newParkingData = parkingData.map((parking) => {
87        if (parking.id === parkingId) {
88          const updatedParking = {
89            ...parking,
90            spots: randomSpots,
91            icon: <ExclamationCircleTwoTone twoToneColor="#ff0000" />,
92          };
93
94          return updatedParking;
95        }
96
97        return parking;
98      });
99    } else if (randomSpots > 800) {
100     newParkingData = parkingData.map((parking) => {
101       if (parking.id === parkingId) {
102         const updatedParking = {
103           ...parking,
104           spots: randomSpots,
105           icon: <CheckCircleTwoTone twoToneColor="#52c41a" />,
106         };
107
108         return updatedParking;
109       }
110
111       return parking;
112     });
113   } else {
114     newParkingData = parkingData.map((parking) => {
115       if (parking.id === parkingId) {
116         const updatedParking = {
117           ...parking,
118           spots: randomSpots,
119           icon: <ClockCircleTwoTone twoToneColor="#ff8000" />,
120         };
121
```

Κώδικας 4.43: Έλεγχος των ελεύθερων θέσεων στάθμευσης, προβολή αυτού του αριθμού στον πίνακα όπως και του αντίστοιχου συμβόλου

```
<tbody>
  <tr>
    <td style={{ padding: "5px" }}>
      <Button
        type="primary"
        size={"small"}
        onClick={changeParkingData}
        style={{
          position: "absolute",
          right: -25,
          top: 29,
        }}
      >
        <ReloadOutlined />
      </Button>
    </td>
```

Κώδικας 4.44: Μορφοποίηση του κουμπιού ανανέωσης των δεδομένων του πίνακα των θέσεων στάθμευσης

Τέλος, όταν πατηθεί το κουμπί που βρίσκεται στην αριστερή πλευρά του πίνακα, καλείται η μέθοδος changeParkingData() και προκαλείται η προαναφερόμενη ανανέωση των δεδομένων (Κώδικας 4.44).

5 Συμπεράσματα και μελλοντικές ιδέες

5.1 Συμπεράσματα

Όπως εξετάσαμε σε βάθος στα προηγούμενα κεφάλαια, οι ευφυείς πόλεις τα τελευταία χρόνια είναι ένα πολύ σημαντικό θέμα, λόγω των ανθρώπινών αναγκών για εξέλιξη και ανάπτυξη του περιβάλλοντος στο οποίο ζούμε.

Στην παρούσα εργασία αρχικά, έγινε λεπτομερής έρευνα για την αναζήτηση των χαρακτηριστικών των έξυπνων πόλεων και ύστερα από όλα όσα συλλέχθηκαν, επιλέχθηκαν αυτά που βάσει της έρευνας μας θεωρούνται τα πιο σημαντικά και κρίσιμα. Έπειτα, ερευνήθηκαν περισσότερο τα στοιχεία αυτών των χαρακτηριστικών, ώστε να εκμαιεύσουμε μόνο την σημαντική πληροφορία, χωρίς κομμάτια που μπορεί να αποπροσανατολίσουν τον διαχειριστή μιας τέτοιας πλατφόρμας και να εξάγει λανθασμένα ή όχι τόσα ακριβή συμπεράσματα, από αυτά που του παρουσιάζονται.

Ένα πολύ κρίσιμο και δύσκολο σημείο στην υλοποίηση αυτής της εργασίας, ήταν η εύρεση δεδομένων πραγματικού χρόνου. Ήταν δύσκολο, διότι τέτοιου τύπου ανοιχτά δεδομένα για την χώρα μας και πιο ειδικά για την πόλη της Θεσσαλονίκης, υπάρχουν ελάχιστα. Μετά από εκτεταμένη αναζήτηση, βρήκαμε και καταλήξαμε σε πηγές που μας παρέχουν τα δεδομένα μέσω API's που χρειαζόμαστε, τα οποία μορφοποιούνται και απεικονίζονται στην πλατφόρμα.

Στην συνέχεια, αφού αναζητήσαμε παρόμοιες με την δική μας, πλατφόρμες, οι οποίες προβάλλουν δεδομένά με σκοπό να βοηθήσουν στην διαχείριση και στην εξέλιξη μιας πόλης, σχεδιάσαμε την διεπαφή του δικού μας dashboard, όπως ιδανικά θα την θέλαμε. Κύριοι γνώμονες για την σχεδίαση, ήταν το φιλικό περιβάλλον, η πρακτικότητα και η ευκολία στην άμεση κατανόηση των προβαλλόμενων δεδομένων.

Τελευταίο βήμα, ήταν η καταγραφή όλων των πληροφοριών που είχαμε συλλέξει όπως και της διαδικασίας υλοποίησης βήμα προς βήμα.

Έχοντας ολοκληρώσει λοιπόν, την υλοποίηση της πλατφόρμας και μελετήσει σε βάθος, όλα τα θεωρητικά μέρη που απαιτούνται για την σχεδίαση της και αναφέρθηκαν στα προηγούμενα κεφάλαια, συμπεραίνουμε αρχικά για τα χαρακτηριστικά των έξυπνων πόλεων, πως τα επόμενα χρόνια θα παρατηρηθεί μεγάλη αύξηση αυτών παγκοσμίως, λόγω των πολλών και ποικίλων πλεονεκτημάτων που έχουν να προσφέρουν στην ανθρωπότητα και στο περιβάλλον το οποίο πλέον χρήζει άμεσης φροντίδας.

Επόμενο συμπέρασμα, είναι ότι για την καλύτερη δυνατή διαχείριση των έξυπνων πόλεων, θα χρειαστούμε τα κατάλληλα εργαλεία για την παρακολούθηση, διαχείριση και εξέλιξη αυτών. Τα εργαλεία αυτά, θα πρέπει να είναι σχεδιασμένα, ύστερα από αρκετή μελέτη, με σκοπό την στοχευμένη και άμεση πληροφόρηση. Επιπλέον, τα δεδομένα που χρησιμοποιούνται, πρέπει να είναι αξιόπιστα και πραγματικού χρόνου για την καλύτερη και συγχρονισμένη με την πραγματικότητα, ενημέρωση του διαχειριστή ή και των κατοίκων, όπου σε μελλοντική επέκταση των δυνατοτήτων της πλατφόρμας, θα δίνεται πρόσβαση σε όλα τα δεδομένα και σε αυτούς.

Τέλος, κατά την διάρκεια της μελέτης και της υλοποίησης κατανοήσαμε πλήρως πως η δημιουργία μιας τέτοιας πλατφόρμας, με καινούργιες για εμάς τεχνολογίες, είναι ένα δύσκολο εγχείρημα, διότι απαιτεί υπομονή, μεράκι και επιθυμία για εξέλιξη. Οι έξυπνες πόλεις όμως, είναι το μέλλον της κοινωνίας που θα ζούμε και αξίζει να μελετήσουμε για να εξελίξουμε αυτό το όραμα, ώστε να προσφέρουμε ένα καλύτερο περιβάλλον σε εμάς και τα παιδιά μας.

5.2 Μελλοντικές ιδέες

Κάποιες ιδέες για μελλοντικές επεκτάσεις των δυνατοτήτων της πλατφόρμας μας, αφορούν μερικά χαρακτηριστικά που για λόγους δυσκολίας εύρεσης ανοιχτών δεδομένων για την πόλης μας ή πίεσης χρόνου δεν καταφέραμε να τα συμπεριλάβουμε, αλλά θα βοηθούσαν στην καλύτερη διαχείριση, εφόσον θα πρόσθεταν αρκετή πληροφορία και θα έκαναν την πόλη μας ακόμα πιο έξυπνη.

Αυτά τα χαρακτηριστικά είναι η ευφυής συλλογή των απορριμμάτων και η παρακολούθηση των tweets των κατοίκων για την πόλη μας στο twitter.

Η έξυπνη συλλογή απορριμμάτων ουσιαστικά είναι μια διαδικασία, η οποία αποτελείται από κάδους απορριμμάτων με συστήματα που υποδεικνύουν την τοποθεσία με συντεταγμένες, όπως και το βάρος του καθενός [17]. Έτσι τα απορριμματοφόρα, μπορούν να δημιουργούν καθημερινά καινούργιες διαδρομές και να επιλέγουν τις πιο σύντομες για να συλλέξουν τα απορρίμματα. Με αυτό τον τρόπο, μειώνεται η εκπομπή καυσαερίων στον αέρα και εξοικονομούνται χρήματα, λόγω της λιγότερης χρήσης πετρελαίου ή βενζίνης από τα απορριμματοφόρα.

Ένα ακόμα χρήσιμο χαρακτηριστικό, είναι η ενσωμάτωση του Twitter. Το Twitter είναι ένα κοινωνικό δίκτυο (social network), το οποίο δημιουργήθηκε και δημοσιεύτηκε το 2006 από το Τζακ Ντόρσεϊ. Η χρήση του είναι κυρίως τα λεγόμενα tweets, τα οποία είναι σύντομα μηνύματα που δεν υπερβαίνουν τους 280 χαρακτήρες και είναι ουσιαστικά ένας τρόπος επικοινωνίας μεταξύ των χρηστών. Για να γίνει περισσότερο κατανοητή η "δύναμη που κρύβουν" τα tweets, αρκεί να γίνει αναφορά στα στατιστικά τους. Σε ένα λεπτό έχουν δημιουργηθεί περίπου 350,000 tweets, 500 εκατομμύρια την ώρα και 200 δισεκατομμύρια τον χρόνο (2020) [95].

Βάση αυτών των στατιστικών, είναι προφανές πως η απεικόνιση των tweets στην πλατφόρμα θα είναι ένας εύκολος και λυσιτελής τρόπος για τους πολίτες, ώστε να επικοινωνούν τις σκέψεις και τις ανησυχίες τους για την πόλη με τον διαχειριστή και αυτός με την σειρά του, να ενημερώνει τους αρμόδιους για να λάβουν τα κατάλληλα μέτρα, καθώς και να προβούν στις όποιες ενέργειες.

Εν κατακλείδι, οι μελλοντικές ιδέες αφορούν χαρακτηριστικά που θα είναι ωφέλιμα για την κοινωνία από οικονομικής άποψης αλλά θα προσφέρουν και ένα αίσθημα πιο εύκολης επικοινωνίας στους πολίτες με τους διαχειριστές.

6 Βιβλιογραφία

[1] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, Y. Portugali, "Smart cities of the future," *The European Physical Journal Special Topics*, vol. 214, pp.481-518, Dec 2012.

[2] THALES, "IoT technology: Why Thales?," THALES, 2021. [Online]. Available: www.thalesgroup.com/en/markets/digital-identity-and-security/iot/customer-cases/why-thales

[3] SHERPA, "How data is used in Smart Cities?," SHERPA, 2020. [Online]. Available: www.project-sherpa.eu/how-data-are-used-in-smart-cities/

[4] Θωμά Αικατερίνη, "Η έξυπνη πόλη, και η συνεισφορά της στα αστικά κέντρα μέσω των κλάστερ," presented at Τμήμα Μηχανικών Χωροταξίας, Πολεοδομίας & Περιφερειακής Ανάπτυξης, Θεσσαλία, Ελλάδα, 2018.

[5] Vladimir Zdraveski, Kostadin Mishev, Dimitar Trajanov, Ljupco Kocarev, "ISO-Standardized Smart City Platform Architecture and Dashboard," In Proc. IEEE Pervasive Computing, 2017, pp. 35.

[6] zarpanews.gr, "Το σχέδιο για να γίνουν τα Χανιά «έξυπνη πόλη»," zarpanews.gr, 2020. [Online]. Available:

www.zarpanews.gr/to-schedio-gia-na-ginoyn-ta-chania-exypni-poli/

[7] Emile Mardacany, "SMART CITIES CHARACTERISTICS: Importance of Built Environment Components," presented at Institution of Engineering and Technology IET Conference on Future Intelligent Cities, London, UK, 2014.

[8] Bismart, "What Exactly is a Smart City?," Bismart, 2019. [Online]. Available: blog.bismart.com/en/what-is-a-smart-city

[9] Mihaela Teliceanu, George Cristian Lazaroiu, Virgil Dumbrava, "Consumption Profile Optimization in Smart City Vision," presented at 10th International Symposiumm on ADVANCED TOPICS IN ELECTRICAL ENGINEERING, Bucharest, Romania, 2018.

[10] Michael R. Wade, Michel Peter Pfaffli, "What is a Smart City anyways?," Global center for digital business transformation, 2016. [Online]. Available: www.imd.org/research-knowledge/articles/what-is-a-smart-city-anyways/

[11] Smart Energy International, "Magnum and Future Lighting work together on smart wireless lighting," Smart Energy International, 2016. [Online]. Available: www.smart-energy.com/regional-news/north-america/future-lighting-solutions-to-market-magnums-smart-lighting-products-worldwide/

[12] Mi Team, "Smart Parking: Ξεκίνα το «Έξυπνο πάρκινγκ» με 20.000 θέσεις πανελλαδικά και τεχνολογία Geo-Location," Xiaomi-Miui Hellas, 2019. [Online]. Available: news.xiaomi-miui.gr/smart-parkingk-me-20-000-theseis-panelladika-kai-texnologia-geo-location/

[13] Fraunhofer IGD, "Smart Living," Fraunhofer IGD, 2021. [Online]. Available: www.igd.fraunhofer.de/en/competences/technologies/smart-living

[14] Md Nazirul Isalm Sarker, Min Wu, Md Altab Hossin, "Smart governance through bigdata: Digital transformation of public agencies," presented at International Conference on Artificial Intelligence and Big Data, 2018, pp. 67.

[15] Δήμος Θεσσαλονίκης, "Ευφυές Περιβάλλον," Δήμος Θεσσαλονίκης, 2021. [Online]. Available: opengov.thessaloniki.gr/smart-city/smart-pillars/smart-environment#

[16] Eukalypron, "Why Smart Cities Need a People-First Mentality," Eukalypton, 2019. [Online]. Available: eukalypton.com/fr/2019/05/23/why-smart-cities-need-a-people-first-mentality/

[17] Insurance Daily News, "Κι όμως, η Ελλάδα αριθμεί 5 «έξυπνες πόλεις»," Insurance Daily News, 2019. [Online]. Available: www.insurancedaily.gr/ki-omos-i-ellada-arithmei-5-exypnes-pol/

[18] Paidis.com, "Στη Λάρισα για την «Έξυπνη πόλη» υψηλόβαθμα στελέχη της CISCO," Paidis.com, 2019. [Online]. Available: paidis.com/2019/05/02/στη-λαρισα-για-την-εξυπνη-πολη-υψηλο/

[19] Leonidas Anthopoulos, *Smart City Emergence.* Amsterdam: Elsevier B.V, 2019.

[20] Δήμος Ιωαννιτών, "IOANNINA SMART CITY," Δήμος Ιωαννιτών, 2018. [Online]. Available: www.diavouleusi.eu/diabouleyseis/ioannnina-smart-city-%CE%B5%CF%80%CE%B9%CF%87%CE%B5%CE%B9%CF%81%CE%B7%CF%83%CE%B9%CE%B1%CE%BA%CF%8C-%CF%83%CF%87%CE%AD%CE%B4%CE%B9%CE%BF-%CF%84%CE%BF%CF%85-%CE%B4%CE%AE%CE%BC%CE%BF%CF%85-%CE%B9/

[21] Kerkida Sport, "Βέροια: Νέο κλειστό γυμναστήριο!," Kerkida Sport, 2021. [Online]. Available: www.kerkidasport.gr/ektos-athlitismoy/20454-veroia-neo-kleisto-gymnastirio

[22] Heraklion Smart City, "Προσφερόμενες ψηφιακές υπηρεσίες του Δήμου Ηρακλείου προς τον επισκέπτη.," Heraklion Smart City, 2021. [Online]. Available: smartcity.heraklion.gr/el/project/prosferomenes-psifiakes-ypiresies-tou-dimou-irakliou-pros-ton-episkepti/

[23] IMD, "IMD Smart City Index 2020," IMD, 2020. [Online]. Available: www.imd.org/smart-city-observatory/smart-city-index/

[24] Laura Mullan, "Top 10 smart cities in the world," Technology magazine, 2019. [Online]. Available: technologymagazine.com/data-and-data-analytics/top-10-smart-cities-world

[25] Martin Whybrow, "How to be a truly smart city: Five lessons from Amsterdam," Smarter Communities.media, 2017. [Online]. Available: smartercommunities.media/truly-smart-city-five-lessons-amsterdam/

[26] Sue Weekes, "Seoul's smart city platform based on 'citizens as mayors philosophy," Smart Cities World, 2020. [Online]. Available: www.smartcitiesworld.net/news/news/seouls-smart-city-platform-based-on-citizens-as-mayors-philosophy-4912

[27] Sabato Angieri, "Smart Cities: Reykjavik," Eni.com, 2021. [Online]. Available: www.eni.com/en-IT/low-carbon/smart-cities-reykjavik.html

[28] Edward Perchard, "Make London the world's leading 'Smart City', says mayor Sadiq Khan," resource, 2017. [Online]. Available: resource.co/article/make-london-world-s-leading-smart-city-says-mayor-sadiq-khan-11920

[29] Michael Tobias, "Save energy and water with a smart building design, while improving comfort for occupants.," Nearby Engineers, 2021. [Online]. Available: www.ny-engineers.com/blog/how-new-york-is-becoming-a-smart-city

[30] Ben Payne, Lee Oon Ling and Alex Gorod, “Towards a governance dashboard for smart cities initiatives: a system of systems approach,” presented at 15th IEEE International Conference of System of Systems Engineering, Budapest, Hungary, 2020.

[31] Pierfancesco Bellini, Paolo Nesi, Michela Paolucci and Imad Zaza, “Smart City architecture for data ingestion and analytics: processes and solutions,” In Proc. IEEE Fourth International Conference on Big Data Computing Service and Applications, 2018, pp. 137.

[32] RenatoToasa, Marisa Maximiano, Catarina Reis, David Guevara, “Data visualization techniques for real-time information - A custom and dynamic dashboard for analyzing surveys’ results,” In Proc. IEEE 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018, pp. 2.

[33] Qi Han, Paolo Nesi, Gianni Panteleo, Irene Paoli, “Smart City Dashboards: Design, Development, and Evaluation,” In Proc. IEEE International Conference on Human-Machine Systems (ICHMS), 2020, pp. 1-2.

[34] Djamilja Oud, “Revealing the city’s rhythms in real time,” Geodan, 2021. [Online]. Available:

www.geodan.com/knowledge-and-innovation/managing-urban-processes-intelligently-with-the-amsterdam-smart-city-dashboard/

[35] Trikalacity, “Smart Trikala old,” Trikalacity, 2021. [Online]. Available:

trikalacity.gr/smart-trikala-old/

[36] Heraklion Smart City, “Home,” Heraklion Smart City, 2021. [Online]. Available:

smartcity.heraklion.gr/el/home/

[37] Km4City, “Sentient City Dashboards,” Km4City, 2021. [Online]. Available:

www.km4city.org

[38] Jeff Fennell, “Weather: Definition & Types,” Bio Differences, 2020. [Online]. Available:

study.com/academy/lesson/weather-definition-types-quiz.html

[39] Rachna C, “Difference Between Weather and Climate,” Bio Differences, 2019. [Online]. Available:

biodifferences.com/difference-between-weather-and-climate.html

[40] University Corporation For Atmospheric Research, “What is Weather?,” University Corporation For Atmospheric Research, 2021. [Online]. Available:

scied.ucar.edu/learning-zone/how-weather-works/weather

[41] Met Office, “Types of weather,” Met Office, 2021. [Online]. Available:

www.metoffice.gov.uk/weather/learn-about/weather/types-of-weather

[42] Thomas Bush, “6 Best Free and Paid Weather APIs,” NORDIC APIS, 2019. [Online]. Available:

nordicapis.com/6-best-free-and-paid-weather-apis/

[43] SEHSUCHT, “Vivaldi’s For Seasons - Composed with Climate Data,” vimeo, 2020. [Online]. Available:

vimeo.com/421003421

[44] healthline, “Hot and Cold: Extreme Temperature,” healthline, 2018. [Online]. Available:

www.healthline.com/health/extreme-temperature-safety

[45] Παναγιώτης-Μάριος Στασινός, “Μελέτη τεχνολογιών του Διαδικτύου των Πραγμάτων για παρακολούθηση και μέτρηση της ποιότητας ατμοσφαιρικών συνθηκών,” πτυχιακή εργασία, Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων, Θεσσαλονίκη, Ελλάδα, 2020.

[46] Wan-Li Cheng, Yu-Song Chen, Junfeng Zhang, T.J. Lyons, Joy-Lynn Pai and Shiang-Hung Chang, *Science of the Total Environment*. Amsterdam: Elsevier B.V, 2007.

[47] Ask about Ireland, “Why use Maps?,” Ask about Ireland, 2008. [Online]. Available:

www.askaboutireland.ie/learning-zone/primary-students/5th-+-6th-class/geography/maps-and-map-reading/what-are-maps/why-use-maps/

[48] Kim Rutledge, Hilary Costa, Erin Sprout, Santani Teng, Melissa McDaniel, Diane Boudreau, Tara Ramroop, Jeff Hunt, Hilary Hall, “MAP,” NATIONAL GEOGRAPHIC, 2011. [Online]. Available:

www.nationalgeographic.org/encyclopedia/map/12th-grade/

[49] Mariana Dorbecker, “10 Reasons Why Maps Are Important,” MAPSHOP, 2019. [Online]. Available:

www.mapshop.com/10-reasons-why-maps-are-important/

[50] Wiki Didactic, “What types of maps are there?,” Wiki Didactic, 2021. [Online]. Available:

edukalife.blogspot.com/2021/03/what-types-of-maps-are-there.html

[51] Κατερίνα Ματέρη, “Google Maps: Νέα εργαλεία προβλέπουν συνωστισμό σε τρένα, λεωφορεία, μετρό,” BestNews, 2021. [Online]. Available:

www.bestnews.gr/google-maps-nea-ergaleia-provlepoyn-synostismo-se-trena-leoforeia-metro/

[52] Περιφέρεια Κεντρικής Μακεδονίας, “Κατηγορίες Πυρκαγιών,” Περιφέρεια Κεντρικής Μακεδονίας, 2012. [Online]. Available:

www.pkm.gov.gr/default.aspx?lang=el-GR&page=104

[53] MARLOWE FIRE AND SECURITY, “Fire extinguisher types,” MARLOWE FIRE AND SECURITY, 2021. [Online]. Available:

marlowefireandsecurity.com/fire-extinguisher-type

[54] ATHENS VOICE, “Meteo: Οι μεγάλες πυρκαγιές του 2021 και οι καμένες εκτάσεις,” ATHENS VOICE, 2021. [Online]. Available:

www.athensvoice.gr/greece/726083_meteo-oi-megales-pyrkagies-toy-2021-kai-oi-kamenes-ektaseis

[55] Nalini Sonawane, Vaishali Jadhav, “Internet of things approach for smart city traffic control, parking management, and vehicle theft control,” presented at Second IEEE International Conference on Smart Systems and Inventive Technology, 2019, pp. 905-906.

[56] Thomas Hohenacker, “Building a smart city: The emergence of clever parking managment,” asmag.com, 2020. [Online]. Available:

www.asmag.com/showpost/27572.aspx

[57] Santiago Mino, “8 Reasons Why PHP Is StillSo Important for Web Development,” JOBSITY, 2021. [Online]. Available:

www.jobsity.com/blog/8-reasons-why-php-is-still-so-important-for-web-development

[58] FutureLearn, “What is Python used for? 10 Practical Python uses,” FutureLearn, 2021. [Online]. Available:

www.futurelearn.com/info/blog/what-is-python-used-for

[59] Joshua Weinstein, “What is Python used for? The Many Uses of Python Programming,” Career Karma, 2020. [Online]. Available:

careerkarma.com/blog/what-python-is-used-for/

[60] Brain Station, “Who Uses Python Today?,” Brain Station, 2021. [Online]. Available:

brainstation.io/career-guides/who-uses-python-today

[61] Java, “What is Java technology and why do i need it?,” Java, 2021. [Online]. Available:

www.java.com/en/download/help/whatis_java.html

[62] Diana Gray, “Why does Java remain so popular?,” Oracle University Blog, 2019. [Online]. Available:

blogs.oracle.com/oracleuniversity/post/why-does-java-remain-so-popular

[63] Swatee Chand, “Top 10 Applications of Java Programming Language,” edureka!, 2019. [Online]. Available:

medium.com/edureka/applications-of-java-11e64f9588b0

[64] stackshare, “Java,” stackshare, 2019. [Online]. Available:

stackshare.io/java

[65] Jay, “why Angular is so Popular in Modern Application Development,” Andola Soft Blog, 2021. [Online]. Available:

http://blog.andolasoft.com/2019/02/why-angular-is-so-popular-in-modern-application-development.html

[66] Hiren Patel, “All you need to know about AngularJS for Creative Web App Development,” tristatetechnology, 2018. [Online]. Available:

www.tristatetechnology.com/blog/angularjs/

[67] Angular, “The modern web developer’s platform,” Angular, 2010. [Online]. Available:

angular.io

[68] DATAQUEST, “SQL vs MySQL: A Simple Guide to the Differences,” DATAQUEST, 2021. [Online]. Available:

www.dataquest.io/blog/sql-vs-mysql/

[69] Mark Smallcombe, “PostgreSQL vs MySQL: The Critical Differences,” Xplenty, 2020. [Online]. Available:

www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/

[70] Priya Pedamkar, “Is MySQL Programming Language?,” educba, 2020. [Online]. Available:

www.educba.com/is-mysql-programming-language/

[71] stackshare, “MySQL,” stackshare, 2020. [Online]. Available:

stackshare.io/mysql

[72] Priya Pedamkar, “JavaScript vs Node JS,” EDUCBA, 2020. [Online]. Available:

www.educba.com/javascript-vs-node-js/

[73] Lalithnarayan C, “Node.js - Fronted or Backend,” Section, 2020. [Online]. Available:

www.section.io/engineering-education/nodejs-frontend-backend/

[74] tutorialspoint, “Node.js - Introduction,” tutorialspoint, 2010. [Online]. Available:

www.tutorialspoint.com/nodejs/index.htm

[75] Airbrake, “JavaScript Framework: 5 Front and Back-End Options,” Airbrake, 2016. [Online]. Available:

airbrake.io/blog/javascript/javascript-frameworks-love

[76] Besant Technologies, “What is Express.js?,” Besant Technologies, 2020. [Online]. Available:

www.besanttechnologies.com/what-is-expressjs

[77] Raushan Jha, “An introduction to React Programming Language,” GreyCampus, 2019. [Online]. Available:

www.greycampus.com/blog/programming/introduction-to-react-programming-language

[78] Nitin Pandit, “What And Why React.js,” C-Sharp Corner, 2021. [Online]. Available:

www.c-sharpcorner.com/article/what-and-why-reactjs/

[79] React, “React,” React, 2021. [Online]. Available:

reactjs.org

[80] Tim Gray, “What MongoDB is and what it does well,” OPTIMAL, 2018. [Online]. Available:

optimalbi.com/what-mongodb-is-and-what-it-does-well/

[81] Ironhack, “What is MongoDB? A practical guide to MongoDB and How to Install It on Catalina OS,” Ironhack, 2019. [Online]. Available:

www.ironhack.com/en/web-development/what-is-mongodb-practical-guide-how-to-use-it-and-install-it-in-os

[82] mongoDB, “MongoDB Cloud,” mongoDB, 2021. [Online]. Available:

www.mongodb.com/cloud

[83] Visual Studio Code, “Why VS Code?,” Visual Studio Code, 2021. [Online]. Available:

code.visualstudio.com/learn

[84] David Ramel, “Time Tracker Says VS Code Is No. 1 Editor for Devs, Some Working 15+ Hours Per Day,” Visual Studio Magazine, 2021. [Online]. Available:

visualstudiomagazine.com/articles/2021/01/13/wakatime-vs-code.aspx

[85] stackshare, “Visual Studio vs Visual Studio Code,” stackshare, 2021. [Online]. Available:

stackshare.io/stackups/visual-studio-vs-visual-studio-code

[86] Postman, “What is Postman?,” Postman, 2021. [Online].) Available:

www.postman.com

[87] testeigtechnologies, “What is postman and how to use postman to test API?, ” testeigtechnologies, 2020. [Online]. Available:

www.testrigtechnologies.com/what-is-postman-and-how-to-use-postman-to-test-api/

[88] Diego Gavilanes, “What is BlazeMeter, What is Postman, and How to Get Started with API Testing,” BlazeMeter, 2021. [Online].:

www.blazemeter.com/blog/what-is-postman-and-how-to-use-it-for-api-testing

[89] digitalcrafts, “What is Postman, and Why Should I Use It?,” digitalcrafts, 2020. [Online]. Available:

www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it

[90] PCMag.com, “Google Maps,” PCMag.com, 2021. [Online]. Available:

www.pcmag.com/encyclopedia/term/google-maps

[91] Rachael Njeri, “How to Integrate the Google Maps API into React Applications,” DigitalOcean, 2019. [Online]. Available:

www.digitalocean.com/community/tutorials/how-to-integrate-the-google-maps-api-into-react-applications

[92] GreeksforGreeks, “ReactJS | Router,” GreeksforGreeks, 2020. [Online]. Available:

www.geeksforgeeks.org/reactjs-router/

[93] MDN Web Docs, “Making asynchronous programming easier with async and await,” MDN Web Docs, 2021. [Online]. Available:

developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await

[94] React, “Hooks at a Glance,” React, 2021. [Online]. Available:

reactjs.org/docs/hooks-overview.html

[95] David Sayce, “The Number of tweets per day in 2020,” David Sayce, 2021. [Online]. Available:

www.dsayce.com/social-media/tweets-day/