

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



-- Πτυχιακή Εργασία --

Έξυπνη Πόλη: Πλατφόρμα Διαχείρισης Προβλημάτων Ενός Δήμου

Νικόλαος Τρυπάκης
Επιβλέπων: Ιωάννης Καρύδης

November 2, 2025

Επιβλέπων

Ιωάννης Καρύδης, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Ιωάννης Καρύδης, Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο
Μάριος Αναγνώστου, Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο
Μάρκος Αυλωνίτης, Καθηγητής,
Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο

Δήλωση

Δηλώνω υπεύθυνα πως κατά την διάρκεια συγγραφής της πτυχιακής και των παραδοτέων της, οποιαδήποτε χρήση εργαλείων τεχνητής νοημοσύνης έγινε αποκλειστικά στο πλαίσιο της ορθής ακαδημαϊκής χρήσης για καλύτερη οργάνωση και διατύπωση, αλλά και για βοηθητικούς σκοπούς. Το τελικό κείμενο έχει γραφτεί πλήρως από εμένα και όλα τα τμήματα έχουν περάσει από δικό μου έλεγχο και επεξεργασία πριν ενσωματωθούν σε αυτό. Τέλος, δεν έχει γίνει λογοκλοπή από άλλο κείμενο και αναλαμβάνω την ευθύνη για την ακρίβεια, την ακεραιότητα και την πρωτοτυπία του κειμένου.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά όλους τους καθηγητές μου για τις γνώσεις που μου προσέφεραν σε όλη την διάρκεια των σπουδών μου. Επίσης, ένα μεγάλο ευχαριστώ πάει στους συμφοιτητές και στους φίλους μου για την βοήθεια και συμπαράσταση τους όλα αυτά τα χρόνια. Πάνω από όλα, ευχαριστώ την οικογένεια μου, η οποία συνεχώς με στήριζε και ήταν πάντα δίπλα μου. Τέλος, ευχαριστώ όλους τους συμμετέχοντες που συνεισέφεραν στην παρούσα εργασία και στην ολοκλήρωση της.

Περίληψη

Σε αυτή την πτυχιακή εργασία σχεδιάζεται και υλοποιείται μία πλατφόρμα αναφορών για έξυπνες πόλεις που ενώνει τους κατοίκους και τις υπηρεσίες του δήμου ώστε να διαχειριστούν καθημερινά αστικά προβλήματα πιο εύκολα και αποδοτικά. Σκοπός της είναι να αντικαταστήσει τις αργές και χειροκίνητες μεθόδους, όπως κλήσεων μέσω τηλεφώνου και φυσικών επισκέψεων, με μία απλούστερη ροή. Η συνολική ιδέα είναι ότι οι πολίτες θα παίρνουν μία φωτογραφία του προβλήματος και την υποβάλλουν στο σύστημα. Έπειτα, οι αρχές διαχειρίζονται τις αναφορές αυτές, οι οποίες εμφανίζονται σε έναν διαδραστικό χάρτη ώστε και οι δύο πλευρές να μπορούν να παρακολουθούν την πρόοδο τους. Η πλατφόρμα χρησιμοποιεί ένα backend φτιαγμένο με Node.js και Express, μία cloud βάση δεδομένων μέσω του MongoDB Atlas, το framework Ionic React για την εφαρμογή για τα κινητά και την βιβλιοθήκη Leaflet για την οπτικοποίηση του χάρτη στην ιστοσελίδα. Η εργασία καλύπτει τις θεωρητικές απαιτήσεις, την αρχιτεκτονική και την υλοποίηση ενός τέτοιου συστήματος. Αρχιτεκτονικά, η πλατφόρμα ακολουθεί ένα σχέδιο τριών επιπέδων, διαχωρίζοντας το κάθε τμήμα της ξεχωριστά και κρατώντας τα όλα συνδεδεμένα μέσω ενός δομημένου API για να βοηθήσει στην συντήρηση και την διευκόλυνση της ανάπτυξης. Το τελικό αποτέλεσμα αξιολογήθηκε από 27 συμμετέχοντες χρησιμοποιώντας ένα ερωτηματολόγιο απαντήσεων από ένα έως πέντε της κλίμακας Likert. Τα αποτελέσματα έδειξαν υψηλές βαθμολογίες και μία θετική συνολική εμπειρία, με τους ερωτώντες να δείχνουν πρόθεση πως θα συνέχιζαν να χρησιμοποιούν μία τέτοια πλατφόρμα και ότι αυτή καταφέρνει τον στόχο της να ενώσει τους πολίτες με τους δήμους των πόλεων.

Abstract

In this thesis, a smart city reporting platform is designed and implemented to connect the citizens with municipality services so they can manage everyday urban problems more easily and effectively. The purpose of it is to replace slow and manual methods, such as phone calls and visits in person, with a simpler approach. The idea is that residents take a photo of the issue and report it to the system. Afterwards, the authorities handle those reports, which are displayed in an interactive map so that both sides can track their progress. The platform uses a backend created with Node.js and Express, a cloud database made with MongoDB Atlas, the Ionic React framework for the mobile application and the Leaflet library for the map display on the website. This project covers the requirements, architecture and implementation of such a system. From a design perspective, the platform follows a three tier approach, separating each part and keeping them all connected through a structured API to help in maintaining and making the development easier. The final product was evaluated using a questionnaire with 27 participants, who provided answers on a Likert scale ranging from one to five. The results showed high scores and an overall positive experience, with the questioners showcasing an intention to continue using the app and that it manages to achieve its goal of uniting the citizens with the municipalities.

Περιεχόμενα

A	Εισαγωγή	1
A.1	ΓΕΝΙΚΟ ΠΛΑΙΣΙΟ	1
A.2	Το πρόβλημα	2
A.3	ΣΤΟΧΟΙ ΚΑΙ ΕΡΩΤΗΜΑΤΑ	4
A.4	ΔΟΜΗ ΕΡΓΑΣΙΑΣ	5
B	Θεωρητικό Τπόβαθρο	6
B.1	ΕΞΤΡΗΗ ΠΟΛΗ	6
B.1.1	Χαρακτηριστικά Έξυπνης Πόλης	7
B.1.2	Θετικά Έξυπνης Πόλης	7
B.1.3	Προκλήσεις και Περιορισμοί	8
B.1.4	Μέλλον των Έξυπνων Πόλεων	9
B.2	ΚΙΝΗΤΕΣ ΚΑΙ ΔΙΑΔΙΚΤΥΑΚΕΣ ΕΦΑΡΜΟΓΕΣ	9
B.2.1	Ο ρόλος των Κινητών Εφαρμογών	10
B.2.2	Εφαρμογές Ιστού ως Συμπληρωματικά Εργαλεία	10
B.2.3	Μελέτες Περιπτώσεων και Εφαρμογές	11
B.2.4	Ενσωμάτωση με IoT και άλλων τεχνολογιών	11
B.2.5	Οφέλη και Εμπόδια	12
B.3	GIS	13
B.3.1	Ο ρόλος των GIS στις Έξυπνες Πόλεις	13
B.3.2	Συμμετοχικό GIS	14
B.4	ΕΡΓΑΛΕΙΑ ΙΣΤΟΣΕΛΙΔΑΣ	14
B.4.1	Leaflet	15
B.5	ΤΕΧΝΟΛΟΓΙΕΣ ΑΝΑΠΤΥΞΗΣ ΣΕ ΚΙΝΗΤΑ	16

B.6 ΕΡΓΑΛΕΙΑ ΚΙΝΗΤΗΣ ΕΦΑΡΜΟΓΗΣ	16
B.6.1 Ionic Framework	17
B.6.1.1 React	17
B.6.1.2 Ionic React	18
B.6.1.3 Capacitor	18
B.6.2 Android και Android Studio	19
B.6.3 Συγχριτικές Επισκοπήσεις και Πρακτικές	19
B.7 ΣΧΕΤΙΚΑ ΕΡΓΑ	20
B.7.1 CitySolution	20
B.7.2 Εφαρμογή με χρήση Ionic	21
B.7.3 Δίδαγμα από τις μελέτες	21
C Ανάλυση και Σχεδιασμός	23
C.1 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΧΡΗΣΤΩΝ	23
C.2 ΕΠΛΟΓΗ ΤΕΧΝΟΛΟΓΙΩΝ	26
C.2.1 Αιτιολόγηση και Λεπτομέρειες	27
C.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΤΥΤΗΜΑΤΟΣ	29
C.3.1 Κινητή Εφαρμογή	30
C.3.2 Server	31
C.3.3 Βάση Δεδομένων	32
C.3.4 Ιστοσελίδα	33
C.3.5 Τελικό Αποτέλεσμα	33
C.4 ΔΙΑΓΡΑΜΜΑΤΑ ΧΡΗΣΗΣ ΚΑΙ ΡΟΗΣ	34
C.4.1 Διάγραμμα Περίπτωσης Χρήσης	34
C.4.2 Διαγράμματα Ροής	35
C.5 ΣΧΕΔΙΑΣΜΟΣ UI/UX	37
C.5.1 Σχεδίαση Διεπαφής	37
C.5.2 Εμπειρία Χρήστη και Ροή	40
C.6 ΣΤΡΑΤΗΓΙΚΕΣ ΑΝΑΠΤΥΞΗΣ	41
C.6.1 Agile	41
C.6.2 Waterfall	42

D Ανάπτυξη Εφαρμογής	44
D.1 ΕΠΙΣΚΟΠΗΣΗ ΥΛΟΠΟΙΗΣΗΣ	44
D.2 BACKEND	45
D.2.1 Αρχικοποίηση	45
D.2.2 Σύνδεση με την Βάση Δεδομένων	46
D.2.3 Στατικά αρχεία	47
D.2.4 Ρύθμιση Μεταμόρφωσης Αρχείων	47
D.2.5 Εγγραφή και Σύνδεση Χρηστών	47
D.2.6 Διαχείριση Αναφορών	49
D.3 ΙΣΤΟΣΕΛΙΔΑ	52
D.3.1 Αρχικοποίηση και Δημιουργία του Χάρτη	53
D.3.2 Ορισμός Εικονιδίων	54
D.3.3 Ανάκτηση Αναφορών και Τοποθέτηση Δεικτών	54
D.3.4 Σελίδα Admin	56
D.4 ΚΙΝΗΤΗ ΕΦΑΡΜΟΓΗ	62
D.4.1 Δομή της Εφαρμογής	62
D.4.2 Σελίδα Καλωσορίσματος	62
D.4.3 Αρχική Σελίδα	65
D.4.4 Σελίδα "Οι Καταχωρήσεις μου"	67
D.4.5 Σελίδα Καταχώρησης	71
D.5 ANDROID STUDIO ΚΑΙ ΔΟΚΙΜΕΣ	74
D.6 ΔΙΑΓΡΑΜΜΑ ΑΝΑΠΤΥΞΗΣ	75
E Πειραματισμός	77
E.1 ΣΚΟΠΟΣ ΠΕΙΡΑΜΑΤΙΣΜΟΥ	77
E.2 ΔΙΑΔΙΚΑΣΙΑ ΠΕΙΡΑΜΑΤΙΣΜΟΥ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ	78
E.2.1 Λειτουργική Επιβεβαίωση	78
E.2.2 Αξιολόγηση	79
E.3 ΣΥΖΗΤΗΣΗ ΤΩΝ ΕΤΡΗΜΑΤΩΝ	90
F Συμπεράσματα και Μελλοντική Εργασία	92
F.1 ΣΥΜΠΕΡΑΣΜΑΤΑ	92
F.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ	93

<i>Περιεχόμενα</i>	viii
Συντμήσεις	98
Γλωσσάρι Εντικών θρων	99

Κατάλογος Σχημάτων

C.1	Αρχιτεκτονική της Πλατφόρμας Αναφοράς Έξυπνων Πόλεων	30
C.2	Συλλογές users και reports της Βάσης Δεδομένων	32
C.3	Διάγραμμα Περίπτωσης Χρήσης	35
C.4	Διάγραμμα Ροής Εφαρμογής	36
C.5	Διάγραμμα Ροής Ιστοσελίδας	37
C.6	Σελίδες της κινητής εφαρμογής	38
C.7	Σελίδα Χάρτη της Ιστοσελίδας	39
C.8	Σελίδα Διαχειριστή της Ιστοσελίδας	39
C.9	Wireflow Διεπαφής Χρήστη Εφαρμογής	41
C.10	Στρατηγική Προσέγγισης Agile	42
C.11	Στρατηγική Προσέγγισης Waterfall	43
D.12	Διάγραμμα Ανάπτυξης Συστήματος	76
E.13	Η εφαρμογή ήταν εύκολη στην κατανόηση και πλοήγηση.	82
E.14	Η διαδικασία δημιουργίας και αποστολής αναφοράς ήταν απλή.	83
E.15	Ο σχεδιασμός και η αισθητική της εφαρμογής ήταν ελκυστικά.	84
E.16	Οι οδηγίες/μηνύματα της εφαρμογής ήταν σαφή και κατανοητά.	85
E.17	Η απεικόνιση των προβλημάτων στον χάρτη ήταν ευδιάκριτη και χρήσιμη.	86
E.18	Η συνολική εμπειρία χρήσης ήταν ευχάριστη και χωρίς δυσκολίες.	87
E.19	Θα συνέχιζα να χρησιμοποιώ την εφαρμογή αν ήταν διαθέσιμη.	88
E.20	Θα πρότεινα την εφαρμογή σε άλλους.	89
E.21	Η εφαρμογή βοηθάει στη βελτίωση της επικοινωνίας πολιτών και δήμου.	90

Κατάλογος Πινάκων

<i>C.1</i> Λειτουργικές Απαιτήσεις	25
<i>C.2</i> Ποιοτικές Απαιτήσεις	25
<i>C.3</i> Οι τεχνολογίες που επιλέχθηκαν	27
<i>E.1</i> Αποτελέσματα ελέγχων λειτουργικότητας της πλατφόρμας	79
<i>E.2</i> Εισαγωγικές Πληροφορίες Χρηστών	81
<i>E.3</i> Εξοικείωση με την τεχνολογία και προηγούμενη εμπειρία με παρόμοιες εφαρμογές	81

Κεφάλαιο Α

Εισαγωγή

A.1 Γενικό Πλαίσιο

Σε αυτή την εργασία παρουσιάζεται η διαδικασία ανάπτυξης μίας ψηφιακής πλατφόρμας όπου στοχεύει στην βελτίωση του τρόπου όπου οι δήμοι των πόλεων ανταποκρίνονται και διαχειρίζονται προβλήματα που έχουν να κάνουν με τις υποδομές τους όταν αναφέρονται από τους πολίτες. Πιο συγκεκριμένα, το έργο εστιάζει στην δημιουργία ενός πρακτικού και αποδοτικού εργαλείου επικοινωνίας μεταξύ των κατοίκων και των τοπικών αρχών μέσω μίας εφαρμογής για κινητά και μίας ιστοσελίδας διαχείρισης. Πριν την εξέταση των τεχνικών διαδικασιών, είναι σημαντική η κατανόηση του ευρύτερου πλαισίου του θέματος, σε ποιο βαθμό ένα τέτοιο σύστημα θεωρείται απαραίτητο καθώς και πόσο βοηθάει στην διαχείριση ενός δήμου και στην συμμετοχή των πολιτών σε αυτόν.

Σε πολλές πόλεις, ειδικά σε μικρότερες, οι τρόποι διαχείρισης όπου χρησιμοποιούνται συνήθως αποτελούνται από ξεπερασμένες ή μη λειτουργικές διαδικασίες. Οι ίδιοι οι κάτοικοι συνήθως αναφέρουν προβλήματα όπως σπασμένα φώτα δρόμου και λακκούβες είτε μέσω από φυσικών επισκέψεων ή τηλεφωνικών κλήσεων. Αυτές οι προσεγγίσεις όχι μόνο κάνουν δύσκολο για τις υπηρεσίες του δήμου να μπορέσουν να διατηρήσουν μία οργάνωση των ενεργών αναφορών αλλά επίσης καθυστερούν την επίλυση αυτών των προβλημάτων (Shama et al., 2024). Συνεπώς, η έλλειψη ενός ολοκληρωμένου συστήματος οδηγεί σε ανεπαρκή αποτελεσματικότητα και μειωμένη έως και μηδενική επικοινωνία. Από την μεριά των πολιτών, η διαδικασία αυτή είναι χρονοβόρα και δεν ανταποδίδει πραγματικά και ουσιώδη αποτελέσματα, κάτι το οποίο τους προκαλεί απογοήτευση και τους οδηγεί στην αδιαφορία της τελείως.

Την ίδια στιγμή όμως, στην σημερινή εποχή οι άνθρωποι έχουν συνηθίσει την ευκολία του ψηφιακού κόσμου στην καθημερινότητα της ζωής τους. Από την πληρωμή λογαριασμών μέχρι και την πρόσβαση σε υπηρεσίες της κυβέρνησης μέσω του διαδικτύου, έχει γίνει πλέον απαραίτητη η ύπαρξη ενός γρήγορου, εύκολου και φιλικού προς τον χρήστη συστηματος. Οι τοπικές αρχές χρειάζονται να προσαρμόσουν τις υπηρεσίες τους για να μπορέσουν να καλύψουν αυτές τις προσδοκίες και να προωθήσουν την συμμετοχή της κοινότητας τους παραπάνω. Παρ' όλα αυτά, η μετάβαση προς την ψηφιακή διαχείριση μέσω εργαλείων συχνά εμποδίζεται λόγω περιορισμένων πόρων και το επίπεδο τεχνικής εμπειρίας. Αυτό οδηγεί σε ένα ψηφιακό κενό μεταξύ τι μπορεί να προσφέρει η τεχνολογία και τι πραγματικά υλοποιούν οι δήμοι.

Μέσω αυτού του έργου, ο σκοπός είναι αυτό το κενό να καλυφθεί. Η ιδέα είναι η δημιουργία μίας πλατφόρμας όπου αξιοποιεί σύγχρονες τεχνολογίες τόσο του ιστού όσο και των κινητών για να γίνει η επικοινωνία πολίτη και πόλης ευκολότερη και γρηγορότερη. Όπως συνοψίζουν οι Aslam και Ullah (2020), οι έξυπνες πόλεις συνήθως υλοποιούν κοινή υποδομή δεδομένων που επιτρέπει τον συγχρονισμό των πληροφοριών σε πραγματικό χρόνο. Επομένως, με την εισαγωγή μίας απλής πλατφόρμας αναφορών, η οποία υποστηρίζεται από κοινή βάση δεδομένων, οι άνθρωποι και οι εργάτες του δήμου μπορούν να επωφεληθούν από την βελτιωμένη αποτελεσματικότητα. Από την μία πλευρά, οι κάτοικοι παίρνουν πρόσβαση σε ένα τρόπο ώστε να συνεισφέρουν στην συντήρηση της πόλης τους. Από την άλλη, οι τοπικές αρχές αποκτούν ένα καλά δομημένο εργαλείο για την διαχείριση των δεδομένων τους, το οποίο τους διευκολύνει στην δουλειά τους.

A.2 Το πρόβλημα

Μία από τις πιο σημαντικές προκλήσεις στις τοπικές διοικήσεις στην σημερινή εποχή είναι η διατήρηση της επικοινωνίας τους μαζί με τους πολίτες. Παρά την διάθεση τεράστιου αριθμού ψηφιακών εργαλείων σε πολλούς τομείς, αυτές συνήθως βασίζονται σε ξεπερασμένα συστήματα ή χειροκίνητες διαδικασίες που κάνουν δύσκολο την διαχείριση των αστικών προβλημάτων αποτελεσματικά. Οι κάτοικοι όπου παρατηρούν προβλήματα στις γειτονιές τους συχνά δεν έχουν κάποιο διαθέσιμο και ευθύ μέσο να τα αναφέρουν. Στις πιο πολλές περιπτώσεις, αυτοί πρέπει να επικοινωνήσουν με τον δήμο μέσω κινητού ή φυσικής παρουσίας, όπου και τα δύο είναι ανεπαρκή και χρονοβόρα. Ως αποτέλεσμα, πολλά προβλήματα παραμένουν άφτιαχτα για μεγάλη περίοδο χρόνου, ενώ οι αρχές ζορίζονται στο να κρατήσουν

έλεγχο και προτεραιότητα των διάφορων αναφορών όπου λαμβάνουν συνεχώς.

Η απουσία ενός τέτοιου βασικού μηχανισμού οδηγεί σε πολλαπλά θέματα. Κάποια δεδομένα μπορεί να χαθούν, η επικοινωνία γίνεται ασυνεπής και οι κάτοικοι δεν έχουν κάποιο τρόπο να παρακολουθούν την πρόοδο των αιτημάτων τους. Οι μικροί δήμοι συχνά δεν έχουν τους απαραίτητους πόρους ώστε να επενδύσουν σε λύσεις πολύπλοκων λογισμικών ή ειδικών συστημάτων, κάτι το οποίο τους αφήνει εξαρτημένους σε ακατάλληλες μεθόδους εργασίας. Αυτή η κατάσταση, όχι μόνο περιορίζει την αποδοτικότητα των υπηρεσιών τους αλλά και μειώνει την εμπιστοσύνη και την συμμετοχή του κόσμου. Οι κάτοικοι νιώθουν πως δεν είναι μέρος της διαδικασίας και οι αρχές χάνουν την ανατροφοδότηση που χρειάζονται για να βελτιωθούν.

Το ερευνητικό πρόβλημα που αντιμετωπίζεται μέσα σε αυτή την εργασία προκύπτει από αυτήν ακριβώς την αποσύνδεση μεταξύ των δύο. Η ανάγκη δηλαδή για μία ενωμένη, απλή και εύκολα προσβάσιμη πλατφόρμα όπου βασίζεται πάνω στην επικοινωνία του πολίτη με τον δήμο. Ένα τέτοιο σύστημα πρέπει να είναι φτιαγμένο για καθημερινούς χρήστες ενώ προσφέρει στους διαχειριστές εργαλεία για αποδοτική κατηγοριοποίηση και παρακολούθηση των καταχωρήσεων. Η πρόκληση οπότε δεν αφορά μόνο την κατασκευή μίας τεχνολογικής υποδομής αλλά και στην διασφάλιση ότι προωθεί την εμπλοκή των χρηστών. Επομένως, το πρόβλημα είναι τεχνικό και κοινωνικό και αφορά το πως θα χτιστεί ένα σύστημα όπου ενισχύει την επικοινωνία ενώ παραμένει εύκολο και ιδανικό για πόλεις μικρού μεγέθους.

Από μία ερευνητική πλευρά, εξετάζεται πως μπορούν να ενσωματωθούν διαθέσιμα και σύγχρονα εργαλεία όπως το Ionic React για την αντιμετώπιση τέτοιων πραγματικών δημοτικών προκλήσεων. Διερευνώνται οι αρχές σχεδιασμού όπου χρειάζονται για την δημιουργία επεκτάσιμων εφαρμογών με επίκεντρο τον χρήστη και βελτιώνουν την ροή πληροφοριών. Αυτή η έρευνα είναι σημαντική ώστε να δείξει πως η ψηφιακή πρόοδος δεν πρέπει μόνο να ωφελεί τις μεγάλες πόλεις αλλά να μπορεί και να βοηθήσει μικρές κοινότητες να αναπτυχθούν. Άρα, ο κύριος σκοπός είναι η επίδειξη πως ο προσεκτικός σχεδιασμός και η επιλογή κατάλληλων σύγχρονων τεχνολογιών μπορούν να μεταμορφώσουν μία μη αποδοτική διαδικασία σε ένα οργανωμένο και λειτουργικό σύστημα.

A.3 Στόχοι και Ερωτήματα

Ο κύριος στόχος είναι η σχεδίαση και η ανάπτυξη μίας λειτουργικής πλατφόρμας η οποία είναι ικανή να βοηθήσει τους δήμους να ανταποκρίνονται σε αστικά προβλήματα πιο αποτελεσματικά με την συνεργασία των κατοίκων. Η πλατφόρμα συνδυάζει μία κινητή εφαρμογή για τους πολίτες και μία διεπαφή ιστού για τους διαχειριστές, χρησιμοποιώντας μία κοινή βάση δεδομένων. Σκοπός είναι η παροχή στους πολίτες έναν εύκολο τρόπο να μπορούν να αναφέρουν θέματα στην πόλη τους και στις αρχές των δήμων ένα δομημένο τρόπο να λαμβάνουν και να ενεργούν πάνω σε αυτές τις πληροφορίες. Υλοποιώντας αυτό το σύστημα, αποδεικνύεται το πως μοντέρνες τεχνολογίες μπορούν να ενισχύσουν την αποδοτικότητα σε μία κοινότητα.

Από μία πιο γενική εικόνα, η εργασία επικεντρώνεται στην προώθηση της συμμετοχής και της συνεργασίας μέσω της τεχνολογίας. Με την παραχώρηση απλών ψηφιακών εργαλείων, είναι πιο πιθανή η αλληλεπίδραση στην αναφορά θέματων καθώς και βελτιώνεται η οργάνωση και ο χρόνος απόκρισης σε αυτά. Επομένως, οι στόχοι δεν παραμένουν μόνο τεχνικοί αλλά και κοινωνικοί.

Για την επίτευξη αυτών των στόχων, ακολουθείται μία ροή προσανατολισμένη στην ανάπτυξη. Πιο συγκεκριμένα:

- Η ανάλυση των βασικών προκλήσεων όπου αντιμετωπίζουν οι δήμοι.
- Η σχεδίαση μίας πλατφόρμας όπου συνδέει άμεσα τις δύο πλευρές του θέματος.
- Η υλοποίηση του συστήματος αυτού χρησιμοποιώντας μοντέρνες τεχνολογίες.
- Η δοκιμή όσον αφορά την χρηστικότητα και λειτουργικότητα, διασφαλίζοντας ότι μπορεί πραγματικά να χρησιμοποιηθεί.
- Η εκτίμηση του πως ένα τέτοιο σύστημα μπορεί να βοηθήσει σε σχέση με παραδοσιακούς μεθόδους.

Επιπρόσθετα, κάποιες ερωτήσεις όπου προκύπτουν με βάση αυτούς τους στόχους είναι:

- Ποιες αρχές σχεδιασμού και τεχνικά frameworks είναι πιο κατάλληλα για την δημιουργία μίας τέτοιας πλατφόρμας;
- Πως μπορεί το σύστημα να παραμείνει απλό και προσβάσιμο παρέχοντας παράλληλα βασικές λειτουργίες;

- Με ποιους τρόπους το αποτέλεσμα προωθεί την συμμετοχικότητα;

Απαντώντας αυτές τις ερωτήσεις, μπορεί να δημιουργηθεί ένα πρωτότυπο. Το αποτέλεσμα εκτιμάται να δείξει πως η στοχευμένη χρήση της τεχνολογίας μπορεί να απλοποιήσει την καθημερινότητα και να βελτιώσει την αξιοπιστία σε μία κοινωνία.

A.4 Δομή Εργασίας

Η εργασία έχει οργανωθεί σε έξι κεφάλαια, με το κάθε ένα να εστιάζει σε ένα διαφορετικό μέρος της προτεινόμενης ψηφιακής πλατφόρμας. Η δομή αυτή στοχεύει στην σταδιακή καθιδρήγηση κάθε βήματος που έγινε από την κατανόηση του θέματος και του προβλήματος, μέχρι και την τελική υλοποίηση και τα συμπεράσματα. Μετά το εισαγωγικό κομμάτι της, προχωράει στο θεωρητικό υπόβαθρο όπου εξερευνεί τις κύριες έννοιες και τεχνολογίες όπου είναι σχετικές και ταιριάζουν με το έργο. Αυτό το κεφάλαιο παρέχει την απαραίτητη βάση για την κατανόηση του σκεπτικού του συστήματος και της χρήσης του.

Μετά την θεωρητική συζήτηση, το θέμα αλλάζει στο πιο τεχνικό κομμάτι αρχίζοντας την ανάλυση και την φάση του σχεδιασμού. Σε αυτό το κομμάτι, αναγνωρίζονται οι απαραίτητες απαιτήσεις και χτίζεται μία λειτουργική αρχιτεκτονική. Ακόμα, εξετάζεται το πως δομήθηκε το σύστημα, οι σχέσεις μεταξύ των λειτουργιών του και οι αποφάσεις σχεδίασης όπου διαμορφώνουν την ανάπτυξη του.

Προχωρώντας, παρουσιάζεται η διαδικασία υλοποίησης, δίνοντας μία αναλυτική εξήγηση για το πως κάθε μέρος αναπτύχθηκε και συνδέθηκε ώστε να δημιουργηθεί ένα πλήρως λειτουργικό σύστημα. Ακόμα, περιγράφεται η λογική πίσω από κάθε κώδικα αναλύοντας πως δουλεύει και με ποιο τρόπο γίνεται η μεταφορά των δεδομένων μεταξύ της εφαρμογής στην βάση δεδομένων και την ιστοσελίδα. Γενικά, αυτό το μέρος στοχεύει να δώσει μία τεχνική εικόνα του συστήματος ως ένα σύνολο και να εξηγήσει πως αυτό εκτελείται.

Το τελευταίο κομμάτι επικεντρώνεται πάνω στην αξιολόγηση του τελικού αποτελέσματος και αντανακλά το πόσο κοντά αυτό φτάνει και εκπληρώνει την εκτίμηση και τους στόχους όπου τέθηκαν από την αρχή. Τέλος, αναφέρονται τα συμπεράσματα της όλης διαδικασίας καθώς και προτείνονται πιθανές βελτιώσεις για περαιτέρω ανάπτυξη για μία ακόμη πιο ολοκληρωμένη λύση πάνω στο θέμα.

Κεφάλαιο Β

Θεωρητικό Τπόβαθρο

B.1 Έξυπνη Πόλη

Όρος “έξυπνη πόλη” έχει γίνει αρκετά διάσημος τα τελευταία χρόνια. Πολλές πόλεις γύρω στον κόσμο προσπαθούν να χρησιμοποιήσουν στρατηγικές και τεχνολογίες που θα κάνουν το περιβάλλον τους πιο αποδοτικό, φιλικό και βιώσιμο για τους πολίτες τους. Δεν υπάρχει όμως μια σταθερή έννοια για την έξυπνη πόλη. Υπάρχουν πολλές εκδοχές ως προς τι σημαίνει πραγματικά. Αυτό γίνεται επειδή, η ιδέα της έξυπνης πόλης συνδέεται με πολλές και διαφορετικές περιοχές όπως τις τεχνολογίες πληροφορικής και επικοινωνίας (ΤΠΕ), την οικονομική βελτίωση της πόλης και την διακυβέρνηση της ίδιας. Σύμφωνα με τους Macadar et al. (2016), η έννοια της άρχισε να γίνεται παγκοσμίως γνωστή κατά τις αρχές του 2000 και έχει από τότε ανεπτυχθεί πλέον να περιέχει νέα στοιχεία όπως την άμεση συμμετοχή των πολιτών της, υποδομές στο ψηφιακό κομμάτι και διάφορες υπηρεσίες χρησιμοποιώντας δεδομένα.

Σε γενικό κομμάτι, μια έξυπνη πόλη μπορεί να προσδιοριστεί ως μια πόλη που χρησιμοποιεί δεδομένα και τεχνολογίες ώστε να βελτιώσει την εμπειρία της ζωής των πολιτών της και να κάνει τις υπηρεσίες των δήμων πιο αποτελεσματικές. Επιπλέον, εξίσου σημαντικό, να καταφέρει να περιορίσει όσο το δυνατό περισσότερο τις περιβαλλοντικές επιπτώσεις της σημερινής κοινωνίας (Macadar et al., 2016). Άρα, ως ορισμός η έξυπνη πόλη δεν έχει να κάνει απλά με την εφαρμογή τεχνολογιών και την υλοποίηση κινητών εφαρμογών αλλά έχει να κάνει με την δημιουργία ενός σωστού οικοσυστήματος όπου συλλέγονται πληροφορίες εύκολα και μπορούν αυτές να χρησιμοποιηθούν για την επίλυση των διάφορων προβλημάτων

όπως η ρύπανση, η καλύτερη απόδοση της ενέργειας και διαχείριση διάφορων καταστροφών.

B.1.1 Χαρακτηριστικά Έξυπνης Πόλης

Οι έξυπνες πόλεις μεταξύ τους συνήθως μπορεί να ακολουθούν και να μοιράζονται τα ίδια χαρακτηριστικά. Συχνά, χρησιμοποιούν προηγμένα συστήματα ΤΠΕ ώστε να συλλέξουν και να επεξεργαστούν στοιχεία, τα οποία στην συνέχεια βοηθούν στην λήψη περαιτέρω αποφάσεων. Για παράδειγμα, ένας κυκλοφοριακός αισθητήρας μπορεί να παρατηρήσει συμφόρηση στους δρόμους και να προτείνει εναλλακτικές κατευθύνσεις προς τους οδηγούς σε πραγματικό χρόνο. Σύμφωνα με τους Arroub et al. (2016), αυτά τα συστήματα συνηθίζουν να βασίζονται πάνω σε τεχνολογίες του διαδικτύου των πραγμάτων (IoT) και αναλύσεις μεγάλων δεδομένων ώστε να μπορούν να λειτουργήσουν σωστά και αποδοτικά.

Ένα από τα πιο σημαντικά χαρακτηριστικά, είναι η συμμετοχή των πολιτών. Μια πόλη δεν μπορεί απλά να θεωρηθεί “έξυπνη” μόνο με βάση τις τεχνολογίες που έχει υλοποιήσει χωρίς να περιλαμβάνει τους ίδιους τους ανθρώπους της στην όλη διαδικασία. Οι πολίτες έχουν τον δικό τους ρόλο, με την χρήση εφαρμογών μπορούν να αναφέρουν πιθανά προβλήματα, να πάρουν πρόσβαση στις δημόσιες υπηρεσίες της πόλης και να δώσουν ανατροφοδότηση πίσω σε αυτή. Με αυτή την λογική, η δημιουργία διάφορων εφαρμογών είναι πολύ σημαντική καθώς ενώνουν τον δήμο με την κοινωνία της πόλης (Arroub et al., 2016).

Επιπρόσθετα, οι έξυπνες πόλεις έχουν ως βασικό στόχο την βιωσιμότητα. Όπως εξηγεί ο Anthopoulos (2015), η σχεδίαση μια τέτοιας πόλης είναι άμεσα συνδεδεμένη με την ανάπτυξη της περιβαλλοντικά. Άρα, σκοπός της είναι η βελτίωση κατανάλωσης ενέργειας και η μείωση εκπομπών αερίων στην ατμόσφαιρα. Αυτή η σύνδεση είναι πολύ σημαντική σε αυτό το κομμάτι επειδή οι κατοικημένες περιοχές καταναλώνουν τεράστιας ποσότητας ενέργεια και παράγουν πολλή ποσότητα απόβλητων.

B.1.2 Θετικά Έξυπνης Πόλης

Την παρόντα πολλά θετικά στην υλοποίηση έξυπνων πόλεων και ένα από τα πιο βασικά είναι πως βελτιώνουν την εμπειρία ζωής των πολιτών τους. Οι ίδιοι μπορούν να απολαύσουν τα διάφορα πλεονεκτήματα της όπως τα βελτιωμένα μέσα μεταφοράς και η ευκολότερη πρόσβαση σε υπηρεσίες της. Με βάση τους Arroub et al. (2016), αυτές οι βελτιώσεις κάνουν τις ίδιες τις πόλεις πιο φιλικές προς τους τουρίστες και τις εταιρείες κάτι το οποίο βοηθάει στην

ανάπτυξη της οικονομίας τους.

Ακόμα, η χρήση τεχνολογιών για την παρακολούθηση και την ελαχιστοποίηση κατανάλωσης, και την αποδοτική διαχείριση ρύπων οδηγεί σε σημαντικά οφέλη. Οι Angelidis και Drakouli (2019) δίνουν έμφαση πως μέσα στην ΕΕ, οι έξυπνες πόλεις θεωρούνται ως βασική στρατηγική για την καταπολέμηση της κλιματικής αλλαγής και για την επίτευξη των υπόλοιπων τους στόχων για το περιβάλλον. Πιο συγκεκριμένα, έξυπνοι μετρητές μπορούν να βοηθήσουν ένα καταναλωτή να επιβλέπει πόση ενέργεια ξοδεύει και να διαμορφώσουν καλύτερα την διανομή ληξιτρικής ενέργειας.

Τέλος, μέσω αναλύσεων και αυτοματοποιήσεων, οι αρχές του δήμου μπορούν να δουλέψουν πιο αποτελεσματικά, να πάρουν καλύτερες αποφάσεις και να αντιδρούν ταχύτερα σε επείγοντα θέματα. Ο Anthopoulos (2015) παρατηρεί πως αυτή η αποτελεσματικότητα προέρχεται από την ενσωμάτωση διαφορετικών συστημάτων και την κοινοποίηση πληροφοριών μεταξύ των τμημάτων του.

B.1.3 Προκλήσεις και Περιορισμοί

Οι έξυπνες πόλεις παρά τα πολλά πλεονεκτήματα τους, αντιμετωπίζουν πολλές προκλήσεις. Ένα από τα πιο σημαντικά μειονεκτήματα τους είναι το χόστος της υλοποίησης μιας τέτοιας ιδέας. Το να κτίζεις υποδομές και να αναπτύσσεις εφαρμογές λογισμικού χρειάζεται πολλές επενδύσεις. Αρκετές πόλεις και ειδικά αυτές που είναι σε χώρες που αναπτύσσονται ακόμα δεν μπορούν να φτάσουν αυτά τα ποσά στα έξοδα τους. Ο Konbr (2019) δίνει ένα τέτοιο παράδειγμα με την πόλη της Αιγύπτου, όπου τα πλάνα τους για μια τέτοια πόλη είναι ο περιορισμός κατανάλωσης ενέργειας και καλύτερη διαχείριση αποβλήτων. Ωστόσο, στο όρυζο του επίσης δείχνει πως αυτά τα πλάνα είναι αρκετά δύσκολα για να γίνουν πραγματικότητα λόγω οικονομικών προκλήσεων και οργάνωσης.

Άλλο ένα σημαντικό πρόβλημα είναι η ασφάλεια των δεδομένων. Αφού η συλλογή προσωπικών πληροφοριών παίζει τόσο σημαντικό ρόλο, υπάρχει πάντα το ρίσκο κυβερνοεπίθεσης και κακόβουλης χρήσης αυτών. Όπως υποστηρίζει ο Anthopoulos (2015), η ασφάλεια θα πρέπει να ενσωματωθεί από την αρχή μέσα στην αρχιτεκτονική των συστημάτων, με την χρήση κρυπτογράφησης για τα δεδομένα, ασφαλισμένων πρωτόκολλων επικοινωνίας και συνεχή έλεγχου.

Τπάρχει ακόμα ο περιορισμός ως προς την τεχνολογία. Αρκετοί πολίτες μπορεί να μην έχουν το ίδιο επίπεδο πρόσβασης σε τεχνολογία κάτι το οποίο δημιουργεί ανισότητα. Οι Angelidis και Drakouli (2019) τονίζουν πως οι στρατηγικές τέτοιων περιοχών θα πρέπει να αποφύγουν τον διαχωρισμό των πολιτών σε διαφορετικές ψηφιακές ομάδες.

Τέλος, η ρύθμιση μιας έξυπνης πόλης μπορεί να είναι περίπλοκη. Για την εκτέλεση τέτοιων λύσεων συχνά είναι υποχρεωτική η συνεργασία μεταξύ πολλαπλών πλευρών όπως κυβερνήσεων, εταιρειών και πολιτών αντίστοιχα. Χωρίς την σωστή διαχείριση και ξεκάθαρων υποχρεώσεων, σχέδια σαν αυτά ενδέχονται να αποτύχουν και να εμφανίσουν συγκρούσεις (Anthopoulos, 2015).

B.1.4 Μέλλον των Έξυπνων Πόλεων

Κοιτάζοντας στο μέλλον, οι έξυπνες πόλεις αναμένονται με την συνεχή ανάπτυξη σε όλους τους τομείς να εμφανίζονται ακόμα πιο συχνά γύρω στον κόσμο. Με την χρήση τεχνητής νοημοσύνης, ανανεωμένων δικτύων και αναλύσεων μεγάλων δεδομένων, οι πόλεις θα μπορούν να παρέχουν πιο προχωρημένες υπηρεσίες (Macadar et al., 2016). Παρόλα αυτά, η επιτυχία τους προχωρώντας μπροστά εξαρτάται σημαντικά από την αντιμετώπιση τωρινών προκλήσεων όπως η ασφάλεια και το κόστος. Αυτά τα θέματα θα πρέπει να επιλυθούν ώστε να μπορούν να θεωρηθούν δίκαιες αλλά και βιώσιμες για όλους, καθώς δεν είναι αρκετή μόνο η επικέντρωση στην τεχνολογία (Arroub et al., 2016).

B.2 Κινητές και Διαδικτυακές Εφαρμογές

Οι τωρινές έξυπνες πόλεις δεν μπορούν να λειτουργήσουν αποτελεσματικά χωρίς να υπάρχει επικοινωνία μεταξύ των πολιτών και του δήμου. Ενώ μέσα στην έννοια εμπλέκονται αρκετά οι ανεπτυγμένες υποδομές και τα IoT συστήματα, η πραγματική αξία έρχεται όταν όλα αυτά είναι διαθέσιμα με ευκολία στους ανθρώπους όπου ζουν εκεί. Οι κινητές και διαδικτυακές εφαρμογές αποτελούν τα απαραίτητα εργαλεία ώστε να γίνει κάτι σαν αυτό πραγματικότητα. Αυτές οι εφαρμογές επιτρέπουν τους κατοίκους να μπορούν να αλληλεπιδράσουν με τις τοπικές αρχές, να αναφέρουν διάφορα ζητήματα και το πιο σημαντικό ακόμα να έχουν παρόν στην λήψη αποφάσεων. Όπως εξηγούν οι Hou et al. (2020), η επιτυχία αυτών των αστικών κυβερνύσεων συχνά βασίζονται στο πόσο καλά οι “κινητές” λύσεις έχουν ενσωματωθεί στις υπηρεσίες τους.

Η όνοδος των τηλεφώνων και η διάδοση της πρόσβασης στο ίντερνετ έχουν δημιουργήσει ένα νέο περιβάλλον στο οποίο όλες οι υπηρεσίες μπορούν πλέον να γίνονται ψηφιακά. Οι άνθρωποι δεν χρειάζονται πλέον να έρθουν σε επαφή με δημοτικά γραφεία για κάθε αίτημα τους. Αντίθετα, μπορούν να χρησιμοποιήσουν εφαρμογές για την υποβολή παραπόνων και δημόσιων έργων καθώς και να ενημερώνονται σε πραγματικό χρόνο πάνω σε αυτά. Αυτή η αλλαγή σε ηλεκτρονικές λειτουργίες αποτελεί μια από τις βασικές πλευρές των έξυπνων πόλεων της σημερινής εποχής.

B.2.1 Ο ρόλος των Κινητών Εφαρμογών

Ένα από τα πιο συχνά εμφανιζόμενα στοιχεία των έξυπνων πόλεων, ως προς τις λύσεις για έναν μέσο κάτοικο, αποτελούν οι κινητές εφαρμογές. Σύμφωνα με τους Hou et al. (2020), αυτές οι εφαρμογές δεν είναι απλά άλλο ένα κανάλι επικοινωνίας αλλά επηρεάζουν ριζικά το πως δουλεύει η διακυβέρνηση. Για παράδειγμα, η χρήση τους πάνω στην διαχείριση της πόλης επιτρέπει τον πολίτη να αναφέρει λακκούβες και σπασμένα φώτα δρόμου, βγάζοντας απλά μόνο μια φωτογραφία και στέλνοντας την μέσω από αυτές. Αυτό δημιουργεί ένα όμεσο τρόπο ανατροφοδότησης μεταξύ της κοινωνίας και των αρχών μειώνοντας τους χρόνους απάντησης και βελτιώνοντας την υπευθυνότητα.

Οι Boulos et al. (2015) δίνουν έμφαση πως τέτοιες εφαρμογές παίζουν τεράστιο ρόλο στην καινοτομία και στην συμμετοχή της κοινότητας. Στον τομέα της υγείας, για παράδειγμα, μπορεί να τους δωθεί η δυνατότητα να παρακολουθούν και να εντοπίζουν πιθανές μεταβολές στα επίπεδα ρύπανσης. Παρομοίως, εφαρμογές που έχουν φτιαχτεί τελείως για συμμετοχή της κοινότητας μπορούν να αφήσουν τους πολίτες να ψηφίζουν σε τοπικά έργα και να δίνουν τις προσωπικές τους αντιδράσεις σε διάφορες αστικές προτάσεις. Αυτές οι λειτουργίες δίνουν στις κινητές εφαρμογές ένα ρόλο πιο σημαντικό από απλά μέσα για την διευκόλυνση, κάνοντας τα εργαλεία της συμμετοχικής διαχείρισης των πόλεων.

B.2.2 Εφαρμογές Ιστού ως Συμπληρωματικά Εργαλεία

Οι κινητές εφαρμογές είναι απαραίτητες για την εν κινήσει πρόσβαση, όμως στο κομμάτι της προσέγγισης οι διαδικτυακές εφαρμογές παραμένουν πιο σημαντικές. Συνήθως, οι πολίτες προτιμούν την χρήση διεπαφών στον υπολογιστή για πιο σημαντικές εργασίες όπου χρειάζονται πιο περίπλοκες ενέργειες. Με βάση τους Aslam και Ullah (2020), οι εφαρμογές ιστού είναι συχνά ενωμένες με την ίδια υποδομή backend με αυτή στις κινητές,

διασφαλίζοντας σταθερότητα και σύνδεση σε πραγματικό χρόνο μεταξύ τους. Άρα, ένα αίτημα από μια εφαρμογή κινητού μπορεί να εμφανίζεται στην διαδικτυακή σελίδα μιας πόλης, δίνοντας στους χρήστες πολλαπλές πλατφόρμες για να αλληλεπιδράσουν. Ακόμα, διαδικτυακοί πίνακες ελέγχου σε σελίδες μπορούν να βοηθήσουν τους διαχειριστές στην οπτικοποίηση των συλλεγμένων δεδομένων από τους κατοίκους. Αυτοί οι πίνακες, ενσωματώνονται σε συστήματα IoT και δείχνουν πληροφορίες από σένσορες, κάμερες και αναφορές σε πραγματικό χρόνο. Αυτός ο συνδυασμός των κινητών και διαδικτυακών πλατφορμών είναι καθοριστικός στο να κάνει τα οικοσυστήματα να δουλεύουν ομαλά.

B.2.3 Μελέτες Περιπτώσεων και Εφαρμογές

Στην Ελλάδα, οι ψηφιακές στρατηγικές των δήμων έχουν άρχισει να εισάγουν κινητές και διαδικτυακές πλατφόρμες ως πρωτοβουλίες για τις έξυπνες πόλεις. Οι Siokas et al. (2019) επισημαίνουν πως πολλές ελληνικές πόλεις τις έχουν υιοθετήσει ήδη για απλές υπηρεσίες όπως την ενημέρωση του πολίτη και την ηλεκτρονική διαχείριση της κυβέρνησης, παρόλο που είναι ακόμα υπό ανάπτυξη. Ωστόσο, η έρευνα τους σημειώνει και προκλήσεις όπου μικρότερες πόλεις δυσκολεύονται να υιοθετήσουν τέτοια συστήματα και υπάρχει έλλειψη τεχνικής υποστήριξης σε αυτές.

Στο εξωτερικό, οι Angelopoulos et al. (2019) μελέτησαν πραγματικές χρήσεις αυτών των εφαρμογών, δείχνοντας το πως οι ψηφιακές λύσεις είναι ενσωματωμένες στις δημοτικές υπηρεσίες. Η ανάλυση τους έδωσε σημασία στην εμπιστοσύνη των πολιτών και την ευκολία χρήσης ως κύριο κλειδί για την έγκριση τους. Παρομοίως, οι Hou et al. (2020) έκαναν μια μελέτη πάνω σε εφαρμογές διαχείρισης της πόλης και βρήκαν πως η αποδοχή από τους χρήστες εξαρτάται από την γρήγορη ενημέρωση και την αξιοπιστία. Οι πολίτες, επομένως, είναι πιο πρόθυμοι να χρησιμοποιήσουν αυτές τις πλατφόρμες όταν παρατηρούν συνεχή ανταπόκριση από τις αρχές. Αντίστοιχα, αν ζητήματα παραμένουν άλιτα χωρίς καθόλου ενημερώσεις, τότε η εμπιστοσύνη τους σε αυτά μειώνεται. Αυτό δείχνει πως οι δήμοι θα πρέπει να βελτιωθούν εσωτερικά ως προς την ροή εργασίας ώστε να μπορούν να εγγυηθούν γρήγορες υπηρεσίες στο κοινό τους.

B.2.4 Ενσωμάτωση με IoT και άλλων τεχνολογιών

Οι εφαρμογές αυτές συνήθως θεωρούνται ως διεπαφές χρηστών για πιο περίπλοκα IoT συστήματα. Σύμφωνα με τους Aslam και Ullah (2020), πολλές υπηρεσίες των πόλεων

βασίζονται πάνω σε δεδομένα όπου συλλέγονται από αισθητήρες που έχουν τοποθετηθεί γύρω σε αυτές. Κάποια είδη αυτών είναι οι σένσορες για την ποιότητα αέρα όπου μπορούν να μεταφέρουν πληροφορίες στους κατοίκους, ειδοποιώντας τους για τα επίπεδα μόλυνσης. Ακόμα, μπορούν να βοηθήσουν και σε έξυπνα συστήματα πάρκινγκ όπου δείχνουν τις διαθέσιμες θέσεις των χώρων αυτών σε πραγματικό χρόνο. Η προσθήκη αυτών σε συνεργασία με τις εφαρμογές κάνουν άμεσα διαθέσιμες σημαντικές πληροφορίες για τους ανθρώπους.

Το backend κομμάτι αυτών των υποδομών συχνά αποτελείται από υπηρεσίες στο cloud όπου αποθηκεύουν και επεξεργάζονται τα δεδομένα. Ακόμα, χρησιμοποιούνται APIs όπου συνδέουν τις εφαρμογές με τις βάσεις δεδομένων και εργαλεία ασφαλείας για την προστασία απορρήτων πληροφοριών. Οι Angelopoulos et al. (2019) δίνουν έμφαση στο πόσο απαραίτητη είναι εμπιστοσύνη σε αυτά τα οικοσυστήματα. Οι χρήστες θα μοιραστούν τα προσωπικά τους δεδομένα μόνο αν πιστέψουν πως αυτά θα προστατευτούν από κακόβουλες χρήσεις. Επομένως, είναι αναγκαίο να υπάρχει δυνατή χρυπτογράφηση και ξεκάθαρες πολιτικές προστασίας ως προς την χειραγώγηση των δεδομένων ώστε οι πλατφόρμες αυτές να θεωρηθούν επιτυχής.

B.2.5 Οφέλη και Εμπόδια

Η χρήση των εφαρμογών στις έξυπνες πόλεις εμφανίζει πολλά οφέλη. Οι κάτοικοι έχουν ευκολότερη πρόσβαση σε υπηρεσίες, μπορούν να αναφέρουν οποιαδήποτε προβλήματα στην στιγμή και να παραλάβουν νέα χωρίς την φυσική τους παρουσία. Αυτό βελτιώνει την σχέση των χρηστών με τις τοπικές αρχές εφόσον έχουν παραπάνω έλεγχο καθώς ενισχύεται η ευκολία χρήσης και η διαφάνεια μεταξύ τους (Hou et al., 2020). Επιπλέον, δημιουργούνται νέες ευκαιρίες για ενεργή συμμετοχή στην λήψη αποφάσεων βοηθώντας τους πολίτες να νιώθουν πως εμπλέκονται παραπάνω στις κοινότητές τους (Boulos et al., 2015). Για τους δήμους, η συνεχή ροή δεδομένων από τους ανθρώπους μαζί με τις IoT συσκεύες τους βοηθάει στην καλύτερη κρίση επιλογών και πιο αποτελεσματική χρήση πόρων.

Όμως, υπάρχουν πολλά εμπόδια για να εφαρμοστεί αυτό επιτυχημένα. Οι Siokas et al. (2019) αναφέρουν πως πολλοί δήμοι έρχονται αντιμέτωποι με οικονομικούς περιορισμούς και έλλειψη τεχνικής εμπειρίας, κάτι το οποίο επιβραδύνει στην ανάπτυξη των κινητών υπηρεσιών. Ακόμα, η ανησυχία για την προστασία δεδομένων και ασφάλειας μπορούν να περιορίσουν τον συνολικό αριθμό χρηστών, όπως σημειώνουν οι Angelopoulos et al. (2019). Άλλο ένα θέμα

είναι η ανισότητα στον ψηφιακό χώρο. Αυτά τα εμπόδια πρέπει να αντιμετωπιστούν μέσω της εκπαίδευσης των πολιτών και δυνατών στρατηγικών ώστε να διασφαλίσουν πως όλοι θα κερδίσουν από τις πρωτοβουλίες αυτές.

B.3 GIS

Τα γεωγραφικά συστήματα πληροφοριών (GIS) είναι απαραίτητα για την διαχείριση και ανάλυση δεδομένων στις έξυπνες πόλεις. Αυτά δίνουν την δυνατότητα στους δήμους και στους πολίτες να οπτικοποιήσουν σύνθετα σετ στοιχείων και να πάρουν αποφάσεις με βάση τις πληροφορίες βασισμένες στην τοποθεσία. Σε μια έξυπνη πόλη, ένα μεγάλο μέγεθος γεγονότων μπορεί να συνδεθεί άμεσα και με μια γεωγραφική θέση. Τα GIS βοηθάνε στην οργάνωση και στην επίδειξη αυτών με τρόπο εύκολο ως προς την κατανόηση και στην επεξεργασία τους (Zhao et al., 2025).

Τέτοια διαδικτυαχά εργαλεία έχουν άρχισει να γίνονται πιο σημαντικά επειδή αφήνουν οποιονδήποτε που έχει πρόσβαση στο ίντερνετ να μπορεί να βλέπει ελεύθερα αυτά τα χωρικά δεδομένα. Σε αντίθεση με τα παραδοσιακά GIS λογισμικά, όπου μπορεί να είναι ακριβά και χρειάζονται ειδικά μηχανήματα, τα τωρινά εργαλεία χρησιμοποιούν πλαίσια ανοικτού κώδικα για την προβολή χαρτών και δεδομένων σε απλούς ιστότοπους.

B.3.1 Ο ρόλος των GIS στις Έξυπνες Πόλεις

Τα γεωγραφικά συστήματα παίζουν ένα σημαντικό ρόλο στην ανάπτυξη μια έξυπνης πόλης. Οι Zhao et al. (2025) σημειώνουν πως οι αναλύσεις από αυτά τα εργαλεία παρέχουν γεωγραφικά πλαίσια όπου χρειάζονται για την αποτελεσματική διαχείριση πόρων. Για παράδειγμα, μπορούν να αναγνωρίσουν μοτίβα κατανάλωσης σε διαφορετικές γειτονιές και να εντοπίσουν κυκλοφοριακές συμφορήσεις. Αυτές οι ιδέες είναι απαραίτητες για βιωσιμότητα και λειτουργική αποτελεσματικότητα.

Σύμφωνα με τους Bovkirk και Aydinoglu (2021), αυτά τα συστήματα ενσωματώνουν δεδομένα πραγματικού χρόνου από αισθητήρες IoT. Με τον συνδυασμό αυτών, οι πλατφόρμες δίνουν την δυνατότητα στις πόλεις να αντιδρούν γρήγορα σε αλλαγές. Οπότε, αν σε μια περιοχή οι ελεγκτές αέρα παρατηρήσουν κάποια μεγάλη τιμή μόλυνσης, οι αρχές μπορούν να εκδώσουν προειδοποιήσεις άμεσα και να δράσουν αντίστοιχα. Αυτή η διαδικασία αποφάσεων σε ταχύ

χρόνο δεν θα ήταν δυνατή χωρίς την ανάλυση χωρικών πληροφοριών που προσφέρουν τα GIS.

Επιπρόσθετα, τα GIS μπορούν να υποστηρίζουν προγνωστικές λειτουργίες, μοντελοποιώντας μελλοντικά σενάρια με βάση παλαιών και τωρινών στοιχείων. Η συγκεκριμένη δυνατότητα είναι ιδιαίτερα χρήσιμη κατά την σχεδίαση όπου οι υπεύθυνοι μπορούν να προσομοιώσουν την επιρροή νέων αποφάσεων. Όλα αυτά κάνουν τα GIS μια βασική τεχνολογία στις στρατηγικές των έξυπνων πόλεων

B.3.2 Συμμετοχικό GIS

Μία σημαντική τάση στα γεωγραφικά συστήματα των έξυπνων πόλεων είναι η εισαγωγή συμμετοχικών προσεγγίσεων όπου οι κάτοικοι συνεισφέρουν δικά τους χωρικά δεδομένα μέσω την χρήση εφαρμογών. Οι Bakowska-Waldmann και Kaczmarek (2021) παρουσιάζουν μια αξιολόγηση των GIS με δημόσια συμμετοχή (PPGIS) και δίνουν έμφαση στον ρόλο τους στην στήριξη των τοπικών κοινοτήτων ώστε να συμμετάσχουν στην αστική σχεδίαση μέσω διαδικτυακών περιβαλλόντων χαρτογράφησης. Εξηγούν πως το PPGIS ενισχύει την επικοινωνία και την ενδυνάμωση των πολιτών δίνοντας την ευκαιρία σε αυτούς να εμπλακούν άμεσα στην συλλογή γεωγραφικών δεδομένων και στην παραγωγή αποφάσεων. Όταν συνεργάζονται με εργαλεία σαν το Leaflet, τα PPGIS μπορούν να διευκολύνουν την δημιουργία φιλικών προς τον χρήστη πλατφορμών που προωθούν την γενική συμμετοχή του κοινού σε σημαντικές πρωτοβουλίες των πόλεων.

B.4 Εργαλεία Ιστοσελίδας

Για την διαδικτυακή απεικόνιση των δεδομένων, η παρούσα εργασία υιοθετεί μία ελαφριά στοίβα HTML και JavaScript με διαδραστικό χάρτη μέσω του Leaflet. Το Leaflet επιλέγεται διότι συνδυάζει απλότητα, επεκτασιμότητα και άριστη προσαρμογή σε κινητά περιβάλλοντα, χαρακτηριστικά που ταιριάζουν με τις ανάγκες ενός δημοτικού συστήματος αναφορών. Ο ίδιος ο ιστότοπος επικοινωνεί με το backend αποκλειστικά μέσω REST API, ώστε η απεικόνιση και η διαχείριση των αναφορών να είναι ανεξάρτητες από την υλοποίηση του διακομιστή.

B.4.1 Leaflet

To Leaflet είναι μία από τις πιο ευρέως χρησιμοποιημένες βιβλιοθήκες για την δημιουργία γεωγραφικών εφαρμογών. Σύμφωνα με την επίσημη τεκμηρίωση του (Leaflet, n.d.), το Leaflet είναι μια ανοικτό κώδικα βιβλιοθήκη της JavaScript σχεδιασμένη για την εύκολη δημιουργία εφαρμογών με χάρτες. Είναι αρκετά ελαφριά, γρήγορη και υποστηρίζει βασικά χαρακτηριστικά για χάρτες όπως ζουμ, δείκτες και η προβολή αναδυόμενων παραθύρων. Η απλοϊκότητα του και το εύρος δυνατοτήτων του το κάνει μία αγαπημένη επιλογή για προγραμματιστές που θέλουν να εισχωρήσουν χάρτες μέσα στις εφαρμογές του χωρίς πολύπλοκα GIS λογισμικά.

Στο πλαίσιο των έξυπνων πόλεων, η ικανότητα του Leaflet να εισάγει εξωτερικά APIs και άλλα προγράμματα το κάνει ένα αρκετά χρήσιμο εργαλείο. Ακόμα, το Leaflet υποστηρίζει διάφορες μορφές γεωγραφικών δεδομένων οπότε μπορεί εύκολα να διαχειριστεί χωρικά σετ πληροφοριών από IoT συσκευές και βάσεις δεδομένων των πόλεων. Ένα παράδειγμα χρήσης του είναι, να μπορούν οι προγραμματιστές να εμφανίσουν τοποθεσίες δημόσιων υποδομών ή και να σημαδέψουν προβλήματα που έχουν καταχωρήσει οι πολίτες.

Οι Karampakakis et al. (2025) δίνουν μια περίπτωση για το πως χρησιμοποιείται το Leaflet σε πράξη στις έξυπνες πόλεις. Στην μελέτη τους, ανέπτυξαν μια διαδικτυακή πλατφόρμα για την ανάλυση και προβολή δεδομένων ανίχνευσης σε αστικές περιοχές. Η εφαρμογή συνδύαζε γραφήματα χρόνου με αλληλεπιδράσιμους χάρτες, επιτρέποντας στους χρήστες να παρατηρήσουν περιβαλλοντικά στοιχεία όπως η θερμοκρασία και η ποιότητα αέρα σε διαφορετικά μέρη της πόλης. Το Leaflet είχε ρόλο του βασικού στοιχείου χαρτογράφησης, παρέχοντας δυναμική οπτικοποίηση και ικανότητες διαδραστικότητας για τους χρήστες.

Πέρα από το Leaflet, υπάρχουν πολλά πλαίσια ανοικτού κώδικα που υποστηρίζουν την ανάπτυξη GIS πλατφορμών Οι Sejati et al. (2020) συζητάνε για πλαίσια όπου χρησιμοποιούν ανοικτά πρότυπα και τεχνολογίες. Η δικιά τους προσέγγιση περιλαμβάνει την εισαγωγή βάσεων με γεωγραφικά δεδομένα με εργαλεία για οπτικοποίηση μέσα από το διαδίκτυο, δίνοντας την δυνατότητα σε χρήστες να παρακολουθούν τυχόν αλλαγές διαχρονικά. Ενώ το Leaflet μπορεί να διαχειριστεί μόνο του το οπτικό κομμάτι, τέτοια πλαίσια παρέχουν το λειτουργικό μέρος της υποδομής για την αποθήκευση και διαχείριση των πληροφοριών. Η χρήση τέτοιων εργαλείων είναι εξίσου σημαντική για μικρότερες πόλεις που δεν μπορούν

να διαθέσουν ακριβά GIS λογισμικά. Με την χρήση αυτών των πλαισίων που συνδυάζουν το Leaflet με άλλες τεχνολογίες, οι πόλεις αυτές μπορούν να εφαρμόσουν προηγμένες χαρτογραφήσεις χωρίς μεγάλα έξοδα.

B.5 Τεχνολογίες Ανάπτυξης σε Κινητά

Η ζήτηση για κινητές εφαρμογές αυξάνεται συνεχώς και ραγδαία με τις εταιρείες να πρέπει να αναπτύξουν εφαρμογές και για Android και για iOS πλατφόρμες. Η δημιουργία δύο διαφορετικών εφαρμογών μπορεί να θεωρηθεί χρονοβόρα και ακριβή. Για αυτό τον λόγο, έχουν γίνει διάσημα τα framework πολλαπλών πλατφορμών, τα οποία δίνουν την δυνατότητα σε έναν προγραμματιστή να γράψει σε μία βάση κώδικα όπου τρέχει σε πολλαπλές συσκευές μειώνοντας σε μεγάλο βαθμό το κόστος και τον χρόνο κατά την ανάπτυξη (Majchrzak & Grønli, 2017). Οι κύριες προσεγγίσεις σε αυτό τον τομέα είναι οι εγγενές, οι υβριδικές και οι προοδευτικές εφαρμογές ιστού (PWA). Οι εγγενές, αλλιώς native, είναι οι εφαρμογές όπου χρησιμοποιούν γλώσσες προγραμματισμού συγκεκριμένης πλατφόρμας όπως Java και Kotlin για Android και Swift για iOS. Αυτές προσφέρουν την καλύτερη απόδοση αλλά χρειάζονται ζεχωριστές βάσεις κώδικα για να λειτουργήσουν. Οι υβριδικές, στην άλλη πλευρά, χρησιμοποιούν συστήματα ιστού όπως HTML, CSS και JavaScript για την κατασκευή εφαρμογών όπου τρέχουν μέσα σε ένα native “δοχείο”. Με αυτή την προσέγγιση, συνδέεται η ελευθερία του προγραμματισμού στον ιστό με την πρόσβαση σε δυνατότητες των συσκευών μέσω native plugins. Τα PWAs προσφέρουν την εμπειρία του διαδικτύου υποστηρίζοντας χρήση χωρίς σύνδεση σε αυτό αλλά μειονεκτούν καθώς δεν έχουν πρόσβαση σε native APIs.

B.6 Εργαλεία Κινητής Εφαρμογής

Στην παρούσα εργασία επιλέγεται η υβριδική προσέγγιση με Ionic React, TypeScript και Capacitor. Ο συνδυασμός αυτός επιτρέπει την χρήση μίας ενιαίας βάσης κώδικα και την αξιοποίηση του οικοσυστήματος της React, ενώ μέσω του Capacitor παρέχεται ασφαλή πρόσβαση σε λειτουργίες της συσκευής, όπως ο γεωεντοπισμός και η κάμερα. Η επιλογή ικανοποιεί τις λειτουργικές απαιτήσεις της πλατφόρμας (λήψη φωτογραφίας και αυτόματη καταγραφή τοποθεσίας) με χαμηλό κόστος ανάπτυξης και συντήρησης.

B.6.1 Ionic Framework

To Ionic είναι ένα framework ανοικτού κώδικα όπου σχεδιάστηκε για την δημιουργία υβριδικών και πολλαπλών πλατφορμών εφαρμογών για κινητά χρησιμοποιώντας web τεχνολογίες. Όταν παρουσιάστηκε για πρώτη φορά το 2013, το Ionic ήταν αρχικά επικεντρωμένο σε Angular αλλά αργότερα επεκτάθηκε σε React και σε άλλα πλαίσια, κάνοντας το ιδανική επιλογή για προγραμματιστές (Ionicframework, n.d.). Το ίδιο προσφέρει μια βιβλιοθήκη γεμάτη με ήδη έτοιμα κομμάτια που βοηθούν στην δημιουργία UI, βελτιώνοντας απίστευτα την σχεδίαση εφαρμογών με ομοιόμορφη μορφή και λειτουργία σε όλες τις πλατφόρμες.

Μία από τις πιο βασικές δυνάμεις του Ionic είναι η χρήση ενός μοναδικού κώδικα για τις πλατφόρμες κάτι το οποίο μειώνει την εξέλιξη του και τα κόστοι διατήρησης (Sonar & Dharmadhikari, 2023). Αυτό είναι ειδικά χρήσιμο σε έργα όπως αυτό μιας έξυπνης πόλης όπου ο χρόνος κυκλοφορίας των εφαρμογών και η ελαχιστοποίηση δαπάνης των πόρων είναι κρίσιμα. Οι εφαρμογές του Ionic τρέχουν σε WebView όπου λειτουργεί σαν πρόγραμμα περιήγησης μέσα σε τοπικό περιβάλλον, δίνοντας τους την δυνατότητα να έχουν πρόσβαση σε λειτουργίες του κινητού ενώ διατηρεί την συμβατότητα ανάμεσα στις διάφορες συσκευές.

Ακόμα, είναι δυνατή η υποστήριξη PWAs. Έτσι, το Ionic μπορεί να εφαρμόσει την ίδια βάση κώδικα ως εφαρμογή διαδικτύου. Αυτή η ελευθερία σιγουρεύει πως οι εφαρμογές μπορούν να χρησιμοποιηθούν από οποιοδήποτε χρήστη σε κινητά, τάμπλετ καθώς και λάπτοπ (Ionicframework, n.d.). Για την μορφή της τελικής διεπαφής χρησιμοποιείται σύστημα με προσαρμόσιμα κομμάτια λογισμικού κάνοντας την εφαρμογή να δουλεύει σε οινόνες ανεξαρτήτως των διαστάσεων χωρίς παραπάνω δουλειά στον προγραμματισμό.

B.6.1.1 React

Η React είναι μια βιβλιοθήκη της JavaScript η οποία είναι ανεπτυγμένη από την Meta για την δημιουργία διεπαφών χρηστών. Η ίδια εισήγαγε την έννοια του εικονικού μοντέλου DOM, το οποίο βελτιώνει την απόδοση μειώνοντας τις άμεσες αλλαγές στο πραγματικό DOM (React, n.d.). Οι εφαρμογές μέσω React χτίζονται με την χρήση επαναχρησιμοποιήσιμων UI μπλοκ όπου μπορούν να συνδυαστούν και να κατασκευάσουν περίπλοκες διεπαφές. Επομένως, βελτιώνεται η οργάνωση του κώδικα και η συνεχής χρήση του κάτι που είναι ιδανικό ως επιλογή για πλαφόρμες που χρειάζονται επεκτασιμότητα.

B.6.1.2 Ionic React

Όπως αναφέρθηκε, το Ionic άρχισε βασισμένο στην Angular αλλά πλέον υποστηρίζει πολλαπλά framework για frontend συμπεριλαμβανομένης της React. Το Ionic React συνδυάζει τις λειτουργίες UI του Ionic με την αρχιτεκτονική της React, κάνοντας το ελκυστικό για όσους είναι οικείοι με το οικοσύστημα της. Με την χρήση των δύο μαζί εμφανίζονται πολλά πλεονεκτήματα. Αρχικά, το πιο σημαντικό είναι πως επιτρέπει στους προγραμματιστές να φτιάχνουν διεπαφές με έτοιμα εργαλεία, το οποίο βοηθάει στην συντήρηση του κώδικα. Ακόμα, χρησιμοποιείται η γλώσσα TypeScript για την σύνταξη κώδικα, όπου εφαρμόζει ασφάλεια και μειώνει τα σφάλματα κατά την ώρα εκτέλεσης (TypeScript, n.d.). Αυτό παίζει μεγάλο ρόλο σε έργα μεγάλης διάστασης που η αξιοπιστία είναι καθοριστική.

Σε αυτή την εργασία, επιλέχθηκε το Ionic React επειδή προσφέρει και ευκολία στην ανάπτυξη αλλά και κατάλληλη απόδοση. Η χρήση της TypeScript ενισχύει περαιτέρω την επιλογή αφού βελτιώνει την ποιότητα του κώδικα κάνοντας την εφαρμογή πιο δομημένη και διατηρήσιμη.

B.6.1.3 Capacitor

Το Capacitor είναι ένα περιβάλλον εκτέλεσης ανοικτού κώδικα φτιαγμένο από την ομάδα του Ionic για την αντικατάσταση του Cordova. Χρησιμοποιείται ως ενδιάμεσο εργαλείο μεταξύ των τεχνολογιών ιστού με τις τοπικές λειτουργίες των συσκευών, επιτρέποντας την κλήση APIs από την JavaScript (Capacitor, n.d.). Το Capacitor υποστηρίζει πολλές υπηρεσίες όπως την πρόσβαση στην κάμερα, τοποθεσία και την αποθήκευση αρχείων οι οποίες είναι απαραίτητες για μοντέρνες εφαρμογές ειδικά στον τομέα της έξυπνης πόλης. Ένα από τα πολλά θετικά που έχει σε σχέση με παλιά εργαλεία όπως το Cordova είναι η καλύτερη ενσωμάτωση σε νέα frameworks. Έχει απλή διαμόρφωση και την δυνατότητα επαναφόρτωσης της διεπαφής σε πραγματικό χρόνο κατά την ώρα της ανάπτυξης, γεγονός που επιταχύνει την διαδικασία αυτή. Ακόμα, παρέχει χρήσιμα plugins όπου μπορούν να χρησιμοποιηθούν για διάφορες διαδικασίες και την δυνατότητα δημιουργίας καινούργιων από την αρχή.

Η επιλογή του Capacitor έγινε για την πρόσβαση στην κάμερα της συσκευής και για την εύρεση της γεωγραφικής τοποθεσίας. Αυτά τα χαρακτηριστικά είναι σημαντικά για την καταχώρηση προβλημάτων στις έξυπνες πόλεις καθώς θα χρειαστούν αν βγάλουν φωτογραφίες από ζημιές στον δρόμο και να τις υποβάλουν μαζί με την τοποθεσία τους.

Το Capacitor το έκανε αυτό εφικτό χωρίς την προσθήκη συναρτήσεων κώδικα πάνω στην πλατφόρμα της συσκευής.

B.6.2 Android και Android Studio

Το Android μπορεί να θεωρηθεί παγκοσμίως το πιο ευρέως χρησιμοποιημένο λειτουργικό σύστημα για κινητά και τάμπλετ. Είναι χτισμένο πάνω σε μια αρχιτεκτονική με πολλά επίπεδα που περιλαμβάνει το kernel του Linux, ένα περιβάλλον εκτέλεσης, βιβλιοθήκες και framework εφαρμογών (Goel & Singal, 2021). Αυτό το σχέδιο κάνει το Android ευέλικτο και επεκτάσιμο το οποίο ευθυγραμμίζεται ιδιαίτερα με τα ζητούμενα των εφαρμογών για τις έξυπνες πόλεις που χρειάζονται αρκετές λειτουργίες. Από προγραμματιστική πλευρά, το ανοικτό μοντέλο του Android και διείσδυση του στην αγορά δικαιολογεί την επιλογή του ως ιδιαίτερη πλατφόρμα ανάπτυξης.

Ενώ το Ionic React και το Capacitor χειρίζονται τον κώδικα, το Android Studio είναι απαραίτητο για την παραγωγή του τελικού προγράμματος για Android. Σύμφωνα με την Thamizharasi (2016), το Android Studio είναι ένα πλήρες περιβάλλον IDE όπου απλοποιεί την ανάπτυξη εφαρμογών με λειτουργίες όπως εξομοιωτές συσκευών και εργαλείο εντοπισμού σφαλμάτων. Η τυπική διαδικασία ροής του προγράμματος αυτού είναι η δημιουργία ενός έργου, το γράψιμο του κώδικα, η κατασκευή της εφαρμογής και η μεταφορά της στην συνδεμένη συσκευή ή προσομοίωση αυτής. Σε αυτή την εργασία, η χρήση αυτού του εργαλείου έγινε μόνο για το τελικό κομμάτι για την δημιουργία APK αρχείου από το έργο και την εγκατάσταση του στην δοκιμαστική συσκευή.

B.6.3 Συγκριτικές Επισκοπήσεις και Πρακτικές

Αρκετές ακαδημαϊκές μελέτες έχουν συγκρίνει διάφορα frameworks πολλαπλών πλατφορμών όπως το Ionic, React Native και Flutter. Οι Majchrzak και Grønli (2017) βρήκαν ότι το Ionic είναι το πιο κατάλληλο όταν είναι ζητούμενη η γρήγορη ανάπτυξη και η σταθερότητα στην διεπαφή χρήστη ενώ η React Native μπορεί να προσφέρει καλύτερη απόδοση σε εφαρμογές με μεγάλη αλληλεπίδραση. Σε μια άλλη μελέτη, αναφέρθηκε πως η εξάρτηση του Ionic πάνω στο WebView της μπορεί να επηρεάσει αρκετά την επίδοση περίπλοκων εφαρμογών αλλά στις πιο πολλές περιπτώσεις αυτό δεν αποτελεί σημαντικό περιορισμό (You & Hu, 2021). Οι Sonar και Dharmadhikari (2023) δίνουν έμφαση στο ότι το κύριο πλεονέκτημα του Ionic είναι η απλοϊκότητα του και η πλούσια βιβλιοθήκη που περιέχει όπου περιορίζει την ανάγκη

για δημιουργία UI από την αρχή. Στην άλλη πλευρά, οι προγραμματιστές θα πρέπει να είναι προσεκτικοί ως προς την βελτίωση της απόδοσης όταν έχουν εφαρμόσει βαριά εφέ όπως κινήσεις και μεγάλα σετ δεδομένων.

B.7 Σχετικά Έργα

Όταν αναπτύσσεται μια κινητή εφαρμογή πάνω στις έξυπνες πόλεις, είναι σημαντική η κατανόηση παρόμοιων συστημάτων και των τεχνολογίων τους. Η ανάλυση άλλων έργων δίνει σημαντικές γνώσεις για οποιεσδήποτε προκλήσεις που πιθανά αντιμετοπίζουν και επιλογές στην σχεδίαση όπου μπορούν να εφαρμοστούν σε νέες υλοποιήσεις. Αυτό το τμήμα αξιολογεί σχετικές μελέτες περίπτωσης που αφορούν εφαρμογές για την αναφορά αστικών προβλημάτων.

B.7.1 CitySolution

Ένα αξιόλογο παράδειγμα είναι η περίπτωση του CitySolution, μια εφαρμογή για έξυπνες πόλεις φτιαγμένη από τους Shama et al. (2024). Η πλατφόρμα αυτή σχεδιάστηκε για την βελτίωση της απόδοσης του Μπανγκλαντές ως προς την διαχείριση παραπόνων από τους δήμους, αντικαθιστώντας τις αργές και χειροκίνητες διαδικασίες με ψηφιακές. Το ίδιο χωρίζεται σε δύο διαφορετικά συστήματα, ένα για τους πολίτες και ένα για το προσωπικό των αρχών. Οι πολίτες μπορούν να αναφέρουν προβλήματα όπως κατεστραμμένους δρόμους καταχωρώντας απλά μια φωτογραφία με κείμενο και την τοποθεσία τους.

Μια κύρια λειτουργία του CitySolution ήταν η προσαρμογή μηχανικής μάθησης για την αυτόματη κατηγοριοποίηση των παραπόνων σε ήδη υπαρχτές κατηγορίες. Αυτό έγινε μέσω της χρήσης μοντέλου βαθιάς μάθησης εκπαιδευμένο από το Google Teachable Machine, κάτι που βοήθησε στην μείωση εργασίας από τους υπαλλήλους της πόλης και επιτάχυνε την όλη διαδικασία. Η εφαρμογή ακόμα υποστηρίζει άλλες γλώσσες για την ικανοποίηση των προτιμήσεων, κάνοντας την προσβάσιμη για περισσότερους χρήστες. Η διαχείριση των δεδομένων έγινε μέσω του Firebase, διασφαλίζοντας ενημερώσεις σε πραγματικό χρόνο και αλληλεπίδραση μεταξύ των πολιτών και των αρχών.

Η ασφάλεια αντιμετωπίστηκε μέσω χαρακτηριστικών όπως ο έλεγχος των υπαλλήλων μέσω QR, η προσαρμογή παρακολούθησης των παραπόνων για τους πολίτες και συστημάτων

που ειδοποιούν για ενημερώσεις της κατάστασης τους. Αυτές οι προσθήκες συνολικά ενίσχυσαν την εμπιστοσύνη των πολιτών στο σύστημα. Ακόμα, η μελέτη αυτή ανέφερε πως πριν το CitySolution τα περισσότερα συστήματα παραπόνων ήταν ανεπαρχής και συχνά χρειαζόταν φυσικές επισκέψεις για την αναφορά ενός θέματος. Φέρνοντας αυτό το αυτοματοποιημένο σύστημα, ο νέος στόχος τους έγινε η ενεργή συμμετοχή των πολιτών και η βελτίωση ανταπόκρισης των δημόσιων υπηρεσιών.

B.7.2 Εφαρμογή με χρήση Ionic

Άλλη μία περίπτωση είναι αυτή από τους Dunka et al. (2017), όπου μελετάνε την ανάπτυξη μια υβριδικής εφαρμογής για κινητά με λειτουργίες χαρτογράφησης. Η εκτέλεση της έγινε συνδυάζοντας το Ionic με το Leaflet για την προβολή γεωγραφικών δεδομένων πάνω σε μία διαδραστική διεπαφή χάρτη. Όπως εξήγησαν οι συγγραφείς, η εφαρμογή τους χτίστηκε ως ένα μονό έργο ικανό να τρέχει και σε Android και iOS μειώνοντας μελλοντική ταλαιπωρία διατήρησης διαφορετικών κώδικων.

Η διαδικασία επικεντρώθηκε στην υλοποίηση του Leaflet για δυναμική απεικόνιση στον χάρτη ενεργοποιώντας χαρακτηριστικά όπως ζουμάρισμα και τοποθέτηση δεικτών. Οι λειτουργίες των συσκευών όπως η κάμερα και η γεωγραφική τοποθεσία έγιναν προσβάσιμες μέσω plugins. Οι ίδιοι ανέφεραν πως αυτή η προσέγγιση μείωση σημαντικά τον χρόνο ανάπτυξης σε σχέση με άλλες πιο παραδοσιακές μεθόδους ενώ διατηρούσαν την εμπειρία του χρήστη ικανοποιητική σε σχέση με τον στόχο της εφαρμογής.

Όσον αφόρα τους περιορισμούς, αναφέρθηκε πως οι υβριδικές εφαρμογές μπορεί πιθανά να αντιμετωπίσουν θέματα στην απόδοση όταν βρίσκονται σε υπερφόρτωση. Ωστόσο, για εφαρμογές όπου επικεντρώνονται πάνω σε οπτικοποίηση δεδομένων αυτά τα προβλήματα δεν έχουν σημαντική επίδραση. Τέλος, η μελέτη συμπεραίνει πως η χρήση framework όπως το Ionic μπορεί να θεωρηθεί ιδανική για πλατφόρμες γρήγορης απόδοσης και γεωγραφικών λειτουργιών.

B.7.3 Διδαγμα από τις μελέτες

Οι δύο μελέτες παρουσιάζουν ομοιότητες με το θέμα αυτής της εργασίας αλλά ταυτόχρονα μεταφέρουν σημαντικές διαφορές όπου επηρέασαν τις αποφάσεις κατά την σχεδίαση. Η πρώτη, κάνει επίδειξη της αποτελεσματικότητας που έχει η προσαρμογή χαρακτηριστικών όπως η

αναφορά προβλημάτων μέσω φωτογραφιών, τοποθεσίας και αυτοματοποίησης σε κατηγορίες. Η τρέχων εργασία μπορεί να μην έχει εφαρμόσει μηχανική μάθηση για κατηγοριοποίηση αλλά παίρνει τις βασικές ιδέες ώστε να δώσει την δυνατότητα στους πολίτες να υποβάλλουν τις δικές τους αναλυτικές αναφορές.

Σε σχέση με την δεύτερη, γίνεται αντιληπτή η σωστή επιλογή της χρήσης του Ionic σε αυτό το έργο. Η διατήρηση μια ενιαίας βάσης κώδικα είναι σημαντική και βοηθάει με τον στόχο δημιουργίας μιας προσιτής και αποδοτικής λύσης για τους δήμους. Ακόμα, η προσθήκη χαρτογράφησης μέσω Leaflet στην μελέτη τους ενισχύει την απόφαση υλοποίησης αυτής της τεχνολογίας για την οπτικοποίηση των αναφορών σε αλληλεπιδραστικό χάρτη.

Κεφάλαιο C

Ανάλυση και Σχεδιασμός

C.1 Ανάλυση Απαιτήσεων Χρηστών

Hανάπτυξη μιας τέτοιας εφαρμογής όπου έχει να κάνει με τις έξυπνες πόλεις έχει χύρια απαίτηση να κατανοηθούν οι πραγματικές ανάγκες των χρηστών αυτής. Η εφαρμογή και ιστοσελίδα που παρουσιάζονται σε αυτή την εργασία, αποτελούνται από δύο βασικές κατηγορίες χρηστών, τους πολίτες και τους υπαλλήλους του δήμου. Ο ρόλος των πολιτών είναι να εντοπίζουν τυχόν προβλήματα και οι ίδιοι να αναφέρουν αυτά στον δήμο χρησιμοποιώντας την εφαρμογή μέσω του κινητού τους. Από την άλλη, οι υπάλληλοι μέσω της διαδικτυακής ιστοσελίδας έχουν την δουλειά να διαχειρίζονται τις αναφορές και να κάνουν αλλαγές σε αυτές ανάλογα με την κατάσταση του προβλήματος.

Στην σημερινή εποχή, η προσέγγιση προς τον σχεδιασμό τέτοιων συστημάτων έχει στραφεί κυρίως προς τον άνθρωπο και όχι απλά μόνο στο κομμάτι της τεχνολογίας. Όπως αναφέρει η μελέτη των Janoskova et al. (2021), από μόνη της η τεχνολογία δεν μπορεί να σιγουρέψει την επιτυχία ενός τέτοιου συστήματος. Αυτά πρέπει να σχεδιαστούν με σκοπό να επιλύσουν τις καθημερινές ανάγκες ενός πολίτη, να είναι προσβάσιμα ως προς την λειτουργία και να ενισχύουν την σύνδεση και εμπιστοσύνη του πολίτη με του δήμου. Συνεπώς, αν δεν καλύπτουν τις ανάγκες των πολιτών και αν η χρήση τους θεωρείται δύσκολη τότε χάνουν τον σκοπό τους και επομένως εγκαταλείπονται από τους χρήστες.

Οι βασικές ανάγκες για τους πολίτες είναι:

- Να μπορούν εύκολα και γρήγορα να καταγράφουν ένα πρόβλημα χωρίς δυσκολίες

- Εύκολη επισύναψη φωτογραφίας και εγγραφή κειμένου
- Αυτόματη καταχώρηση τοποθεσίας της αναφοράς
- Τρόπος ενημέρωσης της κατάστασης της

Αντίστοιχα, οι υπάλληλοι του δήμου χρειάζονται:

- Ένα οργανωμένο περιβάλλον εργασίας όπου φαίνονται οι αναφορές
- Δυνατότητα διαχείρισης των καταχωρήσεων αυτών
- Να μπορούν να ενημερώνουν τις αναφορές

Άρα, σύμφωνα με όσα αναφέρθηκαν, η επιτυχία μιας τέτοιας εφαρμογής έξυπνης πόλης επηρεάζεται αρκετά από το πόσο απλά και ορθά εξυπηρετούνται και καλύπτονται οι ομάδες των πολιτών και των δήμων. Επιπλέον, είναι κατανοητό πως η εμπειρία του χρήστη (UX) είναι σημαντική τόσο για να γίνει η εφαρμογή αποδεκτή όσο και να διατηρήσει η ίδια ενεργή την χρήση της στο μέλλον. Οπότε, αν η όλη διαδικασία φαίνεται κουραστική και πολύπλοκη ως προς την χρήση, τότε όσο τέλεια τεχνολογικά και να είναι πάντα υπάρχει το ενδεχόμενο να μπορεί να αποτύχει στο τέλος.

Η πλατφόρμα που αναπτύχθηκε προσπαθεί να καλύψει όλες αυτές τις ανάγκες και απαιτήσεις που προαναφέρθηκαν. Η εφαρμογή για τα κινητά απλοποιεί αρκετά την διαδικασία της αναφοράς ενός προβλήματος με τρία απλά βήματα. Ο χρήστης αρχικά μέσω της εφαρμογής, φωτογραφίζει το σημείο του προβλήματος, προσθέτει μια σύντομη περιγραφή για αυτό και το σύστημα από μόνο του αποθηκεύει την γεωγραφική θέση της συσκευής του. Κατά την καταχώρηση, τα δεδομένα αυτά αποθηκεύονται και εμφανίζονται στην ιστοσελίδα της πλατφόρμας όπου παρουσιάζονται πάνω σε ένα χάρτη ως τοποθεσίες σε πραγματικό χρόνο. Οι υπεύθυνοι έχουν πρόσβαση σε αυτή την ιστοσελίδα και μπορούν να φιλτράρουν τις ενεργές καταχωρίσεις με βάση την κατάσταση τους και να τις διαχειριστούν ανάλογα με την πράξη του δήμου. Η ενημέρωση και επικοινωνία των δύο αυτών πλευρών γίνεται μέσω των αλλαγών πάνω στην κάθε αναφορά.

Οι κύριες λειτουργίες που απαιτούνται για το σύστημα:

Πίνακας C.1: Λειτουργικές Απαιτήσεις.

Απαιτηση

- 1 Οι πολίτες να μπορούν να υποβάλουν αναφορά προβλήματος με φωτογραφία και περιγραφή.
- 2 Η εφαρμογή καταγράφει αυτόματα τη γεωγραφική τοποθεσία της αναφοράς.
- 3 Οι πολίτες μπορούν να βλέπουν το ιστορικό και την κατάσταση των αναφορών τους.
- 4 Οι υπάλληλοι του δήμου έχουν πρόσβαση σε όλες τις αναφορές μέσω διαδικτυακής πλατφόρμας.
- 5 Οι υπάλληλοι μπορούν να εγκρίνουν, ενημερώνουν ή διαγράφουν αναφορές.
- 6 Το σύστημα αποθηκεύει όλες τις αναφορές σε ασφαλή βάση δεδομένων.

Εκτός από αυτές τις λειτουργίες, το σύστημα πρέπει να πληροί ορισμένες απαιτήσεις προς την ποιότητα:

Πίνακας C.2: Ποιοτικές Απαιτήσεις.

Απαιτηση

- 1 Η εφαρμογή για κινητά πρέπει να λειτουργεί σε όλες τις συσκευές Android.
- 2 Η διαδικτυακή πλατφόρμα πρέπει να είναι συμβατή σε όλους τους browser.
- 3 Οι κλήσεις API πρέπει να έχουν γρήγορο χρόνο απόκρισης.
- 4 Η ανταλλαγή δεδομένων πρέπει να γίνεται με ασφάλεια.
- 5 Η διεπαφή χρήστη πρέπει να είναι απλή και κατανοητή.

Η προσέγγιση αυτή δεν έγινε τυχαία και στηρίζεται πάνω στο γεγονός πως οι πολίτες δεν είναι απλά παθητικοί δέκτες υπηρεσιών αλλά συμβάλλουν μέσω της ενεργής συμμετοχής τους πάνω στον σχεδιασμό του περιβάλλοντος όπου ζούνε εκείνοι. Η συλλογή δεδομένων από την ίδια την κοινότητα της πόλης για προβλήματα της όπως βλάβες, φώτα και λακκούβες μπορεί να καλυτερεύσει την εμπιστοσύνη ανάμεσα τους και να ενισχύσει την αποδοτικότητα του δήμου πάνω σε τέτοια θέματα. Τέλος, το πιο βασικό στοιχείο στην φάση της ανάλυσης των ζητούμενων μπορεί να υπερηφανεί ο ορισμός των ρόλων. Η μεριά του πολίτη έχει τον ρόλο να καταγράψει και να παρατηρεί τυχόν προβλήματα, ενώ η άλλη πλευρά του δήμου λειτουργεί ως διαχειριστής της όλης ενέργειας επεξεργάζοντας και ενημερώνοντας τις δηλώσεις. Η εφαρμογή οφείλει να σχεδιαστεί έτσι ώστε κάθε μια από τις πλευρές να γνωρίζει και να καταλαβάνει τον σκοπό και την δουλειά της, κάτι που ταυτόχρονα διαφοροποιεί αλλά αυξάνει την αποτελεσματικότητα της διαδικασίας.

C.2 Επιλογή Τεχνολογιών

Η επιλογή των κατάλληλων τεχνολογιών για αυτό το έργο ήταν σημαντική για την επίτευξη διατήρησης, επεκτασιμότητας και ασφάλειας. Το όλο σύστημα είναι φτιαγμένο από πολλά διαφορετικά μέρη, την κινητή εφαρμογή, την διαδικτυακή ιστοσελίδα, το backend server και την βάση δεδομένων σε cloud. Κάθε μέρος αποτελείται από τα δικά του εργαλεία όπου είναι αξιόπιστα και είναι συμβατά μεταξύ τους. Το τελικό μοντέλο συνδυάζει τεχνολογίες και ασφαλή διαχείριση δεδομένων κάτι που το κάνει ιδανικό για πλατφόρμα έξυπνης πόλης όπου χρειάζεται να είναι αποδοτικό και κατάλληλο προς την εμπειρία του χρήστη. Οι τεχνολογίες που επιλέχθηκαν και οι ρόλοι τους αναφέρονται στον Πίνακα C.3.

Πίνακας C.3: Οι τεχνολογίες που επιλέχθηκαν

Τεχνολογία	Σκοπός
Ionic React	Υβριδική ανάπτυξη εφαρμογής για κινητά.
Capacitor	Ενσωμάτωση τοπικών λειτουργιών της συσκευής (Κάμερα, GPS).
Leaflet.js	Οπτικοποίηση του χάρτη.
Node.js	Περιβάλλον διαχείρισης αιτημάτων API.
Express.js	Δημιουργία των RESTful APIs.
MongoDB Atlas	Βάση δεδομένων στο cloud για την αποθήκευση αναφορών και δεδομένων χρηστών.
TypeScript	Γλώσσα για καλύτερη συντήρηση και πρόληψη σφαλμάτων.
Bcrypt	Κρυπτογράφηση δεδομένων χρήστη.
Multer	Διαχείριση εικόνων για το backend.
CORS	Επιτρέπει με ασφάλεια τα API αιτήματα μεταξύ κινητού, ιστού και server.

C.2.1 Αιτιολόγηση και Λεπτομέρειες

Ionic React, Capacitor, and Leaflet

Η κινητή εφαρμογή υλοποιήθηκε με την χρήση του Ionic React για τις ικανότητες του πάνω στην ανάπτυξη πολλαπλών πλατφορμών. Το Capacitor συμπεριλήφθηκε ώστε να μπορεί να υπάρχει πρόσβαση σε δυνατότητες όπως κάμερα και τοποθεσία. Τέλος το Leaflet επιλέχθηκε για την προβολή των καταχωρήσεων πάνω σε περιβάλλον χάρτη. Αυτές οι επιλογές αναπτύχθηκαν περισσότερο στο Κεφάλαιο 2.

Node.js

Το λειτουργικό κομμάτι του συστήματος είναι φτιαγμένο με το Node.js, ένα ασύγχρονο σύστημα εκτέλεσης της JavaScript όπου διαχειρίζεται αποτελεσματικά πολλαπλές συνδέσεις. Το Node είναι ευρέως αναγνωρισμένο για την αρχιτεκτονική του, καθιστώντας το κατάλληλο για εφαρμογές πραγματικού χρόνου που περιλαμβάνουν συχνές αλληλεπιδράσεις μέσω API (NodeJS, n.d.).

Express.js

Το Express.js, είναι ένα ελαφρύ framework που βασίζεται στο Node, γνωστό για την δημιουργία τελικών API σημείων και δομών δρομολόγησης (ExpressJS, n.d.). Ακόμα, προσφέρει ενδιάμεσα εργαλεία όπως το Cors και το Multer. Ο συνδυασμός του με το Node επιτρέπει την γρήγορη ανάπτυξη και διαχωρισμό των διαδρομών των λειτουργιών του συστήματος όπως η υποβολή αναφορών και η ανάκτηση των δεδομένων.

Mongo DB Atlas

Η βάση δεδομένων εφαρμόστηκε μέσω του MongoDB Atlas, μιας cloud επιλογής όπου δεν χρησιμοποιεί SQL. Σε αντίθεση με άλλες παραδοσιακές βάσεις, το MongoDB αποθηκεύει τα δεδομένα σε αρχεία BSON όπου μοιάζουν με JSON αλλά επιτρέπουν ταχύτερη πρόσβαση σε αυτά. Αυτό είναι ιδιαίτερα χρήσιμο για την αποθήκευση αναφορών των χρηστών όπου αποτελούνται από διάφορα στοιχεία όπως εικόνα, κείμενο και γεωγραφικά στοιχεία. Το MongoDB εξασφαλίζει υψηλή διαθεσιμότητα και ασφάλεια κάνοντας το εξαιρετική επιλογή για υπηρεσία στο cloud (Chauhan, 2019).

TypeScript

Η λογική της εφαρμογής γράφτηκε σε TypeScript για την συντήρηση του κώδικα και την μείωση σφαλμάτων. Όπως αναφέρει ο Holmberg (2023), η αλλαγή από JavaScript σε TypeScript προσφέρει σημαντικά πλεονεκτήματα. Κάποια από αυτά είναι η καλύτερη ανάγνωση του κώδικα και ο βελτιωμένος εντοπισμός σφαλμάτων κατά την ανάπτυξη. Αυτά τα οφέλη είναι απαραίτητα για μεγάλα έργα όπου προτεραιότητα είναι η δομή και η αξιοπιστία.

Bcrypt

Η ασφάλεια ήταν μια σημαντική ανησυχία ειδικά στην διαχείριση πληροφοριών των χρηστών. Το Bcrypt χρησιμοποιήθηκε για την κρυπτογράφηση με κατακερματισμό των αποθηκευμένων κωδικών πρόσβασης, εγγυώντας πως αυτοί θα παραμείνουν ασφαλείς. Ο κατακερματισμός μειώνει τον κίνδυνο επιθέσεων brute-force και άλλων μεθόδων. Έτσι το Bcrypt μπορεί να θεωρηθεί μια εξαιρετική λύση για την ασφάλεια ελέγχου ταυτότητας (Sriramya & Karthika, 2015).

Multer

Το Multer είναι ένα ενδιάμεσο λογισμικό του Node.js όπου υποστηρίζει δεδομένα πολλαπλών τυμάτων (Multer, n.d.). Αυτό χρειάζεται για την διαχείριση των αρχείων όπως το ανέβασμα των φωτογραφιών από την κινητή εφαρμογή καθώς τις επεξεργάζεται και τις αποθηκεύει για τις επόμενες λειτουργίες του backend.

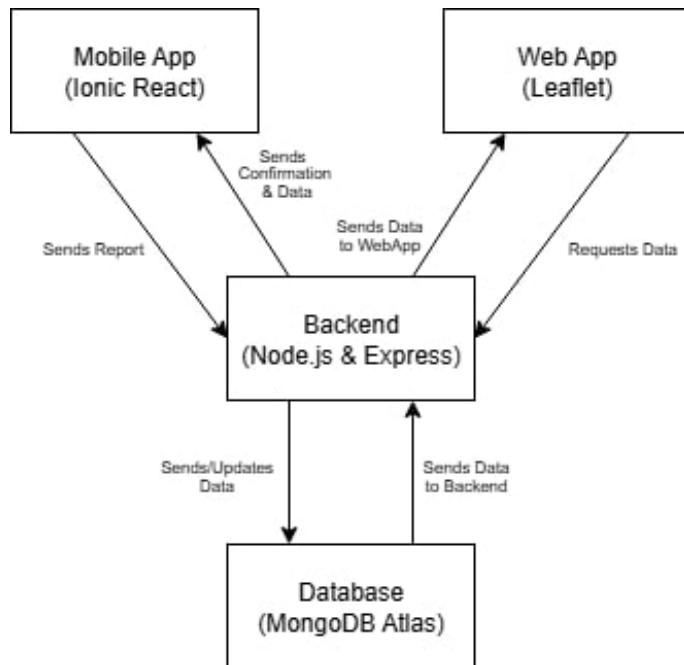
CORS

Το CORS είναι ένα πακέτο του Node όπου λειτουργεί σαν ενδιάμεσο του Express όπου επιτρέπει την ασφαλή κοινή χρήση μεταξύ προελεύσεων, αποτρέποντας προβλήματα ασφαλείας από τα προγράμματα περιήγησης όταν ζητούν πρόσβαση στο API (CORS, n.d.).

C.3 Αρχιτεκτονική Συστήματος

Η πλατφόρμα όπου αναπτύχθηκε για αυτή την εργασία ακολουθεί μία αρχιτεκτονική τριών επιπέδων. Αυτή η απόφαση να διαχωριστούν σε μικρότερα μέρη έγινε για την βελτίωση επεκτασιμότητας και συντήρησης των κώδικων. Έχοντας τις λειτουργίες της πλατφόρμας σε διαφορετικά συστήματα, έγινε ευκολότερη η ενημέρωση και η τροποποίηση οποιουδήποτε κομματιού χωρίς να επηρεάζει άμεσα τα υπόλοιπα.

Η κύρια ιδέα της σχεδίασης είναι ότι οι βασικοί χρήστες, όπου είναι οι πολίτες, αλληλεπιδρούν μέσω της εφαρμογής για κινητά, ενώ η άλλη ομάδα χρηστών, οι υπάλληλοι των δήμων, χρησιμοποιούν την διαδικτυακή ιστοσελίδα. Αυτές οι δύο διεπαφές επικοινωνούν με ένα backend server όπου διαχειρίζεται την ροή δεδομένων και τις λειτουργίες της βάσης. Το server έχει τον ρόλο να βοηθάει στην μεταφορά των πληροφοριών μεταξύ των δύο ομάδων και της βάσης, είτε αυτές υποβάλλουν είτε λαμβάνουν στοιχεία.



Σχήμα C.1: Αρχιτεκτονική της Πλατφόρμας Αναφοράς Έξυπνων Πόλεων

C.3.1 Κινητή Εφαρμογή

Η κινητή εφαρμογή αποτελεί το κύριο εργαλείο για τους κατοίκους. Τους δίνει την δυνατότητα να βγάλουν μια φωτογραφία από ένα πρόβλημα, όπως μια λακκούβα ή ένα σπασμένο φως δρόμου, να γράψουν μια σύντομη περιγραφή και να καταχωρίσουν την αναφορά τους. Κατά την καταχώρηση η γεωγραφική τοποθεσία του προβλήματος λαμβάνεται αυτόματα μέσω της συσκευής. Αυτές οι τρεις πληροφορίες υποβάλλονται στην βάση δεδομένων με το πάτημα ενός κουμπιού.

Η συνολική λειτουργία μπορεί να ακούγεται απλή αλλά οι τεχνολογίες πίσω που την κάνουν δυνατή είναι σημαντικές. Η εφαρμογή δημιουργήθηκε με την χρήση του framework Ionic και της React. Η ίδια εκτελείται σε TypeScript για την μείωση σφαλμάτων και χρησιμοποιεί το Capacitor για την πρόσβαση στις υπηρεσίες της συσκευής που χρειάζεται για να λειτουργεί όπως η κάμερα, GPS και την δυνατότητα ανοίγματος των ρυθμίσεων μέσω εφαρμογής για την πιθανή ενεργοποίηση δικαιωμάτων χειροκίνητα.

Όλες οι πληροφορίες όπου συλλέγονται από την εφαρμογή στέλνονται στο backend μέσω της χρήσης της επιλογής αιτήματος για ποστάρισμα (POST) του API. Η κάθε καταχώρηση αποτελείται από την εικόνα, την περιγραφή κειμένου και το γεωγραφικό πλάτος και μήκος της. Ο χρήστης δεν εμπλέκεται με το κομμάτι της αλληλεπίδρασης με την βάση απευθείας, η εφαρμογή απλά επικοινωνεί με το server και αυτό αναλαμβάνει τις υπόλοιπες ενέργειες.

C.3.2 Server

Το πίσω μέρος της πλατφόρμας είναι το πιο σημαντικό. Κατασκευάστηκε με την χρήση του Node.js με το framework Express.js. Αυτό το κομμάτι του συστήματος είναι υπεύθυνο για την διαχείριση της λογικής του και την μεταφορά των δεδομένων που υποβάλλονται από τους χρήστες. Η δουλειά του είναι να λαμβάνει όλα τα αιτήματα από την κινητή εφαρμογή και την διαδικτυακή ιστοσελίδα και να απαντάει αντίστοιχα σε κάθε ένα.

To server αποτελείται από ένα σύνολο τελικών σημείων API. Για παράδειγμα:

- Όταν οι χρήστες καταχωρούν μία αναφορά, η εφαρμογή στέλνει ένα αίτημα “POST” στο σημείο “/api/report”.
- Όταν η ιστοσελίδα χρειάζεται να ανακτήσει όλα τα δεδομένα, στέλνει ένα αίτημα ανάκτησης “GET” στο αντίστοιχο “/api/report” σημείο.

Με την χρήση του Express είναι εύκολη η οργάνωση και σύνδεση αυτών των διαδρομών. Ακόμα, ξεκλειδώνει την πρόσβαση σε δικά του εργαλεία όπως το Multer και το CORS. Το Multer χρησιμοποιείται για την διαχείριση των φωτογραφιών που στέλνονται από την εφαρμογή, οι οποίες μετά μεταφέρονται στην βάση. Το CORS χρησιμοποιείται ώστε οι δύο διεπαφές του συστήματος να μπορούν να επικοινωνούν με το backend παρόλο που τρέχουν ξεχωριστά μεταξύ τους.

Η ασφάλεια γίνεται μέσω του Bcrypt όπου κρυπτογραφεί τους κωδικούς πρόσβασης των χρήστων. Η κρυπτογράφηση γίνεται μέσω κατακερματισμού με τυχαία σειρά χαρακτήρων και διασφαλίζει πως οι τελικοί κωδικοί που θα αποθηκευτούν στην βάση δεδομένων δεν θα είναι αναγνώσιμοι.

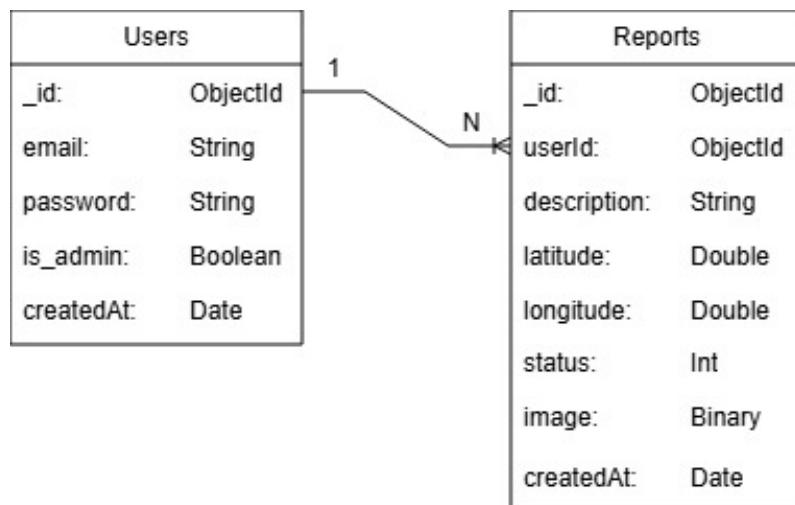
Όλα τα δεδομένα αποθηκεύονται στην cloud βάση MongoDB Atlas. To MongoDB βασίζεται πάνω σε αρχεία οπότε μπορεί να αποθηκεύσει δεδομένα σε διάφορες δομές. Αυτό

κάνει την αποθήκευση των αναφορών πιο εύκολη αφού συμπεριλαμβάνουν πολλά διαφορετικά πεδία.

C.3.3 Βάση Δεδομένων

Η βάση δεδομένων υλοποιείται στην MongoDB Atlas με δύο κύριες συλλογές, τους users και τα reports. Η συλλογή users αποθηκεύει στοιχεία λογαριασμού όπως email, κωδικό, τιμή που προσδιορίζει αν ο χρήστης είναι διαχειριστής και το μοναδικό του id. Η συλλογή reports καταγράφει κάθε αναφορά με ένα id αυτής, το id του χρήστη που την δημιούργησε, μία περιγραφή, την εικόνα, τα γεωγραφικά στοιχεία και την κατάσταση της ως ακέραιο με τιμές 0, 1 και 2. Η σχέση μεταξύ των δύο συλλογών είναι 1:N (ένας χρήστης : πολλές αναφορές) και η σύγκριση των id των χρηστών όπως φαίνεται και στο σχήμα C.2, ελέγχεται στο επίπεδο του backend, ώστε να διασφαλίζεται ότι κάθε αναφορά αντιστοιχίζεται σε υπαρχτό χρήστη.

Το σχήμα παραμένει απλό για γρήγορες εγγραφές και άμεση απεικόνιση στον χάρτη. Η επιλογή ξεχωριστών πεδίων πλάτους και μήκους απλοποιεί την καταχώριση από την κινητή εφαρμογή, ενώ η αποθήκευση εικόνας ως base64 διευκολύνει την ενιαία ροή αποστολής της μέσω του API.



Σχήμα C.2: Συλλογές users και reports της Βάσης Δεδομένων

C.3.4 Ιστοσελίδα

Το τρίτο μέρος της πλατφόρμας είναι η ιστοσελίδα του δήμου. Η διεπαφή αυτή είναι δημοσίως προσβάσιμη και εξυπηρετεί και τις δύο ομάδες χρηστών. Οι πολίτες μπορούν να την χρησιμοποιήσουν για την προβολή των ενεργών καταχωρήσεων πάνω σε χάρτη. Έτσι, οι ίδιοι παραμένουν ενημερωμένοι για τα υπάρχοντα προβλήματα στην πόλη τους και μπορούν να βλέπουν αν έχουν ήδη καταχωρηθεί συγκεκριμένες αναφορές χωρίς την χρήση της εφαρμογής. Στην άλλη πλευρά, οι υπόλληγοι του δήμου έχουν πρόσβαση στην ίδια διεπαφή. Εκτός την προβολή των καταχωρήσεων μπορούν να ενημερώσουν την κατάσταση προόδου και να τις σβήσουν όταν είναι πλέον φτιαγμένα μέσω μία σελίδας διαχειριστή.

Η ιστοσελίδα είναι φτιαγμένη χρησιμοποιώντας HTML, CSS και JavaScript. Ο χάρτης υλοποιήθηκε με την χρήση της βιβλιοθήκης Leaflet για διάδραση. Ενώ φορτώνει η σελίδα, στέλνεται απευθείας ένα αίτημα στο server για την ανάκτηση των δεδομένων από τις καταχωρήσεις. Μόλις λάβει αυτές τις πληροφορίες, το Leaflet εμφανίζει κάθε αναφορά ως ένα σημείο πάνω στον χάρτη βασισμένο στις συντεταγμένες της.

Κάθε σημείο περιλαμβάνει την εικόνα, το κείμενο και την κατάσταση της αντίστοιχης αναφοράς όπου εμφανίζονται πατώντας πάνω του. Επίσης, οι δείκτες έχουν διαφορετικό χρώμα αναλόγως την πρόσδοτο του με κόκκινο να σημαίνει πως το πρόβλημα είναι νέο και δεν έχει γίνει κάποια ενέργεια πάνω του ακόμα, πορτοκαλί για να δείξει ότι είναι σε εξέλιξη η διαδικασία επίλυσης του και πράσινο όταν έχει πλέον διορθωθεί. Η επεξεργασία της κατάστασης των αναφορών γίνεται μέσω αιτήματος διόρθωσης (PATCH) στο API και αντίστοιχα αυτές σβήνονται μέσω αιτήματος διαγραφής (DELETE).

C.3.5 Τελικό Αποτέλεσμα

Κάθε κομμάτι του συστήματος δουλεύει ανεξάρτητα με το άλλο αλλά όλα επικοινωνούν μεταξύ τους ομαλά μέσω του API. Η εφαρμογή για κινητά στέλνει δεδομένα, το backend τα στέλνει για αποθήκευση στην βάση δεδομένων και τα επεξεργάζεται, και η ιστοσελίδα τα οπικοποιεί. Χρησιμοποιώντας αυτή την προσέγγιση, το αποτέλεσμα αφήνει ελεύθερο κάθε μέρος να εξελίσσεται μόνο του. Για παράδειγμα, η εφαρμογή μπορεί να ενημερωθεί ώστε να υποστηρίζει και iOS στο μέλλον χωρίς να επηρεάσει τις υπόλοιπες λειτουργίες. Άρα, κάθε κομμάτι επικεντρώνεται πάνω σε συγκεκριμένες ανάγκες χωρίς να βρίσκεται εμπλεκόμενο με τα υπόλοιπα ή να δημιουργεί συγχρούσεις μεταξύ τους.

C.4 Διαγράμματα Χρήσης και Ροής

Για να γίνει καλύτερα κατανοητό πως τα διαφορετικά μέρη της πλατφόρμας αλληλεπιδρούν το ένα με το άλλο και πως οι χρήστες χρησιμοποιούν το σύστημα, δύο είδη διαγράμματων έχουν δημιουργηθεί, τα διαγράμματα περίπτωσης χρήσης και ροής του συστήματος. Αυτά βοηθούν στην προβολή των διαφορετικών ρόλων στο σύστημα και τα βήματα όπου ακολουθούνται κατά την χρήση κάθε συστήματός.

C.4.1 Διάγραμμα Περίπτωσης Χρήσης

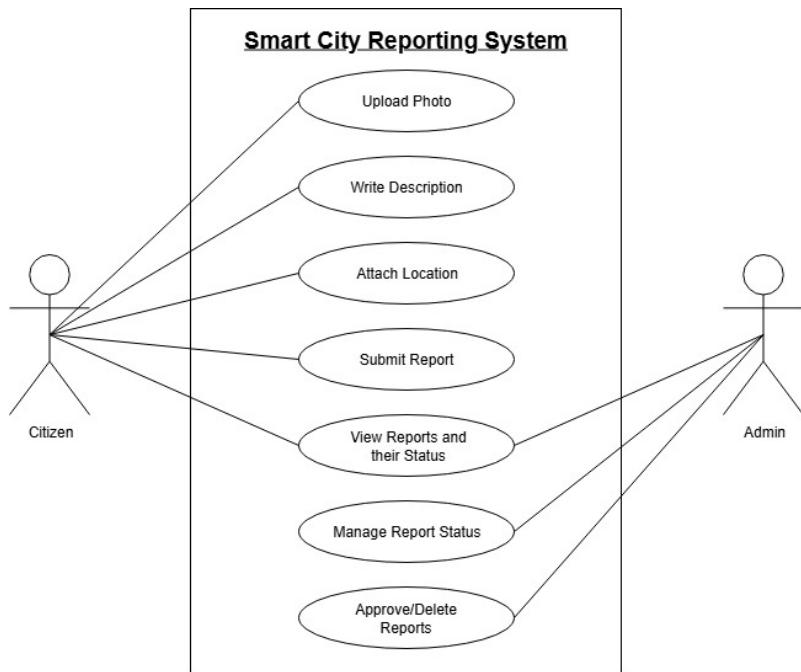
Το σύστημα αποτελείται από δύο ομάδες χρηστών, τους πολίτες και τις αρχές. Οι πολίτες είναι υπεύθυνοι για τον εντοπισμό προβλημάτων στην πόλη. Η αλληλεπίδραση τους γίνεται μέσω της κινητής εφαρμογής. Αυτοί μπορούν να:

- Βγάλουν φωτογραφία το πρόβλημα
- Γράψουν μία σύντομη περιγραφή
- Καταγράψουν αυτόματα την τοποθεσία τους
- Καταχωρίσουν την αναφορά
- Βλέπουν τις καταστάσεις των αναφορών τους

Από την άλλη, οι υπάλληλοι μέσα από την ιστοσελίδα επιβλέπουν και διαχειρίζονται τις αναφορές. Οι χρήσεις που έχουν είναι να:

- Βλέπουν όλες τις αναφορές
- Ενημερώσουν την κατάσταση μίας αναφοράς
- Διαγράψουν αναφορές που έχουν ολοκληρωθεί ή είναι λάθος

Όπως φαίνεται και στο Διάγραμμα Περίπτωσης Χρήσης (Σχήμα C.3), κάθε πλευρά είναι συνδεδεμένη στις ενέργειες που επιτρέπονται να εκτελέσουν. Αυτό δείχνει πως έχουν χωριστεί οι άδειες και πόσο διαφορετικές είναι οι δυνατότητες όπου έχουν οριστεί σε κάθε είδος χρήστη.

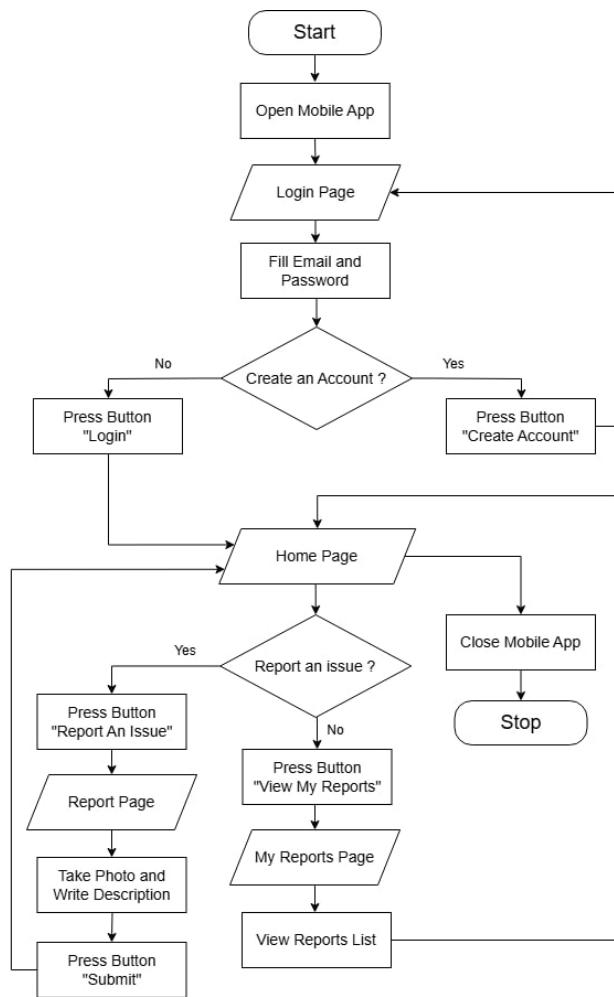


Σχήμα C.3: Διάγραμμα Περίπτωσης Χρήσης

C.4.2 Διαγράμματα Ροής

Διάγραμμα Εφαρμογής

Το διάγραμμα ροής της κινητής εφαρμογής (Σχήμα C.4), δείχνει την πλήρη διαδικασία που θα ακολουθήσει ένας κάτοικος κατά την χρήση της. Ανοίγοντας την εφαρμογή, εμφανίζεται μία φόρμα σύνδεσης του χρήστη. Αν ο χρήστης δεν έχει ένα λογαριασμό, μπορούν να τον δημιουργήσουν απευθείας στην ίδια σελίδα. Αφού δημιουργήσει ή αν έχει ήδη ένα λογαριασμό, τότε συνδέεται κανονικά και μεταφέρεται στην αρχική σελίδα. Σε αυτή, του παρέχονται δύο βασικές επιλογές, να αναφέρει ένα πρόβλημα ή να δει τις αναφορές του. Αν ο χρήστης διαλέξει να αναφέρει ένα πρόβλημα, κατευθύνεται στην σελίδα αναφοράς, όπου μπορεί να βγάλει μία φωτογραφία, να γράψει μία περιγραφή και να υποβάλει το θέμα στο σύστημα. Αντίθετα, διαλέγοντας το κουμπί “View My Reports”, ανοίγει η σελίδα των δηλώσεων του ή και όλων των υπαρχόντων αναφορών. Αυτή η ροή, τελειώνει με τον χρήστη να γυρνάει στην αρχική σελίδα και να κλείνει την εφαρμογή.

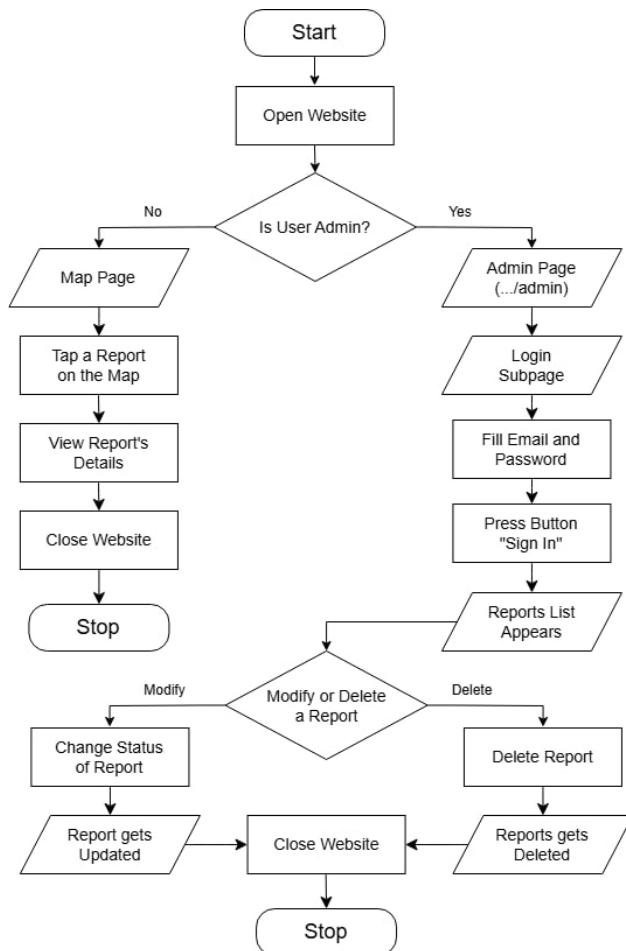


Σχήμα C.4: Διάγραμμα Ροής Εφαρμογής

Διάγραμμα Ιστοσελίδας

Για την ιστοσελίδα, υπάρχουν δύο ρόλοι χρήστη μέσα στο διάγραμμα ροής (Σχήμα C.5), ο απλός χρήστης και ο διαχειριστής, οι οποίοι έχουν αντίστοιχα την δικιά τους διαφορετική αλληλεπίδραση σε αυτή. Οι απλοί επισκέπτες εισέρχονται στην ανοιχτή σελίδα του χάρτη, όπου μπορούν να διαλέξουν μία αναφορά και να δουν τις πληροφορίες της. Από την άλλη, οι διαχειριστές εισέρχονται στην κλειστή σελίδα διαχειριστή, στην οποία θα πρέπει να συνδεθούν με τον λογαριασμό τους. Αφού γίνει η σύνδεση, εμφανίζεται ο λίστα ελέγχου που δείχνει όλες τις ενεργές καταχωρήσεις. Από εκεί, μπορούν είτε να ενημερώσουν την κατάσταση μίας αναφοράς ή να την διαγράψουν τελείως. Η λίστα μετά διαμορφώνεται αντίστοιχα με

την επιλογή του. Η ροή και των δύο φτάνει στο τέλος της όταν αυτοί κλείσουν την ιστοσελίδα.



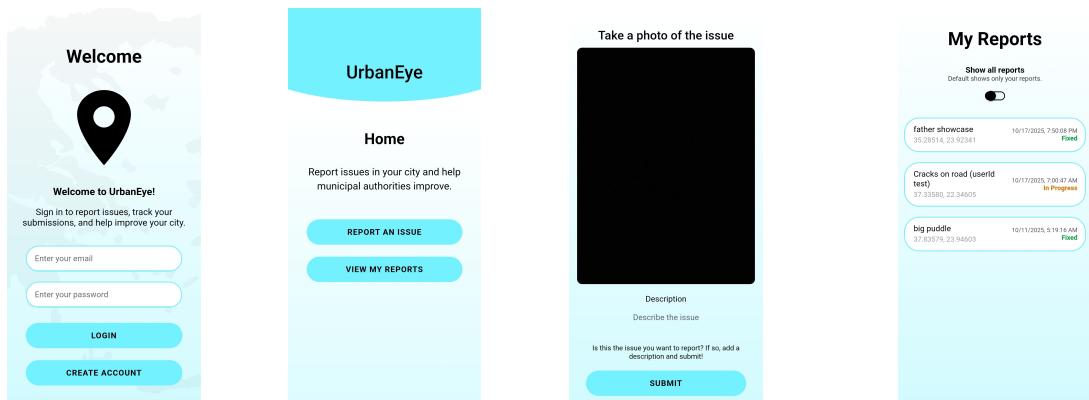
Σχήμα C.5: Διάγραμμα Ροής Ιστοσελίδας

C.5 Σχεδιασμός UI/UX

C.5.1 Σχεδίαση Διεπαφής

Η σχεδίαση της διεπαφής της πλατφόρμας είναι βασισμένη στην απλότητα και σαφήνεια σε όλα τα κομμάτια της. Αφού ο κύριος στόχος του συστήματος είναι να αφήνει τους κατοίκους να αναφέρουν θέματα της πόλης τους εύκολα και τους υπαλλήλους του δήμου να τα διαχειρίζονται αποδοτικά, το σχέδιο έπρεπε να παραμείνει ευθύ και λειτουργικό. Κάθε σελίδα χτίστηκε ώστε να είναι κατανοητή και μινιμαλιστική για την ολοκλήρωση της λειτουργίας της.

Κατά την όλη διάρκεια, ακολουθήθηκε μία λογική προτεραιότητας στα κινητά επειδή η εφαρμογή είναι το βασικό εργαλείο που θα χρησιμοποιηθεί περισσότερο από όλα. Η διεπαφή της εστιάζει στην απευθείας πρόσβαση στις κύριες ενέργειες, οι οποίες είναι η σύνδεση χρήστη, η δημιουργία αναφοράς και η προβολή της. Η σελίδα σύνδεσης επιτρέπει στον χρήστη να συνδεθεί με τον λογαριασμό του γρήγορα, ξεκλειδώνοντας του τις υπόλοιπες σελίδες. Η σελίδα αναφοράς παρέχει ένα απλό τρόπο για τον χρήστη να καταχωρίσει ένα πρόβλημα χρησιμοποιώντας την κάμερα και το GPS, και η σελίδα της λίστας των καταχωρήσεων αναφέρει όλες τις δηλώσεις του χρήστη αυτού ή και όλες τις διαθέσιμες αν το θελήσει ο ίδιος. Για να εισαχθεί ένας χρήστης σε αυτές τις δύο σελίδες, υπάρχει μία ενδιάμεση σελίδα όπου παρέχει την είσοδο μέσω της χρήσης κουμπιών με την σχετική ετικέτα. Αυτή η διαδικασία σχεδιάστηκε ώστε η αποστολή μίας αναφοράς να γίνεται σε σύντομο χρονικό διάστημα χωρίς να καθυστερεί τον χρήστη.



(a) Σελίδα Σύνδεσης

(b) Αρχική Σελίδα

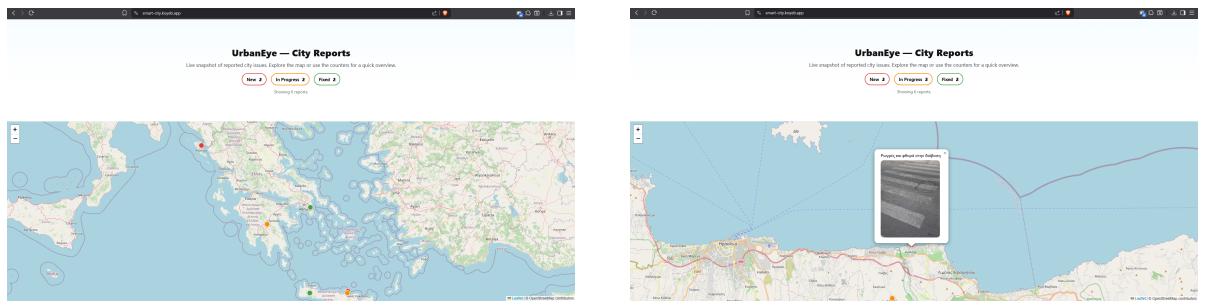
(c) Σελίδα Αναφοράς

(d) Σελίδα Καταχωρήσεων

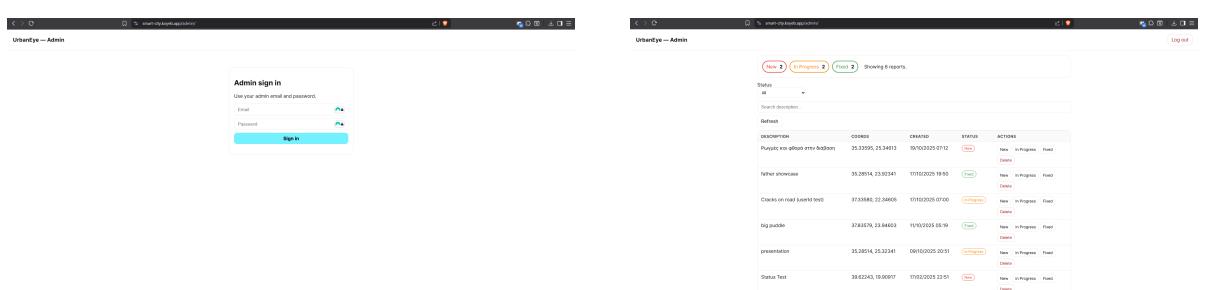
Σχήμα C.6: Σελίδες της κινητής εφαρμογής

Η ιστοσελίδα, όπου χρησιμοποιείται κυρίως από τους υπαλλήλους, εστιάζει κυρίως στην οπτικοποίηση των δεδομένων και την επεξεργασία τους. Ο χάρτης χρησιμεύει ως ένα μέσο για να γίνει αυτή η οπτικοποίηση, όπου τα σημάδια πάνω του εκπροσωπούν τις αναφορές στις διάφορες καταστάσεις. Κάθε κατάσταση έχει το δικό της χρώμα, βοηθώντας τους διαχειριστές στην σελίδα διαχειριστή, αλλά και τους χρήστες, να μπορούν να τα εντοπίσουν άμεσα και εύκολα. Ακόμα, στην βασική σελίδα του χάρτη και στην σελίδα διαχειριστή υπάρχουν μετρητές κάθε κατάστασης αναφοράς δίνοντας μία γρήγορη επισκόπηση χωρίς την

ανάγκη επιπλέον πλοιηγήσεων. Η ίδια η σελίδα διαχειριστή, είναι δομημένη έτσι ώστε να εμφανίζει κάθε αναφορά ξεχωριστά σε μορφή λίστας και του παρέχει επιλογές διαχείρισης ή διαγραφής της.



Σχήμα C.7: Σελίδα Χάρτη της Ιστοσελίδας



Σχήμα C.8: Σελίδα Διαχειριστή της Ιστοσελίδας

Η συνολική οπτική ταυτότητα είναι μοντέρνα και μινιμαλιστική. Η παλέτα χρωμάτων αποτελείται κυρίως από γαλάζιο και άσπρο διατηρώντας μία καθαρή και ουδέτερη εμφάνιση στην εφαρμογή, με τα γράμματα να είναι μαύρα για αντίθεση. Τα κουμπιά είναι στρογγυλεμένα, οι σκιές είναι απαλές κρατώντας το μοντέρνο ύφος της. Τα κείμενα και οι ετικέτες είναι σύντομα και κατανοητά για να παραμείνει η διεπαφή προσβάσιμη σε όλους τους χρήστες. Τέλος, χρησιμοποιήθηκαν σταθερές αποστάσεις και ευθυγραμμίσεις ώστε να διασφαλιστεί η αναγνωστικότητα και να αποφευχθεί η οπτική πολυπλοκότητα κατά την χρήση.

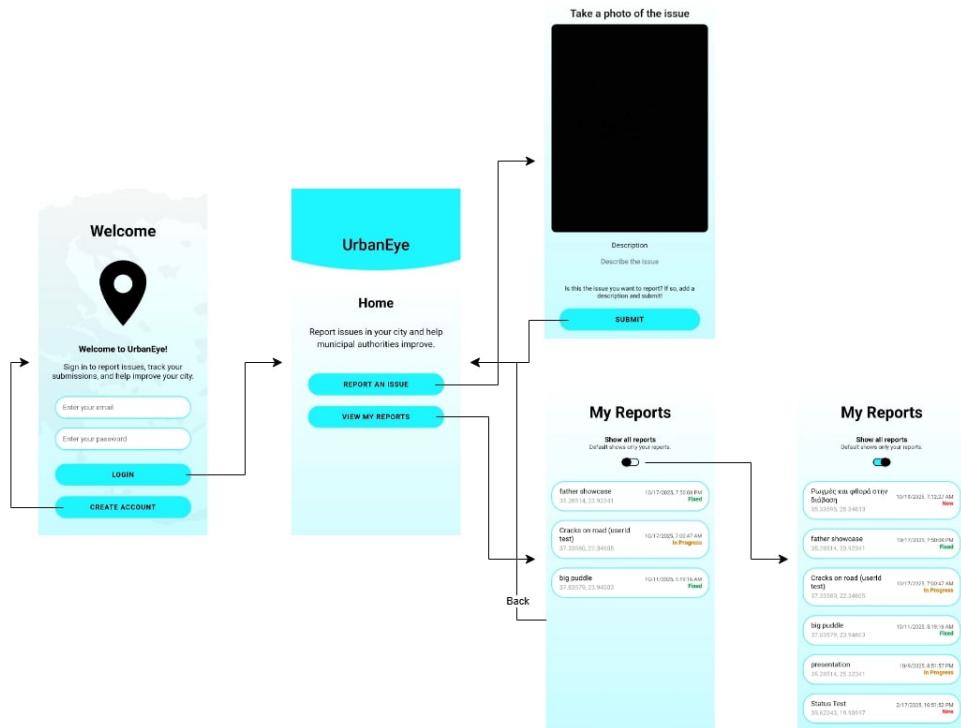
C.5.2 Εμπειρία Χρήστη και Ροή

Ο σχεδιασμός της εμπειρίας χρήστη (UX) είναι στοχευμένος στο να κάνει τις αλληλεπιδράσεις όσο πιο το δυνατόν πιο διαισθητικές και αυτονόητες. Ο χρήστης θα πρέπει να βρεθεί σε θέση να καταλάβει πως να λειτουργεί την εφαρμογή απευθείας χωρίς την ανάγκη προηγούμενων οδηγιών. Για παράδειγμα, κατά την δημιουργία μίας νέας αναφοράς, η εφαρμογή αυτόματα ανιχνεύει την τοποθεσία του χρήστη μέσω του GPS, μειώνοντας την ανάγκη για χειροκίνητης εισαγωγής από αυτόν. Μετά την υποβολή, ένα μήνυμα επιβεβαίωσης εμφανίζεται για να διαβεβαιώσει τον χρήστη πως η ενέργεια εκτελέστηκε επιτυχώς και τον γυρνάει αυτόματα στην σελίδα του βασικού μενού. Ακόμα, προστέθηκαν δείκτες φόρτωσης και σύντομα μηνύματα σφαλμάτων ώστε να δέχεται άμεση ανατροφοδότηση κατά την διάρκεια των λειτουργιών. Η ιστοσελίδα ακολουθεί, επίσης, παρόμοια απλά μοτίβα αλληλεπίδρασης ώστε να παραμείνει η εμπειρία ομοιόμορφη και για τις δύο ομάδες χρηστών, τους πολίτες και τους διαχειριστές.

Η πλοήγηση μεταξύ σελίδων της εφαρμογής έγινε με το σύστημα δρομολόγησης της Ionic, επιτρέποντας για ομαλές μεταβάσεις και διατηρώντας την συμπεριφορά της προς τα πίσω κίνησης στις συσκευές. Αυτό διασφαλίζει συνοχή μεταξύ Android και άλλων πλατφορμών.

Συνολικά, η UI/UX σχεδίαση επιτυγχάνει μία ισορροπία μεταξύ χρηστικότητας και απλότητας. Εστιάζοντας στον μινιμαλισμό και σε σταθερά οπτικά στοιχεία, η πλατφόρμα παρέχει μία ευχάριστη εμπειρία και εμφάνιση και για τους πολίτες και για τους υπαλλήλους του δήμου.

Το ακόλουθο σχήμα C.9 απεικονίζει την κύρια δομή πλοήγησης της εφαρμογής μέσω ενός wireflow διαγράμματος, δείχνοντας την σύνδεση μεταξύ των σελίδων και την λογική ροή των ενεργειών του χρήστη.



Σχήμα C.9: Wireflow Διεπαφής Χρήστη Εφαρμογής

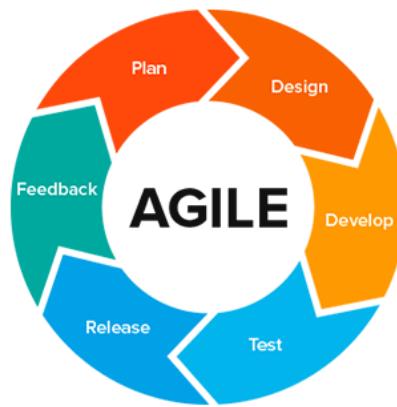
C.6 Στρατηγικές Ανάπτυξης

Η ανάπτυξη της πλατφόρμας δεν βασίστηκε πάνω σε μία μοναδική μεθοδολογία. Κάθε μέρος ακολούθησε την στρατηγική όπου ταίριαζε στο ρόλο του και τις τεχνικές ανάγκες του καλύτερα. Η λογική της κινητής εφαρμογής, της ιστοσελίδας και του backend υλοποιήθηκαν χρησιμοποιώντας μία Agile προσέγγιση, όπου τα χαρακτηριστικά τους, όπως σελίδες και λειτουργίες, δημιουργήθηκαν και δοκιμάστηκαν σε μικρά βήματα και βελτιώθηκαν σταδιακά με βάση τα αποτελέσματα. Αντίθετα, το σχήμα της βάσης δεδομένων ακολούθησε μία πιο δομημένη προσέγγιση όπως αυτή της στρατηγικής Καταρράκτη. Η επιλογή αυτή έγινε επειδή η σχεδίαση της βάσης έπρεπε να γίνει με σαφήνεια από την αρχή και να παραμείνει σταθερή κατά την διάρκεια του έργου.

C.6.1 Agile

Η στρατηγική που ακολουθήθηκε κατά την ανάπτυξη της λογικής των διεπαφών ιστού και κινητού καθώς και του backend, είναι αυτή της Agile. Η κύρια ιδέα πίσω από την Agile

είναι η λειτουργία σε σύντομους κύκλους ανάπτυξης, όπου μικρά κομμάτια του συστήματος δημιουργούνται, δοκιμάζονται και βελτιώνονται πριν την μετάβαση στο επόμενο βήμα. Αυτό ταιριάζει λογικά με την πλατφόρμα, αφού επέτρεπε για κάθε νέα σελίδα, λειτουργία καθώς και τελικά σημεία στο API να επιβεβαιωθεί η σωστή λειτουργικότητα τους στην πράξη και να υλοποιηθούν πλήρως πριν την ένταξη επόμενων.



Σχήμα C.10: Στρατηγική Προσέγγισης Agile

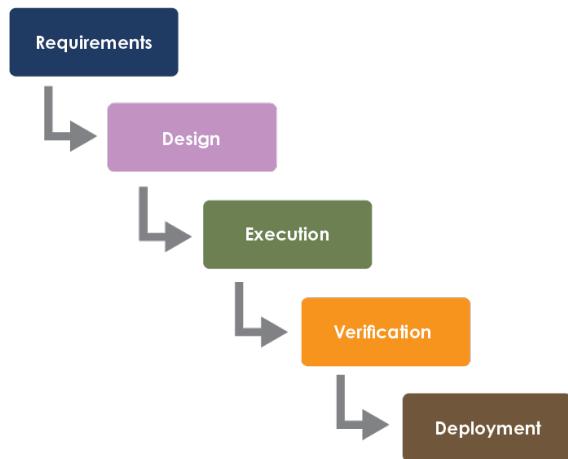
Παρόλο που υπήρχε ένα ξεκάθαρο πλάνο από την αρχή σχετικά με την αρχιτεκτονική του συστήματος και τις κύριες λειτουργίες του, η υλοποίηση ολοκληρώθηκε βήμα βήμα σε επαναλαμβανόμενους κύκλους. Για παράδειγμα, η σύνδεση χρήστη, ενσωμάτωση της κάμερας, η απεικόνιση του χάρτη και οι λειτουργίες διαχειριστή προστέθηκαν ένα την φορά και δοκιμάστηκαν απευθείας. Όποτε εμφανιζόταν κάποιο θέμα κατά την ανάπτυξη, αυτό διορθωνόταν απευθείας στον ίδιο κύκλο χωρίς να επηρεάζει το συνολικό πλάνο. Αυτή η προσέγγιση βοήθησε ώστε να διατηρείται μία λειτουργική έκδοση της πλατφόρμας συνεχώς.

Το backend ακολούθησε την ίδια επαναληπτική λογική. Κάθε τελικό σημείο στο API αναπτύχθηκε ως ένα ανεξάρτητο κομμάτι και δοκιμάστηκε με την εφαρμογή και ιστοσελίδα στον ίδιο κύκλο. Αυτό κρατούσε την όλη διαδικασία οργανωμένη και διευκόλυνε την διόρθωση των σφαλμάτων, καθώς οποιοδήποτε πρόβλημα εμφανιζόταν από νωρίς.

C.6.2 Waterfall

Το σχέδιο της βάσης δεδομένων ακολούθησε μία διαφορετική στρατηγική, αυτή του Καταρράκτη (Waterfall), η οποία ταίριαζε καλύτερα στις ανάγκες της βάσης. Η μεθοδολογία

Καταρράκτη ακολουθεί μία γραμμική ροή ανάπτυξης. Η λογική της είναι η ύπαρξη ενός ξεκάθαρου σχεδίου και η σταδιακή πλήρης υλοποίηση του. Σε αντίθεση με την Agile, αυτή δίνει έμφαση στην σταθερότητα και στον ορισμό της δομής πριν την ανάπτυξη.



Σχήμα C.11: Στρατηγική Προσέγγισης Waterfall

Αφού, η εφαρμογή και η ιστοσελίδα εξαρτώνται από την ίδια δομή δεδομένων, ήταν σημαντικό για τις συλλογές της βάσης και τα πεδία τους να είναι ξεκάθαρα ορισμένα από την αρχή. Οι δύο κύριες συλλογές, “users” και “reports”, είχαν καθοριστεί από το αρχικό στάδιο και παρέμειναν σταθερές σε όλη την διάρκεια της ανάπτυξης.

Η σταθερή αυτή δομή συνέβαλε στην αποφυγή προβλημάτων συμβατότητας και επέτρεψε στο υπόλοιπο σύστημα να εξελιχθεί χωρίς συνεχείς αλλαγές στο μοντέλο δεδομένων. Η προσέγγιση Καταρράκτη ήταν κατάλληλη για αυτή την περίπτωση, επειδή η ιδέα της βασίζεται στην ολοκλήρωση κάθε φάσης χωρίς να απαιτείται να επανασχεδιαστεί ο αρχικός πυρήνας.

Κεφάλαιο D

Ανάπτυξη Εφαρμογής

D.1 Επισκόπηση Υλοποίησης

Σε αυτό το κεφάλαιο παρουσιάζεται η εφαρμογή της όλης πλατφόρμας προχωρώντας από το κομμάτι του σχεδιασμού προς το πως υλοποιείται το σύστημα. Η υλοποίηση της πλατφόρμας έξυπνης πόλης ακολούθησε μία προσέγγιση όπου κάθισε μέρος της αναπτύσσεται ανεξάρτητα αλλά συνδέεται με τα υπόλοιπα μέσω ενός κοινού backend και μίας κοινής βάσης δεδομένων σε cloud. Το backend δημιουργήθηκε με Node.js και Express χρησιμοποιώντας RESTful APIs για την αναφορά, ανάκτηση και επεξεργασία ή διαγραφή των καταχωρήσεων. Στο κομμάτι της βάσης χρησιμοποιήθηκε η cloud υπηρεσία Atlas του MongoDB. Η κινητή εφαρμογή έγινε μέσω Ionic React και TypeScript, με συνδυασμό με το Capacitor για την πρόσβαση σε λειτουργίες του κινητού όπως κάμερα και τοποθεσία. Η διαδικτυακή ιστοσελίδα αναπτύχθηκε με την χρήση JavaScript και HTML σε συνδυασμό με το Leaflet. Η πλατφόρμα δοκιμάστηκε και κατασκευάστηκε για Android χρησιμοποιώντας το εργαλείο Android Studio ώστε να διασφαλιστεί πως όλες οι λειτουργίες δουλεύουν αξιόπιστα. Οι επόμενες ενότητες αναλύουν τα επιμέρους μέρη λεπτομερώς με κομμάτια κώδικα και στιγμιοτύπων ουθόνης για την καλύτερη κατανόηση προς την λειτουργία τους και την διαδικασία της υλοποίησης.

D.2 Backend

To backend είναι το βασικό στοιχείο του συστήματος. Αυτό δουλεύει ενδιάμεσα και ενώνει όλα τα κομμάτια με την βάση δεδομένων. Οι βασικές του λειτουργίες είναι η διαχείριση, μεταμόρφωση και αποθήκευση των δεδομένων. Ακόμα, το ίδιο διαχειρίζεται το ανέβασμα των φωτογραφιών, την ταυτοποίηση και τον χειρισμό των σφαλμάτων. Έτσι διασφαλίζει πως η επικοινωνία μεταξύ ολόκληρης της πλατφόρμας γίνεται σωστά.

D.2.1 Αρχικοποίηση

Το πρώτο κομμάτι του κώδικα είναι η αρχικοποίηση και ρύθμιση των απαραίτητων πακέτων. Αρχικά, ενεργοποιείται το CORS ώστε να επιτρέπεται στις διεπαφές του συστήματος να έχουν την δυνατότητα να στέλνουν αιτήματα από διαφορετικές προελεύσεις, και δηλώνονται οι μεθόδοι όπου θα είναι επιτρεπτοί. Στην συνέχεια, ορίζεται ένα μεγαλύτερο όριο για την διαχείριση των JSON δεδομένων της εφαρμογής όπου είναι απαραίτητο για την αποστολή εικόνων.

```

1 const express = require('express');
2 const { MongoClient, ServerApiVersion } = require('mongodb');
3 const cors = require('cors');
4 const dotenv = require('dotenv');
5 const bcrypt = require('bcryptjs');
6 const multer = require( multer );
7 const path = require( path );
8
9 dotenv.config();
10
11 const app = express();
12
13 app.use(cors({
14   origin: '*',
15   methods: [ 'GET', 'POST', 'PUT', 'DELETE' ] ,
16   allowedHeaders: [ 'Content-Type', 'Authorization' ],
17 }));
18
19 app.use(express.json({ limit: 50mb }));

```

Αυτό το τμήμα εξασφαλίζει ότι ο διακομιστής θα είναι έτοιμος για να λάβει HTTP αιτήματα και πως θα μπορεί να επικοινωνεί χωρίς κάποιες περιορίσεις κατά την περιήγηση. Τέλος, δηλώνεται το dotenv όπου δίνει την δυνατότητα σε πληροφορίες, όπως το string σύνδεσης του MongoDB, να παραμείνουν χρυφές εκτός κώδικα αντί να γράφονται απευθείας σε αυτόν.

D.2.2 Σύνδεση με την Βάση Δεδομένων

Στην συνέχεια, ακολουθεί η σύνδεση στο MongoDB Atlas χρησιμοποιώντας την μεταβλητή μέσω του “.env” αρχείου. Αυτή επιβεβαιώνεται μέσω εντολής ping και με κείμενο στην κονσόλα. Σε περίπτωση που δεν επιβεβαιωθεί αυτή η εντολή και δεν υπάρχει σύνδεση, δημιουργείται ένα σφάλμα στο σύστημα το οποίο εντοπίζεται και εμφανίζεται σχετικό μήνυμα. Αφού γίνει η επαλήθευση, ο κώδικας συνδέεται με την απομακρυσμένη βάση δεδομένων και ορίζει την “smart_city_db” ως βασική βάση της πλατφόρμας καθώς και φορτώνει τις δύο σύλλογές της. Στην συλλογή “reports” βρίσκονται οι πληροφορίες για τις καταχωρήσεις και στην συλλογή “users” βρίσκονται αντίστοιχα τα στοιχεία των λογαριασμών χρηστών.

```

1 const uri = process.env.MONG_CONN;
2 const client = new MongoClient(uri, {
3   serverApi: {
4     version: ServerApiVersion.v1,
5     strict: true,
6     deprecationErrors: true,
7   }
8 });
9
10 async function connectToDB() {
11   try {
12     await client.connect();
13     const db = client.db( admin );
14     const result = await db.command({ ping: 1 });
15     console.log( Connected to MongoDB using MongoClient );
16     console.log( Ping successful: , result);
17   } catch (err) {
18     console.error( Failed to connect to MongoDB , err);
19   }
}

```

```

20 }
21
22 connectToDB();
23
24 const db = client.db( smart_city_db );
25 const reportsCollection = db.collection( reports );
26 const usersCollection = db.collection( users );

```

D.2.3 Στατικά αρχεία

Σε αυτό το μέρος του κώδικα, υλοποιείται η χρήση στατικών αρχείων από το backend και δηλώνεται το frontend. Ο κώδικας δίνει εντολή στο Express να στείλει ότι αρχείο υπάρχει μέσα στο frontend στο αρχικό μονοπάτι της ιστοσελίδας. Αυτό βοηθάει αρκετά στην υποστήριξη απλής φιλοξενίας και γρήγορων δοκιμών κατά την ανάπτυξη.

```

1 app.use(express.static(path.join(__dirname, './frontend')));
2
3 app.get( / , (req, res) => {
4   res.sendFile(path.join(__dirname, './frontend', index.html));
5 });

```

D.2.4 Ρύθμιση Μεταμόρφωσης Αρχείων

Η μεταμόρφωση των φωτογραφιών από την κινητή εφαρμογή γίνεται μέσω του Multer. Αντί να αποθηκεύεται η φωτογραφία, αυτή εισέρχεται προσωρινά στην μνήμη και μετά στέλνεται απευθείας στην βάση δεδομένων ως δυαδικά δεδομένα. Αυτό απλοποιεί αρκετά το API και αποφεύγει την διαχείριση τοπικών αρχείων.

```

1 const storage = multer.memoryStorage();
2 const upload = multer({ storage });

```

D.2.5 Εγγραφή και Σύνδεση Χρηστών

Η διαχείριση των χρηστών γίνεται με δύο λειτουργίες στον κώδικα, μία για να εγγραφεί και μία για να συνδεθεί. Όταν ο χρήστης εγγράφεται, το email και ο κωδικός του ελέγχονται

και πριν αποθηκευτούν στην βάση, ο κωδικός κατακερματίζεται με ασφάλεια με την χρήση του bcrypt.

```

1 app.post('/api/register', async (req, res) => {
2   const { email, password } = req.body;
3
4   if (!email || !password) {
5     return res.status(400).json({ message: 'All fields are required' });
6   }
7   try {
8     const existingUser = await usersCollection.findOne({ email });
9     if (existingUser) {
10       return res.status(400).json({ message: 'Email is already registered' });
11     }
12
13     const saltRounds = 10;
14     const hashedPassword = await bcrypt.hash(password, saltRounds);
15
16     const newUser = { email, password: hashedPassword, createdAt: new Date() };
17     await usersCollection.insertOne(newUser);
18
19     res.status(201).json({ message: 'User registered successfully' });
20   } catch (error) {
21     res.status(500).json({ message: error.message });
22   }
23 });

```

Στο κομμάτι της σύνδεσης γίνεται αναζήτηση στην συλλογή χρηστών της βάσης για το email. Σε περίπτωση που δεν υπάρχει αυτό στο σύστημα εκτυπώνεται πως δεν υπάρχει τέτοιος λογαριασμός. Αντίθετα, γίνεται αυτόματα σύγχριση μέσω του bcrypt μεταξύ του καταχωριμένου κωδικού στην βάση και του κωδικού σύνδεσης. Αν ο κωδικός είναι έγκυρος ή διαφορετικός, ο χρήστης ενημερώνεται μέσω μηνύματος πως συνδέθηκε ή ότι έχει δώσει λανθασμένα στοιχεία αντίστοιχα.

```

1 app.post('/api/login', async (req, res) => {
2   const { email, password } = req.body;

```

```

3
4     if (!email || !password) {
5         return res.status(400).json({ message: 'All fields are required' });
6     }
7     try {
8         const user = await usersCollection.findOne({ email });
9         if (!user) {
10             return res.status(404).json({ message: 'User not found' });
11         }
12         const isPasswordValid = await bcrypt.compare(password, user.password);
13         if (!isPasswordValid) {
14             return res.status(401).json({ message: 'Invalid credentials' });
15         }
16         const safeUser = {
17             _id: user._id,
18             email: user.email,
19             is_admin: !!user.is_admin
20         };
21         return res.status(200).json({
22             message: 'Login successful',
23             user: safeUser,
24         });
25     } catch (error) {
26         res.status(500).json({ message: error.message });
27     }
28 });

```

D.2.6 Διαχείριση Αναφορών

Από την άλλη, υπάρχουν τέσσερις λειτουργίες διαχείρισης για τις αναφορές. Η πρώτη λειτουργία χρησιμοποιείται για την ανάκτηση όλων των αναφορών. Με την χρήση του GET όλες οι αναφορές καλούνται από την βάση δεδομένων και στην συνέχεια οι εικόνες τους μετατρέπονται από δυαδική μορφή σε base64 συμβολοσειρά ώστε να μπορέσουν να εμφανιστούν απευθείας στην ιστοσελίδα χωρίς την χρήση πραγματικής φωτογραφίας από ξεχωριστό server.

```

1 app.get('/api/reports', async (req, res) => {
2   try {
3     const reports = await reportsCollection.find().toArray();
4     const formattedReports = reports.map(report => ({
5       ...report,
6       image: report.image ? `data:image/jpeg;base64,${report.image.toString( base64
7 ))};
8
9     res.json(formattedReports);
10  } catch (error) {
11    res.status(500).json({ message: error.message });
12  }
13});

```

Η δεύτερη λειτουργία είναι για την αποστολή και αποθήκευση των αναφορών μέσω της χρήσης POST διαδρομής. Κάθε αναφορά αποτελείται από κείμενο περιγραφής, τις τιμές γεωγραφικού πλάτους και μήκους, και την εικόνα. Η ανεβασμένη εικόνα διαβάζεται από την μνήμη και αποστέλλεται στην βάση δεδομένων μαζί με τις υπόλοιπες πληροφορίες. Αυτές οι δύο συναρτήσεις οφείλονται για την υποβολή και προβολή των προβλημάτων μέσω της κινητής εφαρμογής και ιστοσελίδας αντίστοιχα.

```

1 app.post('/api/reports', upload.single( image ), async (req, res) => {
2   const { description, latitude, longitude, status = 1 } = req.body;
3
4   if (!description || !latitude || !longitude) {
5     return res.status(400).json({ message: 'All fields are required' });
6   }
7
8   try {
9     let imageBuffer = null;
10  if (req.file) {
11    imageBuffer = req.file.buffer;
12  }
13  console.log( Received Image Blob: , imageBuffer ? Photo Blob Received : No I
14
15  const newReport = {

```

```

16     description,
17     latitude: Number(latitude),
18     longitude: Number(longitude),
19     status: Number(status),
20     image: imageBuffer,
21     createdAt: new Date(),
22   };
23
24   await reportsCollection.insertOne(newReport);
25   res.status(201).json(newReport);
26 } catch (error) {
27   console.error('Error saving report: ', error);
28   res.status(500).json({ message: error.message });
29 }
30 });

```

Οι άλλες δύο συναρτήσεις αναφορών, αφορούν τους διαχειριστές όπου θα έχουν την ικανότητα να επεξεργάζονται τις καταχωρήσεις. Αρχικά, η βασική λειτουργία του διαχειριστή αφορά την κατάσταση μίας αναφοράς, αυτό γίνεται μέσω της χρήσης “PATCH” όπου επιτρέπει την ενημέρωση της χωρίς την τροποποίηση των υπόλοιπων πληροφοριών. Το τελικό αυτό σημείο, δέχεται το ID της αναφοράς μέσω URL και την τιμή της νέας κατάστασης μέσα στο σώμα αιτήματος. Πριν την εφαρμογή της ενημέρωσης, η τιμή ελέγχεται ότι ανήκει στα ήδη ορισμένα αποδεκτά νούμερα (0, 1 και 2). Αν η τιμή είναι σωστή, το σύστημα εντοπίζει αντίστοιχη καταχώρηση μέσα στην βάση και την τροποποιεί αντίστοιχα. Η αλλαγμένη πληροφορία γυρνάει πίσω και εμφανίζεται το αποτέλεσμα σωστά στην διεπαφή.

```

1 app.patch('/api/reports/:id/status', async (req, res) => {
2   const { id } = req.params;
3   const sNum = Number(req.body?.status);
4   if (![0, 1, 2].includes(sNum)) {
5     return res.status(400).json({ message: 'Invalid status. Use 0, 1, or 2.' });
6   }
7   try {
8     const _id = new ObjectId(id);
9     const result = await reportsCollection.findOneAndUpdate(
10       { _id },
11       { $set: { status: sNum, updatedAt: new Date() } },

```

```

12     { returnDocument: 'after' } // returns the updated doc
13   );
14   return res.status(200).json({ message: 'Status updated', report: result.value });
15 } catch (err) {
16   console.error('PATCH status error:', err);
17   return res.status(500).json({ message: err.message });
18 }
19 );

```

Η άλλη ενέργεια όπου διαιθέτει ένας διαχειριστής είναι η αφάίρεση μίας αναφοράς. Αντίστοιχα με το “PATCH”, η διαγραφή γίνεται μέσω του “DELETE”. Η διαδικασία αυτή δουλεύει παρόμοια, χρησιμοποιώντας το ID της αναφοράς και διαγράφοντας την αντίστοιχη καταχώρηση μέσα στην σωστή συλλογή. Αν δεν βρεθεί η συγκεκριμένη αναφορά, εμφανίζεται μήνυμα πως δεν υπάρχει στην βάση δεδομένων, αντίθετα επιστρέφεται ένα μήνυμα επιβεβαίωσης. Αυτή η συνάρτηση χρησιμοποιείται μόνο για την μόνιμη διαγραφή μίας άκυρης ή σε περίπτωση ίδιας διπλής αναφοράς.

```

1 app.delete('/api/reports/:id', async (req, res) => {
2   const { id } = req.params;
3   try {
4     const _id = new ObjectId(id);
5     const result = await reportsCollection.deleteOne({ _id });
6     if (result.deletedCount === 0) {
7       return res.status(404).json({ message: 'Report not found' });
8     }
9     return res.status(200).json({ message: 'Report deleted' });
10  } catch (err) {
11    console.error('DELETE report error:', err);
12    return res.status(500).json({ message: err.message });
13  }
14);

```

D.3 Ιστοσελίδα

Η ιστοσελίδα της πλατφόρμας είναι υπεύθυνη για την διεπαφή των διαχειριστών και την οπτικοποίηση των αναφορών. Η προβολή αυτή γίνεται μέσω ενός αλληλεπιδραστικό χάρτη

φτιαγμένος με την βιβλιοθήκη Leaflet όπου φαίνονται πληροφορίες όπως η τωρινή κατάσταση, η ακριβής τοποθεσία και η φωτογραφία του προβλήματος. Η ίδια επικοινωνεί απευθείας με το API στο backend για να πάρει τα δεδομένα και να τα εμφανίσει σε πραγματικό χρόνο.

D.3.1 Αρχικοποίηση και Δημιουργία του Χάρτη

Η βασική σελίδα της διαδικτυακής εφαρμογής αποτελείται από ένα στατικό HTML αρχείο όπου καθορίζει τον χάρτη χρησιμοποιώντας και συνδέοντας με το Leaflet μέσω CDN σύνδεσης. Η ετικέτα script φορτώνει την βιβλιοθήκη Leaflet όπου παρέχει όλες τις απαραίτητες συναρτήσεις και κλάσεις για την δημιουργία και διαχείριση των χαρτών. Με την χρήση του div με το ID του χάρτη ορίζεται η βασική περιοχή όπου αυτός θα προβάλλεται.

```

1 <link rel= "stylesheet" href= "https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
2 <script src= "https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
3
4 <div id= "map"></div>
5
6 <script>
7   const GREECE_BOUNDS = L.latLngBounds([35.5, 21.0], [40.0, 25.5]);
8
9   const map = L.map('map', { zoomControl: true });
10
11  L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
12    attribution: 'OpenStreetMap contributors',
13    maxZoom: 19
14  }).addTo(map);
15
16  map.fitBounds(GREECE_BOUNDS, { padding: [20, 20] });
17 </script>
```

Ακόμα, ορίζεται μέσω γεωγραφικών ορίων η περιοχή της Ελλάδας στο χάρτη, εξασφαλίζοντας την πλήρη εικόνα των αναφορών σε όλη την χώρα. Το επόμενο βήμα είναι η δημιουργία του χάρτη, το οποίο γίνεται καλώντας την συνάρτηση “L.map()” και παραχωρώντας το ID. Επιπλέον, με την χρήση του “L.tileLayer()” εμφανίζεται ένα νέο επίπεδο όπου οπτικοποιεί τους δρόμους πάνω στον χάρτη μέσω του ανοικτού κώδικα της OpenStreetMap. Με τον συνδυασμό όλων αυτών, ο χρήστης κατά την διάρκεια χρήσης θα έχει την δυνατότητα να περιφέρεται στον χάρτη και να ζουμάρει καθώς όλα φορτώνουν και αλληλεπιδρούν ομαλά.

D.3.2 Ορισμός Εικονιδίων

Κάθε αναφορά εμφανίζεται ως ένας χρωματιστός δείκτης όπου αντιπροσωπεύει την τωρινή κατάσταση της. Σε αυτό το μέρος, τρία διαφορετικά εικονίδια δημιουργούνται για να μπορούν να διαχριθούν τα προβλήματα ανάλογα με την κατάσταση τους. Η χρήση δεικτών με διαφορετικά χρώματα διευκολύνει αυτή την ιδιότητα έχοντας:

- Κόκκινο: ένα νέο πρόβλημα
- Πορτοκαλί: ένα πρόβλημα που επιδιορθώνεται
- Πράσινο: ένα πρόβλημα που διορθώθηκε

Η οπτικοποίηση των καταχωρήσεων γίνεται με την χρήση ελαφριών “div” εικονιδίων της Leaflet με συνδυασμό ενός χρώματος ανάλογα με την κατάσταση του. Η επεξεργασία των εικονιδίων αυτών πραγματοποιείται μέσα στην “L.divIcon” χρησιμοποιώντας την μεταβλητή χρώματος και προσαρμοσμένες τιμές μεγέθους.

```

1 const colorIcon = (color) => L.divIcon({
2   className: 'status-pin',
3   html: `<span style= background:${color} ></span>`,
4   iconSize: [18, 18],
5   iconAnchor: [9, 18]
6 });
7
8 const ICONS = {
9   1: colorIcon('#e53935'), // New
10  2: colorIcon('#fb8c00'), // In progress
11  0: colorIcon('#43a047'), // Fixed
12 };

```

D.3.3 Ανάκτηση Αναφορών και Τοποθέτηση Δεικτών

Μετά την ρύθμιση του χάρτη, η ιστοσελίδα χρειάζεται να λάβει όλες τις καταχωρήσεις από την βάση, το οποίο γίνεται με ένα απλό αίτημα fetch. Η απάντηση του αιτήματος παρέχει δεδομένα σε μορφή JSON και εξασφαλίζει πως καλεί τα πιο πρόσφατα δεδομένα χωρίς την ανάγκη για χειροκίνητη ανανέωση.

```

1 const Hint = document.getElementById('hint');
2 const New = document.getElementById('count-new');
3 const Progress = document.getElementById('count-progress');
4 const Fixed = document.getElementById('count-fixed');
5
6 fetch('/api/reports')
7   .then(r => r.json())
8   .then(reports => {
9     let n=0,p=0,f=0;
10    const layers = [];
11
12    reports.forEach(r => {
13      if (r.status === 1) n++;
14      else if (r.status === 2) p++;
15      else if (r.status === 0) f++;
16
17      const status = [0,1,2].includes(r.status) ? r.status : 'x';
18      const m = L.marker([r.latitude, r.longitude], { icon: ICONS[status] }).addTo(
19        map);
20      let popup = `<b>${r.description ?? 'Report'}</b>`;
21      if (r.image) {
22        popup += `<br><img src= ${r.image} style= width:200px; height:260px; object-fit: cover;`+
23      }
24      m.bindPopup(popup);
25      layers.push(m);
26    });
27    New.textContent = n; Progress.textContent = p; Fixed.textContent = f;
28    Hint.textContent = `Showing ${reports.length} reports.`;
29
30    map.fitBounds(GREECE_BOUNDS, { padding: [20, 20], animate: false });
31  })
32  .catch(err => {
33    console.error('Error fetching reports:', err);
34    Hint.textContent = 'Failed to load reports.';
35    Hint.classList.add('error');
36  });

```

Κάθε εικονίδιο τοποθετείται με βάση τις συντεταγμένες της αναφοράς του. Όταν αυτό πατηθεί, εμφανίζεται ένα πορτραίτο ρορπρ παράθυρο όπου περιέχει την περιγραφή και την εικόνα, κάτι που δίνει την δυνατότητα στους χρήστες να μπορούν να βρουν γρήγορα και εύκολα λεπτομέρειες απευθείας από τον χάρτη. Μόλις όλες οι αναφορές έχουν ενταχθεί στον χάρτη, οι μετρητές στο πάνω μέρος της σελίδας ενημερώνονται και δείχνουν τον αριθμό κάθε κατηγορίας καθώς και τον αριθμό όλων μαζί.

Αν προκύψει κάποιο σφάλμα κατά την εκτέλεση, αυτό εισέρχεται στο “.catch()” το οποίο ενημερώνει το σύστημα και εμφανίζεται σχετικό μήνυμα στην κονσόλα και στην σελίδα ώστε να ενημερωθούν οι χρήστες.

D.3.4 Σελίδα Admin

Η σελίδα “Admin” της ιστοσελίδας είναι ένα απλό αρχείο, όπως και η βασική σελίδα του χάρτη, και χρησιμεύει ως πίνακας ελέγχου όπου ένας διαχειριστής συνδέεται και εκτελεί ενημερώσεις στις υπάρχουσες αναφορές. Η πρόσβαση σε αυτή την σελίδα γίνεται μέσω της ειδικής διεύθυνσης URL “/admin” και η σύνδεση σε αυτή διατηρείται για όσο παραμένει ανοιχτή.

Σύνδεση Διαχειριστή

Κατά την εισαγωγή στην σελίδα, θα πρέπει ο διαχειριστής να συνδεθεί με τον λογαριασμό του. Μέσω της συνάρτησης “AdminLogin()”, ο χρήστης συνδέεται με τα στοιχεία του τα οποία στέλνονται στην βάση μέσω του “/api/login”. Για να πραγματοποιηθεί η σύνδεση, θα πρέπει να ελεγχθεί πρώτα αν ο χρήστης αυτός είναι διαχειριστής μέσω της τιμής “is_admin”, το οποίο γίνεται μέσα στο backend. Αν αυτό πετύχει δημιουργείται ένα session με τα στοιχεία του και εμφανίζεται ο πίνακας ελέγχου. Σε περίπτωση που δεν υπάρχει ο λογαριασμός ή αντίστοιχα υπάρχει αλλά δεν έχει δικαιώματα διαχειριστή, εμφανίζεται σχετικό μήνυμα.

```

1  async function AdminLogin(){
2      err.classList.add('hide'); err.textContent='';
3      const email = u_email.value.trim();
4      const password = u_pass.value;
5      if(!email || !password){
6          err.textContent='Enter email and password.';
```

```

7     err.classList.remove('hide');
8     return;
9 }try{
10    const res = await fetch('/api/login',{
11        method:'POST',
12        headers:{'Content-Type':'application/json'},
13        body:JSON.stringify({email,password})
14    });
15    const data = await res.json();
16    if(!res.ok) throw new Error('Login failed');
17    if(!data?.user?.is_admin) throw new Error('This account is not an admin.');
18    const u = { email: data.user.email, id: String(data.user._id), is_admin: true };
19    setSession(u);
20    showDash();
21    await loadReports();
22 }catch(e){
23     err.textContent = 'Login error';
24     err.classList.remove('hide');
25 }}
```

Φόρτωση Αναφορών

Η βασική συνάρτηση της σελίδας μετά την σύνδεση είναι αυτή της φόρτωσης των αναφορών, η οποία ουσιαστικά καλεί όλες τις υπόλοιπες συναρτήσεις.

```

1 async function loadReports(){
2     const all = await fetchReports();
3     cache = all;
4     const filtered = applyFilters(all);
5     renderRows(filtered);
6     renderCounters(filtered);
7 }
```

Ανάκτηση Αναφορών

Αρχικά, εκτελείται η ανάκτηση των αναφορών από το σημείο “/api/reports”.

```

1  async function fetchReports(){
2      const res = await fetch('/api/reports');
3      const data = await res.json();
4      if(!res.ok) throw new Error('Failed to load reports');
5      return Array.isArray(data)? data : (data?.reports || []);
6  }

```

Φιλτράρισμα Λίστας

Προχωρώντας πραγματοποιείται το φιλτράρισμα στην λίστα δεδομένων με βάση την κατάσταση και την περιγραφή τους. Αν δεν υπάρχει κάποιο ενεργό φίλτρο, η λίστα παραμένει ολόκληρη. Στο τέλος, η τελική λίστα στέλνεται πίσω ταξινομημένη με βάση την ημερομηνία.

```

1  function applyFilters(all){
2      const s = statusFilter.value;
3      const q = searchBox.value.trim().toLowerCase();
4      let list = all;
5      if(s!=='') list = list.filter(r => String(r.status)==s);
6      if(q) list = list.filter(r => (r.description||'').toLowerCase().includes(q));
7      return list.sort((a,b)=> (new Date(b.createdAt)-new Date(a.createdAt)));
8  }

```

Δημιουργία Σειρών

Αφού γίνει το φιλτράρισμα, αρχικά δημιουργούνται οι σειρές της λίστας αναφορών. Η λειτουργία γίνεται μέσω κομματιού html όπου για κάθε αναφορά μέσα στην λίστα δημιουργείται μία σειρά (reportRow). Κάθε σειρά αποτελείται από την περιγραφή, τις συντεταγμένες, την ημερομηνία και την κατάσταση της καταχώρησης. Ακόμα, υπάρχει μία τελευταία στήλη όπου περιέχει κουμπιά ενεργειών ώστε ο διαχειριστής να διαλέξει την κατάσταση που θέλει να θέσει σε μία αναφορά ή να την διαγράψει τελείως. Στο τέλος, η κάθε σειρά προστίθεται στον πίνακα, χωρίς την ανάγκη ανανέωσης της σελίδας.

```

1  function renderRows(list) {
2      tbody.innerHTML = '';

```

```

3   list.forEach(r => {
4     const reportRow = document.createElement('reportRow');
5     reportRow.dataset.id = String(r._id);
6     reportRow.innerHTML =
7       `<td>${r.description}</td>
8       <td>${(r.latitude?.toFixed?.(5) ?? '')}, ${((r.longitude?.toFixed?.(5) ?? ''))} 
9       <td>${toDateStr(r.createdAt)}</td>
10      <td>${statusBadge(r.status)}</td>
11      <td>
12        <div class= actions >
13          <button class= btn data-act= set data-id= ${r._id} data-val= 1 >New</button>
14          <button class= btn data-act= set data-id= ${r._id} data-val= 2 >In Progress</button>
15          <button class= btn data-act= set data-id= ${r._id} data-val= 0 >Fixed</button>
16          <button class= btn danger data-act= del data-id= ${r._id} >Delete</button>
17        </div>
18      </td>`;
19    tbody.appendChild(reportRow);
20  });
21 }

```

Δημιουργία Μετρητών

Μετά την φόρτωση των σειρών, τελευταίοι φορτώνονται οι μετρητές κάθε κατάστασης. Η συνάρτηση, για κάθε είδος κατάστασης, μετράει το σύνολο της ώστε αυτό να οπτικοποιηθεί πάνω από την λίστα. Μαζί με τους μετρητές, εμφανίζεται και ο συνολικός αριθμός όλης της λίστας.

```

1 function renderCounters(list){
2   let n=0,p=0,f=0;
3   list.forEach(r=>{ if(r.status==1) n++; else if(r.status==2) p++; else if(r.status==3) f++; });
4   cNew.textContent=n; cProg.textContent=p; cFix.textContent=f;
5   summary.textContent = `Showing ${list.length} reports.`;
6 }

```

Ενέργειες

Για να την ενημέρωση και διαγραφή των αναφορών, χρησιμοποιείται μία συνάρτηση όπου ενεργοποιείται όταν πατηθεί ένα από τα κουμπιά σε μία σειρά από την λίστα. Αρχικά, ανακτάται το ID και η ενέργεια, και αναλόγως το είδος της εκτελείται διαφορετική σειρά εντολών. Αν η ενέργεια έχει τεθεί σαν “set”, τότε η λαμβάνεται και η τιμή νέας κατάστασης, η οποία τοποθετείται μέσα στο σώμα του αιτήματος “PATCH” με ενσωματωμένο το ID και στέλνεται στο backend ώστε να γίνει η αλλαγή. Αν όμως είναι “del”, τότε στέλνεται σκέτο αίτημα “DELETE” με το ID ώστε να αφαιρεθεί. Αφού γίνει η αντίστοιχη ενέργεια, μετά καλείται ξανά η φόρτωση των αναφορών.

```

1  tbody.addEventListener('click', async (e)=>{
2      const btn = e.target.closest('button[data-act]');
3      if(!btn) return;
4
5      const id = btn.getAttribute('data-id');
6      const act = btn.getAttribute('data-act');
7
8      try{
9          if(act==='set'){
10              const val = Number(btn.getAttribute('data-val'));
11              await fetch(`^/api/reports/${id}/status` , {
12                  method: 'PATCH',
13                  headers: { 'Content-Type': 'application/json' },
14                  body: JSON.stringify({ status: val })
15              });
16          }else if(act==='del'){
17              await fetch(`^/api/reports/${id}` , { method: 'DELETE' });
18          }
19          await loadReports();
20      }catch(err){ alert('Action failed'); }
21  });

```

Διαχείριση του Session

Τέλος, η λειτουργία του session γίνεται με τους παρακάτω τρόπους. Αρχικά, μετά από την σύνδεση, όλα τα στοιχεία του διαχειριστή, σε μορφή JSON, αποθηκεύονται στο sessionStorage κάτω από ένα σταθερό κλειδί. Η συναρτήσεις “setSession()” και “getSession()” χειρίζονται την αποθήκευση και ανάκτηση των πληροφοριών, ενώ το “clearSession()” τις αφαιρεί κατά την αποσύνδεση. Με βάση την ύπαρξη του session με admin, αποφασίζεται αν θα πρέπει να εμφανιστεί η φόρμα σύνδεσης “showLogin()” ή ο πίνακας ελέγχου “showDash()”.

```

1 const session_key = '';
2 const loginBox = document.getElementById('login');
3 const dashBox = document.getElementById('dash');
4 const logoutBtn = document.getElementById('logoutBtn');
5
6 function getSession() {
7     try {return JSON.parse(sessionStorage.getItem(session_key));}
8     catch {return null;}
9 }
10
11 function setSession(obj){sessionStorage.setItem(session_key, JSON.stringify(obj));}
12 function clearSession(){sessionStorage.removeItem(session_key);}
13
14 function showLogin() {
15     loginBox.classList.remove('hide');
16     dashBox.classList.add('hide');
17     logoutBtn.classList.add('hide');
18 }
19 function showDash() {
20     loginBox.classList.add('hide');
21     dashBox.classList.remove('hide');
22     logoutBtn.classList.remove('hide');
23 }
24
25 const sess = getSession();
26 if (sess?.is_admin){showDash(); loadReports();} else {showLogin();}

```

D.4 Κινητή Εφαρμογή

Η κινητή εφαρμογή υεωρείται η βασική διεπαφή της πλατφόρμας και μέσω αυτής οι πολίτες μπορούν να καταχωρίσουν αναφορές για θέματα όπου αντιμετωπίζουν στην πόλη τους. Η ανάπτυξη της έγινε μέσω του Ionic React και της Typescript ώστε να είναι ελαφρία και φιλική προς τον χρήστη. Με την χρήση του Capacitor και των plugin του η ίδια παίρνει πρόσβαση σε λειτουργίες της συσκευής.

D.4.1 Δομή της Εφαρμογής

Η δομή της εφαρμογής χωρίζεται σε διαφορετικές σελίδες για καλύτερη λειτουργία και συντήρηση του κώδικα. Στο αρχείο App.tsx ορίζεται η κύρια λογική δρομολόγησης μέσα στην εφαρμογή και υεωρείται ως το σημείο εισόδου της. Αυτό διαχειρίζεται την πλοήγηση ανάμεσα στις σελίδες χρησιμοποιώντας τον δρομολογητή της Ionic.

Όλες οι σελίδες βρίσκονται μέσα στον φάκελο pages και κάθε μία αντιπροσωπεύει μία λειτουργία της εφαρμογής. Οι λειτουργίες αυτές είναι:

- Η δημιουργία και σύνδεση στον λογαριασμό χρήστη
- Το βασικό μενού της εφαρμογής
- Την λίστα των καταχωρήσεων του χρήστη
- Την φόρμα καταχώρησης των αναφορών

Αυτός ο τρόπος δόμησης βοηθάει στην διατήρηση ενός σαφή διαχωρισμού μεταξύ των οπικών και των λειτουργικών σημείων της εφαρμογής. Ακόμα, διευκολύνει τον εντοπισμό σφαλμάτων και την περαιτέρω ανάπτυξη αφού όλα βρίσκονται στο δικό τους αρχείο.

D.4.2 Σελίδα Καλωσορίσματος

Η πρώτη σελίδα της εφαρμογής είναι αυτή του καλωσορίσματος όπου λειτουργεί ως μία πύλη εισόδου πριν το βασικό μενού. Από εδώ, ο χρήστης μπορεί να δημιουργήσει λογαριασμό και να συνδεθεί με αυτόν. Λειτουργικά η οθόνη αποτελείται από εισόδους κειμένου, επικοινωνεί με το API του backend και κρατάει την ταυτότητα του χρήστη για χρήση στις υπόλοιπες σελίδες της εφαρμογής μετά που θα συνδεθεί.

Αρχικοποίηση Λειτουργιών

Αυτό το κομμάτι χρησιμοποιείται για την αρχικοποίηση των λειτουργιών όπου θα χρειαστούν στην σελίδα ώστε να μπορέσει να υλοποιήσει την δημιουργία και σύνδεση των λογαριασμών των χρηστών.

```

1 const [email, setEmail] = useState('');
2 const [password, setPassword] = useState('');
3 const [busy, setBusy] = useState(false);
4 const [toast, setToast] = useState<{ open: boolean; msg: string }>({ open: false, m
5 const router = useRouter();

```

Στην πράξη το “email” και “password” διατηρούν τις εισόδους και το toast εμφανίζει μηνύματα σε μορφή ειδοποίησης στον χρήστη. Ακόμα, η εντολή “busy” δημιουργεί μία κατάσταση φόρτωσης όταν πραγματοποιούνται ασύγχρονες κλήσεις. Τέλος, το “router” χρησιμοποιείται για την καθοδήγηση του χρήστη στα επόμενα στάδια της εφαρμογής.

Λειτουργία Σύνδεσης

Η λειτουργία σύνδεσης είναι υπεύθυνη για την επαλήθευση υπάρχοντων χρηστών μέσα στην εφαρμογή. Αρχικά, σιγουρεύει πως έχουν εισχαθεί και τα δύο απαραίτητα πεδία του email και του κωδικού αλλιώς στέλνει σχετική ειδοποίηση στον χρήστη. Προχωρώντας, φορτώνει και στέλνει ένα αίτημα POST στο τελικό σημείο “/api/login” στο backend. Το σώμα αιτήματος που περιέχει τις πληροφορίες του χρήστη μεταμορφώνεται σε μορφή JSON και η απάντηση του μετά αναλύεται. Αν η απάντηση είναι επιτυχής το id και το email του χρήστη αποθηκεύονται τοπικά στην συσκευή χρησιμοποιώντας την λειτουργία API “Preferences” του Capacitor. Αυτό επιτρέπει στην εφαρμογή να αποθηκεύει τα δεδομένα για την συγκεκριμένη περίοδο σύνδεσης του χρήστη ώστε να χρησιμοποιηθούν στις επόμενες σελίδες. Ένα μήνυμα επιβεβαίωσης εμφανίζεται για να ενημερώσει τον χρήστη πως συνδέθηκε με επιτυχία και η εφαρμογή μεταφέρεται στην αρχική “Home” σελίδα. Αν η επιβεβαίωση αποτύχει η συνάρτηση εμφανίζει σχετικό μήνυμα σφάλματος. Τέλος, η σελίδα βγαίνει από την λειτουργία φόρτωσης.

```

1 const handleLogin = async () => {
2   if (!email || !password) {
3     setToast({ open: true, msg: 'Please enter both email and password.' });
4     return;

```

```

5      }
6      try {
7          setBusy(true);
8          const response = await fetch('https://smart-city.koyeb.app/api/login', {
9              method: 'POST',
10             headers: { 'Content-Type': 'application/json' },
11             body: JSON.stringify({ email, password }),
12         });
13         const data = await response.json();
14
15         if (response.ok && data.user) {
16             const userEmail = data.user.email ?? email.trim();
17             await Preferences.set({key:'user_id', value: String(userId) });
18             await Preferences.set({ key: 'user_email', value: userEmail });
19             await Preferences.set({key: 'user', value: JSON.stringify({ _id: userId, emai
20             console.log('Saved user info:', { userId, userEmail });
21             setToast({ open: true, msg: 'Login successful!' });
22             router.push('/home', 'root');
23         } else {
24             setToast({open: true, msg: 'Invalid email or password.'});
25         }
26     } catch (err) {
27         console.error('Login error:', err);
28         setToast({ open: true, msg: 'Failed to connect to server.' });
29     } finally {
30         setBusy(false);
31     }
32 };

```

Λειτουργία Εγγραφής

Η λειτουργία εγγραφής διαχειρίζεται την δημιουργία των νέων λογαριασμών χρηστών. Παρομοίως με την σύνδεση, τσεκάρετε ότι τα πεδία είναι συμπληρωμένα. Μετά, μπαίνει στην κατάσταση φόρτωσης και στέλνει τα δεδομένα σε μορφή JSON με αίτημα POST στο σημείο “/api/register”. Το backend επεξεργάζεται τα δεδομένα και επιστρέφει αν δημιουργήθηκε ο λογαριασμός. Αν η δημιουργία έγινε με επιτυχία, ο χρήστης ειδοποιείται πως ο λογαριασμός του έχει δημιουργηθεί και πλέον μπορεί να συνδεθεί σε αυτόν. Αν ο λογαριασμός υπάρχει

ήδη, ενημερώνεται ο χρήστης κατάλληλα αντί να δημιουργηθεί καινούργιος λογαριασμός. Παρομοίως με την σύνδεση, στο τέλος κλείνει η λειτουργία φόρτωσης στην σελίδα.

```

1 const handleRegister = async () => {
2   if (!email || !password) {
3     setToast({ open: true, msg: 'Please enter email and password.' });
4     return;
5   }
6   try {
7     setBusy(true);
8     const response = await fetch('https://smart-city.koyeb.app/api/register', {
9       method: 'POST',
10      headers: { 'Content-Type': 'application/json' },
11      body: JSON.stringify({ email, password }),
12    });
13    const data = await response.json();
14    if (response.ok) {
15      setToast({ open: true, msg: 'Account created successfully! You can now log in' });
16    } else {
17      setToast({ open: true, msg: 'Registration failed.' });
18    }
19  } catch (err) {
20    console.error('Registration error:', err);
21    setToast({ open: true, msg: 'Failed to connect to server.' });
22  } finally {setBusy(false);}
23};

```

D.4.3 Αρχική Σελίδα

Η αρχική σελίδα λειτουργεί ως ένα κεντρικό σημείο πλοήγησης του χρήστη στις άλλες δύο σελίδες της εφαρμογής. Έτσι δίνει στον χρήστη την επιλογή να διαλέξει ο ίδιος αν θέλει να προχωρήσει στην σελίδα καταχώρησης ή στην σελίδα εμφάνισης των καταχωρήσεων του. Η μεταφορά στις επόμενες σελίδες γίνεται μέσω του “useIonRouter();” της Ionic React και υλοποιείται μέσα σε δύο συναρτήσεις.

```

1 const router = useIonRouter();
2 const [presentAlert] = useIonAlert();

```

Μεταφορά στην σελίδα Καταχώρησης

Η κύρια συνάρτηση έχει να κάνει με την μεταφορά στην σελίδα καταχώρησης και διαχειρίζεται την διαδικασία αιτήματος των δικαιωμάτων τοποθεσίας της συσκευής πριν την εισαγωγή του σε αυτή. Όταν πατηθεί το κατάλληλο κουμπί, καλείται απευθείας η “Geolocation.requestPermissions()” του Capacitor όπου ζητάει από τον χρήστη να παραχωρήσει την τωρινή του τοποθεσία. Αν δοθεί η έγκριση, το router μεταφέρει τον χρήστη στην επόμενη σελίδα όπου βρίσκεται η φόρμα καταχώρησης προβλήματος. Αν όμως δεν εγκριθεί, εμφανίζεται μια ειδοποίηση μέσω του “useIonicAlert()” ενημερώνοντας ότι η χρήση τοποθεσίας είναι απαραίτητη για να προχωρήσει παρακάτω. Η ειδοποίηση ακόμα περιέχει την επιλογή στον χρήστη να ανοίξει τις ρυθμίσεις κινητού στην εφαρμογή μέσω του “openAppSettings()” του Capacitor, δίνοντας στον χρήστη την δυνατότητα να δώσει τα δικαιώματα ο ίδιος χειροκίνητα. Αν προκύψει κάποιο σφάλμα, εμφανίζεται κατάλληλη ειδοποίηση όπου ενημερώνει τον χρήστη.

```

1 const handleReportButtonClick = async () => {
2   try {
3     const permissionStatus = await Geolocation.requestPermissions();
4     if (permissionStatus.location === 'granted') {
5       router.push('/report', 'forward');
6     } else if (permissionStatus.location === 'denied') {
7       presentAlert({
8         header: 'Permission Denied',
9         message: 'Location permission is required to report an issue. Please enable
10        buttons: [
11          {
12            text: 'Open Settings',
13            handler: () => {
14              Capacitor?.Plugins?.App?.openAppSettings?();
15            },
16          },
17          'Cancel',
18        ],
19      });
20    }
21  } catch (error) {

```

```

22     console.error('Error requesting location permission', error);
23     presentAlert({
24       header: 'Error',
25       message: 'An error occurred while requesting location permission.',
26       buttons: ['OK'],
27     });
28   }
29 };

```

Μεταφορά στην σελίδα Λίστας Καταχωρήσεων

Η δεύτερη συνάρτηση απλά οδηγεί τον χρήστη στην σελίδα “Οι Καταχωρήσεις μου” η οποία περιέχει μία λίστα με τις καταχωρήσεις του χρήστη αλλά και όλες τις ήδη καταχωρημένες αναφορές μέσα στην βάση. Η διαδικασία της μεταφοράς σε αυτή την περίπτωση δεν χρειάζεται καμία επιπλέον λειτουργία είτε δικαιώματα οπότε ο χρήστης προχωράει κανονικά.

```
1 const goToMyReports = () => router.push('/my-reports', 'forward');
```

D.4.4 Σελίδα ”Οι Καταχωρήσεις μου”

Αυτή η σελίδα είναι υπεύθυνη για την ανάκτηση και την παρουσίαση μία λίστας αναφορών όπου είτε ανήκουν στον συνδεδεμένο χρήστη ή σε όλους τους χρήστες. Γενικά, διαβάζει το id του user, δημιουργεί ένα API αίτημα το οποίο επηρεάζεται από το κουμπί εναλλαγής “Show All” και δέχεται μία απάντηση δεδομένων. Τα δεδομένα αυτά, τα ταξινομεί με βάση την ημερομηνία και τα εμφανίζει ανάλογα.

Αρχικοποίηση

Η αρχικοποίηση της σελίδας περιέχει μία απλή κατάσταση για φόρτωση, τις αναφορές, σφάλματα, το φίλτρο “show all” για να ξέρει ποια δεδομένα να εμφανίσει και το id του χρήστη.

```
1 const [loading, setLoading] = useState(true);
2 const [reports, setReports] = useState<Report[]>([]);
3 const [error, setError] = useState<string | null>(null);
```

```
4 const [showAll, setShowAll] = useState(false);
5 const [userId, setUserId] = useState<string>('');
```

Συνάρτηση Καταστάσεων

Μία σύντομη συνάρτηση όπου μετατρέπει το πεδίο κατάστασης όπου έχει ληφθεί από την βάση για κάθε αναφορά στο κατάλληλο χρώμα και όνομα. Κρατώντας την ξεχωριστά από τα άλλα μέρη του κώδικα βοηθάει κάνοντας τον έλεγχο πιο απλό.

```
1 const statusInfo = (s?: number) => (
2   s === 1 ? { text: 'New', color: '#e53935' } :
3   s === 2 ? { text: 'In Progress', color: '#fb8c00' } :
4   s === 0 ? { text: 'Fixed', color: '#43a047' } :
5   { text: 'Unknown', color: '#9e9e9e' }
6 );
```

Ανάκτηση Δεδομένων

Αυτό το μέρος του κώδικα χειρίζεται την αλλαγή μεταξύ του αν εμφανιστούν τα δεδομένα του χρήστη μόνο ή όλα τα δεδομένα μαζί. Η ανάκτηση τους γίνεται μέσω δύο url στο σημείο “api/report”. Αναλόγως την περίπτωση είτε καλείται το κλασικό αίτημα ή ένα με το id του user ενσωματωμένο. Αφού δεχθεί τα δεδομένα τα ταξινομεί με βάση την ημερομηνία σε φύλουσα σειρά. Τέλος, υπάρχουν χειρισμοί σφαλμάτων και φόρτωσης σε όλη την διαδικασία.

```
1 useEffect(() => {
2   (async () => {
3     try {
4       setLoading(true);
5       setError(null);
6       const { value: uid } = await Preferences.get({ key: 'user_id' });
7       setUserId(uid || '');
8       const url = (!showAll && uid)
9         ? `https://smart-city.koyeb.app/api/reports?userId=${encodeURIComponent(uid)}
10           : 'https://smart-city.koyeb.app/api/reports';
11       const res = await fetch(url);
```

```

12     const data = await res.json();
13     if (!res.ok) throw new Error(data?.message || 'Failed to load reports');
14     const all: Report[] = Array.isArray(data) ? data : data?.reports || [];
15     const sorted = all.sort((a, b) =>
16       new Date(b.createdAt || 0).getTime() - new Date(a.createdAt || 0).getTime());
17   );
18   setReports(sorted);
19 } catch (e: any) {
20   setError(e?.message || 'Failed to load reports');
21 } finally {
22   setLoading(false);
23 }
24 })();
25 }, [showAll]);

```

Η εναλλαγή γίνεται μέσω ενός switch με το “IonToggle” όπου απλά γυρνάει το flag “showAll”. Έτσι γίνεται αλλαγή και επαναλαμβάνεται ξανά η ανάκτηση από το κατάλληλο τελικό σημείο.

```

1 <IonToggle
2   checked={showAll}
3   onIonChange={e => setShowAll(e.detail.checked)}
4 />

```

Φόρτωση Αναφορών στην λίστα

Η λογική της φόρτωσης πριν την εμφάνιση της λίστας αποτελείται από τρία αποτελέσματα. Αρχικά, αν υπάρχει φόρτωση δεδομένων αυτό φαίνεται μέσω ψεύτικων χρησιμοποιώντας το “IonSkeletonText”. Αν δεν υπάρχουν δεδομένα εμφανίζεται ένα απλό μήνυμα “No reports found”. Αν υπάρξει κάποιο σφάλμα εμφανίζεται ένα κόκκινο μήνυμα αποτυχίας φόρτωσης. Τέλος, εμφανίζεται η λίστα κανονικά.

```

1 {loading ? (
2   <IonList>
3     {[...Array(5)].map((_, i) => (
4       <IonItem key={i}>
5         <IonLabel>

```

```

6      <h2><IonSkeletonText animated style={{ width: '60%' }} /></h2>
7      <p><IonSkeletonText animated style={{ width: '40%' }} /></p>
8      </IonLabel>
9      </IonItem>
10     ))}
11   </IonList>
12 ) : error ? (
13   <IonText color= danger >{error}</IonText>
14 ) : reports.length === 0 ? (
15   <IonText>No reports found.</IonText>
16 ) : (
17   <IonList>{/* ... */}</IonList>
18 )

```

Λίστα

Η λίστα δημιουργείται με την χρήση της “IonList”. Κάθε αναφορά μέσα στην λίστα, εμφανίζεται η περιγραφή της και διαμορφώνεται η ημερομηνία της σε καλύτερη μορφή. Ακόμα, τα δεδομένα τοποθεσίας της μειώνονται για καλύτερη ανάγνωση και φορτώνεται η κατάσταση της μαζί με το χρώμα της μέσω του “statusInfo”.

```

1 <IonList>
2   {reports.map((r) => {
3     const dateStr = r.createdAt ? new Date(r.createdAt).toLocaleString() : '';
4     const { text, color } = statusInfo(r.status);
5
6     return (
7       <IonItem key={r._id} lines= full >
8         <IonLabel className= report-left >
9           <h2>{r.description}</h2>
10          <p>
11            {typeof r.latitude === 'number' ? r.latitude.toFixed(5) : r.latitude}, {
12            {typeof r.longitude === 'number' ? r.longitude.toFixed(5) : r.longitude}
13          </p>
14        </IonLabel>
15
16        <div className= report-right >

```

```

17         <div className= report-date >{dateStr}</div>
18         <div className= report-status style={{ color }}>{text}</div>
19     </div>
20   </IonItem>
21 );
22 })
23 </IonList>

```

D.4.5 Σελίδα Καταχώρησης

Η σελίδα καταχώρησης είναι η πιο σημαντική καθώς εδώ εκτελείται η βασική λειτουργία της εφαρμογής η οποία είναι η αποστολή αναφορών. Η διαδικασία αρχίζει βγάζοντας μια φωτογραφία μέσω της κάμερας της συσκευής. Αυτόματα μόλις φορτωθεί η φωτογραφία συλλέγονται και οι γεωγραφικές τιμές της συσκευής εκείνη την στιγμή. Μαζί με μία σύντομη περιγραφή από τον χρήστη όλες αυτές οι πληροφορίες εισάγονται και στέλνονται στην βάση δεδομένων.

Αρχικοποίηση

Όπως και στις άλλες σελίδες αρχικοποιούνται λειτουργίες όπου όλα χρειαστούν στην σελίδα. Κάποιες από αυτές είναι η φωτογραφία, περιγραφή, flag φόρτωσης, συντεταγμένες και ένα μήνυμα επαλήθευσης.

```

1 const [photo, setPhoto] = useState<string | null>(null);
2 const [description, setDescription] = useState<string>('');
3 const [loading, setLoading] = useState<boolean>(true);
4 const [location, setLocation] = useState<{ lat: number; lon: number } | null>(null);
5 const [showToast, setShowToast] = useState(false);
6 const history = useHistory();

```

Λειτουργία Κάμερας

Η συνάρτηση καλεί την κάμερα του κινητού και αποθηκεύει ένα URI. Χρησιμοποιώντας το “resultType: Uri” διατηρείται χαμηλή χρήση μνήμης και μεταφέρεται η μετατροπή σε δυαδικό αρχείο στο βήμα υποβολής.

```

1 const takePhoto = async () => {
2   try {
3     const photo = await Camera.getPhoto({
4       resultType: CameraResultType.Uri,
5       source: CameraSource.Camera,
6       quality: 50,
7       width: 600,
8     });
9     setPhoto(photo.webPath || null);
10  } catch (error) {
11    console.error('Error taking photo', error);
12  }
13};

```

Συλλογή Συντεταγμένων

Μία σύντομη συνάρτηση όπου συλλέγει τις συντεταγμένες της συσκευής και τις αποθηκεύει ως τιμές πλάτους και μήκους πριν την τελική υποβολή.

```

1 const getGeolocation = async () => {
2   try {
3     const position = await Geolocation.getCurrentPosition();
4     setLocation({ lat: position.coords.latitude, lon: position.coords.longitude });
5   } catch (error) {
6     console.error('Error getting geolocation', error);
7   }
8 };

```

Κατάσταση Φόρτωσης

Η σελίδα μπαίνει στο σημείο φόρτωσης καθώς περιμένει τις τιμές από τις συναρτήσεις φωτογραφίας και τοποθεσίας, οι οποίες εκτελούνται ταυτόχρονα.

```

1 useEffect(() => {
2   const init = async () => {

```

```

3     setLoading(true);
4     await Promise.all([takePhoto(), getGeolocation()]);
5     setLoading(false);
6   };
7   init();
8 }, []);
```

Συνάρτηση Υποβολής

Η βασική συνάρτηση της σελίδας είναι αυτή της υποβολής. Αρχίζει διαβάζοντας τα αποθηκευμένα στοιχεία του χρήστη, από τα οποία παίρνει το id του. Αφού το πάρει, οργανώνει τα δεδομένα σε μία φόρμα υποβολής και μετατρέπει την εικόνα από URI σε δυαδική μορφή μέσω του Blob. Τέλος, στέλνει αίτημα POST στο τελικό σημείο “/api/reports” και περιμένει την απάντηση. Αν είναι θετική ειδοποιεί τον χρήστη πως η αναφορά του καταχωρήθηκε με επιτυχία και τον επιστρέφει στην αρχική σελίδα. Αντίστοιχα αν είναι αρνητική ή αν προκύψει κάποιο σφάλμα ο χρήστης ειδοποιείται ανάλογα.

```

1 async function submitReport(description: string, latitude: number, longitude: number) {
2   const { value } = await Preferences.get({ key: 'user' });
3   const user = value ? JSON.parse(value) : null;
4   const userId = user?._id || '';
5   try {
6     const formData = new FormData();
7     formData.append('description', description);
8     formData.append('latitude', latitude.toString());
9     formData.append('longitude', longitude.toString());
10    formData.append('user_id', userId);
11    if (imageUri) {
12      const response = await fetch(imageUri);
13      const blob = await response.blob();
14      formData.append('image', blob, 'report.jpg');
15    }
16    const response = await fetch('https://smart-city.koyeb.app/api/reports', {
17      method: 'POST',
18      body: formData,
19      mode: 'cors',
20    });
21    if (response.ok) {
22      toast.success('Η αναφορά σας έχει καταχωριστεί με επιτυχία!');
23      navigation.replace('Home');
24    } else {
25      toast.error('Δεν ήταν δυνατό να καταχωριστεί η αναφορά σας.');
26    }
27  } catch (error) {
28    toast.error(`Σφάλμα στην υποβολή της αναφοράς: ${error.message}`);
29  }
30}
```

```

20     });
21     const data = await response.json();
22     if (response.ok) {
23       setShowToast(true);
24       setTimeout(() => history.push('/home'), 2000);
25     } else {
26       console.error('Failed to submit report:', data.message);
27     }
28   } catch (error) {
29     console.error('Error submitting report:', error);
30   }
31 }

```

Πριν εκτελεστεί η συνάρτηση υποβολής, υπάρχει σχετικός έλεγχος ότι όλα τα δεδομένα έχουν εισαχθεί με επιτυχία ή όχι κατά το πάτημα του κουμπιού "Submit".

```

1 <IonButton
2   expand= block
3   onClick={() => {
4     if (description && location && photo) {
5       submitReport(description, location.lat, location.lon, photo);
6     } else {
7       alert('Please provide a description and allow location access.');
8     }
9   }}
10 >
11   Submit
12 </IonButton>

```

D.5 Android Studio και Δοκιμές

Η εφαρμογή δοκιμάστηκε απευθείας σε μία πραγματική συσκευή Android χρησιμοποιώντας τις εντολές του Capacitor CLI, χωρίς την εκτέλεση μίας πλήρης ανάπτυξης μέσω του Android Studio. Το ίδιο το Android Studio χρησιμοποιήθηκε χυρίως για την εύκολη εγκατάσταση των απαραίτητων SDKs και εργαλείων για ώστε να μπορεί να αναγνωριστεί η συσκευή καθώς και τον έλεγχο των logs κονσόλας κατά την δοκιμή της εφαρμογής. Η

διαδικασία δοκιμής ήταν αρκετά απλή. Στην συσκευή χρειάζεται να ενεργοποιηθούν τα “Developer Options” και η ρύθμιση “USB Debugging”. Από την μεριά του υπολογιστή, η εφαρμογή δημιουργήθηκε χρησιμοποιώντας την εντολή **npm run build** και συγχρονίστηκε χρησιμοποιώντας την εντολή **npx cap sync android**. Στη συνέχεια, εκτελέστηκε με την εντολή **npx cap run android** όπου μαζί με την επιλογή της σωστής συσκευής, δημιουργεί την εφαρμογή εισάγοντας την σε αυτή. Αυτή η προσέγγιση επέτρεψε γρήγορες ενημερώσεις και δοκιμές χωρίς την ανάγκη ανοίγματος του έργου μέσα σε IDE όπως το Android Studio για την εκτέλεση.

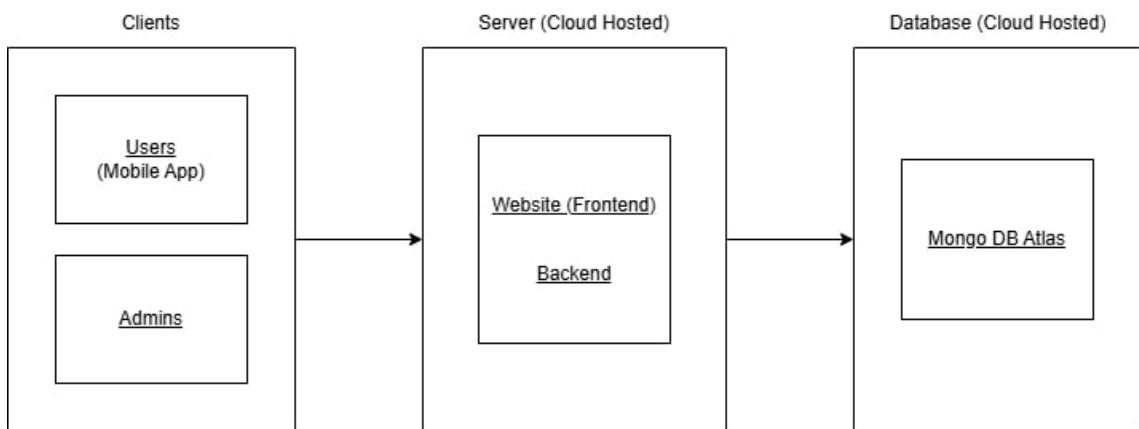
Κατά την διάρκεια των δοκιμών, ελέγχθηκε ότι τα δικαιώματα της κάμερας και της τοποθεσίας λειτουργούσαν κανονικά, δείχνοντας καθαρά μηνύματα. Ακόμα, επιβεβαιώθηκε πως η υποβολή λειτουργούσε όπως ήταν αναμενόμενο κάνοντας αρκετές δοκιμές τραβώντας φωτογραφία, γράφοντας περιγραφή και υποβάλλοντας τα μέσω της εφαρμογής. Τέλος, δοκιμάστηκαν όλες οι υπόλοιπες σελίδες με δοκιμές στην δημιουργία και στην σύνδεση χρήστη, την εμφάνιση της λίστας και την σωστή καθοδήγηση της αρχικής σελίδας. Γενικά, όλα τα τεστ επιβεβαίωσαν ότι η εφαρμογή δουλεύει σωστά και ομαλά στο Android και επικοινωνεί με τις υπηρεσίες του backend.

D.6 Διάγραμμα Ανάπτυξης

Το ανεπτυγμένο σύστημα, αποτελείται από δύο ειδών clients, τους κανονικούς χρήστες και τους διαχειριστές, όπως φαίνεται στο σχήμα D.12. Οι απλοί χρήστες έχουν πρόσβαση στην πλατφόρμα μέσω της κινητής εφαρμογής, η οποία δημιουργείται μέσω ενός αρχείου “apk”, το οποίο μπορεί να εγκατασταθεί σε οποιαδήποτε συσκευή Android χωρίς την ανάγκη ειδικών εργαλείων όπως το Android Studio. Μετά την εγκατάσταση, ο χρήστης μπορεί να ανοίξει και να χρησιμοποιήσει την εφαρμογή. Οι διαχειριστές αποκτούν πρόσβαση στο σύστημα μέσω της ιστοσελίδας χρησιμοποιώντας ένα πρόγραμμα περιήγησης. Οι δύο αυτές διεπαφές επικοινωνούν με τον server μέσω αιτημάτων HTTP (API). Όταν ένας από αυτούς τους χρήστες εκτελεί μία ενέργεια, το αντίστοιχο αίτημα στέλνεται στον server.

Ο server φιλοξενεί σε cloud και το frontend της ιστοσελίδας καθώς και την λογική του backend. Το backend δέχεται εισερχόμενα αιτήματα API, τα διαχειρίζεται και αποφασίζει τις κατάλληλες ενέργειες. Όταν τα δεδομένα πρέπει να αποθηκευτούν, να επεξεργαστούν

ή να ανακτηθούν, τότε αυτό επικοινωνεί με την cloud βάση του MongoDB Atlas. Η σύνδεση μεταξύ τους γίνεται δυνατή μέσω μίας συμβολοσειράς σύνδεσης όπου παρέχεται από την MongoDB. Η βάση δεδομένων δεν συνδέεται ποτέ απευθείας με τα clients καθώς η επικοινωνία γίνεται αποκλειστικά μέσω του backend. Με αυτό τον τρόπο, τα τρία επίπεδα του τελικού συστήματος δουλεύουν εύκολα μαζί, δημιουργώντας μία λειτουργική και ασφαλής πλατφόρμα.



Σχήμα D.12: Διάγραμμα Ανάπτυξης Συστήματος

Κεφάλαιο Ε

Πειραματισμός

E.1 Σκοπός Πειραματισμού

Tο κεφάλαιο αυτό έχει να κάνει με την πειραματική διαδικασία που έγινε ώστε να αξιολογηθεί η πλατφόρμα της έξυπνης πόλης όσον αφορά την λειτουργικότητα, την χρηστικότητα και την απόδοση της. Ακολουθώντας την ολοκλήρωση της φάσης της υλοποίησης, μία σειρά από δοκιμές έγιναν ώστε να βεβαιωθεί πως η κινητή εφαρμογή και η ιστοσελίδα λειτουργούν σωστά, επικοινωνούν αποτελεσματικά με το backend και την βάση δεδομένων και υπάρχει μία ομαλή εμπειρία χρήστη.

Ο πειραματισμός εστιάζει πάνω σε τρεις βασικές φάσεις. Η πρώτη φάση έχει να κάνει με την λειτουργική επαλήθευση, δηλαδή την επιβεβαίωση πως όλες οι βασικές λειτουργίες του συστήματος όπως η υποβολή αναφοράς και η απεικόνιση του χάρτη και των καταχωρήσεων σε αυτόν, δουλεύουν όπως είχαν προοριστεί. Το δεύτερο κομμάτι αφορά την αξιολόγηση από εξωτερικούς, από την ανάπτυξη, χρήστες που είδαν την πλατφόρμα και προσφέρει μία καθαρή και ανεξάρτητη γνώμη σχετικά με τον σχεδιασμό και την εμπειρία του συστήματος. Η τρίτη και τελευταία, εστιάζει στην συζήτηση των ευρημάτων από τα αποτελέσματα των χρηστών σε σχέση με την αρχική εκτίμηση.

Μέσω του συνδυασμού των τριών μεθόδων, παρέχονται αντικειμενικές ενδείξεις σωστής λειτουργίας αλλά και εσωτερικές γνώσεις σχετικά με την ποιότητα του συστήματος και την ευχρηστία του. Έτσι, μπορούν να βγουν κατανοητά αποτελέσματα, τα οποία θα συγχριθούν με τους αρχικούς στόχους.

E.2 Διαδικασία Πειραματισμού και Αποτελέσματα

E.2.1 Λειτουργική Επιβεβαίωση

Με σκοπό να επιβεβαιώθει ότι το σύστημα δουλεύει όπως αναμένεται, ένα σύνολο από στοχευμένα λειτουργικά τεστ πραγματοποιήθηκε πάνω στα βασικά χαρακτηριστικά της κινητής εφαρμογής και της ιστοσελίδας.

Το τελικό τεστ έγινε σε πραγματική Android συσκευή χρησιμοποιώντας την ανεπτυγμένη εφαρμογή. Η ιστοσελίδα ανέβηκε με την χρήση μίας δωρεάν υπηρεσίας φιλοξενίας και χρησιμοποιήθηκε μέσω προγράμματος περιήγησης στον υπολογιστή. Τα δύο αυτά συστήματα επικοινωνούσαν μέσω του backend όπου έτρεχε μαζί με την ιστοσελίδα και εξασφάλιζαν έτσι σύνδεση με την cloud βάση δεδομένων.

Κάθε δοκιμή εστιάζει σε μία κρίσιμη ενέργεια στην ροή εργασίας της πλατφόρμας, από την υποβολή έως και την ανάκτηση των δεδομένων και τις λειτουργίες του διαχειριστή. Όπως φαίνεται στον πίνακα E.1, υπάρχει ένα αναμενόμενο και ένα πραγματικό αποτέλεσμα για κάθε περίπτωση δοκιμής και αν αυτό μπορεί να θεωρηθεί δεκτό.

Όλα τα λειτουργικά τεστ ολοκληρώθηκαν με επιτυχία, επιβεβαιώνοντας ότι η κύρια επικοινωνία μεταξύ της κινητής εφαρμογής και της ιστοσελίδας, του API και της βάσης δεδομένων δουλεύουν με αξιοπιστία.

Κανένα σημαντικό σφάλμα ή κόλλημα δεν εμφανίστηκε και όλες οι διαδικασίες μεταφοράς δεδομένων εκτελέστηκαν όπως αναμενόταν. Μία μικρή καθυστέρηση παρατηρήθηκε μόνο όταν γίνεται η πρώτη ενέργεια κλήσης του API, κάτι το οποίο είναι συνηθισμένο σε δωρεάν υπηρεσίες φιλοξενίας ιστοτόπων καθώς αν υπάρχει απραξία του backend ή της ιστοσελίδας για αρχετή ώρα, αυτή αυτόματα μπαίνει σε κατάσταση αδράνειας.

Πίνακας Ε.1: Αποτελέσματα ελέγχων λειτουργικότητας της πλατφόρμας

Περίπτωση Ελέγχου	Αναμενόμενο Αποτέλεσμα	Πραγματικό Αποτέλεσμα	Πέτυχε;
Εγγραφή και σύνδεση χρήστη	Ο λογαριασμός δημιουργείται και η ταυτοποίηση ολοκληρώνεται	Ο λογαριασμός δημιουργήθηκε και η σύνδεση έγινε με επιτυχία	Ναι
Τυποβολή αναφοράς (φωτογραφία, περιγραφή, GPS)	Η αναφορά αποθηκεύεται στη βάση δεδομένων	Η αναφορά αποθηκεύτηκε και εμφανίστηκε στην βάση δεδομένων	Ναι
Εμφάνιση όλων των αναφορών στον χάρτη	Ανάκτηση όλων των αναφορών με σωστές συντεταγμένες και οπτικοποίηση τους	Οι αναφορές ανακτήθηκαν και εμφανίστηκαν στον χάρτη	Ναι
Ενημέρωση κατάστασης αναφοράς από διαχειριστή	Η αλλαγή ή διαγραφή κατάστασης αντικατοπτρίζεται άμεσα στον χάρτη	Η αναφορά άλλαξε ή διαγράφτηκε και ενημερώθηκε ο χάρτης	Ναι
Επαλήθευση αποστολής και απεικόνισης εικόνας	Οι εικόνες αποθηκεύονται και συνδέονται σωστά με κάθε αναφορά	Η εικόνα αποθηκεύτηκε σωστά στην βάση και εμφανίζεται στην διεπαφή όταν πατηθεί η αναφορά της	Ναι

E.2.2 Αξιολόγηση

Η αξιολόγηση της πλατφόρμας, έχει σκοπό να εκτιμήσει την χρηστικότητα και την απόδοση της στην διαδικασία αναφοράς και διαχείρισης ζητημάτων ενός δήμου. Δεδομένου ότι το σύστημα βρίσκεται σε πειραματικό στάδιο και δεν έχει κυκλοφορηθεί δημόσια, η αξιολόγηση έγινε χυρίως μέσω επίδειξης.

Οι συμμετέχοντες παρακολούθησαν ένα περιεκτικό βίντεο παρουσίασης της εφαρμογής και της ιστοσελίδας, απεικονίζοντας την πλήρη λειτουργικότητα τους από την μεριά του

χρήστη. Η επίδειξη παρουσίαζε την σύνδεση σε λογαριασμό, την δημιουργία μίας αναφοράς με φωτογραφία και περιγραφή μέσω της εφαρμογής, και την οπτικοποίηση της πάνω στον διαδραστικό χάρτη της ιστοσελίδας.

Το τελικό σύνολο του δείγματος έφτασε τις 27 απαντήσεις που συλλέχθηκαν μέσω ενός διαδικτυακού ερωτηματολογίου χρησιμοποιώντας την κλίμακα Likert με πέντε βαθμίδες, όπου το “1” να δηλώνει “Διαφωνώ Απόλυτα” και αντίστοιχα το “5” να σημαίνει “Συμφωνώ Απόλυτα”. Οι ερωτήσεις κάλυψαν τις βασικές πλευρές της εμπειρίας ενός χρήστη, όπως η ευκολία χρήσης, σαφήνεια της πλοιήγησης, καλή ανταπόκριση και γενική ικανοποίηση.

Παρακάτω, παρουσιάζονται τα αποτελέσματα και η ανάλυση κάθε ερώτησης, συγχρίνοντας τις με το πως αυτά τα ευρήματα σχετίζονται με την πραγματική συμπεριφορά του συστήματος που παρατηρήθηκε κατά την υλοποίηση και τις τελικές δοκιμές.

Πληροφορίες Συμμετεχόντων

Πριν τις βασικές ερωτήσεις, οι συμμετέχοντες ζητήθηκαν να παραχωρήσουν ορισμένες βασικές πληροφορίες όπως το φύλο και την ηλικία τους, αλλά και το ιστορικό τους σχετικά με την εξοικείωση τους, με την τεχνολογία και την πιθανή εμπειρία τους με την χρήση παρόμοιων εφαρμογών.

Όπως απεικονίζεται στον πίνακα E.2, η κατανομή των φύλων ήταν ισορροπημένη με 51,9% των ερωτηθέντων να δηλώνουν “Αντρες” και 48,1% να δηλώνουν “Γυναίκες”. Σχετικά με την ηλικία, η πλειοψηφία ήταν στο γχρουπ “Κάτω των 25” με 74,1%, ενώ 11,1% ήταν ανάμεσα σε 25 με 40 χρονών και το υπόλοιπο 14,8% να αντιστοιχεί σε αυτούς πάνω των 40 ετών. Αυτή η κατανομή δείχνει πως το δείγμα αποτελούνταν κυρίως από νεαρούς συμμετέχοντες, τυπικά πιο εξοικειωμένους με την τεχνολογία, αν και η παρουσία μεγαλύτερων ηλικιών παρείχε πολύτιμες πληροφορίες.

Πίνακας Ε.2: Εισαγωγικές Πληροφορίες Χρηστών

Κατηγορία	Γκρουπ	Ποσοστό (%)
Φύλο	Άνδρας	51.9
	Γυναίκα	48.1
Γκρουπ Ηλικίας	Κάτω των 25	74.1
	25–40 χρονών	11.1
	Άνω των 40	14.8

Οι συμμετέχοντες ακόμα, ερωτήθηκαν σε δύο εισαγωγικές αλλά σημαντικές πληροφορίες:

- Την εμπειρία τους με την τεχνολογία (κλίμακα 1-5).
- Αν έχουν ξαναχρησιμοποιήσει παρόμοιες εφαρμογές.

Οι πιο πολλοί, βαθμολόγησαν τον εαυτό με την τιμή 4, υποδηλώνοντας μία γενική μέτρια προς θετική εξοικείωση. Όμως, το 63% από αυτούς, δήλωσε πως δεν έχει ξαναχρησιμοποιήσει κάποιο παρόμοιο τύπο εφαρμογής, κάτι το οποίο σημαίνει πως αυτή η πλατφόρμα ήταν η πρώτη τους επαφή με τέτοιου είδους ψηφιακού συστήματος. Οι απαντήσεις τους εμφανίζονται αναλυτικά στον παρακάτω πίνακα:

Ερώτηση	Απάντηση	Ποσοστό (%)
Εξοικείωση στην τεχνολογία (1-5)	1	7.4
	2	11.1
	3	11.1
	4	55.5
	5	14.9
Εμπειρία με παρόμοια εφαρμογή	Ναι	37.0
	Όχι	63.0

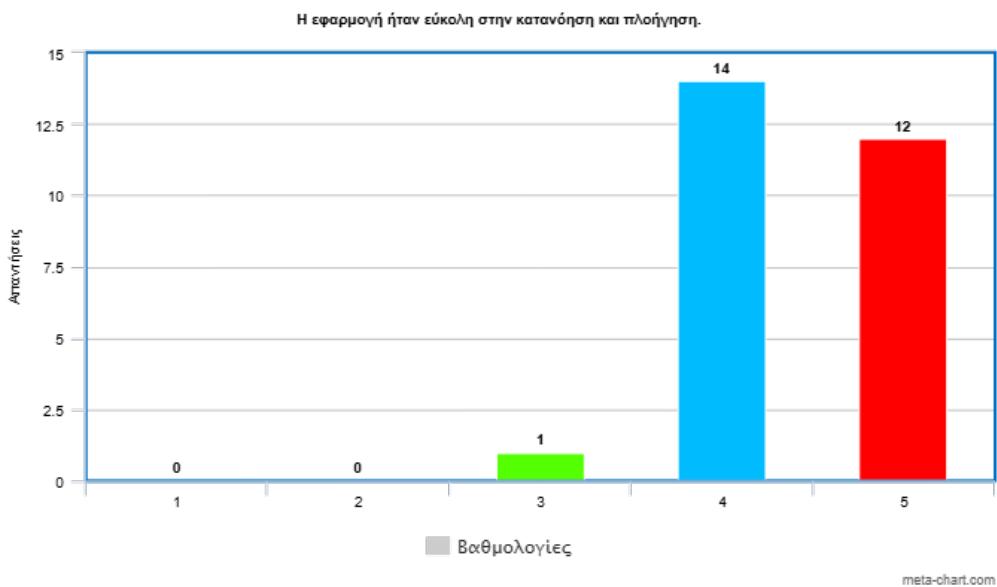
Πίνακας Ε.3: Εξοικείωση με την τεχνολογία και προηγούμενη εμπειρία με παρόμοιες εφαρμογές

Αποτελέσματα Ερωτηματολογίου

Η εφαρμογή ήταν εύκολη στην κατανόηση και πλοήγηση.

- Βαθμολογία 5 (Συμφωνώ απόλυτα): 12
- Βαθμολογία 4 (Συμφωνώ): 14
- Βαθμολογία 3 (Ουδέτερος): 1
- Βαθμολογία 2 (Διαφωνώ): 0
- Βαθμολογία 1 (Διαφωνώ απόλυτα): 0

Η επικρατούσα τιμή είναι το 4 (Συμφωνώ), καθώς έχει την μεγαλύτερη συχνότητα (14). Ο μέσος όρος είναι 4,4 και η διάμεσος 4. Τα αποτελέσματα αυτά δείχνουν ότι σχεδόν όλοι οι συμμετέχοντες αντιλήφθηκαν την δομή και την ροή της εφαρμογής ως κατανοητή και εύχρηστη, κάτι το οποίο επιβεβαιώνει την επιτυχία του σχεδιασμού της διεπαφής του χρήστη. (Σχήμα E.13)

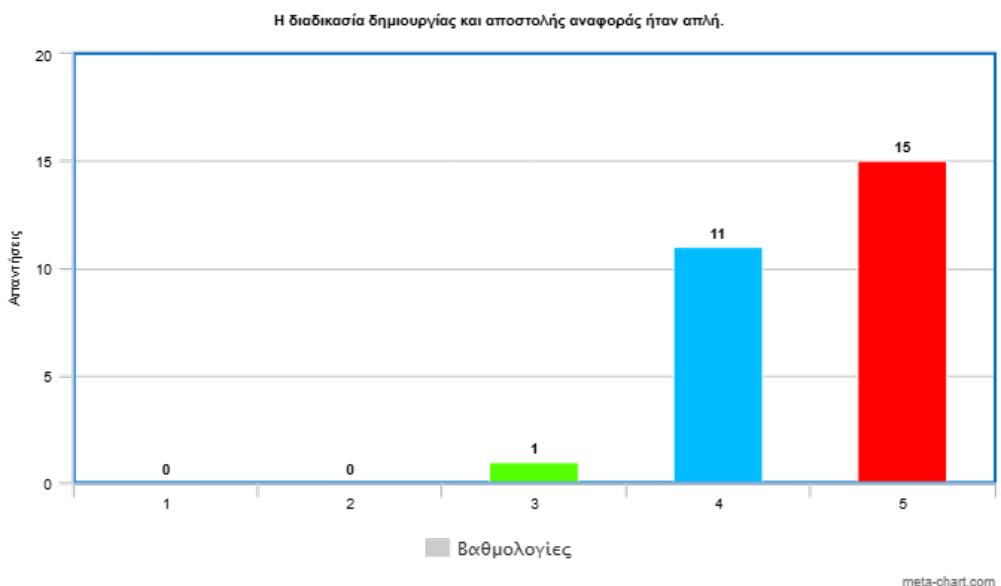


Σχήμα E.13: Η εφαρμογή ήταν εύκολη στην κατανόηση και πλοήγηση.

Η διαδικασία δημιουργίας και αποστολής αναφοράς ήταν απλή.

- Βαθμολογία 5: 15
- Βαθμολογία 4: 11
- Βαθμολογία 3: 1
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 5 (Συμφωνώ απόλυτα), με 15 εμφανίσεις. Ο μέσος όρος είναι 4,5 και η διάμεσος 5. Η υψηλή βαθμολογία δείχνει ότι οι χρήστες βρήκαν την διαδικασία υποβολής αναφοράς απλή και εύκολη, κάτι που διευκολύνει σημαντικά την συμμετοχή πολιτών. (Σχήμα E.14)



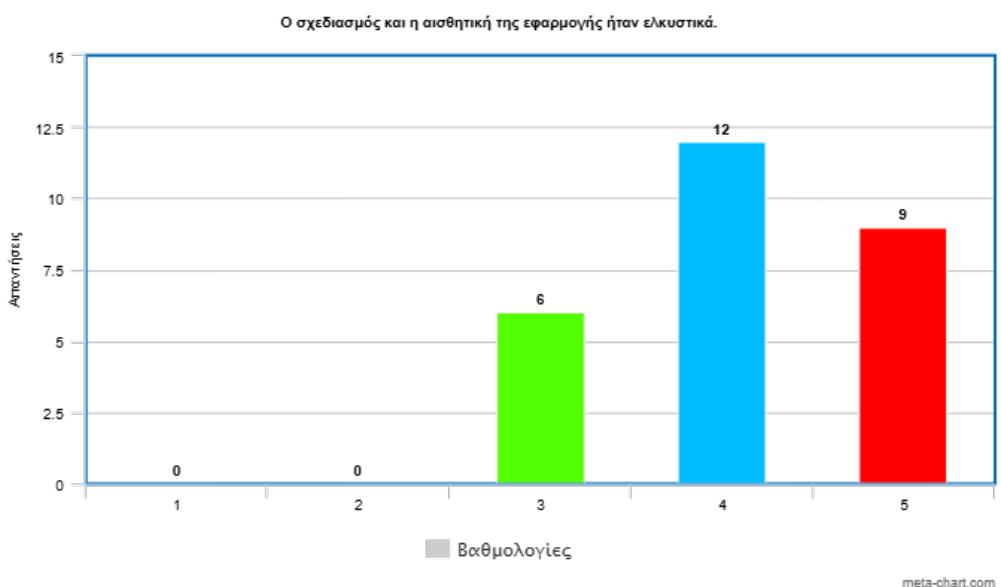
Σχήμα E.14: Η διαδικασία δημιουργίας και αποστολής αναφοράς ήταν απλή.

Ο σχεδιασμός και η αισθητική της εφαρμογής ήταν ελκυστικά.

- Βαθμολογία 5: 9
- Βαθμολογία 4: 12

- Βαθμολογία 3: 6
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 4, με την μεγαλύτερη συχνότητα (12). Ο μέσος όρος είναι 4,1 και η διάμεσος 4. Οι περισσότεροι συμμετέχοντες έκριναν τον οπτικό σχεδιασμό ως ευχάριστο και καλαρό, γεγονός που ενισχύει την καλή πρώτη εντύπωση και την συνολική εμπειρία χρήσης. (Σχήμα E.15)

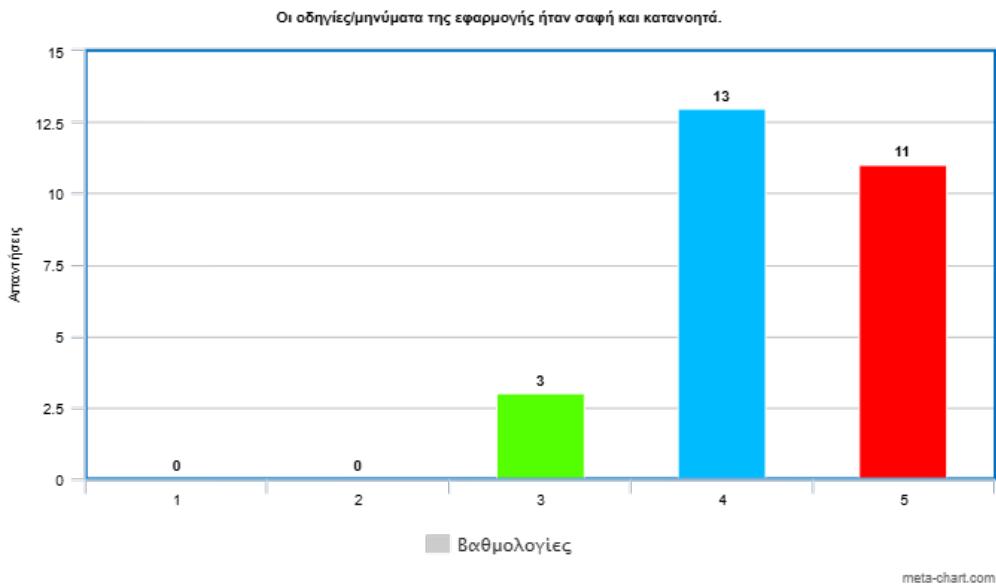


Σχήμα Ε.15: Ο σχεδιασμός και η αισθητική της εφαρμογής ήταν ελκυστικά.

Οι οδηγίες/μηνύματα της εφαρμογής ήταν σαφή και κατανοητά.

- Βαθμολογία 5: 11
- Βαθμολογία 4: 13
- Βαθμολογία 3: 3
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 4. Ο μέσος όρος είναι 4,3 και η διάμεσος 4. Η πλειονότητα των απαντήσεων δείχνει ότι οι χρήστες κατανόησαν τα μηνύματα και τις οδηγίες χωρίς δυσκολία, στοιχείο που υποδηλώνει επιτυχημένη σχεδίαση επικοινωνίας με τον χρήστη. (Σχήμα E.16)

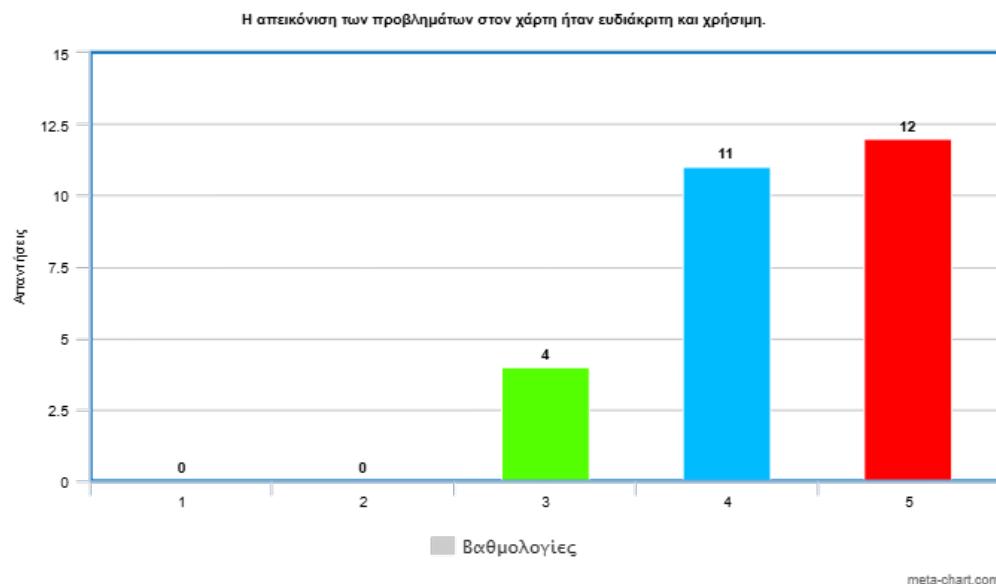


Σχήμα E.16: Οι οδηγίες/μηνύματα της εφαρμογής ήταν σαφή και κατανοητά.

Η απεικόνιση των προβλημάτων στον χάρτη ήταν ευδιάκριτη και χρήσιμη.

- Βαθμολογία 5: 12
- Βαθμολογία 4: 11
- Βαθμολογία 3: 4
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 5. Ο μέσος όρος είναι 4,3 και η διάμεσος 4. Η αξιολόγηση αυτή δείχνει ότι οι χρήστες θεωρούν τον χάρτη και την οπτικοποίηση του ως ένα λειτουργικό και ευδιάκριτο εργαλείο παρουσίασης των αναφορών, που συμβάλλει στην αποτελεσματική απεικόνιση των δεδομένων. (Σχήμα E.17)

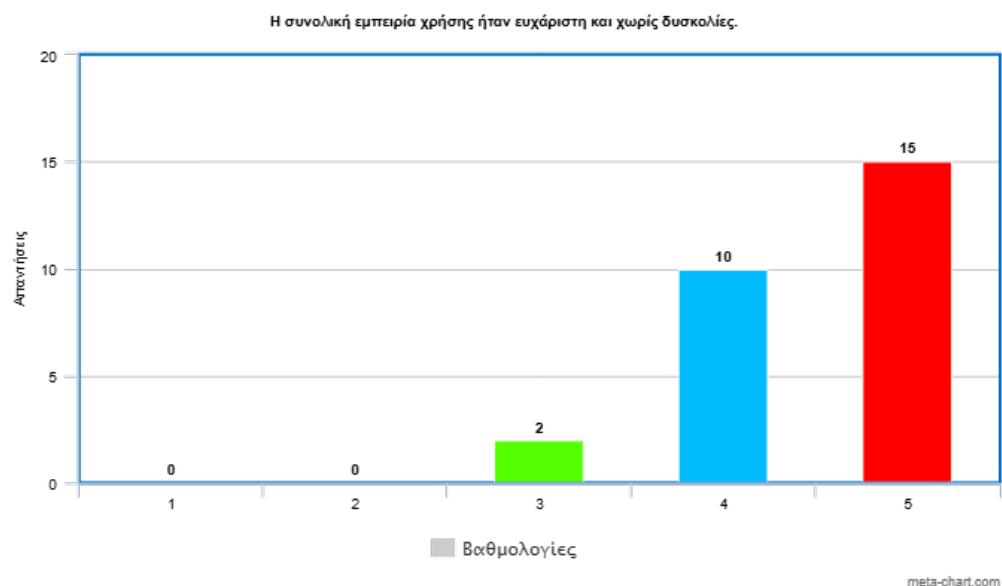


Σχήμα E.17: Η απεικόνιση των προβλημάτων στον χάρτη ήταν ευδιάκριτη και χρήσιμη.

Η συνολική εμπειρία χρήσης ήταν ευχάριστη και χωρίς δυσκολίες.

- Βαθμολογία 5: 15
- Βαθμολογία 4: 10
- Βαθμολογία 3: 2
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επιχρατούσα τιμή είναι 5. Ο μέσος όρος είναι 4,5 και η διάμεσος 5. Οι απαντήσεις καταδεικνύουν ότι οι περισσότεροι χρήστες θεώρησαν την εμπειρία ως θετική χωρίς τεχνικά προβλήματα ή καθυστερήσεις, επιβεβαιώνοντας την αξιοπιστία του συστήματος. (Σχήμα E.18)

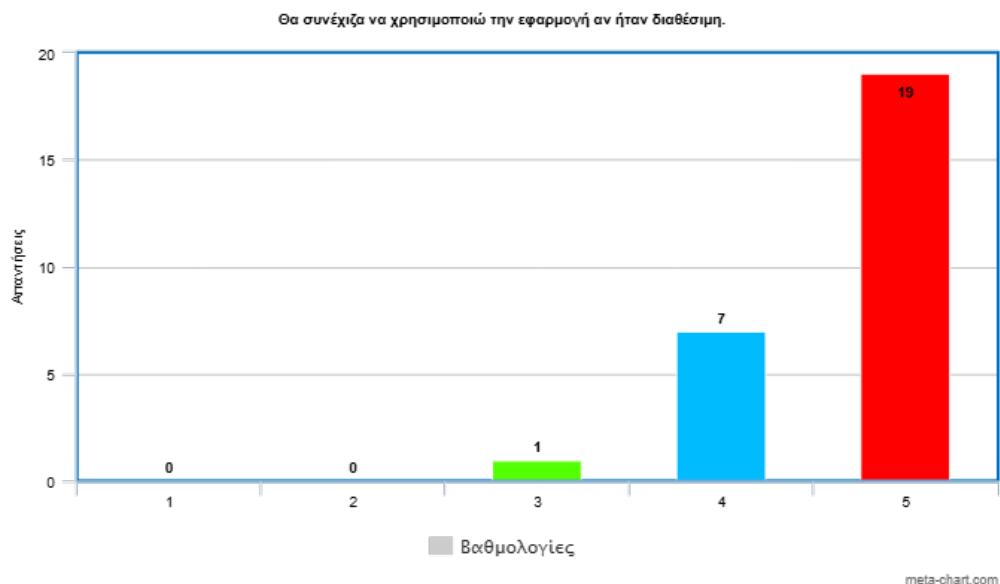


Σχήμα E.18: Η συνολική εμπειρία χρήστης ήταν ευχάριστη και χωρίς δυσκολίες.

Θα συνέχιζα να χρησιμοποιώ την εφαρμογή αν ήταν διαθέσιμη.

- Βαθμολογία 5: 19
- Βαθμολογία 4: 7
- Βαθμολογία 3: 1
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 5. Ο μέσος όρος είναι 4,7 και η διάμεσος 5. Η σχεδόν ομόφωνη θετική πρόθεση δείχνει ότι οι χρήστες θα ήταν πρόθυμοι να συνεχίσουν την χρήση της εφαρμογής, γεγονός που αντικατοπτρίζει υψηλό επίπεδο ικανοποίησης. (Σχήμα E.19)

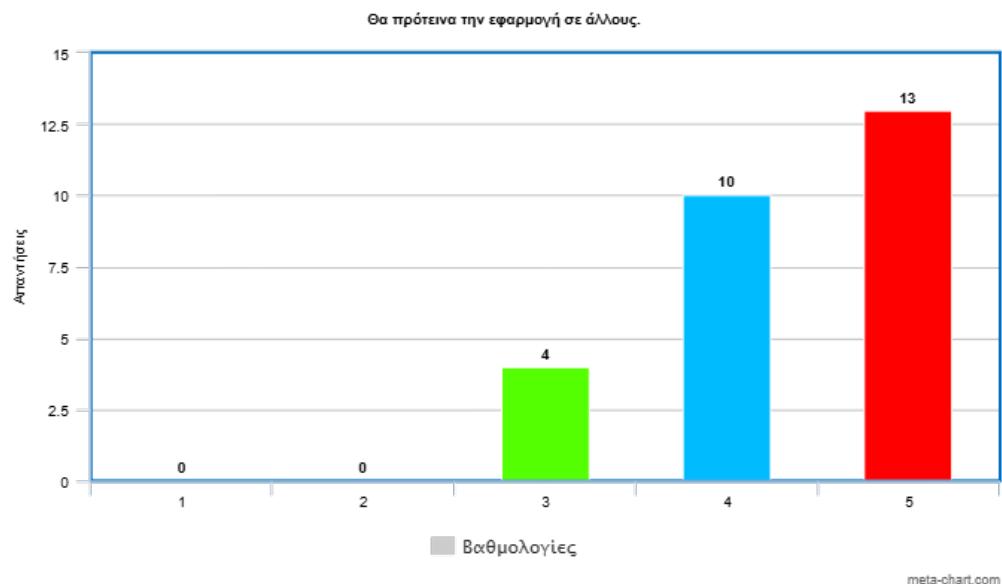


Σχήμα E.19: Θα συνέχιζα να χρησιμοποιώ την εφαρμογή αν ήταν διαθέσιμη.

Θα πρότεινα την εφαρμογή σε άλλους.

- Βαθμολογία 5: 13
- Βαθμολογία 4: 10
- Βαθμολογία 3: 4
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επικρατούσα τιμή είναι 5. Ο μέσος όρος είναι 4,3 και η διάμεσος 4. Η πρόθεση να προτείνουν την εφαρμογή σε άλλους χρήστες δείχνει υψηλό βαθμό αποδοχής και εμπιστοσύνης προς το σύστημα. (Σχήμα E.20)

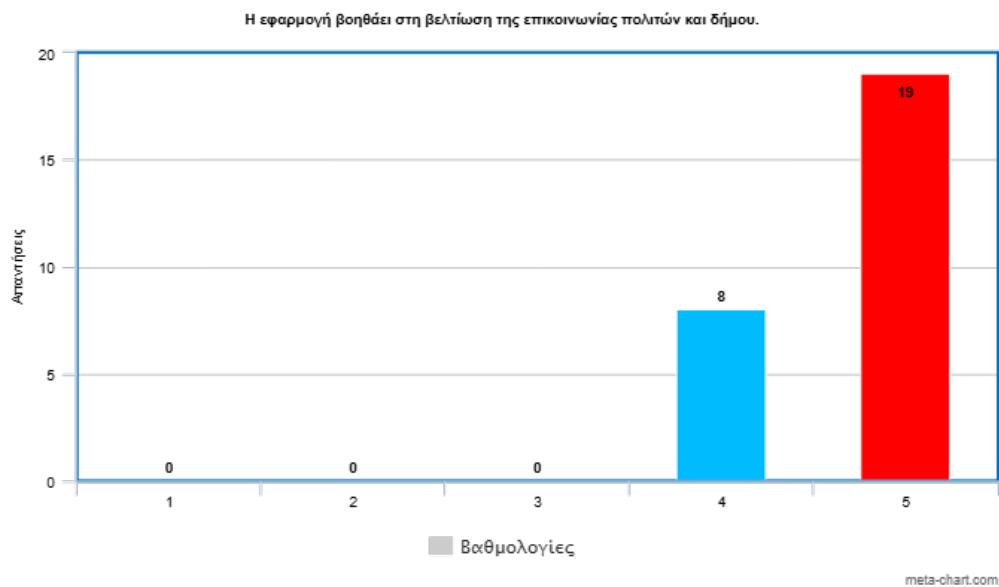


Σχήμα E.20: Θα πρότεινα την εφαρμογή σε άλλους.

Η εφαρμογή βοηθάει στη βελτίωση της επικοινωνίας πολιτών και δήμου.

- Βαθμολογία 5: 19
- Βαθμολογία 4: 8
- Βαθμολογία 3: 0
- Βαθμολογία 2: 0
- Βαθμολογία 1: 0

Η επιχρατούσα τιμή είναι 5. Ο μέσος όρος είναι 4,7 και η διάμεσος 5. Η πλήρως θετική αξιολόγηση επιβεβαιώνει ότι οι χρήστες καταλήγουν πως η εφαρμογή μπορεί να γίνει ένα ουσιαστικό μέσο βελτίωσης της επικοινωνίας μεταξύ των πολιτών και δημοτικών αρχών, κάτι που ήταν σημαντικός στόχος της όλης πλατφόρμας. (Σχήμα E.21)



Σχήμα E.21: Η εφαρμογή βοηθάει στη βελτίωση της επικοινωνίας πολιτών και δήμου.

E.3 Συζήτηση των ευρημάτων

Τα αποτελέσματα της αξιολόγησης παρέχουν μία ολοκληρωμένη κατανόηση των αντιλήψεων που θα έχουν οι χρήστες σχετικά με την χρηστικότητα, την σχεδίαση και την συνολική λειτουργικότητα και αποδοτικότητα της πλατφόρμας. Η σταθερή παρουσία υψηλών βαθμολογιών σε όλες τις δηλώσεις δείχνει πως οι συμμετέχοντες συνολικά θεώρησαν το σύστημα ως αξιόπιστο και καλά δομημένο.

Η τιμή μέσου όρου 4,4 στο αν τους φάνηκε η εφαρμογή και η πλοήγηση της εύκολη, επιβεβαιώνει πως η διεπαφή είναι σχεδιασμένη με σκοπό το να γίνει οικεία γρήγορα. Παρομοίως, η διαδικασία καταχώρησης αναφοράς βαθμολογήθηκε υψηλά (μέσο όρο 4,5), αποδεικνύοντας πως ο μηχανισμός υποβολής θεωρήθηκε απλός και ταχύς. Αυτά τα ευρήματα, επικυρώνουν την εστίαση της εφαρμογής πάνω σε ένα σχεδιασμό με βάση τον χρήστη και τους απλούς τρόπους αλληλεπιδρασης.

Η οπτική και η αίσθηση της εφαρμογής επίσης αξιολογήθηκαν θετικά (4,1), με τους χρήστες να την θεωρούν ως καθαρή και ελκυστική. Ενώ το σκορ ήταν ελάχιστα χαμηλότερο από τα υπόλοιπα, παραμένει σε ένα δυνατό επίπεδο ικανοποίησης, χρειάζοντας πιθανά ένα

μικρό περιθώριο βελτίωσης.

Όσον αφορά την επικοινωνία και την ανατροφοδότηση, παρατηρείται μία συνολική συμφωνία (4.3) πως τα μηνύματα της εφαρμογής ήταν ξεκάθαρα και πως η απεικόνιση των αναφορών στον χάρτη είναι απλή και εύκολα κατανοητή. Αυτά τα αποτελέσματα δείχνουν την επιτυχής αρχιτεκτονική της πλατφόρμας ως προς την ικανότητα να εμφανίζει πληροφορίες στον χρήστη.

Τέλος, η συνολική εμπειρία και η πρόθεση για μελλοντική χρήση, είχαν από τα μεγαλύτερα αποτελέσματα, με μέσους όρους 4,5 και 4,7 αντίστοιχα. Η θετική ανταπόκριση υποδηλώνει υψηλό επίπεδο ικανοποίησης και εμπιστοσύνης σε μία τέτοια πλατφόρμα. Επιπρόσθετα, οι δηλώσεις πως θα πρότειναν την εφαρμογή σε άλλους (4.3) και πως αυτή βοηθάει στην επικοινωνία του πολίτη με τον δήμο (4.7), έφτασαν πολύ υψηλά επίπεδα συμφωνίας, τονίζοντας το πόσο αναγνωρίζεται η αξία ενός τέτοιου συστήματος στον πραγματικό κόσμο.

Συνοψίζοντας, τα αποτελέσματα συνολικά επιβεβαιώνουν πως η πλατφόρμα εκπληρώνει θετικά τους κύριους στόχους της. Παρόλο που υπάρχει παρουσία μικρών διαφοροποιήσεων σε κάποια σημεία, όπως η οπτική, αυτό είναι φυσιολογικό και παρέχει χρήσιμες πληροφορίες για μελλοντικές αλλαγές. Ακόμα, η πλήρης απουσία αρνητικών απαντήσεων υποδηλώνει ότι το σύστημα λειτούργησε χωρίς να παρουσιαστούν λειτουργικά προβλήματα και πως το αποτέλεσμα αποδείχθηκε σύμφωνο με τις αρχικές εκτιμήσεις.

Κεφάλαιο F

Συμπεράσματα και Μελλοντική Εργασία

F.1 Συμπεράσματα

Hολοκλήρωση της εργασίας σηματοδοτεί την επιτυχή σχεδίαση και ανάπτυξη ενός φημιακού συστήματος στοχευμένη να βελτιώσει τον τρόπο όπου οι δήμοι διαχειρίζονται και ενεργούν πάνω σε ζητήματα υποδομών αναφερόμενα από τους πολίτες. Κατά την διάρκεια της διαδικασίας, το έργο έδειξε πως σύγχρονες τεχνολογίες ιστού και κινητών μπορούν να χρησιμοποιηθούν για την επίλυση καυθημερινών προβλημάτων με ένα απλό και διαθέσιμο τρόπο. Συνδυάζοντας την εφαρμογή και την ιστοσελίδα, η πλατφόρμα προσφέρει ένα οργανωμένο τρόπο που συνδέει και τις δύο πλευρές του θέματος.

Μέσω της υλοποίησης του, επαληθεύεται πως μέχρι και μικρές κοινότητες μπορούν να επωφεληθούν από μίας ελαφριάς και ανοικτού κώδικα προσέγγισης για την ψηφιοποίηση των λειτουργιών τους χωρίς την ανάγκη πολύπλοκων συστημάτων που χρειάζονται μεγάλο αριθμό χρημάτων. Η επιλεγμένη ομάδα τεχνολογιών, η οποία αποτελείται από Ionic React, Node.js, Express και MongoDB, αποδείχθηκε πρωτική και υψηλά ικανή ώστε να αναπτυχθεί μία πλατφόρμα πολλαπλών διεπαφών. Η εφαρμογή για κινητά, δίνει με επιτυχία την δυνατότητα στους ανθρώπους να δημιουργήσουν, να περιγράψουν και να στείλουν αναφορές, ενώ η ιστοσελίδα επιτρέπει στους διαχειριστές να μπορούν να παρακολουθούν και να χειριστούν αυτές. Μαζί, δημιουργείται ένα πλήρες οικοσύστημα για αποδοτική αστική διαχείριση.

Από μία πιο γενική εικόνα, παρατηρείται και τονίζεται η σημαντικότητα του σχεδιασμού με προτεραιότητα στον χρήστη για λογισμικά δημόσιων υπηρεσιών. Η πλατφόρμα χτίστηκε με την απλοϊκότητα στο μυαλό, δίνοντας σημασία στην πρόσβαση και σαφήνεια της, ώστε να χρησιμοποιείται και από χρήστες χωρίς πλούσια τεχνολογική εμπειρία. Οι δοκιμές και τα αποτελέσματα των αξιολογήσεων σε σχέση με την εκτίμηση δείχνουν πως το σύστημα δουλεύει και αποδίδει ικανοποιητικά όσον αφορά την χρηστικότητα και την λειτουργία του. Άρα, τα αποτελέσματα αυτά επιβεβαιώνουν πως επιτεύχθηκαν οι αρχικοί στόχοι.

Συνολικά, η εργασία επιδεικνύει πως η τεχνολογία μπορεί να χρησιμοποιηθεί σωστά ως ένα εργαλείο. Η πλατφόρμα προσφέρει ένα πρακτικό παράδειγμα για το πως μία ψηφιακή μεταμόρφωση μπορεί να υλοποιηθεί και σε μικρότερα επίπεδα πόλεων. Τα αποτελέσματα και διδάγματα αυτής της διαδικασίας μπορούν να θεωρηθούν ως μία βάση για μελλοντικές βελτιώσεις και εξελίξεις συμβάλλοντας στο όραμα πιο έξυπνων και ενωμένων πόλεων.

F.2 Μελλοντική Εργασία

Παρόλο που η τρέχουσα κατάσταση της πλατφόρμας καλύβει με επιτυχία τους βασικούς της στόχους, υπάρχουν πολλά σημεία όπου μπορούν να βελτιωθούν και να επεκταθούν περαιτέρω στο μέλλον. Όσο η τεχνολογία εξελίσσεται, νέες ευκαιρίες εμφανίζονται όπου μπορούν κάνουν αυτό το σύστημα πιο ευφυής και προσαρμοσμένο στις ανάγκες των δήμων. Οι παρακάτω ιδέες αναφέρονται σαν πιθανές μελλοντικές κατευθύνσεις ανάπτυξης, στοχεύοντας στην ενίσχυση τόσο λειτουργικά όσο και χρηστικά.

Μία από τις πιο σημαντικές μελλοντικές βελτιώσεις μπορεί να θεωρηθεί η εισαγωγή των μοντέλων τεχνητής νοημοσύνης για την βοήθεια στην ανίχνευση και κατηγοριοποίηση των αναφορών. Για παράδειγμα, αναλύοντας τις φωτογραφίες που ανεβάζουν οι χρήστες, ένα σύστημα AI μπορεί να αναγνωρίσει αυτόματα το είδος του προβλήματος και να το ενσωματώσει στην σωστή κατηγορία. Αυτό μπορεί να σώσει χρόνο στους διαχειριστές αλλά και στους χρήστες κατά την υποβολή, κάνοντας την διαδικασία πιο αποδοτική. Επιπρόσθετα, μπορούν να εκπαιδευτούν αλγόριθμοι μηχανικής μάθησης ώστε να αναλύουν μοτίβα από τις αναφορές, βιοηθώντας τους δήμους να προβλέψουν συχνά προβλήματα ή περιοχές που χρειάζονται πιθανή συντήρηση.

Άλλο ένα σημαντικό βήμα για την καλύτερη προσβασιμότητα της εφαρμογής είναι η επέκταση σε άλλες πλατφόρμες και πιο συγκεκριμένα στις συσκευές iOS. Αυτή την στιγμή, η κινητή εφαρμογή αναπτύχθηκε και υλοποιήθηκε μόνο σε συσκευή Android. Με την επέκταση της υποστήριξης για τους χρήστες με iPhone, η εφαρμογή γίνεται προσβάσιμη σε μεγαλύτερο εύρος του πληθυσμού. Για να γίνει εφικτό αυτό απαιτείται ρύθμιση και ενημέρωση στον υπάρχοντα κώδικα της αφού το Ionic υποστηρίζει πολλαπλές πλατφόρμες χωρίς νέο κώδικα. Ακόμα, χρειάζονται περαιτέρω δοκιμές ώστε η τελική λειτουργία να είναι ίδια χωρίς κάποια προβλήματα όπως περιορισμοί της Apple.

Από την διοικητική πλευρά, οι μελλοντικές ενημερώσεις μπορούν να επικεντρωθούν στην προσθήκη περισσότερων εξελιγμένων εργαλείων. Αυτό θα μπορούσε να περιλαμβάνει λειτουργίες όπως πιο ανεπτυγμένο φιλτράρισμα και κατάταξη των προβλημάτων με βάση των δεδομένων τους. Ακόμα, μπορεί να υλοποιηθεί τρόπος εξαγωγής των δεδομένων σε υπολογιστικά φύλλα για σκοπούς τεχμηρίωσης και για την δημιουργία αυτόματων στατιστικών αναφορών. Άλλη μία βελτίωση μπορεί να είναι η ικανότητα ομαδοποίησης των αναφορών με βάση την γεωγραφική τοποθεσία καθώς και το είδος προβλήματος τους. Αυτό επιτρέπει στις πόλεις να οπτικοποιήσουν παρόμοια ή κοντινά θέματα ως συστάδες πάνω στο χάρτη, κάνοντας πιο εύκολη την αναγνώριση κρίσιμων περιοχών ώστε να σχεδιάσουν πιο στρατηγικά την προσέγγιση τους.

Επιπλέον, μπορούν να ενσωματωθούν ειδοποιήσεις σε πραγματικό χρόνο όπου ενδυναμώνει την επικοινωνιακή σχέση των χρηστών με τις αρχές και την εμπιστοσύνη τους στο σύστημα. Για παράδειγμα, οι πολίτες να δέχονται ενημερώσεις μέσω της εφαρμογής όταν αλλάζει η κατάσταση μίας αναφοράς τους και όταν αυτή ολοκληρωθεί, κάνοντας το σύστημα πιο ελκυστικό και αξιόπιστο.

Συνολικά, αυτές οι βελτιώσεις έχουν στόχο να κάνουν την εφαρμογή πιο δυνατή και πλούσια σε χαρακτηριστικά καθώς και πιο προσαρμόσιμη σε μελλοντικές αστικές προκλήσεις. Η τωρινή έκδοση της χρησιμεύει ως μία σταθερή και ισχυρή βάση για μελλοντικές αναπτύξεις, προσφέροντας τρόπους για συνεχή καινοτομία.

Bibliography

- Angelidis, M., & Drakouli, E. (2019). Έξυπνες και πράσινες πόλεις στην ευρωπαϊκή ένωση.
- Angelopoulos, C. M., Katos, V., Kostoulas, T., Miaoudakis, A., Petroulakis, N., Alexandris, G., Demetriou, G., Morandi, G., Rak, U., Waledzik, K., Panayiotou, M., & Tsatsoulis, C. I. (2019). Ideal-cities - a trustworthy and sustainable framework for circular smart cities. *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 443–450. <https://doi.org/10.1109/DCOSS.2019.00089>
- Anthopoulos, L. (2015). Defining smart city architecture for sustainability. In *Electronic government and electronic participation* (pp. 140–147). IOS Press.
- Arroub, A., Zahi, B., Sabir, E., & Sadik, M. (2016). A literature review on smart cities: Paradigms, opportunities and open problems. *2016 International conference on wireless networks and mobile communications (WINCOM)*, 180–186.
- Aslam, S., & Ullah, H. S. (2020). A comprehensive review of smart cities components, applications, and technologies based on internet of things. *arXiv preprint arXiv:2002.01716*.
- Bakowska-Waldmann, E., & Kaczmarek, T. (2021). The use of ppgis: Towards reaching a meaningful public participation in spatial planning. *ISPRS International Journal of Geo-Information*, 10(9), 581.
- Boulos, M., Tsouros, A., & Holopainen, A. (2015). Social, innovative and smart cities are happy and resilient. *International Journal of Health Geographics*, 14(3).
- Bovkir, R., & Aydinoglu, A. C. (2021). Big urban data visualization approaches within the smart city: Gis-based open-source dashboard example. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46, 125–130.

- Capacitor. (n.d.). Capacitor by ionic - cross-platform apps with web technology. <https://capacitorjs.com/>
- Chauhan, A. (2019). A review on various aspects of mongodb databases. *International Journal of Engineering Research & Technology (IJERT)*, 8(05), 90–92.
- CORS, E. (n.d.). Express cors middleware. <https://expressjs.com/en/resources/middleware/cors.html>
- Dunka, B., Emmanuel, E., & Oyerinde, Y. (2017). Hybrid mobile application based on ionic framework technologies. *International Journal of Recent Advances in Multidisciplinary Research*, 04, 3121–3130.
- ExpressJS. (n.d.). Node.js web application framework. <https://expressjs.com/>
- Goel, M., & Singal, G. (2021). Android os case study. *arXiv preprint arXiv:2104.09487*.
- Holmberg, O. (2023). Migrating from javascript to typescript and its advantages.
- Hou, J., Arpan, L., Wu, Y., Feiock, R., Ozguven, E., & Arghandeh, R. (2020). The road toward smart cities: A study of citizens' acceptance of mobile applications for city services. *Energies*, 13(10), 2496.
- Ionicframework. (n.d.). Introduction to ionic: Ionic documentation. <https://ionicframework.com/docs>
- Janoskova, P., Stofkova, K. R., Kovacikova, M., Stofkova, J., & Kovacikova, K. (2021). The concept of a smart city communication in the form of an urban mobile application. *Sustainability*, 13(17), 9703.
- Karampakkis, P., Ioakeimidou, D., Chatzimisios, P., & Tsintotas, K. A. (2025). A web-based application for smart city data analysis and visualization. *Future Internet*, 17(5), 217.
- Konbr, U. (2019). Smart sustainable cities—vision and reality. *Resourceedings*, 2(1), 101–127.
- Leaflet. (n.d.). An open-source javascript library for interactive maps. <https://leafletjs.com/>
- Macadar, M. A., Porto, J. B., & Luciano, E. (2016). Smart city: A rigorous literature review of the concept from 2000 to 2015. *Electronic Government and Electronic Participation*, 203–210.
- Majchrzak, T., & Grønli, T.-M. (2017). Comprehensive analysis of innovative cross-platform app development frameworks.
- Multer, E. (n.d.). Express multer middleware. <https://expressjs.com/en/resources/middleware/multer.html>

- NodeJS. (n.d.). Node.js - run javascript everywhere. <https://nodejs.org/en>
- React. (n.d.). The library for web and native user interfaces. <https://react.dev/>
- Sejati, A. W., Buchori, I., Rudiarto, I., Silver, C., & Sulistyo, K. (2020). Open-source web gis framework in monitoring urban land use planning: Participatory solutions for developing countries. *Journal of Urban & Regional Analysis*, 12(1).
- Shama, F., Aziz, A., & Deya, L. B. M. (2024). Citysolution: A complaining task distributive mobile application for smart city corporation using deep learning. *SoftwareX*, 27, 101829.
- Siokas, G., Τσακανίκας, Α., & Σιώκας, Ε. (2019). Αποτύπωση των στρατηγικών των ελληνικών ευφυών πόλεων: Μια εμπειρική ανάλυση.
- Sonar, C., & Dharmadhikari, M. (2023). Hybrid mobile app development using ionic framework. *International Journal of Advanced Research in Computer and Communication Engineering*, 12(6). <https://ijarcce.com/papers/hybrid-mobile-app-development-using-ionic-framework/>
- Sriramya, P., & Karthika, R. (2015). Providing password security by salted password hashing using bcrypt algorithm. *ARPJournal of engineering and applied sciences*, 10(13), 5551–5556.
- Thamizharasi, R. (2016). Android mobile application build on android studio. *International Journal of Modern Computer Science (IJMCS)*, 4(1), 1–4.
- TypeScript. (n.d.). Javascript with syntax for types. <https://www.typescriptlang.org/>
- You, D., & Hu, M. (2021). A comparative study of cross-platform mobile application development. *Wellington, New Zealand*, 66.
- Zhao, D., Petrov, A., Ivanov, D., Volkov, S., Smith, J., & Wang, J. (2025). Exploring geographic information systems (gis) for smart city analytics.

Συντμήσεις

API	Application Programming Interface
APK	Android Application Package
CDN	Content Delivery Network
CLI	Command-Line Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
GIS	Geographic Information System
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
JSON	JavaScript Object Notation
PPGIS	Public Participation Geographic Information System
PWA	Progressive Web App
SQL	Structured Query Language
UI	User Interface
UX	User Experience
EE	Ευρωπαϊκή Ένωση
TΠΕ	Τεχνολογίες Πληροφορικής και Επικοινωνιών

Γλωσσάρι Ξενικών Όρων

Admin	Διαχειριστής
App	Εφαρμογή
Backend	Λογισμικό υποσυστήματος
Body	Σώμα
Client	Συσκευή που υποβάλλει αιτήματα σε διαχομιστή
Cloud	Υπολογιστικό νέφος
Dashboard	Πίνακας ελέγχου
Form	Φόρμα
Framework	Πλαίσιο
Frontend	Περιβάλλον χρήστη
Geolocation	Γεωεντοπισμός
Map	Χάρτης
Marker	Δείκτης
Plugin	Πρόσθετο
Popup	Αναδυόμενο
Report	Αναφορά
Server	Διαχομιστής
Session	Συνεδρία
User	Χρήστης
Web	Ιστός