ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη συστήματος "Crowdsensing" για την συλλογική ανίχνευση συνθηκών πόλης μέσω κινητής εφαρμογής

(Development of a "Crowdsensing" system for the collective detection of city conditions via a mobile application)

Ιωάννης Τούμπανος

Επιβλέπων: **Θωμάς Λάγκας,** Αναπληρωτής Καθηγητής
Εξεταστική Επιτροπή: **Βασίλης Τσουκαλάς,** ΕΤΕΠ
Πέτρος Αμανατίδης, Διδάσκων με επαγγελματική εμπειρία

ΚΑΒΑΛΑ

ΟΚΤΩΒΡΙΟΣ 2026

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη συστήματος "Crowdsensing" για την συλλογική ανίχνευση συνθηκών πόλης μέσω κινητής εφαρμογής
(Development of a "Crowdsensing" system for the collective detection of city conditions via a mobile application)

Ιωάννης Τούμπανος
Α.Μ.: 5493

ΕΠΙΒΛΕΠΩΝ: **Θωμάς Λάγκας,** Αναπληρωτής Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Βασίλης Τσουκαλάς,** ΕΤΕΠ
Πέτρος Αμανατίδης, Διδάσκων με επαγγελματική εμπειρία

Copyright@ 2026 Τμήμα Πληροφορικής, Δημοκρίτειο Πανεπιστήμιο Θράκης, https://www.cs.duth.gr/

Το περιεχόμενο της συγκεκριμένης Πτυχιακής Εργασίας αποτελεί πνευματική ιδιοκτησία του/των συγγραφέα/ων, του/της επιβλέποντα/ουσας καθηγητή/τριας και του Τμήματος Πληροφορικής του ΔΠΘ και προστατεύεται από το νόμο περί πνευματικής ιδιοκτησίας (Νόμος 2121/1993 και κανόνες Διεθνούς Δικαίου που ισχύουν στην Ελλάδα).

ΔΗΛΩΣΗ ΤΗΡΗΣΗΣ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπόγραφα ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται με λεπτομέρεια στην πτυχιακή εργασία. Έχω αναφέρει πλήρως και με σαφείς αναφορές όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία, είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

ΕΥΧΑΡΙΣΤΙΕΣ

Με τη σειρά μου θα ήθελα να εκφράσω την ευχαρίστηση μου προς τον επιβλέποντα Καθηγητή κ. Θωμά Λάγκα για την υποστήριξη του και την καθοδήγηση του στην πτυχιακή εργασία.

ΠΕΡΙΛΗΨΗ

Η εξέλιξη των «έξυπνων πόλεων» βασίζεται σε πληθώρα πυκνών, πρόσφατων δεδομένων για την κατάσταση που επικρατεί στις αστικές περιοχές, τα οποία δεδομένα δεν έχουν πάντοτε την δυνατότητα με τα παραδοσιακά δίκτυα να παρέχουν το προαπαιτούμενο χρονικό και χωρικό εύρος. Σε αυτήν την πτυχιακή εργασία σχεδιάστηκε και υλοποιήθηκε μια εφαρμογή crowdsensing, η οποία με το δυναμικό των πολιτών, με την χρήση του κινητού τηλεφώνου του κάθε πολίτη, συλλέγει συνθήκες όχλησης σε αστικό περιβάλλον. Δημιουργήθηκε μια κινητή εφαρμογή τηλεφώνου, όπου μπορεί ένας χρήστης να καταγράφει ανώνυμα την γεωγραφική του θέση, να πραγματοποιεί μια μέτρηση θορύβου με την χρήση του μικροφώνου του κινητού καθώς και να αξιολογεί την αστική όχληση με 5 κατηγορίες. Οι αξιολογήσεις καθώς και οι μετρήσεις του κάθε χρήστη αποθηκεύονται στην πλατφόρμα της Google Firebase και ενεργοποιούνται αυτόματα οι συναρτήσεις Cloud Functions, οι οποίες υπολογίζουν μια τελική βαθμολογία όχλησης, ταξινομούν χωρικά τις καταχωρήσεις σε ένα γεωγραφικό πλέγμα και ενημερώνουν συλλογικά στατιστικά και σε περίπτωση πολλαπλών καταχωρήσεων από πολλούς χρήστες την ίδια χρονική στιγμή με ασφάλεια. Η εφαρμογή αναγνωρίζει αμέσως τα ήδη διαμορφωμένα δεδομένα, έτσι ώστε ο χρήστης να έχει την δυνατότητα να είναι ενημερωμένος για τις συνθήκες όχλησης που επικρατούν στην περιοχή του μέσω ψηφιακού χάρτη, με δυνατότητα προβολής επιμέρους σημείων ή θερμικού χάρτη, μέσω μιας οθόνης συλλογικών στατιστικών. Η επιλογή serverless αρχιτεκτονικής βοηθά να ξεχωρίζει καθαρά το κομμάτι της συλλογής από αυτό της επεξεργασίας των δεδομένων έτσι, δημιουργήθηκε ένα σύστημα που λειτουργεί άμεσα, μπορεί να επεκταθεί κι εύκολα αξιοποιείται για την παρακολούθηση των συνθηκών όχλησης μέσα στην πόλη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη Εφαρμογών Κινητών Συσκευών και Συστημάτων Cloud

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: crowdsensing, Flutter, Firebase, Cloud Functions, αστική όχληση, θερμικός χάρτης (heatmap), serverless αρχιτεκτονική

ABSTRACT

The development of “smart cities” requires dense, up-to-date data on urban conditions, which traditional networks of fixed sensors cannot always provide with the necessary spatial and temporal coverage. This thesis presents the design and implementation of a crowdsensing system that leverages the general public, equipped simply with their mobile phones, for the collective detection of nuisance conditions in an urban environment. A mobile application was developed in Flutter, through which the user, after being anonymously authenticated, records their geographic location, optionally a noise measurement taken via the device's microphone, and subjective ratings for five categories of urban nuisance. These reports are stored on the Firebase platform and automatically trigger a Cloud Function, which computes an overall pollution score, spatially clusters reports into a geographic grid, and updates aggregate statistics, safely even under concurrent submissions from multiple users. The application reads this already processed data directly, allowing the user to monitor the nuisance conditions of their area through an interactive map -- displaying either individual markers or an aggregate heatmap -- and through a statistics screen. The serverless architecture that was chosen allowed for a clear separation between data collection, on the mobile application side, and data processing, on the server side, resulting in a functional, scalable, and readily usable system for monitoring conditions in an urban environment.

SUBJECT AREA: Mobile Application and Cloud System Development
KEYWORDS: crowdsensing, Flutter, Firebase, Cloud Functions, urban nuisance, heatmap, serverless architecture

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ XIV
ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ XV
1Ο ΚΕΦΑΛΑΙΟ: ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ 1

1.1 CROWDSENSING – ΣΥΜΜΕΤΟΧΙΚΗ ΑΙΣΘΗΤΗΡΕΥΣΗ (PARTICIPATORY SENSING) 1

1.1.1 ΚΑΤΗΓΟΡΙΕΣ CROWDSENSING 1

1.1.2 ΕΦΑΡΜΟΓΕΣ CROWDSENSING ΣΕ ΑΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ 2

1.2 ΕΞΥΠΝΕΣ ΠΟΛΕΙΣ ΚΑΙ ΠΕΡΙΒΑΛΛΟΝΤΙΚΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗ (SMART CITIES) 2

1.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ CLOUD ΚΑΙ SERVERLESS (FIREBASE) 3

1.4 ΑΝΑΛΥΣΗ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ HEATMAPS 4

1.5 ΑΝΑΠΤΥΞΗ ΔΙΑΠΛΑΤΦΟΡΜΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (FLUTTER) 5

1.6 ΑΙΣΘΗΤΗΡΕΣ ΚΙΝΗΤΩΝ ΣΥΣΚΕΥΩΝ ΚΑΙ ΜΕΤΡΗΣΗ ΘΟΡΥΒΟΥ 5

2Ο ΚΕΦΑΛΑΙΟ: ΣΥΛΛΟΓΗ ΑΠΑΙΤΗΣΕΩΝ 6

2.1 ΕΙΣΑΓΩΓΗ ΣΤΗ ΣΥΛΛΟΓΗ ΑΠΑΙΤΗΣΕΩΝ 6

2.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ ΕΦΑΡΜΟΓΗΣ 6

2.3 ΜΗ ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ 7

2.4 ΑΠΑΙΤΗΣΕΙΣ SERVER-SIDE ΕΠΕΞΕΡΓΑΣΙΑΣ 8

3Ο ΚΕΦΑΛΑΙΟ: ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ 8

3.1 ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ 8

3.1.1 ΠΑΡΟΜΟΙΑ ΣΥΣΤΗΜΑΤΑ ΣΤΟ ΔΙΑΔΙΚΤΥΟ 10

3.1.2 ΚΟΙΝΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ 10

3.2 ΟΡΟΙ ΚΑΙ ΣΥΝΤΟΜΕΥΣΕΙΣ 10

3.2.1 ΑΝΑΦΟΡΑ (REPORT) 10

3.2.2 ΒΑΘΜΟΛΟΓΙΑ ΟΧΛΗΣΗΣ (POLLUTION SCORE) 11

3.2.3 ΚΕΛΙ ΠΛΕΓΜΑΤΟΣ ΚΑΙ ΘΕΡΜΙΚΟΣ ΧΑΡΤΗΣ (HEATMAP) 11

3.2.4 ΣΤΑΤΙΣΤΙΚΑ ΔΕΔΟΜΕΝΑ 11

3.3 ΛΕΙΤΟΥΡΓΙΚΗ ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ 11

3.3.1 ΑΝΩΝΥΜΗ ΤΑΥΤΟΠΟΙΗΣΗ ΧΡΗΣΤΩΝ 11

3.3.2 ΚΑΤΑΧΩΡΙΣΗ ΑΝΑΦΟΡΑΣ 12

3.3.3 ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ 12

3.3.4 ΧΑΡΤΗΣ ΚΑΙ ΣΤΑΤΙΣΤΙΚΑ 12
4Ο ΚΕΦΑΛΑΙΟ: ΠΕΡΙΓΡΑΦΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ 12
4.1 ΑΠΑΙΤΟΥΜΕΝΟΙ ΠΟΡΟΙ ΚΑΙ ΑΔΕΙΕΣ ΣΥΣΚΕΥΗΣ 12
4.2 ΑΝΤΙΜΕΤΩΠΙΣΗ ΣΦΑΛΜΑΤΩΝ ΚΑΙ ΑΝΑΚΤΗΣΗ 13
4.3 ΔΟΜΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ 13
4.4 ΑΞΙΟΠΙΣΤΙΑ ΕΦΑΡΜΟΓΗΣ 14
4.5 ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 14
4.6 ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ 14
4.6.1 ΔΟΜΗ ΦΑΚΕΛΩΝ ΤΗΣ FLUTTER ΕΦΑΡΜΟΓΗΣ 14
4.6.2 ΔΟΜΗ ΦΑΚΕΛΩΝ ΤΩΝ CLOUD FUNCTIONS 15
4.7 Η ΠΛΑΤΦΟΡΜΑ FIREBASE 16
5Ο ΚΕΦΑΛΑΙΟ: ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ (FLUTTER) 16
5.1 ΓΕΝΙΚΗ ΔΟΜΗ ΚΑΙ ΠΛΟΗΓΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 16
5.2 ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ (ΚΛΑΣΗ ENVIRONMENTALREPORT) 18
5.3 ΟΘΟΝΗ ΥΠΟΒΟΛΗΣ ΑΝΑΦΟΡΑΣ (SUBMITREPORTSCREEN) 22
5.4 ΟΘΟΝΗ ΧΑΡΤΗ (MAPSCREEN) 24
5.5 ΟΘΟΝΗ ΣΤΑΤΙΣΤΙΚΩΝ (STATISTICSSCREEN) 27
5.6 ΥΠΗΡΕΣΙΕΣ ΕΦΑΡΜΟΓΗΣ 29
5.6.1 FIREBASESERVICE 29
5.6.2 LOCATIONSERVICE 30
5.6.3 NOISESERVICE 30
5.7 ΓΕΩΓΡΑΦΙΚΟ ΦΙΛΤΡΑΡΙΣΜΑ ΣΤΑΤΙΣΤΙΚΩΝ ΑΝΑ ΝΟΜΟ 30
6Ο ΚΕΦΑΛΑΙΟ: ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ SERVER-SIDE ΕΠΕΞΕΡΓΑΣΙΑΣ (CLOUD FUNCTIONS) 31
6.1 ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ONREPORTCREATED 31
6.2 ΥΠΟΛΟΓΙΣΜΟΣ ΒΑΘΜΟΛΟΓΙΑΣ ΟΧΛΗΣΗΣ (POLLUTION SCORE) 32
6.3 ΧΩΡΙΚΗ ΟΜΑΔΟΠΟΙΗΣΗ ΚΑΙ ΕΝΗΜΕΡΩΣΗ ΠΛΕΓΜΑΤΟΣ (HEATMAP_CELLS) 33
6.4 ΕΝΗΜΕΡΩΣΗ ΣΥΓΚΕΝΤΡΩΤΙΚΩΝ ΣΤΑΤΙΣΤΙΚΩΝ (STATS/SUMMARY, DAILY_STATS) 34

6.5 ΑΤΟΜΙΚΟΤΗΤΑ ΚΑΙ ΣΥΝΑΛΛΑΓΕΣ (TRANSACTIONS) 35

6.6 ΑΝΑΠΤΥΞΗ ΚΑΙ ΠΑΡΑΚΟΛΟΥΘΗΣΗ 36

7Ο ΚΕΦΑΛΑΙΟ: ΔΟΚΙΜΕΣ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ 36

7.1 ΜΕΘΟΔΟΛΟΓΙΑ ΔΟΚΙΜΩΝ 36

7.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΔΟΚΙΜΕΣ 37

7.3 ΣΥΝΟΨΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΔΟΚΙΜΩΝ 40

7.4 ΔΟΚΙΜΗ ΤΑΥΤΟΧΡΟΝΗΣ ΠΡΟΣΒΑΣΗΣ ΠΟΛΛΑΠΛΩΝ ΧΡΗΣΤΩΝ (CONCURRENCY TEST) 40

8Ο ΚΕΦΑΛΑΙΟ: ΣΥΜΠΕΡΑΣΜΑΤΑ 43

8.1 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ 43

8.2 ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ 44

ΑΝΑΦΟΡΕΣ 45

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Εικόνα 1: Αρχιτεκτονική και ροή δεδομένων του συστήματος crowdsensing. 9

Εικόνα 2: Δομή φακέλων της Flutter εφαρμογής στο Visual Studio Code. 15

Εικόνα 3: Δομή φακέλου των Cloud Functions. 16

Εικόνα 4: Mockup της οθόνης υποβολής αναφοράς. 17

Εικόνα 5: Mockup της οθόνης χάρτη. 17

Εικόνα 6: Αρχική οθόνη της εφαρμογής με τη μπάρα πλοήγησης. 18

Εικόνα 7: Η κλάση EnvironmentalReport (lib/models/environmental_report.dart). 21

Εικόνα 8: Οθόνη υποβολής αναφοράς (SubmitReportScreen). 23

Εικόνα 9: Η μέθοδος _submit() στην κλάση SubmitReportScreen. 24

Εικόνα 10: Οθόνη χάρτη σε προβολή μεμονωμένων αναφορών (markers). 25

Εικόνα 11: Οθόνη χάρτη σε προβολή θερμικού χάρτη (heatmap). 26

Εικόνα 12: Εναλλαγή προβολής σημείων/θερμικού χάρτη στην κλάση MapScreen. 27

Εικόνα 13: Οθόνη στατιστικών (StatisticsScreen). 28

Εικόνα 14: Η μέθοδος submitReport() της κλάσης FirebaseService. 29

Εικόνα 15: Ορισμός της συνάρτησης onReportCreated (functions/index.js). 32

Εικόνα 16: Υπολογισμός αναγνωριστικού κελιού και ενημέρωση της συλλογής heatmap_cells. 34

Εικόνα 17: Χρήση συναλλαγής (transaction) της Cloud Firestore. 35

Εικόνα 18: Οι συλλογές της βάσης δεδομένων στην κονσόλα Firebase. 36

Εικόνα 19: Καταγραφή (log) επιτυχούς εκτέλεσης της συνάρτησης onReportCreated. 36

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1: Συλλογές της βάσης δεδομένων Firestore 13

Πίνακας 2: Πεδία της κλάσης EnvironmentalReport. 19

Πίνακας 3: Περιπτώσεις λειτουργικών δοκιμών (test cases) της εφαρμογής. 37

Πίνακας 4: Αποτελέσματα δοκιμής ταυτόχρονης πρόσβασης πολλαπλών χρηστών. 41

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Crowdsensing	Συλλογική/πληθοποριστική αισθητήρευση
Participatory sensing	Συμμετοχική αισθητήρευση
Opportunistic sensing	Ευκαιριακή αισθητήρευση
Smart city	Έξυπνη πόλη
Serverless	Αρχιτεκτονική χωρίς διαχείριση διακομιστή
Backend-as-a-Service (BaaS)	Υπηρεσία υποστήριξης εξυπηρετητή ως υπηρεσία
Heatmap	Θερμικός χάρτης
Spatial data	Χωρικά δεδομένα
Grid clustering	Ομαδοποίηση σε γεωγραφικό πλέγμα

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

GPS	Global Positioning System
IoT	Internet of Things
API	Application Programming Interface
UI	User Interface
UX	User Experience
JSON	JavaScript Object Notation
BaaS	Backend as a Service
IDE	Integrated Development Environment
SDK	Software Development Kit
ΔΠΘ	Δημοκρίτειο Πανεπιστήμιο Θράκης

1ο ΚΕΦΑΛΑΙΟ: ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

1.1 CROWDSENSING – ΣΥΜΜΕΤΟΧΙΚΗ ΑΙΣΘΗΤΗΡΕΥΣΗ (PARTICIPATORY SENSING)

Το Crowdsensing(mobile crowdsensing) συγκροτεί ένα μοντέλο συγκέντρωσης δεδομένων με ένα μεγάλο πλήθος συμμετεχόντων να αξιοποιεί τις δυνατότητες ενός έξυπνου κινητού τηλεφώνου με την βοήθεια των αισθητήρων(gps, επιταχυνσιόμετρο, μικρόφωνο κ.α.) για την συλλογή πληροφοριών σχετικά με το περιβάλλον τους[1][2]. Η έντονη εξάπλωση των έξυπνων κινητών, σε συνδυασμό με την ένταξη πολλαπλών αισθητήρων σε αυτά, παρέχει στον χρήστη την ικανότητα να προσφέρει σε μαζικά έργα παρακολούθησης χωρίς την ανάγκη εγκατάστασης πολυέξοδου και ειδικευμένου εξοπλισμού[3].

Το 2006 εισήγαγαν την συμμετοχική αισθητήρευση(participatory sensing) οι Burke κ.α. και γεννήθηκε η ιδέα του crowdsensing το οποίο έχει τις ρίζες του στη συμμετοχική αισθητήρευση [4]. Αξιοσημείωτη παρατήρηση ήταν ότι τα σημερινά κινητά τηλέφωνα προσφέρουν στον χρήστη μια μεγάλη ποικιλία αισθητήρων όπως gps, επιταχυνσιόμετρο, μικρόφωνο, κάμερα κ.ά., καθώς βρίσκονται στην κατοχή ενός μεγάλου μέρους του πληθυσμού, που έχουν την δυνατότητα να λειτουργούν ως ένα μεγάλο κατανεμημένο δίκτυο αισθητήρων που καλύπτουν την πόλη ολόκληρη, δίχως το κόστος εγκατάστασης και συντήρησης σταθερών υποδομών και έχει ως αποτέλεσμα η συλλογή δεδομένων από τους χρήστες να παρέχει μια υπηρεσία.

Το crowdsensing αποτελεί μια λύση για προβλήματα σε αστικό επίπεδο. Παρά τα πλεονεκτήματα αυτά, πρέπει να σημειωθεί ότι η ποιότητα των δεδομένων ποικίλλει, διότι οι μετρήσεις έρχονται από ανομοιογενείς και μη ειδικευμένους χρήστες και χρειάζεται τα δεδομένα να ελέγχονται από μηχανισμούς ελέγχου και φιλτραρίσματος. Παράλληλα, τίθεται ζήτημα η προστασία της ιδιωτικότητας του χρήστη, καθώς τα δεδομένα συνοδεύονται από την γεωγραφική θέση των χρηστών[5]. Ορισμένες από τις σχεδιαστικές αποφάσεις της παρούσας εργασίας —όπως η ανώνυμη συμμετοχή και η παρουσίαση των δεδομένων σε συγκεντρωτική μορφή— στοχεύουν ακριβώς στην αντιμετώπιση μέρους αυτών των προκλήσεων.

1.1.1 ΚΑΤΗΓΟΡΙΕΣ CROWDSENSING

Με βάση την βιβλιογραφία εντοπίζονται δύο βασικές κατηγορίες crowdsensing συστημάτων:η ευκαιριακή αισθητήρευση (opportunistic sensing), κατά την οποία τα

δεδομένα της συλλέγονται αυτόματα από τους αισθητήρες της συσκευής χωρίς άμεση εμπλοκή του χρήστη, και η συμμετοχική αισθητήρευση (participatory sensing), κατά την οποία ο χρήστης συμμετέχει ενεργά στην διαδικασία συλλογής, καταγράφοντας ο ίδιος μια μέτρηση ή μια υποκειμενική αξιολόγηση[1]. Το σύστημα που υλοποιείται στην παρούσα εργασία εντάσσεται στην δεύτερη κατηγορία, καθώς ο χρήστης καταγράφει ενεργά τόσο αντικειμενικές μετρήσεις, όπως η ένταση του θορύβου μέσω του μικροφώνου της συσκευής, όσο και υποκειμενικές αξιολογήσεις, όπως η ένταση δυσοσμίας ή η θερμική δυσφορία.

1.1.2 ΕΦΑΡΜΟΓΕΣ CROWDSENSING ΣΕ ΑΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

Το crowdsensing χρησιμοποιείται για την παρακολούθηση περιβαλλοντικών συνθηκών σε αστικά περιβάλλοντα, όπως στην χαρτογράφηση της ποιότητας του αέρα, του θορύβου και της κυκλοφοριακής συμφόρησης. Αξίζει επίσης να σημειωθεί ότι ένα χαρακτηριστικό παράδειγμα είναι το σύστημα NoiseTube[6], το οποίο επιτρέπει σε χρήστες έξυπνων τηλεφώνων να μετρούν το επίπεδο θορύβου μέσω του μικροφώνου της συσκευής τους και να συνεισφέρουν σε έναν συλλογικό χάρτη θορύβου. Η παρούσα εργασία υιοθετεί ανάλογη φιλοσοφία, επεκτείνοντάς την ώστε να καλύπτει πολλαπλές διαστάσεις αστικής όχλησης — θόρυβο, οσμή, θερμική άνεση, ποιότητα αέρα και οπτική ρύπανση — μέσα από ένα ενιαίο σύστημα.

1.2 ΕΞΥΠΝΕΣ ΠΟΛΕΙΣ ΚΑΙ ΠΕΡΙΒΑΛΛΟΝΤΙΚΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗ (SMART CITIES)

Τα αστικά περιβάλλοντα που αναφέρονται με τον όρο <<έξυπνη πόλη>> αξιοποιούν τις τεχνολογίες επικοινωνιών και πληροφορικής για την καλυτέρευση της ποιότητας ζωής των πολιτών, την υποστήριξη τεκμηριωμένης λήψης αποφάσεων από τους αρμόδιους φορείς και την αποδοτικότερη διαχείριση των πόρων[7]. Κύριο χαρακτηριστικό μιας έξυπνης πόλης αποτελεί η υποδομή συλλογής δεδομένων σε πραγματικό χρόνο, ενώ μέχρι τώρα η συλλογή δεδομένων γινόταν μέσω δικτύων αισθητήρων εγκατεστημένα σε ολόκληρη την πόλη, γνωστά ως Διαδίκτυο των πραγμάτων(Internet of Things-IoT). Το crowdsensing προσφέρει μια εναλλακτική, χαμηλού κόστους προσέγγιση, αξιοποιώντας τις ήδη υπάρχουσες συσκευές των πολιτών αντί για εξειδικευμένο εξοπλισμό, γεγονός που διευκολύνει σημαντικά την κλιμάκωση ενός τέτοιου συστήματος σε μεγάλες γεωγραφικές περιοχές.

Ιδιαίτερη αξία αποκτούν τα δεδομένα που παράγονται από τους ίδιους τους χρήστες στο πλαίσιο των έξυπνων πόλεων. Αντίθετα, σε σύγκριση με τα δίκτυα σταθερών

αισθητήρων του IoT(Internet of Things), τα οποία απαιτούν επένδυση σε εξοπλισμό και συνεχή συντήρηση [7], η χρήση των κινητών τηλεφώνων των πολιτών δίνει την δυνατότητα στην πύκνωση της χρονικής και χωρικής κάλυψης με χαμηλό κόστος. Η εναρμόνιση αυτή εντάσσεται στο ευρύτερο πεδίο του αστικού υπολογισμού (Urban Computing), το οποίο μελετά πώς τα μεγάλα σύνολα αστικών δεδομένων μπορούν να αξιοποιηθούν για την κατανόηση και τη βελτίωση της λειτουργίας των πόλεων [8]. Η καταγραφή της ηχορύπανσης, η περιβαλλοντική παρακολούθηση και άλλες μορφές όχλησης αποτελούν μια από τις πλέον διαδεδομένες εφαρμογές αυτής της λογικής

1.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ CLOUD ΚΑΙ SERVERLESS (FIREBASE)

Η δημιουργία συστημάτων crowdsensing απαιτεί μια υποδομή που να έχει την δυνατότητα, να αποθηκεύει και να επεξεργάζεται ένα μεγάλο όγκο πληροφοριών που αποστέλλονται ασύγχρονα από ένα πλήθος κινητών τηλεφώνων. Η αρχιτεκτονική serverless είναι μια πιο σύγχρονη προσέγγιση της ανάπτυξης των εφαρμογών, στην οποία ο προγραμματιστής αναθέτει την διαχείριση των εξυπηρετητών(servers) σε έναν τρίτο πάροχο υπηρεσιών cloud, ο οποίος τίθεται να αναλάβει την διαθεσιμότητα, την αυτόματη κλιμάκωση καθώς και την ασφάλεια της υποδομής. Η προσέγγιση αυτή είναι γνωστή ως Backend-as-a-Service(BaaS). Η πλατφόρμα της Google που χρησιμοποιήθηκε είναι η Firebase η οποία είναι μια διαδεδομένη λύση για serverless και προσφέρει μια NoSql βάση δεδομένων σε πραγματικό χρόνο(Cloud Firestore), συναρτήσεις που εκτελούνται αυτόματα ως απόκριση σε συμβάντα(Cloud Functions) και υπηρεσίες αυθεντικοποίησης χρηστών(Firebase Authentication). Αυτές οι τρεις υπηρεσίες χρησιμοποιούνται στην παρούσα εργασία, όπως αναλύονται αναλυτικά στα Κεφάλαια 4 και 6.

Σύμφωνα με τον ορισμό του NIST, το υπολογιστικό νέφος (cloud computing) είναι ένα μοντέλο που επιτρέπει την κατ' απαίτηση (on-demand) πρόσβαση σε ένα κοινό σύνολο διαμορφώσιμων υπολογιστικών πόρων, οι οποίοι μπορούν να δεσμεύονται και να αποδεσμεύονται γρήγορα με ελάχιστη διαχειριστική προσπάθεια [9]. Πάνω σε αυτό το μοντέλο έχει αναπτυχθεί η αρχιτεκτονική serverless, στην οποία ο προγραμματιστής δεν διαχειρίζεται καθόλου τους διακομιστές: γράφει μεμονωμένες συναρτήσεις (Functions as a Service) που εκτελούνται από τον πάροχο του νέφους μόνο όταν ενεργοποιηθούν από κάποιο συμβάν, ενώ η κλιμάκωση και η διαθεσιμότητα εξασφαλίζονται αυτόματα [10].

Το βασικό πλεονέκτημα του μοντέλου serverless είναι ότι το κόστος συνδέεται αποκλειστικά με την πραγματική χρήση —δεν υπάρχει διακομιστής που να λειτουργεί, και να χρεώνεται, συνεχώς— και ότι απαλλάσσει τον προγραμματιστή από τη διαχείριση της υποδομής [11]. Η αρχιτεκτονική αυτή είναι κατάλληλη για ένα σύστημα crowdsensing, όπου ο όγκος εργασίας είναι δύσκολα προβλέψιμος και εγγενώς μεταβλητός. Στην κατηγορία των μη σχεσιακών βάσεων δεδομένων (NoSql) ανήκει η Cloud Firestore, η οποία σε σύγκριση με τις παραδοσιακές σχεσιακές βάσεις, δεν διαχειρίζεται τα δεδομένα σε πίνακες με αυστηρό σχήμα, αλλά σε πιο ελαστικές δομές, εν προκειμένω, σε συλλογές (Collections) εγγράφων (Documents) μορφής JSON.

. Το μοντέλο αυτό προσφέρει ευκολία στην οριζόντια κλιμάκωση και ευελιξία στην εξέλιξη του σχήματος, χαρακτηριστικά που ταιριάζουν σε εφαρμογές με μεγάλο όγκο ημιδομημένων δεδομένων που καταφθάνουν ασύγχρονα [12].

1.4 ΑΝΑΛΥΣΗ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ HEATMAPS

Τα δεδομένα που συλλέγονται από συστήματα crowdsensing συνοδεύονται πάντοτε από γεωγραφικές συντεταγμένες (γεωγραφικό πλάτος και μήκος), γεγονός που τα καθιστά χωρικά δεδομένα (spatial data). Η ανάλυση χωρικών δεδομένων επιτρέπει την ανίχνευση προτύπων και τάσεων που δεν είναι εμφανή όταν τα δεδομένα εξετάζονται μεμονωμένα, όπως, για παράδειγμα, ο εντοπισμός περιοχών με συστηματικά αυξημένα επίπεδα όχλησης. Μία από τις πλέον διαδεδομένες τεχνικές οπτικοποίησης χωρικών δεδομένων είναι ο θερμικός χάρτης (heatmap), στον οποίο κάθε γεωγραφική περιοχή αποδίδεται με χρώμα ανάλογα με την τιμή ενός δείκτη, όπως η μέση όχληση. Στην παρούσα εργασία, η δημιουργία του heatmap βασίζεται σε ομαδοποίηση των αναφορών ανά κελί ενός κανονικού γεωγραφικού πλέγματος (grid clustering) και στον υπολογισμό του μέσου όρου ανά κελί, μια προσέγγιση που περιγράφεται αναλυτικά στο Κεφάλαιο 6.

Η ανάλυση χωρικών δεδομένων αποτελεί αντικείμενο των Γεωγραφικών Συστημάτων Πληροφοριών (GIS) και περιλαμβάνει, μεταξύ άλλων, τεχνικές χωρικής ομαδοποίησης (spatial aggregation) [13]. Μία απλή και υπολογιστικά αποδοτική τεχνική είναι η διαίρεση του χώρου σε ένα κανονικό πλέγμα (grid) από κελιά σταθερού μεγέθους και η άθροιση των μετρήσεων που εμπίπτουν σε κάθε κελί. Η προσέγγιση αυτή, που υιοθετείται και στην παρούσα εργασία, μετατρέπει ένα πλήθος μεμονωμένων και διάσπαρτων σημείων σε μια συνεχή, ευανάγνωστη επιφάνεια τιμών.

Η οπτικοποίηση των συγκεντρωτικών αυτών τιμών γίνεται συνήθως με τη μορφή θερμικού χάρτη (heatmap), μιας αναπαράστασης στην οποία η ένταση ενός μεγέθους αποδίδεται με διαβαθμίσεις χρώματος [14]. Ο θερμικός χάρτης επιτρέπει στον χρήστη να εντοπίζει με μια ματιά τις περιοχές υψηλής όχλησης, χωρίς να χρειάζεται να ερμηνεύσει μεμονωμένες αριθμητικές τιμές, και για τον λόγο αυτό χρησιμοποιείται ευρέως σε εφαρμογές περιβαλλοντικής παρακολούθησης.

1.5 ΑΝΑΠΤΥΞΗ ΔΙΑΠΛΑΤΦΟΡΜΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (FLUTTER)

Για την ανάπτυξη κινητών εφαρμογών υπάρχουν δύο βασικές προσεγγίσεις. Η εγγενής (native) ανάπτυξη προϋποθέτει τη συγγραφή ξεχωριστού κώδικα για κάθε λειτουργικό σύστημα (Android, iOS), ενώ η διαπλατφορμική (cross-platform) ανάπτυξη επιτρέπει τη συγγραφή ενός ενιαίου κώδικα ο οποίος εκτελείται σε πολλές πλατφόρμες. Η δεύτερη προσέγγιση μειώνει σημαντικά τον χρόνο και το κόστος ανάπτυξης και συντήρησης, γι' αυτό και προτιμήθηκε στην παρούσα εργασία.

Το Flutter είναι ένα ανοικτού κώδικα framework διαπλατφορμικής ανάπτυξης της Google, το οποίο χρησιμοποιεί τη γλώσσα προγραμματισμού Dart [15]. Το βασικό του χαρακτηριστικό είναι ότι η διεπαφή χρήστη δομείται εξ ολοκλήρου από «widgets»: κάθε στοιχείο της οθόνης (ένα κουμπί, ένα κείμενο, μια διάταξη) είναι ένα widget, και τα widgets συντίθενται ιεραρχικά ώστε να σχηματίσουν πιο σύνθετες οθόνες. Σε αντίθεση με άλλες προσεγγίσεις, το Flutter δεν βασίζεται στα εγγενή στοιχεία διεπαφής κάθε πλατφόρμας, αλλά σχεδιάζει το ίδιο τα γραφικά μέσω μιας δικής του μηχανής απόδοσης, εξασφαλίζοντας πανομοιότυπη εμφάνιση και υψηλή απόδοση σε όλες τις συσκευές [16]. Για τους λόγους αυτούς επιλέχθηκε ως το εργαλείο ανάπτυξης της κινητής εφαρμογής της παρούσας εργασίας.

1.6 ΑΙΣΘΗΤΗΡΕΣ ΚΙΝΗΤΩΝ ΣΥΣΚΕΥΩΝ ΚΑΙ ΜΕΤΡΗΣΗ ΘΟΡΥΒΟΥ

Η λειτουργία ενός συστήματος crowdsensing στηρίζεται στους αισθητήρες που ενσωματώνουν οι κινητές συσκευές. Ο σημαντικότερος για την παρούσα εφαρμογή είναι ο δέκτης GPS (Global Positioning System), ο οποίος προσδιορίζει τη γεωγραφική θέση (πλάτος και μήκος) της συσκευής και επιτρέπει την αντιστοίχιση κάθε αναφοράς με ένα συγκεκριμένο σημείο του χάρτη. Η ακρίβεια του εντοπισμού εξαρτάται από παράγοντες όπως η ορατότητα των δορυφόρων, γι' αυτό και προβλέπονται εναλλακτικοί μηχανισμοί για τις περιπτώσεις όπου το σήμα είναι ασθενές.

Ο δεύτερος αισθητήρας που αξιοποιείται είναι το μικρόφωνο, μέσω του οποίου η εφαρμογή εκτιμά τη στάθμη του περιβαλλοντικού θορύβου. Η ένταση του ήχου μετριέται σε ντεσιμπέλ (dB), μια λογαριθμική μονάδα κατάλληλη για το ευρύ φάσμα της ανθρώπινης ακοής. Η χρήση του μικροφώνου του κινητού για τη χαρτογράφηση του αστικού θορύβου έχει μελετηθεί εκτενώς στη βιβλιογραφία, σε συστήματα όπως το Ear-Phone [17] και το NoiseSPY [18]. Τα συστήματα αυτά κατέδειξαν ότι, παρά τους περιορισμούς ακρίβειας των μη βαθμονομημένων μικροφώνων, οι συλλογικές μετρήσεις πλήθους χρηστών μπορούν να δώσουν μια αξιόπιστη συνολική εικόνα της ηχορύπανσης μιας περιοχής.

2ο ΚΕΦΑΛΑΙΟ: ΣΥΛΛΟΓΗ ΑΠΑΙΤΗΣΕΩΝ

2.1 ΕΙΣΑΓΩΓΗ ΣΤΗ ΣΥΛΛΟΓΗ ΑΠΑΙΤΗΣΕΩΝ

Οι απαιτήσεις του συστήματος προέκυψαν από την επίσημη εκφώνηση της πτυχιακής εργασίας, όπως δόθηκε από τον επιβλέποντα Καθηγητή κ. Θωμά Λάγκα, η οποία περιγράφει ένα σύστημα crowdsensing αποτελούμενο από δύο βασικά υποσυστήματα: μια κινητή εφαρμογή για τη συλλογή δεδομένων από τους χρήστες και μια υποδομή στην πλευρά του server για την αποθήκευση και τη βασική επεξεργασία των δεδομένων αυτών. Με βάση την εκφώνηση, οι απαιτήσεις αναλύθηκαν και ομαδοποιήθηκαν σε λειτουργικές απαιτήσεις της εφαρμογής (Ενότητα 2.2), μη λειτουργικές απαιτήσεις (Ενότητα 2.3) και απαιτήσεις της επεξεργασίας στην πλευρά του server (Ενότητα 2.4), οι οποίες αποτέλεσαν τη βάση για την ανάλυση συστήματος που ακολουθεί στο Κεφάλαιο 3.

2.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ ΕΦΑΡΜΟΓΗΣ

Με βάση την εκφώνηση, η κινητή εφαρμογή θα πρέπει να υποστηρίζει τις ακόλουθες λειτουργίες:

- Ανώνυμη ταυτοποίηση του χρήστη κατά την πρώτη εκκίνηση της εφαρμογής, χωρίς την ανάγκη δημιουργίας λογαριασμού.
- Λήψη της τρέχουσας γεωγραφικής θέσης (γεωγραφικό πλάτος/μήκος) του χρήστη μέσω GPS, κατόπιν σχετικής άδειας.
- Μέτρηση της στάθμης θορύβου (σε dB) μέσω του μικροφώνου της συσκευής και αυτόματη ταξινόμησή της σε επίπεδο θορύβου.

- Καταγραφή υποκειμενικών αξιολογήσεων από τον χρήστη για πέντε κατηγορίες αστικής όχλησης — επίπεδο θορύβου, δυσοσμία, θερμική δυσφορία, ποιότητα αέρα και οπτική ρύπανση — καθώς και προαιρετικών ελεύθερων σημειώσεων.
- Υποβολή της αναφοράς (report), η οποία περιλαμβάνει τη θέση, τη χρονοσήμανση, τις μετρήσεις και τις αξιολογήσεις, στην κεντρική βάση δεδομένων.
- Προβολή χάρτη με τις μεμονωμένες αναφορές ως σημεία (markers), χρωματισμένα ανάλογα με τον βαθμό όχλησης της κάθε αναφοράς.
- Προβολή θερμικού χάρτη (heatmap) που συνοψίζει οπτικά τη συνολική όχληση ανά γεωγραφική περιοχή.
- Προβολή συγκεντρωτικών στατιστικών (κατανομές ανά κατηγορία αξιολόγησης, μέσος όρος θορύβου, τάση πλήθους αναφορών των τελευταίων ημερών).

2.3 ΜΗ ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ

Πέρα από τις λειτουργικές απαιτήσεις, τέθηκαν και οι εξής μη λειτουργικές απαιτήσεις:

- Ενημέρωση σε πραγματικό χρόνο: τόσο ο χάρτης όσο και τα στατιστικά ενημερώνονται αυτόματα μόλις προστεθούν νέα δεδομένα, χωρίς χειροκίνητη ανανέωση.
- Χαμηλό κόστος λειτουργίας: η υποδομή του server πρέπει να είναι οικονομικά προσιτή για έναν φοιτητή, αξιοποιώντας το πρότυπο πληρωμής pay-as-you-go με σαφές όριο προϋπολογισμού.
- Προστασία προσωπικών δεδομένων: δεν συλλέγονται προσωπικά στοιχεία των χρηστών· η ταυτοποίηση γίνεται ανώνυμα.
- Ανθεκτικότητα σε ασθενές σήμα GPS: σε περίπτωση αδυναμίας άμεσου εντοπισμού θέσης, η εφαρμογή επιχειρεί εναλλακτικά τη χρήση της τελευταίας γνωστής θέσης αντί να αποτυγχάνει.
- Απλότητα εγκατάστασης και χρήσης: η εφαρμογή απευθύνεται σε απλούς πολίτες και όχι σε εξειδικευμένο προσωπικό, άρα η διεπαφή χρήστη πρέπει να είναι απλή και κατανοητή.
- Επεκτασιμότητα: η αρχιτεκτονική πρέπει να μπορεί να υποστηρίξει αυξανόμενο αριθμό χρηστών και αναφορών χωρίς αλλαγές στην υποδομή, αξιοποιώντας την αυτόματη κλιμάκωση της αρχιτεκτονικής serverless.

2.4 ΑΠΑΙΤΗΣΕΙΣ SERVER-SIDE ΕΠΕΞΕΡΓΑΣΙΑΣ

Η εκφώνηση της εργασίας θέτει ρητά την απαίτηση για «βασική επεξεργασία δεδομένων» στην πλευρά του server, πέραν της απλής αποθήκευσης. Με βάση αυτό, διαμορφώθηκαν οι εξής απαιτήσεις:

- Αυτόματη ενεργοποίηση της επεξεργασίας αμέσως μόλις καταχωρίζεται στη βάση δεδομένων μια νέα αναφορά, χωρίς παρέμβαση του χρήστη ή προγραμματισμένη εκτέλεση σε τακτά χρονικά διαστήματα.
- Ανίχνευση χωρικών προτύπων, δηλαδή εντοπισμός γεωγραφικών περιοχών με συστηματικά αυξημένη όχληση, μέσω ομαδοποίησης των αναφορών σε ένα κανονικό γεωγραφικό πλέγμα.
- Υπολογισμός συγκεντρωτικών στατιστικών (κατανομές ανά κατηγορία αξιολόγησης, μέσος όρος στάθμης θορύβου σε dB).
- Υπολογισμός τάσεων στον χρόνο, μέσω καταμέτρησης του πλήθους αναφορών ανά ημέρα.
- Διασφάλιση της συνέπειας των δεδομένων ακόμη και σε περίπτωση ταυτόχρονων αναφορών από πολλούς χρήστες.

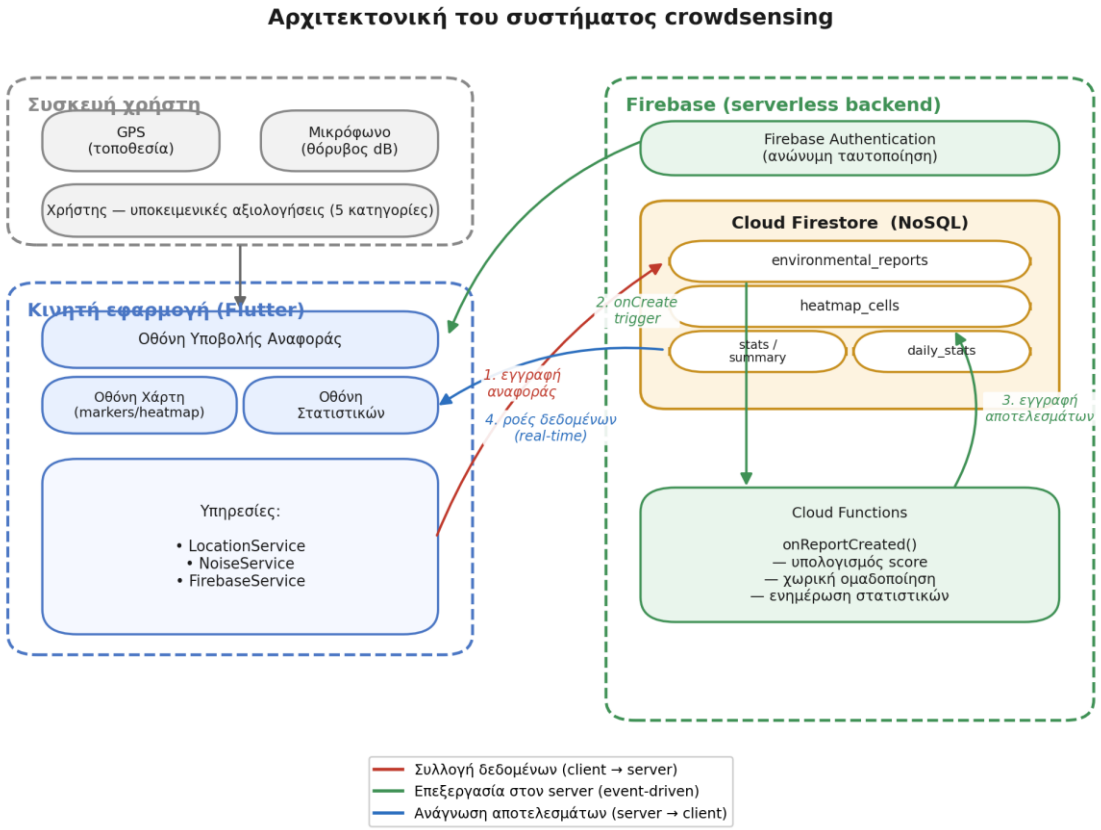
3ο ΚΕΦΑΛΑΙΟ: ΑΝΑΛΥΣΗ ΣΥΣΤΗΜΑΤΟΣ

3.1 ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ

Το σύστημα που αναπτύχθηκε αποτελείται από δύο βασικά υποσυστήματα, τα οποία επικοινωνούν μέσω της πλατφόρμας Firebase: μια κινητή εφαρμογή (Android, υλοποιημένη σε Flutter/Dart), μέσω της οποίας οι χρήστες καταγράφουν και υποβάλλουν αναφορές, και μια σειρά από συναρτήσεις στην πλευρά του server (Cloud Functions), οι οποίες επεξεργάζονται αυτόματα κάθε νέα αναφορά. Η ροή δεδομένων του συστήματος έχει ως εξής: ο χρήστης καταγράφει μια αναφορά μέσω της εφαρμογής, η οποία αποθηκεύεται απευθείας στη συλλογή environmental_reports της βάσης δεδομένων Cloud Firestore, μαζί με τη γεωγραφική της θέση και τη χρονοσήμανση. Η αποθήκευση αυτής της νέας αναφοράς ενεργοποιεί αυτόματα μια συνάρτηση Cloud Function (onReportCreated), η οποία υπολογίζει μια βαθμολογία όχλησης για την αναφορά, ενημερώνει το αντίστοιχο κελί ενός γεωγραφικού πλέγματος στη συλλογή heatmap_cells, ενημερώνει τα συνολικά στατιστικά στη συλλογή stats και το πλήθος αναφορών της ημέρας στη συλλογή daily_stats. Τέλος, η εφαρμογή διαβάζει ζωντανά αυτά τα ήδη επεξεργασμένα δεδομένα, ώστε ο χάρτης και τα στατιστικά να

ενημερώνονται αυτόματα, χωρίς η ίδια η εφαρμογή να χρειάζεται να επαναλαμβάνει τους υπολογισμούς. Η αρχιτεκτονική αυτή ικανοποιεί άμεσα τη διπλή απαίτηση της εκφώνησης: αφενός τη συλλογή δεδομένων από την κινητή εφαρμογή, αφετέρου την αποθήκευση και τη βασική επεξεργασία τους σε ένα σύστημα cloud.

Η συνολική αρχιτεκτονική του συστήματος, καθώς και η ροή των δεδομένων μεταξύ της κινητής εφαρμογής και της πλευράς του server, αποτυπώνονται συνοπτικά στην Εικόνα 1.



Εικόνα 1: Αρχιτεκτονική και ροή δεδομένων του συστήματος crowdsensing.

Όπως φαίνεται στην Εικόνα 1, η κινητή εφαρμογή είναι υπεύθυνη αποκλειστικά για τη συλλογή των δεδομένων (γεωγραφική θέση, μέτρηση θορύβου, υποκειμενικές αξιολογήσεις) και για την παρουσίαση των αποτελεσμάτων στον χρήστη. Κάθε νέα αναφορά που εγγράφεται στη συλλογή environmental_reports της Cloud Firestore ενεργοποιεί αυτόματα τις συναρτήσεις Cloud Functions, οι οποίες εκτελούν την επεξεργασία (υπολογισμό βαθμολογίας όχλησης, χωρική ομαδοποίηση σε πλέγμα και ενημέρωση συγκεντρωτικών στατιστικών) και εγγράφουν τα αποτελέσματα πίσω στη βάση. Από εκεί, η εφαρμογή τα διαβάζει σε πραγματικό χρόνο, χωρίς η ίδια να επιβαρύνεται με οποιαδήποτε επεξεργασία.

3.1.1 ΠΑΡΟΜΟΙΑ ΣΥΣΤΗΜΑΤΑ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

Συστήματα crowdsensing για την παρακολούθηση περιβαλλοντικών συνθηκών έχουν αναπτυχθεί και μελετηθεί εκτενώς τα τελευταία χρόνια [1]. Το πλέον χαρακτηριστικό παράδειγμα, στο οποίο βασίστηκε εν μέρει η προσέγγιση της παρούσας εργασίας, είναι το σύστημα NoiseTube [6], στο οποίο οι χρήστες μετρούν το επίπεδο θορύβου μέσω του μικροφώνου του κινητού τους και οι μετρήσεις τους αποτυπώνονται σε έναν συλλογικό χάρτη θορύβου. Αντίστοιχα συστήματα έχουν αναπτυχθεί και για άλλες περιβαλλοντικές παραμέτρους, όπως η ποιότητα του αέρα, ακολουθώντας κατά κανόνα την ίδια γενική αρχή: συλλογή γεωαναφερμένων μετρήσεων από πλήθος χρηστών και συγκεντρωτική οπτικοποίησή τους.

3.1.2 ΚΟΙΝΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Τα συστήματα αυτού του τύπου μοιράζονται συνήθως τα εξής κοινά χαρακτηριστικά: μια κινητή εφαρμογή-πελάτη, η οποία συλλέγει μετρήσεις ή αξιολογήσεις μαζί με τη γεωγραφική τους θέση· μια κεντρική υποδομή αποθήκευσης των δεδομένων· και μια οπτικοποίηση, συνήθως σε μορφή χάρτη, των συγκεντρωτικών αποτελεσμάτων. Η παρούσα εργασία διαφοροποιείται σε δύο σημεία: πρώτον, αντί να περιορίζεται σε μία μόνο παράμετρο (π.χ. μόνο θόρυβος), συνδυάζει πέντε διαφορετικές κατηγορίες αστικής όχλησης σε ένα ενιαίο σύστημα· δεύτερον, η επεξεργασία των δεδομένων δεν γίνεται σε προγραμματισμένα χρονικά διαστήματα (batch processing) αλλά αυτόματα, σε πραγματικό χρόνο, μέσω μιας αρχιτεκτονικής βασισμένης σε συμβάντα (event-driven), αξιοποιώντας τις δυνατότητες της αρχιτεκτονικής serverless που παρουσιάστηκε στην Ενότητα 1.3.

3.2 ΟΡΟΙ ΚΑΙ ΣΥΝΤΟΜΕΥΣΕΙΣ

Στην ενότητα αυτή ορίζονται συνοπτικά οι βασικοί όροι του συστήματος, όπως χρησιμοποιούνται στη συνέχεια της ανάλυσης.

3.2.1 ΑΝΑΦΟΡΑ (REPORT)

Ως αναφορά (report) ορίζεται η βασική μονάδα δεδομένων του συστήματος: ένα σύνολο πληροφοριών που καταχωρίζει ο χρήστης σε μία υποβολή, το οποίο περιλαμβάνει τη γεωγραφική θέση, τη χρονοσήμανση, τη μετρηθείσα στάθμη θορύβου (εφόσον μετρήθηκε), τις υποκειμενικές αξιολογήσεις για τις πέντε κατηγορίες όχλησης και προαιρετικές ελεύθερες σημειώσεις.

3.2.2 ΒΑΘΜΟΛΟΓΙΑ ΟΧΛΗΣΗΣ (POLLUTION SCORE)

Η βαθμολογία όχλησης (pollution score) είναι ένας ενιαίος αριθμητικός δείκτης, με τιμές από 0 έως 100, ο οποίος προκύπτει από τον μέσο όρο των πέντε επιμέρους αξιολογήσεων μιας αναφοράς, κανονικοποιημένων ως προς το εύρος τιμών της καθεμιάς. Όσο υψηλότερη είναι η τιμή, τόσο πιο επιβαρυμένες θεωρούνται οι συνθήκες που περιγράφει η αναφορά.

3.2.3 ΚΕΛΙ ΠΛΕΓΜΑΤΟΣ ΚΑΙ ΘΕΡΜΙΚΟΣ ΧΑΡΤΗΣ (HEATMAP)

Για την ανίχνευση χωρικών προτύπων, ο γεωγραφικός χώρος διαιρείται νοητά σε ένα κανονικό πλέγμα κελιών σταθερού μεγέθους. Κάθε αναφορά αντιστοιχίζεται στο κελί εντός του οποίου βρίσκεται η θέση της, με βάση τις συντεταγμένες της. Για κάθε κελί υπολογίζεται ο μέσος όρος της βαθμολογίας όχλησης όλων των αναφορών που έχουν αντιστοιχιστεί σε αυτό, καθώς και το πλήθος τους. Το σύνολο των κελιών, οπτικοποιημένο με χρωματική διαβάθμιση ανάλογη της μέσης βαθμολογίας τους, αποτελεί τον θερμικό χάρτη (heatmap) του συστήματος.

3.2.4 ΣΤΑΤΙΣΤΙΚΑ ΔΕΔΟΜΕΝΑ

Ως στατιστικά δεδομένα ορίζονται τα συγκεντρωτικά μεγέθη που υπολογίζονται από το σύνολο των αναφορών του συστήματος: το συνολικό πλήθος αναφορών, η κατανομή των αναφορών ανά κατηγορία αξιολόγησης, ο μέσος όρος της μετρηθείσας στάθμης θορύβου σε ντεσιμπέλ, καθώς και το πλήθος αναφορών ανά ημέρα για τις τελευταίες επτά ημέρες.

3.3 ΛΕΙΤΟΥΡΓΙΚΗ ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ

Στη συνέχεια αναλύονται λεπτομερέστερα οι βασικές λειτουργίες του συστήματος, όπως προέκυψαν από τις απαιτήσεις του Κεφαλαίου 2.

3.3.1 ΑΝΩΝΥΜΗ ΤΑΥΤΟΠΟΙΗΣΗ ΧΡΗΣΤΩΝ

Κατά την πρώτη εκκίνηση της εφαρμογής, ο χρήστης ταυτοποιείται αυτόματα και ανώνυμα μέσω της υπηρεσίας Firebase Authentication, χωρίς να απαιτείται η καταχώριση προσωπικών στοιχείων ή η δημιουργία λογαριασμού. Στον χρήστη αποδίδεται ένα μοναδικό αναγνωριστικό, το οποίο συνοδεύει τις αναφορές που υποβάλλει, χωρίς ωστόσο να αποκαλύπτει την ταυτότητά του.

3.3.2 ΚΑΤΑΧΩΡΙΣΗ ΑΝΑΦΟΡΑΣ

Για την καταχώριση μιας νέας αναφοράς, η εφαρμογή λαμβάνει αρχικά τη γεωγραφική θέση του χρήστη μέσω GPS. Στη συνέχεια, ο χρήστης μπορεί προαιρετικά να εκτελέσει μια μέτρηση θορύβου μέσω του μικροφώνου της συσκευής, η οποία μετατρέπεται αυτόματα σε αντίστοιχο επίπεδο θορύβου, και υποχρεωτικά καταχωρίζει τις υποκειμενικές του αξιολογήσεις για τις πέντε κατηγορίες όχλησης, μέσω απλών επιλογέων διαβαθμισμένης κλίμακας. Κατόπιν επιβεβαίωσης, η αναφορά αποστέλλεται στη βάση δεδομένων.

3.3.3 ΑΠΟΘΗΚΕΥΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Κάθε αναφορά αποθηκεύεται στη συλλογή environmental_reports της Cloud Firestore. Η αποθήκευση αυτή ενεργοποιεί αυτόματα τη συνάρτηση onReportCreated στην πλευρά του server, η οποία εκτελεί τη βασική επεξεργασία που περιγράφηκε στην Ενότητα 3.1: ενημερώνει το αντίστοιχο κελί πλέγματος, τα συνολικά στατιστικά και το ημερήσιο πλήθος αναφορών. Η ενημέρωση αυτών των συγκεντρωτικών δεδομένων γίνεται με τρόπο ασφαλή ακόμη και όταν πολλοί χρήστες υποβάλλουν αναφορές ταυτόχρονα, μέσω του μηχανισμού συναλλαγών (transactions) της Firestore.

3.3.4 ΧΑΡΤΗΣ ΚΑΙ ΣΤΑΤΙΣΤΙΚΑ

Η εφαρμογή παρέχει δύο συμπληρωματικούς τρόπους οπτικοποίησης των δεδομένων. Ο πρώτος είναι ένας διαδραστικός χάρτης, στον οποίο ο χρήστης μπορεί να εναλλάσσει την προβολή μεμονωμένων αναφορών ως σημείων (markers) με την προβολή του συνολικού θερμικού χάρτη (heatmap). Ο δεύτερος είναι μια οθόνη στατιστικών, η οποία παρουσιάζει τις κατανομές αξιολογήσεων ανά κατηγορία, τον μέσο όρο θορύβου και ένα γράφημα με την τάση του πλήθους αναφορών των τελευταίων επτά ημερών. Και οι δύο οθόνες διαβάζουν απευθείας τα ήδη επεξεργασμένα δεδομένα από τον server, όπως περιγράφηκε στην Ενότητα 3.1.

4ο ΚΕΦΑΛΑΙΟ: ΠΕΡΙΓΡΑΦΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

4.1 ΑΠΑΙΤΟΥΜΕΝΟΙ ΠΟΡΟΙ ΚΑΙ ΑΔΕΙΕΣ ΣΥΣΚΕΥΗΣ

Η εκτέλεση της εφαρμογής σε πραγματική συσκευή Android προϋποθέτει τους εξής πόρους και άδειες: πρόσβαση στο σύστημα εντοπισμού θέσης (GPS) της συσκευής, για τη λήψη των γεωγραφικών συντεταγμένων κάθε αναφοράς· πρόσβαση στο μικρόφωνο, για την προαιρετική μέτρηση της στάθμης θορύβου· και σύνδεση στο διαδίκτυο,

απαραίτητη για την επικοινωνία με την υποδομή Firebase. Οι απαιτήσεις μνήμης και επεξεργαστικής ισχύος είναι σχετικά χαμηλές, καθώς η εφαρμογή δεν εκτελεί βαριές τοπικές επεξεργασίες — η ουσιαστική επεξεργασία δεδομένων μεταφέρεται στον server, όπως περιγράφηκε στο Κεφάλαιο 3. Η εφαρμογή στοχεύει συσκευές με Android 6.0 (API level 23) και νεότερο.

4.2 ΑΝΤΙΜΕΤΩΠΙΣΗ ΣΦΑΛΜΑΤΩΝ ΚΑΙ ΑΝΑΚΤΗΣΗ

Δεδομένου ότι η εφαρμογή βασίζεται σε εξωτερικούς πόρους της συσκευής (GPS) και του δικτύου, οι οποίοι δεν είναι πάντοτε άμεσα διαθέσιμοι, κρίθηκε απαραίτητο η εφαρμογή να αντιμετωπίζει με ανθεκτικό τρόπο τυχόν σφάλματα ή καθυστερήσεις. Χαρακτηριστικό παράδειγμα αποτελεί ο εντοπισμός θέσης: αν η συσκευή δεν καταφέρει να λάβει ακριβή θέση εντός συγκεκριμένου χρονικού ορίου, η εφαρμογή επιχειρεί εναλλακτικά να χρησιμοποιήσει την τελευταία γνωστή θέση της συσκευής, αντί να εμφανίσει σφάλμα ή να παραμείνει σε κατάσταση αναμονής επ' αόριστον. Αντίστοιχα, τυχόν σφάλματα κατά την υποβολή μιας αναφοράς (π.χ. λόγω απώλειας σύνδεσης) αναφέρονται στον χρήστη με κατανοητό μήνυμα, χωρίς να προκαλούν κατάρρευση της εφαρμογής.

4.3 ΔΟΜΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Η βάση δεδομένων Cloud Firestore οργανώνεται σε τέσσερις συλλογές (Πίνακας 1), τρεις από τις οποίες ενημερώνονται αυτόματα από τη συνάρτηση onReportCreated στην πλευρά του server.

Πίνακας 1: Συλλογές της βάσης δεδομένων Firestore

Συλλογή	Περιεχόμενο
environmental_reports	Οι μεμονωμένες αναφορές των χρηστών (θέση, χρονοσήμανση, μετρήσεις, αξιολογήσεις).
heatmap_cells	Τα κελιά του γεωγραφικού πλέγματος, με τη μέση βαθμολογία όχλησης και το πλήθος αναφορών ανά κελί.
stats/summary	Συγκεντρωτικά στατιστικά όλων των αναφορών (κατανομές ανά κατηγορία,

	μέσος όρος θορύβου).
daily_stats	Πλήθος αναφορών ανά ημέρα, για το γράφημα τάσης των τελευταίων 7 ημερών.

4.4 ΑΞΙΟΠΙΣΤΙΑ ΕΦΑΡΜΟΓΗΣ

Η αξιοπιστία της εφαρμογής διασφαλίζεται σε δύο επίπεδα. Στο επίπεδο του κινητού, όλες οι κρίσιμες λειτουργίες (λήψη θέσης, υποβολή αναφοράς) περικλείονται σε δομές χειρισμού εξαιρέσεων (try/catch), ώστε ένα μεμονωμένο σφάλμα να μην οδηγεί σε κατάρρευση ολόκληρης της εφαρμογής. Στο επίπεδο του server, η χρήση συναλλαγών (transactions) κατά την ενημέρωση των συγκεντρωτικών δεδομένων εξασφαλίζει ότι δεν θα προκύψουν ασυνέπειες ακόμη και σε περίπτωση ταυτόχρονων αναφορών από πολλούς χρήστες. Επιπλέον, η σχεδίαση της αρχιτεκτονικής serverless μεταθέτει την ευθύνη της διαθεσιμότητας και της αυτόματης ανάκαμψης από σφάλματα υποδομής στον πάροχο cloud (Google Firebase), αντί αυτή να βαρύνει τον προγραμματιστή.

4.5 ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν δύο γλώσσες προγραμματισμού, ανάλογα με το υποσύστημα. Η κινητή εφαρμογή αναπτύχθηκε στη γλώσσα Dart, μέσω του πλαισίου (framework) Flutter της Google, το οποίο επιτρέπει την ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες (Android, iOS, web) από μία ενιαία βάση κώδικα, παρότι στην παρούσα εργασία η εφαρμογή στοχεύει αποκλειστικά την πλατφόρμα Android. Οι συναρτήσεις της πλευράς του server (Cloud Functions) αναπτύχθηκαν στη γλώσσα JavaScript, μέσω του περιβάλλοντος εκτέλεσης Node.js, το οποίο υποστηρίζεται εγγενώς από την πλατφόρμα Firebase.

4.6 ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ

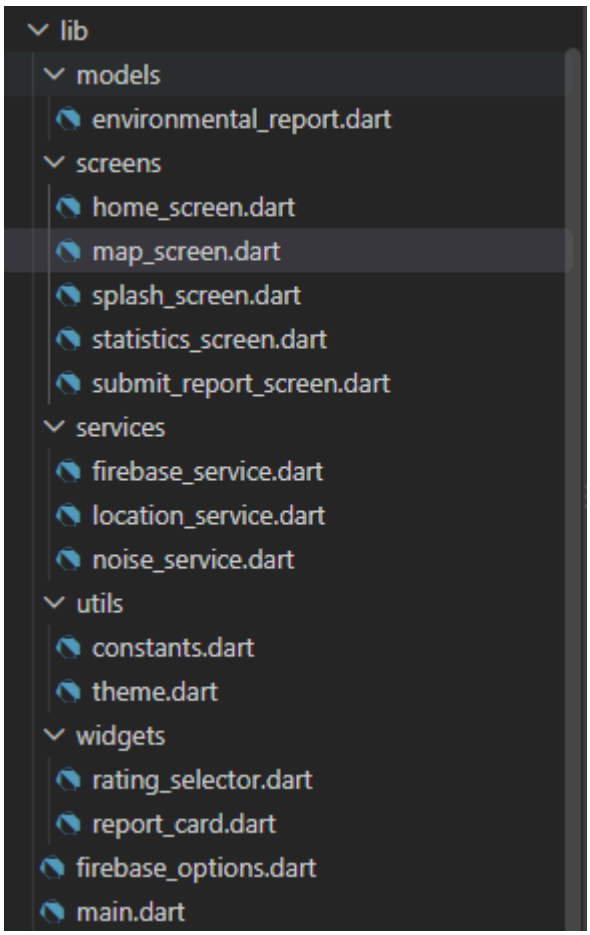
Για την ανάπτυξη ολόκληρου του συστήματος χρησιμοποιήθηκε το περιβάλλον Visual Studio Code, σε συνδυασμό με το Flutter SDK και το Android SDK για τη μεταγλώττιση και τη δοκιμή της εφαρμογής σε πραγματική συσκευή.

4.6.1 ΔΟΜΗ ΦΑΚΕΛΩΝ ΤΗΣ FLUTTER ΕΦΑΡΜΟΓΗΣ

Ο πηγαίος κώδικας της κινητής εφαρμογής οργανώνεται στον φάκελο lib/ σε επιμέρους υποφακέλους ανάλογα με τον ρόλο κάθε αρχείου: ο φάκελος models/ περιέχει τις

κλάσεις που αναπαριστούν τα δεδομένα του συστήματος (π.χ. την κλάση EnvironmentalReport)· ο φάκελος services/ περιέχει τις κλάσεις που επικοινωνούν με εξωτερικές υπηρεσίες, όπως το Firebase (FirebaseService), το GPS (LocationService) και το μικρόφωνο (NoiseService)· ο φάκελος screens/ περιέχει τις οθόνες της εφαρμογής (υποβολή αναφοράς, χάρτης, στατιστικά)· και ο φάκελος widgets/ περιέχει επαναχρησιμοποιήσιμα οπτικά στοιχεία της διεπαφής χρήστη.

Η οργάνωση του πηγαίου κώδικα της κινητής εφαρμογής στο περιβάλλον Visual Studio Code παρουσιάζεται στην Εικόνα 2.

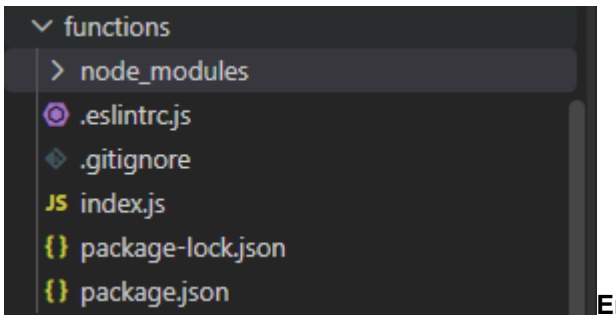


Εικόνα 2: Δομή φακέλων της Flutter εφαρμογής στο Visual Studio Code.

4.6.2 ΔΟΜΗ ΦΑΚΕΛΩΝ ΤΩΝ CLOUD FUNCTIONS

Ο κώδικας της πλευράς του server βρίσκεται σε ξεχωριστό φάκελο (functions/), ανεξάρτητο από την εφαρμογή Flutter, όπως απαιτεί η αρχιτεκτονική του Firebase. Περιλαμβάνει το κύριο αρχείο (index.js), όπου ορίζεται η συνάρτηση onReportCreated, καθώς και το αρχείο εξαρτήσεων (package.json) με τις βιβλιοθήκες firebase-admin και firebase-functions.

Αντίστοιχα, η δομή του φακέλου της πλευράς του server παρουσιάζεται στην Εικόνα 3.



Εικόνα 3: Δομή φακέλου των Cloud Functions.

4.7 Η ΠΛΑΤΦΟΡΜΑ FIREBASE

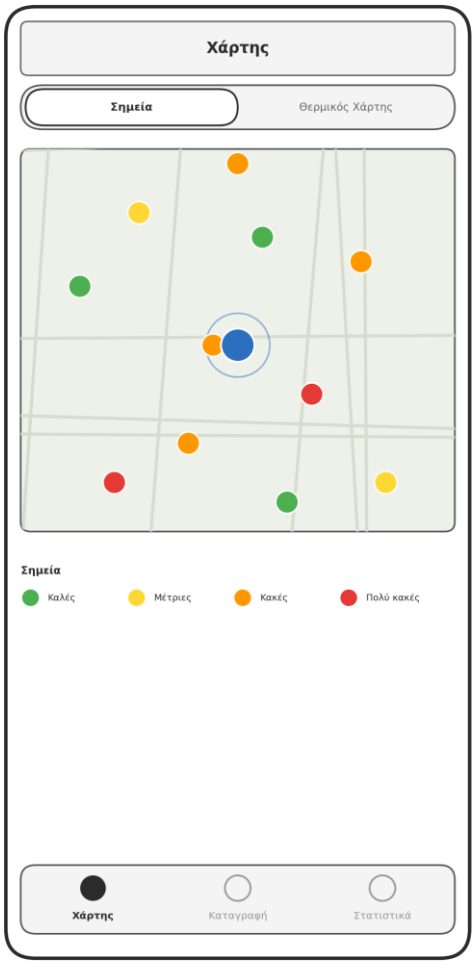
Η Firebase είναι μια πλατφόρμα ανάπτυξης εφαρμογών της Google, η οποία προσφέρει έτοιμες υπηρεσίες backend χωρίς να απαιτείται η διαχείριση δικού της server (serverless / Backend-as-a-Service), όπως παρουσιάστηκε στην Ενότητα 1.3. Στην παρούσα εργασία αξιοποιήθηκαν οι εξής υπηρεσίες της: η Cloud Firestore, μια NoSQL βάση δεδομένων σε πραγματικό χρόνο, στην οποία αποθηκεύονται όλες οι συλλογές του συστήματος· η Firebase Authentication, για την ανώνυμη ταυτοποίηση των χρηστών· και οι Cloud Functions, για την εκτέλεση της συνάρτησης onReportCreated στην πλευρά του server. Η χρήση της πλατφόρμας απαιτεί την αναβάθμιση του έργου στο πρόγραμμα χρέωσης Blaze (πληρωμή ανάλογα με τη χρήση), το οποίο ωστόσο διαθέτει γενναιόδωρο δωρεάν όριο χρήσης, με αποτέλεσμα το κόστος λειτουργίας μιας εφαρμογής σε επίπεδο πτυχιακής εργασίας να παραμένει αμελητέο.

5ο ΚΕΦΑΛΑΙΟ: ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ (FLUTTER)

5.1 ΓΕΝΙΚΗ ΔΟΜΗ ΚΑΙ ΠΛΟΗΓΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

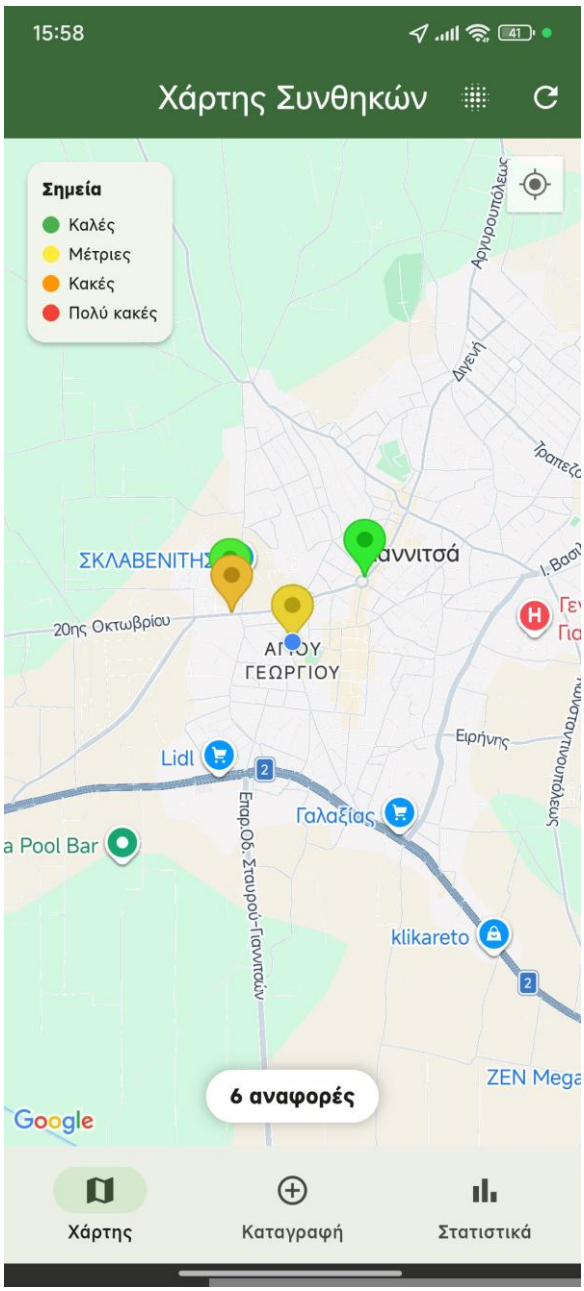
Πριν από την υλοποίηση, σχεδιάστηκαν πρόχειρα mockups των βασικών οθονών, ώστε να αποτυπωθεί η επιθυμητή διάταξη των στοιχείων της διεπαφής και η ροή αλληλεπίδρασης με τον χρήστη. Οι Εικόνες 4 και 5 παρουσιάζουν ενδεικτικά τα mockups της οθόνης υποβολής αναφοράς και της οθόνης χάρτη αντίστοιχα.

Εικόνα 4: Mockup της οθόνης υποβολής αναφοράς.



Εικόνα 5: Mockup της οθόνης χάρτη.

Η ρίζα της εφαρμογής αναλαμβάνει αρχικά την ανώνυμη ταυτοποίηση του χρήστη μέσω της Firebase Authentication, όπως περιγράφηκε στην Ενότητα 3.3.1· όσο η διαδικασία αυτή βρίσκεται σε εξέλιξη, εμφανίζεται μια απλή οθόνη αναμονής (loading indicator). Μόλις ο χρήστης ταυτοποιηθεί, εμφανίζεται η κύρια οθόνη της εφαρμογής, η οποία οργανώνεται γύρω από μια κάτω μπάρα πλοήγησης (bottom navigation bar) με τρεις καρτέλες, αντιστοιχισμένες στις τρεις βασικές λειτουργικές απαιτήσεις του Κεφαλαίου 3: «Χάρτης», «Καταγραφή» και «Στατιστικά». Ως προεπιλεγμένη καρτέλα κατά την εκκίνηση εμφανίζεται ο «Χάρτης». Η εναλλαγή μεταξύ των καρτελών γίνεται τοπικά, χωρίς επανα-φόρτωση δεδομένων που έχουν ήδη ανακτηθεί, ώστε η μετάβαση να είναι άμεση.



Εικόνα 6: Αρχική οθόνη της εφαρμογής με τη μπάρα πλοήγησης.

5.2 ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ (ΚΛΑΣΗ ENVIRONMENTALREPORT)

Η βασική οντότητα του συστήματος αναπαρίσταται στην κινητή εφαρμογή από την κλάση EnvironmentalReport, τα πεδία της οποίας συνοψίζονται στον Πίνακα 2. Η κλάση αυτή διαθέτει μεθόδους μετατροπής προς και από τη μορφή εγγράφου της Firestore (toFirestore/fromFirestore), ώστε η σειριοποίηση και αποσειριοποίηση των δεδομένων να γίνεται σε ένα ενιαίο, επαναχρησιμοποιήσιμο σημείο του κώδικα, αντί να επαναλαμβάνεται σε κάθε οθόνη που διαβάζει ή γράφει αναφορές.

Αξίζει να σημειωθεί ότι το πεδίο deviceTemperature είναι προαιρετικό (nullable) και ορίζεται στο μοντέλο δεδομένων κάτω από το ίδιο σχόλιο κώδικα («Measured sensor

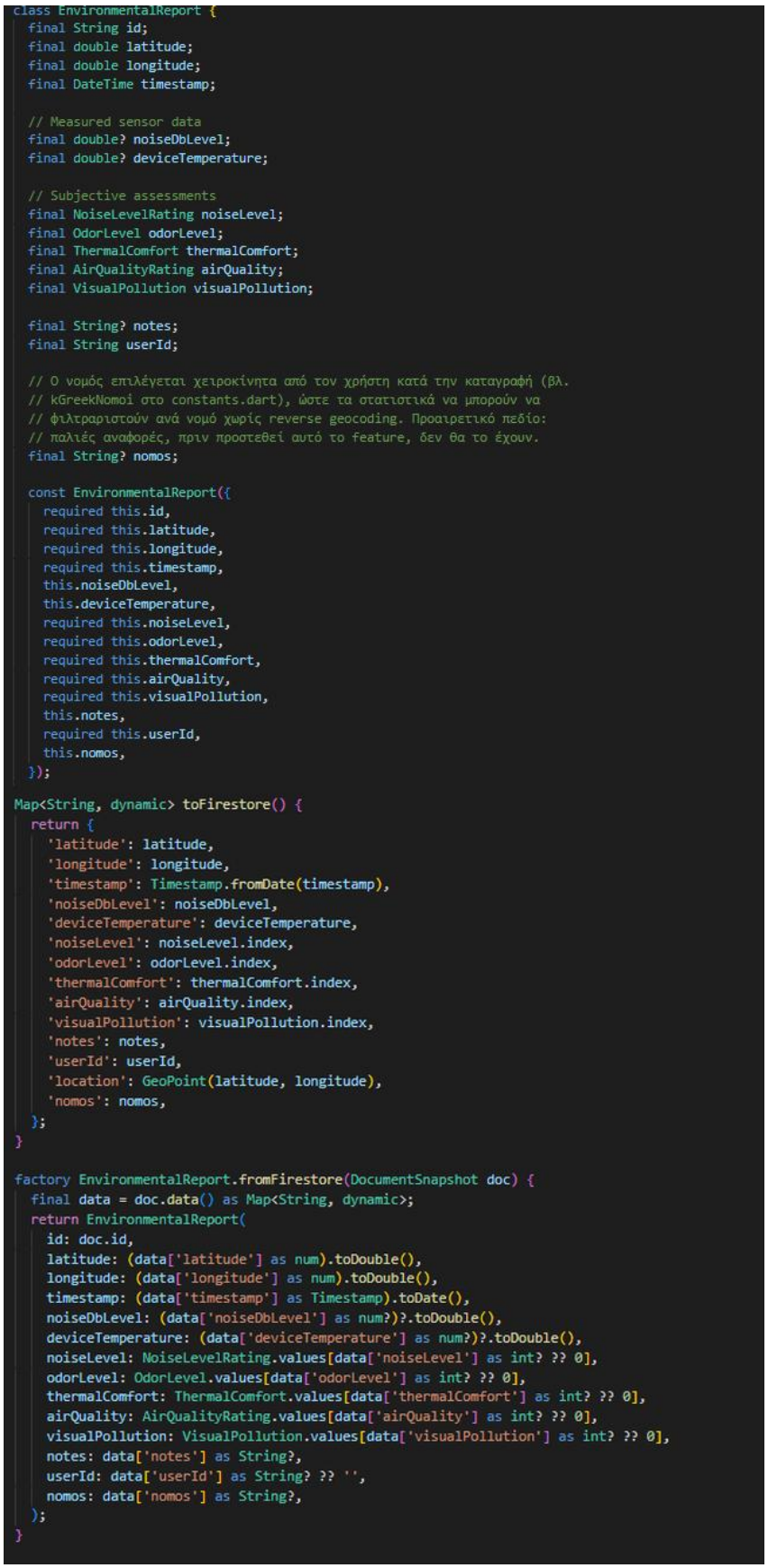
data») με το noiseDbLevel, αντιμετωπιζόμενο συμμετρικά μαζί του τόσο στον constructor όσο και στη σειριοποίηση προς/από το Firestore. Στην τρέχουσα έκδοση της εφαρμογής, ωστόσο, το πεδίο αυτό δεν συμπληρώνεται από καμία λειτουργία ανάγνωσης αισθητήρα θερμοκρασίας κατά την υποβολή αναφοράς και παραμένει πάντα null· έχει προβλεφθεί στο σχήμα δεδομένων ως σημείο μελλοντικής επέκτασης, σε αντίθεση με τις πέντε υποκειμενικές αξιολογήσεις του χρήστη, οι οποίες συμπληρώνονται πάντα κατά την υποβολή.

Πίνακας 2: Πεδία της κλάσης EnvironmentalReport.

Πεδίο	Περιγραφή
id	String — μοναδικό αναγνωριστικό του εγγράφου στη Firestore.
latitude	double — γεωγραφικό πλάτος της θέσης της αναφοράς.
longitude	double — γεωγραφικό μήκος της θέσης της αναφοράς. Κατά την αποθήκευση (toFirestore()), το latitude/longitude συνδυάζονται επιπλέον σε ένα πεδίο GeoPoint με όνομα location, για ενδεχόμενα γεωχωρικά ερωτήματα.
timestamp	DateTime — χρονοσήμανση καταχώρισης της αναφοράς.
noiseDbLevel	double? — μετρηθείσα στάθμη θορύβου σε ντεσιμπέλ μέσω του μικροφώνου (nullable, εφόσον εκτελέστηκε μέτρηση).
deviceTemperature	double? — θερμοκρασία περιβάλλοντος (nullable)· προβλέπεται στο μοντέλο δεδομένων, δεν συμπληρώνεται όμως από κάποιον αισθητήρα στην τρέχουσα υλοποίηση.

noiseLevel	NoiseLevelRating (enum) — υποκειμενική αξιολόγηση του χρήστη για το επίπεδο θορύβου.
odorLevel	OdorLevel (enum) — υποκειμενική αξιολόγηση του χρήστη για τη δυσοσμία.
thermalComfort	ThermalComfort (enum) — υποκειμενική αξιολόγηση του χρήστη για τη θερμική δυσφορία.
airQuality	AirQualityRating (enum) — υποκειμενική αξιολόγηση του χρήστη για την ποιότητα του αέρα.
visualPollution	VisualPollution (enum) — υποκειμενική αξιολόγηση του χρήστη για την οπτική ρύπανση.
notes	String? — προαιρετικές ελεύθερες σημειώσεις του χρήστη.
userId	String — ανώνυμο αναγνωριστικό χρήστη (Firebase Authentication).
nomos	String? — ο νομός που επιλέγει ο χρήστης από στατική λίστα κατά την καταγραφή (nullable)· χρησιμοποιείται για το φιλτράρισμα στατιστικών ανά νομό (βλ. Ενότητα 5.7). Προστέθηκε μετά την έναρξη συλλογής δεδομένων, οπότε παλαιότερες αναφορές δεν το διαθέτουν.

Η Εικόνα 7 παρουσιάζει την κλάση EnvironmentalReport, όπως ορίζεται στο αρχείο lib/models/environmental_report.dart, με τα πεδία του Πίνακα 2 και τις μεθόδους μετατροπής toFirestore()/fromFirestore() που χρησιμοποιούνται για την αποθήκευση και την ανάκτηση των αντικειμένων από τη Cloud Firestore.



```dart
class EnvironmentalReport {
  final String id;
  final double latitude;
  final double longitude;
  final DateTime timestamp;

  // Measured sensor data
  final double? noiseDbLevel;
  final double? deviceTemperature;

  // Subjective assessments
  final NoiseLevelRating noiseLevel;
  final OdorLevel odorLevel;
  final ThermalComfort thermalComfort;
  final AirQualityRating airQuality;
  final VisualPollution visualPollution;

  final String? notes;
  final String userId;

  // Ο νομός επιλέγεται χειροκίνητα από τον χρήστη κατά την καταγραφή (βλ.
  // kGreekNomoi στο constants.dart), ώστε τα στατιστικά να μπορούν να
  // φιλτραριστούν ανά νομό χωρίς reverse geocoding. Προαιρετικό πεδίο:
  // παλιές αναφορές, πριν προστεθεί αυτό το feature, δεν θα το έχουν.
  final String? nomos;

  const EnvironmentalReport({
    required this.id,
    required this.latitude,
    required this.longitude,
    required this.timestamp,
    this.noiseDbLevel,
    this.deviceTemperature,
    required this.noiseLevel,
    required this.odorLevel,
    required this.thermalComfort,
    required this.airQuality,
    required this.visualPollution,
    this.notes,
    required this.userId,
    this.nomos,
  });

Map<String, dynamic> toFirestore() {
  return {
    'latitude': latitude,
    'longitude': longitude,
    'timestamp': Timestamp.fromDate(timestamp),
    'noiseDbLevel': noiseDbLevel,
    'deviceTemperature': deviceTemperature,
    'noiseLevel': noiseLevel.index,
    'odorLevel': odorLevel.index,
    'thermalComfort': thermalComfort.index,
    'airQuality': airQuality.index,
    'visualPollution': visualPollution.index,
    'notes': notes,
    'userId': userId,
    'location': GeoPoint(latitude, longitude),
    'nomos': nomos,
  };
}

factory EnvironmentalReport.fromFirestore(DocumentSnapshot doc) {
  final data = doc.data() as Map<String, dynamic>;
  return EnvironmentalReport(
    id: doc.id,
    latitude: (data['latitude'] as num).toDouble(),
    longitude: (data['longitude'] as num).toDouble(),
    timestamp: (data['timestamp'] as Timestamp).toDate(),
    noiseDbLevel: (data['noiseDbLevel'] as num?)?.toDouble(),
    deviceTemperature: (data['deviceTemperature'] as num?)?.toDouble(),
    noiseLevel: NoiseLevelRating.values[data['noiseLevel'] as int? ?? 0],
    odorLevel: OdorLevel.values[data['odorLevel'] as int? ?? 0],
    thermalComfort: ThermalComfort.values[data['thermalComfort'] as int? ?? 0],
    airQuality: AirQualityRating.values[data['airQuality'] as int? ?? 0],
    visualPollution: VisualPollution.values[data['visualPollution'] as int? ?? 0],
    notes: data['notes'] as String?,
    userId: data['userId'] as String? ?? '',
    nomos: data['nomos'] as String?,
  );
}
```

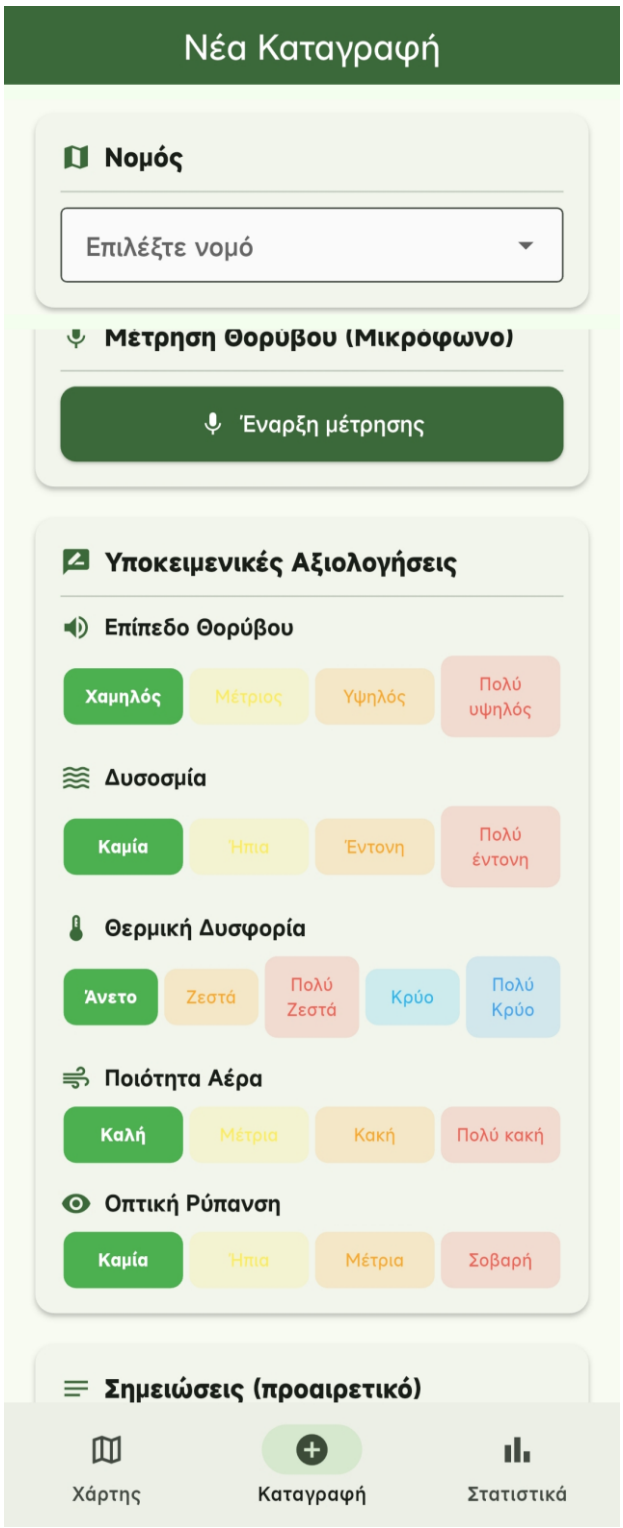
Εικόνα 7: Η κλάση EnvironmentalReport (lib/models/environmental_report.dart).

Όπως φαίνεται στην Εικόνα 7, η μέθοδος toFirestore() μετατρέπει το αντικείμενο σε Map<String, dynamic> προς εγγραφή στη Firestore, ενώ η μέθοδος (factory

constructor) fromFirestore() εκτελεί την αντίστροφη διαδικασία κατά την ανάγνωση ενός εγγράφου, ώστε η μετατροπή να μη χρειάζεται να επαναλαμβάνεται σε κάθε οθόνη.

5.3 ΟΘΟΝΗ ΥΠΟΒΟΛΗΣ ΑΝΑΦΟΡΑΣ (SUBMITREPORTSCREEN)

Η οθόνη SubmitReportScreen υλοποιεί τη λειτουργία καταχώρισης αναφοράς που περιγράφηκε στην Ενότητα 3.3.2. Κατά την εμφάνισή της, ζητείται αυτόματα η τρέχουσα θέση του χρήστη μέσω του LocationService (Ενότητα 5.6.2), με σχετική ένδειξη φόρτωσης όσο αναμένεται η απάντηση του GPS. Ο χρήστης μπορεί προαιρετικά να πατήσει ένα κουμπί μέτρησης θορύβου, το οποίο ενεργοποιεί το NoiseService (Ενότητα 5.6.3) για μερικά δευτερόλεπτα και εμφανίζει την εκτιμημένη στάθμη σε ντεσιμπέλ. Στη συνέχεια, ο χρήστης καταχωρίζει τις υποκειμενικές του αξιολογήσεις για τις πέντε κατηγορίες όχλησης, επιλέγοντας σε κάθε κατηγορία ένα από τα διαθέσιμα διακριτά επίπεδα μέσω αντίστοιχων κουμπιών επιλογής (τέσσερα επίπεδα για το επίπεδο θορύβου, τη δυσοσμία, την ποιότητα αέρα και την οπτική ρύπανση, και πέντε επίπεδα για τη θερμική δυσφορία), και ένα προαιρετικό πεδίο ελεύθερου κειμένου επιτρέπει επιπλέον σημειώσεις. Το κουμπί υποβολής παραμένει ανενεργό μέχρι να έχει προσδιοριστεί η θέση, ώστε να αποφεύγεται η αποστολή ελλιπών αναφορών· μετά την επιτυχή υποβολή, ο χρήστης ενημερώνεται με σχετικό μήνυμα επιβεβαίωσης (SnackBar) και η φόρμα επανέρχεται στην αρχική της κατάσταση.



Εικόνα 8: Οθόνη υποβολής αναφοράς (SubmitReportScreen).

Η Εικόνα 9 παρουσιάζει τη μέθοδο _submit() της κλάσης SubmitReportScreen, η οποία εκτελείται κατά το πάτημα του κουμπιού υποβολής.

```dart
Future<void> _submit() async {
  if (_position == null) {
    _showSnack('Απαιτείται τοποθεσία GPS');
    return;
  }
  if (_selectedNomos == null) {
    _showSnack('Επιλέξτε νομό');
    return;
  }
  setState(() => _isSubmitting = true);
  try {
    final report = EnvironmentalReport(
      id: const Uuid().v4(),
      latitude: _position!.latitude,
      longitude: _position!.longitude,
      timestamp: DateTime.now(),
      noiseDbLevel: _currentDb > 0 ? _currentDb : null,
      noiseLevel: NoiseLevelRating.values[_noiseLevelIdx],
      odorLevel: OdorLevel.values[_odorIdx],
      thermalComfort: ThermalComfort.values[_thermalIdx],
      airQuality: AirQualityRating.values[_airQualityIdx],
      visualPollution: VisualPollution.values[_visualIdx],
      notes: _notesController.text.trim().isEmpty ? null : _notesController.text.trim(),
      userId: FirebaseService.instance.currentUserId ?? 'anonymous',
      nomos: _selectedNomos,
    ); // EnvironmentalReport
    await FirebaseService.instance.submitReport(report);
    if (mounted) {
      _showSnack('Η αναφορά υποβλήθηκε επιτυχώς!', isError: false);
      _reset();
    }
  } catch (e) {
    _showSnack('Σφάλμα υποβολής: $e');
  } finally {
    if (mounted) setState(() => _isSubmitting = false);
  }
}
```

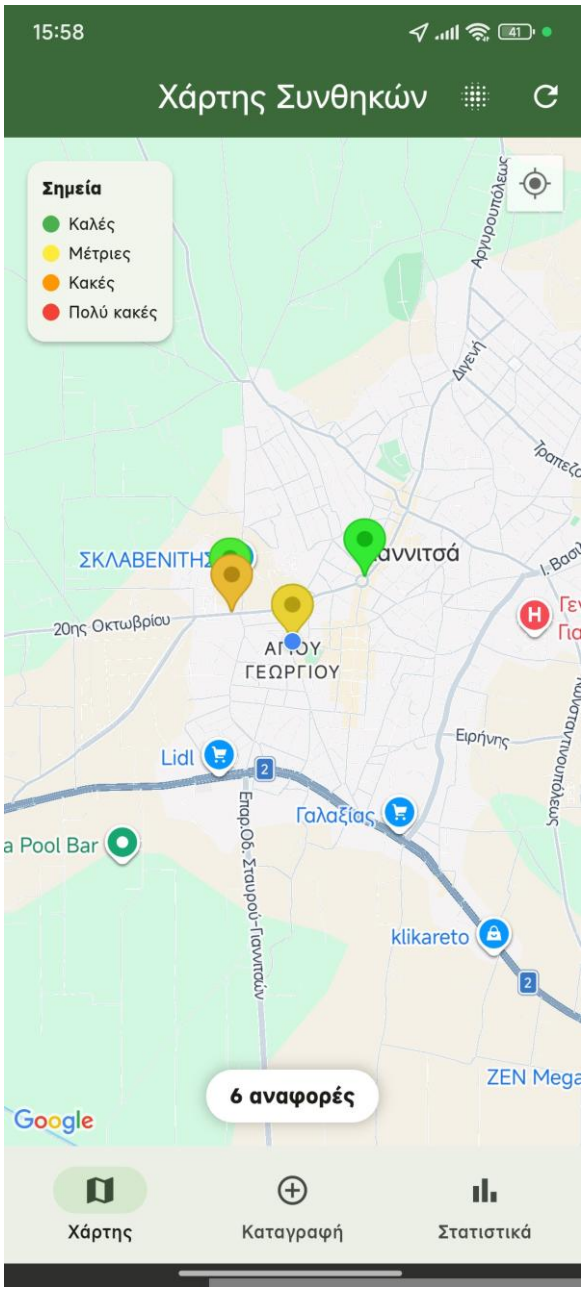
Εικόνα 9: Η μέθοδος _submit() στην κλάση SubmitReportScreen.

Η μέθοδος αυτή συγκεντρώνει τη θέση που ανακτήθηκε από το LocationService, την προαιρετική μέτρηση θορύβου και τις πέντε αξιολογήσεις, δημιουργεί ένα αντικείμενο EnvironmentalReport και καλεί την submitReport() του FirebaseService (Ενότητα 5.6.1), πριν εμφανίσει στον χρήστη το μήνυμα επιβεβαίωσης της Ενότητας 5.3.

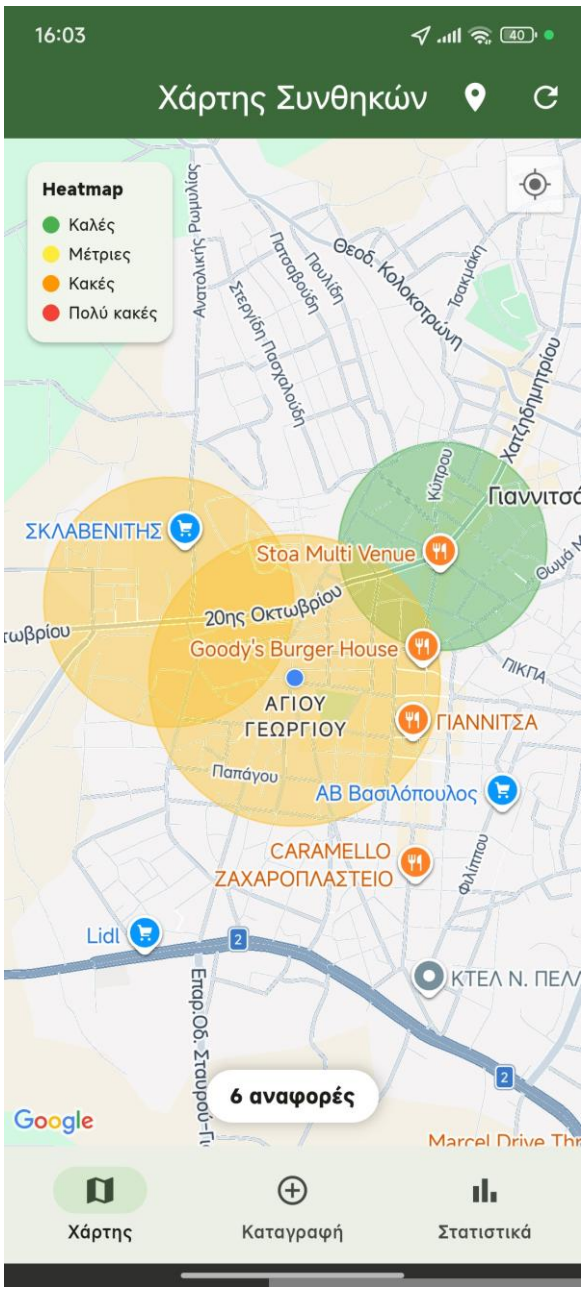
5.4 ΟΘΟΝΗ ΧΑΡΤΗ (MAPSCREEN)

Η οθόνη MapScreen υλοποιεί τη διττή οπτικοποίηση που περιγράφηκε στην Ενότητα 3.3.4, βασισμένη στο πρόσθετο (package) google_maps_flutter. Ένα κουμπί εναλλαγής στο πάνω μέρος της οθόνης επιτρέπει στον χρήστη να μεταβαίνει μεταξύ δύο προβολών. Στην πρώτη προβολή, η εφαρμογή διαβάζει σε πραγματικό χρόνο τη συλλογή environmental_reports και τοποθετεί ένα σημείο (marker) για κάθε μεμονωμένη αναφορά, το οποίο, όταν πατηθεί, εμφανίζει συνοπτικές πληροφορίες της αναφοράς. Στη δεύτερη προβολή, η εφαρμογή διαβάζει τη συλλογή heatmap_cells και αποδίδει κάθε κελί του γεωγραφικού πλέγματος ως έναν χρωματισμένο κύκλο (Circle), με το χρώμα να αντιστοιχίζεται στη μέση βαθμολογία όχλησης του κελιού και την ακτίνα να αυξάνεται με το πλήθος των αναφορών του κελιού. Σε αμφότερες τις περιπτώσεις, η

οθόνη δεν εκτελεί καμία δική της επεξεργασία δεδομένων, παρά μόνο διαβάζει και αποδίδει τα ήδη επεξεργασμένα αποτελέσματα του server, σύμφωνα με την αρχή που τέθηκε στην Ενότητα 3.1.

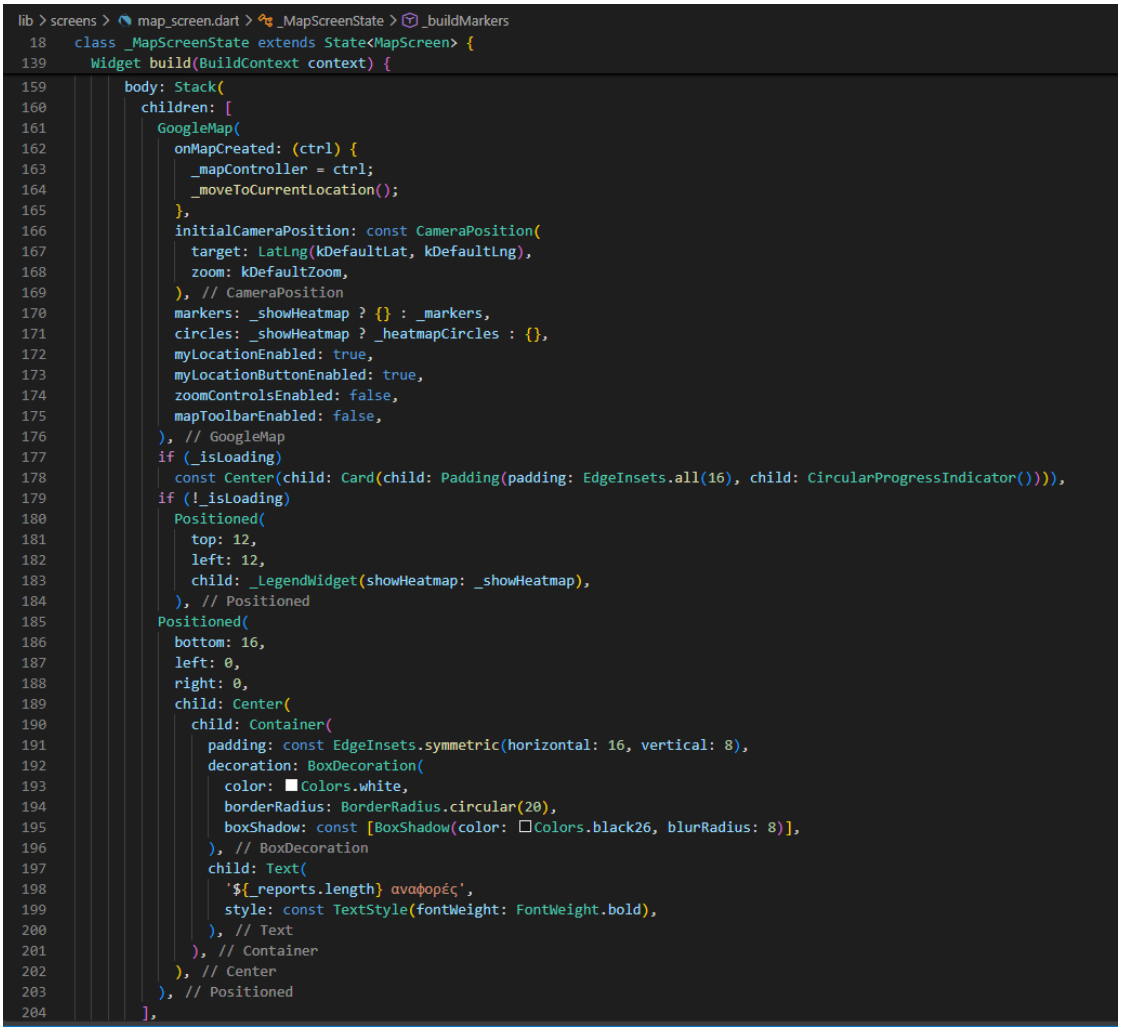


Εικόνα 10: Οθόνη χάρτη σε προβολή μεμονωμένων αναφορών (markers).



Εικόνα 11: Οθόνη χάρτη σε προβολή θερμικού χάρτη (heatmap).

Η Εικόνα 12 παρουσιάζει το σημείο της κλάσης MapScreen όπου γίνεται η εναλλαγή μεταξύ των δύο προβολών του χάρτη.



Εικόνα 12: Εναλλαγή προβολής σημείων/θερμικού χάρτη στην κλάση MapScreen.

Η λογική αυτή διαβάζει, ανάλογα με την επιλεγμένη προβολή, είτε τη ζωντανή ροή (stream) της συλλογής environmental_reports είτε της συλλογής heatmap_cells μέσω του FirebaseService, και αποδίδει αντίστοιχα Marker ή Circle αντικείμενα πάνω στον χάρτη, όπως περιγράφηκε στην Ενότητα 5.4.

5.5 ΟΘΟΝΗ ΣΤΑΤΙΣΤΙΚΩΝ (STATISTICSSCREEN)

Η οθόνη StatisticsScreen ακολουθεί το ίδιο μοντέλο ανάγνωσης έτοιμων, συγκεντρωτικών δεδομένων. Συνδέεται με το έγγραφο stats/summary για να παρουσιάσει την κατανομή των αξιολογήσεων ανά κατηγορία όχλησης (σε μορφή ραβδογράμματος) και τον μέσο όρο της μετρηθείσας στάθμης θορύβου, και με τη συλλογή daily_stats για να εμφανίσει ένα γράφημα με την τάση του πλήθους αναφορών των τελευταίων επτά ημερών. Επειδή η εφαρμογή διαβάζει τα δεδομένα αυτά μέσω ζωντανής ροής (stream), κάθε νέα αναφορά που επεξεργάζεται ο server αντικατοπτρίζεται αυτόματα στα γραφήματα, χωρίς να απαιτείται χειροκίνητη ανανέωση από τον χρήστη.

Εικόνα 13: Οθόνη στατιστικών (StatisticsScreen).

5.6 ΥΠΗΡΕΣΙΕΣ ΕΦΑΡΜΟΓΗΣ

Η επικοινωνία της εφαρμογής με εξωτερικούς πόρους (πλατφόρμα Firebase, GPS, μικρόφωνο) δεν γίνεται απευθείας από τις οθόνες, αλλά έχει απομονωθεί σε τρεις ξεχωριστές κλάσεις υπηρεσιών, στον φάκελο lib/services/ (Ενότητα 4.6.1). Η προσέγγιση αυτή επιτρέπει την επαναχρησιμοποίηση της ίδιας λογικής από πολλαπλές οθόνες και διευκολύνει την κεντρική διαχείριση σφαλμάτων που περιγράφηκε στην Ενότητα 4.2.

5.6.1 FIREBASESERVICE

Η κλάση FirebaseService συγκεντρώνει όλη την επικοινωνία με την Cloud Firestore και τη Firebase Authentication: την ανώνυμη ταυτοποίηση του χρήστη κατά την εκκίνηση, την αποστολή μιας νέας αναφοράς στη συλλογή environmental_reports, καθώς και τις ζωντανές ροές (streams) ανάγνωσης από τις συλλογές environmental_reports, heatmap_cells, stats/summary και daily_stats, τις οποίες καταναλώνουν οι οθόνες MapScreen και StatisticsScreen μέσω widgets τύπου StreamBuilder.

Η Εικόνα 14 παρουσιάζει τη μέθοδο submitReport() της κλάσης FirebaseService, η οποία είναι υπεύθυνη για την αποστολή μιας νέας αναφοράς στη Cloud Firestore.

```dart
Future<void> submitReport(EnvironmentalReport report) async {
  await _db.collection(_collection).add(report.toFirestore());
}

Future<List<EnvironmentalReport>> getReports({int limit = 200}) async {
  final snapshot = await _db
      .collection(_collection)
      .orderBy('timestamp', descending: true)
      .limit(limit)
      .get();
  return snapshot.docs.map(EnvironmentalReport.fromFirestore).toList();
}

Stream<List<EnvironmentalReport>> reportsStream() {
  return _db
      .collection(_collection)
      .orderBy('timestamp', descending: true)
      .limit(200)
      .snapshots()
      .map((snap) => snap.docs.map(EnvironmentalReport.fromFirestore).toList());
}
```

Εικόνα 14: Η μέθοδος submitReport() της κλάσης FirebaseService.

Η μέθοδος αυτή απλώς μετατρέπει το αντικείμενο EnvironmentalReport σε Map μέσω της toFirestore() (Ενότητα 5.2) και το εγγράφει ως νέο έγγραφο στη συλλογή environmental_reports, ενέργεια η οποία ενεργοποιεί αυτόματα τη συνάρτηση onReportCreated στην πλευρά του server (Ενότητα 6.1).

5.6.2 LOCATIONSERVICE

Η κλάση LocationService περιτυλίγει το πρόσθετο geolocator, αναλαμβάνοντας τον έλεγχο και, εφόσον απαιτείται, το αίτημα των απαραίτητων αδειών θέσης (Ενότητα 4.1), καθώς και την ανάκτηση των τρεχόντων συντεταγμένων της συσκευής. Σε περίπτωση που η ανάκτηση της θέσης αποτύχει ή υπερβεί ένα χρονικό όριο, η υπηρεσία επιστρέφει ένα ελεγχόμενο σφάλμα, το οποίο η οθόνη SubmitReportScreen μετατρέπει σε κατανοητό μήνυμα προς τον χρήστη, σύμφωνα με τη στρατηγική ανθεκτικότητας της Ενότητας 4.2.

5.6.3 NOISESERVICE

Η κλάση NoiseService αναλαμβάνει την προαιρετική μέτρηση θορύβου μέσω του μικροφώνου της συσκευής. Κατά την ενεργοποίησή της, συλλέγει για μερικά δευτερόλεπτα δείγματα πλάτους του ηχητικού σήματος και τα μετατρέπει σε μια εκτιμώμενη στάθμη θορύβου σε ντεσιμπέλ, η οποία επιστρέφεται στην οθόνη SubmitReportScreen για να συμπεριληφθεί στην υπό κατάρτιση αναφορά. Η μέτρηση αυτή είναι καθαρά ενδεικτική και δεν αποσκοπεί στην ακρίβεια ενός βαθμονομημένου ηχομέτρου, αλλά στην παροχή μιας αντικειμενικής συμπληρωματικής ένδειξης δίπλα στις υποκειμενικές αξιολογήσεις του χρήστη.

5.7 ΓΕΩΓΡΑΦΙΚΟ ΦΙΛΤΡΑΡΙΣΜΑ ΣΤΑΤΙΣΤΙΚΩΝ ΑΝΑ ΝΟΜΟ

Κατά την επίδειξη της εφαρμογής στον επιβλέποντα καθηγητή, διατυπώθηκε η παρατήρηση ότι τα συγκεντρωτικά στατιστικά της Ενότητας 5.5 (και τα υποκείμενα έγγραφα stats/summary, daily_stats της Ενότητας 6.4) είναι παγκόσμια, χωρίς καμία γεωγραφική διάσταση: μία αναφορά από τα Γιαννιτσά και μία αναφορά από την Αθήνα προσμετρώνται στο ίδιο σύνολο, χωρίς δυνατότητα προβολής στατιστικών για μία μόνο περιοχή. Ζητήθηκε διερεύνηση του κατά πόσο αυτό μπορεί να αλλάξει ώστε τα στατιστικά να μπορούν να φιλτράρονται ανά περιοχή.

Εξετάστηκαν δύο εναλλακτικές προσεγγίσεις για τον ορισμό μιας "περιοχής". Η πρώτη προϋπέθετε αντίστροφη γεωκωδικοποίηση (reverse geocoding) των συντεταγμένων κάθε αναφοράς σε πραγματικό όνομα νομού, μέσω εξωτερικού API — προσέγγιση που προσθέτει εξωτερική εξάρτηση, ενδεχόμενο κόστος και ένα επιπλέον σημείο αποτυχίας στη ροή υποβολής. Η δεύτερη προϋπέθετε επαναχρησιμοποίηση του γεωγραφικού πλέγματος του heatmap (Ενότητα 6.3) με πολύ αδρότερο μέγεθος κελιού, με το μειονέκτημα ότι τα όρια των κελιών θα ήταν αυθαίρετα τετράγωνα και όχι πραγματικά

διοικητικά όρια. Επιλέχθηκε τελικά μια τρίτη, απλούστερη λύση: ο ίδιος ο χρήστης επιλέγει τον νομό του από στατική λίστα κατά την υποβολή της αναφοράς, γεγονός που αποφεύγει κάθε εξωτερική εξάρτηση διατηρώντας παράλληλα ουσιαστικές, αναγνωρίσιμες γεωγραφικές μονάδες.

Η υλοποίηση προσέθεσε ένα προαιρετικό πεδίο nomos στην κλάση EnvironmentalReport (Ενότητα 5.2) και στο αντίστοιχο έγγραφο Firestore, καθώς και ένα υποχρεωτικό dropdown επιλογής νομού στην οθόνη υποβολής αναφοράς (Ενότητα 5.3), με τον ίδιο τρόπο επικύρωσης πριν την υποβολή όπως και η θέση GPS. Στο FirebaseService (Ενότητα 5.6.1) προστέθηκε η μέθοδος getStatisticsForNomos(nomos), η οποία —σε αντίθεση με την getStatistics() που διαβάζει το προϋπολογισμένο, παγκόσμιο έγγραφο stats/summary— εκτελεί ερώτημα ισότητας where('nomos', isEqualTo: nomos) στη συλλογή environmental_reports και υπολογίζει τις κατανομές τοπικά, στο κινητό, πάνω στο ήδη φιλτραρισμένο σύνολο αποτελεσμάτων. Το ερώτημα ισότητας ευρετηριάζεται αυτόματα από τη Firestore, χωρίς να απαιτείται καμία αλλαγή στους κανόνες ασφαλείας, σε ευρετήρια (indexes) ή στις Cloud Functions. Στην οθόνη Στατιστικών (Ενότητα 5.5) προστέθηκε αντίστοιχο dropdown, με προεπιλογή τη συνολική εικόνα όλης της χώρας (ίδια συμπεριφορά με πριν) και δυνατότητα εναλλαγής σε προβολή φιλτραρισμένη ανά νομό.

Επειδή το πεδίο nomos προστέθηκε μετά την έναρξη συλλογής δεδομένων, οι ήδη υπάρχουσες αναφορές δεν το διαθέτουν και συνεχίζουν να προσμετρώνται μόνο στη συνολική, μη φιλτραρισμένη προβολή — δεν εμφανίζονται σε καμία προβολή φιλτραρισμένη ανά νομό. Δεδομένου του μικρού όγκου των έως τώρα καταγεγραμμένων αναφορών, προτιμήθηκε αυτή η απλή, χωρίς κίνδυνο, αντιμετώπιση αντί για αυτοματοποιημένη αναδρομική ταξινόμηση παλαιών εγγράφων, η οποία θα δικαιολογούνταν σε μεγαλύτερη κλίμακα δεδομένων.

6ο ΚΕΦΑΛΑΙΟ: ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ SERVER-SIDE ΕΠΕΞΕΡΓΑΣΙΑΣ (CLOUD FUNCTIONS)

6.1 ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ONREPORTCREATED

Η βασική επεξεργασία της πλευράς του server υλοποιείται σε μία και μόνο συνάρτηση, την onReportCreated, γραμμένη σε JavaScript πάνω στο Node.js (Ενότητα 4.5) και ορισμένη στο αρχείο index.js (Ενότητα 4.6.2). Η συνάρτηση αυτή δηλώνεται ως ενεργοποιητής (trigger) της Cloud Firestore, ο οποίος πυροδοτείται αυτόματα σε κάθε δημιουργία νέου εγγράφου στη συλλογή environmental_reports, χωρίς να απαιτείται

καμία ενέργεια από την κινητή εφαρμογή πέραν της αρχικής υποβολής. Κατά την ενεργοποίησή της, η συνάρτηση λαμβάνει ως παράμετρο τα δεδομένα της νέας αναφοράς και εκτελεί διαδοχικά τρία βήματα επεξεργασίας: τον υπολογισμό της βαθμολογίας όχλησης (Ενότητα 6.2), την ενημέρωση του αντίστοιχου κελιού του γεωγραφικού πλέγματος (Ενότητα 6.3) και την ενημέρωση των συγκεντρωτικών στατιστικών (Ενότητα 6.4). Η προσέγγιση αυτή, βασισμένη σε συμβάντα (event-driven), μεταφέρει το σύνολο της ουσιαστικής επεξεργασίας δεδομένων στον server, ακολουθώντας την αρχή σχεδιασμού που τέθηκε στην Ενότητα 3.1, και επιτρέπει η κινητή εφαρμογή να παραμένει απλή, καταναλώνοντας απλώς τα ήδη επεξεργασμένα αποτελέσματα.

Η Εικόνα 15 παρουσιάζει τον ορισμό της συνάρτησης onReportCreated στο αρχείο functions/index.js.

```
exports.onReportCreated = onDocumentCreated(
    {document: "environmental_reports/{reportId}", region: "europe-west1"},
    async (event) => {
      const snap = event.data;
      if (!snap) {
        logger.warn("Δεν υπάρχουν δεδομένα στο event, παραλείπεται.");
        return;
      }
      const report = snap.data();
      const reportId = event.params.reportId;

      const lat = report.latitude;
      const lng = report.longitude;
      if (typeof lat !== "number" || typeof lng !== "number") {
        logger.warn(
            `Η αναφορά ${reportId} δεν έχει έγκυρες συντεταγμένες.`,
        );
        return;
      }
```

Εικόνα 15: Ορισμός της συνάρτησης onReportCreated (functions/index.js).

Όπως φαίνεται στην Εικόνα 15, η συνάρτηση δηλώνεται ως trigger τύπου onDocumentCreated (2ης γενιάς Cloud Functions, από το πρόσθετο firebase-functions/v2/firestore) πάνω στη διαδρομή environmental_reports/{reportId} και λαμβάνει ως όρισμα ένα στιγμιότυπο (snapshot) του νέου εγγράφου, από το οποίο εξάγονται τα δεδομένα της αναφοράς προς περαιτέρω επεξεργασία.

6.2 ΥΠΟΛΟΓΙΣΜΟΣ ΒΑΘΜΟΛΟΓΙΑΣ ΟΧΛΗΣΗΣ (POLLUTION SCORE)

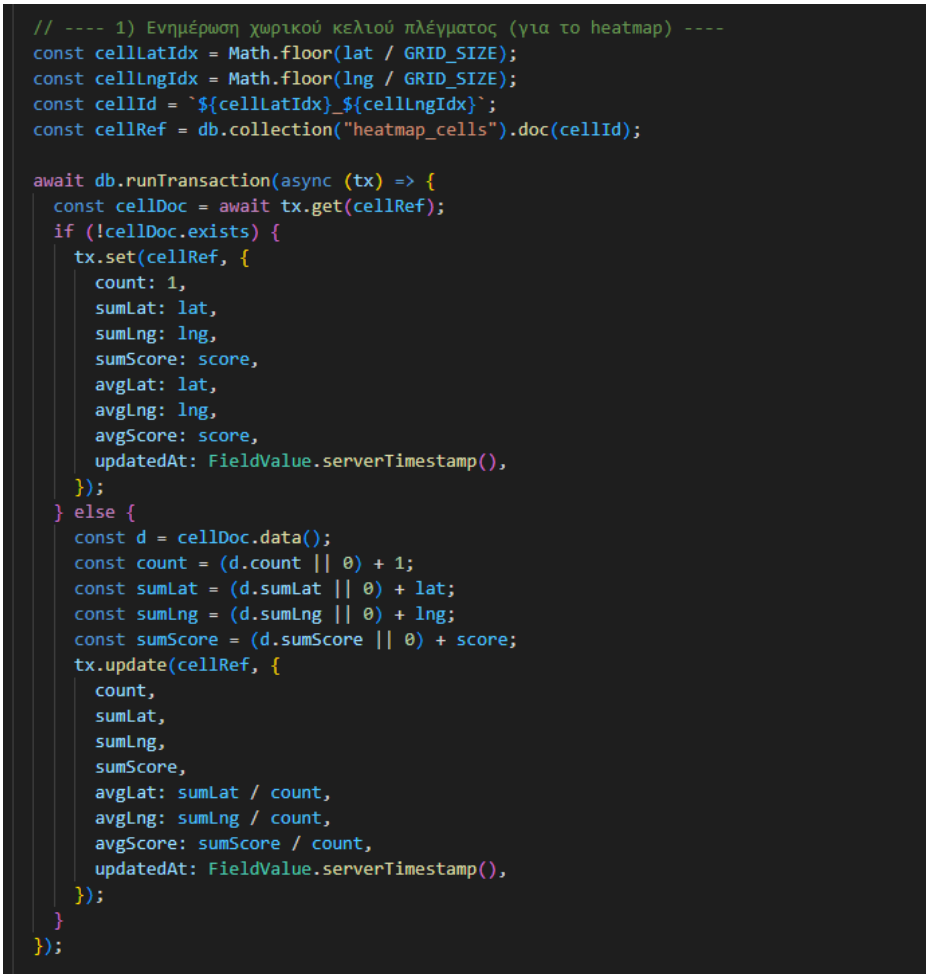
Το πρώτο βήμα της onReportCreated είναι ο υπολογισμός της βαθμολογίας όχλησης της νέας αναφοράς, όπως ορίστηκε στην Ενότητα 3.2.2: οι πέντε επιμέρους

αξιολογήσεις κανονικοποιούνται στο εύρος 0-100 και υπολογίζεται ο μέσος τους όρος. Ο υπολογισμός αυτός εκτελείται μία και μόνο φορά, κατά τη στιγμή της δημιουργίας της αναφοράς, και το αποτέλεσμα αποθηκεύεται ως επιπλέον πεδίο στο ίδιο έγγραφο της συλλογής environmental_reports (denormalization). Με τον τρόπο αυτό, όποιος επόμενος υπολογισμός χρειάζεται τη βαθμολογία μιας αναφοράς -- όπως η ενημέρωση ενός κελιού πλέγματος ή των συγκεντρωτικών στατιστικών -- τη διαβάζει απευθείας, χωρίς να την ανασυνθέτει από τις επιμέρους αξιολογήσεις.

6.3 ΧΩΡΙΚΗ ΟΜΑΔΟΠΟΙΗΣΗ ΚΑΙ ΕΝΗΜΕΡΩΣΗ ΠΛΕΓΜΑΤΟΣ (HEATMAP_CELLS)

Το δεύτερο βήμα αντιστοιχίζει τη γεωγραφική θέση της αναφοράς σε ένα κελί του κανονικού πλέγματος που περιγράφηκε στην Ενότητα 3.2.3. Η αντιστοίχιση αυτή γίνεται διαιρώντας τις συντεταγμένες της αναφοράς με τη σταθερά GRID_SIZE = 0,005 (σε μοίρες γεωγραφικού πλάτους/μήκους) και στρογγυλοποιώντας το πηλίκο προς τα κάτω (floor) σε κάθε διάσταση, ώστε όλες οι θέσεις που βρίσκονται εντός της ίδιας μικρής γεωγραφικής περιοχής να παράγουν το ίδιο αναγνωριστικό κελιού (π.χ. τη συνένωση των στρογγυλοποιημένων συντεταγμένων σε μία συμβολοσειρά). Με βάση αυτό το αναγνωριστικό, η συνάρτηση ανακτά (ή δημιουργεί, εφόσον πρόκειται για την πρώτη αναφορά στο συγκεκριμένο κελί) το αντίστοιχο έγγραφο της συλλογής heatmap_cells και ενημερώνει το πλήθος των αναφορών του κελιού, καθώς και τον τρέχοντα μέσο όρο βαθμολογίας όχλησης, συνυπολογίζοντας τη νέα τιμή. Το σύνολο αυτών των ενημερωμένων κελιών αποτελεί ακριβώς τα δεδομένα που καταναλώνει η προβολή θερμικού χάρτη της οθόνης MapScreen (Ενότητα 5.4).

Η Εικόνα 16 παρουσιάζει το τμήμα κώδικα που υπολογίζει το αναγνωριστικό του κελιού πλέγματος και ενημερώνει το αντίστοιχο έγγραφο της συλλογής heatmap_cells.



```
// ---- 1) Ενημέρωση χωρικού κελιού πλέγματος (για το heatmap) ----
const cellLatIdx = Math.floor(lat / GRID_SIZE);
const cellLngIdx = Math.floor(lng / GRID_SIZE);
const cellId = `${cellLatIdx}_${cellLngIdx}`;
const cellRef = db.collection("heatmap_cells").doc(cellId);

await db.runTransaction(async (tx) => {
  const cellDoc = await tx.get(cellRef);
  if (!cellDoc.exists) {
    tx.set(cellRef, {
      count: 1,
      sumLat: lat,
      sumLng: lng,
      sumScore: score,
      avgLat: lat,
      avgLng: lng,
      avgScore: score,
      updatedAt: FieldValue.serverTimestamp(),
    });
  } else {
    const d = cellDoc.data();
    const count = (d.count || 0) + 1;
    const sumLat = (d.sumLat || 0) + lat;
    const sumLng = (d.sumLng || 0) + lng;
    const sumScore = (d.sumScore || 0) + score;
    tx.update(cellRef, {
      count,
      sumLat,
      sumLng,
      sumScore,
      avgLat: sumLat / count,
      avgLng: sumLng / count,
      avgScore: sumScore / count,
      updatedAt: FieldValue.serverTimestamp(),
    });
  }
});
```

Εικόνα 16: Υπολογισμός αναγνωριστικού κελιού και ενημέρωση της συλλογής heatmap_cells.

Στο σημείο αυτό οι συντεταγμένες της αναφοράς στρογγυλοποιούνται στη σταθερή ακρίβεια που περιγράφηκε στην Ενότητα 6.3, ώστε να παραχθεί το αναγνωριστικό cellId, με βάση το οποίο ανακτάται ή δημιουργείται το αντίστοιχο έγγραφο της συλλογής heatmap_cells.

6.4 ΕΝΗΜΕΡΩΣΗ ΣΥΓΚΕΝΤΡΩΤΙΚΩΝ ΣΤΑΤΙΣΤΙΚΩΝ (STATS/SUMMARY, DAILY_STATS)

Το τρίτο βήμα ενημερώνει παράλληλα δύο συγκεντρωτικά έγγραφα. Το έγγραφο stats/summary συγκεντρώνει δεδομένα σε επίπεδο ολόκληρου του συστήματος: το συνολικό πλήθος αναφορών, την κατανομή των αξιολογήσεων ανά κατηγορία όχλησης και τον μέσο όρο της μετρηθείσας στάθμης θορύβου, όπως ορίστηκαν στην Ενότητα 3.2.4. Παράλληλα, στη συλλογή daily_stats ενημερώνεται (ή δημιουργείται) το έγγραφο που αντιστοιχεί στην τρέχουσα ημερολογιακή ημέρα, προσαυξάνοντας το πλήθος αναφορών της. Η οθόνη StatisticsScreen (Ενότητα 5.5) διαβάζει απευθείας τα δύο αυτά έγγραφα/συλλογή, χωρίς να απαιτείται καμία δική της συγκεντρωτική επεξεργασία επί του συνόλου των επιμέρους αναφορών.

6.5 ΑΤΟΜΙΚΟΤΗΤΑ ΚΑΙ ΣΥΝΑΛΛΑΓΕΣ (TRANSACTIONS)

Καθένα από τα τρία βήματα ενημέρωσης των Ενοτήτων 6.3 και 6.4 συνίσταται σε μια ανάγνωση της τρέχουσας τιμής (πλήθος, μέσος όρος) και μια εγγραφή της νέας, ενημερωμένης τιμής. Εάν πολλοί χρήστες υποβάλλουν αναφορές ταυτόχρονα για το ίδιο κελί πλέγματος ή την ίδια ημέρα, μια απλή ανάγνωση-και-εγγραφή θα κινδύνευε να χάσει ενημερώσεις (lost update), καθώς δύο ταυτόχρονες εκτελέσεις της συνάρτησης θα μπορούσαν να διαβάσουν την ίδια αρχική τιμή και να αγνοήσουν η μία την ενημέρωση της άλλης. Για τον λόγο αυτό, όλες οι ενημερώσεις εκτελούνται μέσα σε συναλλαγές (transactions) της Cloud Firestore, οι οποίες εγγυώνται ότι η ανάγνωση και η εγγραφή κάθε εγγράφου εκτελούνται ως ενιαία, ατομική πράξη· σε περίπτωση σύγκρουσης, η Firestore επαναλαμβάνει αυτόματα τη συναλλαγή, εξασφαλίζοντας ορθά αποτελέσματα ακόμη και υπό υψηλή ταυτοχρονισμότητα υποβολών.

Η Εικόνα 17 παρουσιάζει τη χρήση συναλλαγής (transaction) της Cloud Firestore κατά την ενημέρωση ενός κελιού πλέγματος.

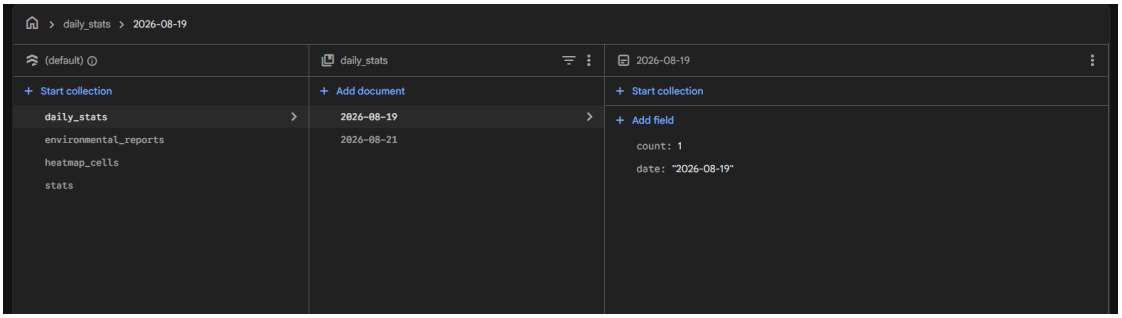
```
await db.runTransaction(async (tx) => {
  const cellDoc = await tx.get(cellRef);
  if (!cellDoc.exists) {
    tx.set(cellRef, {
      count: 1,
      sumLat: lat,
      sumLng: lng,
      sumScore: score,
      avgLat: lat,
      avgLng: lng,
      avgScore: score,
      updatedAt: FieldValue.serverTimestamp(),
    });
  } else {
    const d = cellDoc.data();
    const count = (d.count || 0) + 1;
    const sumLat = (d.sumLat || 0) + lat;
    const sumLng = (d.sumLng || 0) + lng;
    const sumScore = (d.sumScore || 0) + score;
    tx.update(cellRef, {
      count,
      sumLat,
      sumLng,
      sumScore,
      avgLat: sumLat / count,
      avgLng: sumLng / count,
      avgScore: sumScore / count,
      updatedAt: FieldValue.serverTimestamp(),
    });
  }
});
```

Εικόνα 17: Χρήση συναλλαγής (transaction) της Cloud Firestore.

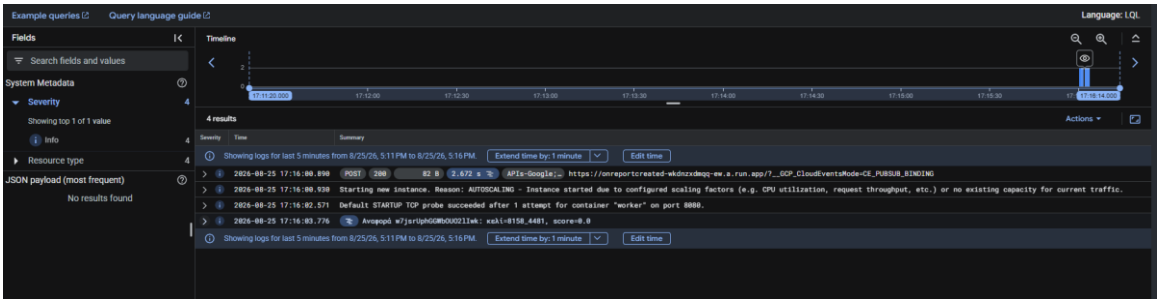
Όπως εξηγήθηκε στην Ενότητα 6.5, η ανάγνωση της τρέχουσας τιμής και η εγγραφή της ενημερωμένης τιμής εκτελούνται εντός της ίδιας συναλλαγής, ώστε να αποφεύγεται η απώλεια ενημερώσεων όταν πολλαπλές αναφορές αφορούν το ίδιο κελί ταυτόχρονα.

6.6 ΑΝΑΠΤΥΞΗ ΚΑΙ ΠΑΡΑΚΟΛΟΥΘΗΣΗ

Η ανάπτυξη (deployment) της συνάρτησης στην υποδομή της Google γίνεται μέσω του Firebase CLI, με την εντολή firebase deploy --only functions, η οποία πακετάρει τον κώδικα του φακέλου functions/ (Ενότητα 4.6.2) και τον αναρτά στην πλατφόρμα, χωρίς ο προγραμματιστής να διαχειρίζεται ο ίδιος οποιονδήποτε server ή υποδομή. Μετά την ανάπτυξη, κάθε εκτέλεση της onReportCreated -- καθώς και τυχόν σφάλματα κατά την εκτέλεσή της -- καταγράφεται αυτόματα στην υπηρεσία Cloud Logging, στην οποία ο προγραμματιστής μπορεί να ανατρέξει μέσω της κονσόλας Firebase, για την παρακολούθηση της συμπεριφοράς και την διάγνωση τυχόν προβλημάτων του συστήματος σε πραγματικές συνθήκες λειτουργίας.



Εικόνα 18: Οι συλλογές της βάσης δεδομένων στην κονσόλα Firebase.



Εικόνα 19: Καταγραφή (log) επιτυχούς εκτέλεσης της συνάρτησης onReportCreated.

7ο ΚΕΦΑΛΑΙΟ: ΔΟΚΙΜΕΣ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΣΥΣΤΗΜΑΤΟΣ

7.1 ΜΕΘΟΔΟΛΟΓΙΑ ΔΟΚΙΜΩΝ

Οι δοκιμές που παρουσιάζονται στην ενότητα αυτή είναι χειροκίνητες λειτουργικές δοκιμές (manual functional testing), δηλαδή δοκιμές «μαύρου κουτιού» (black-box testing) στις οποίες κάθε λειτουργία της εφαρμογής εκτελέστηκε από τον χρήστη-δοκιμαστή μέσω της κανονικής διεπαφής χρήστη, χωρίς πρόσβαση ή έλεγχο του εσωτερικού κώδικα κατά την εκτέλεση. Οι δοκιμές έγιναν σε πραγματική κινητή συσκευή Android, με ενεργή σύνδεση δικτύου και ενεργοποιημένες τις απαιτούμενες άδειες τοποθεσίας και μικροφώνου, ώστε οι συνθήκες εκτέλεσης να προσομοιώνουν όσο το

δυνατόν πιστότερα την πραγματική χρήση της εφαρμογής από έναν τελικό χρήστη. Στόχος των δοκιμών ήταν η επαλήθευση ότι κάθε λειτουργική απαίτηση της Ενότητας 2.2 υλοποιείται ορθά στην τελική εφαρμογή, καλύπτοντας το πλήρες κύκλωμα από την καταγραφή μιας παρατήρησης στη συσκευή του χρήστη έως την αυτόματη επεξεργασία της στον server (Cloud Functions) και την προβολή του αποτελέσματος στον χάρτη και στα στατιστικά.

7.2 ΛΕΙΤΟΥΡΓΙΚΕΣ ΔΟΚΙΜΕΣ

Με βάση τις λειτουργικές απαιτήσεις της Ενότητας 2.2, καταρτίστηκε ένα σύνολο περιπτώσεων δοκιμών (test cases) για τον χειροκίνητο έλεγχο της ορθής λειτουργίας της εφαρμογής. Στον Πίνακα 3 παρουσιάζονται το σενάριο, τα βήματα εκτέλεσης και το αναμενόμενο αποτέλεσμα κάθε δοκιμής, καθώς και το πραγματικό αποτέλεσμα και η κατάστασή της (Pass/Fail) κατά την εκτέλεσή τους.

Πίνακας 3: Περιπτώσεις λειτουργικών δοκιμών (test cases) της εφαρμογής.

Α/Α	**Σενάριο Δοκιμής**	**Βήματα Εκτέλεσης**	**Αναμενόμενο Αποτέλεσμα**	**Πραγματικό Αποτέλεσμα**	**Κατάσταση (Pass/Fail)**
1	Ανώνυμη ταυτοποίηση χρήστη	Πρώτη εκκίνηση της εφαρμογής, χωρίς προηγούμενη σύνδεση.	Η εφαρμογή πραγματοποιεί αυτόματα ανώνυμη ταυτοποίηση (Firebase Anonymous Authentication), χωρίς να ζητά διαπιστευτήρια από τον χρήστη.	Ανώνυμη ταυτοποίηση εκτελέστηκε αυτόματα κατά την εκκίνηση, χωρίς εμφάνιση φόρμας σύνδεσης.	Επιτυχής (Pass)
2	Λήψη γεωγραφικής θέσης μέσω	Παραχώρηση άδειας τοποθεσίας κατά την πρώτη	Η εφαρμογή λαμβάνει το γεωγραφικό πλάτος/μήκος της	Ο χρήστης εντοπίστηκε σωστά στον χάρτη	Επιτυχής (Pass)

	GPS	χρήση· μετάβαση στην καρτέλα «Χάρτης».	τρέχουσας θέσης του χρήστη και τον εντοπίζει στον χάρτη.	(Εικόνα 6)· τα markers εμφανίστηκαν στις σωστές συντεταγμένες (Εικόνα 10).	
3	Μέτρηση στάθμης θορύβου	Μετάβαση στην καρτέλα «Καταγραφή» και εκκίνηση μέτρησης μέσω του μικροφώνου της συσκευής.	Εμφανίζεται η μετρημένη στάθμη θορύβου σε dB καθώς και η αντίστοιχη αυτόματη ταξινόμηση επιπέδου θορύβου.	Η στάθμη θορύβου (dB) και το επίπεδο θορύβου εμφανίστηκαν σωστά στη φόρμα (Εικόνα 8).	Επιτυχής (Pass)
4	Καταγραφή αξιολογήσεων πέντε κατηγοριών όχλησης	Επιλογή τιμής για καθεμία από τις πέντε κατηγορίες (θόρυβος, δυσοσμία, θερμική δυσφορία, ποιότητα αέρα, οπτική ρύπανση) και προαιρετική συμπλήρωση ελεύθερων σημειώσεων.	Οι επιλεγμένες τιμές εμφανίζονται ενημερωμένες στη φόρμα πριν την υποβολή.	Οι επιλογές στις 5 κατηγορίες (4/4/5/4/4 τιμές) καταγράφηκαν σωστά στη φόρμα (Εικόνα 8).	Επιτυχής (Pass)
5	Υποβολή αναφοράς	Πάτημα του κουμπιού υποβολής	Δημιουργείται νέο έγγραφο στη συλλογή Firestore	Δημιουργήθηκε νέο έγγραφο	Επιτυχής (Pass)

	(report)	(μέθοδος _submit()) μετά τη συμπλήρωση της φόρμας.	environmental_reports, με τα πεδία θέσης, χρονοσήμανσης, μετρήσεων και αξιολογήσεων.	στο environmental_reports (Εικόνα 18)· η onReportCreated εκτελέστηκε επιτυχώς (Εικόνα 19).	
6	Προβολή σημείων (markers) στον χάρτη	Μετάβαση στην καρτέλα «Χάρτης» μετά την υποβολή τουλάχιστον μίας αναφοράς.	Εμφανίζεται δείκτης (marker) στο σημείο υποβολής, χρωματισμένος ανάλογα με τον βαθμό όχλησης της αναφοράς.	Εμφανίστηκε marker στο σημείο υποβολής, χρωματισμένος ανάλογα με την όχληση (Εικόνα 10).	Επιτυχής (Pass)
7	Προβολή θερμικού χάρτη (heatmap)	Πάτημα του κουμπιού εναλλαγής προβολής στην καρτέλα «Χάρτης».	Ο χάρτης εναλλάσσεται σε προβολή θερμικού χάρτη, με κύκλους (Circle) χρωματισμένους ανά κελί πλέγματος.	Ο χάρτης πέρασε σε προβολή heatmap με κύκλους χρωματισμένους ανά κελί πλέγματος (Εικόνα 11).	Επιτυχής (Pass)
8	Προβολή συγκεντρωτικών	Μετάβαση στην καρτέλα «Στατιστικά».	Εμφανίζονται συγκεντρωτικά στατιστικά (κατανομές ανά	Εμφανίστηκαν τα συγκεντρωτικά	Επιτυχής (Pass)

	στατιστικών		κατηγορία αξιολόγησης, μέσος όρος θορύβου, τάση πλήθους αναφορών των τελευταίων ημερών).	στατιστικά από τη συλλογή stats/summary (Εικόνα 13).	

7.3 ΣΥΝΟΨΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΔΟΚΙΜΩΝ

Και οι οκτώ (8) περιπτώσεις λειτουργικών δοκιμών του Πίνακα 3 ολοκληρώθηκαν επιτυχώς (Pass): η εφαρμογή υλοποιεί ορθά όλες τις λειτουργικές απαιτήσεις της Ενότητας 2.2, από την ανώνυμη ταυτοποίηση και την υποβολή μιας αναφοράς έως την αυτόματη επεξεργασία της στον server και την προβολή του αποτελέσματος στον χρήστη μέσω χάρτη, θερμικού χάρτη και στατιστικών. Δεν εντοπίστηκε κάποιο λειτουργικό σφάλμα κατά την εκτέλεση των δοκιμών. Οι γνωστοί περιορισμοί του συστήματος σχετίζονται κυρίως με την εξάρτησή του από εξωτερικούς πόρους της συσκευής (GPS, δίκτυο), για τους οποίους έχει ήδη προβλεφθεί ανθεκτική διαχείριση σφαλμάτων, όπως περιγράφεται στην Ενότητα 4.2, καθώς και με το γεγονός ότι οι παρούσες δοκιμές καλύπτουν τη χειροκίνητη λειτουργική συμπεριφορά της εφαρμογής και όχι πτυχές όπως η απόδοση υπό υψηλό φόρτο. Η συμπεριφορά του συστήματος υπό ταυτόχρονη πρόσβαση πολλαπλών χρηστών εξετάζεται ξεχωριστά, με πραγματική δοκιμή, στην Ενότητα 7.4.

7.4 ΔΟΚΙΜΗ ΤΑΥΤΟΧΡΟΝΗΣ ΠΡΟΣΒΑΣΗΣ ΠΟΛΛΑΠΛΩΝ ΧΡΗΣΤΩΝ (CONCURRENCY TEST)

Πέραν των χειροκίνητων λειτουργικών δοκιμών των Ενοτήτων 7.2–7.3, κρίθηκε απαραίτητο να ελεγχθεί ρητά η συμπεριφορά του συστήματος όταν πολλαπλοί χρήστες υποβάλλουν αναφορές ταυτόχρονα, καθώς πρόκειται για ρεαλιστικό σενάριο χρήσης μιας εφαρμογής crowdsensing που σχετίζεται άμεσα με τα κοινόχρηστα συγκεντρωτικά έγγραφα της Ενότητας 6.4 (heatmap_cells, stats/summary, daily_stats), τα οποία ενημερώνονται από πολλαπλές, ενδεχομένως ταυτόχρονες, εκτελέσεις της συνάρτησης onReportCreated. Για τον σκοπό αυτό αναπτύχθηκε ξεχωριστό αυτοματοποιημένο σενάριο δοκιμής (Node.js, βιβλιοθήκη firebase), το οποίο συνδέεται στο πραγματικό, αναπτυγμένο (deployed) έργο Firebase του συστήματος, πραγματοποιεί ανώνυμη

ταυτοποίηση με τον ίδιο τρόπο όπως η εφαρμογή, και υποβάλλει πραγματικά έγγραφα στη συλλογή environmental_reports μέσω της ίδιας ακριβώς κλήσης (addDoc) που χρησιμοποιεί η εφαρμογή Flutter, ώστε η δοκιμή να ενεργοποιεί την πραγματική συνάρτηση onReportCreated και όχι κάποια προσομοίωσή της. Οι εγγραφές αποστέλλονται ταυτόχρονα μέσω Promise.allSettled, ώστε να προκαλείται πραγματικός ανταγωνισμός (contention) στα κοινόχρηστα έγγραφα, και στη συνέχεια το σενάριο δοκιμής περιμένει έως ότου ολοκληρωθεί η ασύγχρονη επεξεργασία στον server και συγκρίνει τις τελικές τιμές των συγκεντρωτικών εγγράφων με τις τιμές που αναμένονται αν καμία ενημέρωση δεν έχει χαθεί.

Εκτελέστηκαν δύο σενάρια. Στο πρώτο (Δοκιμή 1), N=30 αναφορές υποβλήθηκαν ταυτόχρονα με γεωγραφικές συντεταγμένες που πέφτουν όλες στο ίδιο κελί του πλέγματος — η χειρότερη περίπτωση, καθώς και οι 30 εκτελέσεις της συνάρτησης προσπαθούν να ενημερώσουν, μέσα στην ίδια συναλλαγή (transaction) της Ενότητας 6.5, το ίδιο έγγραφο heatmap_cells/{cellId}. Στο δεύτερο (Δοκιμή 2), M=20 αναφορές υποβλήθηκαν ταυτόχρονα σε K=4 διαφορετικά κελιά (5 ανά κελί), για τον έλεγχο της συνολικής συμπεριφοράς υπό ηπιότερο ανταγωνισμό.

Στη Δοκιμή 1, και οι 30 εγγραφές ολοκληρώθηκαν επιτυχώς εντός 1.753 ms, και το σύνολο των επηρεαζόμενων συγκεντρωτικών εγγράφων ενημερώθηκε πλήρως εντός 7,1 δευτερολέπτων. Ο έλεγχος επιβεβαίωσε ότι το πλήθος (count) του κελιού-στόχου αυξήθηκε κατά ακριβώς 30, το άθροισμα βαθμολογίας (sumScore) και ο μέσος όρος (avgScore = sumScore / count) προέκυψαν αριθμητικά συνεπή, το συνολικό πλήθος (stats/summary.total) αυξήθηκε κατά 30, το ημερήσιο πλήθος (daily_stats.count) αυξήθηκε επίσης κατά 30, και —μετά από ξεχωριστό έλεγχο της πραγματικής δομής του εγγράφου stats/summary, βλ. παρακάτω— και οι τέσσερις κατανομές (noiseLevelDist, airQualityDist, odorDist, thermalDist) ενημερώθηκαν επίσης σωστά κατά 30 η καθεμία. Καμία από τις 30 ταυτόχρονες ενημερώσεις δεν χάθηκε. Στη Δοκιμή 2, και οι 20 εγγραφές ολοκληρώθηκαν επιτυχώς εντός 762 ms, με πλήρη ενημέρωση των συγκεντρωτικών εντός 2,9 δευτερολέπτων· το πλήθος ενημερώθηκε σωστά και στα 4 κελιά (5/5 ανά κελί) και το συνολικό πλήθος (total) αυξήθηκε κατά 20.

Πίνακας 4: Αποτελέσματα δοκιμής ταυτόχρονης πρόσβασης πολλαπλών χρηστών.

Α/Α	**Έλεγχος**	**Αναμενόμενο**	**Πραγματικό**	**Κατάσταση**
1	Δ1 – πλήθος	+30	+30	Επιτυχής

	αναφορών στο κελί-στόχο			(Pass)
2	Δ1 – άθροισμα βαθμολογίας κελιού (sumScore)	+30 × 45,00	+1350,00	Επιτυχής (Pass)
3	Δ1 – μέσος όρος κελιού (avgScore = sumScore/count)	45,00	45,00	Επιτυχής (Pass)
4	Δ1 – συνολικό πλήθος (stats/summary.total)	+30	+30	Επιτυχής (Pass)
5	Δ1 – κατανομές noiseLevelDist / airQualityDist / odorDist / thermalDist	+30 η καθεμία	+30 η καθεμία	Επιτυχής (Pass)
6	Δ1 – ημερήσιο πλήθος (daily_stats.count)	+30	+30	Επιτυχής (Pass)
7	Δ2 – πλήθος ανά κελί (4 κελιά × 5 αναφορές)	+5 ανά κελί	+5 ανά κελί	Επιτυχής (Pass)
8	Δ2 – συνολικό πλήθος	+20	+20	Επιτυχής (Pass)

	(stats/summary .total)			

Κατά την πρώτη εκτέλεση της δοκιμής, ο έλεγχος των δύο κατανομών noiseLevelDist και airQualityDist φάνηκε αρχικά να αποτυγχάνει (τιμή 0 αντί της αναμενόμενης), ενώ όλοι οι υπόλοιποι έλεγχοι της ίδιας κλήσης συνάρτησης —συμπεριλαμβανομένου του total— ήταν επιτυχείς. Με απευθείας επιθεώρηση της ακατέργαστης δομής του εγγράφου stats/summary διαπιστώθηκε ότι δεν επρόκειτο για απώλεια δεδομένων, αλλά για σφάλμα στον τρόπο ελέγχου: τα ονόματα πεδίων με δυναμικό κλειδί (π.χ. noiseLevelDist.${report.noiseLevel}) που χρησιμοποιούνται στην Ενότητα 6.4 μέσα σε κλήση set(..., {merge: true}) αποθηκεύονται από το Admin SDK ως πεδία πρώτου επιπέδου με κυριολεκτική τελεία στο όνομα (π.χ. "noiseLevelDist.2"), και όχι ως εμφωλευμένος χάρτης, όπως θα συνέβαινε με τη μέθοδο update(). Μετά τη διόρθωση του ελέγχου ώστε να διαβάζει το σωστό, πραγματικό όνομα πεδίου, και οι 12 έλεγχοι και των δύο σεναρίων ήταν επιτυχείς (βλ. Πίνακα 4). Το εύρημα αυτό δεν επηρεάζει την ορθότητα του συστήματος — οι τιμές ενημερώνονται σωστά και ατομικά σε κάθε περίπτωση — αποτελεί ωστόσο χρήσιμη τεχνική λεπτομέρεια για όποιον προσπελάσει τα πεδία αυτά απευθείας.

Συνολικά γράφτηκαν 50 πραγματικές δοκιμαστικές αναφορές (30 στη Δοκιμή 1, 20 στη Δοκιμή 2) στο ενεργό έργο Firebase, χωρίς καμία απώλεια ή ασυνέπεια σε κανένα από τα συγκεντρωτικά έγγραφα. Η δοκιμή επιβεβαιώνει εμπειρικά ότι ο μηχανισμός συναλλαγής (transaction) της Ενότητας 6.5 εξασφαλίζει σωστή σειριοποίηση των ταυτόχρονων ενημερώσεων στο ίδιο έγγραφο heatmap_cells, και ότι οι ατομικές (atomic) πράξεις FieldValue.increment() της Ενότητας 6.4 δεν χάνουν ενημερώσεις υπό ταυτόχρονη φόρτωση. Η δοκιμή αυτή δεν υποκαθιστά δοκιμή απόδοσης υπό πραγματικό φορτίο πολλών εκατοντάδων ή χιλιάδων ταυτόχρονων χρηστών, καλύπτει όμως άμεσα το ερώτημα ορθότητας (correctness) που τέθηκε: τι συμβαίνει όταν περισσότεροι από έναν χρήστες γράφουν ταυτόχρονα.

8ο ΚΕΦΑΛΑΙΟ: ΣΥΜΠΕΡΑΣΜΑΤΑ

8.1 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα πτυχιακή εργασία σχεδιάστηκε και υλοποιήθηκε ένα πλήρες σύστημα crowdsensing για τη συλλογική ανίχνευση συνθηκών όχλησης σε αστικό περιβάλλον, μέσω μιας κινητής εφαρμογής. Ο αρχικός στόχος ήταν η ανάπτυξη ενός συστήματος στο οποίο απλοί πολίτες, χωρίς εξειδικευμένο εξοπλισμό, θα μπορούν να καταγράφουν

τις υποκειμενικές τους παρατηρήσεις για πέντε κατηγορίες αστικής όχλησης, οι παρατηρήσεις αυτές να συγκεντρώνονται και να επεξεργάζονται αυτόματα σε ένα σύστημα cloud, και το αποτέλεσμα να αποδίδεται πίσω στους χρήστες σε μορφή κατανοητή, μέσω χάρτη και στατιστικών. Ο στόχος αυτός επιτεύχθηκε: η εφαρμογή που αναπτύχθηκε σε Flutter επιτρέπει την ανώνυμη, ταχεία υποβολή μιας αναφοράς, ενώ οι συναρτήσεις Cloud Functions που περιγράφηκαν στο Κεφάλαιο 6 αναλαμβάνουν αυτόματα τον υπολογισμό της βαθμολογίας όχλησης, τη χωρική ομαδοποίηση των αναφορών σε ένα γεωγραφικό πλέγμα και την ενημέρωση συγκεντρωτικών στατιστικών, χωρίς καμία επιπλέον ενέργεια από τον χρήστη.

Η αρχιτεκτονική που επιλέχθηκε -- μια αρχιτεκτονική serverless, βασισμένη σε συμβάντα (event-driven), πάνω στην πλατφόρμα Firebase -- αποδείχθηκε ιδιαίτερα κατάλληλη για το συγκεκριμένο πρόβλημα. Επέτρεψε τον διαχωρισμό της κινητής εφαρμογής, η οποία παραμένει απλή και αφοσιωμένη στη συλλογή δεδομένων και στην παρουσίαση ήδη επεξεργασμένων αποτελεσμάτων, από την ουσιαστική επεξεργασία των δεδομένων, η οποία εκτελείται κεντρικά στον server. Παράλληλα, μηχανισμοί όπως οι συναλλαγές (transactions) της Cloud Firestore (Ενότητα 6.5) εξασφάλισαν την ορθότητα των συγκεντρωτικών δεδομένων ακόμη και υπό ταυτόχρονες υποβολές πολλών χρηστών, χωρίς να απαιτηθεί η συγγραφή ή η συντήρηση οποιασδήποτε δικής μας υποδομής server.

Συνολικά, το τελικό σύστημα ικανοποιεί τη διττή απαίτηση που τέθηκε στην αρχή της εργασίας: αφενός τη συλλογή δεδομένων περιβαλλοντικής όχλησης από πλήθος χρηστών μέσω μιας απλής κινητής εφαρμογής, αφετέρου την αποθήκευση, την επεξεργασία και την οπτικοποίησή τους μέσω ενός σύγχρονου συστήματος cloud, παρέχοντας μια πρακτική, επεκτάσιμη και άμεσα αξιοποιήσιμη λύση για την παρακολούθηση συνθηκών σε αστικό περιβάλλον.

8.2 ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ

Το σύστημα που αναπτύχθηκε αφήνει αρκετά περιθώρια για μελλοντική επέκταση. Μια πρώτη κατεύθυνση είναι η ενσωμάτωση δεδομένων από πρόσθετους αισθητήρες, πέραν του μικροφώνου της συσκευής, όπως εξωτερικοί αισθητήρες ποιότητας αέρα συνδεδεμένοι μέσω Bluetooth, ώστε οι υποκειμενικές αξιολογήσεις των χρηστών να συνδυάζονται με αντικειμενικές μετρήσεις για περισσότερες από τις πέντε κατηγορίες όχλησης. Μια δεύτερη κατεύθυνση αφορά τη διασφάλιση της ποιότητας των δεδομένων: με μεγαλύτερο αριθμό χρηστών, θα ήταν σκόπιμη η προσθήκη μηχανισμών ανίχνευσης

ακραίων ή αναξιόπιστων τιμών (outlier detection) στην πλευρά του server, πριν αυτές συνυπολογιστούν στα συγκεντρωτικά στατιστικά και στον θερμικό χάρτη.

Μια τρίτη κατεύθυνση είναι η ενημέρωση των χρηστών με ενεργό τρόπο, αντί της παθητικής προβολής του χάρτη: αξιοποιώντας την ήδη υπάρχουσα υποδομή Cloud Functions, θα ήταν εφικτή η αποστολή ειδοποιήσεων (push notifications) όταν ο μέσος όρος βαθμολογίας όχλησης ενός κελιού πλέγματος υπερβαίνει ένα όριο, ενημερώνοντας άμεσα τους χρήστες που βρίσκονται ή κινούνται προς την αντίστοιχη περιοχή. Τέλος, όπως και σε συγγενή συστήματα crowdsensing, η χρησιμότητα της εφαρμογής θα ενισχυόταν σημαντικά με τη διεύρυνση της βάσης χρηστών σε πραγματικές συνθήκες λειτουργίας, τόσο ώστε τα στατιστικά να γίνουν πιο αντιπροσωπευτικά, όσο και ώστε τα ήδη συλλεγμένα δεδομένα να μπορούν πιθανώς να αξιοποιηθούν, σε συνεργασία με τοπικούς φορείς, ως συμπληρωματική πηγή πληροφορίας για την παρακολούθηση συνθηκών σε αστικό περιβάλλον.

ΑΝΑΦΟΡΕΣ

[1] R. K. Ganti, F. Ye, and H. Lei, «Mobile crowdsensing: Current state and future challenges,» IEEE Communications Magazine, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[2] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, «Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm,» ACM Computing Surveys, vol. 48, no. 1, pp. 1–31, 2015.

[3] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, «A survey of mobile phone sensing,» IEEE Communications Magazine, vol. 48, no. 9, pp. 140–150, Sept. 2010.

[4] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, «Participatory sensing,» in Workshop on World-Sensor-Web (WSW'06), ACM SenSys, Boulder, CO, USA, 2006.

[5] H. Ma, D. Zhao, and P. Yuan, «Opportunities in mobile crowd sensing,» IEEE Communications Magazine, vol. 52, no. 8, pp. 29–35, Aug. 2014.

[6] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels, «NoiseTube: Measuring and mapping noise pollution with mobile phones,» in Information Technologies in Environmental Engineering (ITEE 2009), Thessaloniki, Greece, Springer, 2009, pp. 215–228.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, «Internet of Things for smart cities,» IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[8] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, «Urban computing: Concepts, methodologies, and applications,» ACM Transactions on Intelligent Systems and Technology, vol. 5, no. 3, pp. 1–55, 2014.

[9] P. Mell and T. Grance, «The NIST definition of cloud computing,» National Institute of Standards and Technology, Special Publication 800-145, 2011.

[10] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, «Serverless computing: Current trends and open problems,» in Research Advances in Cloud Computing, Springer, 2017, pp. 1–20.

[11] L. Moroney, The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017.

[12] P. J. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Upper Saddle River, NJ: Addison-Wesley, 2012.

[13] P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, Geographic Information Science and Systems, 4th ed. Hoboken, NJ: Wiley, 2015.

[14] L. Wilkinson and M. Friendly, «The history of the cluster heat map,» The American Statistician, vol. 63, no. 2, pp. 179–184, 2009.

[15] E. Windmill, Flutter in Action. Shelter Island, NY: Manning Publications, 2020.

[16] Flutter, «Flutter documentation,» Google. [Online]. Available: https://docs.flutter.dev

[17] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, «Ear-Phone: An end-to-end participatory urban noise mapping system,» in Proc. 9th ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN'10), Stockholm, Sweden, 2010, pp. 105–116.

[18] E. Kanjo, «NoiseSPY: A real-time mobile phone platform for urban noise monitoring and mapping,» Mobile Networks and Applications, vol. 15, no. 4, pp. 562–574, 2010.