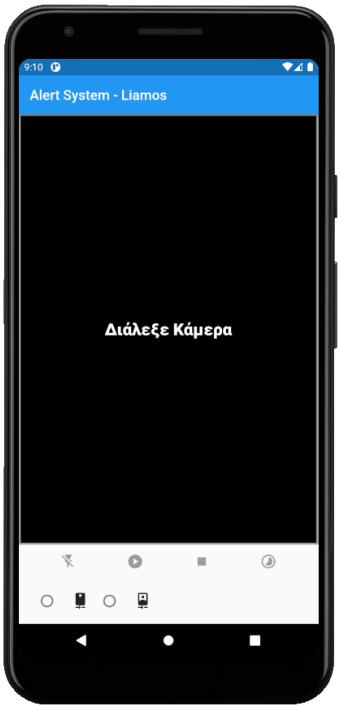
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Φορητό σύστημα απομακρυσμένης επίβλεψης χώρου»



Φοιτητής

Λιάμος Δημήτριος 516073

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Φεβρουάριος 2022

Φορητό σύστημα απομακρυσμένης επίβλεψης χώρου

Κωδικός: 21224

Φοιτητής: Λιάμος Δημήτριος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 30-03-2021

Ημερομηνία περάτωσης Π.Ε. 10-01-2022

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

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

Περίληψη

Η εργασία αυτή αφορά την υλοποίηση φορητού συστήματος που θα επιβλέπει ένα χώρο με κάμερα και θα ειδοποιεί τον χρήστη για παραβίαση. Για παράδειγμα το σύστημα μπορεί να χρησιμοποιηθεί από ταξιδιώτες που θέλουν να παρακολουθούν το δωμάτιο του ξενοδοχείου που διανέμουν όταν λείπουν από αυτό.

Υλοποιήθηκε ένα σύστημα παρακολούθησης χώρου ανιχνεύοντας κίνηση σε αυτόν και αποστέλλει ειδοποίηση στον χρήστη μέσω διαδικτύου στο κινητό του. Το κινητό τοποθετείται σε ένα χώρο επίβλεψης με την κάμερα να επιβλέπει το οπτικό της πεδίο. Συνήθως εστιάζει στην είσοδο του δωματίου αλλά θα μπορούσε να είναι και κάποια πόρτα μπαλκονιού ή παράθυρο. Η εφαρμογή στο κινητό παρακολουθεί με βίντεο το πεδίο και επεξεργάζεται την διαφορά μεταξύ δύο διαδοχικών frames για να ανιχνεύσει κίνηση στον χώρο. Κατά την ανίχνευση κίνησης αποστέλλει την ειδοποίηση σε server.

« Portable monitoring and security system»

Abstract

This work concerns the implementation of a portable system that monitor a space with sensors and notify the user when detect a violation. For example, the system can be used by travelers who want to monitor the hotel room when they are away.

A monitoring system was implemented by detecting motion-violation and sending a notification to the user via internet on his mobile phone. The phone is placed in a surveillance area with the camera monitoring its field of view. It usually focuses on the entrance of the room but it could also be a balcony door or window. The mobile application monitors the field with video and processes the difference between two consecutive frames to detect movement in space. When detect a motion it sends the notification to server.

Ευχαριστίες

Να ευχαριστήσω τους γονείς μου και τον κ. Τσιακμάκη για τη πολύτιμη καθοδήγηση του.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Οι οικιακές κάμερες ασφαλείας παρακολουθούν το σπίτι τόσο από μέσα όσο και από έξω, λειτουργώντας ως ένα επιπλέον ζευγάρι μάτια και αυτιά για την παρακολούθηση της ιδιοκτησίας. Υπάρχουν πολλά πλεονεκτήματα της εγκατάστασης καμερών ασφαλείας εσωτερικών και εξωτερικών χώρων, από τη δυνατότητά τους να αποτρέπουν τους εγκληματίες έως άλλες εφαρμογές, όπως οθόνες για κατοικίδια και μωρά. [1-3]

Η αξιοπιστία είναι ένα από τα κύρια πλεονεκτήματα των καμερών ασφαλείας, είτε είναι ενσύρματες ή ασύρματες. Μπορούν να αυξήσουν την ασφάλεια του σπιτιού επιτρέποντάς την επίβλεψη της ιδιοκτησίας τοπικά ή απομακρυσμένα, με ζωντανή ροή του σπιτιού σε smartphone ή υπολογιστή και λήψη άμεσων ειδοποιήσεων για οποιαδήποτε ασυνήθιστη δραστηριότητα.

Οι κάμερες ασφαλείας αναγνωρίζονται ως εξαιρετικός αποτρεπτικός παράγοντας για τους εγκληματίες. Οι ειδικοί δεν συνιστούν να βασίζεται ο ιδιοκτήτης αποκλειστικά σε κάμερες για ασφάλεια, αλλά ένα πλήρες σύστημα οικιακής ασφάλειας είναι η καλύτερη άμυνα. Ωστόσο, οι κάμερες εξακολουθούν να είναι σημαντικές.

Οι κάμερες ασφαλείας του σπιτιού δεν είναι χρήσιμες μόνο για την αστυνόμευση της ιδιοκτησίας. Είναι επίσης χρήσιμες για την παρακολούθηση των παιδιών και των κατοικίδιων.

Τα τελευταία χρόνια παρατηρείται αυξημένη ενοικίαση δωμάτιών μέσω booking.com και Airbnb.com. Ένα από τα κύρια προβλήματα αυτής της χρήσης είναι η ασφάλεια του χώρου. Δεν γνωρίζουμε ποιος είναι ο ιδιοκτήτης ή ποιος βρισκόταν στο δωμάτιο πριν από εμάς ή αν ο προηγούμενος πελάτης είχε βγάλει αντικλείδια ή ξέρει τον ίδιο κωδικό που ανήκει την πόρτα. Έτσι αυξάνεται η ανάγκη παρακολούθησης του χώρου όταν λείπουμε από αυτόν. Πρέπει να τοποθετηθεί ένα γρήγορο σύστημα ασφαλείας για να παρακολουθεί τον χώρο όταν εμείς θα λείπουμε.

Η εργασία αυτή αφορά την υλοποίηση φορητού συστήματος που θα επιβλέπει ένα χώρο με κάμερα και θα ειδοποιεί τον χρήστη για παραβίαση. Για παράδειγμα το σύστημα μπορεί να χρησιμοποιηθεί από ταξιδιώτες που θέλουν να παρακολουθούν το δωμάτιο του ξενοδοχείου που διανέμουν όταν λείπουν από αυτό.

Το σύστημα παρακολούθησης χώρου ανιχνεύει κίνηση σε αυτόν και αποστέλλει ειδοποίηση στον χρήστη μέσω διαδικτύου στο κινητό του. Το κινητό τοποθετείται σε ένα χώρο επίβλεψης με την κάμερα να επιβλέπει το οπτικό της πεδίο. Συνήθως εστιάζει στην είσοδο του δωματίου αλλά θα μπορούσε να είναι και κάποια πόρτα μπαλκονιού ή παράθυρο. Η εφαρμογή στο κινητό παρακολουθεί με βίντεο το

πεδίο και επεξεργάζεται την διαφορά μεταξύ δύο διαδοχικών frames για να ανιχνεύσει κίνηση στον χώρο. Κατά την ανίχνευση κίνησης αποστέλλει την ειδοποίηση σε server.

Η εφαρμογή στο κινητό που τοποθετείται στο δωμάτιο και παρακολουθεί το χώρο, ο server που υλοποιήθηκε με PHP και η βάση MySQL που αποθηκεύονται οι ειδοποιήσεις (εικόνες κτλ) και τέλος η δεύτερη εφαρμογή που έχει ο χρήστης για να παρακολουθεί τις ειδοποιήσεις από τον server όταν βρίσκεται απομακρυσμένα από τον χώρο παρακολούθησης. Η πρώτη εφαρμογή στο κινητό υλοποιήθηκε χρησιμοποιώντας το Flutter ενώ η δεύτερη το MIT App Inventor.

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια μικρή εισαγωγή της εργασίας και οι στόχοι της.

Στο δεύτερο κεφάλαιο παρουσιάζεται η εισαγωγή στο σύστημα παρακολούθησης.

Στο τρίτο κεφάλαιο περιγράφεται η τεχνολογία και τα εργαλεία που χρησιμοποιήθηκαν για να υλοποιηθεί το έργο.

Στο τέταρτο κεφάλαιο αναλύονται οι μονάδες και οι εφαρμογές που χρησιμοποιήθηκαν.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα ενώ στο τέλος της εργασίας παρατίθεται το παράρτημα με κάποιους από τους κώδικες που χρησιμοποιήθηκαν στην εργασία.

Κεφάλαιο 2ο: Το σύστημα παρακολούθησης

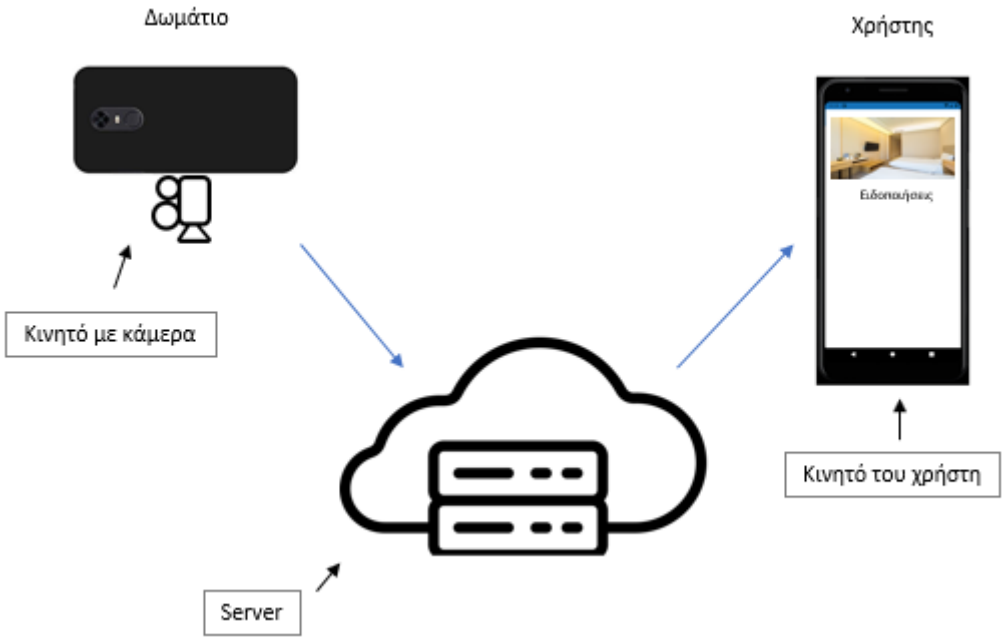
2.1 Εισαγωγή

Υπάρχουν διάφοροι τρόποι για την παρακολούθηση ενός χώρου αλλά λίγοι μπορούν να χρησιμοποιηθούν γρήγορα και εύκολα όταν είμαστε ταξίδι σε ένα ξενοδοχείο ή ενοικιαζόμενο δωμάτιο. Ένας από αυτούς είναι να τοποθετήσουμε μια ip camera συνδεδεμένη στο wifi του χώρου και να την επιβλέπουμε με το κινητό. Αυτό βέβαια προϋποθέτει την προσθήκη στις αποσκευές μας μια ip κάμερα και η ρύθμιση της όταν βρεθούμε στο δωμάτιο.

Οι περισσότεροι άνθρωποι έχουν τουλάχιστον ένα παλιό smartphone που μαζεύει σκόνη σε ένα συρτάρι. Μπορεί να πουληθεί το παλιό τηλέφωνο ή να γίνει ανταλλαγή για ένα κλάσμα της τιμής αγοράς του, αλλά αν εξακολουθεί να είναι ενεργοποιημένο, υπάρχουν πολλές καλές χρήσεις για τα παλιά τηλέφωνα στο σπίτι. Δίνεται η δυνατότητα μετατροπής του παλιού τηλεφώνου Android σε μια οθόνη μωρού ή συσκευή ηχογράφησης ή ραδιόφωνο. Ένας από τους πιο χρήσιμους τρόπους για να επαναχρησιμοποιηθεί ένα παλιό τηλέφωνο είναι η μετατροπή του σε οικιακή κάμερα ασφαλείας. Επίσης θα γίνει εξοικονόμηση χρημάτων αναβαθμίζοντας ένα παλιό τηλέφωνο αντί να αγοραστεί μια νέα κάμερα ασφαλείας στο σπίτι.

Στην εργασία αυτή κατασκευάστηκε ένα σύστημα παρακολούθησης χώρου ανιχνεύοντας κίνηση σε αυτόν και αποστέλλοντας ειδοποίηση στην χρήστη μέσω διαδικτύου στο κινητό του.

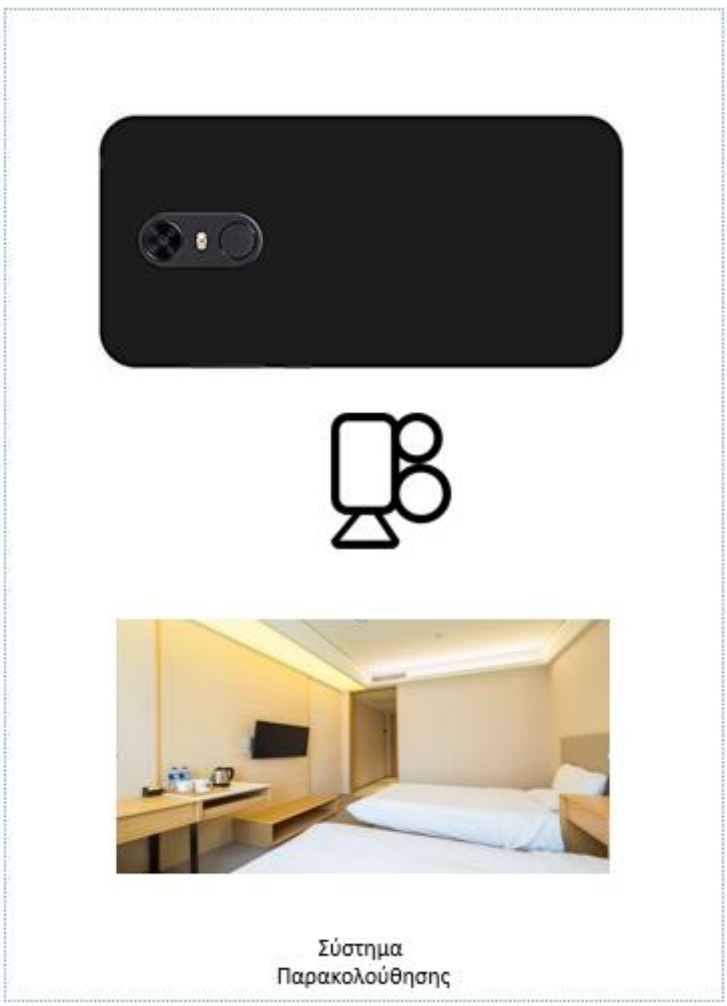
Το σύστημα που υλοποιήθηκε παρουσιάζεται στην Εικόνα 2.1.



Εικόνα 2.1: Σύστημα παρακολούθησης

Το κινητό τοποθετείται στον χώρο επίβλεψης (δωμάτιο) με την κάμερα να ‘κοιτάει’ το πεδίο που είναι για παρακολούθηση. Συνήθως εστιάζει στην είσοδο του δωματίου αλλά θα μπορούσε να είναι και κάποια πόρτα μπαλκονιού ή παράθυρο.

Στην Εικόνα 2.2. παρουσιάζεται ο χώρος παρακολούθησης όπου φαίνεται το πεδίο όρασης της κάμερας.



Εικόνα 2.2: Πεδίο παρακολούθησης από την κάμερα

Η εφαρμογή στο κινητό παρακολουθεί εικόνα-εικόνα το πεδίο και επεξεργάζεται την διαφορά μεταξύ τους για να ανιχνεύσει κίνηση στον χώρο.

Στην Εικόνα 2.23. παρασουσιάζεται ο χώρος παρακολούθησης όπου φαίνεται η παρουσία ανθρώπου και η ανίχνευση κίνησης.

Εικόνα 2.3: Πεδίο και ανίχνευση κίνησης

Βασική προϋπόθεση για την επικοινωνία της εφαρμογής του κινητού με τον χρήστη είναι η σύνδεση του με το διαδίκτυο μέσω του Wifi του ξενοδοχείου ή του δωματίου ή μέσω χρήση δεδομένων.

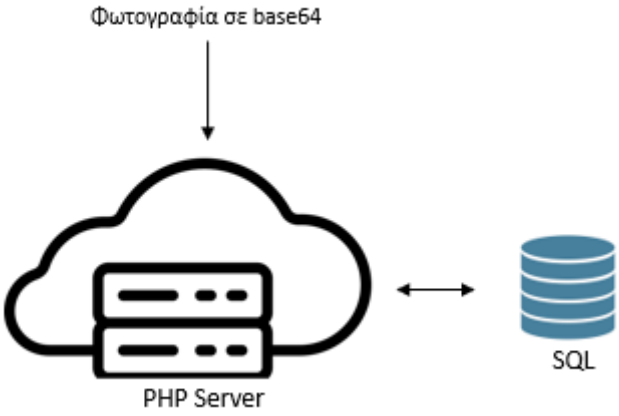
Στην Εικόνα 2.4 παρουσιάζεται η επικοινωνία εφαρμογής με το server. Αφού η εφαρμογή μέσω της κάμερας ανιχνεύσει κίνηση στο πεδίο που εποπτεύει καταγράφει το γεγονός ως ειδοποίηση και αποστέλλει τη φωτογραφία στο server μαζί με την ειδοποίηση.

Στην Εικόνα 2.5 παρουσιάζεται ο τύπος της φωτογραφίας κατά την αποστολή της στο server και ότι η ειδοποίηση και η φωτογραφία αποθηκεύεται σε βάση δεδομένων.

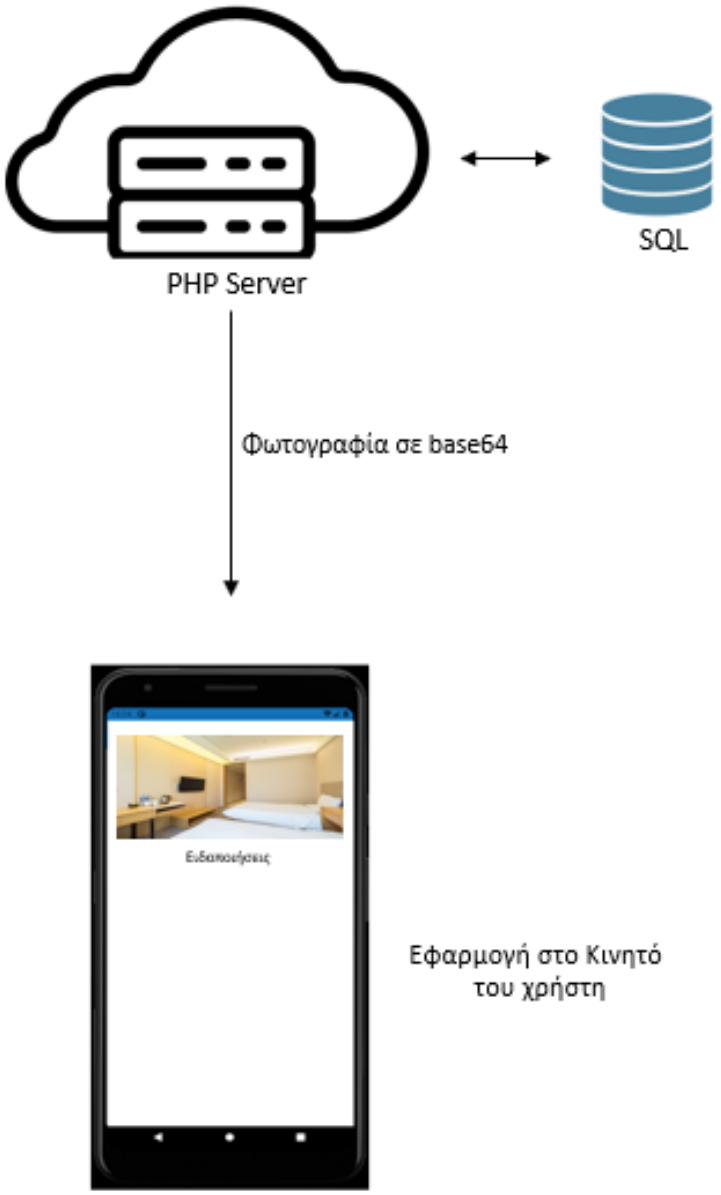
Ο server έχει υλοποιηθεί με PHP και Apache και η βάση δεδομένων είναι MySQL.

Η φωτογραφία μετατρέπεται σε base64 για την αποστολή της φωτογραφίας σαν ένα απλό κείμενο από χαρακτήρες αλλά για την εύκολη αποθήκευση και ανάγνωση της στην/από βάση δεδομένων.

Εικόνα 2.4: Επικοινωνία εφαρμογής με το server



Εικόνα 2.5: Αποστολή φωτογραφίας στο server και αποθήκευση σε βάση δεδομένων



Εικόνα 2.6: Αποστολή φωτογραφίας στο server και αποθήκευση σε βάση δεδομένων

Στην Εικόνα 2.6 παρουσιάζεται η επικοινωνία του server με την τελική εφαρμογή του χρήστη, ο οποίος βρίσκεται συνήθως εκτός χώρου και περιμένει συνδεδεμένος στο διαδίκτυο για να λάβει ειδοποιήσεις όταν ανιχνευτεί κίνηση. Η τελική εφαρμογή θα λάβει τις σχετικές πληροφορίες από το server για τις ειδοποιήσεις και η φωτογραφία θα μεταφερθεί κωδικοποιημένο με base64. Η εφαρμογή θα αποκωδικοποιήσει την εικόνα και θα την προβάλλει στην οθόνη.

Κεφάλαιο 3ο: Εισαγωγή στις τεχνολογίες

Στο κεφάλαιο αυτό θα περιγραφούν τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για να υλοποιηθεί η εργασία.

Οι εφαρμογές στα κινητά υλοποιήθηκαν για Android λειτουργικό σύστημα και ο προγραμματισμός με Flutter και MIT app inventor. Ο server υλοποιήθηκε με PHP και βάση που χρησιμοποιήθηκε είναι η MySQL.

3.1 Android

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα και βασισμένο σε Linux για κινητές συσκευές όπως π.χ smartphone και υπολογιστές tablet. Το Android αναπτύχθηκε από την Open Handset Alliance, υπό την ηγεσία από την Google και άλλες εταιρείες. Το Android προσφέρει μια ενοποιημένη προσέγγιση για την ανάπτυξη εφαρμογών για κινητές συσκευές που σημαίνει ότι οι προγραμματιστές πρέπει να αναπτύσσονται μόνο για το Android και οι εφαρμογές τους θα πρέπει να μπορούν να εκτελούνται διαφορετικές συσκευές που υποστηρίζονται από Android. Η πρώτη έκδοση beta του Android Software Development Kit (SDK) κυκλοφόρησε από την Google το 2007, ενώ η πρώτη εμπορική έκδοση, το Android 1.0, κυκλοφόρησε τον Σεπτέμβριο του 2008. Στις 27 Ιουνίου 2012 η Google ανακοίνωσε την επόμενη έκδοση Android, 4.1 Jelly Bean το οποίο είναι μια σταδιακή ενημέρωση με πρωταρχικό στόχο τη βελτίωση του διεπαφή χρήστη όσον αφορά τη λειτουργικότητα και την απόδοση. Ο πηγαίος κώδικας για Android είναι διαθέσιμος με άδειες λογισμικού ελεύθερου και ανοιχτού κώδικα. Η Google δημοσιεύει το μεγαλύτερο μέρος του κώδικα με την άδεια Apache έκδοση 2.0 και τον πυρήνα Linux σύμφωνα με τη Άδεια GNU. [4]

Το Android είναι ένα ισχυρό λειτουργικό σύστημα που ανταγωνίζεται το Apple iOS και υποστηρίζει εξαιρετικές δυνατότητες.

Λίγες από αυτές αναφέρονται παρακάτω:

► Όμορφο Παρουσιαστικό

Η βασική οθόνη του Android OS παρέχει μια όμορφη και διαισθητική διεπαφή χρήστη.

►Συνδεσιμότητα

GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC και WiMAX.

► Αποθήκευση

Διαθέτει την SQLite, μια ελαφριά σχεσιακή βάση δεδομένων, χρησιμοποιείται για σκοπούς αποθήκευσης δεδομένων.

► Υποστήριξη μέσων ενημέρωσης

H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF και BMP.

► Μηνύματα

SMS και MMS

► Φυλλομετρητής

Βασίζεται στη μηχανή διάταξης WebKit ανοιχτού κώδικα, σε συνδυασμό με τη μηχανή JavaScript V8 του Chrome που υποστηρίζει HTML5 και CSS3.

► Πολλαπλής αφής

Το Android έχει εγγενή υποστήριξη για multi-touch, η οποία αρχικά ήταν διαθέσιμη σε συσκευές.

► Multi-tasking

Ο χρήστης μπορεί να μεταπηδήσει από τη μια εργασία στην άλλη και ταυτόχρονα διάφορες εφαρμογές μπορούν να εκτελεστούν ταυτόχρονα.

► Γραφικά στοιχεία με δυνατότητα αλλαγής μεγέθους

Τα γραφικά στοιχεία έχουν δυνατότητα αλλαγής μεγέθους, επομένως οι χρήστες μπορούν να τα επεκτείνουν για να εμφανίσουν περισσότερο περιεχόμενο ή να τα συρρικνώσουν για εξοικονόμηση χώρου.

► Wi-Fi Direct

Μια τεχνολογία που επιτρέπει στις εφαρμογές να ανακαλύπτουν και να ζευγαρώνουν απευθείας, μέσω μιας σύνδεσης peer-to-peer υψηλού εύρους ζώνης.

► Πολυγλωσσία

Υποστηρίζει κείμενο μονής κατεύθυνσης και αμφίδρομης κατεύθυνσης.

Διεπαφή

Η προεπιλεγμένη διεπαφή χρήστη του Android βασίζεται κυρίως σε άμεσο χειρισμό, χρησιμοποιώντας εισόδους αφής που αντιστοιχούν σε πραγματικές ενέργειες, όπως σάρωση, άγγιγμα και αντίστροφο άγγιγμα για χειρισμό αντικειμένων στην οθόνη, μαζί με ένα εικονικό πληκτρολόγιο. Τα χειριστήρια παιχνιδιών και τα φυσικά πληκτρολόγια πλήρους μεγέθους υποστηρίζονται μέσω Bluetooth ή USB. Η απόκριση στην είσοδο του χρήστη έχει σχεδιαστεί για να είναι άμεση και παρέχει μια ρευστή διεπαφή αφής, χρησιμοποιώντας συχνά τις δυνατότητες δόνησης της συσκευής για την παροχή απτικής ανάδρασης στον χρήστη. Το εσωτερικό υλικό, όπως επιταχυνσιόμετρα, γυροσκόπια και αισθητήρες εγγύτητας χρησιμοποιούνται από ορισμένες εφαρμογές για να ανταποκρίνονται σε πρόσθετες ενέργειες του χρήστη όπως για παράδειγμα προσαρμόζοντας την οθόνη από κατακόρυφο σε οριζόντιο, ανάλογα με τον προσανατολισμό της συσκευής.

Αρχική οθόνη

Οι συσκευές Android εκκινούνται στην αρχική οθόνη, τον κύριο «κόμβο» πλοήγησης και πληροφοριών στις συσκευές Android, ανάλογο με τον επιτραπέζιο υπολογιστή που βρίσκεται σε προσωπικούς υπολογιστές. Οι αρχικές οθόνες Android αποτελούνται συνήθως από εικονίδια εφαρμογών και γραφικά στοιχεία. Τα εικονίδια εφαρμογών εκκινούν τη συσχετισμένη εφαρμογή, ενώ τα γραφικά στοιχεία εμφανίζουν ζωντανά περιεχόμενο που ενημερώνεται αυτόματα, όπως μια πρόγνωση καιρού, τα εισερχόμενα email του χρήστη ή μια ετικέτα ειδήσεων απευθείας στην αρχική οθόνη. Μια αρχική οθόνη μπορεί να αποτελείται από πολλές σελίδες, μεταξύ των οποίων ο χρήστης μπορεί να κάνει σάρωση εμπρός και πίσω. Οι εφαρμογές τρίτων που είναι διαθέσιμες στο Google Play και σε άλλα καταστήματα εφαρμογών μπορούν να επαναπροσδιορίσουν το θέμα της αρχικής οθόνης ακόμη και να μιμηθούν την εμφάνιση άλλων λειτουργικών συστημάτων. Οι περισσότεροι κατασκευαστές προσαρμόζουν την εμφάνιση και τις δυνατότητες των συσκευών τους Android για να διαφοροποιηθούν από τους ανταγωνιστές τους.

Γραμμή κατάστασης

Στο επάνω μέρος της οθόνης υπάρχει μια γραμμή κατάστασης, η οποία εμφανίζει πληροφορίες σχετικά με τη συσκευή και τη σύνδεσή της. Αυτή η γραμμή κατάστασης μπορεί να τραβηχτεί (σύρεται) προς τα κάτω για να αποκαλυφθεί μια οθόνη ειδοποιήσεων όπου οι εφαρμογές εμφανίζουν σημαντικές πληροφορίες ή ενημερώσεις, καθώς και γρήγορη πρόσβαση στα χειριστήρια του συστήματος και τις εναλλαγές, όπως φωτεινότητα οθόνης, ρυθμίσεις συνδεσιμότητας (WiFi, Bluetooth, δεδομένα κινητής τηλεφωνίας). λειτουργία ήχου και φακός. Οι προμηθευτές μπορούν να εφαρμόσουν εκτεταμένες ρυθμίσεις, όπως η δυνατότητα προσαρμογής της φωτεινότητας του φακού.

Ειδοποιήσεις

Οι ειδοποιήσεις είναι σύντομες, έγκαιρες και περιέχουν σχετικές πληροφορίες σχετικά με την εφαρμογή όταν δεν χρησιμοποιείται. Όταν πατηθούν οι χρήστες οδηγούνται σε μια οθόνη εντός της εφαρμογής που σχετίζεται με την ειδοποίηση. Ξεκινώντας με εκδόσεις άνω του Android 4.1 οι επεκτάσιμες ειδοποιήσεις επιτρέπουν στο χρήστη να πατήσει ένα εικονίδιο στην ειδοποίηση για να επεκταθεί και να εμφανίσει περισσότερες πληροφορίες και πιθανές ενέργειες εφαρμογής απευθείας από την ειδοποίηση.

Λίστες εφαρμογών

Οι εφαρμογές Android συνήθως αναπτύσσονται στη γλώσσα Java χρησιμοποιώντας το Λογισμικό SDK. Μόλις αναπτυχθούν, οι εφαρμογές Android μπορούν να τοποθετηθούν εύκολα και να ‘κατέβουν’ και εγκαταστηθούν μέσω από ένα κατάστημα όπως το Google Play Store. Το Android τροφοδοτεί εκατοντάδες εκατομμύρια κινητές συσκευές σε περισσότερες από 200 χώρες σε όλο τον κόσμο κόσμος. Είναι η μεγαλύτερη εγκατεστημένη βάση οποιασδήποτε κινητής πλατφόρμας και αναπτύσσεται γρήγορα.

Μια οθόνη "Όλες οι εφαρμογές" εμφανίζει όλες τις εγκατεστημένες εφαρμογές, με τη δυνατότητα για τους χρήστες να σύρουν μια εφαρμογή από τη λίστα στην αρχική οθόνη. Η πρόσβαση στη λίστα εφαρμογών μπορεί να γίνει χρησιμοποιώντας μια κίνηση ή ένα κουμπί, ανάλογα με την έκδοση Android. Μια οθόνη "Πρόσφατα", γνωστή και ως "Επισκόπηση", επιτρέπει στους χρήστες να αλλάζουν μεταξύ εφαρμογών που χρησιμοποιήθηκαν πρόσφατα. Η πρόσφατη λίστα μπορεί να εμφανίζεται δίπλα-δίπλα ή να επικαλύπτεται, ανάλογα με την έκδοση Android και τον κατασκευαστή.

Κουμπιά πλοήγησης

Πολλά πρώιμα smartphone με λειτουργικό Android ήταν εξοπλισμένα με ένα αποκλειστικό κουμπί αναζήτησης για γρήγορη πρόσβαση σε μια μηχανή αναζήτησης ιστού και τη δυνατότητα εσωτερικής αναζήτησης μεμονωμένων εφαρμογών. Οι πιο πρόσφατες συσκευές επιτρέπουν συνήθως την πρώτη με παρατεταμένο πάτημα ή σάρωση μακριά από το κουμπί αρχικής οθόνης. Το αποκλειστικό κλειδί επιλογής, γνωστό και ως κλειδί μενού, και η προσομοίωσή του στην οθόνη, δεν υποστηρίζεται πλέον από την έκδοση Android 10. Η Google συνιστά στους προγραμματιστές εφαρμογών για κινητά να εντοπίζουν τα μενού μέσα στη διεπαφή χρήστη. Σε πιο πρόσφατα τηλέφωνα, η θέση του καταλαμβάνεται από ένα πλήκτρο εργασίας που χρησιμοποιείται για πρόσβαση στη λίστα των εφαρμογών που χρησιμοποιήθηκαν πρόσφατα όταν ενεργοποιείται. Ανάλογα με τη συσκευή, το παρατεταμένο πάτημά της μπορεί να προσομοιώνει το πάτημα του κουμπιού μενού ή να ενεργοποιεί την προβολή διαχωρισμένης οθόνης, η τελευταία από τις οποίες είναι η προεπιλεγμένη συμπεριφορά.

Προβολή διαιρεμένης οθόνης

Η υποστήριξη για προβολή διαίρεσης οθόνης έχει προστεθεί από την έκδοση Android 7.0. Τα πρώτα προσαρμοσμένα smartphone που βασίζονται σε Android που είναι γνωστό ότι διέθεταν λειτουργία προβολής διαχωρισμένης οθόνης είναι τα Samsung Galaxy S3 και Note 2.

Φόρτιση ενώ είναι απενεργοποιημένο

Όταν συνδέεται ή αποσυνδέεται η τροφοδοσία φόρτισης και όταν ενεργοποιείται σύντομα το κουμπί λειτουργίας ή το κουμπί αρχικής οθόνης, ενώ η συσκευή είναι απενεργοποιημένη, εμφανίζεται στην οθόνη ένας οπτικός μετρητής μπαταρίας του οποίου η εμφάνιση ποικίλλει μεταξύ των προμηθευτών, επιτρέποντας στον χρήστη να εκτιμήσει γρήγορα την κατάσταση φόρτισης.

Απτικό εφέ με ηχοσύζευξη

Στις αρχές του 2021 κυκλοφόρησε μια έκδοση Android που υποστηρίζει τη σύγχρονη δόνηση που μπορεί να ρυθμιστεί για να συμπληρώνει τον ήχο.

Εφαρμογές

Πολλές, σχεδόν όλες, συσκευές Android διαθέτουν προεγκατεστημένες εφαρμογές Google, όπως το Gmail, οι Χάρτες Google, το Google Chrome, το YouTube, η Μουσική Google Play, οι Ταινίες & TV Google Play και πολλά άλλα.

Ανάπτυξη λογισμικού Android και Google Play

Οι εφαρμογές οι οποίες επεκτείνουν τη λειτουργικότητα των συσκευών Android έχουν γραφτεί χρησιμοποιώντας το κιτ ανάπτυξης λογισμικού Android (SDK). Τελευταία χρησημοποιείται η γλώσσα προγραμματισμού Kotlin, η οποία αντικατέστησε την Java ως η προτιμώμενη γλώσσα της Google για την ανάπτυξη εφαρμογών Android. Η Java εξακολουθεί να υποστηρίζεται αφού ήταν αρχικά η μόνη επιλογή για προγράμματα και συχνά αναμιγνύεται με το Kotlin. Η Java ή άλλες γλώσσες JVM, όπως η Kotlin, μπορούν να συνδυαστούν με C/C++. Η γλώσσα προγραμματισμού Go υποστηρίζεται επίσης, αν και με περιορισμένο σύνολο διεπαφών προγραμματισμού εφαρμογών (API).

Το SDK περιλαμβάνει ένα ολοκληρωμένο σύνολο εργαλείων ανάπτυξης και περιλαμβάνει ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες λογισμικού, έναν εξομοιωτή συσκευής, τεκμηρίωση, δείγμα κώδικα και εκπαιδευτικά προγράμματα. Αρχικά, το υποστηριζόμενο περιβάλλον ολοκληρωμένης ανάπτυξης (IDE) της Google ήταν το Eclipse χρησιμοποιώντας την προσθήκη Εργαλείων ανάπτυξης Android (ADT). Το 2014 η Google κυκλοφόρησε το Android Studio, βασισμένο στο IntelliJ IDEA, ως το κύριο IDE για την ανάπτυξη εφαρμογών Android. Υπάρχουν και άλλα εργαλεία ανάπτυξης, όπως ένα κιτ native ανάπτυξης (NDK) για εφαρμογές ή επεκτάσεις σε C ή C++, το Google App Inventor, ένα οπτικό περιβάλλον για αρχάριους προγραμματιστές και διάφορα πλαίσια εφαρμογών ιστού για κινητές συσκευές πολλαπλών πλατφορμών. Το 2014 η Google αποκάλυψε ένα εργαλείο βασισμένο στο Apache Cordova για τη μεταφορά εφαρμογών ιστού Chrome HTML 5 στο Android.

Το Android διαθέτει πολλές εφαρμογές τρίτων, τις οποίες μπορούν να αποκτήσουν οι χρήστες με λήψη και εγκατάσταση του αρχείου APK (πακέτο εφαρμογής Android) της εφαρμογής ή με λήψη τους χρησιμοποιώντας ένα πρόγραμμα αποθήκευσης εφαρμογών που επιτρέπει στους χρήστες να εγκαταστήσουν, να ενημερώσουν και να αφαιρέσουν εφαρμογές από τις συσκευές τους. Το Google Play Store είναι το κύριο κατάστημα εφαρμογών που είναι εγκατεστημένο σε συσκευές Android. Το Google Play Store επιτρέπει στους χρήστες να περιηγούνται, να κατεβάζουν και να ενημερώνουν εφαρμογές που δημοσιεύονται από την Google και τρίτους προγραμματιστές. Από το 2021, υπάρχουν περισσότερες από τρία εκατομμύρια εφαρμογές διαθέσιμες για Android στο Play Store. Ορισμένες εταιρείες κινητής τηλεφωνίας προσφέρουν απευθείας χρέωση μέσω κινητού τηλεφώνου για αγορές εφαρμογών Google Play, όπου το κόστος της εφαρμογής προστίθεται στον μηνιαίο λογαριασμό του χρήστη. Από το 2017, υπάρχουν πάνω από ένα δισεκατομμύριο ενεργοί χρήστες το μήνα για το Gmail, το Android, το Chrome, το Google Play και τους Χάρτες.

Λόγω του ανοιχτής κώδικα του Android, υπάρχουν επίσης διάφορες αγορές εφαρμογών τρίτων για το Android, είτε για να παρέχουν ένα υποκατάστατο για συσκευές που δεν επιτρέπεται να αποστέλλονται

με το Google Play Store, είτε για να παρέχουν εφαρμογές που δεν μπορούν να προσφερθούν στο Google Play Store λόγω παραβιάσεων πολιτικής ή για άλλους λόγους. Παραδείγματα αυτών των καταστημάτων τρίτων περιλαμβάνουν το Amazon Appstore και το SlideMe.

Το 2020, η Google αφαίρεσε αρκετές εφαρμογές Android από το Play Store, καθώς διαπιστώθηκε ότι παραβιάζουν τους κανόνες συλλογής δεδομένων της.

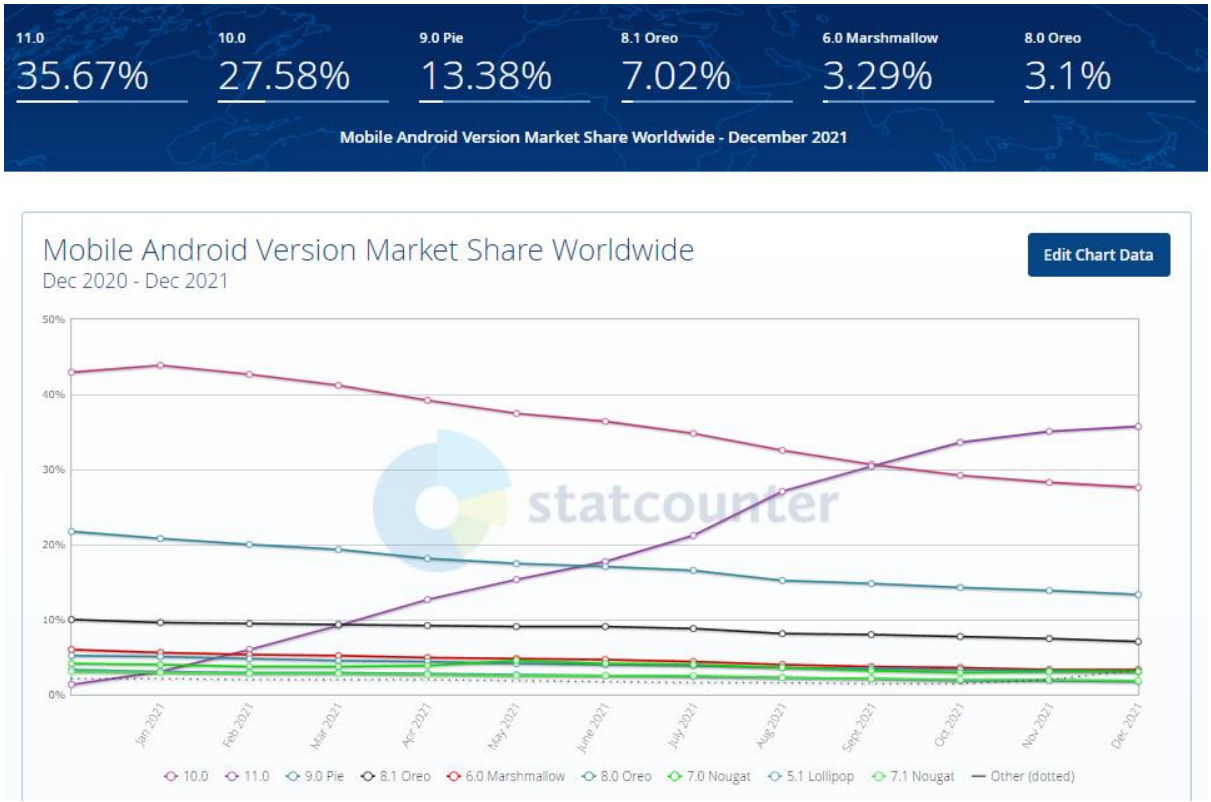
Πυρήνας Linux

Ο πυρήνας του Android βασίζεται στους κλάδους μακροπρόθεσμης υποστήριξης (LTS) του πυρήνα Linux. Από το 2021, το Android χρησιμοποιεί τις εκδόσεις 4.14 ή 5.4 του πυρήνα Linux.

Η παραλλαγή του πυρήνα Linux του Android έχει επιπλέον αρχιτεκτονικές με αλλαγές που εφαρμόζονται από την Google εκτός του τυπικού κύκλου ανάπτυξης πυρήνα Linux, όπως η συμπερίληψη στοιχείων όπως δέντρα συσκευών. Ορισμένες λειτουργίες πρόσθεσε η Google στον πυρήνα του Linux, ιδίως μια δυνατότητα διαχείρισης ενέργειας.

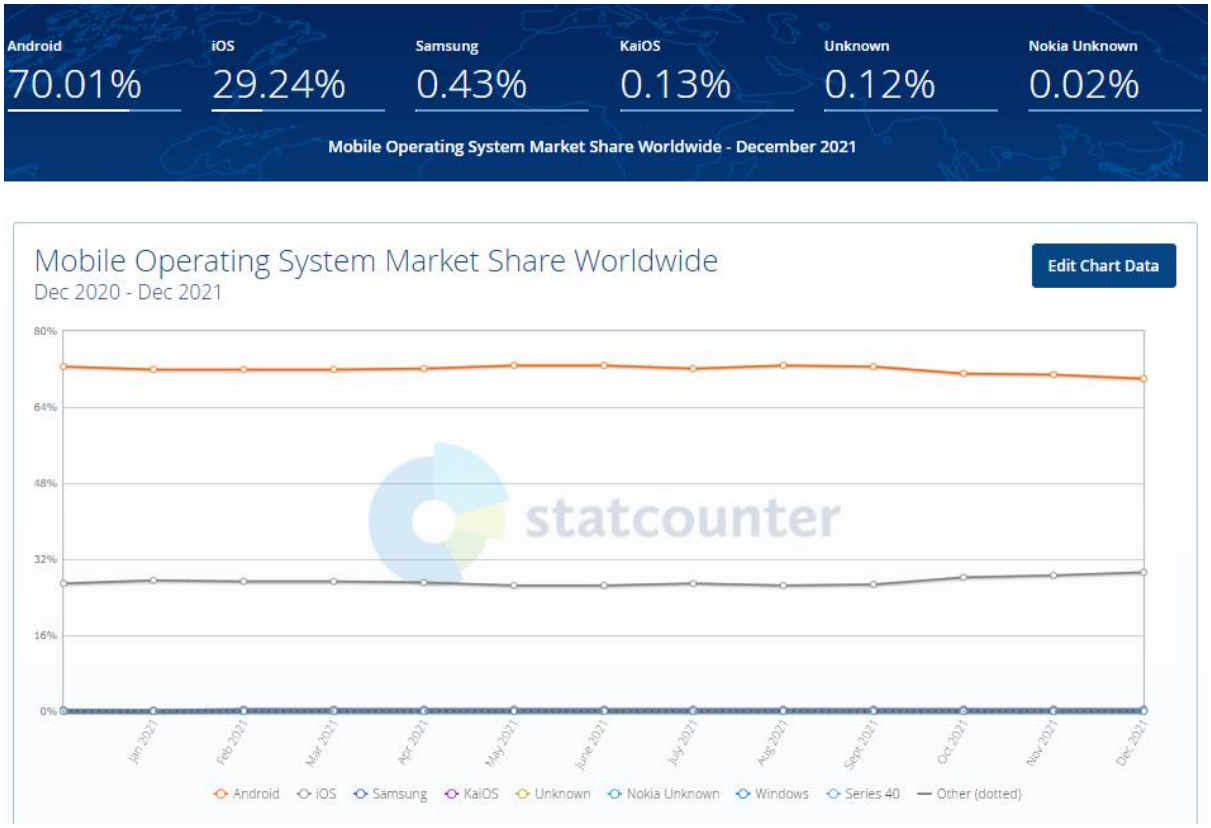
Το Android είναι μια διανομή Linux σύμφωνα με το Ίδρυμα Linux και τον επικεφαλής ανοιχτού κώδικα της Google, Chris DiBona. Άλλοι λένε ότι το Android δεν είναι Linux με την παραδοσιακή έννοια διανομής Linux που μοιάζει με Unix. Το Android δεν περιλαμβάνει τη Βιβλιοθήκη GNU C και ορισμένα άλλα στοιχεία που βρίσκονται συνήθως σε διανομές Linux.

Με την κυκλοφορία του Android Oreo το 2017, η Google άρχισε να απαιτεί οι συσκευές που κατασκευάζονται με νέα SoC να διαθέτουν έκδοση πυρήνα Linux πάνω από 4.4 για λόγους ασφαλείας.



Εικόνα 3.1: Μερίδιο αγοράς για διάφορες εκδόσεις Android[5]

Στη Εικόνα 3.1 παρουσιάζεται η σύγκριση μεταξύ των εκδόσεων Android παγκοσμίως για το 2021. Φαίνεται ότι επικρατεί η έκδοση 11 και να ακολουθεί από πολύ κοντά η έκδοση 10.

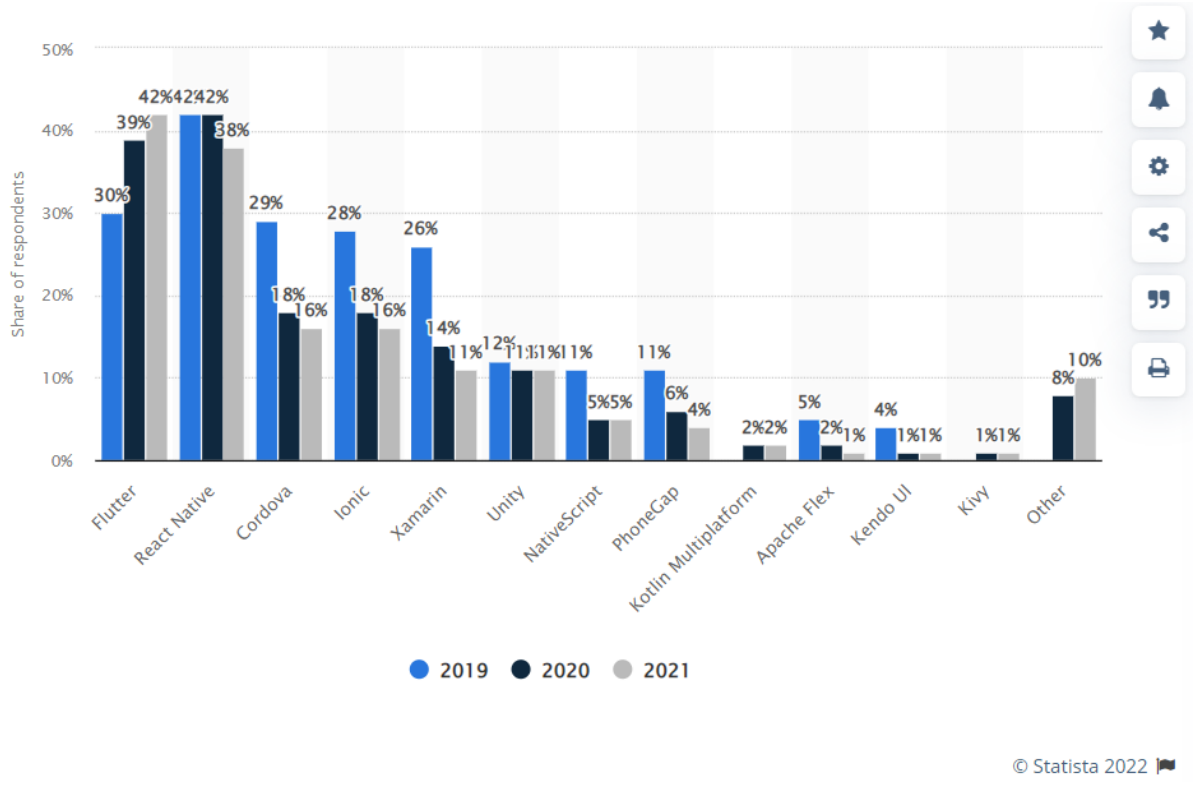


Εικόνα 3.2: Μερίδιο αγοράς για διάφορα λειτουργικά συστήματα και κινητά[6]

Στην Εικόνα 3.2 παρουσιάζεται η σύγκριση για το μερίδιο αγοράς για διάφορα λειτουργικά συστήματα και κινητά για το 2021. Φαίνεται ότι επικρατεί το Android με 70%.

3.2 Flutter

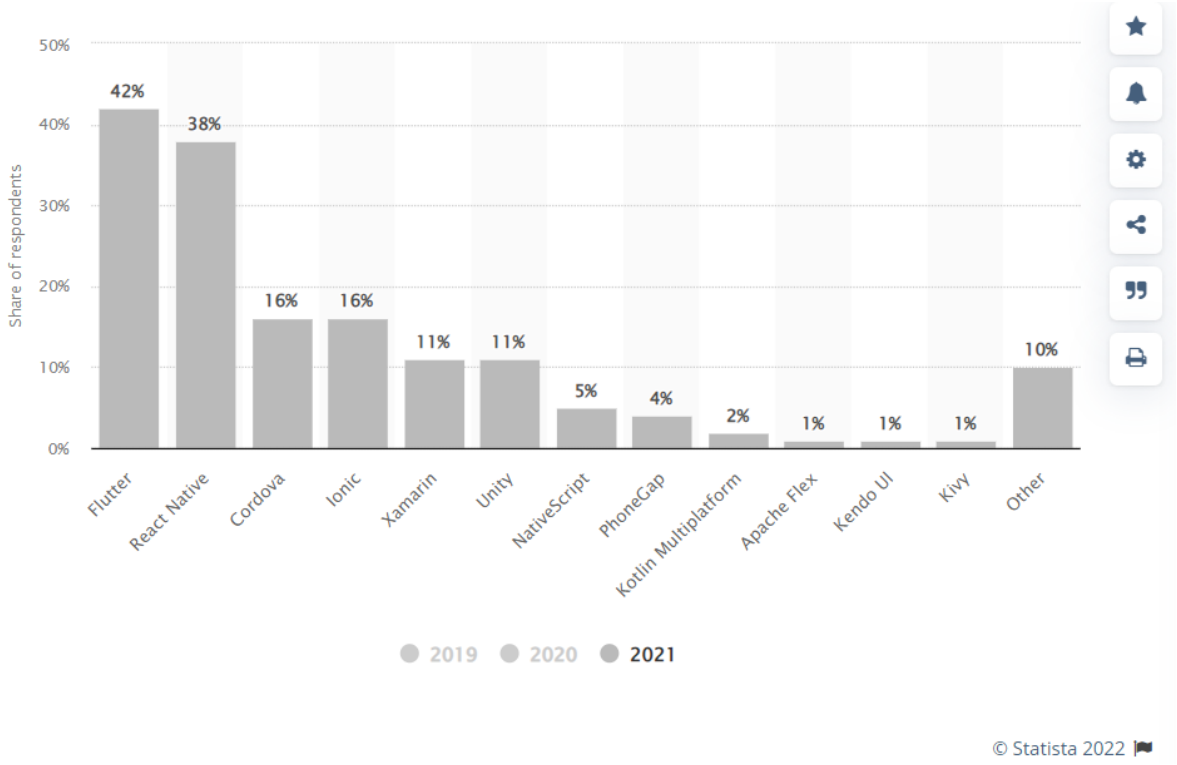
Με την ταχεία άνοδο των πλαισίων για φορητές συσκευές πολλαπλών πλατφορμών όπως το Ionic, το React Native και Xamarin, η Google αποφάσισε να μπει στο παιχνίδι και να αναπτύξει το δικό της πλαίσιο με υποστήριξη τόσο για Android όσο και για iOS με χρήση της ίδιας βάσης κώδικα. Έτσι κατέληξε στο Flutter ρο οποίο είναι ένα SDK ανάπτυξης εφαρμογών για κινητά ανοιχτού κώδικα που αναπτύχθηκε κυρίως και χορηγείται από την Google. Χρησιμοποιείται για την ανάπτυξη εφαρμογών για Android και iOS. Το Flutter είναι γραμμένο σε C, C++ και Dart και χρησιμοποιεί τη μηχανή γραφικών Skia. Προσφέρει ένα πλούσιο σύνολο από πλήρως προσαρμόσιμα γραφικά στοιχεία για τη δημιουργία εγγενών διεπαφών, συμπεριλαμβανομένου του όμορφου σχεδιασμού υλικού βιβλιοθήκη και γραφικά στοιχεία Cupertino (iOS), πλούσια API κίνησης, ομαλή φυσική κύλιση, ευαισθητοποίηση πλατφόρμας και γρήγορη επαναφόρτωση που βοηθά στη γρήγορη δημιουργία διεπαφής χρήστη χωρίς να χάνεται η κατάσταση. Διαθέτει εξομοιωτές και προσομοιωτές και οποιοδήποτε υλικό για iOS και Android. Όλα αυτά τα εξαιρετικά χαρακτηριστικά έχουν βοηθήσει το Flutter να απογειωθεί πολύ γρήγορα και οι προγραμματιστές άρχισαν να το εμπιστεύονται. Με το Flutter να κερδίζει δυναμική, φαίνεται απίθανο να εξαφανιστεί σύντομα. [7]



Εικόνα 3.3: Πλατφόρμες προγραμματισμού για φορητές συσκευές πολλαπλών πλατφορμών που χρησιμοποιούνται από προγραμματιστές λογισμικού σε όλο τον κόσμο 2019-2021 [8]

Στην Εικόνα 3.3 παρουσιάζεται η ποσοστιαία σύγκριση των IDE προγραμματισμού για φορητές συσκευές πολλαπλών πλατφορμών που χρησιμοποιούνται από προγραμματιστές λογισμικού σε όλο τον κόσμο 2019-2021.

Φαίνεται ότι το 2021 το Flutter ξεπερνάει το React Native και έχει ανοδική πορεία.



Εικόνα 3.4: Πλατφόρμες προγραμματισμού για φορητές συσκευές πολλαπλών πλατφορμών που χρησιμοποιούνται από προγραμματιστές λογισμικού σε όλο τον κόσμο 2021 [8]

Στην Εικόνα 3.4 φαίνεται ξεκάθαρα ότι το 2021 το Flutter ξεπερνάει το React Native με ποσοστό 42%.

Γενικά, η ανάπτυξη μιας εφαρμογής για κινητά είναι μια πολύπλοκη και προκλητική εργασία. Υπάρχουν πολλά πλαίσια διαθέσιμα για την ανάπτυξη μιας εφαρμογής για κινητά. Το Android παρέχει native πλαίσιο που βασίζεται σε γλώσσα Java και το iOS παρέχει ένα native πλαίσιο που βασίζεται σε Objective-C / Shift γλώσσα. Ωστόσο, για να αναπτύξουμε μια εφαρμογή που να υποστηρίζει και τα δύο λειτουργικά συστήματα, πρέπει να κωδικοποιήσουμε σε δύο διαφορετικές γλώσσες χρησιμοποιώντας δύο διαφορετικούς κώδικες. Για να ξεπεραστεί αυτή η πολυπλοκότητα, υπάρχουν κώδικες για κινητά που υποστηρίζουν και τα δύο λειτουργικά συστήματα. Αυτοί οι κώδικες ποικίλλουν από κώδικες υβριδικών εφαρμογών για κινητά που βασίζεται σε HTML μέχρι ένα σε σύνθετο πλαίσιο με κώδικες συγκεκριμένης γλώσσας. Ανεξάρτητα από την απλότητά τους ή πολυπλοκότητα, αυτά τα πλαίσια έχουν πάντα πολλά μειονεκτήματα, ένα από τα κύρια μειονέκτημα είναι η αργή τους απόδοση. Το Flutter είναι ένα απλό και υψηλής απόδοσης πλαίσιο βασισμένο στο Dart γλώσσα, παρέχει υψηλή απόδοση αποδίδοντας τη διεπαφή χρήστη απευθείας στο λειτουργικό σύστημα και όχι μέσω native κώδικα. Το Flutter προσφέρει επίσης πολλά έτοιμα προς χρήση widget (UI) για τη δημιουργία μιας σύγχρονης εφαρμογής. Αυτά τα γραφικά στοιχεία είναι βελτιστοποιημένα για περιβάλλον κινητών και για το σχεδιασμό της εφαρμογής χρησιμοποιώντας γραφικά στοιχεία είναι τόσο απλά όσο ο σχεδιασμός

HTML. Η εφαρμογή Flutter είναι η ίδια ένα widget. Τα γραφικά στοιχεία Flutter υποστηρίζουν επίσης κινούμενα σχέδια. Η λογική της εφαρμογής βασίζεται στον αντιδραστικό προγραμματισμό. Αλλάζοντας την κατάσταση του widget, το Flutter θα αλλάζει αυτόματα (αντιδραστικός προγραμματισμός) συγκρίνοντας την κατάσταση του γραφικού στοιχείου (παλιά και νέα) και θα αποδίδει στο γραφικό στοιχείο τις απαραίτητες αλλαγές αντί να αποδώσει ξανά ολόκληρο το γραφικό στοιχείο.

Μερικά από τα χαρακτηριστικά του Flutter είναι:

 Γρήγορη ανάπτυξη.

 Σύγχρονο και αντιδραστικό πλαίσιο.

 Χρησιμοποιεί τη γλώσσα προγραμματισμού Dart και είναι πολύ εύκολη στην εκμάθησή της.

 Όμορφες διεπαφές χρήστη.

 Μεγάλος κατάλογος widget.

 Εκτελεί την ίδια διεπαφή χρήστη για πολλές πλατφόρμες.

 Εφαρμογή υψηλής απόδοσης.

Το Flutter συνοδεύεται από όμορφα και προσαρμόσιμα widget για υψηλή απόδοση και εξαιρετική εφαρμογή για κινητά. Το Flutter προσφέρει πολλά πλεονεκτήματα όπως αναφέρονται παρακάτω:

 Το Dart διαθέτει ένα μεγάλο αποθετήριο πακέτων λογισμικού που επιτρέπει την επέκταση των δυνατοτήτων της εφαρμογής.

 Οι προγραμματιστές πρέπει να γράψουν μόνο μια ενιαία βάση κώδικα και για τις δύο εφαρμογές (και τα δύο Android και πλατφόρμες iOS).

 Το Flutter μπορεί να επεκταθεί και σε άλλη πλατφόρμα στο μέλλον.

 Το Flutter χρειάζεται λιγότερο έλεγχο. Λόγω της μοναδικής βάσης του κώδικα, αρκεί να γραφούν αυτοματοποιημένες δοκιμές μία φορά και για τις δύο πλατφόρμες.

 Η απλότητα του Flutter το καθιστά καλό εργαλείο για γρήγορη ανάπτυξη. Η προσαρμογή του η ικανότητα και η επεκτασιμότητα το καθιστούν ακόμα πιο ισχυρό.

 Με το Flutter, οι προγραμματιστές έχουν τον πλήρη έλεγχο των γραφικών στοιχείων και της διάταξής τους.

 Το Flutter προσφέρει εξαιρετικά εργαλεία προγραμματιστών με ένα γρήγορο hot reload.

Μειονεκτήματα του Flutter

 Εφόσον είναι κωδικοποιημένο σε γλώσσα Dart, ένας προγραμματιστής πρέπει να μάθει νέα γλώσσα

 Το σύγχρονο πλαίσιο προσπαθεί να διαχωρίσει τη λογική και τη διεπαφή χρήστη όσο το δυνατόν περισσότερο, αλλά στο Flutter, η διεπαφή χρήστη και η λογική αναμειγνύονται.

 Το Flutter είναι ένα ακόμη πλαίσιο για τη δημιουργία εφαρμογών για κινητά. Οι προγραμματιστές έχουν μια δύσκολη επιλογή σωστών εργαλείων ανάπτυξης.

Εγκατάσταση – Μικρός οδηγός

Εγκατάσταση στα Windows:

Βήμα 1: Μετάβαση στη διεύθυνση URL, https://flutter.dev/docs/get-started/install/windows και ε λήψη του πιο πρόσφατου Flutter SDK.

Βήμα 2: Αποσυμπιέστε το αρχείο zip σε έναν φάκελο, όπως C:\flutter\

Βήμα 3: Ενημερώστε τη διαδρομή συστήματος για να συμπεριλάβει τον κατάλογο flutter bin.

Βήμα 4: Το Flutter παρέχει ένα εργαλείο, το flutter γιατρό για να ελέγξει ότι όλες οι απαιτήσεις του flutter

flutter doctor

Βήμα 5: Η εκτέλεση της παραπάνω εντολής θα αναλύσει το σύστημα και θα εμφανίσει την αναφορά του όπως φαίνεται παρακάτω:

[√] Flutter (σταθερό κανάλι, v1.2.1, σε Microsoft Windows [Έκδοση

10.0.17134.706], locale en-US)

[√] Android toolchain - ανάπτυξη για συσκευές Android (έκδοση Android SDK

28.0.3)

[√] Android Studio (έκδοση 3.2)

[√] VS Code, έκδοση 64-bit (έκδοση 1.29.1)

[!] Συνδεδεμένη συσκευή

! Δεν υπάρχουν διαθέσιμες συσκευές

Η αναφορά λέει ότι όλα τα εργαλεία ανάπτυξης είναι διαθέσιμα, αλλά η συσκευή δεν είναι συνδεδεμένη.

Μπορούμε να το διορθώσουμε συνδέοντας μια συσκευή Android μέσω USB ή ξεκινώντας ένα Android εξομοιωτή.

Βήμα 6: Εγκαταστήστε το πιο πρόσφατο Android SDK, εάν το αναφέρει ο γιατρός του flutter

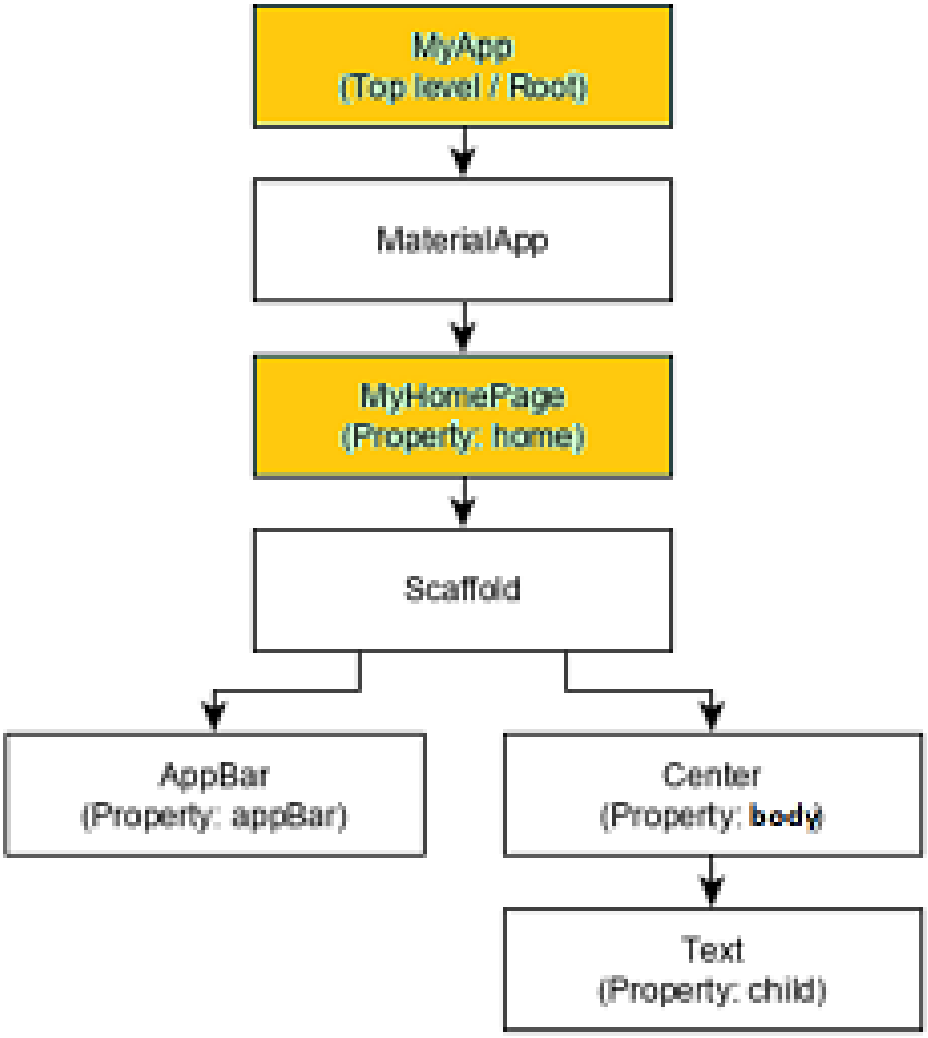
Βήμα 7: Εγκαταστήστε το πιο πρόσφατο Android Studio, εάν το αναφέρει ο γιατρός του flutter

Βήμα 8: Ξεκινήστε έναν εξομοιωτή Android ή συνδέστε μια πραγματική συσκευή Android στο σύστημα.

Βήμα 9: Εγκαταστήστε την προσθήκη Flutter and Dart για το Android Studio. Παρέχει πρότυπο εκκίνησης σε δημιουργήστε νέα εφαρμογή Flutter, μια επιλογή εκτέλεσης και εντοπισμού σφαλμάτων της εφαρμογής Flutter στο Android το ίδιο το στούντιο.

Widgets

Η βασική ιδέα του πλαισίου Flutter είναι το In Flutter, Τα πάντα είναι widget. Τα Widgets είναι βασικά στοιχεία διεπαφής χρήστη που χρησιμοποιούνται για τη δημιουργία της διεπαφής χρήστη της εφαρμογής. Στο Flutter, η εφαρμογή είναι η ίδια ένα widget. Η εφαρμογή είναι το widget ανώτατου επιπέδου και αυτό το UI δημιουργείται χρησιμοποιώντας ένα ή περισσότερα γραφικά στοιχεία τα οποία πάλι δημιουργούνται χρησιμοποιώντας τα παιδιά του widgets. Αυτή η δυνατότητα σύνθεσης βοηθά τη δημιουργία μιας διεπαφής χρήστη οποιασδήποτε πολυπλοκότητας. Για παράδειγμα, η ιεραρχία γραφικών στοιχείων μια απλής εφαρμογής προσδιορίζεται στο ακόλουθο διάγραμμα:



Εικόνα 3.5: Ιεραρχία γραφικών στοιχείων μια απλής εφαρμογής στο Flutter

MyApp είναι το γραφικό στοιχείο που δημιουργήθηκε από τον χρήστη και έχει δημιουργηθεί χρησιμοποιώντας το εγγενές γραφικό στοιχείο Flutter.

 MaterialApp έχει μια αρχική ιδιότητα για να καθορίσει τη διεπαφή χρήστη της αρχικής σελίδας, που είναι και πάλι ένα γραφικό στοιχείο που δημιουργήθηκε από τον χρήστη.

 MyHomePage δημιουργείται χρησιμοποιώντας ένα άλλο εγγενές widget flutter, το Scaffold.

 Scaffold έχει δύο ιδιότητες – body και appBar.

 body χρησιμοποιείται για τον καθορισμό της κύριας διεπαφής χρήστη και το appBar για τον καθορισμό της κεφαλίδας της διεπαφής του χρήστη.

 Header UI χτίζεται με χρήση γραφικού στοιχείου flutter, το AppBar και το Body UI δημιουργείται με χρήση Center widget.

 The Center widget έχει μια ιδιότητα, Child, η οποία αναφέρεται στο πραγματικό περιεχόμενο και είναι build χρησιμοποιώντας το γραφικό στοιχείο κειμένου.

Gestures

Τα γραφικά στοιχεία Flutter υποστηρίζουν την αλληλεπίδραση μέσω ενός ειδικού γραφικού στοιχείου, του GestureDetector. Το GestureDetector είναι ένα αόρατο γραφικό στοιχείο που έχει τη δυνατότητα να καταγράφει τις αλληλεπιδράσεις των χρηστών όπως ως πάτημα, μεταφορά, κ.λπ., του θυγατρικού του γραφικού στοιχείου. Πολλά native γραφικά στοιχεία υποστήριξης Flutter αλληλεπίδραση μέσω της χρήσης του GestureDetector. Μπορούμε επίσης να ενσωματώσουμε διαδραστική δυνατότητα στο υπάρχον γραφικό στοιχείο συνθέτοντάς το με το γραφικό στοιχείο GestureDetector.

Τα γραφικά στοιχεία Flutter υποστηρίζουν τη συντήρηση παρέχοντας ένα ειδικό γραφικό στοιχείο, το StatefulWidget. Το γραφικό στοιχείο πρέπει να προέρχεται από το γραφικό στοιχείο StatefulWidget για υποστήριξη κατάστασης συντήρησης και όλα τα άλλα γραφικά στοιχεία θα πρέπει να προέρχονται από το StatelessWidget. Τα γραφικά στοιχεία Flutter είναι αντιδραστικά. Η εκ νέου απόδοση βελτιστοποιείται με την εύρεση της διαφοράς μεταξύ παλιάς και νέας διεπαφή χρήστη γραφικού στοιχείου και απόδοση μόνο των απαραίτητων αλλαγών.

Επίπεδα -Layers

Η πιο σημαντική έννοια του πλαισίου Flutter είναι ότι το πλαίσιο ομαδοποιείται σε πολλαπλή κατηγορία ως προς την πολυπλοκότητα. Ένα επίπεδο δημιουργείται χρησιμοποιώντας το αμέσως επόμενο επίπεδο του. Το πάνω στρώμα είναι widget ειδικά για Android και iOS. Το επόμενο επίπεδο έχει όλα τα native γραφικά στοιχεία flutter. Το επόμενο επίπεδο είναι το επίπεδο απόδοσης, το οποίο είναι στοιχείο απόδοσης χαμηλού επιπέδου και αποδίδει τα πάντα την εφαρμογή flutter.

Τα ακόλουθα σημεία συνοψίζουν την αρχιτεκτονική του Flutter:

• Στο Flutter, τα πάντα είναι ένα γραφικό στοιχείο και ένα σύνθετο γραφικό στοιχείο αποτελείται ήδη από υπάρχοντα γραφικά στοιχεία.

• Διαδραστικές λειτουργίες μπορούν να ενσωματωθούν χρησιμοποιώντας το GestureDetector

widget.

• Η κατάσταση ενός γραφικού στοιχείου μπορεί να διατηρηθεί όποτε είναι απαραίτητο το StatefulWidget widget.

• Το Flutter προσφέρει πολυεπίπεδη σχεδίαση έτσι ώστε να μπορεί να προγραμματιστεί οποιοδήποτε στρώμα ανάλογα με την πολυπλοκότητα της εργασίας.

3.3 Dart

Η Dart είναι μια γλώσσα προγραμματισμού ανοιχτού κώδικα γενικής χρήσης. Αναπτύχθηκε αρχικά από την Google. Η Dart είναι μια αντικειμενοστραφή γλώσσα με σύνταξη τύπου C. Υποστηρίζει έννοιες προγραμματισμού όπως διεπαφές, κλάσεις, σε αντίθεση με άλλες γλώσσες προγραμματισμού η Dart δεν υποστηρίζει πίνακες. Οι συλλογές Dart μπορούν να χρησιμοποιηθούν για την αναπαραγωγή δομών δεδομένων όπως πίνακες,

Ο παρακάτω κώδικας δείχνει ένα απλό πρόγραμμα Dart:

```
void main()
{
  print ("Liamos Dart");
}
```

Μεταβλητές και τύποι δεδομένων

Η μεταβλητή ονομάζεται θέση αποθήκευσης και οι τύποι δεδομένων αναφέρονται απλώς στον τύπο και το μέγεθος της δεδομένα που σχετίζονται με μεταβλητές και συναρτήσεις.

Το Dart χρησιμοποιεί τη λέξη-κλειδί var για να δηλώσει τη μεταβλητή. Η σύνταξη του var ορίζεται παρακάτω,

```
var name = 'Liamos';
```

Η τελική και η λέξη-κλειδί const χρησιμοποιούνται για τη δήλωση σταθερών.

Ορίζονται ως εξής:

```
void main() {
final b = 4;
  const pi = 3,14;
  print(b);
```

```
 print(pi);
}
```

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, τα γραφικά στοιχεία είναι τα πάντα στο πλαίσιο Flutter.

Προχωρώντας θα περιγραφθεί η πραγματική ιδέα πίσω από τη δημιουργία των γραφικών στοιχείων και των διαφορετικών τύπων γραφικών στοιχείων διαθέσιμα στο πλαίσιο Flutter.

Παράδειγμα το γραφικό στοιχείο MyHomePage της εφαρμογής Hi Liamos. Ο κωδικός για αυτό το σκοπό δίνεται παρακάτω:

```
class MyHomePage extends StatelessWidget {
 MyHomePage({Key key, this.title}) : super(key: key);
 final String title;
 @override
 Widget build(BuildContext context) {
 return Scaffold(
 appBar: AppBar(
 title: Text(this.title),
 ),
 body: Center(
 child: Text(
 'Hi Liamos',
 )),
 );
 }
}
```

Δημιουργήθηκε ένα νέο γραφικό στοιχείο επεκτείνοντας το StatelessWidget. Το StatelessWidget απαιτεί μόνο μια ενιαία έκδοση μεθόδου για την υλοποίηση την παραγόμενη κλάση του. Η μέθοδος κατασκευής λαμβάνει το περιβάλλον περιβάλλοντος που είναι απαραίτητο για τη δημιουργία των γραφικών στοιχείων μέσω της παραμέτρου BuildContext και επιστρέφει το γραφικό στοιχείο που δημιουργεί. Στον κώδικα χρησιμοποιήθηκε ο τίτλος ως ένα από τα ορίσματα του κατασκευαστή Εδώ η μέθοδος κατασκευής καλεί τη μέθοδο κατασκευής του Scaffold, η οποία με τη σειρά της καλεί τη

δημιουργία μέθοδος του AppBar και του Center για τη δημιουργία της διεπαφής. Τέλος, η μέθοδος δημιουργίας Center καλεί τη μέθοδο δημιουργίας κειμένου.

3.4 PHP

Ο Προεπεξεργαστής Υπερκειμένου PHP (PHP) είναι μια γλώσσα προγραμματισμού που επιτρέπει στους προγραμματιστές Ιστού να δημιουργούν δυναμικό περιεχόμενο που αλληλεπιδρά με βάσεις δεδομένων. Η PHP χρησιμοποιείται βασικά για την ανάπτυξη εφαρμογών λογισμικού που βασίζονται στο web.

Η PHP ξεκίνησε ως ένα μικρό έργο ανοιχτού κώδικα που εξελίχθηκε καθώς όλο και περισσότεροι άνθρωποι έβρισκαν πόσο χρήσιμη ήταν.Το 1994 κυκλοφόρησε η πρώτη έκδοση της PHP.[9]

 Η PHP είναι ένα αναδρομικό αρκτικόλεξο για το "PHP: Hypertext Preprocessor".

 Η PHP είναι μια γλώσσα δέσμης ενεργειών από την πλευρά του διακομιστή που είναι ενσωματωμένη σε HTML. Χρησιμοποιείται για τη διαχείριση δυναμικού περιεχόμενου, βάσεις δεδομένων, παρακολούθηση περιόδων σύνδεσης και για δημιουργία ολόκληρων τοποθεσιών ηλεκτρονικού εμπορίου.

 Είναι ενσωματωμένη με μια σειρά από δημοφιλείς βάσεις δεδομένων, συμπεριλαμβανομένων των MySQL, PostgreSQL, Oracle και Microsoft SQL Server.

 Η PHP είναι zippy στην εκτέλεσή της, ειδικά όταν μεταγλωττίζεται ως λειτουργική μονάδα Apache στην πλευρά του Unix. Ο διακομιστής MySQL, μόλις ξεκινήσει, εκτελεί περίπλοκα ερωτήματα σε πολύ γρήγορο χρόνο.

 Η PHP υποστηρίζει μεγάλο αριθμό μεγάλων πρωτοκόλλων όπως POP3, IMAP και LDAP.

 Πρόσθεσε υποστήριξη για Java και αρχιτεκτονικές κατανεμημένων αντικειμένων υποστηρίζοντας δυνατότητα ανάπτυξης n-tier.

 Η σύνταξη της PHP είναι C-Like.

Οι άλλες χρήσεις της PHP είναι:

 Η PHP μπορεί να χειρίζεται φόρμες, δηλαδή να συλλέγει δεδομένα από αρχεία, να αποθηκεύει δεδομένα σε ένα αρχείο

 Πρόσβαση σε μεταβλητές cookies και δημιουργεί cookie.

 Μέσω email μπορεί να στείλει δεδομένα, να επιστρέψει δεδομένα στον χρήστη.

 Μπορεί να κρυπτογραφήσει δεδομένα.

 Χρησιμοποιώντας την PHP μπορεί να περιοριστεί η πρόσβαση των χρηστών σε ορισμένες σελίδες του ιστότοπού.

 Προσθέτει, διαγράφει, τροποποιεί στοιχεία στη βάση δεδομένων

Μερικά από τα βασικά χαρακτηριστικά της PHP είναι:

 Ασφάλεια

 Ευελιξία

 Απλότητα

 Οικειότητα

 Αποτελεσματικότητα

Ένας απλός κώδικας παρουσιάζεται παρακάτω:

```
<html>
<head>
<title>Γεια</title>
</head>
< body >
  <?php echo "Hi Liamos";?>
</body>
</html>
```

Θα παράγει το εξής αποτέλεσμα:

Hi Liamos

Παρατηρώντας την έξοδο HTML του παραπάνω παραδείγματος,φαίνεται ότι ο κώδικας PHP δεν είναι υπάρχει στο αρχείο που αποστέλλεται από τον διακομιστή στο πρόγραμμα περιήγησής στο Web.

Όλος ο κώδικας της PHP που υπάρχει στην ιστοσελίδα υποβάλλεται σε επεξεργασία και αφαιρείται από τη σελίδα. το μόνο που επέστρεψε στον πελάτη από το διακομιστή Ιστού είναι καθαρή HTML.

Για την ανάπτυξη και εκτέλεση ιστοσελίδων PHP πρέπει να εγκατασταθούν τρία στοιχεία το σύστημα του υπολογιστή.

Διακομιστής Ιστού (browser) - Η PHP λειτουργεί με σχεδόν όλο το λογισμικό διακομιστή Ιστού

Για το server θα χρειαστεί Apache, ngnix ή Διακομιστής Πληροφοριών Διαδικτύου (IIS)

Βάση δεδομένων – Η PHP υποστηρίζει όλες τις γνωστές βάσεις και οι πιο πολλοί προγραμματιστές προτιμούν την MySQL.

PHP Parser - Για την επεξεργασία κώδικα PHP πρέπει να εγκατασταθεί ένας αναλυτής-κειμενογράφος

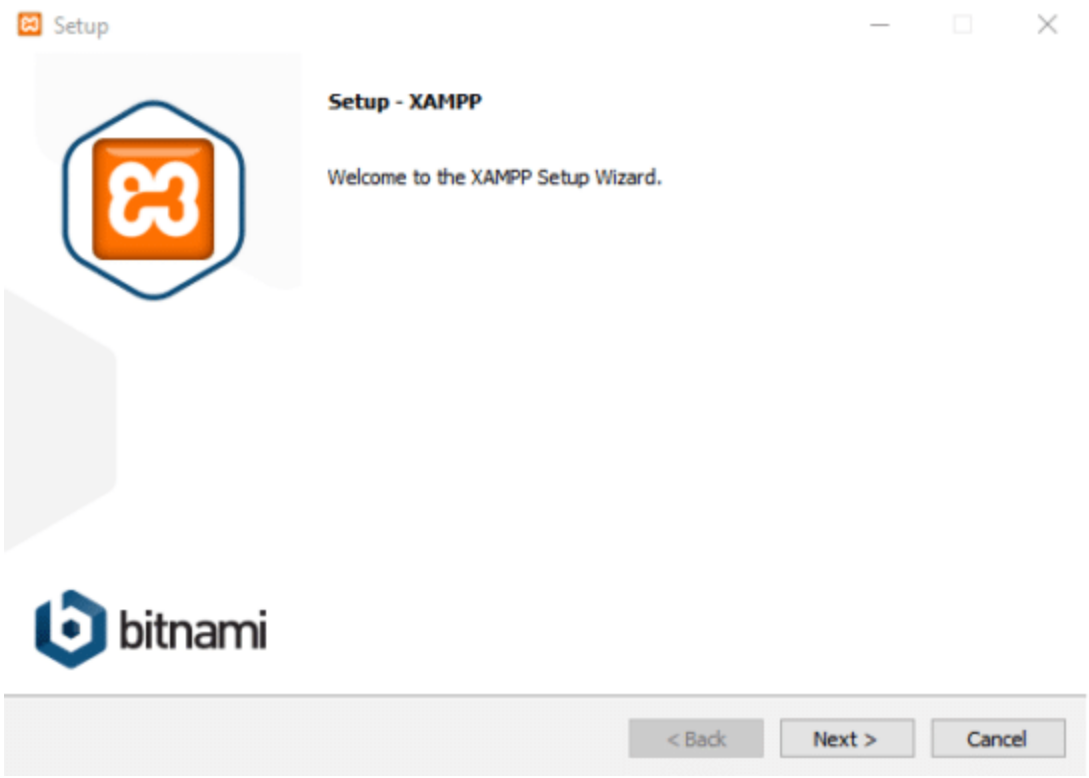
3.4.1 Server + PHP

Το Apache Xampp είναι ένα εργαλείο ανοιχτού κώδικα που χρησιμοποιείται για την εκτέλεση εφαρμογών Ιστού PHP ή PERL τοπικά χρησιμοποιώντας έναν διακομιστή Ιστού. Είναι διαθέσιμο για όλα τα μεγάλα λειτουργικά συστήματα και είναι δημοφιλές στους χρήστες των Windows για την τοπική κατασκευή και δοκιμή των εφαρμογών Ιστού τους. Παρακάτω θα διερευνηθεί πώς να ξεκινήσει κάποιος με το Xampp. Θα δημιουργηθεί μια εφαρμογή Web σε PHP και θα χρησιμοποιηθούν βάσεις δεδομένων MySQL με τη βοήθεια του εργαλείου PHPMyAdmin. [10]

Το Xampp είναι ένα πακέτο διακομιστών Ιστού δωρεάν και ανοιχτού κώδικα πολλαπλών πλατφορμών που αναπτύχθηκε από την Apache Friends. Έρχεται φορτωμένο με διακομιστή HTTP Apache, MariaDB και MySQL. Διαθέτει επίσης διερμηνείς για τις γλώσσες προγραμματισμού PHP και PERL. Το Xampp είναι μια από τις πιο δημοφιλείς λύσεις για την εκτέλεση εφαρμογών PHP τοπικά στα Windows.

Λήψη Xampp

Η λήψη του Xampp είναι εξαιρετικά εύκολη και δεν απαιτεί ειδικές οδηγίες. Πατώντας στο Google "Xampp". Στη συνέχεια κλικ στο κουμπί Λήψη για Windows. Αυτό θα πρέπει να πραγματοποιήσει λήψη ενός αρχείου .exe στον υπολογιστή. Με την ολοκλήρωση της λήψης εκτελέστε το εκτελέσιμο αρχείο.



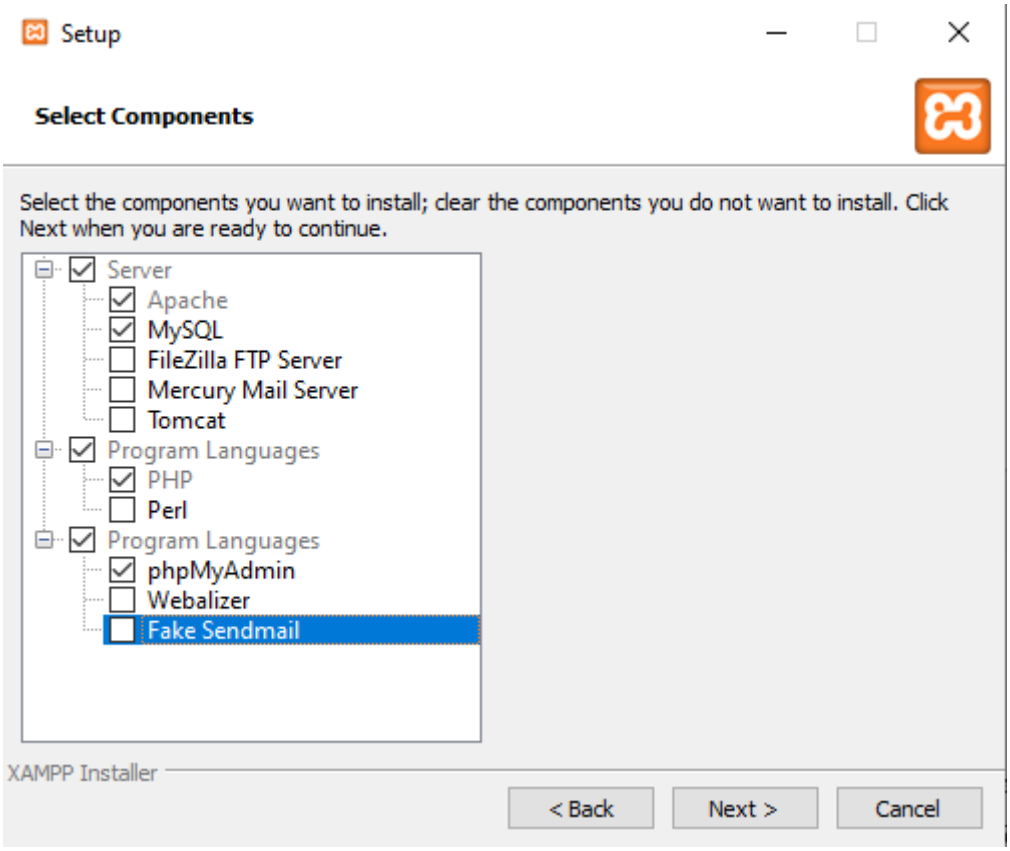
Εικόνα 3.6: Εγκατάσταση XAMPP

Εγκατάσταση του Xampp

Με το άνοιγμα του εκτελέσιμου αρχείου, θα δείτε το παράθυρο εγκατάστασης στην οθόνη όπως φαίνεται στην Εικόνα 3.6. Κάντε κλικ στο Επόμενο και ξεκινήστε την εγκατάσταση. Θα σας ζητηθεί να επιλέξετε τα στοιχεία προς εγκατάσταση. Μπορείτε είτε να επιλέξετε και να εγκαταστήσετε όλα τα στοιχεία του πακέτου είτε να εγκαταστήσετε μόνο αυτά που χρειάζεστε.

Επιλέξτε PHP, MySQL, Apache και PHPMyAdmin.

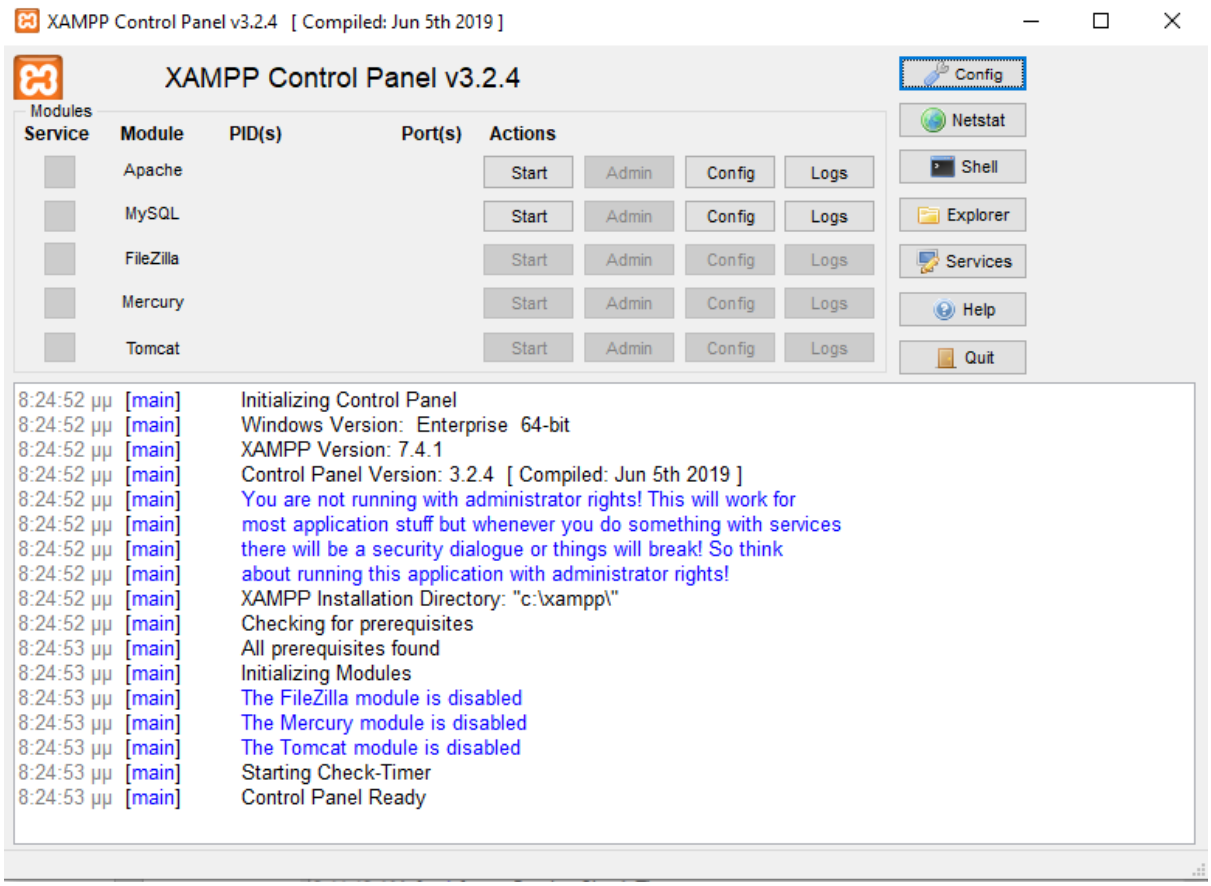
Για απλότητα, επιλέξαμε και εγκαταστήσαμε όλα τα εξαρτήματα, όπως φαίνεται στην Εικόνα 3.7.



Εικόνα 3.7: Εγκατάσταση μόνο των σημαντικών στοιχείων

Θα ζητηθεί η εγκατάσταση της επέκτασης Bitnami για WordPress κλπ. μαζί με το πακέτο Xampp. Αποεπιλέξτε το πλαίσιο.

Στη συνέχεια θα πρέπει να επιλέξετε τη θέση όπου θέλετε να εγκατασταθεί το Xampp. Με την ολοκλήρωση της εγκατάστασης, επιλέξτε «Εκκίνηση Xampp» και κάντε κλικ στο Τέλος.



Εικόνα 3.8: Το Control Panel του XAMPP

Το Xampp

Μετά την επιτυχή εγκατάσταση και εκτέλεση του Xampp, προβάλλεται ο πίνακας ελέγχου Xampp στην οθόνη, όπως φαίνεται στην Εικόνα 3.8. Χρησιμοποιώντας αυτόν τον πίνακα, μπορείτε να ξεκινήσετε ή να σταματήσετε τις υπηρεσίες που απαιτούνται για την ανάπτυξη και την εκτέλεση της εφαρμογής Ιστού. Κάντε κλικ στο κουμπί Έναρξη για τον Apache, ο οποίος θα χρησιμεύσει ως τοπικός διακομιστής Ιστού και την MySQL η οποία είναι η βάση δεδομένων για την εφαρμογή Ιστού PHP.

Ανοίγουμε το πρόγραμμα περιήγησής (chrome ή firefox) στο Web και πληκτρολογήστε localhost ή 127.0.0.1, που είναι η προεπιλεγμένη διεύθυνση IP για τον τοπικό διακομιστή Web.

Ανάπτυξη της εφαρμογής Web PHP στον τοπικό διακομιστή

Έστω ότι θέλουμε να δημιουργίσυμε μια νέα εφαρμογή Ιστού «Hi Liamos» με χρήση PHP. Για να το κάνετε αυτό, πρέπει να εντοπίσετε το φάκελο htdocs και να βάλετε τον κώδικα PHP-το αρχείο του με προέκταση .php- σε αυτόν το φάκελο.

Μεταβείτε στο σημείο όπου κατεβάσατε και εγκαταστήσατε το Xampp. Συνήθως είναι

C:/Xampp/ εκτός αν έχετε επιλέξει διαφορετικό κατάλογο προορισμού κατά την εγκατάσταση.

Στο φάκελο Xampp, εντοπίστε τον κατάλογο htdocs.

Επιλέξτε όλα τα αρχεία και τους φακέλους στον κατάλογο htdocs και διαγράψτε τα. Στη συνέχεια, δημιουργήστε ένα μόνο αρχείο με το όνομα index.php. Ένα αρχείο php σε μια εφαρμογή Ιστού είναι ένα αρχείο που ανοίγει από προεπιλογή όταν γίνεται πρόσβαση στον τομέα της εφαρμογής Ιστού.

Ανοίξτε το αρχείο index.php χρησιμοποιώντας οποιοδήποτε πρόγραμμα επεξεργασίας κώδικα - κειμενογράφο. Μόλις ανοίξει, γράψτε τον παρακάτω κώδικα και πατήστε Αποθήκευση.

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "My first PHP script!";
?>
</body>
</html>
```

και πατήστε στον browser http://localhost

για να δείτε το PHP script να τρέχει

Χρήση MySQL με Xampp

Αφού δημιοργήσαμε μια απλή εφαρμογή Ιστού με χρήση PHP, θα προχωρήσουμε στη σύνδεση της εφαρμογής με μια βάση δεδομένων. Υπάρχουν δύο βασικές υπηρεσίες στον πίνακα ελέγχου του Xampp. Ο ένας είναι ο διακομιστής Web Apache και ο άλλος είναι η βάση δεδομένων MySQL. Η MySQL είναι μια δωρεάν και ανοιχτού κώδικα σχεσιακή βάση δεδομένων που χρησιμοποιείται ευρέως σε πολλές εφαρμογές Ιστού στο Διαδίκτυο.

Θα χρησιμοποιήσουμε το εργαλείο PHPMyAdmin για πρόσβαση σε βάσεις δεδομένων MySQL. Αυτό το εργαλείο είναι γραμμένο σε PHP για να χειρίζεται τη διαχείριση της MySQL. Για να αποκτήσετε πρόσβαση στο PHPMyAdmin, ανοίξτε μια νέα καρτέλα στο πρόγραμμα περιήγησής σας στο Web και πληκτρολογήστε localhost/phpmyadmin ή 127.0.0.1/phpmyadmin. Αυτό θα πρέπει να ανοίξει τη διεπαφή PHPMyAdmin.

Στο αριστερό παράθυρο του PHPMyAdmin, όπως φαίνεται στην Εικόνα 3.9 υπάρχει μια λίστα με προφορτωμένες βάσεις δεδομένων που μπορείτε να χρησιμοποιήσετε. Μπορούμε να δημιουργήσουμε μια δική μας βάση δεδομένων,

Εικόνα 3.9: Διαχειριστικό της MySQL μέσω phpMyAdmin

3.5 MySql

Μια βάση δεδομένων είναι μια ξεχωριστή εφαρμογή που αποθηκεύει μια συλλογή δεδομένων. Κάθε βάση δεδομένων έχει ένα ή περισσότερα ξεχωριστά API για τη δημιουργία, την πρόσβαση, τη διαχείριση, την αναζήτηση και την αναπαραγωγή των δεδομένων που διατηρεί.

Μπορούν επίσης να χρησιμοποιηθούν και άλλα είδη αποθήκευσης δεδομένων, όπως αρχεία στο σύστημα αρχείων ή μεγάλοι πίνακες κατακερματισμού στη μνήμη, αλλά η ανάκτηση και η εγγραφή δεδομένων δεν θα ήταν τόσο γρήγορη και εύκολη με τέτοιου είδους συστήματα.

Σήμερα, υπάρχουν συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) για την αποθήκευση και τη διαχείριση τεράστιου όγκου δεδομένων. Αυτή ονομάζεται σχεσιακή βάση δεδομένων επειδή όλα τα δεδομένα αποθηκεύονται σε διαφορετικούς πίνακες και οι σχέσεις δημιουργούνται χρησιμοποιώντας πρωτεύοντα κλειδιά ή άλλα κλειδιά γνωστά ως Ξένα κλειδιά.

Ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS) είναι ένα λογισμικό που δίνει τη δυνατότητα υλοποίησης μιας βάσης δεδομένων με πίνακες, στήλες και ευρετήρια. Εγγυάται την ακεραιότητα αναφοράς μεταξύ σειρών διαφόρων πινάκων. Ενημερώνει αυτόματα τα ευρετήρια. Ερμηνεύει ένα ερώτημα SQL και συνδυάζει πληροφορίες από διάφορους πίνακες.

Ορολογία RDBMS

Μερικές ορολογίες της MySQL που σχετίζονται με τη βάση δεδομένων είναι:

Βάση δεδομένων − Μια βάση δεδομένων είναι μια συλλογή πινάκων, με σχετικά δεδομένα.

Πίνακας − Ένας πίνακας είναι ένας πίνακας με δεδομένα. Ένας πίνακας σε μια βάση δεδομένων μοιάζει με ένα απλό υπολογιστικό φύλλο.

Στήλη − Μία στήλη (στοιχείο δεδομένων) περιέχει δεδομένα του ίδιου είδους

Σειρά − Μια σειρά (πλειάδα, καταχώρηση ή εγγραφή) είναι μια ομάδα σχετικών δεδομένων, για παράδειγμα τα δεδομένα μιας συνδρομής.

Πλεονασμός − Αποθήκευση δεδομένων δύο φορές, πλεονάζουσα για να γίνει το σύστημα πιο γρήγορο.

Πρωτεύον κλειδί − Ένα πρωτεύον κλειδί είναι μοναδικό. Μια βασική τιμή δεν μπορεί να εμφανιστεί δύο φορές σε έναν πίνακα. Με ένα κλειδί, μπορείτε να βρείτε μόνο μία σειρά.

Ξένο κλειδί − Ένα ξένο κλειδί είναι η συνδετική ακίδα μεταξύ δύο πινάκων.

Σύνθετο κλειδί − Ένα σύνθετο κλειδί (σύνθετο κλειδί) είναι ένα κλειδί που αποτελείται από πολλές στήλες, επειδή μια στήλη δεν είναι επαρκώς μοναδική.

Ευρετήριο − Ένα ευρετήριο σε μια βάση δεδομένων μοιάζει με ευρετήριο στο πίσω μέρος ενός βιβλίου.

Ακεραιότητα αναφοράς − Η ακεραιότητα αναφοράς διασφαλίζει ότι μια τιμή ξένου κλειδιού οδηγεί πάντα σε μια υπάρχουσα σειρά.

Βάση δεδομένων MySQL

Η MySQL είναι ένα γρήγορο, εύχρηστο RDBMS που χρησιμοποιείται για πολλές μικρές και μεγάλες επιχειρήσεις. Η MySQL αναπτύσσεται, διατίθεται στην αγορά και υποστηρίζεται από την σουηδική εταιρεία MySQL AB. Η MySQL γίνεται τόσο δημοφιλής για πολλούς λόγους:

Η MySQL κυκλοφορεί με άδεια ανοιχτού κώδικα.

Η MySQL λειτουργεί πολύ γρήγορα και λειτουργεί καλά ακόμα και με μεγάλα σύνολα δεδομένων.

Η MySQL χρησιμοποιεί μια τυπική μορφή της γνωστής γλώσσας δεδομένων SQL.

Η MySQL λειτουργεί σε πολλά λειτουργικά συστήματα και με πολλές γλώσσες όπως PHP, C, C++, JAVA, Python, Javascript κλπ.

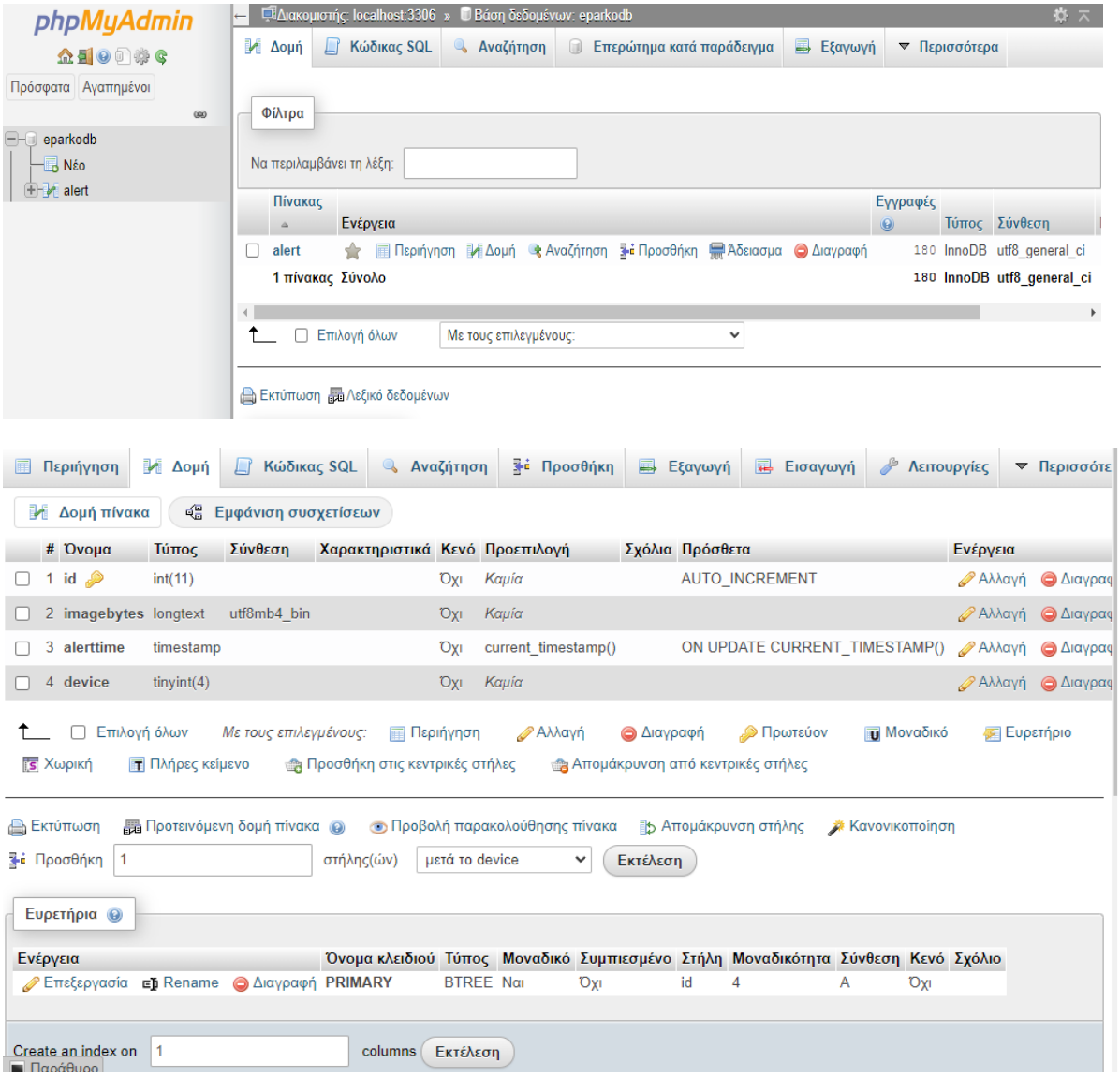
Η MySQL είναι πολύ φιλική προς την PHP, την πιο αξιόλογη γλώσσα για την ανάπτυξη ιστού.

Το MySQL είναι ένα πολύ ισχυρό πρόγραμμα από μόνο του. Διαχειρίζεται ένα μεγάλο υποσύνολο της λειτουργικότητας των πιο ακριβών και ισχυρών πακέτων βάσεων δεδομένων.

Η MySQL υποστηρίζει μεγάλες βάσεις δεδομένων, έως και 50 εκατομμύρια σειρές ή περισσότερες σε έναν πίνακα. Το προεπιλεγμένο όριο μεγέθους αρχείου για έναν πίνακα είναι 4 GB.

Η MySQL είναι προσαρμόσιμη. Η άδεια GPL ανοιχτού κώδικα επιτρέπει στους προγραμματιστές να τροποποιούν το λογισμικό MySQL ώστε να ταιριάζει στα δικά τους συγκεκριμένα περιβάλλοντα.

Για να δείξουμε τη χρήση της MySQL, θα δημιουργήσουμε έναν πίνακα με το όνομα «alert». Σε αυτόν τον πίνακα, θα δημιουργήσουμε τις στήλες που φαίνονται στην Εικόνα 3.10.



Εικόνα 3.10: Πίνακας και η Δομή του - MySQL

Στη συνέχεια, θα προσπαθήσουμε να συνδέσουμε την εφαρμογή Ιστού με αυτήν τη βάση δεδομένων και να ανακτήσουμε τα δεδομένα από τον πίνακα alert. Η Εικόνα 3.11 δείχνει τον πίνακα με δεδομένα.

Εικόνα 3.11: Δεδομένα του Πίνακα - MySQL

Μέσω PHP συνδεόμαστε με την βάση σύμφωνα με τον παρακάτω κώδικα

```
$servername = "localhost";

$username = "eparkouser";

$password = "eparkousernovalid ";

$dbname = "eparkodb";


// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}
```

και πραγματοποιούμε ανάγνωση των δεδομένων μέσω

```
$sql = "SELECT * FROM `alert` ORDER BY `id` DESC";
```

```
$result = $conn->query($sql);


if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
   $arr = array('imagebytes' => $row["imagebytes"], 'alerttime' => $row["alerttime"]);
        echo json_encode($arr);
} else {
  return 0;
}
```

3.6 MIT Inventor

Το MIT App Inventor είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών ιστού που αρχικά παρεχόταν από την Google και τώρα διατηρείται από το Ινστιτούτο Τεχνολογίας της Μασαχουσέτης (MIT). Επιτρέπει στους νεοεισερχόμενους στον προγραμματισμό υπολογιστών να δημιουργούν λογισμικό(εφαρμογές) εφαρμογών για δύο λειτουργικά συστήματα (OS): Android και iOS, το οποίο, από το 2019, βρίσκεται σε τελική δοκιμή beta. Είναι δωρεάν λογισμικό ανοιχτού κώδικα. [11]

Χρησιμοποιεί ένα γραφικό περιβάλλον χρήστη (GUI) πολύ παρόμοιο με τις γλώσσες προγραμματισμού Scratch (γλώσσα προγραμματισμού) και το StarLogo το οποίο επιτρέπει στους χρήστες να μεταφέρουν και να τοποθετήσουν οπτικά αντικείμενα για να δημιουργήσουν μια εφαρμογή που μπορεί να εκτελεστεί σε συσκευές Android.

Το App-Inventor Companion είναι το πρόγραμμα που επιτρέπει στην εφαρμογή την εκτέλεση και τον εντοπισμό σφαλμάτων.

Το App Inventor και τα άλλα έργα βασίζονται και ενημερώνονται από θεωρίες μάθησης δομιστών, οι οποίες τονίζουν ότι ο προγραμματισμός μπορεί να αποτελέσει όχημα για την εμπλοκή ισχυρών ιδεών μέσω της μάθησης.

Το App Inventor υποστηρίζει επίσης τη χρήση δεδομένων cloud μέσω ενός πειραματικού στοιχείου Firebase.

Το MIT App Inventor είναι ένα διαισθητικό, οπτικό περιβάλλον προγραμματισμού που επιτρέπει σε όλους να δημιουργούν πλήρως λειτουργικές εφαρμογές για smartphone και tablet Android και iOS. Στο MIT App Inventor μπορούν να έχουν μια απλή πρώτη εφαρμογή σε λειτουργία σε λιγότερο από 60

λεπτά. Επιπλέον, το εργαλείο βασίζεται σε μπλοκ και διευκολύνει τη δημιουργία πολύπλοκων εφαρμογών υψηλής απόδοσης σε πολύ λιγότερο χρόνο από τα παραδοσιακά περιβάλλοντα προγραμματισμού. Το έργο MIT App Inventor επιδιώκει να πρπωθήσει την ανάπτυξη λογισμικού ενδυναμώνοντας όλους τους ανθρώπους να περάσουν από την κατανάλωση τεχνολογίας στη δημιουργία τεχνολογίας.

Τα προγράμματα κωδικοποίησης που βασίζονται σε μπλοκ εμπνέουν πνευματική και δημιουργική ενδυνάμωση. Το MIT App Inventor προχωρά πέρα από αυτό για να παρέχει πραγματική ενδυνάμωση στα παιδιά να κάνουν τη διαφορά ο οποίος είναι ένας τρόπος να επιτύχουν κοινωνικό αντίκτυπο ανυπολόγιστης αξίας για τις κοινότητές τους.

Κεφάλαιο 4ο: Ανάλυση των μονάδων

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθούν οι μονάδες που απαρτίζουν το έργο. Θα αναλυθεί η εφαρμογή στο κινητό που τοποθετείται στο δωμάτιο και παρακολουθεί το χώρο, ο server που υλοποιήθηκε με PHP και η βάση MySQL που αποθηκεύονται οι ειδοποιήσεις (εικόνες κτλ) και τέλος η δεύτερη εφαρμογή που έχει ο χρήστης για να παρακολουθεί τις ειδοποιήσεις από τον server όταν βρίσκεται απομακρυσμένα από τον χώρο παρακολούθησης.

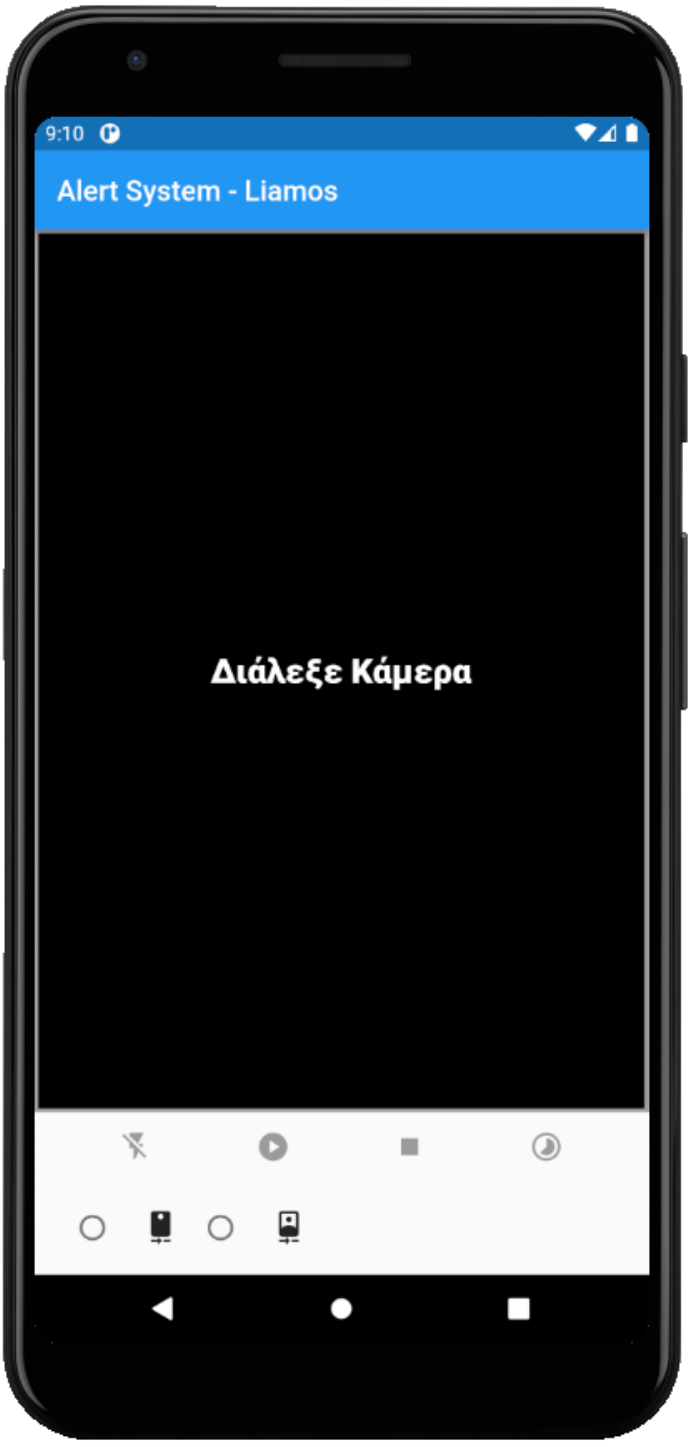
Η πρώτη εφαρμογή στο κινητό υλοποιήθηκε χρησιμοποιώντας το Flutter ενώ η δεύτερη το MIT App Inventor.

4.2 Ανάλυση εφαρμογής στο κινητό παρακολούθησης

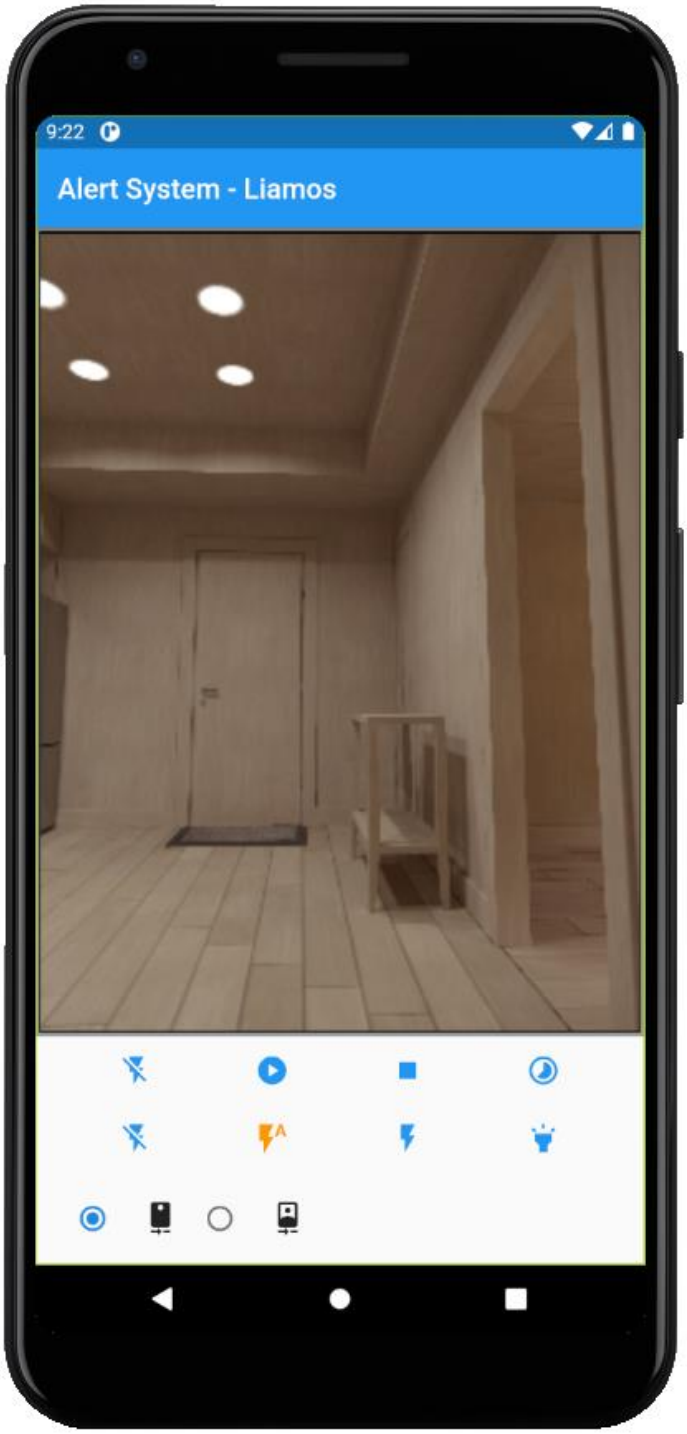
Στην Εικόνα 4.1 φαίνεται η αρχική οθόνη της εφαρμογής στο κινητό παρακολούθησης. Ο χρήστης αφού τοποθετήσει το κινητό στη θέση που η κάμερα του επιβλέπει τον χώρο που επιθυμεί πρέπει να επιλέξει την κάμερα. Συνηθώς επιλέγεται η πίσω κάμερα του κινητού ώστε η οθόνη να μην φαίνεται προς τον χώρο που επιβλέπει.

Δίνεται η δυνατότητά να χρησιμοποιηθεί και η μπροστά κάμερα για κάποιον χρήστη που θα ήθελε να πειραματιστεί ή να το χρησιμοποιήσει ως κεντρική λόγω χαμηλότερης ανάλυσης.

Να τονισθεί ότι η ανίχνευση κίνησης στο πεδίο παρακολούθησης πραγματοποιείται χωρίς πρόσβαση στο διαδίκτυο αλλά η αποστολή της εικόνας και του σήματος συναγερμού γίνεται μέσω διαδικτύου σε ειδικά διαμορφωμένο server και στη συνέχεια ο χρήστης μπορεί να ενημερωθεί μέσω μια άλλης εφαρμογής που έχει στο κινητό του.

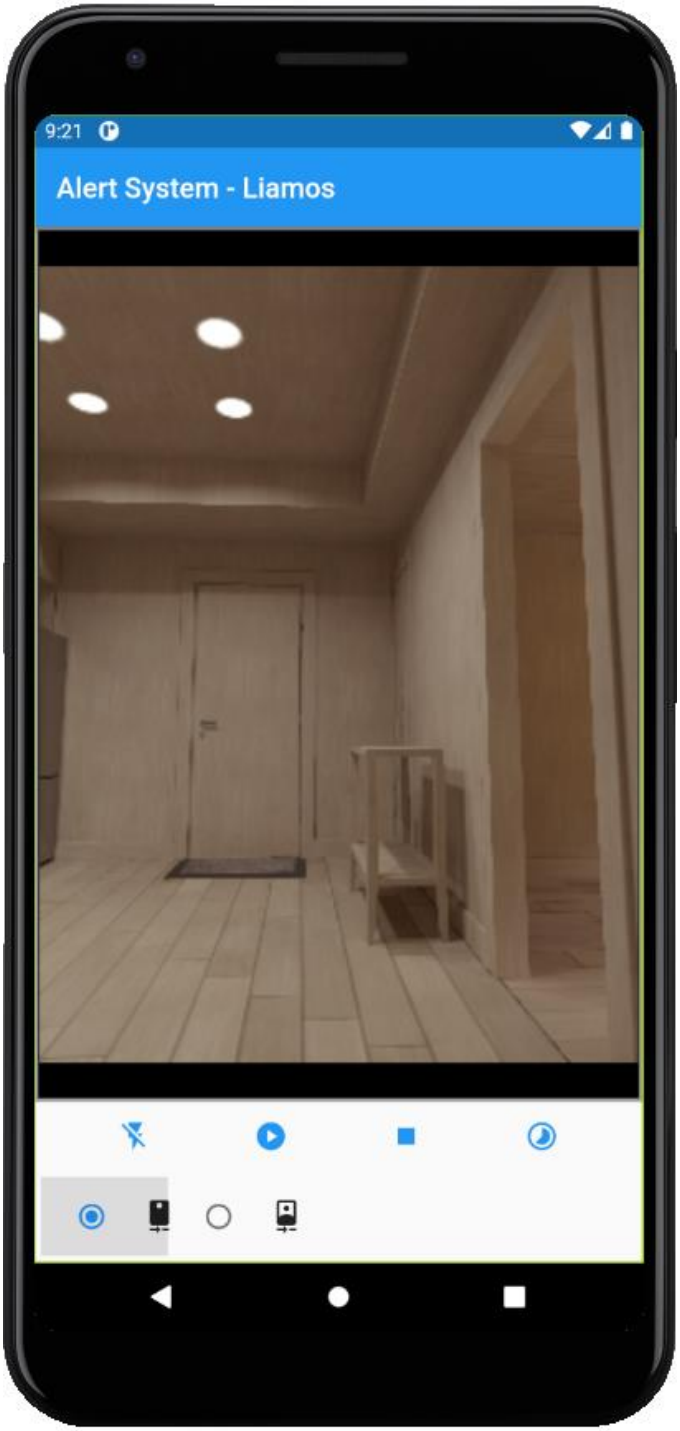


Εικόνα 4.1: Αρχική οθόνη της εφαρμογής στο κινητό παρακολούθησης



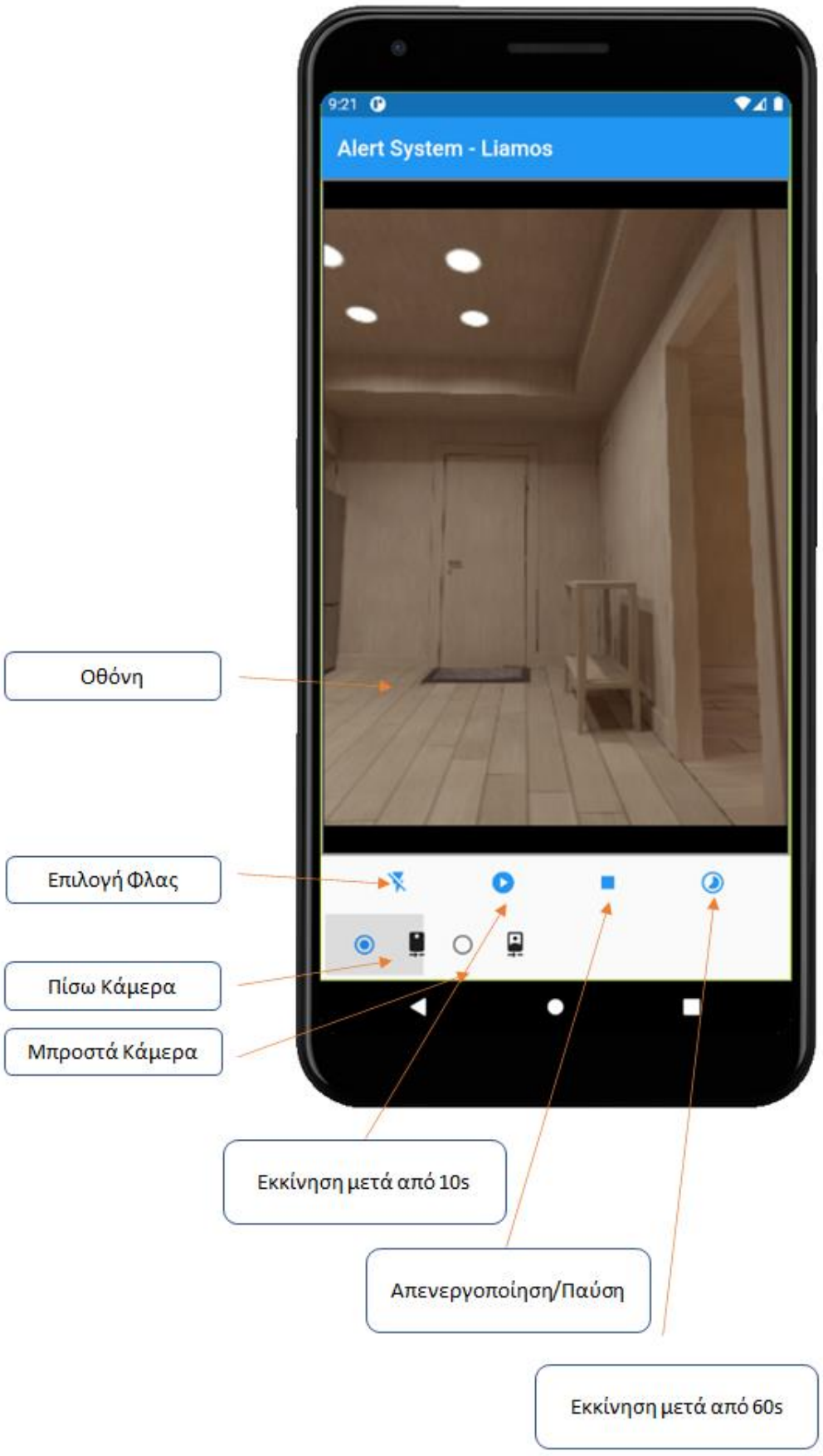
Εικόνα 4.2: Αρχική οθόνη της εφαρμογής με ενεργοποιημένο το μενού ρύθμισης του φλας

Στην Εικόνα 4.2 παρουσιάζεται η αρχική οθόνη της εφαρμογής με ενεργοποιημένο το μενού ρύθμισης του φλας, όπου ο χρήστης μπορεί να επιλέξει σε αυτόματο, μόνιμο ή απουσία του φλας κατά την λειτουργία ανίχνευσης.



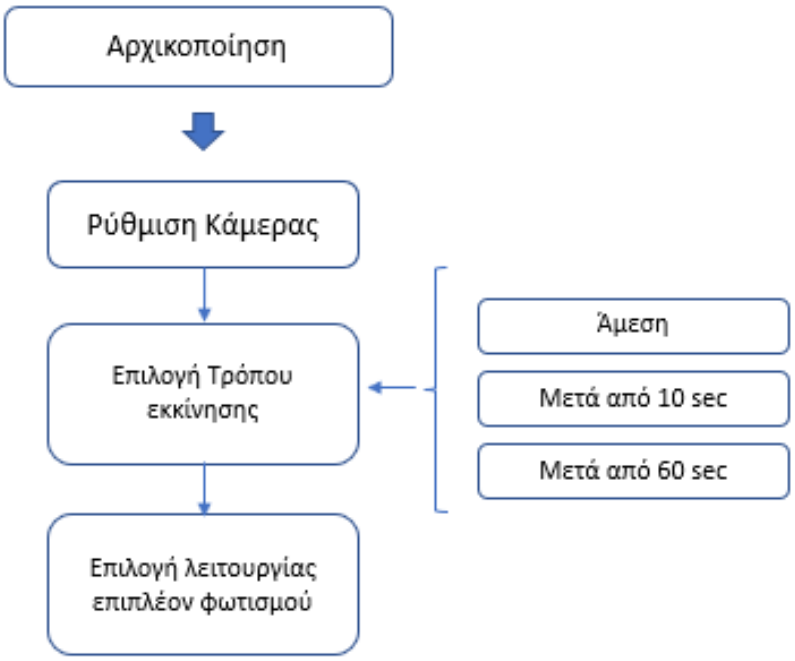
Εικόνα 4.3: Αρχική οθόνη της εφαρμογής με ενεργοποιημένη τη μπροστινή κάμερα

Στην Εικόνα 4.3 φαίνεται η αρχική οθόνη της εφαρμογής με ενεργοποιημένη τη μπροστινή κάμερα. Η εικόνα που παρουσιάζεται σαν κεντρική προέρχεται από τη μηχανή adb μέσω του flutter.



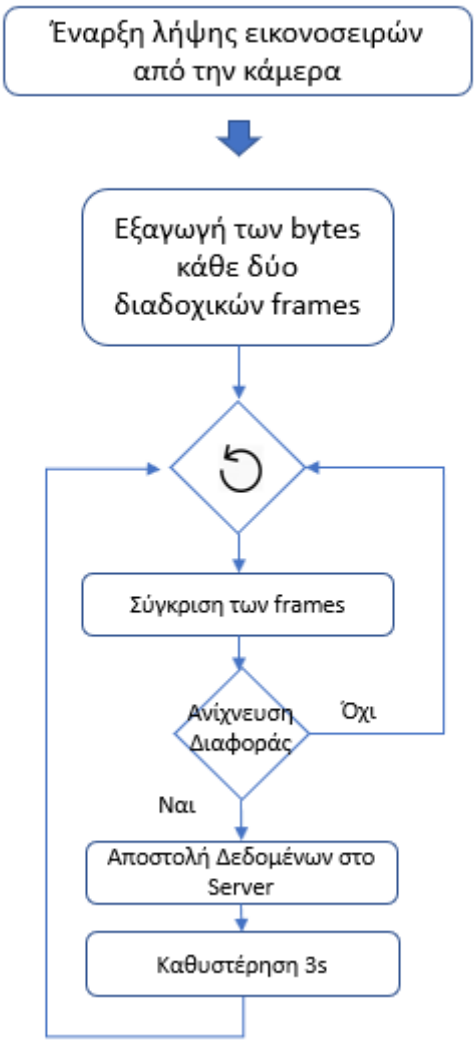
Εικόνα 4.4: Οθόνη της εφαρμογής με διευκρινήσεις για κάθε κουμπί-λειτουργία

Στην Εικόνα 4.4 φαίνεται η οθόνη της εφαρμογής με διευκρινήσεις για κάθε κουμπί-λειτουργία. Υπάρχουν δύο κουμπιά εκκίνησης της επίβλεψης του χώρου μέσω κάμερας, το ένα ξεκινάει την εφαρμογή μετά από 10 δευτερόλεπτα και το άλλο μετά από 60 δευτερόλεπτα. Ο λόγος που υπάρχει μια καθυστέρηση είναι λόγω του χρόνου που χρειάζεται ο χρήστης να φύγει από το οπτικό/εποπτικό πεδίο της κάμερας για να μην εντοπίζει ο η εφαρμογή τον ίδιο. Η εφαρμογή μπορεί να σταματήσει με το κουμπί Απενεργοποίησης/Παύσης.



Εικόνα 4.5: Διάγραμμα λειτουργίας εφαρμογής παρακολούθησης

Στην Εικόνα 4.5 παρουσιάζεται το διάγραμμα βασικής λειτουργίας εφαρμογής παρακολούθησης όσον αφορά την πλοήγηση και την εκκίνησης της.



Εικόνα 4.6: Διάγραμμα ανίχνευσης κίνησης στον χώρο

Στην Εικόνα 4.6 παρουσιάζεται το διάγραμμα ανίχνευσης κίνησης συγκρίνοντας δύο διαδοχικές εικόνες/εικονοσειρές.

Η σύγκριση πραγματοποιείται με τον παρακάτω τρόπο:

Συγκρίνονται δύο διαδοχικά frames.

Χρησιμοποιείται ο αλγόριθμος IMED για σύγκριση εικόνων με ευκλείδεια απόσταση εικόνας.

Το μέγεθος των εικόνων αλλάζει στις ίδιες διαστάσεις (αν οι διαστάσεις δεν ταιριάζουν)

και είναι σε κλίμακα του γκρι. Κατά τον υπολογισμό της απόστασης εφαρμόζεται ένα γκαουσιανό θάμπωμα μεταξύ των εντάσεων των εικονοστοιχείων. Η χωρική σχέση λαμβάνεται υπόψη στη συνάρτηση guassian για τη μείωση της επίδρασης μικρών διαταραχών.

```
sum(exp(-distance([i], [j]) ^2 / 2 * pi * sigma^2) * (src1[i] - src2[i]) *(src1[j] - src2[j]))
```
(Εξ.4.1)

```
        compareImages(
                src1: bytesofimage1,
                src2: bytesofimage2,
                algorithm: IMED(blurRatio: 0.001))
            .then((value) => {
                  print('Difference1: ${value * 100}%'),
                  if (value * 100 > 0.2)
                    {
```

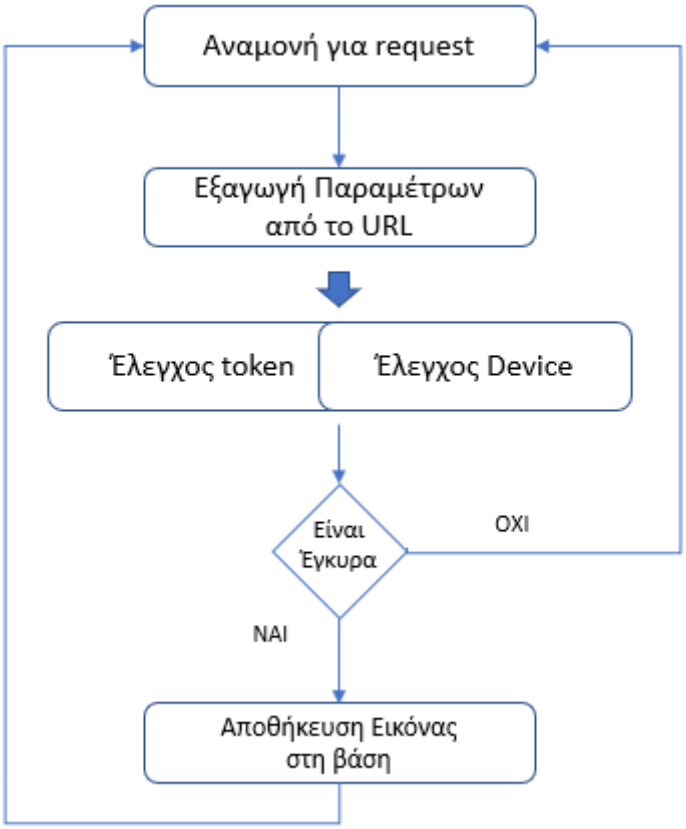
Επιστρέφει την ποσοστιαία διαφορά τους και όταν είναι πάνω από μια τιμή που ορίζεται εμπειρικά 0.2 τότε ενεργοποιεί τον συναγερμό.

Αφού ανιχνευτεί η κίνηση τότε πραγματοποιείται αποστολή της τελευταίας εικονοσειράς με τον χρόνο εντοπισμού στον PHP server. Να σημειωθεί ότι η εικόνα μετατρέπεται σε base64 σύμφωνα με το παρακάτω τρόπο:

```
          client.post(Uri.parse('https://url.com'),
                  headers: {},
                  body: {
                  'imagebytes': base64Encode(bytesofimage2),
                  'alerttime': alerttime.toString(),
                  'device': device,
                  'token': token
                  }),
```

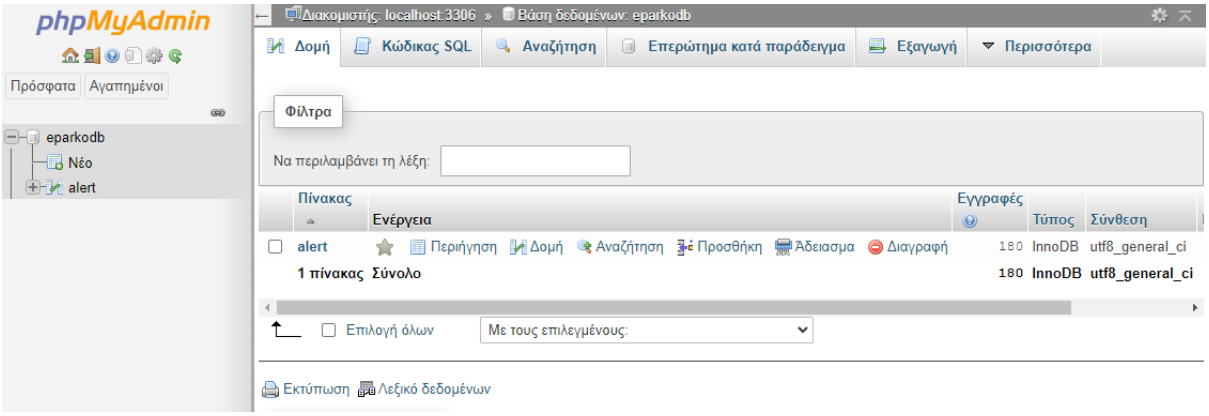
4.3 Ο PHP εξυπηρετητής-server και η βάση δεδομένων

Χρησιμοποιήθηκε ο xampp για να ενεργοποιηθεί ο apache server που εξυπηρετεί την PHP και ο mysql server που εξυπηρετεί τη βάση δεδομένων.

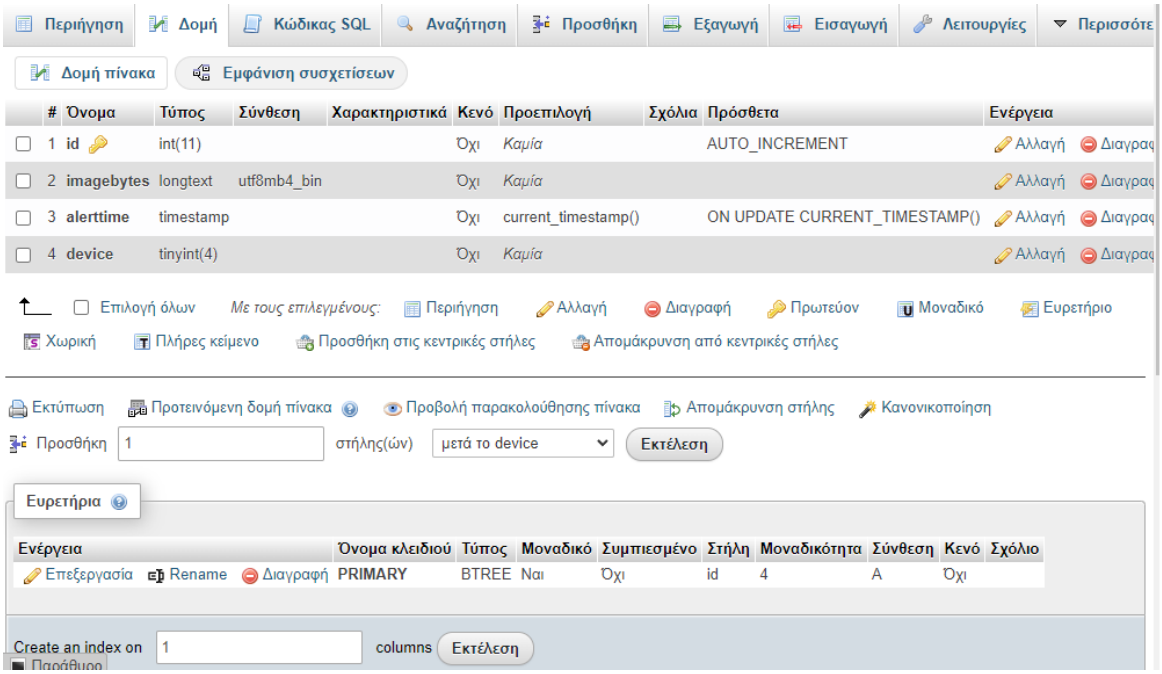


Εικόνα 4.7: Διάγραμμα για τη λήψη και αποθήκευση της ειδοποίησης στο server

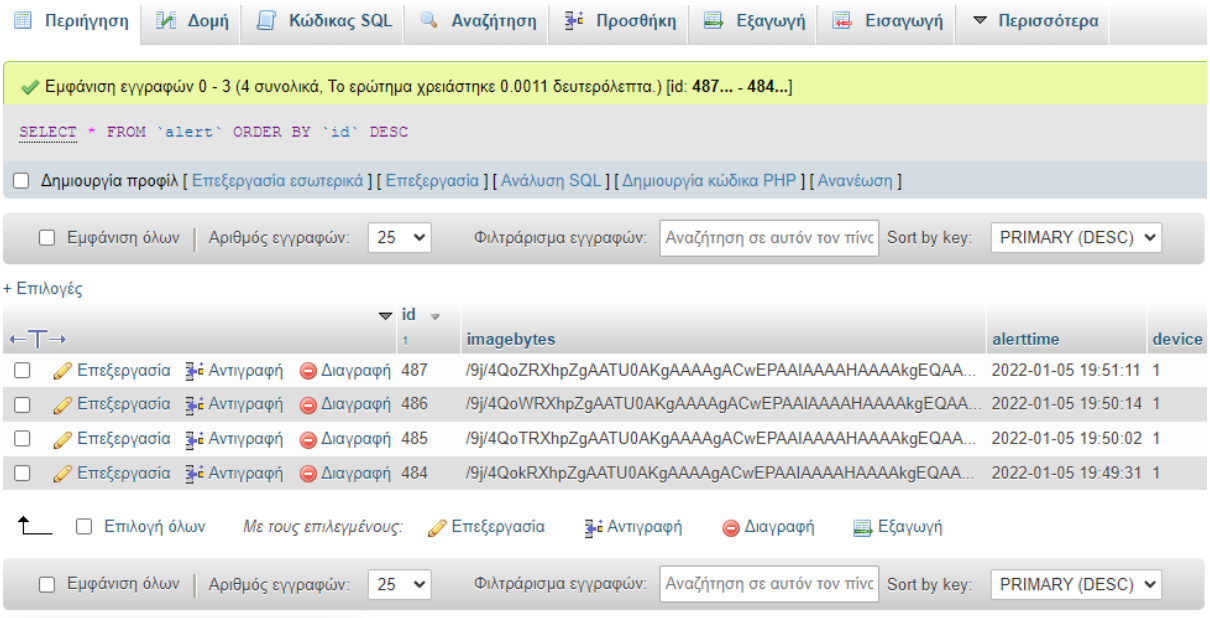
Στην Εικόνα 4.7 παρουσιάζεται το διάγραμμα για τη λήψη και αποθήκευση της ειδοποίησης στο server. Το PHP API αναμένει συνεχώς για νέες ειδοποιήσεις από κάποιο κινητό που επιβλέπει έναν χώρο. Όταν λαμβάνει ένα τέτοιο σήμα τότε εξάγει τις παραμέτρους από το URL και προχωράει σε έλεγχο του token και του device για να επιβεβαιώσει ότι αφορά επιτρεπόμενες συσκευές. Στη συνέχεια προχωράει αποθήκευση στη βάση που φαίνεται στην Εικόνα 4.8.



Εικόνα 4.8: Η βάση με τον πίνακα alert



Εικόνα 4.9: Η δομή του πίνακα alert



Εικόνα 4.10: Το περιεχόμενο του πίνακα alert με τις ειδοποιήσεις

Στην Εικόνα 4.9 φαίνεται η δομή του πίνακα alert και τι τύπος χρησιμοποιήθηκε για κάθε πεδίο.

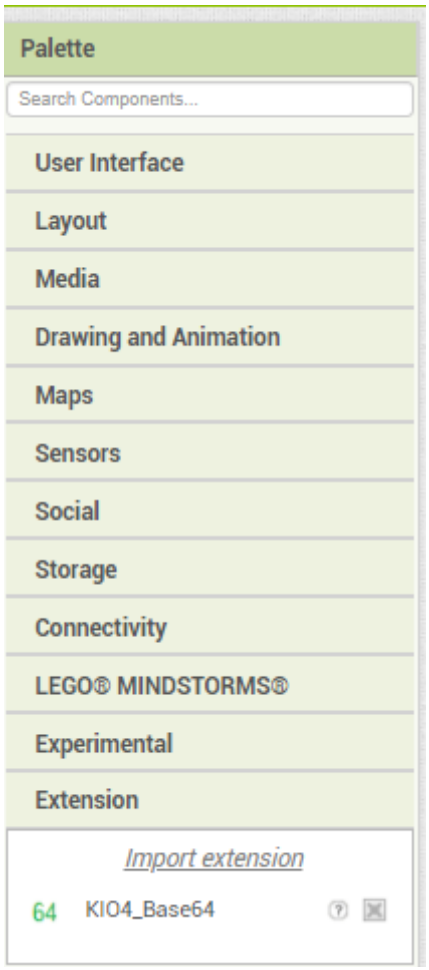
Στην Εικόνα 4.10 φαίνεται το περιεχόμενο του πίνακα alert με τις ειδοποιήσεις. Παρατηρούμε ότι η κάθε εικόνα αποθηκεύεται στο πεδίο imagebytes σε base64.

4.4 Η τελική εφαρμογή του χρήστη για την επίβλεψη και λήψη ειδοποιήσεων

Όπως περιεγράφηκε στο προηγούμενο υποκεφάλαιο οι ειδοποιήσεις (εικόνα και χρόνος) αποθηκεύονται στη βάση στο server.

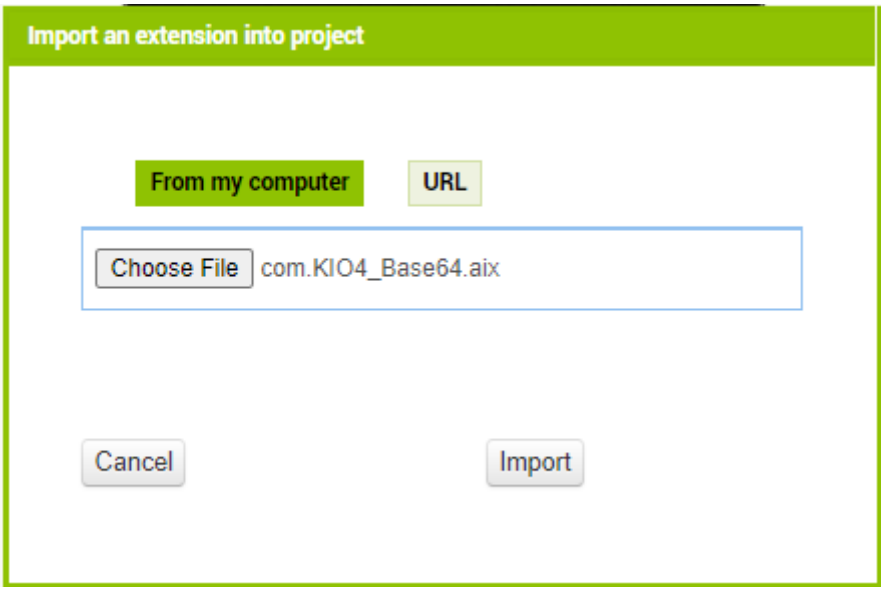
Δημιουργήθηκε ακόμα μια εφαρμογή για κινητό Android για τον χρήστη ώστε να επιβλέπει την ειδοποίηση όταν βρίσκεται μακριά από τον χώρο (δωμάτιο ξενοδοχείου ή κάτι άλλο) που τοποθέτησε το σύστημα επίβλεψης.

Η υλοποίηση πραγματοποιήθηκε με το App MIT Inventor για λόγους εκπαιδευτικούς.



Εικόνα 4.11: Υποστήριξη προσθήκης στο App Inventor μέσω της Palette

Για να μπορεί να υλοποιηθεί η μετατροπή της εικόνας από base64 χρησιμοποιήθηκε ένα ειδικό πρόσθετο το KIO4_Base64 όπως φαίνεται και στην Εικόνα 4.11.

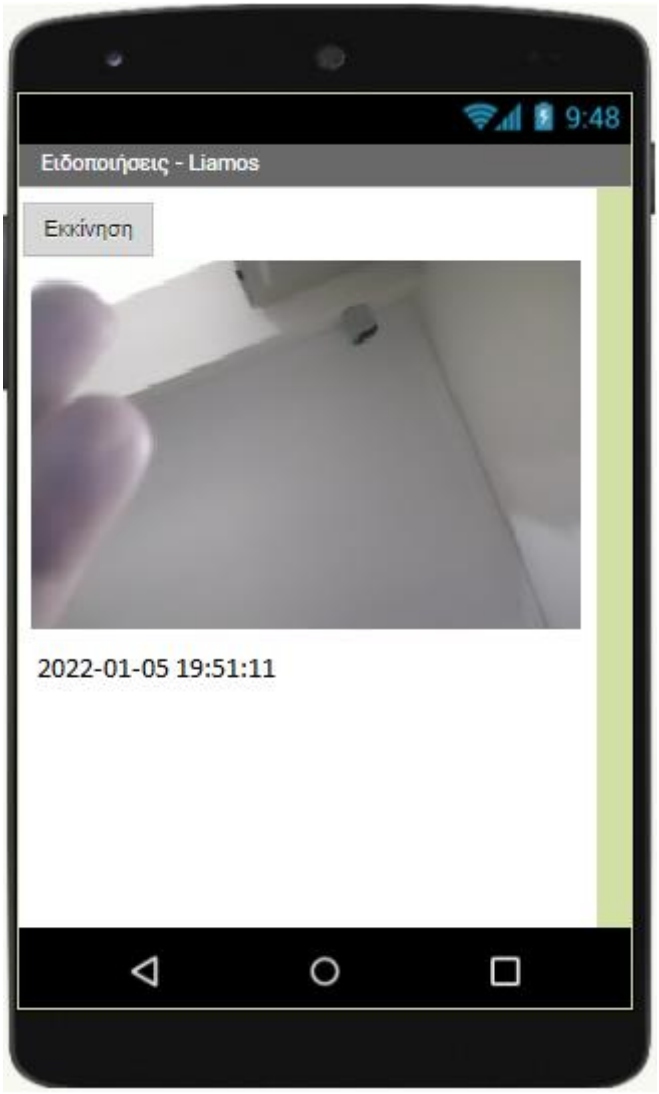


Εικόνα 4.12: Φόρτωση προσθήκης στο App Inventor μέσω της Palette

Εικόνα 4.13: Κεντρική οθόνη της εφαρμογής εμφάνισης ειδοποιήσεων

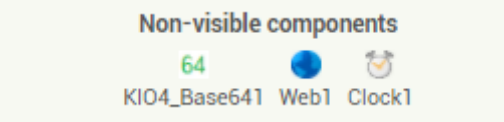
Στην Εικόνα 4.12 παρουσιάζεται το παράθυρο για τη φόρτωση προσθήκης στο App Inventor μέσω της Palette.

Στην Εικόνα 4.13 παρουσιάζεται η κεντρική οθόνη της εφαρμογής εμφάνισης ειδοποιήσεων. Διαθέτει κουμπί εκκίνησης και προβάλλεται η τελευταία ειδοποίηση (εικόνα και ημερομηνία) όπως φαίνεται στην Εικόνα 4.14.



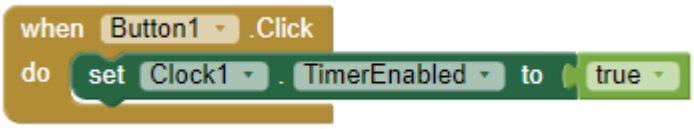
Εικόνα 4.14: Κεντρική οθόνη με προβολή ειδοποίησης

Προχωρώντας στην εξήγηση του προγραμματισμού και υλοποίησης της εφαρμογής να τονισθεί ότι ειδικές εντολές έχουν εισαχθεί πέρα από την προσθήκη για το base64, όπως το Web1 για την υποστήριξη επικοινωνίας με το διαδίκτυο (να καλέσει το PHP API και να λάβει την ειδοποίηση) και Clock1 για την προσθήκη Timer για την συνεχή λήψη των ειδοποιήσεων.

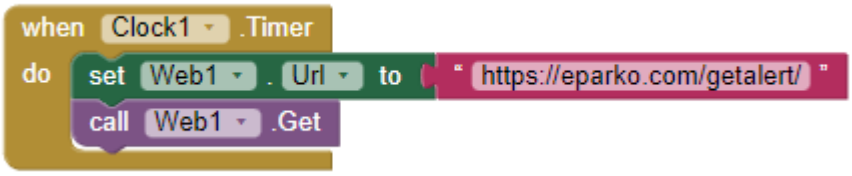


Εικόνα 4.15: Ειδικά στοιχεία για την εφαρμογή

Εικόνα 4.16: Αρχικοποίηση μεταβλητών

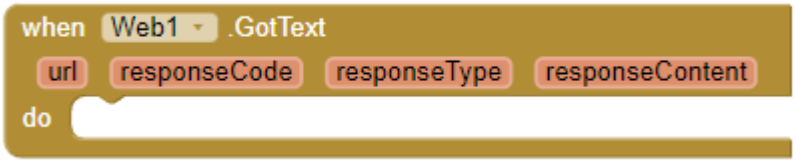


Εικόνα 4.17: Ενεργοποίηση του χρονισμού - Timer



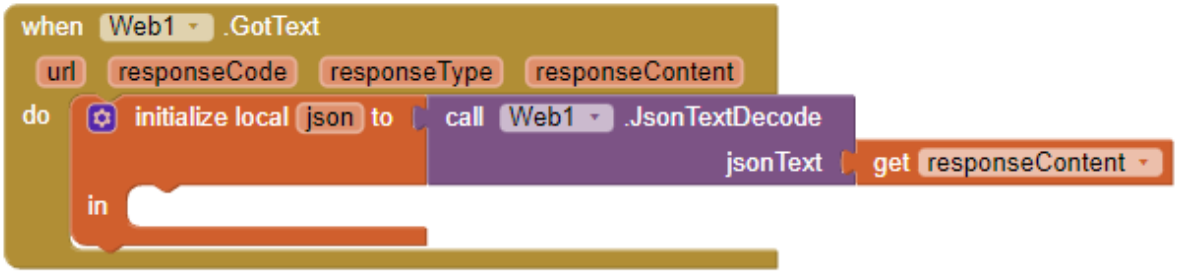
Εικόνα 4.18: Ενεργοποίηση του Web1 για σύνδεση-request της ειδοποίησης από το Server

Παρακάτω θα γίνει μια μικρή περιγραφή της διαδικασίας λήψης της απάντησης-Response μετά το Request που κάνει η εφαρμογή στο Server όπως φαίνεται από στις Εικόνες 4.16-4.18.



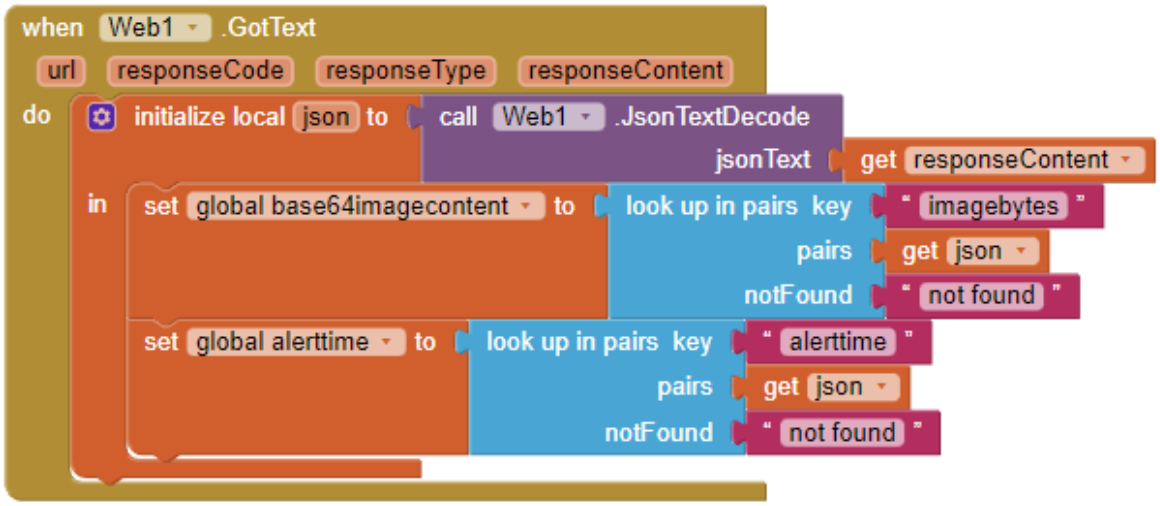
Εικόνα 4.19: Ειδικό block του Web1 τη λήψη απάντησης

Το Web έχει ειδικό block για να διαχειριστεί την λήψη της απάντησης από το Server, όπως φαίνεται στην Εικόνα 4.19.



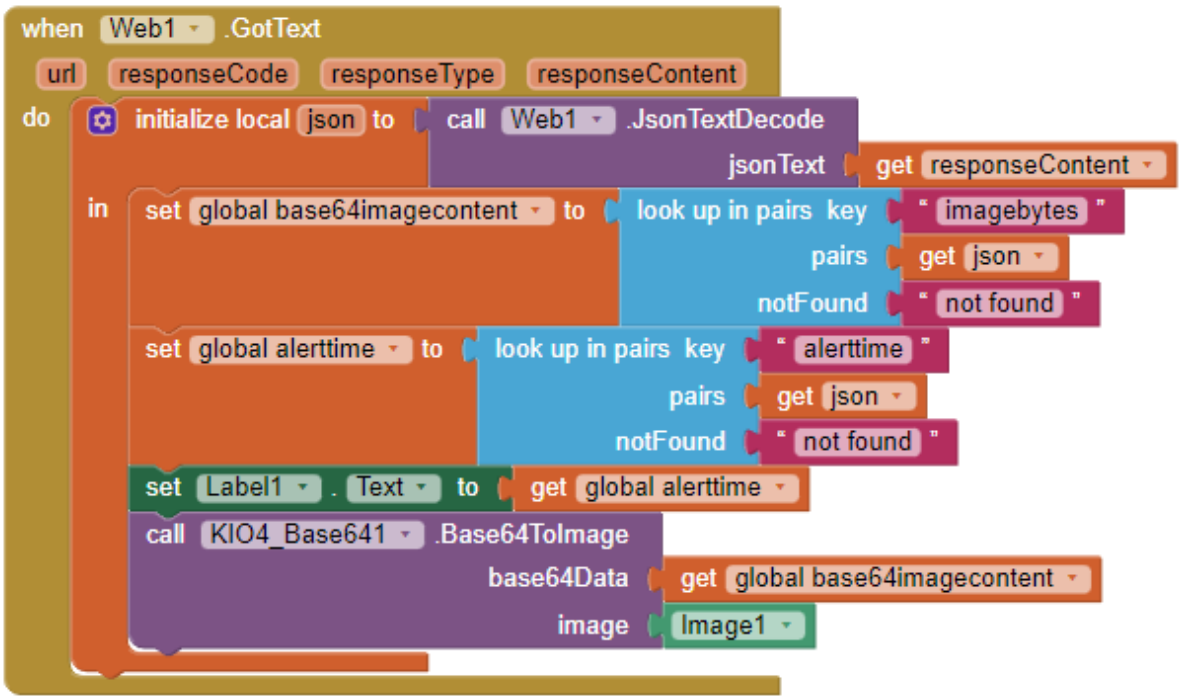
Εικόνα 4.20: Λήψη, μετατροπή και αποθήκευση της απάντησης

Σε αυτό το block λαμβάνεται το responseContent που είναι η απάντηση του server, μετατρέπεται σε κείμενο από json και τοποθετείται στην μεταβλητή json, όπως φαίνεται στην Εικόνα 4.20.



Εικόνα 4.21: Διαχωρισμός απάντησης σε εικόνα και ημερομηνία

Όπως φαίνεται στην Εικόνα 4.21 το περιεχόμενο της μεταβλητής json αναλύεται και διαχωρίζεται σε δύο μέρη, την εικόνα και την ημερομηνία.



Εικόνα 4.22: Διαχωρισμός απάντησης σε εικόνα και ημερομηνία

Στο πράσινο block set Label1.Text to … χρησιμοποιείται για να τοποθετηθεί η ημερομηνία ειδοποίησης.

Στο τελευταίο μωβ block χρησιμοποιείται το πρόσθετο KIO4 για την μετατροπή της base64 εικόνας (η οποία είναι σε String) σε εικόνα συμβατή με το App Inventor ώστε να γίνει αναπαράσταση στην οθόνη.

4.5 Ασφάλεια στο σύστημα παρακολούθησης, στη μετάδοση και στη λήψη

Ασφάλεια χρήσης της εφαρμογής στο κινητό που επιβλέπει τον χώρο

Αν κάποιος παραβιάσει τον χώρο μπορεί να κλέψει και το κινητό. Προτού γίνει αυτό θα αντιληφθεί την παρουσία του και θα στείλει ειδοποίηση. Η εφαρμογή έχει δημιουργηθεί με Flutter για Android κινητό. Για να λειτουργήσει σωστά η εφαρμογή πρέπει να έχει πρόσβαση στο διαδίκτυο, να επιβλέπει υπό σωστή γωνία τον χώρο, να είναι συνέχεια στην τροφοδοσία και σε περίπτωση νύχτας να έχει το δωμάτιο ένα μικρό φωτισμό ή να είναι ενεργοποιημένο το φλας του κινητού.

Ασφάλεια πρόσβασης στον PHP Server (API) και από τις δύο πλευρές

Στη πρώτη περίπτωση λαμβάνεται υπόψη η χρήση της βάσης του server από κινητά επίβλεψης από διάφορους χρήστες. Στη δεύτερη περίπτωση πρέπει να διαφυλαχτεί ότι κάποιος εισάγει ειδοποιήσεις στη βάση που είναι πιστοποιημένος. Για να ικανοποιηθούν και τα δύο τότε ορίζονται δύο μεταβλητές/πεδία, το πρώτο αφορά το device το οποίο είναι το μοναδικό id της συσκευής που δίνεται από τον δημιουργό. Το δεύτερο αφορά το token που στέλνεται μαζί με το κάθε request και συγκρίνεται στο server για να πιστοποιήσει την εγκυρότητα του request σε αυτό.

Η εφαρμογή που έχει ο χρήστης για να παρακολουθεί την ειδοποίηση δεν διαθέτει κάποια συγκεκριμένη ασφάλεια απλά ζητάει την τελευταία ειδοποίηση από την βάση που αφορούν τη συσκευή device.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Στην εργασία αυτή παρουσιάστηκε και κατασκευάστηκε ένα σύστημα παρακολούθησης χώρου ανιχνεύοντας κίνηση σε αυτόν και αποστέλλοντας ειδοποίηση στον χρήστη μέσω διαδικτύου στο κινητό του. Το κινητό τοποθετείται σε ένα χώρο επίβλεψης (όπως δωμάτιο) με την κάμερα να επιβλέπει το οπτικό της πεδίο. Συνήθως εστιάζει στην είσοδο του δωματίου αλλά θα μπορούσε να είναι και κάποια πόρτα μπαλκονιού ή παράθυρο. Η εφαρμογή στο κινητό παρακολουθεί με βίντεο το πεδίο και επεξεργάζεται την διαφορά μεταξύ δύο διαδοχικών frames για να ανιχνεύσει κίνηση στον χώρο.

Κατά την ανίχνευση κίνησης αποστέλλει την ειδοποίηση σε server. Βασική προϋπόθεση για την επικοινωνία της εφαρμογής του κινητού με τον χρήστη είναι η σύνδεση του με το διαδίκτυο μέσω του Wifi του ξενοδοχείου ή του δωματίου ή μέσω χρήση δεδομένων. Αφού η εφαρμογή μέσω της κάμερας ανιχνεύσει κίνηση στο πεδίο που εποπτεύει καταγράφει το γεγονός ως ειδοποίηση και αποστέλλει τη φωτογραφία στο server μαζί με την ειδοποίηση και αποθηκεύεται σε βάση δεδομένων. Ο server έχει υλοποιηθεί με PHP και Apache και η βάση δεδομένων είναι MySQL. Η φωτογραφία μετατρέπεται σε base64 για την αποστολή της φωτογραφίας σαν ένα απλό κείμενο από χαρακτήρες αλλά για την εύκολη αποθήκευση και ανάγνωση της στην/από βάση δεδομένων. Κατά την λήψη της ειδοποίησης από τη δεύτερη εφαρμογή του χρήστη η εικόνα αποκωδικοποιείται από το την base64 μορφή της.

Η ασφάλεια του συστήματος περιγράφθηκε στο προηγούμενο κεφάλαιο και μπορούμε να συμπεράνουμε ότι η χρήση του devide id και του token σε κάθε requests διασφαλίζει την αυθεντικοποίηση του χρήστη.

Όσον αφορά τις βελτιώσεις από τις πιο σημαντικές είναι η βελτίωση της ταχύτητας της επεξεργασίας των διαδοχικών εικόνων. Επίσης, η χρήση βελτιωμένων αλγορίθμων αναγνώρισης ανθρώπινης παρουσίας στο χώρο. Να μπορεί να γίνει εξουσιοδοτημένη χρήση των εφαρμογών μέσω login/logout και να γίνει καλύτερη διαχείριση των ανεξάρτητων συσκευών.

Τέλος, είναι αναγκαία η καλύτερη διαχείριση όλων των ειδοποιήσεων από την δεύτερη εφαρμογή, όπως η προβολή όλων των ειδοποιήσεων και η διαγραφή κάποιας.

Να προστεθεί ότι μια πού καλή εξέλιξη της εφαρμογής θα ήταν και μια πραγματικού χρόνου επίβλεψη μέσω βίντεο αλλά αυτό προϋποθέτει πολύ μεγάλη ταχύτητα διαδικτύου και μικρή ανάλυση στη εικόνα και μικρή ταχύτητα λήψης βίντεο (fps).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, A review of video surveillance systems, Journal of Visual Communication and Image Representation, Volume 77, 2021, 103116, ISSN 1047-3203, https://doi.org/10.1016/j.jvcir.2021.103116

[2] Vacharee Prashyanusorn, Paiboon Prashyanusorn, Somkuan Kaviya, Yusaku Fujii, Preecha P. Yupapin, The Use of Security Cameras with Privacy Protecting Ability, Procedia Engineering, Volume 8, 2011, Pages 301-307, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2011.03.056

[3] Khodadin, Fatimah and Sameerchand Pudaruth. "An Intelligent Camera Surveillance System with Effective Notification Features." *International Journal of Computing and Digital Systems* 10 (2020): 2-11.

[4] https://www.android.com/

[5] https://gs.statcounter.com/android-version-market-share/mobile/worldwide/

[6] https://gs.statcounter.com/os-market-share/mobile/worldwide

[7] https://flutter.dev/

[8] https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

[9] https://www.php.net/

[10] https://www.apachefriends.org/index.html

[11] https://appinventor.mit.edu/

ΠΑΡΑΡΤΗΜΑ Α: PHP κώδικας

Στο παράρτημα αυτό αναφέρονται τα βασικά κομμάτια του κώδικα που χρησιμοποιήθηκε.

PHP Server /index.php

```
<?php

$imagebytes = $_POST['imagebytes'];
$alerttime = $_POST['alerttime'];
$device = $_POST['device'];
$token = $_POST['token'];

//if ($alerttime == null) {
//    $alerttime = 0;
//}

if ($device == null) {
        http_response_code(400);
        return 0;
}
else
{
        if ($device != 1)
        http_response_code(400);
        return 0;
}

if ($token== null) {
        http_response_code(400);
        return 0;
}
else
{
        if ($token != "a3B5e")
        http_response_code(400);
        return 0;
}


if ($imagebytes == null) {
        http_response_code(400);
        return 0;
        //$imagebytes = '';
}

$servername = "localhost";
```

```
$username = "eparkouser";
$password = "eparkouserxxxxxx ";
$dbname = "eparkodb";


// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO alert (imagebytes, alerttime, device)
VALUES ('".$imagebytes."', LOCALTIME(), 1)";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
        http_response_code(400);
        //echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
```

PHP Server getalert/index.php

```
<?php

$servername = "localhost";
$username = "eparkouser";
$password = "eparkouserxxxxxx";
$dbname = "eparkodb";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM `alert` ORDER BY `id` DESC";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
        $row = $result->fetch_assoc();
    $arr = array('imagebytes' => $row["imagebytes"], 'alerttime' => $row["alerttime"]);
```

```
        echo json_encode($arr);
} else {
  return 0;
}

$conn->close();
```

ΠΑΡΑΡΤΗΜΑ Β: Flutter κώδικας

Στο παράρτημα αυτό αναφέρονται τα βασικά κομμάτια του κώδικα που χρησιμοποιήθηκε.

```
import 'dart:async';
import 'dart:io';
import 'dart:typed_data';
import 'dart:convert';

import 'package:camera/camera.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:video_player/video_player.dart';
import 'package:image_compare/image_compare.dart';
import 'package:http/http.dart' as http;

class CameraExampleHome extends StatefulWidget {
  @override
  _CameraExampleHomeState createState() {
    return _CameraExampleHomeState();
  }
}

/// Returns a suitable camera icon for [direction].
IconData getCameraLensIcon(CameraLensDirection direction) {
  switch (direction) {
```

```
    case CameraLensDirection.back:
      return Icons.camera_rear;
    case CameraLensDirection.front:
      return Icons.camera_front;
    case CameraLensDirection.external:
      return Icons.camera;
    default:
      throw ArgumentError('Unknown lens direction');
  }
}

void logError(String code, String? message) {
  if (message != null) {
    print('Error: $code\nError Message: $message');
  } else {
    print('Error: $code');
  }
}

class _CameraExampleHomeState extends State<CameraExampleHome>
    with WidgetsBindingObserver, TickerProviderStateMixin {
  CameraController? controller;
  XFile? imageFile;
  XFile? videoFile;
  VideoPlayerController? videoController;
  VoidCallback? videoPlayerListener;
  bool enableAudio = true;
  double _minAvailableExposureOffset = 0.0;
  double _maxAvailableExposureOffset = 0.0;
  double _currentExposureOffset = 0.0;
  late AnimationController _flashModeControlRowAnimationController;
  late Animation<double> _flashModeControlRowAnimation;
  late AnimationController _exposureModeControlRowAnimationController;
```

```
late Animation<double> _exposureModeControlRowAnimation;
late AnimationController _focusModeControlRowAnimationController;
late Animation<double> _focusModeControlRowAnimation;
double _minAvailableZoom = 1.0;
double _maxAvailableZoom = 1.0;
double _currentScale = 1.0;
double _baseScale = 1.0;
Color _iconAlertColor = Colors.green;
int startmonitor = 0;

// Counting pointers (number of user fingers on screen)
int _pointers = 0;

@override
void initState() {
  super.initState();
  _ambiguate(WidgetsBinding.instance)?.addObserver(this);

  _flashModeControlRowAnimationController = AnimationController(
    duration: const Duration(milliseconds: 300),
    vsync: this,
  );
  _flashModeControlRowAnimation = CurvedAnimation(
    parent: _flashModeControlRowAnimationController,
    curve: Curves.easeInCubic,
  );
  _exposureModeControlRowAnimationController = AnimationController(
    duration: const Duration(milliseconds: 300),
    vsync: this,
  );
  _exposureModeControlRowAnimation = CurvedAnimation(
    parent: _exposureModeControlRowAnimationController,
    curve: Curves.easeInCubic,
```

```
  );
  _focusModeControlRowAnimationController = AnimationController(
    duration: const Duration(milliseconds: 300),
    vsync: this,
  );
  _focusModeControlRowAnimation = CurvedAnimation(
    parent: _focusModeControlRowAnimationController,
    curve: Curves.easeInCubic,
  );
}

@override
void dispose() {
  _ambiguate(WidgetsBinding.instance)?.removeObserver(this);
  _flashModeControlRowAnimationController.dispose();
  _exposureModeControlRowAnimationController.dispose();
  super.dispose();
}

@override
void didChangeAppLifecycleState(AppLifecycleState state) {
  final CameraController? cameraController = controller;

  // App state changed before we got the chance to initialize.
  if (cameraController == null || !cameraController.value.isInitialized) {
    return;
  }

  if (state == AppLifecycleState.inactive) {
    cameraController.dispose();
  } else if (state == AppLifecycleState.resumed) {
    onNewCameraSelected(cameraController.description);
  }
```

```
}

final GlobalKey<ScaffoldState> _scaffoldKey = GlobalKey<ScaffoldState>();

@override
Widget build(BuildContext context) {
  return Scaffold(
    key: _scaffoldKey,
    appBar: AppBar(
      title: const Text('Alert System - Liamos'),
    ),
    body: Column(
      children: <Widget>[
        Expanded(
          child: Container(
            child: Padding(
              padding: const EdgeInsets.all(1.0),
              child: Center(
                child: _cameraPreviewWidget(),
              ),
            ),
            decoration: BoxDecoration(
              color: Colors.black,
              border: Border.all(
                color:
                    controller != null && controller!.value.isRecordingVideo
                        ? Colors.redAccent
                        : Colors.grey,
                width: 3.0,
              ),
            ),
          ),
        ),
```

```
        //_captureControlRowWidget(),
        _modeControlRowWidget(),
        Padding(
          padding: const EdgeInsets.all(5.0),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.start,
            children: <Widget>[
              _cameraTogglesRowWidget(),
              _thumbnailWidget(),
            ],
          ),
        ),
      ],
    ),
  );
}

/// Display the preview from the camera (or a message if the preview is not available).
Widget _cameraPreviewWidget() {
  final CameraController? cameraController = controller;

  if (cameraController == null || !cameraController.value.isInitialized) {
    return const Text(
      'Διάλεξε Κάμερα',
      style: TextStyle(
        color: Colors.white,
        fontSize: 24.0,
        fontWeight: FontWeight.w900,
      ),
    );
  } else {
    return Listener(
      onPointerDown: (_) => _pointers++,
```

```
      onPointerUp: (_) => _pointers--,
      child: CameraPreview(
        controller!,
        child: LayoutBuilder(
            builder: (BuildContext context, BoxConstraints constraints) {
          return GestureDetector(
            behavior: HitTestBehavior.opaque,
            onScaleStart: _handleScaleStart,
            onScaleUpdate: _handleScaleUpdate,
            onTapDown: (details) => onViewFinderTap(details, constraints),
          );
        }),
      ),
    );
  }
}

void _handleScaleStart(ScaleStartDetails details) {
  _baseScale = _currentScale;
}

Future<void> _handleScaleUpdate(ScaleUpdateDetails details) async {
  // When there are not exactly two fingers on screen don't scale
  if (controller == null || _pointers != 2) {
    return;
  }

  _currentScale = (_baseScale * details.scale)
      .clamp(_minAvailableZoom, _maxAvailableZoom);

  await controller!.setZoomLevel(_currentScale);
}
```

```
/// Display the thumbnail of the captured image or video.
Widget _thumbnailWidget() {
  final VideoPlayerController? localVideoController = videoController;

  return Expanded(
    child: Align(
      alignment: Alignment.centerRight,
      child: Row(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          localVideoController == null && imageFile == null
              ? Container()
              : SizedBox(
                  child: (localVideoController == null)
                      ? (
                          // The captured image on the web contains a network-accessible URL
                          // pointing to a location within the browser. It may be displayed
                          // either with Image.network or Image.memory after loading the image
                          // bytes to memory.
                          kIsWeb
                              ? Image.network(imageFile!.path)
                              : Image.file(File(imageFile!.path)))
                      : Container(
                          child: Center(
                            child: AspectRatio(
                                aspectRatio:
                                    localVideoController.value.size != null
                                        ? localVideoController
                                            .value.aspectRatio
                                        : 1.0,
                                child: VideoPlayer(localVideoController)),
                          ),
                          decoration: BoxDecoration(
```

```
                        border: Border.all(color: Colors.pink)),
                    ),
                width: 64.0,
                height: 64.0,
              ),
        ],
      ),
    ),
  );
}

/// Display a bar with buttons to change the flash and exposure modes
Widget _modeControlRowWidget() {
  return Column(
    children: [
      Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        mainAxisSize: MainAxisSize.max,
        children: <Widget>[
          IconButton(
            icon: Icon(Icons.flash_off),
            color: Colors.blue,
            onPressed: controller != null ? onFlashModeButtonPressed : null,
          ),
          // The exposure and focus mode are currently not supported on the web.
          ...(!kIsWeb
              ? [

                ]
              : []),
          IconButton(
            icon: Icon(Icons.play_circle),
            color: Colors.blue,
```

```
          onPressed:
              controller != null ? onStartMonitorButtonPressed : null,
          ),
          IconButton(
            icon: Icon(Icons.stop),
            color: Colors.blue,
            onPressed: controller != null ? onStopMonitorButtonPressed : null,
          ),
          IconButton(
            icon: Icon(Icons.timelapse_rounded),
            color: Colors.blue,
            onPressed: controller != null
                ? onStarMonitortWithDelayButtonPressed
                : null,
          )
        ],
      ),
      _flashModeControlRowWidget(),
      //_exposureModeControlRowWidget(),
      //_focusModeControlRowWidget(),
    ],
  );
}

Widget _flashModeControlRowWidget() {
  return SizeTransition(
    sizeFactor: _flashModeControlRowAnimation,
    child: ClipRect(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        mainAxisSize: MainAxisSize.max,
        children: [
          IconButton(
```

```
  icon: Icon(Icons.flash_off),
  color: controller?.value.flashMode == FlashMode.off
      ? Colors.orange
      : Colors.blue,
  onPressed: controller != null
      ? () => onSetFlashModeButtonPressed(FlashMode.off)
      : null,
),
IconButton(
  icon: Icon(Icons.flash_auto),
  color: controller?.value.flashMode == FlashMode.auto
      ? Colors.orange
      : Colors.blue,
  onPressed: controller != null
      ? () => onSetFlashModeButtonPressed(FlashMode.auto)
      : null,
),
IconButton(
  icon: Icon(Icons.flash_on),
  color: controller?.value.flashMode == FlashMode.always
      ? Colors.orange
      : Colors.blue,
  onPressed: controller != null
      ? () => onSetFlashModeButtonPressed(FlashMode.always)
      : null,
),
IconButton(
  icon: Icon(Icons.highlight),
  color: controller?.value.flashMode == FlashMode.torch
      ? Colors.orange
      : Colors.blue,
  onPressed: controller != null
      ? () => onSetFlashModeButtonPressed(FlashMode.torch)
```

```
                : null,
          ),
        ],
      ),
    ),
  );
}

Widget _exposureModeControlRowWidget() {
  final ButtonStyle styleAuto = TextButton.styleFrom(
    primary: controller?.value.exposureMode == ExposureMode.auto
        ? Colors.orange
        : Colors.blue,
  );
  final ButtonStyle styleLocked = TextButton.styleFrom(
    primary: controller?.value.exposureMode == ExposureMode.locked
        ? Colors.orange
        : Colors.blue,
  );

  return SizeTransition(
    sizeFactor: _exposureModeControlRowAnimation,
    child: ClipRect(
      child: Container(
        color: Colors.grey.shade50,
        child: Column(
          children: [
            Center(
              child: Text("Exposure Mode"),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              mainAxisSize: MainAxisSize.max,
```

```
  children: [
    TextButton(
      child: Text('AUTO'),
      style: styleAuto,
      onPressed: controller != null
          ? () =>
              onSetExposureModeButtonPressed(ExposureMode.auto)
          : null,
      onLongPress: () {
        if (controller != null) {
          controller!.setExposurePoint(null);
          showInSnackBar('Resetting exposure point');
        }
      },
    ),
    TextButton(
      child: Text('LOCKED'),
      style: styleLocked,
      onPressed: controller != null
          ? () =>
              onSetExposureModeButtonPressed(ExposureMode.locked)
          : null,
    ),
    TextButton(
      child: Text('RESET OFFSET'),
      style: styleLocked,
      onPressed: controller != null
          ? () => controller!.setExposureOffset(0.0)
          : null,
    ),
  ],
),
Center(
```

```
            child: Text("Exposure Offset"),
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            mainAxisSize: MainAxisSize.max,
            children: [
              Text(_minAvailableExposureOffset.toString()),
              Slider(
                value: _currentExposureOffset,
                min: _minAvailableExposureOffset,
                max: _maxAvailableExposureOffset,
                label: _currentExposureOffset.toString(),
                onChanged: _minAvailableExposureOffset ==
                        _maxAvailableExposureOffset
                    ? null
                    : setExposureOffset,
              ),
              Text(_maxAvailableExposureOffset.toString()),
            ],
          ),
        ],
      ),
    ),
  );
}

Widget _focusModeControlRowWidget() {
  final ButtonStyle styleAuto = TextButton.styleFrom(
    primary: controller?.value.focusMode == FocusMode.auto
        ? Colors.orange
        : Colors.blue,
  );
```

```
final ButtonStyle styleLocked = TextButton.styleFrom(
  primary: controller?.value.focusMode == FocusMode.locked
      ? Colors.orange
      : Colors.blue,
);

return SizeTransition(
  sizeFactor: _focusModeControlRowAnimation,
  child: ClipRect(
    child: Container(
      color: Colors.grey.shade50,
      child: Column(
        children: [
          Center(
            child: Text("Focus Mode"),
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            mainAxisSize: MainAxisSize.max,
            children: [
              TextButton(
                child: Text('AUTO'),
                style: styleAuto,
                onPressed: controller != null
                    ? () => onSetFocusModeButtonPressed(FocusMode.auto)
                    : null,
                onLongPress: () {
                  if (controller != null) controller!.setFocusPoint(null);
                  showInSnackBar('Resetting focus point');
                },
              ),
              TextButton(
                child: Text('LOCKED'),
```

```
                style: styleLocked,
                onPressed: controller != null
                    ? () => onSetFocusModeButtonPressed(FocusMode.locked)
                    : null,
              ),
            ],
          ),
        ],
      ),
    ),
  );
}

/// Display the control bar with buttons to take pictures and record videos.
Widget _captureControlRowWidget() {
  final CameraController? cameraController = controller;

  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      IconButton(
        icon: const Icon(Icons.camera_alt),
        color: Colors.blue,
        onPressed: cameraController != null &&
              cameraController.value.isInitialized &&
              !cameraController.value.isRecordingVideo
            ? onTakePictureButtonPressed
            : null,
      ),
      IconButton(
        icon: const Icon(Icons.videocam),
```

```
  color: Colors.blue,
  onPressed: cameraController != null &&
        cameraController.value.isInitialized &&
        !cameraController.value.isRecordingVideo
      ? onVideoRecordButtonPressed
      : null,
),
IconButton(
  icon: cameraController != null &&
        cameraController.value.isRecordingPaused
      ? Icon(Icons.play_arrow)
      : Icon(Icons.pause),
  color: Colors.blue,
  onPressed: cameraController != null &&
        cameraController.value.isInitialized &&
        cameraController.value.isRecordingVideo
      ? (cameraController.value.isRecordingPaused)
        ? onResumeButtonPressed
        : onPauseButtonPressed
      : null,
),
IconButton(
  icon: const Icon(Icons.stop),
  color: Colors.red,
  onPressed: cameraController != null &&
        cameraController.value.isInitialized &&
        cameraController.value.isRecordingVideo
      ? onStopButtonPressed
      : null,
),
IconButton(
  icon: const Icon(Icons.pause_presentation),
  color:
```

```
          cameraController != null && cameraController.value.isPreviewPaused
              ? Colors.red
              : Colors.blue,
        onPressed:
          cameraController == null ? null : onPausePreviewButtonPressed,
      ),
    ],
  );
}

/// Display a row of toggle to select the camera (or a message if no camera is available).
Widget _cameraTogglesRowWidget() {
  final List<Widget> toggles = <Widget>[];

  final onChanged = (CameraDescription? description) {
    if (description == null) {
      return;
    }

    onNewCameraSelected(description);
  };

  if (cameras.isEmpty) {
    return const Text('No camera found');
  } else {
    for (CameraDescription cameraDescription in cameras) {
      toggles.add(
        SizedBox(
          width: 90.0,
          child: RadioListTile<CameraDescription>(
            title: Icon(getCameraLensIcon(cameraDescription.lensDirection)),
            groupValue: controller?.description,
            value: cameraDescription,
```

```
            onChanged:
                controller != null && controller!.value.isRecordingVideo
                    ? null
                    : onChanged,
          ),
        ),
      );
    }
  }

  return Row(children: toggles);
}

//String timestamp() => DateTime.now().millisecondsSinceEpoch.toString();

void showInSnackBar(String message) {
  // ignore: deprecated_member_use
  _scaffoldKey.currentState?.showSnackBar(SnackBar(content: Text(message)));
}

void onViewFinderTap(TapDownDetails details, BoxConstraints constraints) {
  if (controller == null) {
    return;
  }

  final CameraController cameraController = controller!;

  final offset = Offset(
    details.localPosition.dx / constraints.maxWidth,
    details.localPosition.dy / constraints.maxHeight,
  );
  cameraController.setExposurePoint(offset);
  cameraController.setFocusPoint(offset);
```

```
}

void onStartMonitorButtonPressed() {
  print("onStartMonitorButtonPressed");
  startmonitor = 1;
  if (mounted)
    setState(() {
      startmonitor = startmonitor;
    });
}

void onStopMonitorButtonPressed() {
  startmonitor = 0;
  if (mounted)
    setState(() {
      startmonitor = startmonitor;
    });
}

void onStarMonitortWithDelayButtonPressed() {
  Timer(Duration(seconds: 10), () {
    startmonitor = 1;
    if (mounted)
      setState(() {
        startmonitor = startmonitor;
      });
  });
}

int countdifs = 0;
Uint8List? bytesofimage1, bytesofimage2;
int pio = 0;
```

```
//MAIN
void onNewCameraSelected(CameraDescription cameraDescription) async {
 if (controller != null) {
  await controller!.dispose();
 }

 onSetFlashModeButtonPressed(FlashMode.off);
 setFlashMode(FlashMode.off);

 final CameraController cameraController = CameraController(
  cameraDescription,
  //kIsWeb ? ResolutionPreset.max : ResolutionPreset.medium,
  kIsWeb ? ResolutionPreset.medium : ResolutionPreset.medium,
  enableAudio: enableAudio,
  imageFormatGroup: ImageFormatGroup.jpeg,
 );

 controller = cameraController;

 // If the controller is updated then update the UI.
 cameraController.addListener(() {
  if (mounted) setState(() {});
  if (cameraController.value.hasError) {
   showInSnackBar(
       'Camera error ${cameraController.value.errorDescription}');
  }
 });

 try {
  await cameraController.initialize();
  await Future.wait([
   // The exposure mode is currently not supported on the web.
   ...(!kIsWeb
```

```
        ? [
            cameraController
                .getMinExposureOffset()
                .then((value) => _minAvailableExposureOffset = value),
            cameraController
                .getMaxExposureOffset()
                .then((value) => _maxAvailableExposureOffset = value)
          ]
        : []),
    cameraController
        .getMaxZoomLevel()
        .then((value) => _maxAvailableZoom = value),
    cameraController
        .getMinZoomLevel()
        .then((value) => _minAvailableZoom = value),
  ]);
} on CameraException catch (e) {
  _showCameraException(e);
}

if (mounted) {
  setState(() {
    _iconAlertColor = _iconAlertColor;
    startmonitor = startmonitor;
  });
}

String imagebytes = base64.encode(utf8.encode('bitakia eikonas'));
var alerttime = 2;
var device = 1;

cameraController.startImageStream((CameraImage img) {
  print("startImageStream");
```

```
print(startmonitor);
if (startmonitor == 1) {
  print("startImageStream 1");

  if (pio == 2) {
    bytesofimage2 = img.planes[0].bytes;
    pio = 1;
  } else {
    if (pio != 0) {
      bytesofimage1 = img.planes[0].bytes;
      pio = 2;
    }
  }

  if (pio == 0) {
    bytesofimage1 = img.planes[0].bytes;
    bytesofimage2 = img.planes[0].bytes;
    pio = 1;
  }

  var client = http.Client();
  compareImages(
        src1: bytesofimage1,
        src2: bytesofimage2,
        algorithm: IMED(blurRatio: 0.001))
      .then((value) => {
          print('Difference1: ${value * 100}%'),
          if (value * 100 > 0.2)
            {
              print('------------------------------: ${value * 100}%'),


                client.post(Uri.parse('https://enasite.com'),
```

```
                    headers: {},
                    body: {
                      'imagebytes': base64Encode(img.planes[0].bytes),
                      'alerttime': alerttime.toString(),
                                                          'token': 'a3B5e'
                      'device': device.toString()
                    }),
                }
            });
    }
  });
}

void onTakePictureButtonPressed() {
  takePicture().then((XFile? file) {
    if (mounted) {
      setState(() {
        imageFile = file;
        videoController?.dispose();
        videoController = null;
      });
      if (file != null) showInSnackBar('Picture saved to ${file.path}');
    }
  });
}

void onFlashModeButtonPressed() {
  if (_flashModeControlRowAnimationController.value == 1) {
    _flashModeControlRowAnimationController.reverse();
  } else {
    _flashModeControlRowAnimationController.forward();
    _exposureModeControlRowAnimationController.reverse();
    _focusModeControlRowAnimationController.reverse();
```

```
  }
 }

 void onExposureModeButtonPressed() {
  if (_exposureModeControlRowAnimationController.value == 1) {
   _exposureModeControlRowAnimationController.reverse();
  } else {
   _exposureModeControlRowAnimationController.forward();
   _flashModeControlRowAnimationController.reverse();
   _focusModeControlRowAnimationController.reverse();
  }
 }

 void onAudioModeButtonPressed() {
  enableAudio = !enableAudio;
  if (controller != null) {
   onNewCameraSelected(controller!.description);
  }
 }

 void onCaptureOrientationLockButtonPressed() async {
  try {
   if (controller != null) {
    final CameraController cameraController = controller!;
    if (cameraController.value.isCaptureOrientationLocked) {
     await cameraController.unlockCaptureOrientation();
     showInSnackBar('Capture orientation unlocked');
    } else {
     await cameraController.lockCaptureOrientation();
     showInSnackBar(
       'Capture orientation locked to ${cameraController.value.lockedCaptureOrientation.toString().split('.').last}');
    }
```

```
    }
  } on CameraException catch (e) {
    _showCameraException(e);
  }
}

void onSetFlashModeButtonPressed(FlashMode mode) {
  setFlashMode(mode).then((_) {
    if (mounted) setState(() {});
    showInSnackBar('Flash mode set to ${mode.toString().split('.').last}');
  });
}

void onSetExposureModeButtonPressed(ExposureMode mode) {
  setExposureMode(mode).then((_) {
    if (mounted) setState(() {});
    showInSnackBar('Exposure mode set to ${mode.toString().split('.').last}');
  });
}

void onSetFocusModeButtonPressed(FocusMode mode) {
  setFocusMode(mode).then((_) {
    if (mounted) setState(() {});
    showInSnackBar('Focus mode set to ${mode.toString().split('.').last}');
  });
}

void onVideoRecordButtonPressed() {
  startVideoRecording().then((_) {
    if (mounted) setState(() {});
  });
}
```

```
void onStopButtonPressed() {
  stopVideoRecording().then((file) {
    if (mounted) setState(() {});
    if (file != null) {
      showInSnackBar('Video recorded to ${file.path}');
      videoFile = file;
      _startVideoPlayer();
    }
  });
}

Future<void> onPausePreviewButtonPressed() async {
  final CameraController? cameraController = controller;

  if (cameraController == null || !cameraController.value.isInitialized) {
    showInSnackBar('Error: select a camera first.');
    return;
  }

  if (cameraController.value.isPreviewPaused) {
    await cameraController.resumePreview();
  } else {
    await cameraController.pausePreview();
  }

  if (mounted) setState(() {});
}

void onPauseButtonPressed() {
  pauseVideoRecording().then((_) {
    if (mounted) setState(() {});
    showInSnackBar('Video recording paused');
  });
```

```
}

void onResumeButtonPressed() {
  resumeVideoRecording().then((_) {
    if (mounted) setState(() {});
    showInSnackBar('Video recording resumed');
  });
}

Future<void> startVideoRecording() async {
  final CameraController? cameraController = controller;

  if (cameraController == null || !cameraController.value.isInitialized) {
    showInSnackBar('Error: select a camera first.');
    return;
  }

  if (cameraController.value.isRecordingVideo) {
    // A recording is already started, do nothing.
    return;
  }

  try {
    await cameraController.startVideoRecording();
  } on CameraException catch (e) {
    _showCameraException(e);
    return;
  }
}

Future<XFile?> stopVideoRecording() async {
  final CameraController? cameraController = controller;
```

```
  if (cameraController == null || !cameraController.value.isRecordingVideo) {
    return null;
  }

  try {
    return cameraController.stopVideoRecording();
  } on CameraException catch (e) {
    _showCameraException(e);
    return null;
  }
}

Future<void> pauseVideoRecording() async {
  final CameraController? cameraController = controller;

  if (cameraController == null || !cameraController.value.isRecordingVideo) {
    return null;
  }

  try {
    await cameraController.pauseVideoRecording();
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
}

Future<void> resumeVideoRecording() async {
  final CameraController? cameraController = controller;

  if (cameraController == null || !cameraController.value.isRecordingVideo) {
    return null;
  }
```

```
  try {
    await cameraController.resumeVideoRecording();
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
}

Future<void> setFlashMode(FlashMode mode) async {
  if (controller == null) {
    return;
  }

  try {
    await controller!.setFlashMode(mode);
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
}

Future<void> setExposureMode(ExposureMode mode) async {
  if (controller == null) {
    return;
  }

  try {
    await controller!.setExposureMode(mode);
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
```

```
}

Future<void> setExposureOffset(double offset) async {
  if (controller == null) {
    return;
  }

  setState(() {
    _currentExposureOffset = offset;
  });
  try {
    offset = await controller!.setExposureOffset(offset);
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
}

Future<void> setFocusMode(FocusMode mode) async {
  if (controller == null) {
    return;
  }

  try {
    await controller!.setFocusMode(mode);
  } on CameraException catch (e) {
    _showCameraException(e);
    rethrow;
  }
}

Future<void> _startVideoPlayer() async {
  if (videoFile == null) {
```

```
    return;
  }

  final VideoPlayerController vController = kIsWeb
      ? VideoPlayerController.network(videoFile!.path)
      : VideoPlayerController.file(File(videoFile!.path));

  videoPlayerListener = () {
    if (videoController != null && videoController!.value.size != null) {
      // Refreshing the state to update video player with the correct ratio.
      if (mounted) setState(() {});
      videoController!.removeListener(videoPlayerListener!);
    }
  };
  vController.addListener(videoPlayerListener!);
  await vController.setLooping(true);
  await vController.initialize();
  await videoController?.dispose();
  if (mounted) {
    setState(() {
      imageFile = null;
      videoController = vController;
    });
  }
  await vController.play();
}

Future<XFile?> takePicture() async {
  final CameraController? cameraController = controller;
  if (cameraController == null || !cameraController.value.isInitialized) {
    showInSnackBar('Error: select a camera first.');
    return null;
  }
```

```
  }

  void _showCameraException(CameraException e) {
    logError(e.code, e.description);
    showInSnackBar('Error: ${e.code}\n${e.description}');
  }
}

class CameraApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: CameraExampleHome(),
    );
  }
}

List<CameraDescription> cameras = [];

Future<void> main() async {
  // Fetch the available cameras before initializing the app.
  try {
    WidgetsFlutterBinding.ensureInitialized();
    cameras = await availableCameras();
  } on CameraException catch (e) {
    logError(e.code, e.description);
  }
  runApp(CameraApp());
}

T? _ambiguate<T>(T? value) => value;
```