Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ένας τοπικός Android Proxy για καταγραφή, επιτήρηση και διαχείριση των διαδικτυακών διασυνδέσεων** **A Local Android Proxy for Monitoring and Policying Internet Connections**
Ονοματεπώνυμο Φοιτητή	**Θεοφάνης Κούβαρης**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός μητρώου	**ΜΠΣΠ14040**
Καθηγητής	**Αλέπης Ευθύμιος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος - 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Επίκ. Καθηγητής	Κωνσταντίνος Πατσάκης Επίκ. Καθηγητής	Όνομα Επώνυμο Βαθμίδα

1 Περίληψη (Abstract)

Στην παρακάτω εργασία θα δείτε πώς ο proxy Operando μπορεί να επεκταθεί για να καταγράφει τις κινήσεις του δικτύου ενός κινητού Android, να βοηθά στην κατηγοριοποίησή τους, στη μελέτη τους, να τις δημοσιεύει σε server ώστε να επαναχρησιμοποιηθούν και να ανακαλύπτει εφαρμογές που τρέχουν σε πραγματικό χρόνο με βάση τα URL που χρησιμοποιούνται από αυτές.

Στην εισαγωγή θα βρείτε μια σύντομη περιγραφή του τι κάνει η εφαρμογή και ποιες τεχνολογίες χρησιμοποιήθηκαν. Το κύριο μέρος της εργασίας αποτελείται από τρεις ενότητες. Η πρώτη αφορά θεωρητικά θέματα που μας απασχόλησαν κατά την διάρκειά της, όπως σε ποια android λειτουργικά είναι εφικτή η χρησιμοποίηση της εφαρμογής και πώς θα μπορούσαμε να βρούμε αν μια εφαρμογή τρέχει στο κινητό από τα δεδομένα που καταγράφονται από τον proxy. Η δεύτερη ενότητα αφορά το πώς όλα αυτά υλοποιήθηκαν από τεχνικής άποψης. Αναλυτικότερα, σε αυτή την ενότητα καταγράψαμε το σχήμα της βάσης, τη διαδικασία της εγκατάστασης, τις κλάσεις, τον web server, την επικοινωνία με τον server και τέλος τις νέες οθόνες και πώς αυτές χρησιμοποιούνται. Η τρίτη ενότητα αφορά τα συμπεράσματα που βγάλαμε για το ποια είναι η αξία της εργασίας, τι θα μπορούσε να γίνει σαν επόμενο βήμα και το αν έτρεξε σαν project η εργασία. Κλείνοντας την εργασία υπάρχει ένας μικρός επίλογος, απαραίτητα παραρτήματα και οι πηγές που μας βοήθησαν.

In the following thesis you will see how the Operando proxy can be extended to capture the web calls that an Android mobile sent on the network and help us to categorize them, study them, publish them on a server and how this can help as to discover real-time applications based on URLs used by the applications.

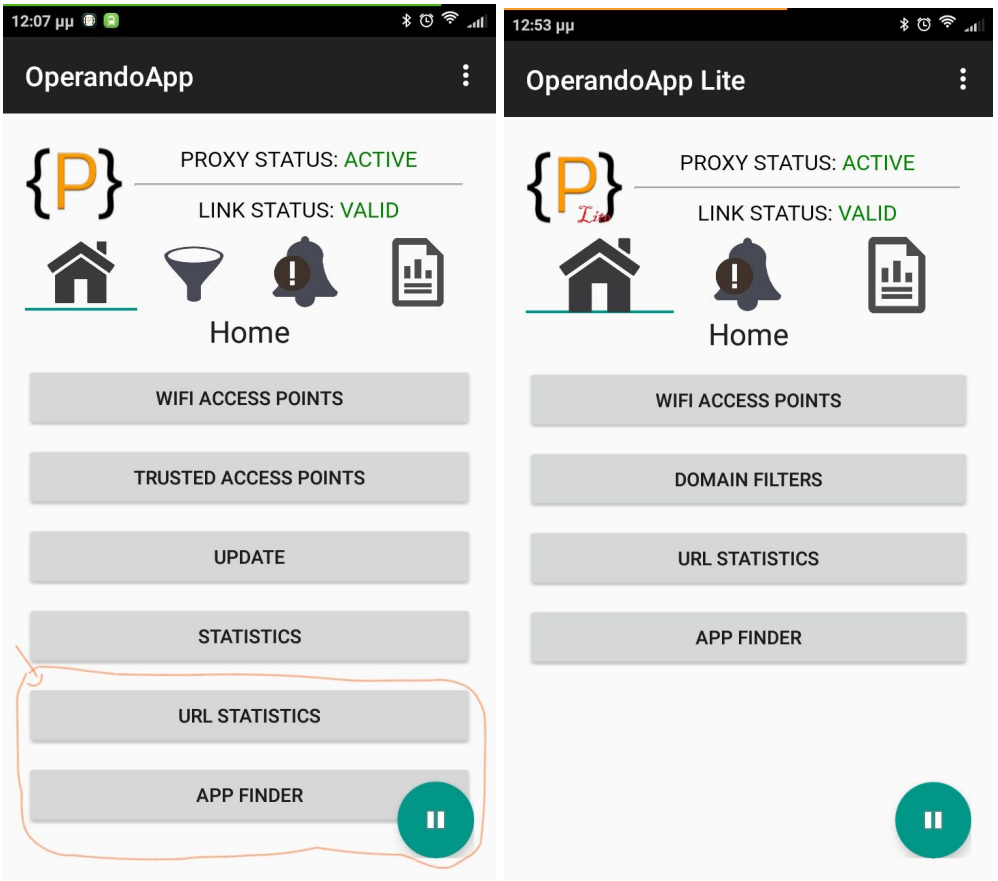
In the introduction you will find a brief description of what the application does and what technologies have been used. The main part of the thesis consists of three sections. The first concerns the theoretical issues that concerned us during the course of the thesis, such as on which android is operationally feasible to use the application and how we could find out if an application runs on the android from the data recorded by the proxy. The second section concerns the technical implementation. In more details, in this section we have recorded the database scheme, the installation process, the classes, the web server, the communication with the server and finally the new screens and how they are used. The third section concerns the conclusions we have drawn about what is the value of the thesis, what could be done as a next step and whether we ran thesis as a project. Closing the thesis, there is a small epilogue, necessary annexes and the sources that helped us.

2 Περιεχόμενα

3 Εισαγωγή

3.1 Περίληψη δυνατοτήτων εφαρμογής

Η εφαρμογή, που μας δόθηκε, η Operando μπορούσε ήδη να κάνει proxy όλες τις κινήσεις του δικτύου του κινητού και να κάνει block κάποιο domain. Έχοντας γραφτεί όμως σε παλιότερες εκδόσεις του android και μη κρατώντας σε βάση στατιστικά για των κινήσεων του διαδικτύου, άφηνε ανεξερεύνητο μέρος των δυνατοτήτων της. Έτσι σε αυτήν την εργασία ο proxy μεταφέρθηκε στην τελευταία έκδοση του Android Studio και στο τελευταίο gradle εμπλουτίστηκε με νέες οθόνες και λειτουργίες. Η εφαρμογή πια καταγράφει πλήρως τις κινήσεις σε δίκτυα, τις παρουσιάζει σε οθόνες και βοηθά στην κατηγοριοποίηση τους .Μπορεί επίσης να τις στείλει σε διασυνδεδεμένα συστήματα για περαιτέρω στατιστική αξιοποίηση και να μαζέψει πολιτικές από αυτά. Συνάμα επεκτάθηκε η δυνατότητα της εφαρμογής στο να φιλτράρει τις κινήσεις στο δίκτυο αφού πια μπορεί να μπλοκάρει κινήσεις με βάση το ιστορικό και να δημιουργεί alerts. Τέλος ανιχνεύει ενεργές εφαρμογές στο κινητό με ακρίβεια, με βάση το αποτύπωμα τους στο διαδίκτυο.



3.2 Τεχνολογίες που χρησιμοποιήθηκαν

Λόγο του ότι η εργασία είχε πολλά πεδία εφαρμογής και γίνεται κυρίως για ερευνητικούς λόγους χρησιμοποιήσαμε πολλά νέα framework και τεχνολογίες.

- **Gradle** είναι ένα custom build tool, και χρησιμοποιείτε για την δημιουργία του apk αρχείου της εφαρμογής μας.
- **Volley** είναι μια βιβλιοθήκη HTTP που κάνει ευκολότερη τη δικτύωση για εφαρμογές Android και, το πιο σημαντικό, πιο γρήγορη.

- **Node.js** είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον Javascript.
- **Swagger** είναι το μεγαλύτερο εργαλείο ανάπτυξης API στον κόσμο
- **Termux** ισχυρός εξομοιωτής τερματικού που συνδυάζεται με μια εκτεταμένη συλλογή από Linux πακέτα.
- **Heroku** είναι μια πλατφόρμα Cloud που επιτρέπει να χτίζουμε, να παραδίδουμε, να παρακολουθούμε και να μεγαλώνουμε τις εφαρμογές μας αυτοματοποιημένα.

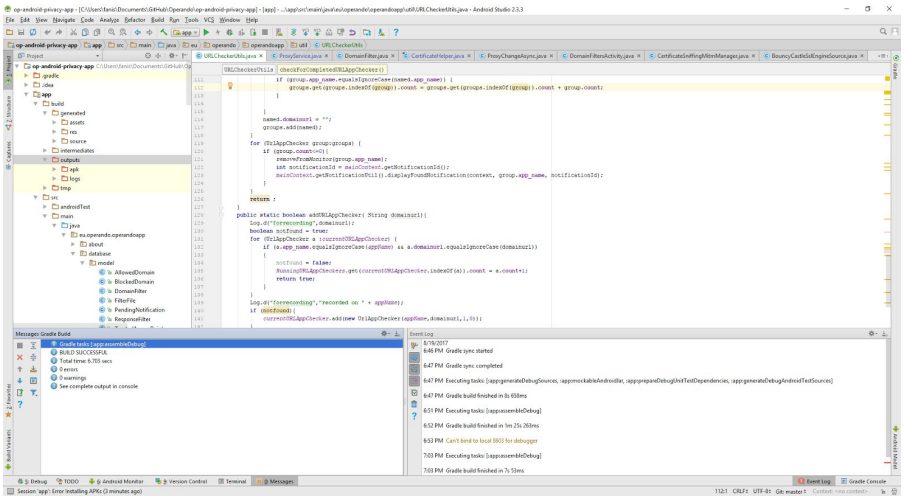
Τα παραπάνω χρησιμοποιήθηκαν εκτενώς κατά την διάρκεια τη εργασίες και θα παρουσιαστούν σε βάθος στο συγκεκριμένο έγγραφο.

4 Θεωρητικά θέματα

4.1 Η σημασία του Android Studio

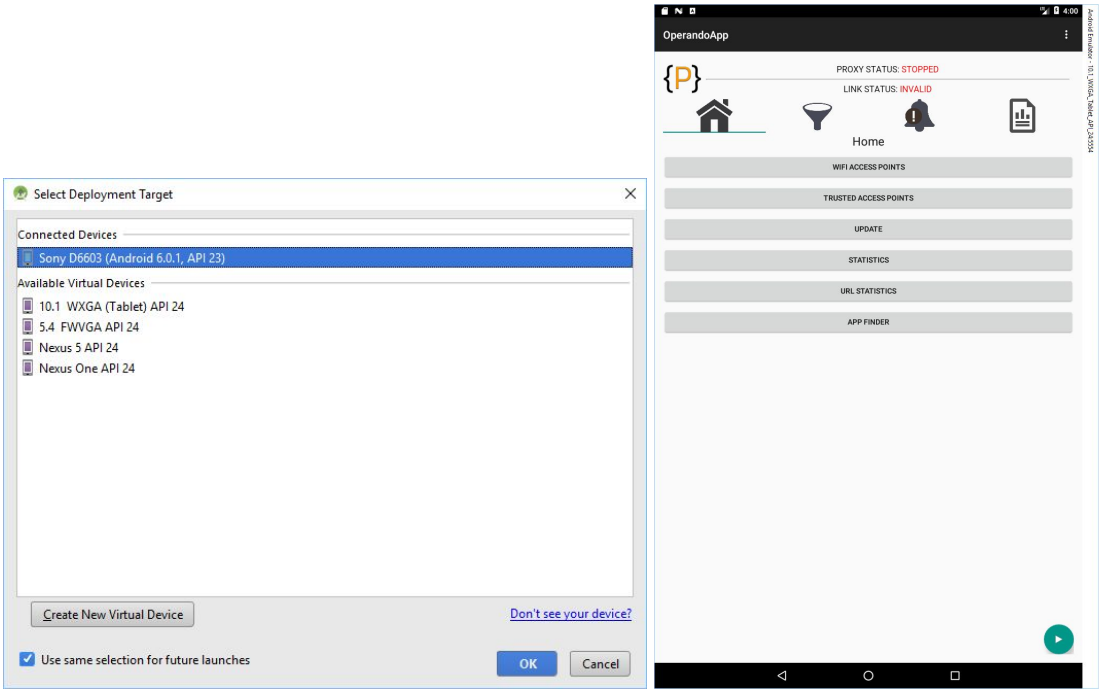
Το Android Studio ανακοινώθηκε στις 16 Μαΐου του 2013 στο συνέδριο Google I/O είναι διαθέσιμο με την άδεια Apache Licence 2 και είναι σίγουρα το καλύτερο αυτήν την στιγμή δωρεάν προγραμματιστικό περιβάλλον για Android. Η δύναμή του έγκειται στο γεγονός ότι μπορείς να φτιάξεις εύκολα εφαρμογές και να τις δοκιμάσεις σε πολλών μορφών emulated κινητά και σε πραγματικό harware με δυνατότητα debug.

Η επιλογή του συγκεκριμένου προϊόντος έναντι του Eclipse φαινόταν μονόδρομος καθώς στην ουσία σε καθοδηγεί στην δημιουργία του κώδικα και σε βοηθά στην ανίχνευση τυπογραφικών και λογικών λαθών.



Η επιφάνεια εργασίας του studio είναι τακτοποιημένη ενώ σου επιτρέπει να παρέμβεις στο τρόπο που είναι τα πάντα τοποθετημένα και να προσθέσεις νέες δυνατότητες. Το gradle , οι προτάσεις πιθανού κώδικα και το εύκολο debuging μας συντρόφευσαν σε αυτό το project. Χωρίς το Android Studio και τα εργαλεία που παρέχει, η κατανόηση του κώδικα της εφαρμογής

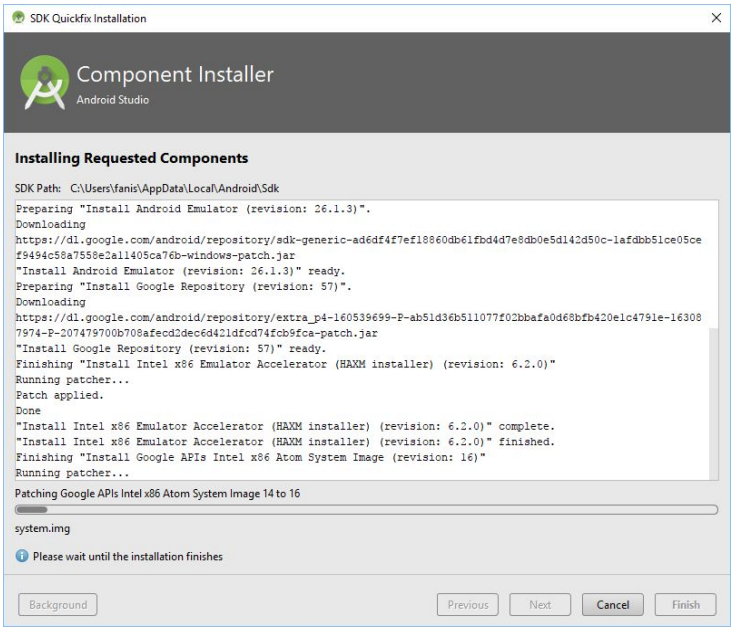
op-android-privacy-app για την οποία δεν είχαμε documentation θα ήταν αδύνατη.



Για την εκτέλεση του κώδικα, με την βοήθεια του Android Studio χρησιμοποιήσαμε όπως φαίνεται και στην εικόνα τόσο emulators κινητών όσο και πραγματικά κινητά και tablet. Τόσο συσκευές που θεωρούνται νέας τεχνολογίας όσο και παλαιότερης.

Μας δυσκόλεψαν τα πολλά versions του λειτουργικού android αλλά και ο διαφορετικός τρόπος που οι κατασκευάστριες εταιρίες προσεγγίζουν από πλευράς ασφάλειας το κινητό.

Μόνο αρνητικό όμως που βρέθηκε κατά την διαδικασία του development ,και που αφορά το περιβάλλον προγραμματισμού, ήταν η συνεχής απαίτηση του εργαλείου για updates και ο χώρος που καταλαμβάνει στο δίσκο. Κάτι αναμενόμενο για ένα εργαλείο με το πλουραλισμό δυνατοτήτων του συγκεκριμένου



4.2 Συμβατότητα ανά version του android.

Παρατηρήσαμε γράφοντας και τεστάροντας τον κώδικά μας ότι τα κινητά και οι εικονικές συσκευές αντιδρούσαν διαφορετικά. Παρακάτω ακολουθεί ένας πίνακας που παρουσιάζει το πως αντιδρά η κάθε συσκευή σε κάθε εκδοσή του android.

Android Api Version	22		22
Android Version	Lollipop 5.0		Lollipop 5.1.1
Device	Sony Xperia Z3	Virtaul Device 22	Samsung Galaxy Tab 4 LTE
Full App	Proxy fully working	Proxy fully working	Proxy fully working. But system warning you
Lite App (Proxy keeping only the Urls)	Proxy working. Gathering only urls	Proxy working. Gathering only urls	Proxy working. Gathering only urls

Android Api Version	23		24	
Android Version	Marshmallow 6.0		Nougat 7.0	
Device	Redmi 4x MIUI | One Plus OxygenOS	Virtaul Device 23	Galaxy A3	Virtaul Device 24
Full App	Google apps like chrome do not deliver web pages	Google apps like chrome do not deliver web pages	Google apps like chrome do not deliver web pages	Google apps like chrome do not deliver web pages
Lite App (Proxy keeping only the Urls)	Proxy working. Gathering only urls	Proxy working. Gathering only urls	Proxy working. Gathering only urls	Proxy working. Gathering only urls

Android Api Version	25	26
Android Version	Nougat 7.1.1	O 8.0
Device	Virtaul Device 25	Virtaul Device 25
Full App	Google apps like chrome do not deliver web pages	Operando proxy not working on the google emulator because of a CPU compatibility issue. But proxy settings is the same as 7.1.1
Lite App (Proxy keeping only the Urls)	Proxy working. Gathering only urls	Operando proxy not working on the google emulator because of a CPU compatibility issue. But proxy settings is the same as 7.1.1

Είναι προφανές από τους παραπάνω πίνακες ότι η ασφάλεια για την google είναι πολύ σημαντική ειδικά στις πιο πρόσφατες εκδόσεις του λειτουργικού της. Μετά την έκδοση 6.0 στην ουσία δεν επιτρέπει την χρήση proxy όταν αυτός αλλάζει το content αφού για όλα τα site και τα προγράμματα της google απαιτείτε να γίνονται accept πιστοποιητικά σαν επικίνδυνα, ενώ μερικές φορές ούτε αυτό είναι αρκετό ή δυνατό.

Παρόλα αυτά στην lite έκδοση όπου έχουμε αφαιρέσει την έκδοση πιστοποιητικού ,τα φίλτρα του content, τον “man on the middle” και στην οποία απλά γίνεται invoke το url ο χρήστης μπορεί να καταγράφει την κίνηση του δικτύου χωρίς προβλήματα και να την χρησιμοποιεί .

4.3 Δικτυακή ταυτότητα. Μια ανακάλυψη που άλλαξε τον στόχο της εργασίας.

Ένας από τους βασικούς στόχους της εργασίας ήταν να βρεθεί ένας τρόπος να καταλαβαίνουμε τις εφαρμογές που τρέχουν στο device χωρίς να έχουμε root access. Αν και αρχικά δεν φαινόταν εφικτό κάτι τέτοιο γρήγορα παρουσιάστηκε μια διέξοδος. Με την ανάλυση των δεδομένων που μαζεύαμε με σκοπό την δημιουργία στατιστικών και γνωρίζοντας τις εφαρμογές που χρησιμοποιούσαμε, αρχίσαμε να ξεχωρίσουμε κάποια μοτίβο. Ναι πράγματι μετά από ενδελεχή ελέγχο τον δεδομένων παρατηρήσαμε ότι κάθε νέα api based εφαρμογή ,δηλαδή κάθε σωστά φτιαγμένη νέα εφαρμογή, αφήνει ένα μοναδικό δικτυακό αποτύπωμα εύκολα ανιχνεύσιμο. Αυτό συνήθως αποτελείτε από μια κλήση στο προφίλ του χρήστη, μερικές κλήσεις για επανάκτηση δεδομένων και αρκετές κλήσεις σε τρίτα api.

Η ερευνά μας όμως δεν σταμάτησε εκεί όπως θα περιγράψουμε και παρακάτω η εφαρμογή είναι ικανή να ανιχνεύσει την ενεργοποίηση μιας κατασκοπευτικής υποεφαρμογής (spyware) ,ενός malware ή ακόμα και εφαρμογών που τρέχουν σε web σελίδες σαν addons. Αν και σε καμία περίπτωσή δεν μπορεί να αντικαταστήσει ένα κλασικό anti-mallware αφού ο proxy μας κάνει monitor μόνο το δίκτυο σίγουρα μπορεί να δράσει επιπρόσθετα για μεγαλύτερη ασφάλεια και για να μην είναι αναγκαίος ο συνεχής επανέλεγχος όλου του συστήματος. Επίσης η δυνατότητα να ανιχνεύει εφαρμογές που βρίσκονται “κρυμμένες” ακόμα και σε άλλες σελίδες μπορεί να χρησιμοποιηθεί για καλύτερο parental control και για προστασία των προσωπικών δεδομένων.

Η παραπάνω ανακάλυψή έστρεψε την εργασία από μια αυστηρή development προσέγγιση σε μια περισσότερο θεωρητική αφού η μελέτη των στατιστικών στοιχείων αλλά και ο πειραματισμός με αυτό το νέο δεδομένο έφερνε συνέχεια μπροστά μας νέα πεδία εφαρμογής που έπρεπε να καταγραφούν και να δοκιμαστούν κατά το δυνατό. Ζητήματα που μας εντυπωσίασαν ήταν η χρήση των Apis της Gooogle από σχεδόν το σύνολο των εφαρμογών, η

συνεχείς κλήσεις αποστολής στατιστικών από τις εφαρμογές, η πολυπλοκότητα των κλήσεων και ο κατακερματισμός των apis μεγάλων εφαρμογών όπως για παράδειγμα του facebook.

Συνοψίζοντας καταλήγουμε στο ότι, το να κάνουμε monitor το δίκτυο του κινητού είναι σαν να κάνουμε monitor τις ίδιες τις εφαρμογές που τρέχουν σε αυτό. Ο μόνος περιορισμός μας, έρχεται από την υπολογιστική δύναμη και τη μνήμη του κινητού. Με την αύξηση των πόρων που είναι διαθέσιμοι σε ένα κινητό, την οποία φέρνει η πρόοδος της τεχνολογίας, θα είναι εύκολο να καταγράφουμε άμεσα κάθε εφαρμογή ή υπόεφαρμογη που τρέχει στο κινητό με μία ταυτότητα. Μια ταυτότητα που θα την αντιπροσωπεύει και όχι με ένα φτιαχτό όνομα που της έδωσε ο δημιουργός της. Επιπρόσθετα αυτή ταυτότητα θα μπορούσε να σταλεί για ανάλυση σε συστήματα τεχνητής νοημοσύνης ώστε να εξαχθούν περαιτέρω συμπεράσματα .Συμπεράσματα που μπορεί να βοηθήσουν στην καταπολέμηση των malware , της κλοπή πνευματικής ιδιοκτησίας, της διαρροής προσωπικών δεδομένων ακόμα και την υπεράσπιση της παιδικής αθωότητας.

Η αλήθεια είναι ότι κατά την διάρκεια της υλοποίησης της συγκεκριμένης εφαρμογής είδαμε τον εαυτό μας να χρησιμοποιεί την εφαρμογή πολύ πιο ενεργά από όσο επιβαλλόταν αφού είχε πολύ καλά αποτελέσματα στην διαδικτυακή προστασία του κινητού μας. Αν δεν ήταν απαιτητική σε επίπεδο resources και δεν ήταν προβληματική για μερικά security components που χρησιμοποιούμε θα την καλωσορίζαμε στην καθημερινότητα μας. Σίγουρα για ανθρώπους που θέλουν να έχουν τον απόλυτο έλεγχο είναι ένα απαραίτητο εργαλείο σε μια εποχή που το Internet είναι άμεσα συνδεδεμένο με την φορητή συσκευή επικοινωνία μας.

4.4 Διαχωρισμός application και frontend & Reusabılity κώδικα

Στην εποχή μας, η τεχνολογία εξελίσσεται με τέτοια ταχύτητα όπου για οποιοδήποτε project αυτού του μεγέθους αξία υπάρχει μόνο αν όλη η μέρη της μπορούν να χρησιμοποιηθούν σε άλλες εφαρμογές. Για να γίνει κάτι τέτοιο ο διαχωρισμός του κώδικα σε λογικές ανεξάρτητες οντότητες είναι επιβεβλημένος Εμείς στα πλαίσια της εργασίας ακολουθήσαμε τη δομή που προτείνεται από το Android Studio. Η δομή αυτή είχε ακολουθηθεί και στο προηγούμενο project. Το frontend βρίσκεται στους καταλόγους res ενώ το application στον κατάλογο java. Για να είναι εύκολα reusable ο κώδικας τα layout χρησιμοποιούν άλλα μικρότερα layout drawable και εικόνες. Όσο αφορά το application κάθε αυτό η κλάση activity της οθόνης χρησιμοποιεί services που με την σειρά τους χρησιμοποιούν αλλά micro-services, φίλτρα και την βάση δεδομένων.

4.5 Reusable δεδομένα

Τα δεδομένα μιας εφαρμογής πρέπει και αυτά με την σειρά τους να μπορούν να γίνουν διαθέσιμα σε άλλες εφαρμογές. Έτσι στην περίπτωση μας όσο αφορά την βάση οι πίνακες είναι προσβάσιμοι μέσω προκατασκευασμένων query τα οποία είναι οργανωμένα σε κλάσεις μέσω των οποίων με api based νοοτροπία έχουμε πρόσβαση στα δεδομένα.

Τα δεδομένα αυτά αποθηκεύονται, στην εφαρμογή, σε πίνακες το δυνατόν με γενικευμένη μορφή και με απλά ονόματα ώστε να μπορούν να χρησιμοποιηθούν από πάνω από ένα services. Services τα οποία μπορούν με την σειρά τους να μοιραστούν τα δεδομένα με πάνω από μία οθόνη.

4.6 Φιλικό προς τον χρήστη UI

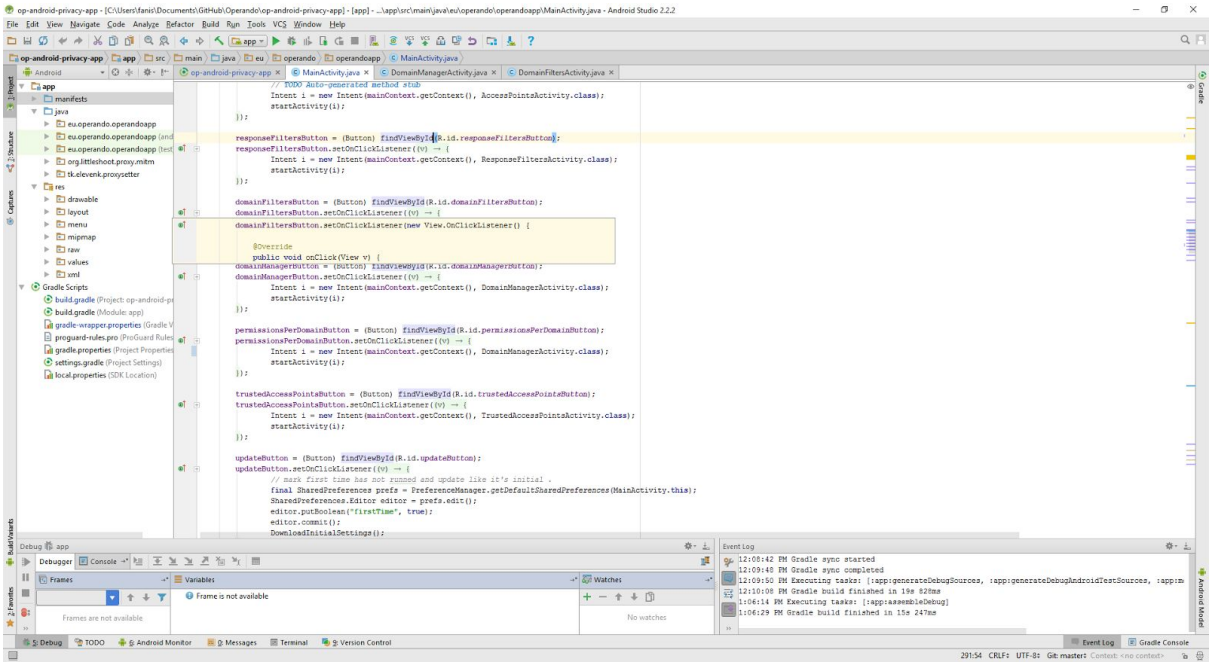
Την σημερινή εποχή οι χρήστες των εφαρμογών έχουν εκπαιδευτεί να κάνουν τα πάντα με λίγες κινήσεις. Δυστυχώς η εφαρμογή ήταν ήδη έτοιμη οπότε οι παρεμβάσεις μας σε αυτό το τομέα έπρεπε να είναι περιορισμένες ώστε να μην φαίνεται η διαφορά. Έτσι αν και κρατήθηκε το θέμα πού είχε η αρχική εφαρμογή όπως και χρωματικές αποχρώσεις θεωρήθηκε αναγκαίο στις 2 νέες οθόνες να γίνουν κάποιες μικρο-αλλαγές ειδικά στο πώς εμφανίζονται οι λίστες και οι επιλογές

στο χρήστη ώστε σε μια οθόνη να έχουμε περισσότερη πληροφορία και να μην χρειάζεται ο χρήστης να γυρνά στην αρχική οθόνη .

5 Τεχνική υλοποίηση και περιγραφή

5.1 Μεταφορά στο Android Studio

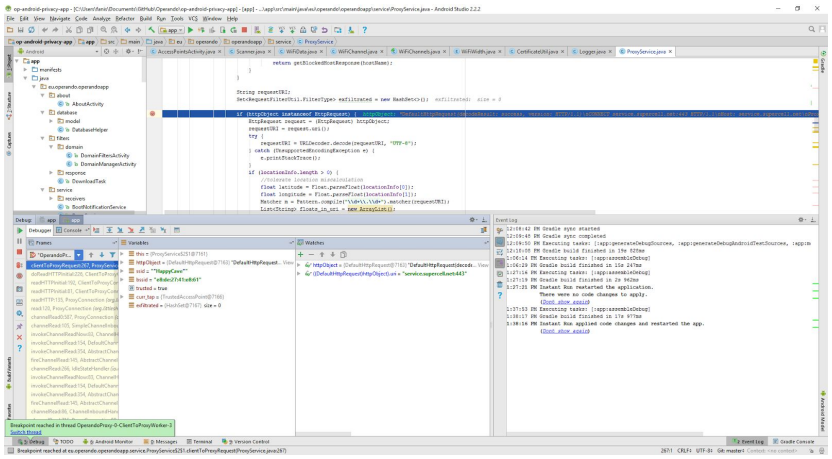
Η εφαρμογή μεταφέρθηκε πλήρως στο Android Studio v 2.3.3



Αυτό μας έδωσε μια πιο καλή οπτική του κώδικα και άφθονα εργαλεία για Debug και για βελτίωση του κώδικα.

5.2 Debug και τι χρειάστηκε να αλλάξουμε από τον κώδικα.

Έτσι σε πρώτη φάση κάνοντας Debug και διαβάζοντας τον κώδικα κάναμε ένα πρόχειρο Documentation στην λύση για να βρούμε τις διεπαφές που θα τοποθετήσουμε τον πρόσθετο κώδικα.



Είδαμε λοιπόν ότι τα βασικά components που μας αφορούν και πρέπει να αλλάξουν είναι τα :

1) Proxyservice.java εδώ προσθέσαμε τα απαραίτητα για να ενημερώνονται με την κίνηση του δικτύου όλα τα νέα components που θα περιγράψουμε
2) DatabaseHelper.java εδώ προσθέσαμε όλον τον απαραίτητο κώδικα που χρειάζεται ώστε να αποθηκεύσουμε τα δεδομένα που χρειαζόμαστε και να τα επαναχρησιμοποιούμε.
3) Build.Gradle εδώ προσθέσαμε μέσω του gradle τις απαραίτητες βιβλιοθήκες.
4) MainActivity.java εδώ ξεκινά η εφαρμογή.
5) AndroidManifest.xml εδώ δηλώθηκαν τα νέα components.

Οι διασυνδέσεις αυτές μας επέτρεψαν να καταγράφουμε επαρκώς όλες τις κινήσεις στο δίκτυο με την βοήθεια του κώδικα που περιγράφουμε παρακάτω.

5.3 Σχήμα βάσης κίνηση πληροφορίας

Η βάση πια έχει δύο νέους σημαντικούς πίνακες στον client. Παρακάτω περιγράφετε το Data Model για τους νέους πίνακες του client

5.3.1 Statistics Table

Column	Data Type	Description
DomainURL	string	Όνομα domain
Count	int	1-999 μετρά πόσες φορές χρησιμοποιήθηκε ένα domain με μέγιστο το 999
Hidden	int	0,1
Modified	datetime	καταγράφει την χρονική στιγμή που έγινε η τελευταία χρήση του URL
SourceActivity	string	Καταγραφή πιθανού app ή Activity που ζητά την συγκεκριμένη κίνηση
Category	string	κατηγοριοποίηση των URL
ExtraInfo	string	Αποθήκευση σε μορφή json πληροφοριών που δεν πρόκειται να χρησιμοποιούνται συχνά. Για παράδειγμα “προτεινόμενες κατηγορίες”, ”επικινδυνότητα site”...

5.3.2 UrlAppChecker Table

Column	Data Type	Description
App_Name	string	Όνομα
DomainUrl	string	Url domain

Count	int	Πόσες φορές ζητηθηκέ το url στο δωσμένο διάστημα
Duration	int	Μετά από πόσα δευτερόλεπτα θα πρέπει να ξανά ελεγχθεί το condition
InitiatorDomain	string	Το domain που κάνει trigger τον rule

5.4 Προσθήκη μηχανισμού ανεύρεσης domain

Η εφαρμογή έχει την δυνατότητα να αναγνωρίσει κάθε νέο domain ενώ ο χρήστης μπορεί εύκολα να κατηγοριοποιήσει αυτά τα domain με λίγα κλικ.

Λόγο της πολυπλοκότητας των domains χρειάστηκε να τα απλοποιήσουμε μεσω της παρακάτω συνάρτησης.

```
public static String getDomainName(String url) {

    String domain = "";

    try {

        URI uri = new URI(url);

        domain = uri.getHost();

    } catch (URISyntaxException ex) {

        return "";

    }

    return domain.startsWith("www.") ? domain.substring(10) : domain;

}

.
```

5.5 Στατιστικά

Στην σημερινή εποχή η πληροφορία είναι δεμένη με την χρήση. Το να χρησιμοποιείς κάποιες λειτουργίες χωρίς να έχεις στατιστικά χρήσης είναι επίπονο και πολλές φορές μάταιο. Στο κώδικα που θα δείτε παρακάτω προσθέσαμε μια κλάση για την διαχείριση των στατιστικών στοιχείων καθώς και αρκετές συναρτήσεις που κάνουν την εφαρμογή μέσω της πληροφορίας πιο χρήσιμη για μας :

5.5.1 Statistics class

Παραθέτουμε την κλάση που περιγράφει τα στατιστικά.

```
package eu.operando.operandoapp.database.model;

/**

* Created by fanis on 1/1/2017.
```

```
    */
    public class UrlStatistic {

       public String domainurl;

       public int count;

       public String modified;

       public String sourceactivity;

       public int hidden;

       public String category;

       public String extrainfo;


       public UrlStatistic(String domainurl,int count,String modified,int
hidden,String sourceactivity,String category){

            this.domainurl = domainurl;

            this.count = count;

            this.hidden = hidden;

            this.modified = modified;

            this.sourceactivity = sourceactivity;

            this.category = category;

            this.extrainfo = extrainfo;

       }

       public UrlStatistic(){

            this.domainurl = null;

            this.count = 0;

            this.hidden = 0;

            this.modified = null;

            this.sourceactivity = null;

            this.category = null;

            this.extrainfo = null;

       }

    }
```

5.5.2 Αποθήκευση χρήσης domain

Η αποθήκευση στη βάση γίνεται με τον παρακάτω κώδικα

```
    public void createUrlStatistic(String domain) {

       final String dom = domain;

       new Thread(new Runnable() {

           @Override

           public void run() {

               UrlStatistic urlStatistic = null;

               try {

                   urlStatistic = getUrlStatistic(dom);

                   if (urlStatistic.domainurl == null )

                   {

                       urlStatistic = new
UrlStatistic(dom,1,null,0,null,null);

                       createUrlStatistic(urlStatistic);

                   }

                   else

                   {

                       addToURLStatistics(urlStatistic);

                   }

               } catch (Exception e) {

                   Log.d("ERROR", e.getMessage());

               }

           }

       }).start();

    }
```

5.5.3 Απόκρυψη χρήσης domain

Σε περίπτωση που κάποιο στατιστικό δεν ενδιαφέρει τον χρήστη αυτός μπορεί να το κρύψει

```
    public void hideUnhideURLStatistics(UrlStatistic urlStatistic) {

       final String url =  urlStatistic.domainurl;

                   SQLiteDatabase db =
DatabaseHelper.this.getWritableDatabase();

                   try {
```

```
                    db.execSQL("UPDATE " + TABLE_URLSTATISTICS + " SET " +
KEY_HIDDEN + " = 1 - " + KEY_HIDDEN + " , "+ KEY_MODIFIED + " = '" +
getDateTime() + "' WHERE " + KEY_DOMAINURL + "= '" + url + "'");

                    } catch (SQLException sqle) {

                        sqle.getMessage();

                    }

    }
```

5.5.4 Ενημέρωση χρήσης domain

Για να διατηρήσουμε μια σωστή δομή στον κώδικα και το reusability έχουμε συνάρτηση για το update.

```
    public int updateUrlStatistic(UrlStatistic urlstatistic) {

       SQLiteDatabase db = this.getWritableDatabase();

       ContentValues values = new ContentValues();

       values.put(KEY_DOMAINURL, urlstatistic.domainurl);

       values.put(KEY_COUNT, urlstatistic.count);

       values.put(KEY_MODIFIED, getDateTime());

       values.put(KEY_HIDDEN, urlstatistic.hidden);

       values.put(KEY_SOURCEACTIVITY, urlstatistic.sourceactivity);

       values.put(KEY_CATEGORY, urlstatistic.category);

       // insert row

       int id = (int) db.update(TABLE_URLSTATISTICS,  values,KEY_DOMAINURL + "
= '" +  urlstatistic.domainurl +"'",null);

       return id;

    }
```

5.5.5 Αναζήτηση χρήσης domain

Παρακάτω ο κώδικας που αφορά την αναζήτηση στην βάση.

```
    public UrlStatistic getUrlStatistic(String domain) {

       UrlStatistic urlStatistic = new UrlStatistic();

       try {

           String selectQuery = "SELECT * FROM " + TABLE_URLSTATISTICS + "
WHERE " + KEY_DOMAINURL + " = '" + domain + "'";

           SQLiteDatabase db = this.getReadableDatabase();

           Cursor c = db.rawQuery(selectQuery, null);
```

```
            if (c.moveToFirst()) {


                urlStatistic = new UrlStatistic(

                        c.getString(c.getColumnIndex(KEY_DOMAINURL))

                        , c.getInt(c.getColumnIndex(KEY_COUNT))

                        , c.getString(c.getColumnIndex(KEY_MODIFIED))

                        , c.getInt(c.getColumnIndex(KEY_HIDDEN))

                        , c.getString(c.getColumnIndex(KEY_SOURCEACTIVITY))

                        , c.getString(c.getColumnIndex(KEY_CATEGORY))

                );

            }

        } catch (Exception e) {

            Log.d("ERROR", e.getMessage());

        }

        return urlStatistic;

    }
```

5.5.6 Κατηγοριοποίηση χρήσης domain

Προσθήκη κατηγορίας σε κάποιο στατιστικό για κατηγοριοποίηση και κατανόηση του.

```
    public void setCategoryURLStatistics(UrlStatistic urlStatistic,String
categoty) {

        final String url =  urlStatistic.domainurl;

        SQLiteDatabase db = DatabaseHelper.this.getWritableDatabase();

        try {

            if (categoty == "") {

                db.execSQL("UPDATE " + TABLE_URLSTATISTICS + " SET " +
KEY_CATEGORY + " = NULL , " + KEY_MODIFIED + " = '" + getDateTime() + "' WHERE
" + KEY_DOMAINURL + "= '" + url + "'");

            }else {

                db.execSQL("UPDATE " + TABLE_URLSTATISTICS + " SET " +
KEY_CATEGORY + " = '" + categoty + "' , " + KEY_MODIFIED + " = '" +
getDateTime() + "' WHERE " + KEY_DOMAINURL + "= '" + url + "'");

            }

            } catch (SQLException sqle) {

            sqle.getMessage();
```

```
        }

    }
```

5.5.7 Αύξηση χρήσης domain

Για να μην γεμίζουμε με άχρηστα δεδομένα υπάρχει ειδικό update για την επαναχρησιμοποίηση ενός domain.

```
    public void addToURLStatistics(UrlStatistic urlStatistic) {

       final String url =  urlStatistic.domainurl;

       if (urlStatistic.count <= 998) {

           new Thread(new Runnable() {

               @Override

               public void run() {

                   SQLiteDatabase db =
DatabaseHelper.this.getWritableDatabase();

                   try {

                       db.execSQL("UPDATE " + TABLE_URLSTATISTICS + " SET " +
KEY_COUNT + " = " + KEY_COUNT + "+1 , "+ KEY_MODIFIED + " = '" + getDateTime()
+ "' WHERE " + KEY_DOMAINURL + "= '" + url + "'");

                   } catch (SQLException sqle) {

                       sqle.getMessage();

                   }

               }

           }).start();

       }

    }
```

5.5.8 Λίστα χρήσης domain

Συλλογή λίστας της χρήσης των domain που είναι κρυμμένα ή όχι.

```
    public List<UrlStatistic> getUrlStatistics(int includeHidden) {

       List<UrlStatistic> urlStatistics = new ArrayList<>();

       try {

           String selectQuery = "SELECT * FROM " + TABLE_URLSTATISTICS;

           if (includeHidden ==0) {selectQuery = selectQuery + " where " +
KEY_HIDDEN +" = 0";}
```

```
            selectQuery = selectQuery + " ORDER BY " + KEY_MODIFIED +" DESC
LIMIT 20 ";

            SQLiteDatabase db = this.getReadableDatabase();

            Cursor c = db.rawQuery(selectQuery, null);

            if (c.moveToFirst()) {

                do {

                    UrlStatistic urlStatistic = new UrlStatistic(

                            c.getString(c.getColumnIndex(KEY_DOMAINURL))

                            , c.getInt(c.getColumnIndex(KEY_COUNT))

                            , c.getString(c.getColumnIndex(KEY_MODIFIED))

                            , c.getInt(c.getColumnIndex(KEY_HIDDEN))

                            , c.getString(c.getColumnIndex(KEY_SOURCEACTIVITY))

                            , c.getString(c.getColumnIndex(KEY_CATEGORY))

                    );

                    urlStatistics.add(urlStatistic);

                } while (c.moveToNext());

            }

        } catch (Exception e) {

            Log.d("ERROR", e.getMessage());

        }

        return urlStatistics;

    }
```

5.5.9 Φίλτρο χρήσης domain

Ο παρακάτω κώδικας χρησιμεύει για να φέρνεις μόνο τα επιθυμητά στατιστικά σε λίστα.

```
    public List<UrlStatistic> getUrlStatistics(String searchString,int
includeHidden) {

        List<UrlStatistic> urlStatistics = new ArrayList<>();

        try {

            String selectQuery = "SELECT * FROM " + TABLE_URLSTATISTICS

                    + " where ( "+ KEY_CATEGORY +" like '%" + searchString +
"%'  OR " + KEY_DOMAINURL +" like '%" + searchString + "%' ) " ;

            if (includeHidden ==0) {selectQuery = selectQuery + " and " +
KEY_HIDDEN +" = 0 ";}
```

```
            selectQuery = selectQuery + " ORDER BY " + KEY_MODIFIED +" DESC
LIMIT 20 ";

            SQLiteDatabase db = this.getReadableDatabase();

            Cursor c = db.rawQuery(selectQuery, null);

            if (c.moveToFirst()) {

                do {

                    UrlStatistic urlStatistic = new UrlStatistic(

                            c.getString(c.getColumnIndex(KEY_DOMAINURL))

                            , c.getInt(c.getColumnIndex(KEY_COUNT))

                            , c.getString(c.getColumnIndex(KEY_MODIFIED))

                            , c.getInt(c.getColumnIndex(KEY_HIDDEN))

                            , c.getString(c.getColumnIndex(KEY_SOURCEACTIVITY))

                            , c.getString(c.getColumnIndex(KEY_CATEGORY))

                    );

                    urlStatistics.add(urlStatistic);

                } while (c.moveToNext());

            }

        } catch (Exception e) {

            Log.d("ERROR", e.getMessage());

        }

        return urlStatistics;

    }

    public UrlStatistic getUrlStatistic(String domain) {

        UrlStatistic urlStatistic = new UrlStatistic();

        try {

            String selectQuery = "SELECT * FROM " + TABLE_URLSTATISTICS + "
WHERE " + KEY_DOMAINURL + " = '" + domain + "'";

            SQLiteDatabase db = this.getReadableDatabase();

            Cursor c = db.rawQuery(selectQuery, null);

            if (c.moveToFirst()) {


                urlStatistic = new UrlStatistic(
```

```
                    c.getString(c.getColumnIndex(KEY_DOMAINURL))
                    , c.getInt(c.getColumnIndex(KEY_COUNT))
                    , c.getString(c.getColumnIndex(KEY_MODIFIED))
                    , c.getInt(c.getColumnIndex(KEY_HIDDEN))
                    , c.getString(c.getColumnIndex(KEY_SOURCEACTIVITY))
                    , c.getString(c.getColumnIndex(KEY_CATEGORY))
            );

        }

    } catch (Exception e) {
        Log.d("ERROR", e.getMessage());
    }
    return urlStatistic;
}
```

5.6 Running app Logger

Βασικό ζητούμενο της εργασίας αυτής ήταν να βρεθεί τρόπος μέσω της πληροφορίας να πιστοποιείται η χρήση μιας εφαρμογής. Ο τρόπος μετά τα πρώτα unit test ήταν προφανής και η ανακάλυψη όπως προαναφέραμε μας ενθουσίασε . Όλες οι νέες εφαρμογές όσο και καλά να έχουν γραφτεί λόγο της εξάρτησης τους από τα api calls και της ανάγκης για συνεχή και γρήγορη ενημέρωση δημιουργούν μια ταυτότητα ροής api κλήσεων. Για να το κάνουμε πιο σαφές θα περιγράψουμε μια τέτοια ταυτότητα. Έστω ότι τρέχει η ΕΦΑΡΜΟΓΗ_Α, τότε περιοδικά ενώ γίνεται χρήση της θα καταγράφετε σε ένα δοσμένο timeFrame .

- 1 κλήση για να επικαιροποιηθεί το profil του χρήστη
- 1 κλήση ανανέωσης του κλειδιού ασφαλείας
- 2 κλήσεις σε συνεργαζόμενο api (π.χ. το g.click) και
- 3 κλήσεις που θα ζητάνε το κεντρικό αντικείμενο δεδομένων

Αυτή η διαπίστωση με την βοήθεια ενός καλού data model μπορεί να φέρει σημαντικά αποτελέσματα όπως την δημιουργία λίστας ενεργών εφαρμογών σε έναν client ή ακόμα και την ανίχνευση ενός spyware το οποίο έχει κρυφτεί στον ίδιο τον κώδικα μιας “σπασμένης” εφαρμογής.

Για να χρησιμοποιήσουμε την παραπάνω πληροφορία φτιάξαμε τις παρακάτω συναρτήσεις

5.6.1 Αποθήκευση χρήσης αναγνωριστικού

Η δημιουργία μιας εγγραφής του αναγνωριστικού βασισμένου στα στατιστικά χρήσης του internet από την εφαρμογή σε δοσμένο χρονικό διάστημα γίνεται από τον παρακάτω κώδικα.

```
public int createUrlAppChecker(UrlAppChecker urlAppChecker) {
    SQLiteDatabase db = this.getWritableDatabase();
```

```
        ContentValues values = new ContentValues();
        values.put(KEY_APP_NAME, urlAppChecker.app_name);
        values.put(KEY_DOMAINURL, urlAppChecker.domainurl);
        values.put(KEY_COUNT, urlAppChecker.count);
        values.put(KEY_DURATION, urlAppChecker.duration);
        // insert row
        int id = (int) db.insert(TABLE_URLAPPCHECKER, null, values);
        return id;
    }
    public int createUrlAppCheckers(List<UrlAppChecker> urlAppCheckers) {
        SQLiteDatabase db = this.getWritableDatabase();

        int count =0 ;
        for (UrlAppChecker urlAppChecker:urlAppCheckers) {
            count = count +   createUrlAppChecker(urlAppChecker);
        }
        return count;
    }
```

5.6.2 Διαγραφή χρήσης αναγνωριστικού

Ενώ η διαγραφή παρακάτω.

```
    public boolean removeUrlAppChecker(String app_name) {
        SQLiteDatabase db = this.getWritableDatabase();
        int id = (int) db.delete(TABLE_URLAPPCHECKER, KEY_APP_NAME + "='" +
app_name + "'", null);
        return id >= 0;
    }
```

5.6.3 Δημιουργία πίνακα χρήσης αναγνωριστικού

Ολόκληρο το αντικείμενο που αποτελεί το αναγνωριστικό δημιουργείτε παρακάτω

```
    public List<UrlAppChecker> drawUrlAppCheckerApps(String app) {
        List<UrlAppChecker> checkers = new ArrayList<>();
        try {
```

```
            String selectQuery = "SELECT "+ KEY_APP_NAME
                    + "," +  KEY_DOMAINURL
                    + "," +  KEY_COUNT
                    + "," +  KEY_DURATION
                    + " FROM " + TABLE_URLAPPCHECKER
                    +" WHERE "+ KEY_APP_NAME + "like '%" + app + "%'";
            SQLiteDatabase db = this.getReadableDatabase();
            Cursor c = db.rawQuery(selectQuery, null);
            if (c.moveToFirst()) {
                do {
                    UrlAppChecker checker = new UrlAppChecker(
                        c.getString(c.getColumnIndex(KEY_APP_NAME))
                            ,c.getString(c.getColumnIndex(KEY_DOMAINURL))
                            ,c.getInt(c.getColumnIndex(KEY_COUNT))
                            ,c.getInt(c.getColumnIndex(KEY_DURATION))
                    );
                    checkers.add(checker);
                } while (c.moveToNext());
            }
        } catch (Exception e) {
            Log.d("ERROR", e.getMessage());
        }
        return checkers;
    }
```

5.6.4 Ενημέρωση πίνακα χρήσης αναγνωριστικού

Η δημιουργία συνάρτησης ενημέρωσης του πίνακα δεν κρίθηκε απαραίτητη καθώς η μόνη περίπτωση χρήσης θα ήταν η δημιουργία νέου αναγνωριστικού . Έτσι προτιμήθηκε η υποκατάσταση της από την διαγραφή και δημιουργία εκ νέου του αναγνωριστικού

5.6.5 Λίστα χρήσης αναγνωριστικού

Στο παρόν κώδικα βλέπουμε το πως μαζεύουμε ένα αναγνωριστικό σε object από την βάση.

```
    public List<UrlAppChecker> getUrlAppChecker(String app_name) {
```

```
        List<UrlAppChecker> urlAppCheckers = new ArrayList<>();
        try {
            String selectQuery = "SELECT * FROM " + TABLE_URLAPPCHECKER;
            selectQuery = selectQuery + " where " + KEY_APP_NAME +" =
app_name";
            selectQuery = selectQuery + " ORDER BY " + KEY_COUNT +" DESC";
            SQLiteDatabase db = this.getReadableDatabase();
            Cursor c = db.rawQuery(selectQuery, null);
            if (c.moveToFirst()) {
                do {
                    UrlAppChecker urlAppChecker = new UrlAppChecker(
                            c.getString(c.getColumnIndex(KEY_APP_NAME))
                            , c.getString(c.getColumnIndex(KEY_DOMAINURL))
                        , c.getInt(c.getColumnIndex(KEY_COUNT))
                            , c.getInt(c.getColumnIndex(KEY_DURATION))
                    );
                    urlAppCheckers.add(urlAppChecker);
                } while (c.moveToNext());
            }
        } catch (Exception e) {
            Log.d("ERROR", e.getMessage());
        }
        return urlAppCheckers;
    }
```

5.6.6 Λίστα χρήσης αναγνωριστικών

Παρακάτω μαζεύουμε μια λίστα με όλα τα application

```
    public List<String> getUrlAppCheckerApps() {
        List<String> strings = new ArrayList<>();
        try {
            String selectQuery = "SELECT Distinct "+ KEY_APP_NAME + " FROM " +
TABLE_URLAPPCHECKER;
```

```
            SQLiteDatabase db = this.getReadableDatabase();

            Cursor c = db.rawQuery(selectQuery, null);

            if (c.moveToFirst()) {

                do {

                    String urlAppChecker =
c.getString(c.getColumnIndex(KEY_APP_NAME));

                    strings.add(urlAppChecker);

                } while (c.moveToNext());

            }

        } catch (Exception e) {

            Log.d("ERROR", e.getMessage());

        }

        return strings;

    }
```

5.6.7 Αναγνώριση εφαρμογής στον proxy

Για να αναγνωρίσουμε τις εφαρμογές την ώρα που τρέχουν έχουμε το παρακάτω.

```
    public static void addToMonitor(String checker_name,List<UrlAppChecker>
urlAppCheckers){

        for (UrlAppChecker urlAppChecker:urlAppCheckers) {

            if (checker_name.equals(urlAppChecker.app_name)) {

                NamedUrlAppChecker namedUrlAppChecker = new
NamedUrlAppChecker(checker_name, urlAppChecker);

                RunningURLAppCheckers.add(namedUrlAppChecker);

            }

        }




    }

    public static void removeFromMonitor(String checker_name){


        Iterator<NamedUrlAppChecker> itr = RunningURLAppCheckers.iterator();

        // remove all even numbers
```

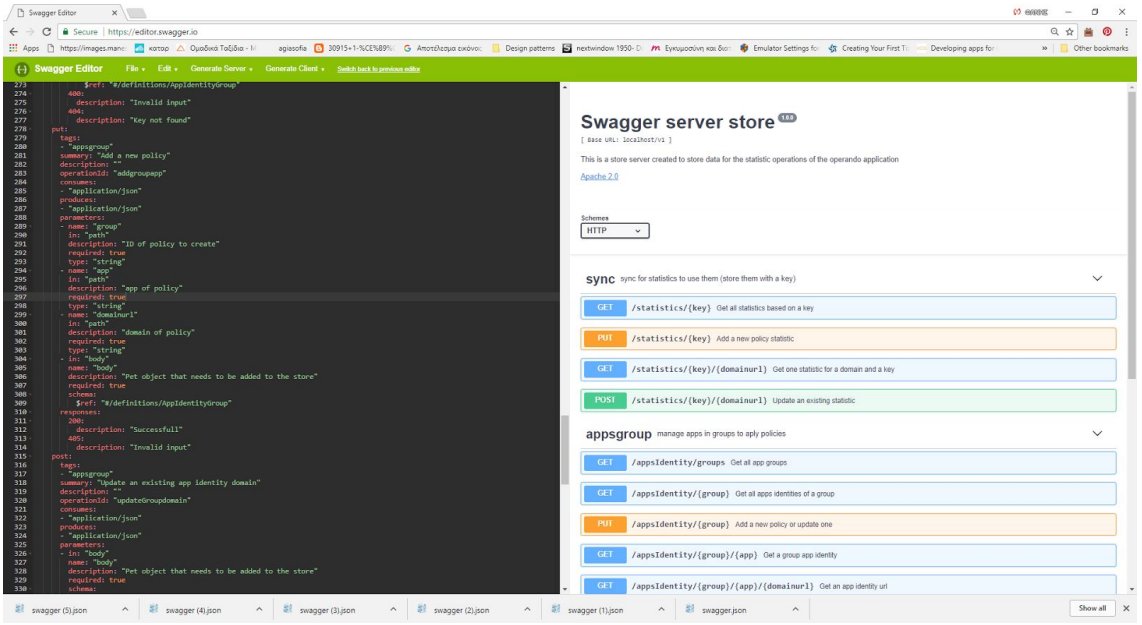
```
        while (itr.hasNext()) {
            NamedUrlAppChecker curr = itr.next();
            if (curr.app_name.equals(checker_name)) { itr.remove(); }
        }

    }
    public static void startRecording(String app_name){
        currentURLAppChecker =  new ArrayList<>();
        appName = app_name;
        recording = true;
        startDate = new java.util.Date();
    }
    public static List<UrlAppChecker> stopRecording(String app_name) {
        recording = false;
        java.util.Date now = new java.util.Date();
        int duration = (int)((now.getTime() - startDate.getTime()) / 1000);
        if (app_name.toString() == "mock".toString()) {
            for (int x = 10; x < 200; x = x + 9) {
                UrlAppChecker addme = new
UrlAppChecker(app_name,"http://appname" + x,  1,duration);
                currentURLAppChecker.add(addme);
            }
        }
        else {
            for (UrlAppChecker a : currentURLAppChecker) {
                a.duration = duration;
                a.count=1;
            }
        }
        appName = null;
        return currentURLAppChecker;
    }
```

```
    public static boolean isRecording(){
       if (appName ==null) {return false;}
       else {return true;}
    }
    public static void checkURLAppChecker(Context context, String domainurl){

       Iterator<NamedUrlAppChecker> itr = RunningURLAppCheckers.iterator();
       // remove all even numbers
       while (itr.hasNext()) {
           NamedUrlAppChecker curr = itr.next();
           if (curr.domainurl.equalsIgnoreCase(domainurl)) {
               int possision =RunningURLAppCheckers.indexOf(curr);
               curr.count = 0;
               RunningURLAppCheckers.set(possision,curr);
               checkForCompletedURLAppChecker(context);
           }
       }


    //   checkForCompletedURLAppChecker(context);
       return;
    }
    public static void checkForCompletedURLAppChecker(Context context){
       List<NamedUrlAppChecker> groups = new ArrayList<>();
       for (NamedUrlAppChecker named :RunningURLAppCheckers) {
           boolean notfound = true;
           for (UrlAppChecker group:groups) {
               if (group.app_name.equalsIgnoreCase(named.app_name)) {
                   groups.get(groups.indexOf(group)).count =
groups.get(groups.indexOf(group)).count + group.count;
                   notfound =false;
               }
```

```
            }
            if(notfound) {
                named.domainurl = "";
                groups.add(named);
            }
        }
        for (UrlAppChecker group:groups) {
            if (group.count<=0){
                removeFromMonitor(group.app_name);
                int notificationId = mainContext.getNotificationId();

mainContext.getNotificationUtil().displayFoundNotification(context,
group.app_name, notificationId);
            }
        }
        return ;
    }
```

5.7 Αποστολή ή ανάκληση δεδομένων από ένα server

Στους στόχους της εργασίας είναι να αποδειχτεί ότι θα μπορούσαμε σε μια τέτοια εφαρμογή να έχουμε μια σύνδεση με server ώστε να δίνονται πολιτικές και να κρατούνται στατιστικά σε ένα δίκτυο εταιρίας ή οργανισμού. Για να διερευνηθεί κάτι τέτοιο θα φτιάξουμε ένα server όπου θα υλοποιεί όλα τα βασικά endpoints για συγχρονισμό του κινητού αλλά θα στέλνει δεδομένα.



5.7.1 Δημιουργία server για δοκιμές με node

Για την δημιουργία ενός τέτοιου server χρησιμοποιήσαμε μια api first προσέγγιση η οποία τείνει να γίνει μόδα λόγο τις καθαρής και γρήγορης επίλυσης προβλημάτων που προσφέρει. Χρησιμοποιήσαμε μερικά πολύ γνωστά framework. Μεταξύ αυτών

Το swagger 2.0 που είναι ένα από τα πιο σύγχρονα πετυχημένα framework και η πηγή εκατοντάδων API developper tools. Με την χρήση του μπορούμε :

- να σχεδιάζουμε τα api,
- να επανακατασκευάζουμε(lifecycling) εύκολα τα api μας ,
- να δημιουργήσουμε διαδραστικό documentation ,
- να δημιουργούμε αυτόματα κώδικα,
- να εγκαθιστούμε αυτόματα τα endpoints
- και τέλος να τεστάρουμε εύκολα τον κωδικά μας.

Το node είναι μια runtime μηχανή για javascript γραμμένη με την βοήθεια της όγδοης έκδοσης της μηχανής του γνωστού μας chrome. Η οποία:

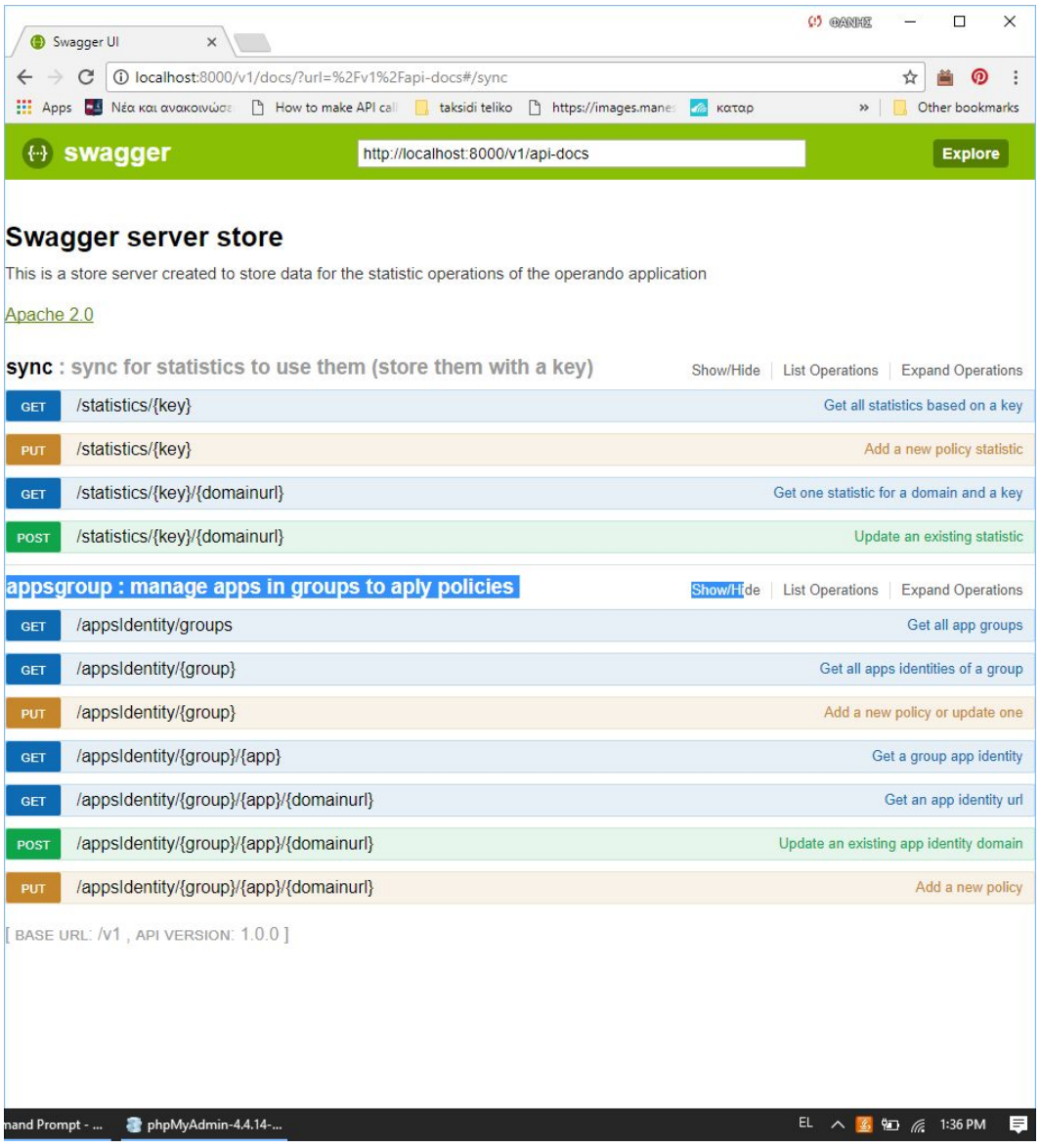
- Χρησιμοποιεί event-driven, non-blocking I/O μοντέλα τα οποία την κάνουν αποτελεσματική και συνάμα ελαφριά.
- Έχει δικό της σύστημα πακέτων το οποία χρησιμοποιείτε από όλη την κοινότητα των προγραμματιστών web εφαρμογών.
- Μπορείς εύκολα να μετατρέψεις με την βοήθεια της κάθε web βιβλιοθήκη και να την χρησιμοποιείς για δουλειές σε servers και scipts και cron jobs
- Τέλος είναι αρκετά cross platform

Στην περίπτωση μας , όπως επιβάλει το framework, αρχικά καταγράψαμε όλα τα endpoints με την βοήθεια ενός swagger editor. Σε αυτά τα endpoints φροντίσαμε να υπάρχουν GET REST CALLS με λίστες για όλους τους τύπους δεδομένων που αποθηκεύουμε στον server αλλά και PUT και POST verbs για να γίνεται το απαραίτητο populate των δεδομένων. Έπειτα επιλέξαμε το swaggerize του Node που δημιουργεί όλα τα interfaces που υπάρχουν στο swagger σε express node server. Έτσι για να δημιουργήσουμε τα api calls τρέξαμε

```
$ npm install -g yo
$ npm install -g generator-swaggerize
$ mkdir apiserver
$ cd apiserver
$ yo swaggerize
```

και επιλέγουμε το swagger json όταν μας ζητά το definition api

Στο σημείο αυτό θέλουμε να σημειώσουμε πως για να μην γεμίσουμε το έγγραφο περιττή πληροφορία παραθέτουμε σαν documentation το ίδιο το swagger file το οποίο περιέχει στην κατά το δυνατό πιο συνοπτική μορφή την ακριβή περιγραφή των κλήσεων. Το παρακάτω swagger json έγγραφό φαίνεται σε interactive μορφη σε οποιοδήποτε swagger editor υποστηρίζει swagger 2.0 όπως ο http://editor.swagger.io/ ή στο http://localhost:8000/v1/docs του server μας



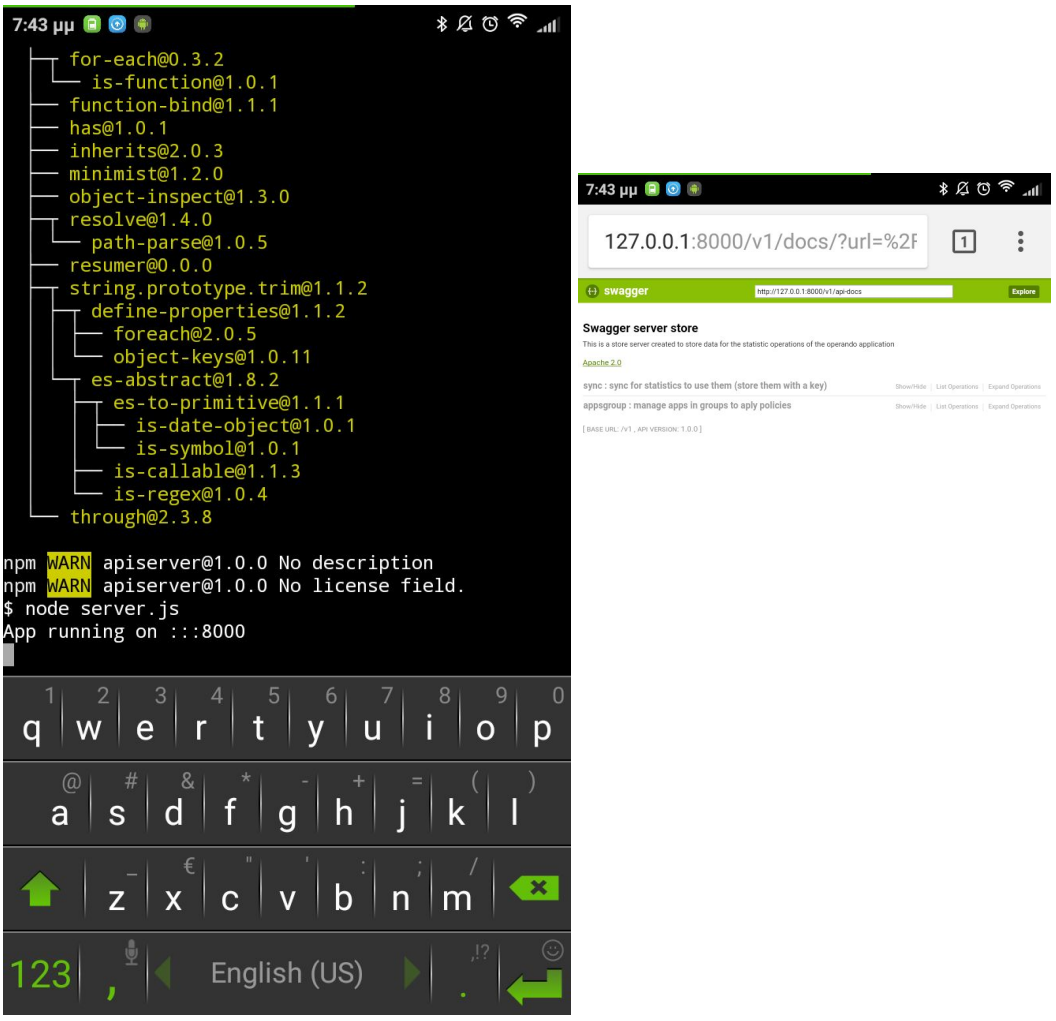
Για να προσομοιώσουμε την βάση των δεδομένων του server χρησιμοποιήσαμε την nodejs lowDB, μια βιβλιοθήκη που σου επιτρέπει να κρατάς και να διαχειρίζεσαι δεδομένα που αποθηκεύονται σε json σε φυσικά αρχεία. Τα json αρχεία καθώς δεν είναι η ιδανικότερη υποδομή για τα δεδομένα δεν χρεώνονται από τους παρόχους cloud υπηρεσιών σαν το Heroku. . Η παραπάνω βιβλιοθήκη-βάση μας φάνηκε πολύ χρήσιμη και σίγουρα θα συνεχίσουμε να την χρησιμοποιούμε για frontend εφαρμογές και για sandbox εφαρμογές. Ακολουθούν παραδείγματα του κώδικα που γράφει στην βάση του server.

```
db.defaults({ statistics: [], appidentity: [] }).write();

...

        try {
         //  var datares ={"Statistics": [] };
         var   datares = db
             .get("statistics")
             .filter({ name: req.params.key,domainurl: req.params.domainurl })
             .value();
           res.status(200).send(datares[0]);
```

```
            } catch (err) {
              res.status(500).send(errorsmsg[500]());
            }

...

            try {
              var  input=req.body;
              delete input.id;
              input.name = req.params.key;
              input.domainurl = req.params.domainurl;
              var datares ={"Statistics": [] };
               datares.Statistics = db
                .get("statistics")
                .find({ name: req.params.key,domainurl: req.params.domainurl })
                .assign(input)
                .write()
              res.status(200).send(datares);
            } catch (err) {
              res.status(500).send(errorsmsg[500]());
            }

...

        try {
           var datares = db
             .get("appidentity")
             .push(req.body)
             .write();
           res.status(200).send(errorsmsg[200]());
         } catch (err) {
           res.status(500).send(errorsmsg[500]());
         }
```

5.7.2 Run node server στο Android

Για να μπορέσουμε να κάνουμε debug με τον server που φτιάξαμε, μετατρέψαμε το ίδιο το κινητό σε node server. Κάναμε install το Termux που δίνει ένα terminal ώστε να χρησιμοποιήσεις το android σαν linux κάναμε σε αυτό update το linux source.list και μετά εκτελέσαμε “apt install nodejs”. Έτσι είχαμε ποια στο κινητό μας εγκατεστημένα τα npm και node. Έπειτα δίνοντας δικαιώματα στο termux να πειράζει τα αρχεία της συσκευής με την εντολή “termux-setup-storage” μπορέσαμε να αντιγράψουμε τα εκτελέσιμα του server και να τον κάνουμε install στο κινητό όπως περιγράφεται στην προηγούμενη παράγραφο. Αφού τρέξουμε το εκτελέσιμο του server έχουμε τον server να τρέχει σε μια πόρτα στο localhost του κινητού.



Το παραπάνω πρέπει να τονίσουμε ότι μας εντυπωσίασε καθώς έτσι αποδεικνύεται ότι μπορούμε να χρησιμοποιήσουμε όλο το οπλοστάσιο που δίνει το node για να φτιάξουμε εφαρμογές σε κινητά.

5.7.3 Upload του server σε cloud υπηρεσία

Για να μπορέσουμε να έχουμε πρόσβαση στον server εκτός δικτύου έπρεπε να επιλέξουμε κάποια cloud υπηρεσία ώστε να ανεβάσουμε το προαναφερόμενο server. Δοκιμάσαμε αρκετές λύσεις και τελικά καταλήξαμε στην Heroku που ήταν πρακτικά η μόνη υπηρεσία που πρόσφερε χώρο για την nodejs εφαρμογή μας και είχε και ικανοποιητικό χρόνο deployment της εφαρμογής. Στο παρακάτω site περιγράφετε εκτενώς το πως χρησιμοποιείται https://devcenter.heroku.com/articles/getting-started-with-nodejs#introduction

5.7.4 Voley! Νέα βιβλιοθήκη για κλήσεις σε web api.

Πώς όμως θα καλέσουμε αυτόν τον server από την εφαρμογή μας. Η τεχνολογία καθημερινά γίνεται πιο απαιτητική σε διαδικτυακές συνδέσεις αλλά ευτυχώς ταυτόχρονα δημιουργούνται και λύσεις. Σε αυτά τα πλαίσια η google πήρε υπό την στέγη της , βελτίωσε και παρουσίασε μια βιβλιοθήκη που αναλαμβάνει να στέλνει εύκολα κλήσεις στο internet μέσω μιας ουράς. Η voley που παρουσιάζεται αναλυτικά από την ίδια την google στην σελίδα https://developer.android.com/training/volley/index.html μπορεί :

- να προγραμματίζει αυτόματα αιτήματα στο δικτύου.
- να δημιουργήσει πολλαπλές ταυτόχρονες συνδέσεις δικτύου.

- να διαχειρίζεται cache μνήμες και την συνδέση τους στο δίσκο και στη μνήμη.
- να υποστηρίξει ιεράρχηση των κλήσεων.
- επίσης μπορεί να ακυρώσει αιτήματα ομαδοποιημένα ή μη.
- να υποστηρίξει κάθε λογής σενάρια καθώς δίνει πολλές δυνατότητες προσαρμογής.
- επίσης μέσω της αυστηρής σειράς που ζητά τα request να διορθώνει τυχών λάθει στο UI από ασύγχρονες κλήσεις
- τέλος να διαχειριστεί με επιτυχία σφάλματα .

Αυτή η σχεδόν μαγική βιβλιοθήκη η οποία χρησιμοποιούμε και εμείς είναι διαθέσιμη ελεύθερα στο github στην διεύθυνση https://github.com/google/volley.

5.7.5 Παράδειγμα ανάκλησης δεδομένων

Παρακάτω παραθέτουμε τον κώδικα με το πως με την βοήθεια της voley μπορέσαμε να καλέσουμε το server για να πάρουμε δεδομένα. Μάλιστα η voley δίνει την δυνατότητα να δουλέψουμε απευθείας με json κάτι που μας έκανε την ζωή εξαιρετικά εύκολη.

```
    public class connectWithServer {

       final private RequestQueue nQueue;

       final private Context cContext;

       final private String apiurl = "http://localhost:8000/v1/";

       public connectWithServer(Context currentContext) {

           nQueue = Volley.newRequestQueue(currentContext);

           cContext = currentContext;

       }

    ...


    public void TestGetSync() {

       //test call api sync

       JSONObject syncurl = new JSONObject();

       JSONObject syncdata = new JSONObject();

       String currentResponse = "";




       // Request a string response from the provided URL.

       JsonObjectRequest stringRequest = new
JsonObjectRequest(Request.Method.GET,

               apiurl + "appsIdentity/groups", null,

               new Response.Listener<JSONObject>() {

                   @Override
```

```
                    public void onResponse(JSONObject response) {
                        // Display the first 500 characters of the response
string.
                        String mstring = "";
                        try {
                            mstring = response.toString(2);
                        } catch (JSONException e) {
                            mstring = "";
                            e.printStackTrace();
                        }

                        Toast.makeText(cContext,
                                "Response is: " + mstring,
                                Toast.LENGTH_SHORT).show();
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                //error
                if (error == null || error.networkResponse == null) {
                    return;
                }
                //get status code here
                final String statusCode =
String.valueOf(error.networkResponse.statusCode);
                //get response body and parse with appropriate encoding
                try {
                    Toast.makeText(cContext,
                            "Fail in: " + new
String(error.networkResponse.data, "UTF-8"),
                            Toast.LENGTH_SHORT).show();
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
```

```
                }
            }
        });
        // Add the request to the RequestQueue.
        stringRequest.setTag("main");
        nQueue.add(stringRequest);
    }
    ...
        public void closeQueue() {
            if (nQueue != null) {
                nQueue.cancelAll("main");
            }
        }
    }
```

Μπορείτε να παρατηρήσετε ότι η βιβλιοθήκη δίνει την δυνατότητα τόσο να προσθέσεις κλήσεις στην ουρά όσο και να ακυρώσεις τις κλήσεις κατά βούληση. Το παραπάνω είναι εξαιρετικά βολικό ώστε να σταματήσεις τυχών streams όταν καταστρέφεται κάποιο activity content.

```
    @Override
    protected void onStop () {
        super.onStop();
        serverConnection.closeQueue();
    }
```

5.7.6 Παράδειγμα αποστολής δεδομένων

Με την ίδια ευκολία στείλαμε και δεδομένα στον server μέσω post.

```
    public void TestSendApp(UrlAppChecker currentUrlAppChecker) {
        //test call api sync
        JSONObject syncurl = new JSONObject();
        JSONObject syncreq = new JSONObject();
        JSONObject json = new JSONObject();
        JSONObject manJson = new JSONObject();
        try {
            syncreq.put("id", 1);
```

```
            syncreq.put("name", "checkapp");
            syncreq.put("domainurl", currentUrlAppChecker.domainurl);
            syncreq.put("app_name", currentUrlAppChecker.app_name);
            syncreq.put("count", currentUrlAppChecker.count);
            syncreq.put("duration", currentUrlAppChecker.duration);
        }
        catch (JSONException e) {
            e.printStackTrace();
        }
        JSONObject syncdata = new JSONObject();
        String currentResponse = "";
        // Request a string response from the provided URL.
        JsonObjectRequest stringRequest = new
JsonObjectRequest(Request.Method.POST,
                apiurl + "appsIdentity/groups", syncreq,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        // Display the first 500 characters of the response
string.
                        String mstring = "";
                        try {
                            mstring = response.toString(2);
                        } catch (JSONException e) {
                            mstring = "";
                            e.printStackTrace();
                        }
                        Toast.makeText(cContext,
                                "Response is: " + mstring,
                                Toast.LENGTH_SHORT).show();
                    }
                }, new Response.ErrorListener() {
```

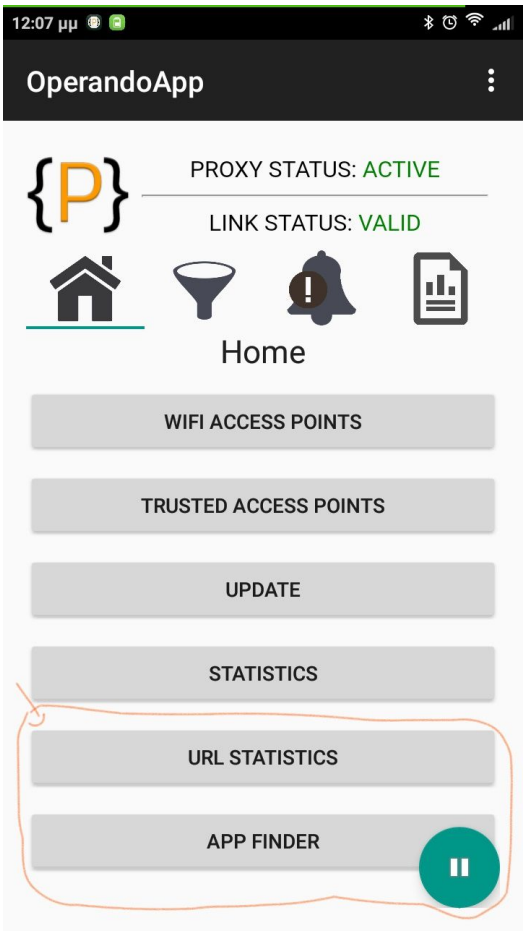
```
            @Override
            public void onErrorResponse(VolleyError error) {
                //error
                if (error == null || error.networkResponse == null) {
                    return;
                }
                //get status code here
                final String statusCode =
String.valueOf(error.networkResponse.statusCode);
                //get response body and parse with appropriate encoding
                try {
                    Toast.makeText(cContext,
                            "Fail in: " + new
String(error.networkResponse.data, "UTF-8"),
                            Toast.LENGTH_SHORT).show();
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }
            }
        });
        // Add the request to the RequestQueue.
        stringRequest.setTag("main");
        nQueue.add(stringRequest);
    }
```

Αν και το να έχεις ένα πλήρες server συνδεδεμένο στην εφαρμογή σου έχει πολύ μεγάλο value το να φτιάχναμε κάτι τέτοιο σε μια εργασία που αφορά το android θα ξέφευγε αρκετά από τους στόχους μας. Πάρα ταύτα έχουν γίνει τα απαραίτητα βήματα που δίνουν ένα πλήρες proof of concept .

5.8 Reports-Screens

Όσο αφορά τις οθόνες παρακάτω έχουμε τα frames και μια μικρή περιγραφή για αυτά.

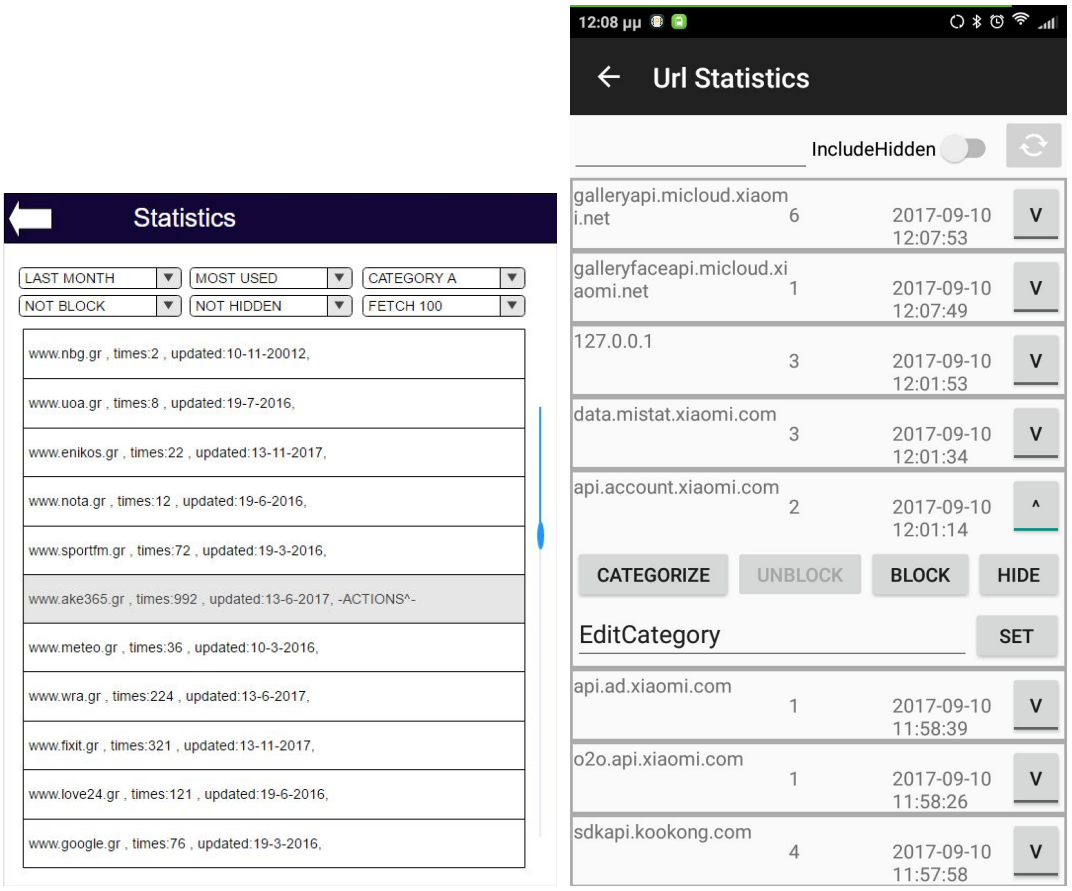
5.8.1 Νέες λειτουργίες



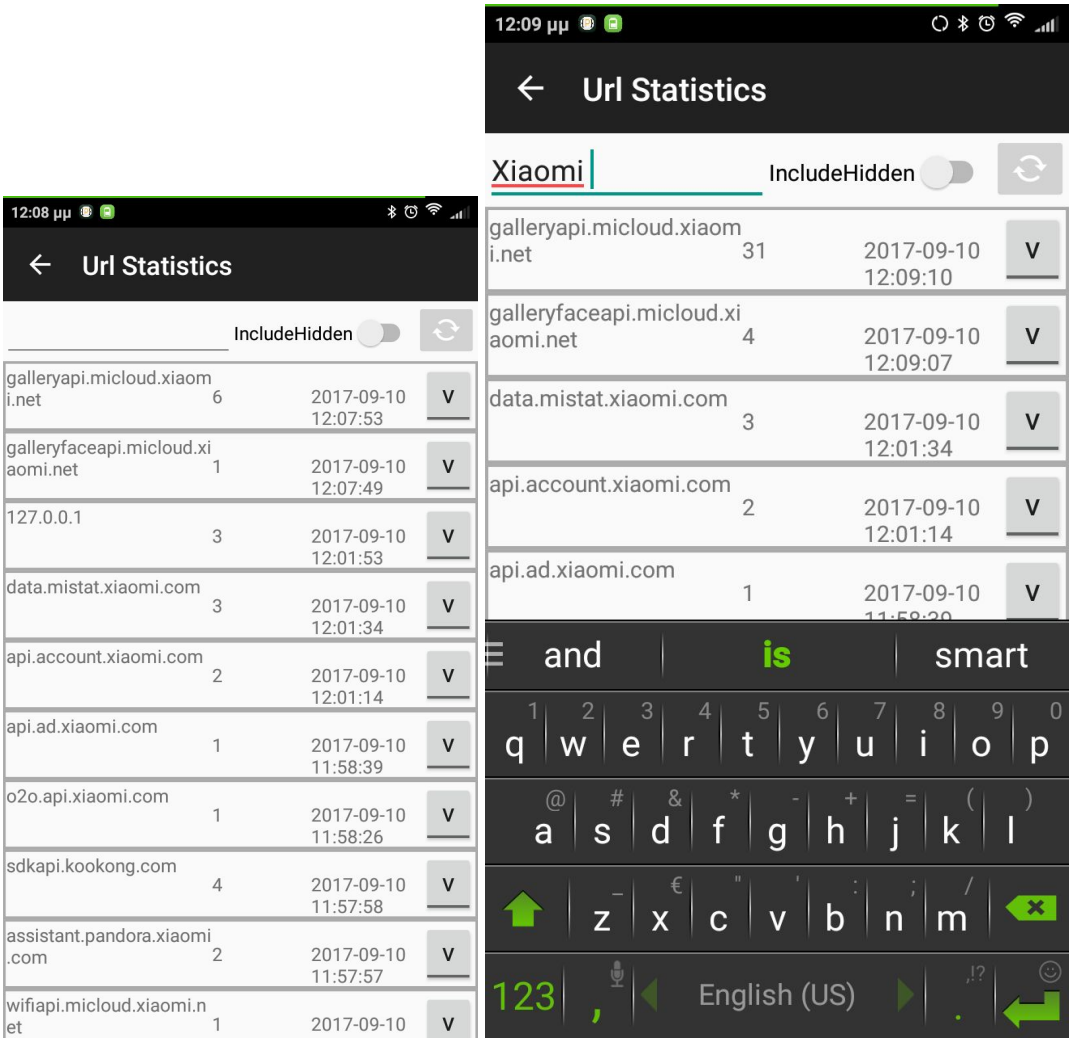
Στην αρχική οθόνη μετά τις αλλαγές βλέπουμε δύο νέα κουμπιά. Το ένα μας κατευθύνει στα στατιστικά που κρατιούνται για κάθε κίνηση μας στο internet , θα τα σχολιάσουμε εκτενώς παρακάτω. Το δεύτερο μας στέλνει στην οθόνη που αφορά την καταγραφή και ανακάλυψη εφαρμογών.

5.8.2 Προβολή στατιστικών και διαχείριση.

Πιο συγκεκριμένα για την οθόνη των στατιστικών στα whiteframe που είχαμε σχεδιάσει είχε σαφώς δυο ξεχωριστά μέρη που στην ουσία μόνο το ένα είχε το interaction με το χρήστη. Στην τελική οθόνη επηρεασμένοι από τις τελευταίες εφαρμογές της google έχουμε μεταφέρει ενέργειες που αφορούν τα items σε accordaion στο ίδιο το αντικείμενο URL στατιστικό και έχουμε ένα text φίλτρο που αφορά κατηγορίες και λεκτικά .

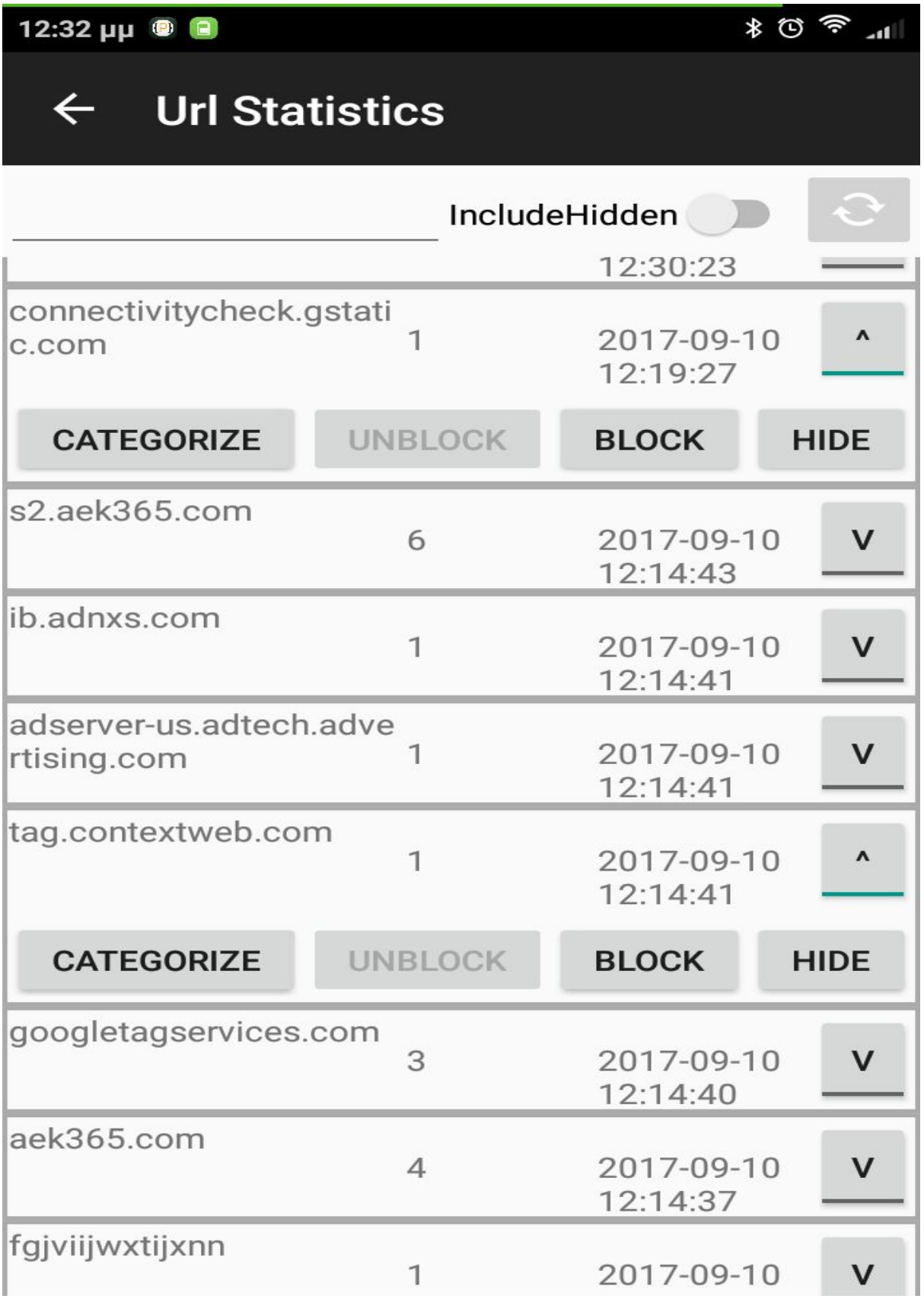


Στην ουσία αρχικά έχουμε μια λίστα με όλα τα full domains που έχουν χτυπηθεί τελευταία. Στην λίστα μας επίσης βλέπουμε εκτός από το domain, πόσες φορές έχει χτυπηθεί κάποιο από αυτά και την χρονική στιγμή που χτυπήθηκε τελευταία φορά. Στο counter έχει μπει το 999 σαν μέγιστο καθώς μερικά site όπως της google είναι λογικό να χτυπηθούν χιλιάδες φορές και θα ήταν για εμάς περιττή πληροφορία που θα χαλούσε τις οθόνες μας. Η ημερομηνία αποφασίσαμε να φαίνεται σε δύο σειρές για να έρχεται καλύτερα στο μάτι.



Σε κάθε ένα από τα URL στατιστικά μπορούμε να ορίσουμε κατηγορίες (TAGS) να το κρύψουμε ώστε να μην τα βρίσκουμε μπροστά μας συνεχώς στην κεντρική οθόνη ή ακόμα και να το μπλοκάρουμε με την βοήθεια του block module που υπήρχε ήδη στην εφαρμογή.

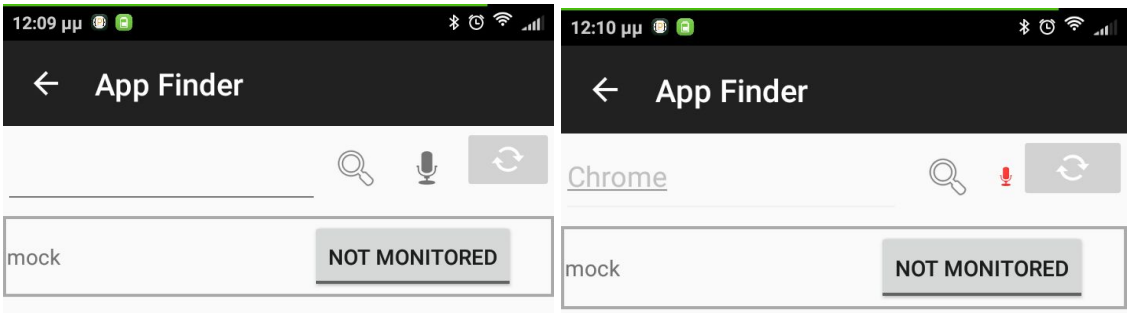
Κατά την διάρκεια δοκιμής της εφαρμογής μας φροντίζαμε κάθε φορά μετά την χρήση του Internet να μπαίνουμε στην συγκεκριμένη σελίδα να μπλοκάρουμε κάθε domain πού είτε μας ήταν άγνωστο είτε αφορούσε διαφημιστικά είτε έμοιαζε με spyware και μετά να κρύβουμε όσα από αυτά δεν θέλαμε να ξαναδούμε. Οι παραπάνω ενέργειες είχαν σαν προφανές αποτέλεσμα την επόμενη φορά που μπαίναμε στο ίδιο site να μην είχαμε καθυστερήσεις από blockware και συνάμα τα στατιστικά μας ποια να αφορούν νέες διαδικτυακές ενέργειες


12:32 μμ
Url Statistics
IncludeHidden
12:30:23
connectivitycheck.gstatic.com
1
2017-09-10 12:19:27
^
CATEGORIZE
UNBLOCK
BLOCK
HIDE
s2.aek365.com
6
2017-09-10 12:14:43
V
ib.adnxs.com
1
2017-09-10 12:14:41
V
adserver-us.adtech.advertising.com
1
2017-09-10 12:14:41
V
tag.contextweb.com
1
2017-09-10 12:14:41
^
CATEGORIZE
UNBLOCK
BLOCK
HIDE
googletagservices.com
3
2017-09-10 12:14:40
V
aek365.com
4
2017-09-10 12:14:37
V
fgjviijwxtijxnn
1
2017-09-10
V

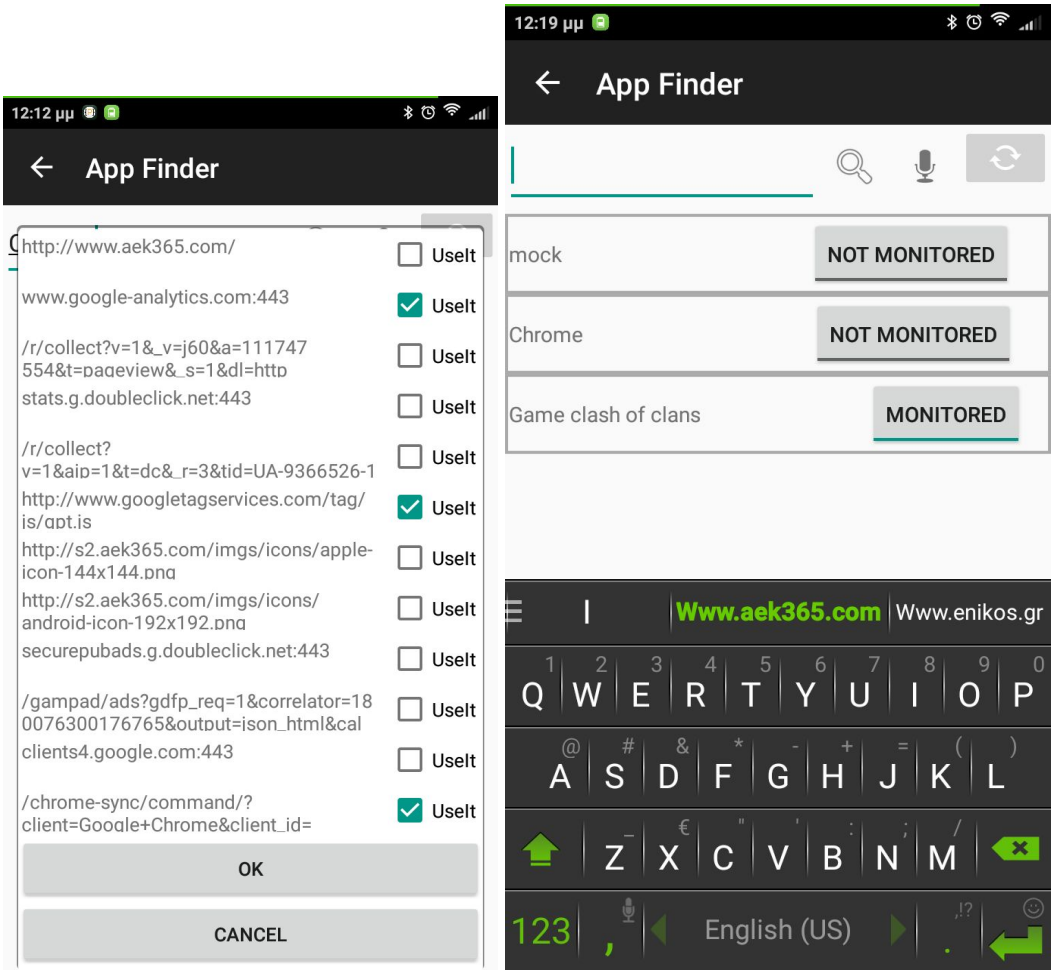
5.8.3 Προβολή και διαχείριση αναγνωριστικών ενεργών εφαρμογών.

Στις παρακάτω εικόνες περιγράφετε πλήρως το πώς μπορούμε να καταγράψουμε την ταυτότητα μιας εφαρμογής από τις δικτυακές τις κινήσεις και να ελέγξουμε αν αυτή τρέχει στο κινητό μας.

Βήμα 1) Αρχικά δίνουμε ένα όνομα στην εφαρμογή και πατάμε καταγραφή



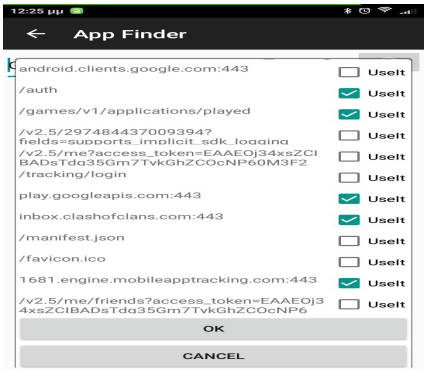
Βήμα 2) Έπειτα χρησιμοποιούμε την εφαρμογή που θέλουμε να ταυτοποιήσουμε, ξαναπατάμε το κουμπί καταγραφή και βλέπουμε μια λίστα από Rest Calls και sites.

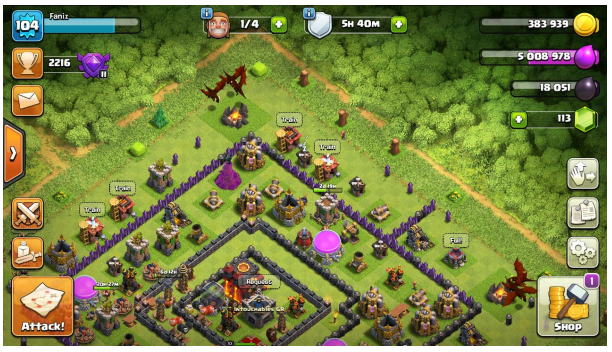


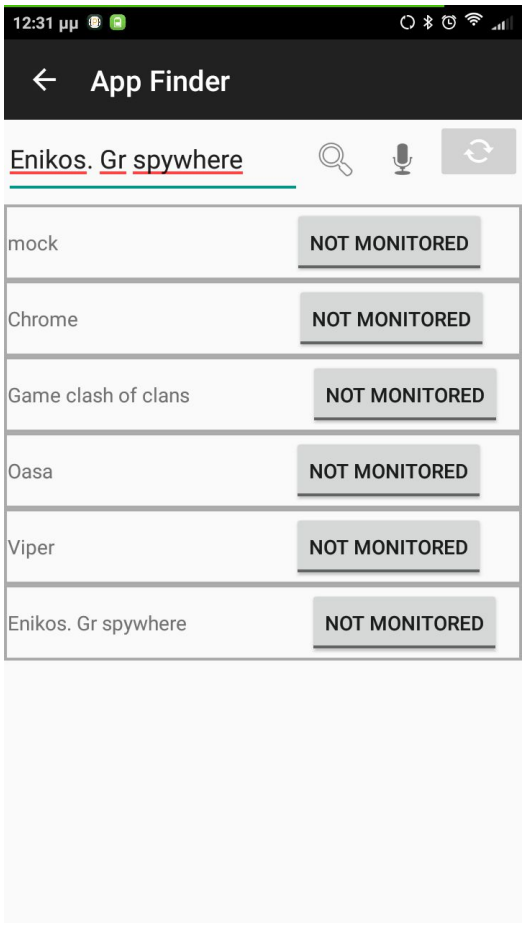
Βήμα 3) Από αυτά ξεχωρίζουμε όσα δεν χαρακτηρίζουν την εφαρμογή και διαλέγουμε εκείνα που θεωρούμε ότι μοναδικοποιούν την εφαρμογή. Οι παραπάνω εικόνες αφορούν την

ταυτότητα της εφαρμογής του chrome. Ενώ οι αμέσως επόμενες εικόνες αφορούν ένα πολύ δημοφιλές παιχνίδι το Clans of Clan.

Βήμα 4) Επαναλαμβάνοντας αυτή την διαδικασία έχουμε μια λίστα από ταυτότητες εφαρμογών της οποίες μπορούμε να κάνουμε monitor.





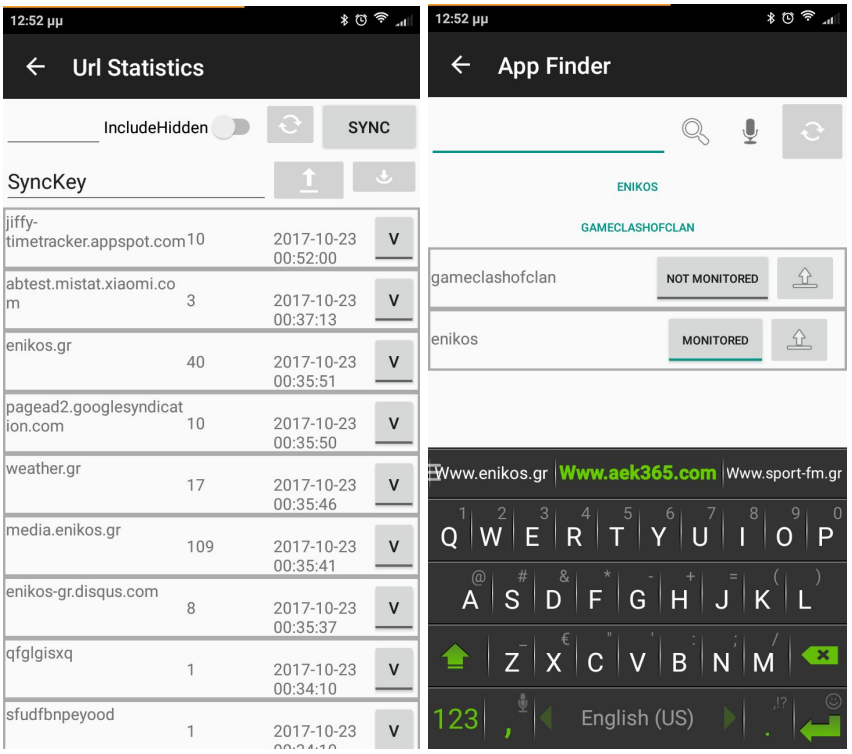


Βήμα 5) Κάνοντας monitor τις εφαρμογές παίρνουμε alerts για το πότε τρέχουν.

5.8.1 Συγχρονισμός με τον Server

Για τη σύνδεση με τον server, δημιουργήθηκαν αρκετά νέα κουμπιά και drop down menu. Στη σελίδα των στατιστικών των url υπάρχει πια ένα κουμπί sync. Αν πατηθεί ανοίγει επιλογές για

upload και download δεδομένων, καθώς επίσης και ενα πεδίο για την εισαγωγή λεκτικού που θα αντιπροσωπεύει τα δεδομένα προς συνχρονισμό. Παράδειγμα γράφοντας στο πεδίο κειμένου initial και πατώντας το κουμπί download θα εισαχθούν στην εφαρμογή από το server στατιστικά που έχουμε αποθηκεύσει ως δεδομένα αρχικοποίησης. Στην σελίδα αναγνώρισης εφαρμογών μπορεί να γίνει upload το αναγνωριστικό κάποιας εφαρμογής, επίσης μέσω του search button εμφανίζεται λίστα εφαρμογών από το server οι οποίες είναι διαθέσιμες για download.



6 Συμπεράσματα

6.1 Value εφαρμογής

Η εφαρμογή αυτή αναμφίβολα έχει τεράστιο value. Value που προκύπτει τόσο επειδή μπορεί να χρησιμοποιηθεί αυτούσια από χρήστες που θέλουν να πάρουν πίσω στο έλεγχο τους την δικτυακή κίνηση στο κινητό τους όσο γιατί οι συναρτήσεις αυτής της εφαρμογής μπορούν να χρησιμοποιηθούν σαν βάση από άλλες εφαρμογές.

Μέσω αυτής της εφαρμογής εμείς μάθαμε τα κακός κείμενα σε site που χρησιμοποιούμε καθημερινά και καταφέραμε να βρούμε τρόπο να ανακαλύπτουμε εφαρμογές που τρέχουν στο backround χωρίς να το γνωρίζουμε. Συνάμα ελπίζουμε να αφήσαμε παρακαταθήκη για δικτυακές εφαρμογές που μπορούν να τρέχουν είτε στο ίδιο το κινητό είτε σε κάποιο ρουτερ, dns ή vpn και μπορεί να καταγράφει τα url που χρησιμοποιούνται από κάποιο device σε πραγματικό χρόνο. Εκτός από την παρέμβαση όμως πρωτογενώς στις δικτυακές κινήσεις είναι σαφές ότι οι συναρτήσεις που γράφτηκαν μπορούν να λειτουργήσουν και ως βάση για εφαρμογές που κάνουν focus αποκλείστηκα στην ασφάλεια και μπορούν να λειτουργήσουν επικουρικά στα κλασικά antivirus.

Εν κατακλείδι αν και δεν είχαμε αρκετή εμπειρία σε εφαρμογές κινητών είμαστε σίγουροι ότι εφαρμογές σαν και αυτή θα αποτελέσουν το νέο trend της εποχής της διαδικτυακής σύνδεσης.

6.2 Πράγματα που δεν έχουν γίνει και απομένουν για μελλοντική έρευνα , Πιθανές επεκτάσεις .

Στον σχεδιασμό της εφαρμογής έχουν ήδη προβλεφθεί άλλες λειτουργίες που δεν προστέθηκαν σε αυτήν την έκδοση όπως

- Ο server θα προτείνει σε συσκευές κατηγοριοποιήσεις βάση στατιστικών στοιχείων.
- Memory cleaner. Η εφαρμογή λόγο της συνεχής παρακολούθησης του δικτύου της συσκευής πρέπει να αποκτήσει περισσότερες πολιτικές για την διαχείριση των δεδομένων και της μνήμης ή ακόμα και να χωριστεί σε μικρότερες διαφορετικές πιο ελαφριές εφαρμογές.

6.3 Η εργασία σαν project

Από την πρώτη στιγμή η εργασία αυτή αντιμετωπίστηκε σαν project. Σύμφωνα με το PMI, ένα project πρέπει να έχει ένα σαφές σημαντικό σκοπό και αυτός ο σκοπός να εκπληρωθεί μέσα σε ένα χρονικό διάστημα το οποίο έχει αρχή και τέλος.

Η αρχή δόθηκε πριν περίπου ένα χρόνο με την ανάθεση της. Μετά από 2 μήνες και αρκετή ανάλυση τον εναλλακτικών καταγράφτηκε ένα σαφές scope που είναι και ο τίτλος της εργασίας αυτής και ξεκίνησε η δουλειά . Μετά από προσπάθεια αρκετών μηνών για την κατανόηση της εφαρμογής που δόθηκε ως βάση, κάναμε τις απαραίτητες προσθήκες κωδικά ώστε να πετύχει ο στόχος μας. Με το παρόν έγγραφο προσπαθούμε να κρατήσουμε documentation, να καταγράψουμε τα παραδοτέα , τις δοκιμές που κάναμε και τέλος να δώσουμε ένα κλείσιμο. Ένα κλείσιμο που θα δοθεί με την παράδοση και την βαθμολόγηση της εργασίας αυτής.

7 Επίλογος

Η εργασία ήταν για εμάς μια μοναδική εμπειρία καθώς οι προκλήσεις ήταν τεράστιες. Από την μια μεριά η σχεδόν μηδενική επαφή με την Java την οποία σχεδόν δεν είχαμε δει μέχρι να ξεκινήσει η ενασχόληση μας με τον proxy που μας δόθηκε να αλλάξουμε.Από την άλλη η συνεχείς καινούριες ιδέες - δυνατότητες που μας έδινε ο proxy και το android studio σε συνδυασμό με τις νέες τεχνολογίες μας έκανε να μην χάσουμε στιγμή το ενδιαφέρον μας. Η Java μια πλήρης γλώσσα προγραμματισμού φτιαγμένη για κάθε λογής native εφαρμογή, μας εντυπωσίασε με το πλουραλισμό της και τις άπειρες βιβλιοθήκες που έχει για το android. Ο αντικειμενοστραφής χαρακτήρας της java και ο συνδυασμός της με την xml και την json έφτιαξε ένα πίνακα που νιώθαμε ότι μπορούμε να ζωγραφίσουμε όποιο πρόγραμμα θέλαμε. Πρέπει βέβαια να πούμε ότι η κατανόηση της Java μας πήρε μεγάλο μέρος της εργασίας μας, πράγμα όμως που δεν μετανιώνουμε σε καμία περίπτωση. Ο proxy που έπρεπε να αλλάξουμε είχε τεράστιες δυνατότητες όμως είχε στηριχτεί σε μια αρκετά διαδεδομένη και παλιά υλοποίηση η οποία μας δημιούργησε μεγάλα προβλήματα ειδικά αφού η ίδια η google πολλές φορές την αναγνώριζε σαν πιθανό spyware. Και εδώ βέβαια τα αποτελέσματα τις εργασίας είναι πολύτιμα καθώς ο κώδικας μας μπορεί να δουλέψει εύκολα με οποιαδήποτε proxy, waf, firewall βλέπει τα url. Σε κάθε περίπτωση η τεχνολογίες , τα framework και οι συναρτήσεις που χρησιμοποιήσαμε είναι το μέλλον της τεχνολογίας παρακολούθησης δικτύου. Τεχνολογίες που είναι προφανές ότι θα είναι για όλους μας το δεξί χέρι στο να βάλουμε σε τάξη το πως και γιατί χρησιμοποιούμε κάθε τι στο διαδίκτυο μέσω των δεκάδων smart συσκευών που χρησιμοποιούμε. Ευχαριστούμε το πανεπιστήμιο για αυτήν την εμπειρία.

8 Παραρτήματα

8.1 Περιγραφή του Server σε Swagger


```
    {
  "swagger": "2.0",
  "info": {
    "description": "This is a store server created to store data for the statistic operations of
the operando application",
    "version": "1.0.0",
    "title": "Swagger server store",
    "license": {
      "name": "Apache 2.0",
      "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
    }
  },
  "host": "localhost",
  "basePath": "/v1",
  "tags": [
    {
      "name": "sync",
      "description": "sync for statistics to use them (store them with a key)"
    },
    {
      "name": "appsgroup",
      "description": "manage apps in groups to aply policies "
    }
  ],
  "schemes": [
    "http"
  ],
  "paths": {
    "/statistics/{key}": {
      "get": {
       "tags": [
         "sync"
       ],
       "summary": "Get all statistics based on a key",
       "description": "",
       "operationId": "getpolicy",
       "produces": [
         "application/json"
       ],
       "parameters": [
         {
           "name": "key",
           "in": "path",
           "description": "ID of policy",
           "required": true,
           "type": "string"
         }
       ],
       "responses": {
         "200": {
           "description": "Succesful operation",
           "schema": {
             "type": "object",
             "properties": {
               "Statistics": {
                 "type": "array",
                 "items": {
                   "$ref": "#/definitions/StatisticPolicy"
                 }
               }
             }
           }
         }
```

```
        },
        "400": {
          "description": "Invalid input"
        },
        "404": {
          "description": "Key not found"
        }
      }
    },
    "put": {
      "tags": [
        "sync"
      ],
      "summary": "Add a new policy statistic",
      "description": "",
      "operationId": "addpolicy",
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "key",
          "in": "path",
          "description": "ID of policy",
          "required": true,
          "type": "string"
        },
        {
          "in": "body",
          "name": "body",
          "description": "policy statistic object that needs to be added to the store",
          "required": true,
          "schema": {
            "$ref": "#/definitions/StatisticPolicy"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "Successfull"
        },
        "405": {
          "description": "Invalid input"
        }
      }
    }
  },
  "/statistics/{key}/{domainurl}": {
    "get": {
      "tags": [
        "sync"
      ],
      "summary": "Get one statistic for a domain and a key",
      "description": "",
      "operationId": "getpolicyurl",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "key",
          "in": "path",
          "description": "ID of policy to create",
          "required": true,
          "type": "string"
```

```
      },
      {
        "name": "domainurl",
        "in": "path",
        "description": "statistic domain",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "Succesful operation",
        "schema": {
          "$ref": "#/definitions/StatisticPolicy"
        }
      },
      "400": {
        "description": "Invalid input"
      },
      "404": {
        "description": "Key-domain not found"
      }
    }
  },
  "post": {
    "tags": [
      "sync"
    ],
    "summary": "Update an existing statistic",
    "description": "",
    "operationId": "updatePeturl",
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "key",
        "in": "path",
        "description": "ID of policy statistic",
        "required": true,
        "type": "string"
      },
      {
        "name": "domainurl",
        "in": "path",
        "description": "statistic domain",
        "required": true,
        "type": "string"
      },
      {
        "in": "body",
        "name": "body",
        "description": "policy statistic object that needs to be added to the store",
        "required": true,
        "schema": {
          "$ref": "#/definitions/StatisticPolicy"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "Successfull"
      },
      "405": {
        "description": "Invalid input"
```

```
        }
      }
    }
  },
  "/appsIdentity/groups": {
    "get": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Get all app groups",
      "description": "",
      "operationId": "getgroups",
      "produces": [
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "Succesful operation",
          "schema": {
            "$ref": "#/definitions/AppIdentityGroups"
          }
        },
        "400": {
          "description": "Invalid input"
        },
        "404": {
          "description": "Key not found"
        }
      }
    }
  },
  "/appsIdentity/{group}": {
    "get": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Get all apps identities of a group",
      "description": "",
      "operationId": "getgroup",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "group",
          "in": "path",
          "description": "Group of apps",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "Succesful operation",
          "schema": {
            "type": "object",
            "properties": {
              "AppIdentities": {
                "type": "array",
                "items": {
                  "$ref": "#/definitions/AppIdentityGroup"
                }
              }
            }
          }
        },
        "400": {
          "description": "Invalid input"
```

```
        },
        "404": {
          "description": "Key not found"
        }
      }
    },
    "put": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Add a new policy or update one",
      "description": "",
      "operationId": "addgroup",
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "group",
          "in": "path",
          "description": "ID of policy to create",
          "required": true,
          "type": "string"
        },
        {
          "in": "body",
          "name": "body",
          "description": "Pet object that needs to be added to the store",
          "required": true,
          "schema": {
            "$ref": "#/definitions/AppIdentityGroup"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "Successfull"
        },
        "405": {
          "description": "Invalid input"
        }
      }
    }
  },
  "/appsIdentity/{group}/{app}": {
    "get": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Get a group app identity",
      "description": "",
      "operationId": "getgroupapp",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "group",
          "in": "path",
          "description": "ID of policy",
          "required": true,
          "type": "string"
        },
        {
          "name": "app",
```

```
          "in": "path",
          "description": "app of policy",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "Succesful operation",
          "schema": {
            "type": "object",
            "properties": {
              "AppIdentities": {
                "type": "array",
                "items": {
                  "$ref": "#/definitions/AppIdentityGroup"
                }
              }
            }
          }
        },
        "400": {
          "description": "Invalid input"
        },
        "404": {
          "description": "Key not found"
        }
      }
    }
  },
  "/appsIdentity/{group}/{app}/{domainurl}": {
    "get": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Get an app identity url",
      "description": "",
      "operationId": "getgroupappdomain",
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "group",
          "in": "path",
          "description": "ID of policy",
          "required": true,
          "type": "string"
        },
        {
          "name": "app",
          "in": "path",
          "description": "app of policy",
          "required": true,
          "type": "string"
        },
        {
          "name": "domainurl",
          "in": "path",
          "description": "domain of policy",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "Succesful operation",
          "schema": {
```

```
            "$ref": "#/definitions/AppIdentityGroup"
          }
        },
        "400": {
          "description": "Invalid input"
        },
        "404": {
          "description": "Key not found"
        }
      }
    },
    "put": {
      "tags": [
        "appsgroup"
      ],
      "summary": "Add a new policy",
      "description": "",
      "operationId": "addgroupapp",
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "parameters": [
        {
          "name": "group",
          "in": "path",
          "description": "ID of policy to create",
          "required": true,
          "type": "string"
        },
        {
          "name": "app",
          "in": "path",
          "description": "app of policy",
          "required": true,
          "type": "string"
        },
        {
          "name": "domainurl",
          "in": "path",
          "description": "domain of policy",
          "required": true,
          "type": "string"
        },
        {
          "in": "body",
          "name": "body",
          "description": "Pet object that needs to be added to the store",
          "required": true,
          "schema": {
            "$ref": "#/definitions/AppIdentityGroup"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "Successfull"
        },
        "405": {
          "description": "Invalid input"
        }
      }
    },
    "post": {
      "tags": [
        "appsgroup"
```

```
        ],
        "summary": "Update an existing app identity domain",
        "description": "",
        "operationId": "updateGroupdomain",
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "group",
            "in": "path",
            "description": "ID of policy to create",
            "required": true,
            "type": "string"
          },
          {
            "name": "app",
            "in": "path",
            "description": "app of policy",
            "required": true,
            "type": "string"
          },
          {
            "name": "domainurl",
            "in": "path",
            "description": "domain of policy",
            "required": true,
            "type": "string"
          },
          {
            "in": "body",
            "name": "body",
            "description": "Pet object that needs to be added to the store",
            "required": true,
            "schema": {
              "$ref": "#/definitions/AppIdentityGroup"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "Successfull"
          },
          "400": {
            "description": "Invalid ID supplied"
          },
          "404": {
            "description": "App identity url not found"
          },
          "405": {
            "description": "Validation exception"
          }
        }
      }
    }
  },
  "definitions": {
    "StatisticPolicy": {
      "type": "object",
      "required": [
        "name",
        "domainurl",
        "count"
      ],
      "properties": {
```

```
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "name": {
        "type": "string",
        "example": "policyA"
      },
      "domainurl": {
        "type": "string",
        "example": "www.enikos.gr"
      },
      "count": {
        "type": "integer",
        "format": "int64",
        "example": 2
      },
      "modified": {
        "type": "string",
        "example": "2001.07.04 AD at 12:08:56 PDT"
      },
      "sourceactivity": {
        "type": "string",
        "example": "operando"
      },
      "hidden": {
        "type": "string",
        "example": "no"
      },
      "category": {
        "type": "string",
        "example": "news"
      },
      "extrainfo": {
        "type": "string",
        "example": "{'for initialatation':'yes'}"
      }
    }
  },
  "AppIdentityGroup": {
    "type": "object",
    "required": [
      "name",
      "app_name",
      "count"
    ],
    "properties": {
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "name": {
        "type": "string",
        "example": "initialgroup"
      },
      "domainurl": {
        "type": "string",
        "example": "www.enikos.gr"
      },
      "app_name": {
        "type": "string",
        "example": "ClashOfClanshGame"
      },
      "count": {
        "type": "integer",
        "format": "int64",
        "example": 2
      },
```

```
        "duration": {
          "type": "integer",
          "format": "int64",
          "example": 21
        }
      }
    },
    "AppIdentityGroups": {
      "type": "object",
      "properties": {
        "apps": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "name": {
                "type": "string",
                "example": "initialgroup"
              }
            }
          }
        }
      }
    }
  }
}
```

9 Βιβλιογραφία - Πηγές

https://stackoverflow.com/

https://developer.android.com/

http://www.techrepublic.com/blog/it-security/the-basics-of-using-a-proxy-server-for-privacy-and-security/

https://en.wikipedia.org/wiki/Proxy_server

https://github.com/