Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εξατομικευμένος Έλεγχος Λογισμικού Φορητών Συσκευών** **Personalized Mobile Software Testing**
Ονοματεπώνυμο Φοιτητή	**Γεώργιος Αρκουλέας**
Πατρώνυμο	**Ανδρέας**
Αριθμός Μητρώου	**ΜΠΠΛ/ 17003**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευάγγελος Σακκόπουλος Επίκουρος Καθηγητής	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Διαγραμμάτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Σακκόπουλο για την ανάθεση της μεταπτυχιακής μου διατριβής και της εξαιρετικής του πρότασης που με προάγει τόσο σε γνωστικό όσο και σε επαγγελματικό επίπεδο, τον αγαπημένο φίλο, συνάδελφο και συμφοιτητή Γκίνη Δημήτριο (Android Developer) που μου διέθεσε την εφαρμογή και τον πηγαίο κώδικά της ώστε να χρησιμοποιηθεί το εργαλείο που επέλεξα, τους συναδέλφους μου Πηγαδά Βασίλειο (Senior Android Developer) και Στάικο Γεώργιο (Senior Software Developer) για την επίβλεψη και την επιστημονικά άρτια καθοδήγησή τους κατά τη διάρκεια του σχεδιασμού και της ανάπτυξης αυτής της εργασίας καθώς επίσης και για τις καίριες συμβουλές που μου έδωσαν για τη βελτίωση του εργαλείου που χρησιμοποιήθηκε και τις επεμβάσεις που χρειάστηκε να γίνουν σε επίπεδο κώδικα.

Τέλος, ευχαριστίες οφείλω στην οικογένειά μου, στη σύντροφό μου και στους φίλους μου για την ηθική υποστήριξη και βοήθεια σε όλα τα χρόνια των σπουδών, όπως επίσης για την κατανόηση και την ενθάρρυνση κατά τη διάρκεια εκπόνησης της μεταπτυχιακής μου διατριβής.

Περίληψη

Στόχος της παρούσας Μεταπτυχιακής Διατριβής αποτελεί η χρήση και παρουσίαση βασικών δυνατοτήτων του εργαλείου «Android Espresso». Αρχικά παρατίθενται συνοπτικά εισαγωγικές πληροφορίες για το λειτουργικό σύστημα Android, καθώς και τα αποτελέσματα της έρευνας για τέτοιου είδους εργαλεία. Στην συνέχεια στα πλαίσια της εργασίας, παρουσιάζεται το στήσιμο και οι δυνατότητες ενός τέτοιου εργαλείου που προωθούν τον όρο «testing» σε ανώτερο επίπεδο από αυτό του χειροκίνητου που είναι γνωστό μέχρι σήμερα αλλά και μια σύντομη παρουσίαση της εφαρμογής που χρησιμοποιήθηκε για την παρουσίαση. Τέλος περιγράφονται λεπτομερώς και αναλύονται όλα τα σενάρια που χρησιμοποιήθηκαν για τον έλεγχο της εφαρμογής που επιλέχθηκε καθώς και συμπεράσματα τα οποία προέρχονται από την δημιουργία ενός πλάνου ελέγχου (Test Plan) που ακολουθήθηκε.

Abstract

The purpose of this Master Thesis is the use and presentation of basic features of the "Android Espresso" tool. The first is a brief introduction to the Android operating system, as well as the results of research on such tools. Then in the context of the project, the setting up and possibilities of such a tool are presented, which promote the term "testing" at a higher level than that of the manual, known to date, as well as a brief presentation of the application used for the implementation.

1. Εισαγωγή

Στη σημερινή εποχή η ραγδαία αύξηση των κινητών συσκευών και όχι μόνο, αλλά και η ανάγκη για υπερπληροφόρηση, έχει οδηγήσει στην αυξανόμενη εμφάνιση σύγχρονων ηλεκτρονικών εφαρμογών και υπηρεσιών, που έχουν ως στόχο να διευκολύνουν την απομακρυσμένη επικοινωνία και να παρέχουν στους χρήστες τη δυνατότητα να πλοηγηθούν και να λάβουν την απαραίτητη πληροφορία που επιθυμούν ταχύτατα. Σε αυτό έχει συμβάλει ιδιαίτερα το διαδίκτυο και η άνθιση της υπολογιστικής νέφους(cloud computing) τα τελευταία χρόνια, τεχνολογίες οι οποίες έχουν δώσει τη δυνατότητα στο τελικό χρήστη σε οποιοδήποτε σημείο του κόσμου βρίσκεται να έχει άμεσα αποτελέσματα και επαρκή πληροφόρηση.

Στην παρούσα μεταπτυχιακή διατριβή θα παρουσιαστεί η ανάπτυξη ενός εργαλείου αυτοματοποιημένου ελέγχου μιας εφαρμογής σε περιβάλλον Android για την αναζήτηση ταινιών και πώς οι νέες τεχνολογίες «testing» θα βοηθήσουν στη βελτίωση του τελικού αποτελέσματος ανάπτυξης μιας εφαρμογής αλλά και στη ποιότητά της.

Στο παρόν κεφάλαιο θα αναφερθούμε στο σκοπό εκπόνησης της διπλωματικής εργασίας, στο χώρο προβλήματος και σε παρόμοια εργαλεία που συνδράμουν στον ίδιο σκοπό

1.1 ΧΩΡΟΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Τις προηγούμενες δεκαετίες όταν ακόμα η τεχνολογία δεν είχε παρεισφρήσει στη καθημερινότητά μας, δεν υπήρξε και η ανάγκη χρήσης τέτοιου είδους εργαλείων όπως το «Android Espresso» καθώς δεν υπήρχε και κάποιο αντικείμενο εφαρμογής τους.

Στη συνέχεια, με την πάροδο των ετών και με τη πρόοδο της τεχνολογίας σε επίπεδο «Software» αλλά και «Hardware» τα προαναφερθέντα, σταδιακά άρχισαν να κάνουν την εμφάνισή τους, φυσικά σε αρκετά εξειδικευμένες περιπτώσεις και με λίγους ανθρώπους να διαθέτουν την κατάρτιση και τη γνώση να τα χρησιμοποιήσουν. Με την ραγδαία εξέλιξη και των φορητών συσκευών δημιουργήθηκαν οι κατάλληλες εφαρμογές σε περιβάλλον Android/iOS/Windows και όχι μόνο, με συνέπεια τη δημιουργία πρόσφορου εδάφους αλλά και

της ανάγκης για χρήσης βελτιωμένου «testing» καθώς οι απαιτήσεις της αγοράς ολοένα και μεγάλωναν.

1.2 ΣΤΟΧΟΙ ΤΗΣ ΕΡΓΑΣΙΑΣ

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής αποτελεί η μελέτη και παρουσίαση ενός τέτοιου εργαλείου μέσα από το φιλικό περιβάλλον που παρέχεται από το λειτουργικό σύστημα Android καθώς διαθέτει ένα τέτοιο αυτοματοποιημένο εργαλείο ενσωματωμένο στις βιβλιοθήκες του. Το βασικό χαρακτηριστικό του επιλεχθέντος εργαλείου είναι η μεγάλη ευκολία που παρέχει στο χρήστη του κυρίως σε επίπεδο «UI(User Interface) Testing», ώστε να βελτιστοποιεί στο μέγιστο την ποιότητα της τελικής εφαρμογής και να αποφεύγονται μελλοντικά προβλήματα. Η δυνατότητα εκτέλεσης σεναρίων σε κάθε κομμάτι της εκάστοτε εφαρμογής γίνεται με πολύ φιλικό προς το χρήστη τρόπο καθώς και αρκετά γρήγορο. Το εργαλείο μπορεί να ελέγξει σε μεγάλο ποσοστό μεμονωμένα κομμάτια της εφαρμογής αλλά και περεταίρω σημεία που αφορούν τη συσκευή και χρησιμοποιούνται από τις εφαρμογές με την ανάλογη υλοποίηση μεθόδων ή κλήση άλλων βιβλιοθηκών.

2. Ιστορία και Εύρος του Software Testing

Σε αυτό το κεφάλαιο θα αναφερθεί η αρχή της έννοιας του λογισμικού χειροκίνητων και αυτοματοποιημένων δοκιμών (Software Testing) καθώς και το πως αυτό εντάχθηκε στο κύκλο ζωής του λογισμικού και των εφαρμογών (Software Life Cycle).

2.1 ΕΝΝΟΙΑ ΤΟΥ TESTING

Το «testing» (διαδικασία εκτέλεσης δοκιμών/ελέγχου λογισμικού) είναι στην ουσία μια διαδικασία κατά την οποία εκτελείται ένα πρόγραμμα ή και όχι με σκοπό την εύρεση λαθών και άλλων προβλημάτων σε λογισμικά και εφαρμογές. Επιπρόσθετα μπορεί να χρησιμοποιηθεί και για την δοκιμή συγκεκριμένων δυνατοτήτων των εκάστοτε εφαρμογών ώστε να αξιολογηθούν τα όρια που έχουν επιλεχθεί από τους προγραμματιστές ως προς το αν πληρούν τις προδιαγραφές.

Σκοπός της διαδικασίας του «testing» είναι κατά κύριο λόγο η βελτίωση της ποιότητας του λογισμικού ή των εφαρμογών που τίθενται υπό δοκιμή. Αξίζει να αναφερθεί ότι καμία διαδικασία ελέγχου λογισμικού ή εφαρμογών δεν μπορεί να εγγυηθεί απόλυτη προστασία από προβλήματα ή bugs μετά την παράδοση σε περιβάλλον παραγωγικό, μπορούν όμως να ελαχιστοποιηθούν σε μεγάλο βαθμό και αυτό έχει ως αποτέλεσμα την ικανοποίηση του εκάστοτε πελάτη και τη μείωση των εξόδων των εταιρειών που έχουν υιοθετήσει τη διαδικασία αυτή. Δεδομένο είναι ότι η διαδικασία του «testing» δε μπορεί να τελειώσει, θεωρητικά, συνεπώς το τέλος δίνεται από τους μηχανικούς που το εκτελούν θέτοντας έναν βασικό κανόνα, το λογισμικό που δοκιμάζεται να πληροί έστω τις βασικές προδιαγραφές του σχεδιασμού του χωρίς ενδιάμεσα προβλήματα.

Στη συνέχεια θα παρατεθούν και κάποιες βασικές κατηγορίες ελέγχου λογισμικού καθώς και κάποιες μέθοδοι που έχουν διαδοθεί περισσότερο τα τελευταία χρόνια (1).

2.2 ΙΣΤΟΡΙΑ ΚΑΙ ΕΞΕΛΙΞΗ ΕΛΕΓΧΟΥ ΛΟΓΙΣΜΙΚΟΥ (SOFTWARE TESTING)

Η έννοια του ελέγχου λογισμικού σχεδόν έφτασε και διαπιστώθηκε η ανάγκη της μαζί με την ανάπτυξη λογισμικού. Η πρώτη φορά που εκτελέστηκε ανάπτυξη λογισμικού ήταν στις 21 Ιουνίου του 1948 από τον Tom Kilburn στο Πανεπιστήμιο του Μάντσεστερ της Αγγλίας. Στην ουσία γράφτηκε ένα κομμάτι κώδικα και λογισμικού με τη βοήθεια μαθηματικών υπολογισμών αλλά και οδηγίες κώδικα μηχανής. Το «Debugging» που εκτελέστηκε και συνεχίστηκε για τις δύο επόμενες δεκαετίες μπορεί να θεωρηθεί ως η πρώτη πρώιμη μορφή testing η οποία χρησιμοποιείται και επιλέγεται σχεδόν μετά από την οποιαδήποτε ανάπτυξη λογισμικού σε κάθε μορφής εφαρμογή ή λογισμικό.

Περίπου στο 1980, εταιρείες της εποχής και ομάδες ανάπτυξης λογισμικού αποφάσισαν να διαχωρίσουν και να απομονώσουν περισσότερο την έννοια του «Testing» με σκοπό να γίνει ξεχωριστή διαδικασία ή και διαδικασίες οι οποίες θα έχουν σαν κύριο στόχο την επίλυση προβλημάτων που θα προλαμβάνονται πριν η τελική μορφή του προϊόντος φτάσει στο χρήστη/πελάτη. Με αυτή την επιλογή πολύ γρήγορα ενσωματώθηκε στον κύκλο ζωής του λογισμικού και παρέμεινε αναπόσπαστο κομμάτι μέχρι και σήμερα.

Το 1990 δόθηκε και η ονομασία «Quality Assurance» (διασφάλιση ποιότητας) η οποία είναι πλέον αρκετά διαδεδομένη και απαραίτητη σε κάθε φορέα που αναπτύσσει ή εμπορεύεται λογισμικό. Τέλος αξίζει να σημειωθεί ότι η διαδικασία του ελέγχου και των δοκιμών σήμερα δεν αναφέρεται καθαρά και μόνο στον έλεγχο και στις δοκιμές αλλά και σε μια διαδικασία σχεδιασμού και βημάτων, πολυσύνθετη με μεγάλο εύρος που προσφέρει τα επιθυμητά αποτελέσματα σε αρκετά μεγάλο ποσοστό εκτέλεσης της από το ανάλογο ανθρώπινο δυναμικό που διαθέτει τη γνώση, την εμπειρία και φυσικά τους πόρους.

Η επιτομή της εξέλιξης αυτής έφτασε μέσα στη τελευταία δεκαετία (2010-2020) στην οποία έχει εμφανιστεί και διαδίδεται με εκθετικό ρυθμό η έννοια του «Test Automation», χρησιμοποιώντας εξελιγμένο λογισμικό και την τεχνολογική αιχμή, εκτελούνται δοκιμές και έλεγχοι με ελαχιστοποιημένη την ανθρώπινη παρέμβαση με σκοπό την εκμηδένιση των προβλημάτων λογισμικού. Φυσικά αυτές οι τεχνολογίες απαιτούν εξειδικευμένη γνώση και αντίληψη πολλών εννοιών, καθώς και τα λογισμικά που εξυπηρετούν αυτό το σκοπό βρίσκονται

ακόμα σε πειραματικό ή πρώιμο στάδιο. Όπως όλα δείχνουν όμως μπορούμε να είμαστε πολύ αισιόδοξοι για το μέλλον καθώς γίνονται ολοένα και πιο φιλικά σε όποιον σκοπεύει να τα χρησιμοποιήσει και να ασχοληθεί (2).

2.3 ΚΑΤΗΓΟΡΙΕΣ ΕΛΕΓΧΟΥ ΛΟΓΙΣΜΙΚΟΥ (SOFTWARE TESTING)

Υπάρχουν διαφόρων ειδών έλεγχοι και δοκιμές στις οποίες μπορεί να υποβληθεί κάποια εφαρμογή ή ένα λογισμικό. Καθεμία από αυτές έχει διαφορετικούς στόχους και στρατηγική. Στη συνέχεια παρατίθενται οι συνηθέστερες μαζί με μια σύντομη περιγραφή-επεξήγηση: (2) (3)

- Acceptance Testing: Επιβεβαίωση ότι όλο το σύστημα που πατάει το εκάστοτε λογισμικό/εφαρμογή υπό δοκιμή λειτουργεί όπως προβλέπεται κυρίως το κομμάτι που αφορά τις «business» προδιαγραφές.
- Integration Testing: Επιβεβαίωση ότι τα συνεργαζόμενα κομμάτια και συναρτήσεις του εκάστοτε λογισμικού/εφαρμογής που ελέγχεται συνεργάζονται κανονικά.
- Unit Testing: Επιβεβαίωση ότι κάθε ένα ξεχωριστά επιμέρους κομμάτι του εκάστοτε λογισμικού/εφαρμογής που ελέγχεται λειτουργεί κανονικά. Αυτό το είδος ελέγχου μπορεί να φτάσει ακόμα στο μικρότερο κομμάτι που μπορεί να τεθεί υπό δοκιμή.
- Performance Testing: Έλεγχος και δοκιμή του εκάστοτε λογισμικού/εφαρμογής σε πραγματικές συνθήκες πίεσης και όγκου ώστε να διαπιστωθούν τα όρια και κατά πόσο αυτά πληρούνται.
- Smoke Testing: Εκτέλεση σύντομων και «high level» σεναρίων με σκοπό το γρήγορο έλεγχο και δοκιμή των βασικών λειτουργιών του λογισμικού ή της εφαρμογής. Συνήθως πραγματοποιείται αμέσως μετά την ολοκλήρωση του «development», συχνά εκτελείται και από τον ίδιο τον προγραμματιστή και τα αποτελέσματα καθορίζουν το αν θα τεθεί η νέα έκδοση σε βαθύτερο «testing» το οποίο συνεπάγεται και μεγαλύτερο κόστος ή αν θα βγει σε παραγωγικό περιβάλλον ως έχει.
- Regression Testing: Επιβεβαιώνεται με τη χρήση αυτής της μεθόδου ότι δεν επηρεάζονται υπάρχουσες λειτουργικότητες μετά τις προσθήκες νέων λειτουργιών και δυνατοτήτων της υπό ελέγχου εφαρμογής/λογισμικού.

- Sanity Testing: Επιβεβαιώνει για τις περιπτώσεις που δεν μπορεί να εκτελεστεί πλήρως το «Regression Testing» μέσω ελέγχων μενού, συναρτήσεων και εντολών (High Level) ότι η λειτουργικότητα παραμένει όπως έχει προβλεφθεί μετά από αλλαγές και προσθήκες δυνατοτήτων.
- Stress Testing: Η συγκεκριμένη κατηγορία αφορά ξεκάθαρα τον εντοπισμό του ορίου του λογισμικού/εφαρμογής υπό έλεγχο με μοναδικό στόχο να την οδηγήσει σε αποτυχία ώστε να το διαπιστώσει. Δεν χρησιμοποιείται σε παραγωγικό περιβάλλον και δεν αποτελεί διαδικασία λειτουργική όπως οι προαναφερθείσες παρά μόνο χρησιμοποιείται σε ιδιαίτερες περιπτώσεις.
- Usability Testing: Επιβεβαιώνεται μέσα από τη συγκεκριμένη κατηγορία το κατά πόσο ο πελάτης που χρησιμοποιεί το υπό δοκιμή λογισμικό/εφαρμογή έχει μια καλή και αρμονική εμπειρία ώστε να φτάσει στο τέλος κάποια διεργασία που εκείνος έχει επιλέξει.

2.4 ΜΕΘΟΔΟΙ ΚΑΙ ΤΕΧΝΙΚΕΣ

Όπως έχει ήδη αναφερθεί, η διαδικασία του ελέγχου λογισμικού και εφαρμογών απαρτίζεται και από συγκεκριμένες μεθόδους και τεχνικές οι οποίες αποτελούν πυλώνες καθώς και βρίσκουν ακόμα μεγαλύτερο εύρος εφαρμογής τους σε κάθε εργαλείο που έχει δημιουργηθεί για το σκοπός της διασφάλισης ποιότητας. Σε αυτό το κομμάτι θα δούμε τις κυριότερες και πιο διαδεδομένες από αυτές τις μεθόδους καθώς και λίγες πληροφορίες για τη καθεμία από αυτές:

- Black-box Testing: Κάνοντας χρήση της συγκεκριμένης μεθόδου, η οποία μπορεί να περιλαμβάνει διαφόρων ειδών τεχνικές που κατηγοριοποιούν τη διαδικασία εκτέλεσής της, στην ουσία η περισσότερη πληροφορία που σχετίζεται με τη δημιουργία του λογισμικού, τη δομή και τις αρχικές προδιαγραφές παραμένει κρυφή ενώ σαν στόχο έχει τη πληθώρα σεναρίων και ελέγχων για να έρθει το επιθυμητό αποτέλεσμα έχοντας το ελάχιστο δυνατό κόστος.
- White-box Testing: Με τη συγκεκριμένη μέθοδο, η διαδικασία που ακολουθείται είναι ακριβώς η αντίθετη της προαναφερθείσας. Δηλαδή όλη η πληροφορία που αφορά τη δομή και το σχεδιασμό του λογισμικού ή της εφαρμογής στην οποία θα

εκτελεστεί έλεγχος ποιότητας είναι εμφανής και διαθέσιμη με σκοπό την εξασφάλιση της αποτελεσματικότητας αλλά και την αύξηση του κόστους και του χρόνου. Όλος ο σχεδιασμός της διαδικασίας ελέγχου και των σεναρίων γίνεται επάνω σε λεπτομέρειες του προϊόντος με μικρότερες πιθανότητες αστοχίας συγκριτικά με κάθε άλλη μέθοδο.

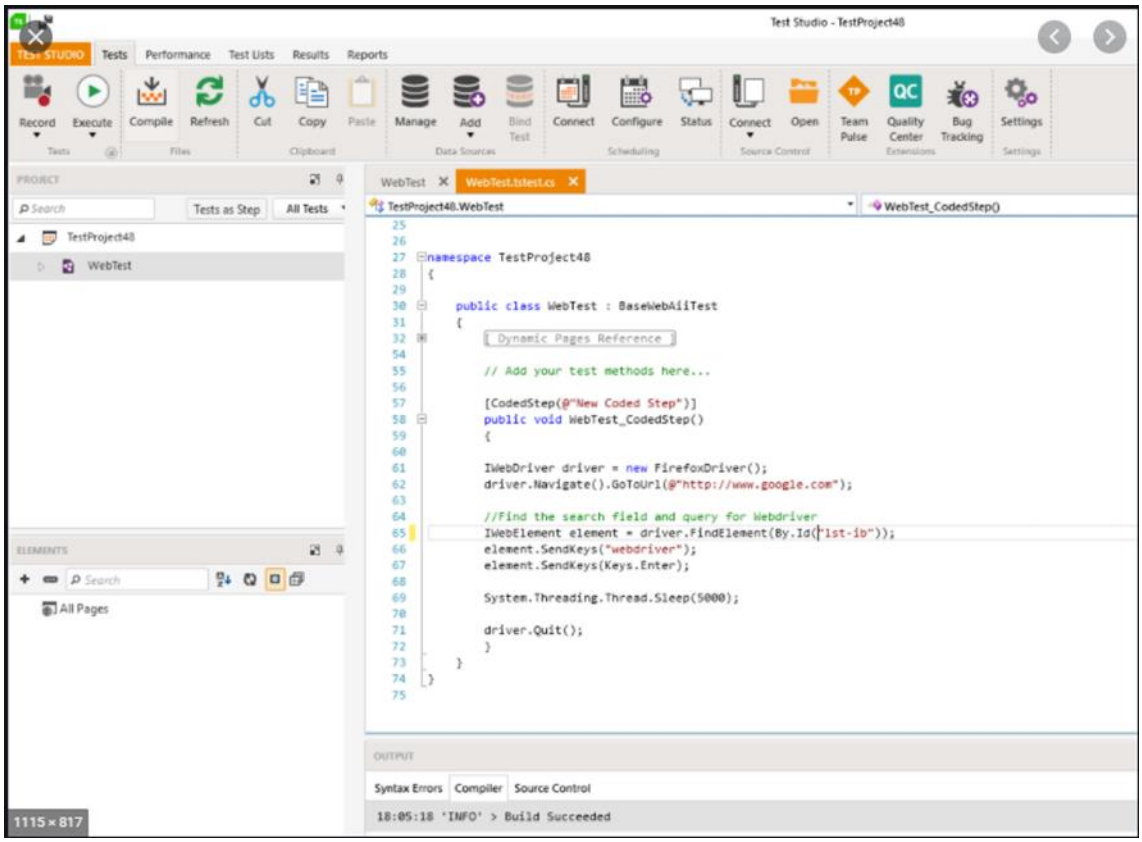
- Gray-box Testing: Η μέθοδος αυτή αποτελεί μια μίξη των δύο προηγούμενων καθώς τα εμφανή στοιχεία του λογισμικού που θα τεθεί σε έλεγχο και δοκιμές είναι ως ένα βαθμό προσβάσιμα χωρίς όμως να προσφέρονται με κάθε λεπτομέρεια για την εκτέλεση της διαδικασίας διασφάλισης ποιότητας. Αντίστοιχα κάπου στη μέση βρίσκονται και τα ανάλογα κόστη.
- Security Testing: Όπως είναι ξεκάθαρο και από το όνομα, η συγκεκριμένη μέθοδος βασίζεται στον έλεγχο και τις δοκιμές για τη προστασία των ευαίσθητων δεδομένων ενός λογισμικού ή εφαρμογής τα οποία μπορεί να διαρρεύσουν σε τρίτους. Αποτελεί μια από τις βασικότερες μεθόδους πάνω στην οποία πάντα δίνεται βάση καθώς δεν πρόκειται μόνο για την προστασία πηγαίου κώδικα αλλά και στοιχείων προσώπων ή ακόμα και εταιρειών. Συνεπώς πρέπει να εκτελείται με τον ανάλογο σχεδιασμό, τη προσοχή στη λεπτομέρεια και τον μέγιστο αριθμό σεναρίων ώστε να διασφαλιστεί η απόλυτη επιτυχία. Όπως είναι γνωστό τίποτα στην επιστήμη της πληροφορικής δε μπορεί να θεωρηθεί ασφαλές στο 100%, όμως οι πιθανότητες οποιασδήποτε διαρροής πρέπει να ελαχιστοποιηθούν στο μέγιστο με σκοπό την αποφυγή είτε επιθέσεων προσώπων είτε κακόβουλου λογισμικού ειδικά σχεδιασμένου για ενέργειες υποκλοπής (1) (2).

3. Εργαλεία Οργάνωσης Δοκιμών για Ανάπτυξη Εφαρμογών

Σε αυτό το κεφάλαιο θα παρουσιαστούν αντίστοιχες εφαρμογές, θα αναφερθούν οι καινοτομίες και οι παροχές τους. Όλες οι εφαρμογές αφορούν τα λειτουργικά συστήματα Android/iOS/Windows/Linux/macOS.

3.1 SELENIUM

Το Selenium είναι ένα φορητό «framework» για τη δοκιμή εφαρμογών ιστού. Το Selenium παρέχει ένα εργαλείο αναπαραγωγής για τη σύνταξη λειτουργικών δοκιμών χωρίς την ανάγκη εκμάθησης μιας γλώσσας δοκιμαστικής δέσμης ενεργειών (Selenium IDE). Παρέχει επίσης μια δοκιμαστική γλώσσα για συγκεκριμένους τομείς (Selenese) για τη σύνταξη δοκιμών σε διάφορες δημοφιλείς γλώσσες προγραμματισμού, όπως C #, Groovy, Java, Perl, PHP, Python, Ruby και Scala. Οι δοκιμές μπορούν στη συνέχεια να εκτελεστούν με τα περισσότερα σύγχρονα προγράμματα περιήγησης ιστού. Το Selenium εκτελείται σε Windows, Linux και macOS. Είναι λογισμικό ανοιχτού κώδικα που κυκλοφόρησε με το Apache License 2.0 (4).



Εικόνα 1. Selenium Tests με τη χρήση «Test Studio» (4)

Το εργαλείο χρησιμοποιείται από το 2012 που μέσα σε 2 χρόνια είχε ραγδαία άνοδο από την ευρύτερη κοινότητα. Σήμερα έχει αρκετά μεγάλη πτώση καθώς έχει αντικατασταθεί με άλλα εργαλεία και έχει διατηρήσει τα νούμερά του κυρίως στις «Web εφαρμογές».

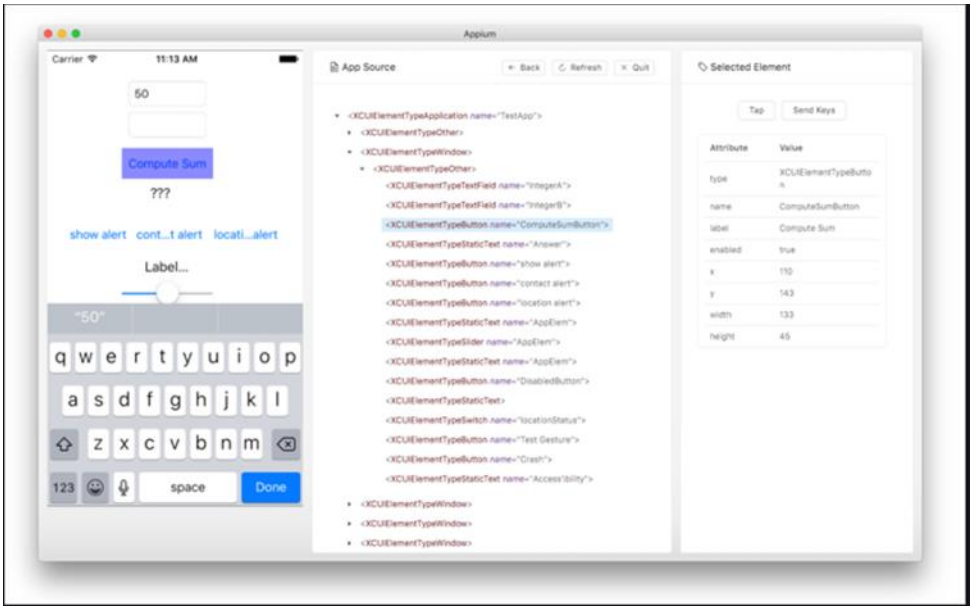
Παρακάτω μερικές από τις δυνατότητες του εργαλείου:

- Δυνατότητα Ανάγνωσης
- Δυνατότητα Ανάγνωσης – Γραφής
- Δυνατότητα Remote Web Driver
- Δυνατότητα Grid Specific πλέγματος
- Δυνατότητα Browser Specific φυλλομετρητών σε επίπεδο path
- Δυνατότητα ελέγχου UI σε επίπεδο φυλλομετρητή(Browser)
- Δυνατότητα ελέγχου κουμπιών και ρυθμίσεων

3.2 APPIUM

Το Appium είναι ένα εργαλείο ανοιχτού κώδικα για την αυτοματοποίηση native εφαρμογών, ιστού για κινητά και υβριδικών εφαρμογών, σε πλατφόρμες iOS για κινητά, Android για κινητά και υπολογιστές Windows. Οι native εφαρμογές είναι αυτές που έχουν γραφτεί χρησιμοποιώντας SDK iOS, Android ή Windows. Οι εφαρμογές ιστού για κινητά είναι προσβάσιμες εφαρμογές μέσω προγράμματος περιήγησης για κινητά (το Appium υποστηρίζει το Safari σε iOS και Chrome ή την ενσωματωμένη εφαρμογή «Πρόγραμμα περιήγησης» σε Android). Οι υβριδικές εφαρμογές έχουν ένα wrapper γύρω από μια "προβολή ιστού" - έναν native έλεγχο που επιτρέπει την αλληλεπίδραση με περιεχόμενο ιστού. Έργα όπως το Apache Cordova ή το Phonegap διευκολύνουν τη δημιουργία εφαρμογών χρησιμοποιώντας τεχνολογίες ιστού που στη συνέχεια συνδυάζονται σε ένα native wrapper, δημιουργώντας μια υβριδική εφαρμογή.

Είναι σημαντικό ότι το Appium είναι "cross-platform": σας επιτρέπει να γράφετε δοκιμές σε πολλές πλατφόρμες (iOS, Android, Windows), χρησιμοποιώντας το ίδιο API. Αυτό επιτρέπει την επαναχρησιμοποίηση κώδικα μεταξύ δοκιμών iOS, Android και Windows (5).



Εικόνα 2. Appium Tests με τη χρήση «Sauce Labs» (5)

Παρακάτω μερικές από τις δυνατότητες του εργαλείου:

- Δυνατότητα Ανάγνωσης
- Δυνατότητα Ανάγνωσης – Γραφής
- Δυνατότητα αυτοματοποιημένων ελέγχων σε native, hybrid, mobile και desktop εφαρμογές
- Δυνατότητα επικοινωνίας των εφαρμογών μεταξύ διαφορετικής πλατφόρμας

3.3 MOCKITO

Το Mockito αποτελεί ένα διάσημο mock framework το οποίο επιτρέπει τη διαμόρφωση και τη δημιουργία διαφόρων mock objects. Είναι ένα αρκετά εύχρηστο εργαλείο που απλοποιεί την εκτέλεση δοκιμών σε μέρη ενός κώδικα (Unit Testing).

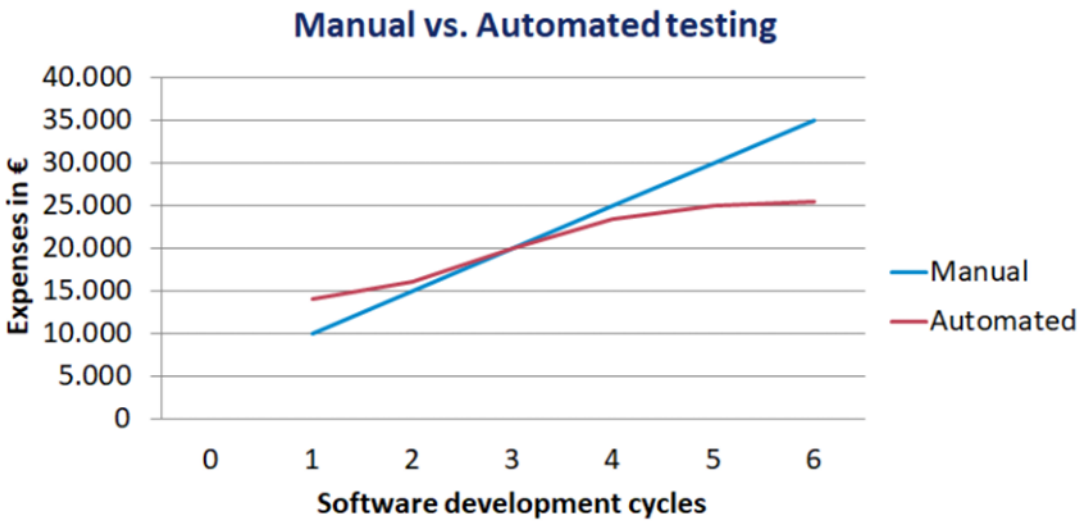
Εικόνα 3. Unit test με χρήση Mockito στον Android Test φάκελο (6)

Παρακάτω μερικές από τις δυνατότητες του εργαλείου:

- Απομόνωση των εξωτερικών dependencies και ενσωμάτωση των mocks στον υπό δοκιμή κώδικα
- Παράλληλη εκτέλεση και έλεγχος του υπό δοκιμή κώδικα
- Επαλήθευση του κώδικα που εκτελέστηκε σωστά

3.4 MANUAL VS AUTOMATED TESTING

Ο αυτοματοποιημένος έλεγχος εφαρμογών Android μειώνει το κόστος και εξοικονομεί πολύ χρόνο κατά τη διάρκεια των κύκλων ανάπτυξης. Η αρχική επένδυση ως προς το κόστος και τον χρόνο είναι υψηλή για αυτοματοποιημένο έλεγχο. Ωστόσο, μελέτες και αναλύσεις με την πάροδο των ετών έχουν αποδείξει ότι η απόδοση της δοκιμής μέσω των αντίστοιχων εργαλείων είναι πάντα υψηλότερη.



Εικόνα 4. QF Test Analysis (7)

Manual	Automated
Hours (10x135)= 1,350 hours	**Hours** (3x21x16x5) + (7x135) = Total of 5985 hours
Cost $78/hour	**Cost** $17.5/hour

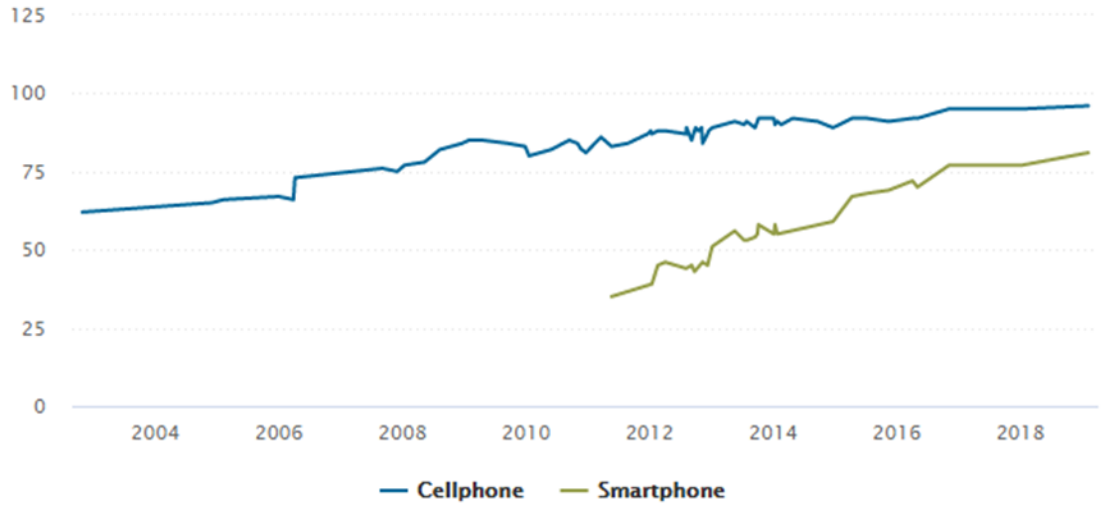
Εικόνα 5. Ανάλυση σε επίπεδο αριθμών (8)

4. Φορητές συσκευές

Οι φορητές συσκευές έχουν ενταχθεί πλέον για τα καλά στη ζωή μας. Στο ακόλουθο κεφάλαιο παρουσιάζονται οι δύο πιο διαδεδομένες φορητές συσκευές (“έξυπνες συσκευές”, tablets) με κάποια χαρακτηριστικά τους καθώς και οι ανάγκες που ικανοποιούν, οι οποίες ταυτίζονται αλλά και διαφέρουν σε μεγάλο βαθμό.

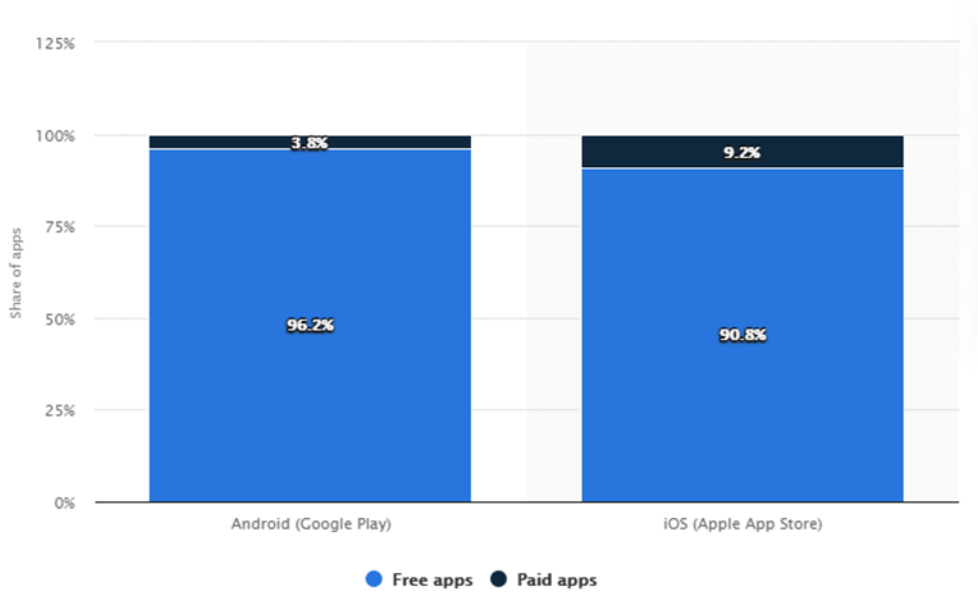
4.1 ΈΞΥΠΝΑ ΤΗΛΕΦΩΝΑ («ΕΞΥΠΝΕΣ ΣΥΣΚΕΥΕΣ»)

Τα πρώτα έξυπνα τηλέφωνα εμφανίστηκαν το 1984 με την μορφή των PDA, είχαν βασικές δυνατότητες ηλεκτρονικού υπολογιστή χωρίς όμως την ικανότητα να συνδεθούν στο διαδίκτυο. Το πρώτο κινητό τηλέφωνο με δυνατότητες PDA ήταν ένα πρωτότυπο IBM που κατασκευάστηκε το 1992 και διατέθηκε στο εμπόριο το 1994, ήταν η πρώτη συσκευή που μπορεί να αναφέρεται ως «έξυπνο τηλέφωνο», έστω και αν αυτός ο όρος δεν είχε ακόμη ευρέως χρησιμοποιηθεί. Είχε την δυνατότητα να πραγματοποιεί και να δέχεται κλήσεις καθώς επίσης να στέλνει και να λαμβάνει φαξ ή e-mails. Ο όρος «έξυπνο τηλέφωνο» δεν εμφανίστηκε ως το 1997, όταν η Ericsson περιέγραψε το GS 88 ως Smartphone (Έξυπνο Τηλέφωνο). Τα «έξυπνα τηλέφωνα» έγιναν δημοφιλή με την κυκλοφορία των λειτουργικών συστημάτων iOS της Apple το 2007 και του Android από τη Google το 2008 τα οποία και κατέχουν το μεγαλύτερο μερίδιο της αγοράς των «έξυπνων» κινητών συσκευών (9).



Διάγραμμα 1. Χρήστες «έξυπνων συσκευών» (https://www.pewresearch.org/internet/fact-sheet/mobile/) (10)

Βασικό στοιχείο των "έξυπνων συσκευών" είναι ότι έχουν πρόσβαση σε λειτουργίες και προγράμματα που μέχρι να εμφανιστούν είχαν μόνο οι υπολογιστές. Διαθέτουν οθόνες αφής, υποστήριξη 4G/5G τεχνολογίας, ενσωματωμένο Wi-Fi για ασύρματη σύνδεση δικτύου, αισθητήρες θέσης, κίνησης καθώς και φωτογραφική μηχανή. Ένα ακόμα πλεονέκτημα που έχουν οι "έξυπνες συσκευές", που τελικά τα έκανε να κυριαρχήσουν στην αγορά, είναι ότι διαθέτουν ηλεκτρονικά καταστήματα τα οποία προσφέρουν πληθώρα εφαρμογών, με τις περισσότερες να διατίθενται εντελών δωρεάν. Αυτό γίνεται σε ολοένα και περισσότερες εφαρμογές αλλά για να μπορέσει ο χρήστης να έχει πρόσβαση σε όλες τις λειτουργίες θα πρέπει να προβεί σε αγορά αυτών ή να παρακολουθήσει κάποιες διαφημίσεις. Ο παρακάτω πίνακας είναι μέρος έρευνας μέχρι τον Δεκέμβριο του 2020 όπου παρατηρούμε πληθώρα εφαρμογών οι οποίες είναι δωρεάν σε ηλεκτρονικά καταστήματα (9).



Διάγραμμα 2. Κατανομή δωρεάν και επί πληρωμή εφαρμογών στο App. Store & στο Play Store μέχρι το Δεκέμβριο 2020 (https://www.statista.com/) (11)

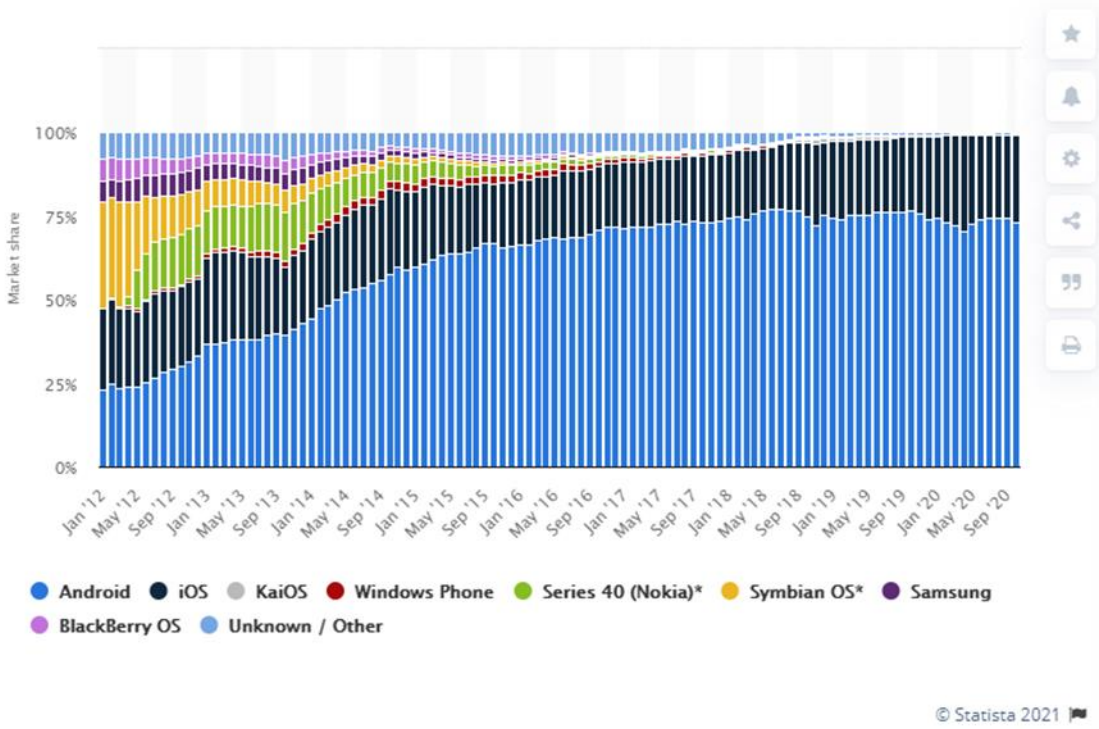
4.2 ΦΟΡΗΤΗ ΣΥΣΚΕΥΗ – ΤΑΜΠΛΕΤΑ (TABLET)

Μια συσκευή ταμπλέτα ή tablet διαθέτει τις δυνατότητες των "έξυπνων συσκευών" έχοντας μεγαλύτερη οθόνη από αυτά και αισθητά μικρότερη από τους σταθερούς ή φορητούς υπολογιστές. Τα tablets υπερτερούν των "έξυπνων τηλεφώνων" καθώς λόγω των μεγαλύτερων διαστάσεων διαθέτουν μεγαλύτερη μπαταρία, καλύτερη αυτονομία, διαθέτουν περισσότερες θύρες επέκτασης και είναι πιο εύχρηστα στις παράλληλες εργασίες (Multitasking). Τα λογισμικά στα «Tablets» είναι τα προαναφερθέντα, δηλαδή Android και iOS. Τα τελευταία χρόνια, λόγω της εξέλιξης της τεχνολογίας, οι ταμπλέτες είναι πλέον σε θέση να υποστηρίξουν πλήρως το λογισμικό Windows 10 της Microsoft και με την χρήση πληκτρολογίου μετατρέπεται σε φορητό υπολογιστή ακόμα και για επαγγελματική χρήση (9).

4.3 ΛΟΓΙΣΜΙΚΑ ΓΙΑ ΈΞΥΠΝΑ ΤΗΛΕΦΩΝΑ

Τα λογισμικά για έξυπνα τηλέφωνα έχουν ικανότητες τις οποίες έχουν οι ηλεκτρονικοί υπολογιστές με τις περισσότερες είναι χρήσιμες στις σύγχρονες φορητές συσκευές. Τη περίοδο που διανύουμε οι επιλογές λογισμικών είναι σχετικά περιορισμένες, τα προηγούμενα χρόνια, όταν τα "έξυπνα τηλέφωνα" πρωτοεμφανίστηκαν, κάθε κατασκευαστής φορητών συσκευών

είχε το δικό του λογισμικό για την εκάστοτε συσκευή. Τα λογισμικά που κυριάρχησαν είναι τα Android, iOS και το Windows Phone με το τελευταίο να μην καταφέρει να ανταγωνιστεί τα υπόλοιπα λογισμικά, συνεπώς σταμάτησε η κυκλοφορία του τον Οκτώβριο του 2017. Έως τον Σεπτέμβριο του 2020 το Android κατείχε το 78% της αγοράς ενώ το iOS το 22%.



Διάγραμμα 3. Κατανομή λογισμικών στην αγορά από το Γενάρη του 2012 ως το Σεπτέμβρη 2020 (https://www.statista.com/) (11)

Το γεγονός ότι η πλατφόρμα Android έχει το κατά πολύ μεγαλύτερο μερίδιο της αγοράς καθιστά την απασχόληση αλλά και ανάπτυξη εφαρμογών σε αυτό το λογισμικό ιδιαίτερα ασφαλής επιλογή (λόγω της χρήσης της) και με αρκετές προοπτικές για το μέλλον. Στη συνέχεια θα γίνει μια σύντομη αναφορά στα λογισμικά που υπάρχουν στην αγορά αυτήν τη περίοδο καθώς και του Windows Phone που πλέον η ανάπτυξή του έχει διακοπεί (11) (9).

4.3.1 ANDROID

Είναι το λογισμικό που αναπτύχθηκε από τη Google και θα μας απασχολήσει στη παρούσα διπλωματική διατριβή. Παρατηρώντας το παραπάνω διάγραμμα, είναι η γρηγορότερα αναπτυσσόμενη πλατφόρμα και χρησιμοποιείται από αρκετούς κατασκευαστές. Στην αγορά

διατίθενται πολλές διαφορετικές φορητές συσκευές σε διάφορες κατηγορίες τιμών, προσαρμοζόμενες στις απαιτήσεις και τις ανάγκες του κάθε χρήστη. Περισσότερο θα αναφερθούμε στα παρακάτω κεφάλαια.

4.3.2 iOS

Το 2007 παρουσιάστηκε το πρώτο iPhone από την Apple το οποίο σαν συσκευή αποτελούσε πραγματικά επανάσταση και την αρχή μιας νέας γενιάς έξυπνων φορητών συσκευών. Η αντικατάσταση του φυσικού πληκτρολογίου με την οθόνης αφής ήταν αυτό που ξεχώρισε στο μεγαλύτερο βαθμό καθώς και πολλές λειτουργίες που έκαναν ευχάριστη και πιο φιλική την ενασχόληση με τη συσκευή στο χρήστη.

Η Apple παρέχει στις συσκευές της το δικό της λογισμικό, το iOS. Σήμερα έχει φτάσει στην έκδοσή 14.4 έχοντας αναπτύξει ειδικά τροποποιημένο λογισμικό για τη ταμπλέτα της το πλέον γνωστό iPad. Το λογισμικό πατάει επάνω στο macOS, ο κώδικας γράφεται στις γλώσσες Objective-C και Swift. Το λογισμικό αυτό είναι κλειστού κώδικα και οι συσκευές που διατίθενται, μόνο για το συγκεκριμένο λειτουργικό, κατασκευάζονται αποκλειστικά από την Apple (9).

Εικόνα 6. Το iOS και οι συσκευές της Apple. (https://www.apple.com) (12)

4.3.3 WINDOWS PHONE/MOBILE

Προσπαθώντας η Microsoft να αποσπάσει ένα μερίδιο στην αγορά από τη Google και την Apple, το 2010 παρουσίασε το λογισμικό Windows Phone. Είναι μια εξέλιξη των Windows Mobile που ως τότε χρησιμοποιούνταν από PDA. Η ανάπτυξη έγινε σε αυτό το λειτουργικό σύστημα μέσω των γλωσσών C#, Visual Basic, και C++. Η Microsoft με τα Windows Phones,

προσπάθησε να ενσωματώσει το γραφικό περιβάλλον που χρησιμοποιούσε ήδη στους σταθερούς υπολογιστές στα έξυπνα τηλέφωνά της, έτσι ώστε οι χρήστες να έχουν τη μέγιστη εξοικείωση με το λογισμικό της. Δυστυχώς δεν πήγαν όλα όπως είχε προβλεφθεί και το 2015 η Microsoft ανακοίνωσε την τελευταία έκδοση του λογισμικού της, το Windows Phone 10. Ο περιορισμός στο πλήθος των εφαρμογών και ο υψηλός ανταγωνισμός ήταν οι κύριες αιτίες που τα Windows Phone έφτασαν στο τέλος. Τον Οκτώβριο του 2017 ανακοίνωσε επίσημα η Microsoft τη παύση της ανάπτυξης και της συνέχειας των Windows Mobiles (9).

Εικόνα 7. Το Windows Phone σε Nokia συσκευές. (https://www.theverge.com) (13)

4.4 ANDROID ΚΑΙ IOS

Η Google μέσα από την εξέλιξη του Android κατάφερε να ξεπεράσει σε πωλήσεις την Apple και το iOS. Κύριο χαρακτηριστικό σε αυτή τη ραγδαία εξάπλωση είναι η δωρεάν διάθεση του λογισμικού της. Με την ένταξη των ολοένα και περισσότερων κατασκευαστών, αυξήθηκε η ποικιλία στην αγορά με τις τιμές να κυμαίνονται από τα 50€ ως τα 1400€, έτσι ο υποψήφιος αγοραστής έχει πληθώρα επιλογών για την αγορά ενός «έξυπνου τηλεφώνου». Σημαντικό ρόλο έχει παίξει και η τιμολογιακή πολιτική της Apple δίνοντας μεγαλύτερη ώθηση στην αγορά

Android συσκευών, καθώς με αρκετά λίγα χρήματα συγκριτικά, είχαν ισάξιες δυνατότητες και χαρακτηριστικά. Παρακάτω θα αναφερθούμε εν συντομία στα θετικά και αρνητικά που έχει το κάθε λογισμικό.

4.4.1 ΘΕΤΙΚΑ ΤΟΥ ANDROID

Το κυριότερο θετικό χαρακτηριστικό του Android είναι πληθώρα δυνατοτήτων που δίνει στο χρήστη. Έχοντας μεγάλη ποικιλία συσκευών ο χρήστης μπορεί να αποκτήσει μια συσκευή στις απαιτήσεις του στοχεύοντας στο χαρακτηριστικό που έχει περισσότερη σημασία (κάμερα, οθόνη, μπαταρία κλπ.). Επιπρόσθετα, προσφέρεται στο τελικό χρήστη μια τεράστια ποικιλία εφαρμογών (η πλειοψηφία των οποίων είναι δωρεάν). Από τεχνικής άποψης, η ανάπτυξη εφαρμογών είναι αρκετά πιο εύκολη καθώς μπορεί κάποιος να προγραμματίσει από οποιοδήποτε ηλεκτρονικό υπολογιστή (οι εφαρμογές στο iOS απαιτούν συσκευές της Apple για την ανάπτυξή τους) και μπορούν να γραφτούν στις γλώσσες Java και Kotlin με την πρώτη να αποτελεί μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού (14) (9).

4.4.2 ΘΕΤΙΚΑ ΤΟΥ iOS

Το πιο σημαντικό θετικό χαρακτηριστικό του iOS είναι ότι είναι πιο ασφαλές και σταθερό σαν λογισμικό καθώς είναι φτιαγμένο για να δουλεύει σε μικρό αριθμό συσκευών. Επιπρόσθετα το λειτουργικό αυτό είναι σχεδόν πάντα ενημερωμένο καθώς η Apple έχει δεσμευτεί να προσφέρει ενημερώσεις για τουλάχιστον 3 χρόνια από τη κάθε αγορά συσκευής. Επίσης αρκετοί χρήστες έχουν αγαπήσει και εξοικειωθεί με το οικοσύστημα που τους προσφέρεται, το οποίο στην συνέχεια τους κάνει σταθερούς στην επιλογή του με αποτέλεσμα να αυξάνονται οι πωλήσεις των Apple συσκευών στο σύνολο. Τέλος οι Apple συσκευές είναι αρκετά προσεγμένες τόσο στην εμφάνιση όσο και στα χαρακτηριστικά που προσφέρουν στο καταναλωτή χωρίς αυτό να δικαιολογεί τις πολύ υψηλές τιμές (12) (9).

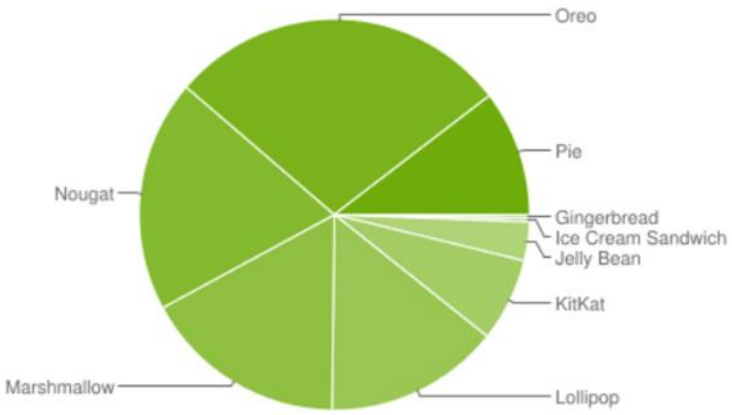
4.4.3 ΑΡΝΗΤΙΚΑ ΤΟΥ ANDROID

Στο Android η μεγάλη ποικιλία χαρακτηριστικών αλλά και διαφορετικών συσκευών όπως για παράδειγμα οθόνες διαφόρων διαστάσεων και αναλύσεων και οι πολυάριθμοι αισθητήρες αποτελούν στοιχεία που οι προγραμματιστές καλούνται να λαμβάνουν υπόψη στην ανάπτυξη

των εφαρμογών ώστε αυτές να εκτελούνται αποτελεσματικά και ομοιόμορφα σε όλες τις συσκευές. Αυτό απαιτεί επιπλέον δουλειά και χρόνο από τεχνικής πλευράς (14) (9).

Διάγραμμα 4. Μεγέθη οθονών συσκευών και πυκνότητα (https://developer.android.com/about/dashboards) (14)

Το κυριότερο αρνητικό χαρακτηριστικό του Android συγκριτικά με το iOS είναι το πρόβλημα του κατακερματισμού (Fragmentation). Ο κατακερματισμός στα λογισμικά φορητών συσκευών αφορά τις περιπτώσεις στις οποίες οι χρήστες χρησιμοποιούν στις συσκευές τους παλαιότερες εκδόσεις του λειτουργικού συστήματος, ενώ άλλοι νεότερες. Το πρόβλημα αυτό εμφανίζεται καθώς κάποιοι κατασκευαστές παύουν να ενημερώνουν τις συσκευές με τις μεταγενέστερες εκδόσεις εξαιτίας περιορισμού των δυνατοτήτων των συσκευών ή λόγω της παλαιότητάς τους. Επιπρόσθετα η εξέλιξη του λογισμικού φέρνει και κάποιες δυνατότητες οι οποίες δεν μπορούν να εφαρμοστούν σε προγενέστερες εκδόσεις με αποτέλεσμα οι προγραμματιστές να μην έχουν τη δυνατότητα να παρέχουν υποστήριξη σε συσκευές με παλαιότερα λογισμικά. Αντιθέτως η Apple υποστηρίζει για τουλάχιστον 3 χρόνια τις συσκευές της και παρέχει συχνότερες αναβαθμίσεις (14) (9).



Διάγραμμα 5. Εκδόσεις Android (https://developer.android.com/about/dashboards) (14)

4.4.4 ΑΡΝΗΤΙΚΑ ΤΟΥ iOS

Το iOS από τη πλευρά του, λόγω του κλειστού κώδικα έχει πολλούς περιορισμούς που δεν συναντώνται στο Android. Ο κυριότερος εντοπίζεται στη συνδεσιμότητα μεταξύ συσκευών. Για παράδειγμα, ένας χρήστης Android μπορεί να στείλει ένα αρχείο μέσω του Bluetooth σε έναν άλλο χρήστη, στο iOS αυτή η δυνατότητα είναι κλειδωμένη και υποχρεωτικά η αποστολή θα πρέπει να γίνει μέσω του iCloud. Επιπρόσθετα, το Android υποστηρίζει τη σύνδεση σκληρού δίσκου για τη μεταφορά αρχείων χωρίς να είναι προϋπόθεση η χρήση υπολογιστή. Αντίθετα, στο iOS για τη μεταφορά δεδομένων θα πρέπει να χρησιμοποιηθεί το Apple iTunes που αποτελεί εφαρμογή που έχει αναπτυχθεί από την Apple.

Ταυτόχρονα, οι χρήστες στο iOS δεν έχουν την δυνατότητα να παραμετροποιήσουν το περιβάλλον διασύνδεσης. Ο χρήστης μπορεί μόνο να τροποποιήσει τη σειρά των εφαρμογών και τη ταπετσαρία, ενώ στο Android παρέχεται η δυνατότητα να τροποποιηθεί η συσκευή σε μεγάλο βαθμό (12) (9).

5. Εφαρμογές Έξυπνων τηλεφώνων

Τα “έξυπνες τηλέφωνα” έχουν ενσωματωθεί στην αγορά έχοντας μια σταθερή πορεία προτίμησης από το αγοραστικό κοινό με μεγάλες προοπτικές εξέλιξης. Το κυριότερο χαρακτηριστικό των «έξυπνων τηλεφώνων» είναι οι εφαρμογές που διατίθενται για αυτά όπως και η μεγάλη τους ποικιλία.

Αρκετές εταιρείες έχουν ήδη αναπτύξει τις δικές τους εφαρμογές για φορητές συσκευές, προσφέροντας διασκέδαση και όχι μόνο στους χρήστες, καθώς μια εφαρμογή για κινητό τηλέφωνο μπορεί να εξυπηρετήσει προσωπικές, επαγγελματικές και επιχειρησιακές ανάγκες.

5.1 ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ

Η ανάπτυξη εφαρμογών στο Android λειτουργικό σύστημα έχει αποκτήσει πλέον αρκετή ωριμότητα, κάτι που αποδεικνύεται από τα αποτελέσματα στις ίδιες τις εφαρμογές και την συνεχώς εξελισσόμενη τεχνολογία τους. Σε πρώτη φάση ο προγραμματιστής έπρεπε να χρησιμοποιήσει το ολοκληρωμένο περιβάλλον συγγραφής κώδικα (IDE) Eclipse και κάποιων πρόσθετων add-ons για να προχωρήσει σε ανάπτυξη εφαρμογών. Έπειτα με τη σημαντική συμβολή της Google στη JetBrains, η οποία έχει κυκλοφορήσει το επιτυχημένο IDE IntelliJ IDEA, έφερε στην αγορά το Δεκέμβριο του 2014 το Android Studio το οποίο και αποτελεί σήμερα το βασικό IDE της Google για την ανάπτυξη εφαρμογών που προορίζονται για το περιβάλλον Android. Η κύρια γλώσσα προγραμματισμού για το Android είναι η Java, γεγονός το οποίο βοήθησε το Android να διαδοθεί περισσότερο καθώς αποτελούσε από παλαιότερα μια από τις πρώτες επιλογές γλώσσας αντικειμενοστραφούς προγραμματισμού. Προσπαθώντας η Google να κάνει τη γλώσσα πιο εύχρηστη στο Android Studio, τον Οκτώβριο του 2017 παρουσίασε τη Kotlin ως μια πολύ καλή εναλλακτική γλώσσα και λύση ενώ τον Μάιο του 2019 ανακοίνωσε πως η Kotlin θα αποτελεί πλέον τη βασική γλώσσα προγραμματισμού στο Android. Συνεπώς, ένας προγραμματιστής μπορεί να φτιάξει εφαρμογές Android για τις εκδόσεις από 4.0 ως 10,

απευθείας με τη χρήση του Android Studio ως ένας έτοιμος IDE, με τη χρήση φυσικά των Java & Kotlin και διαθέτοντας τες στο Play Store ή και σε άλλα αντίστοιχα.

5.2 ΤΥΠΟΙ ΕΦΑΡΜΟΓΩΝ ΚΙΝΗΤΩΝ ΣΥΣΚΕΥΩΝ

Εντοπίζονται τρεις τύποι εφαρμογών για φορητές συσκευές, οι εγγενείς εφαρμογές (Native applications), οι εφαρμογές ιστού (Web applications) και οι υβριδικές εφαρμογές (Hybrid applications). Για κάθε είδος υπάρχουν και τα αντίστοιχα θετικά και αρνητικά χαρακτηριστικά τα οποία θα αναφέρουμε παρακάτω.

5.2.1 ΕΓΓΕΝΕΙΣ ΕΦΑΡΜΟΓΕΣ (NATIVE APPLICATIONS)

Οι εγγενείς εφαρμογές θεωρούνται ο πλέον συνήθης τύπος εφαρμογής. Δημιουργούνται για συγκεκριμένες πλατφόρμες και είναι γραμμένες σε γλώσσες που υποστηρίζει η αντίστοιχη πλατφόρμα. Από την Apple και τη Google παρέχονται στους προγραμματιστές τα ανάλογα εργαλεία για την ανάπτυξή τους, τα στοιχεία αλληλεπίδρασης και το ανάλογο SDK. Στην αγορά προτιμάται ιδιαίτερα η ανάπτυξη εγγενών εφαρμογών για φορητά τηλέφωνα και όχι μόνο λόγω των αρκετών θετικών χαρακτηριστικών που προσφέρονται συγκριτικά με άλλα είδη εφαρμογών.

Κάποια από τα θετικά αυτά χαρακτηριστικά είναι η ταχύτητα και η πολύ καλή απόκρισή τους καθώς δημιουργούνται για συγκεκριμένη πλατφόρμα κάθε φορά, η αξιοποίηση αλληλεπιδράσεων προγραμματισμού ανάμεσα σε εφαρμογές (API) και βιβλιοθήκες, οι προγραμματιστές έχουν στη διάθεσή τους όλη τη γκάμα των δυνατοτήτων και λειτουργιών και δεν απαιτείται συνεχής διερεύνηση και βοήθεια από το διαδίκτυο.

Το βασικό μειονέκτημά τους είναι ότι είναι αρκετά δύσκολη η ανάπτυξή τους και δεν μπορούν να εκτελεστούν σε άλλες πλατφόρμες, συνεπώς η προσαρμοστικότητά τους (9).

5.2.2 ΕΦΑΡΜΟΓΕΣ ΙΣΤΟΥ (WEB APPLICATIONS)

Οι εφαρμογές ιστού για “έξυπνες συσκευές” είναι δικτυακές εφαρμογές κατάλληλα σχεδιασμένες για να χρησιμοποιούνται σε φορητές συσκευές και ανοίγουν μέσω του φυλλομετρητή (Browser) της κάθε συσκευής με την επίσκεψή στην ανάλογη ιστοσελίδα της εφαρμογής. Επιπρόσθετα, οι εφαρμογές ιστού δεν χρειάζεται να ληφθούν από ηλεκτρονικά καταστήματα εφαρμογών, όπως οι εγγενείς εφαρμογές. Καμιά εφαρμογή ιστού δεν

αποθηκεύεται στην αντίστοιχη συσκευή και δεν χρειάζεται να δεσμεύσει αποθηκευτικό χώρο. Οι εφαρμογές ιστού παρέχουν τη λύση στο πρόβλημα του κατακερματισμού στο οποίο έχουμε αναφερθεί σε προηγούμενο κεφάλαιο, διότι αναπτύσσονται για να διαθέτουν δυνατότητα προσαρμογής σε όλους τους τύπους οθονών. Αυτό γίνεται χάρις σε συγκεκριμένα πλαίσια (frameworks) τα οποία μπορούν να εκτελούνται σε διαφορετικά λογισμικά, χωρίς να δεσμεύονται από τα τεχνικά χαρακτηριστικά και τις δυνατότητες της κάθε φορητής συσκευής.

Οι εφαρμογές ιστού διαθέτουν αρκετά θετικά χαρακτηριστικά, δύο βασικά είναι η εύκολη κατασκευή τους και το σχετικά μικρό κόστος δημιουργίας και συντήρησης σε σχέση με τις εγγενείς εφαρμογές. Επίσης κάθε εφαρμογή που δημιουργείται μπορεί να εκτελεστεί σε διάφορα λογισμικά, με μόνο προαπαιτούμενο η συσκευή να διαθέτει φυλλομετρητή ιστοσελίδων.

Οι εφαρμογές ιστού ωστόσο έχουν και μια σειρά αρνητικών χαρακτηριστικών. Ο χρήστης πρέπει να πληκτρολογήσει την ιστοσελίδα της εφαρμογής, κάτι το οποίο ισοδυναμεί με μια υποδεέστερη εμπειρία χρήσης. Επιπρόσθετα, οι εφαρμογές ιστού είναι αρκετά πιο «αργές», με λιγότερο διαδραστική εμπειρία και καλαίσθητες πινελιές συγκριτικά με τις εγγενείς εφαρμογές. Κλείνοντας, παρατηρείται έλλειψη πλήρους αξιοποίησης βοηθητικού υλικού (hardware) των συσκευών και χαμηλότερο επίπεδο ασφαλείας (9).

5.2.3 ΥΒΡΙΔΙΚΕΣ ΕΦΑΡΜΟΓΕΣ (HYBRID APPLICATIONS)

Οι υβριδικές εφαρμογές μπορούν να εκτελεστούν σε διαφορετικές πλατφόρμες και η συμπεριφορά τους μοιάζει με τις εγγενείς εφαρμογές. Πρόκειται για έναν συνδυασμό εγγενών εφαρμογών και ιστού. Οι χρήστες μπορούν να προχωρήσουν σε εγκατάσταση στην εκάστοτε συσκευή τους όπως σε μια εγγενή εφαρμογή, αλλά στην πραγματικότητα θα έχουν μια εφαρμογή ιστού.

Αυτές οι εφαρμογές έχουν σα βάση γλώσσες και πρωτόκολλα HTML, CSS και Javascript ενώ εκτελούνται σε κάποιο WebView. Μια υβριδική εφαρμογή αποτελείται από δύο κομμάτια. Το πρώτο μέρος είναι η καθαρή συγγραφή του κώδικα που αναπτύσσεται με τη χρήση γλωσσών όπως οι Javascript, HTML και CSS. Το δεύτερο μέρος είναι ένα κέλυφος το οποίο λαμβάνεται για

να αναλάβει να φορτώσει τον κώδικα που αναφέραμε πριν χρησιμοποιώντας το ανάλογο «Client».

Σαν κύρια θετικά χαρακτηριστικά αυτών των εφαρμογών είναι η γρήγορη ανάπτυξή τους, η οικονομικότερη δημιουργία σε σύγκριση με μια εγγενή εφαρμογή, η προσβασιμότητα στα αντίστοιχα API της συσκευής καθώς και η μη απαίτηση φυλλομετρητή ιστοσελίδων αντίθετα με τις εφαρμογές ιστού στις οποίες είναι προαπαιτούμενο.

Από την άλλη, υπάρχουν και κάποια βασικά μειονεκτήματα που πρέπει να αναφερθούν. Οι υβριδικές εφαρμογές είναι σημαντικά πιο αργές και όχι τόσο διαδραστικές από τις εγγενείς εφαρμογές, διαθέτουν περιορισμένη δυνατότητα παραμετροποίησης συγκριτικά με τις εγγενείς εφαρμογές και είναι πιο απαιτητική η ανάπτυξή τους σε σύγκριση με τις εφαρμογές ιστού (9).

5.2.4 ΕΠΙΛΟΓΗ ΚΑΤΑΛΛΗΛΗΣ ΜΕΘΟΔΟΥ

Η επιλογή της σωστής μεθόδου ανάπτυξης βασίζεται στο εκάστοτε επιχειρησιακό μοντέλο. Λαμβάνοντας υπόψη παράγοντες όπως ο προϋπολογισμός, η πολυπλοκότητα που απαιτείται για την επιθυμητή εφαρμογή, η ποιότητα της εμπειρίας χρήστη (User Experience) καθώς και η ταχύτητα που ιδανικά θα έχει η εφαρμογή που βρίσκεται υπό ανάπτυξη.

Όποια μέθοδος και αν επιλεχθεί, η τελική έκδοση της εφαρμογής θα πρέπει να είναι όσο το δυνατόν πιο γρήγορη και αξιόπιστη. Είναι πλέον δεδομένο στην αγορά ότι οι χρήστες με το πέρασμα του χρόνου γίνονται αρκετά πιο απαιτητικοί από την εκάστοτε εφαρμογή που χρησιμοποιούν αναζητώντας συνεχώς περισσότερες δυνατότητες και προοπτικές. Τέλος, είναι πολύ βασικό να παρέχεται η δυνατότητα υποστήριξης μελλοντικών τροποποιήσεων, εφόσον αυτό κριθεί απαραίτητο, για να βρίσκονται μέσα στον αρκετά μεγάλο πλέον ανταγωνισμό των ημερών μας.

6. Παρουσίαση της Εφαρμογής

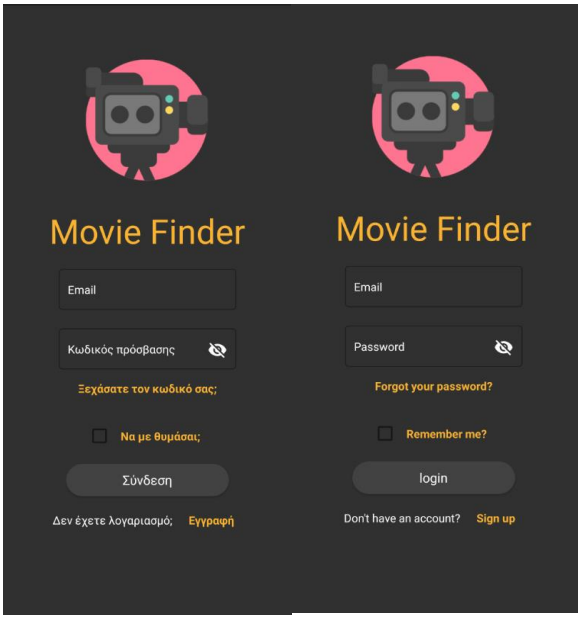
Σε αυτό το κεφάλαιο θα παρουσιάσουμε την εφαρμογή που χρησιμοποιήσαμε για την υλοποίηση σεναρίων αυτοματοποιημένου ελέγχου με το επιλεχθέν εργαλείο. Μέσω στιγμιότυπων (screenshots) θα προσπαθήσουμε να κάνουμε ευκολότερη την κατανόηση της εφαρμογής μας. Τέλος, μπορεί να χρησιμοποιηθεί η εν λόγω παρουσίαση και ως εγχειρίδιο χρήστη (user manual) από τον εκάστοτε ενδιαφερόμενο.

Σκοπός της μεταπτυχιακής εργασίας είναι η χρήση μιας εφαρμογής με την οποία θα μπορούν να παρουσιαστούν εμφανώς οι δυνατότητες του «Android Espresso» στο μεγαλύτερο μέρος τους και θα είναι κατανοητές από τον μέσο χρήστη, είτε πρόκειται για προγραμματιστή (Android Developer) είτε για μηχανικό διασφάλισης ποιότητας λογισμικού(Quality Assurance Engineer). Για τ ον παραπάνω σκοπό επιλέχθηκε μια εφαρμογή μέτριας πολυπλοκότητας που είναι φιλική προς το χρήστη και θα αναδείξει τις δυνατότητες του εργαλείου με κατανοητό και εύκολο τρόπο.

6.1 ΑΡΧΙΚΗ ΔΙΕΠΑΦΗ

Κατά την είσοδο του χρήστη στην εφαρμογή του, μέσω ενός animation εμφανίζεται η σελίδα σύνδεσης. Ο χρήστης εδώ έχει την δυνατότητα να συνδεθεί στην εφαρμογή, να ανακτήσει τον κωδικό του σε περίπτωση που τον έχει ξεχάσει καθώς και να δημιουργήσει χρήστη στην περίπτωση που δεν έχει. Τα πεδία “email” και “password” είναι υποχρεωτικά και σε περίπτωση που δεν συμπληρωθούν εμφανίζεται το σχετικό μήνυμα λάθους. Στην περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό του του δίνεται η δυνατότητα να ανακτήσει την πρόσβασή του μέσα από την λειτουργία “ξεχάσατε τον κωδικό σας” η οποία, κάνοντας χρήση της FireBase, αποστέλλεται ένα e-mail στο χρήστη μέσω ενός συνδέσμου επαναφοράς του κωδικού πρόσβασης. Δεδομένου ότι ο χρήστης έχει συμπληρώσει τα απαραίτητα πεδία και έχει πατήσει το κουμπί “Σύνδεση” η εφαρμογή κάνει ταυτοποίηση αυτών των στοιχείων μέσω της Firebase, στην περίπτωση που τα στοιχεία δεν πληρούν τις ανάλογες προϋποθέσεις,

εμφανίζεται μήνυμα λάθους στο χρήστη, αν τα στοιχεία είναι σωστά ο χρήστης θα μεταβεί στην αρχική οθόνη (Home Page). Στη περίπτωση που ο χρήστης έχει ενεργοποιήσει το checkbox “να με θυμάσαι” πριν πατηθεί το κουμπί για τη σύνδεση, θα θυμάται το email του χρήστη τις επόμενες φορές που θα εκκινήσει η εφαρμογή. Επίσης, για τη περίπτωση που ο χρήστης δεν έχει κάνει εγγραφή επιλέγοντας τον ανάλογο σύνδεσμο “Εγγραφή” μπορεί να δημιουργήσει ένα νέο χρήστη.

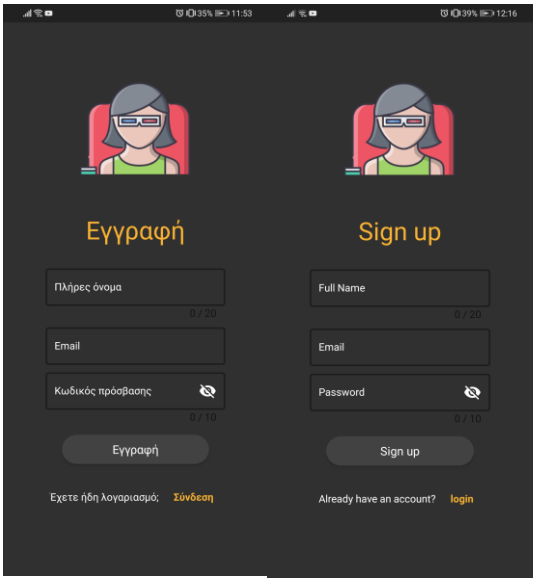


Εικόνα 8. Αρχική Διεπαφή σε Ελληνικά και Αγγλικά

6.2 ΟΘΟΝΗ ΕΓΓΡΑΦΗΣ ΧΡΗΣΤΗ

Η συγκεκριμένη οθόνη αναφέρεται στη δημιουργία ενός νέου χρήστη. Εντοπίζονται σε αυτή τρία υποχρεωτικά πεδία και ένας σύνδεσμος επιστροφής του χρήστη στην προηγούμενη οθόνη Σύνδεσης. Τα τρία συγκεκριμένα αυτά πεδία είναι το όνομα χρήστη, το ηλεκτρονικό ταχυδρομείο και ο προσωπικός κωδικός. Σε αυτά τα πεδία «τρέχουν» κάποιοι πρόσθετοι έλεγχοι, το όνομα θα πρέπει να είναι μέχρι και είκοσι χαρακτήρες, το ηλεκτρονικό ταχυδρομείο πρέπει να έχει μορφοποίηση email και να μην χρησιμοποιείται ήδη, καθώς και ο προσωπικός κωδικός να είναι από έξι έως και δέκα χαρακτήρες. Στη περίπτωση που κάποια από αυτές τις προϋποθέσεις δεν πληρούνται θα εμφανίζεται ένα σχετικό μήνυμα λάθους. Σε περίπτωση που

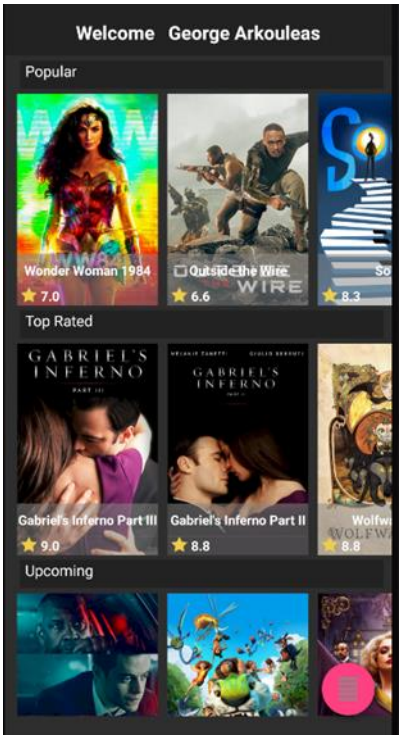
τα στοιχεία είναι εντάξει θα εμφανίζεται ένα μήνυμα επιτυχίας και ο χρήστης θα μεταφέρεται αυτόματα στην οθόνη Σύνδεσης.



Εικόνα 9. Εγγραφή Χρήστη σε Ελληνικά και Αγγλικά

6.3 HOME PAGE

Το Home Page αποτελεί την βασική σελίδα της εφαρμογής μας. Στο χρήστη εμφανίζεται ένα μήνυμα που τον καλωσορίζει, το οποίο εμφαίνεται ως “Καλώς ήρθες” καθώς και το όνομα που έχει ήδη δηλωθεί από το χρήστη. Στη συνέχεια παρουσιάζονται τέσσερις κατηγορίες ταινιών οι δημοφιλείς, οι κορυφαίες, αυτές που θα κυκλοφορήσουν σύντομα και αυτές παίζονται τώρα. Οι κατηγορίες εμφανίζονται με τη χρήση RecyclerView συνεπώς ο χρήστης έχει τη δυνατότητα να κάνει κύλιση οθόνης για να δει τις υπόλοιπες κατηγορίες. Στο κάτω μέρος της οθόνης υπάρχει το κουμπί “επιστροφή στην κορυφή” το οποίο επιτρέπει στον χρήστη να γυρίσει στην κορυφή της οθόνης εύκολα. Στο κάτω δεξιά μέρος της οθόνης υπάρχει ένα floating button το οποίο επιτρέπει στον χρήστη τη πρόσβαση και σε άλλα μέρη της εφαρμογής. Ονομαστικά αναφέρονται οι σελίδες που είναι επισκέψιμες: Αναζήτηση ταινίας, Λεπτομερής αναζήτηση ταινίας, Αναζήτηση ταινίας με τη χρήση της κάμερας, Προφίλ χρήστη και Αποσύνδεση από το λογαριασμό. Επιτρέπεται στο χρήστη να διαλέξει κάποια από τις ταινίες που εμφανίζονται στο Home Page, με αυτόν το τρόπο θα μεταφερθεί αυτόματα στη σελίδα με τις λεπτομέρειες που αφορούν την επιλεχθείσα ταινία.

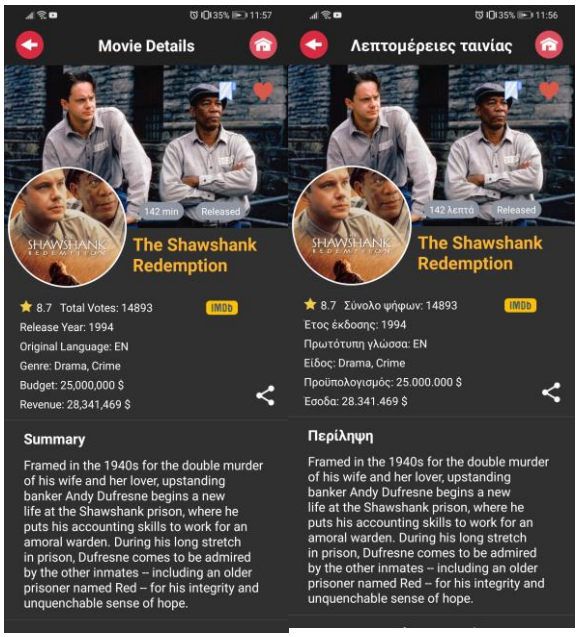


Εικόνα 10. Home Page

6.4 ΣΤΟΙΧΕΙΑ ΤΑΙΝΙΑΣ

Σε αυτή την οθόνη εμφανίζονται τα στοιχεία της επιλεχθείσας ταινίας. Ο χρήστης έχει την δυνατότητα να δει αυτή την οθόνη απευθείας από το Home Page, την Αναζήτηση ταινίας, τη Λεπτομερή αναζήτηση ταινίας, την Αναζήτηση ταινίας με τη χρήση της κάμερας περνώντας μέσα από τις λεπτομέρειες ηθοποιού, από την λίστα των αγαπημένων καθώς και από τη λίστα παρακολούθησης. Αρχίζοντας από τον header, ο χρήστης βλέπει σαν αρχικό τίτλο την ονομασία της οθόνης (Movie Details) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη σελίδα ή και απευθείας στο Home Page. Στην συνέχεια εμφανίζεται στο χρήστη το αντίστοιχο «poster» της ταινίας (αν υπάρχει η διαθεσιμότητα) στο οποίο φαίνεται η κατάσταση (Status) της ταινίας (π.χ. αν έχει ήδη κυκλοφορήσει) και η διάρκειά της σε λεπτά. Στο σημείο αυτό παρέχεται στο χρήστη η δυνατότητα να προσθέσει ή να αφαιρέσει κάποια ταινία από τη λίστα αγαπημένων ή από τη λίστα παρακολούθησης που έχει ο ίδιος διαμορφώσει. Ακριβώς από κάτω παρουσιάζεται η κύρια εικόνα της ταινίας (αν υπάρχει η ανάλογη διαθεσιμότητα) και ακριβώς δεξιά της ο αντίστοιχος τίτλος. Από κάτω εμφαίνεται μια λίστα με πληροφορίες που αφορούν τη ταινία, συγκεκριμένα η βαθμολόγηση που έχει γίνει στη ταινία και οι συνολικοί ψήφοι που έχει λάβει

από το κοινό, το έτος αρχικής κυκλοφορίας της, η γλώσσα στην οποία κυκλοφόρησε, η κατηγορία της, ο συνολικός προϋπολογισμός που διατέθηκε, τα κέρδη που επέφερε, κάποιοι σύνδεσμοι για το κοινωνικό μέσο «Facebook» και για την πολύ γνωστή σελίδα-βάση «IMDB» καθώς και η επιλογή για τη κοινοποίησή της σε διάφορα άλλα μέσα που ο χρήστης επιθυμεί, όποια πληροφορία δεν είναι στην ανάλογη διαθεσιμότητα, δεν εμφανίζεται στη συγκεκριμένη οθόνη. Στη συνέχεια, ο χρήστης μπορεί να διαβάσει μια περίληψη που αφορά την υπόθεση της ταινίας, μια λίστα με τους ηθοποιούς που πρωταγωνιστούν και πάλι σε μορφή «RecyclerView» όπου με την επιλογή του αντίστοιχου ηθοποιού, ο χρήστης θα μεταφερθεί σε κάποια άλλη οθόνη με τις ανάλογες πληροφορίες που αφορούν τον επιλεχθέντα ηθοποιό καθώς και μια ακόμα λίστα με όλες τις συ σχετιζόμενες ταινίες που ο ηθοποιός έχει συμμετάσχει, σε μορφή «RecyclerView» και πάλι, όπου επιλέγοντας κάποια από αυτές τις ταινίες ο χρήστης θα μεταφερθεί στην αντίστοιχη σελίδα της ταινίας που επιλέχθηκε μέσω της σελίδας του ηθοποιού.

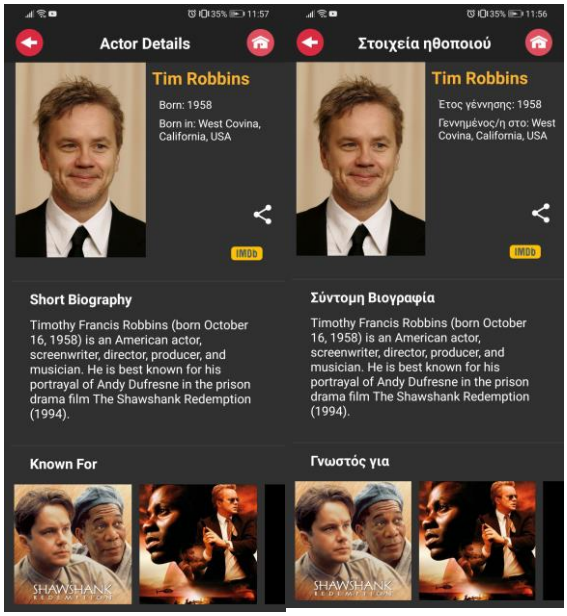


Εικόνα 11. Στοιχεία Ταινίας

6.5 ΣΤΟΙΧΕΙΑ ΗΘΟΠΟΙΟΥ

Η οθόνη που αφορά τα στοιχεία του ηθοποιού ακολουθεί μια παραπλήσια λογική με την οθόνη που αφορά τα στοιχεία ταινίας, αυτό κυρίως έγινε για λόγους ομοιομορφίας αλλά και για την ομαλότερη εμπειρία που απολαμβάνει ο χρήστης. Αρχίζοντας από τον «header», ο χρήστης

ξεκινάει με την προβολή του τίτλου, το όνομα της συγκεκριμένης οθόνης, το οποίο είναι φυσικά το «Actor Details» και του δίνεται η δυνατότητα να επιστρέψει στην αμέσως προηγούμενη οθόνη ή ακόμα και στο «Home Page». Έπειτα εμφαίνεται μια κύρια φωτογραφία του ηθοποιού (αν υπάρχει η ανάλογη διαθεσιμότητα) και ακριβώς δεξιά το όνομα του αντίστοιχου ηθοποιού, το έτος γέννησης, ο τόπος γέννησης, η ημερομηνία θανάτου (εάν δε βρίσκεται στη ζωή), ο ανάλογος σύνδεσμος που μεταφέρει το χρήστη στην αντίστοιχη σελίδα του «IMDB» και του κοινωνικού μέσου «Facebook», καθώς και η επιλογή κοινοποίησης. Ακριβώς κάτω παρουσιάζεται ένα σύντομο βιογραφικό σημείωμα καθώς και κάποιες άλλες ταινίες στις οποίες έχει συμμετάσχει και έχει γίνει γνωστός, σε μορφή «RecyclerView», συνεπώς διαλέγοντας ο χρήστης κάποια ταινία, θα μεταφερθεί στην ανάλογη σελίδα της όπως έχει προαναφερθεί και σε προηγούμενες οθόνες.

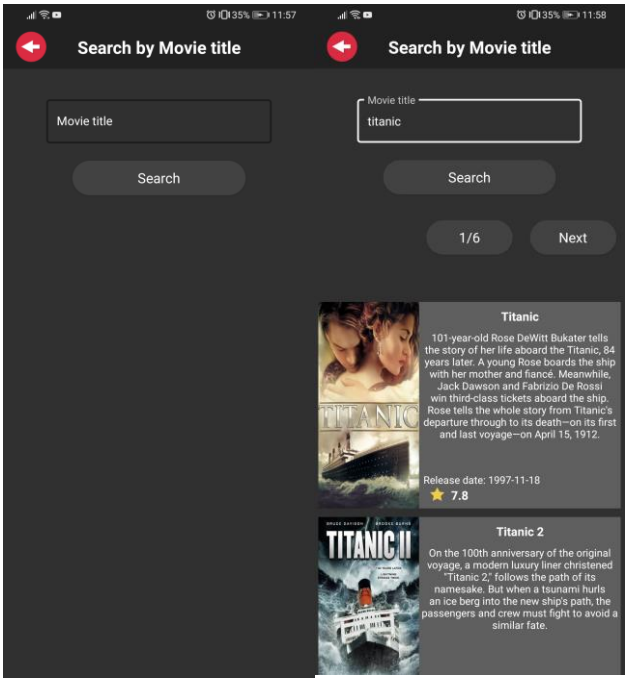


Εικόνα 12. Στοιχεία Ηθοποιού

6.6 ΑΝΑΖΗΤΗΣΗ ΤΑΙΝΙΑΣ

Η οθόνη που σχετίζεται με την αναζήτηση ταινίας δίνει στο χρήστη τη δυνατότητα να κάνει αναζήτηση με βάση τον τίτλο της επιθυμητής ταινίας. Σε πρώτη φάση έρχεται ο «header», στον οποίο αναφέρεται το όνομα της συγκεκριμένης οθόνης στο χρήστη, δηλαδή το «Search by Movie Title» και παρέχεται η δυνατότητα επιστροφής στη προηγούμενη οθόνη, ακριβώς κάτω

βρίσκεται ένα πεδίο όπου μπορεί να συμπληρωθεί η ταινία που επιθυμεί να αναζητήσει καθώς και το ίδιο το κουμπί της αναζήτησης. Εφόσον συμπληρωθεί το πεδίο και πατηθεί το κουμπί που αναφέρθηκε, εκτελείται αναζήτηση στη βάση του «IMDB», στην περίπτωση μη εμφάνισης αποτελεσμάτων μετά το πέρας της διαδικασίας, ο χρήστης θα λάβει σχετικό μήνυμα, ενώ σε αντίθετη περίπτωση εύρεσης αποτελέσματος, εμφανίζεται μήνυμα με το σύνολο των αποτελεσμάτων σε αριθμό. Τα αποτελέσματα αυτά εμφανίζονται σε πλήθος των είκοσι, συνεπώς για τη περίπτωση που είναι περισσότερα, υποστηρίζεται «pagination», έτσι ώστε ο χρήστης να μπορεί να πλοηγηθεί με ευκολία σε αυτά. Τα αποτελέσματα αυτά περιλαμβάνουν κάποιες βασικές πληροφορίες για κάθε ταινία όπως η κύρια φωτογραφία της, ο τίτλος της, μια μικρή περιγραφή, το πρώτο έτος κυκλοφορίας της και τη βαθμολογία της με βάση τη γνώμη του κοινού. Εάν ο χρήστης επιλέξει κάποια από αυτές τις ταινίες, θα μεταφερθεί στη κεντρική σελίδα της.

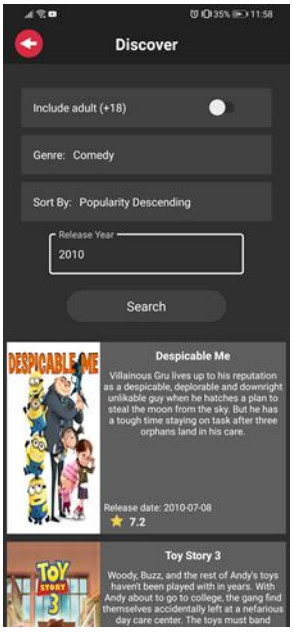


Εικόνα 13. Αναζήτηση ταινίας

6.7 ΑΝΑΚΑΛΥΨΗ ΤΑΙΝΙΑΣ

Η οθόνη που αφορά την ανακάλυψη ταινίας παρέχει στο χρήστη τη δυνατότητα να ανακαλύψει διάφορες ταινίες οι οποίες πιθανώς θα συνάδουν με τις προτιμήσεις του και θα τον

ενδιαφέρουν. Πρώτη επαφή έχει ο χρήστης με τον «header», όπου φαίνεται ως τίτλος το κύριο όνομα της συγκεκριμένης οθόνης (Discover) και παρέχεται όπως και στις προαναφερθείσες η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω είναι διαθέσιμα φίλτρα αναζήτησης προς διευκόλυνση της εμπειρίας του χρήστη, όπως για παράδειγμα το αν η ταινία είναι για ενήλικες (η επιλογή γίνεται με διακόπτη), το είδος που ενδεχόμενα να ενδιαφέρεται ο χρήστης (η επιλογή γίνεται μέσα από λίστα), το πως επιθυμεί να γίνεται η ταξινόμηση (η επιλογή γίνεται μέσα από λίστα) και το έτος αρχικής κυκλοφορίας της επιθυμητής ταινίας (γίνεται μέσω πεδίου). Όλα τα πεδία που εμφανίζονται προς συμπλήρωση είναι υποχρεωτικό να γεμίσουν εκτός από το αν ο τίτλος ταινίας που αναζητείται θα είναι κατάλληλος μόνο για ενήλικες. Επιλέγοντας ο χρήστης το κουμπί της αναζήτησης που αναφέρθηκε και νωρίτερα, με τη χρήση των επιλεχθέντων φίλτρων θα γίνει αναζήτηση στη βάση του «IMDB» και θα εμφανιστούν τα αντίστοιχα αποτελέσματα με μέγιστο αριθμό το είκοσι. Εάν ο χρήστης επιλέξει κάποια από αυτές τις ταινίες, θα μεταφερθεί στη κεντρική σελίδα της.

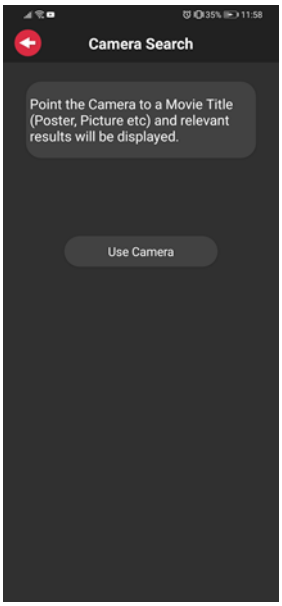


Εικόνα 14. Ανακάλυψη ταινίας

6.8 ΑΝΑΖΗΤΗΣΗ ΤΑΙΝΙΑΣ ΜΕΣΩ ΚΑΜΕΡΑΣ

Η οθόνη που αφορά την αναζήτηση ταινίας μέσω κάμερας παρέχει στο χρήστη τη δυνατότητα να αναζητήσει μια ταινία με βάση τον τίτλο χρησιμοποιώντας την ίδια την κάμερα

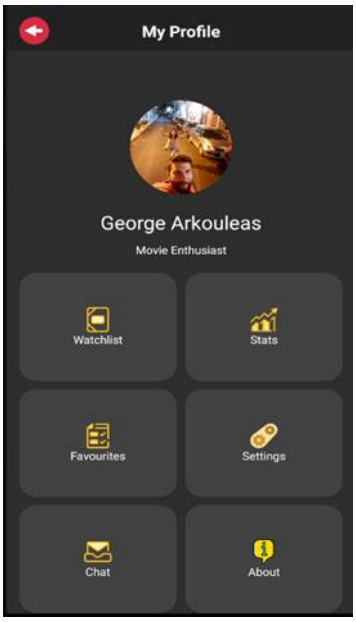
που διαθέτει η συσκευή του. Σε πρώτη φάση στο χρήστη παρουσιάζεται ο «header», εκεί εμφανίζεται σε μορφή τίτλου το κύριο όνομα της οθόνης (Camera Search) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω εντοπίζονται πληροφορίες αναφορικά με τη χρήση αυτής της λειτουργικότητας καθώς και το κουμπί άμεσης χρήσης της κάμερας. Για τη πρώτη φορά χρήσης αυτής της λειτουργίας θα ζητηθεί από το χρήστη να δοθούν τα απαραίτητα δικαιώματα για το άνοιγμα και τη χρήση της κάμερας από την εφαρμογή. Ανοίγοντας την κάμερα ο χρήστης έχει τη δυνατότητα να εστιάζει είτε σε κάποια φωτογραφία είτε σε όποιο άλλο αντικείμενο εμφανίζεται και με μορφή κειμένου. Αφού εγκριθεί η φωτογραφία από το κείμενο που έχει αναγνωριστεί, ο χρήστης θα χρειαστεί να συμπληρώσει ένα πεδίο μέσα από το οποίο μπορούν να γίνουν βελτιώσεις. Με το πάτημα του κουμπιού της αναζήτησης, εκτελείται αναζήτηση στη βάση του «IMDB», στην περίπτωση μη εμφάνισης αποτελεσμάτων μετά το πέρας της διαδικασίας, ο χρήστης θα λάβει σχετικό μήνυμα, ενώ σε αντίθετη περίπτωση εύρεσης αποτελέσματος, εμφανίζεται μήνυμα με το σύνολο των αποτελεσμάτων σε αριθμό. Τα αποτελέσματα αυτά εμφανίζονται σε πλήθος των είκοσι, συνεπώς για τη περίπτωση που είναι περισσότερα, υποστηρίζεται «pagination», έτσι ώστε ο χρήστης να μπορεί να πλοηγηθεί με ευκολία σε αυτά. Τα αποτελέσματα αυτά περιλαμβάνουν κάποιες βασικές πληροφορίες για κάθε ταινία όπως η κύρια φωτογραφία της, ο τίτλος της, μια μικρή περιγραφή, το πρώτο έτος κυκλοφορίας της και τη βαθμολογία της με βάση τη γνώμη του κοινού. Εάν ο χρήστης επιλέξει κάποια από αυτές τις ταινίες, θα μεταφερθεί στη κεντρική σελίδα της.



Εικόνα 15. Αναζήτηση ταινίας μέσω κάμερας

6.9 ΤΟ ΠΡΟΦΙΛ ΜΟΥ

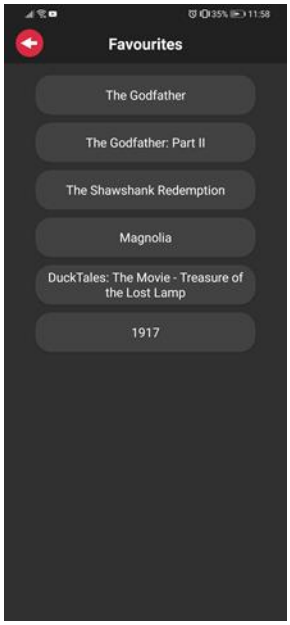
Η οθόνη που αφορά το προφίλ του χρήστη παρέχει τη δυνατότητα να δει ο χρήστης αποθηκευμένες ταινίες που έχει κρατήσει στο λογαριασμό του ή στη λίστα του, να επεξεργαστεί τα προσωπικά του στοιχεία καθώς και να επικοινωνήσει με άλλους χρήστες που χρησιμοποιούν την εφαρμογή. Σε πρώτη φάση στο χρήστη παρουσιάζεται ο «header», εκεί εμφανίζεται σε μορφή τίτλου το κύριο όνομα της οθόνης (My Profile) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω εντοπίζονται η φωτογραφία προφίλ του, το όνομα και η λεζάντα που έχει διαλέξει. Ακριβώς κάτω από αυτά ο χρήστης έχει τις ακόλουθες επιλογές: τη λίστα αγαπημένων ταινιών, τη λίστα παρακολούθησης, το παράθυρο διαλόγων/συνομιλιών, τα στατιστικά του, τις σχετικές ρυθμίσεις καθώς και τα “σχετικά με την εφαρμογή”. Στο σημείο αυτό έχει τη δυνατότητα ο χρήστης να ενημερωθεί για την τρέχουσα έκδοση της εφαρμογής αλλά και για τον δημιουργό της εφαρμογής.



Εικόνα 16. Το προφίλ μου

6.10 ΛΙΣΤΑ ΑΓΑΠΗΜΕΝΩΝ

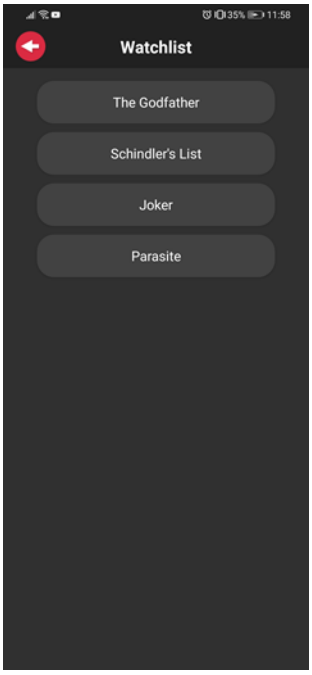
Η οθόνη που αφορά τη λίστα αγαπημένων παρέχει στο χρήστη τη δυνατότητα να αναζητήσει απευθείας στη λίστα των ταινιών που έχει αποθηκεύσει στις αγαπημένες του μέσα από την οθόνη των λεπτομερειών ταινίας. Πρώτη επαφή έχει ο χρήστης με τον «header», όπου φαίνεται ως τίτλος το κύριο όνομα της συγκεκριμένης οθόνης (Favourites) και παρέχεται όπως και στις προαναφερθείσες η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω είναι η λίστα που με τη μορφή «RecyclerView» προβάλλονται τα ονόματα των συμπεριλαμβανομένων ταινιών του χρήστη. Όλα τα στοιχεία αποθηκεύονται ή και ανακτώνται μέσω της «FireBase» στην οποία έχει αποθηκευτεί αποκλειστικά ο τίτλος της επιλεγμένης ταινίας και το «ID» της τα οποία είναι απαραίτητα για τη μεταφορά στη σελίδα με τα στοιχεία ταινίας εφόσον το επιθυμεί ο χρήστης.



Εικόνα 17. Λίστα Αγαπημένων

6.11 ΛΙΣΤΑ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ

Η οθόνη που αφορά τη λίστα παρακολούθησης παρέχει στο χρήστη τη δυνατότητα να κάνει αναζήτηση στις ταινίες που έχει επιλέξει να κρατήσει για να τις παρακολουθήσει μελλοντικά, μέσω της οθόνης στοιχεία ταινίας. Σε πρώτη φάση στο χρήστη παρουσιάζεται ο «header», εκεί εμφανίζεται σε μορφή τίτλου το κύριο όνομα της οθόνης (Watchlist) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω είναι η λίστα που με τη μορφή «RecyclerView» προβάλλονται τα ονόματα των συμπεριλαμβανομένων ταινιών του χρήστη. Όλα τα στοιχεία αποθηκεύονται ή και ανακτώνται μέσω της «FireBase» στην οποία έχει αποθηκευτεί αποκλειστικά ο τίτλος της επιλεγμένης ταινίας και το «ID» της τα οποία είναι απαραίτητα για τη μεταφορά στη σελίδα με τα στοιχεία ταινίας εφόσον το επιθυμεί ο χρήστης.



Εικόνα 18. Λίστα Παρακολούθησης

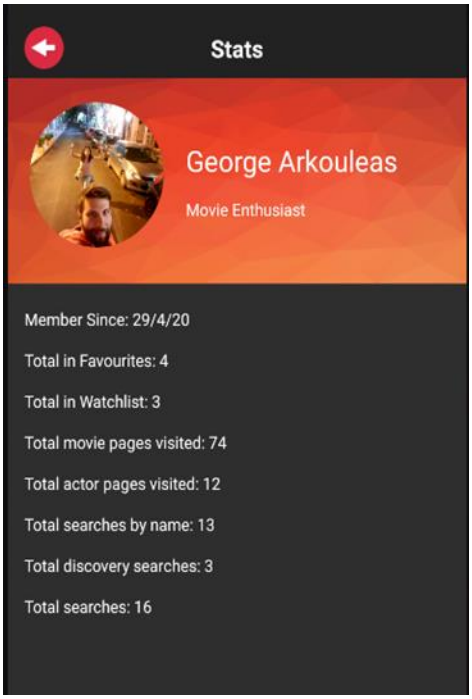
6.12 CHAT

Η οθόνη που αφορά την δυνατότητα των χρηστών να συνομιλούν για οτιδήποτε τους ενδιαφέρει, είτε αυτό αφορά κάποια ταινία είτε όχι. Το «Chat» της εφαρμογής διατίθεται ανοιχτό σε όλους τους χρήστες και δεν απαιτείται καμία επιπλέον ρύθμιση. Πρώτη επαφή έχει ο χρήστης με τον «header», όπου φαίνεται ως τίτλος το κύριο όνομα της συγκεκριμένης οθόνης (Chat) και παρέχεται όπως και στις προαναφερθείσες η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω είναι η λίστα που με τη μορφή «RecyclerView» προβάλλονται όλα τα μηνύματα που έχουν ανταλλάξει οι χρήστες. Με την είσοδο του ο χρήστης στη συγκεκριμένη οθόνη θα μεταφερθεί μεταφέρεται χωρίς να χρειαστεί να κάνει κάποια ενέργεια στο πλέον πρόσφατο μήνυμα.

6.13 ΣΤΑΤΙΣΤΙΚΑ

Η οθόνη που αφορά την δυνατότητα των χρηστών να μπορούν να δουν τη συνολική τους δραστηριότητα στην εφαρμογή σε επίπεδο αριθμών και στατιστικών. Σε πρώτη φάση στο χρήστη παρουσιάζεται ο «header», εκεί εμφανίζεται σε μορφή τίτλου το κύριο όνομα της οθόνης (Stats) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς στο

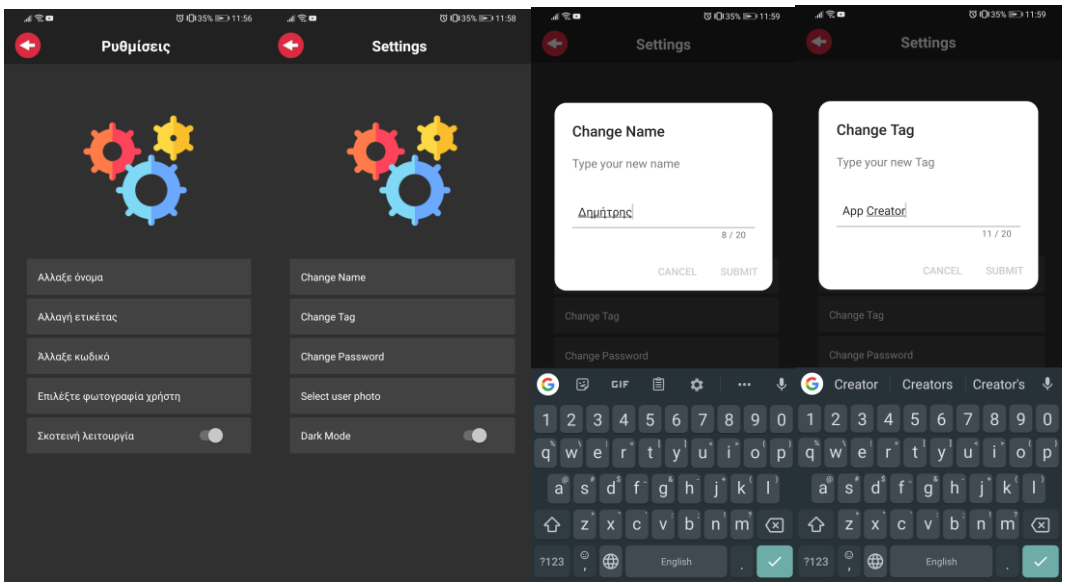
κάτω μέρος εμφανίζονται τα εξής δεδομένα: “Μέλος από”, ημερομηνία της πρώτης εγγραφής του χρήστη στην εφαρμογή, το “Σύνολο στα Αγαπημένα” δηλαδή το πλήθος των συνολικών ταινιών που έχουν αποθηκευτεί ως αγαπημένες, “Συνολικά στην λίστα παρακολούθησης” δηλαδή οι συνολικές ταινίες που έχουν επιλεχθεί για μελλοντική παρακολούθηση, οι “Συνολικές σελίδες ταινιών που προβλήθηκαν” δηλαδή το πλήθος των επισκέψεων στην οθόνη Στοιχεία Ταινίας, οι “Συνολικές σελίδες ηθοποιού που προβλήθηκαν” δηλαδή το πλήθος των επισκέψεων στη σελίδα Στοιχεία Ηθοποιού, οι “Συνολικές αναζητήσεις με βάση το όνομα” δηλαδή το πλήθος των αναζητήσεων που πραγματοποιήθηκαν από την οθόνη Αναζήτηση ταινίας, οι “Συνολικές αναζητήσεις ανακάλυψης” δηλαδή το πλήθος των αναζητήσεων που πραγματοποιήθηκαν από την οθόνη Ανακάλυψη ταινίας, οι “Συνολικές αναζητήσεις με κάμερα” δηλαδή το πλήθος των αναζητήσεων που πραγματοποιήθηκαν από την οθόνη Αναζήτηση ταινίας μέσω κάμερας, οι “Συνολικές αναζητήσεις” δηλαδή το πλήθος όλων των αναζητήσεων που πραγματοποιήθηκαν από όλες τις οθόνες.



Εικόνα 19. Στατιστικά

6.14 ΡΥΘΜΙΣΕΙΣ

Η οθόνη που αφορά την δυνατότητα των χρηστών να μπορούν να διαχειριστούν το λογαριασμό τους. Πρώτη επαφή έχει ο χρήστης με τον «header», όπου φαίνεται ως τίτλος το κύριο όνομα της συγκεκριμένης οθόνης (Settings) και παρέχεται όπως και στις προαναφερθείσες η δυνατότητα επιστροφής σε προηγούμενη οθόνη, ακριβώς κάτω μπορεί να αλλαχθεί το όνομα του χρήστη εφόσον το επιθυμεί, καθώς και η λεζάντα του, να περάσει στην οθόνη «Αλλαγή Κωδικού», να περάσει στην οθόνη «Αλλαγή Φωτογραφίας» ή και να αλλάξει το κύριο θέμα της εφαρμογής (για παράδειγμα εναλλαγή από light mode σε dark mode και αντιστρόφως) με τη χρήση ενός απλού «διακόπτη» (switcher). Η αλλαγή του ονόματος και τις λεζάντας πραγματοποιείται μέσω της «FireBase».

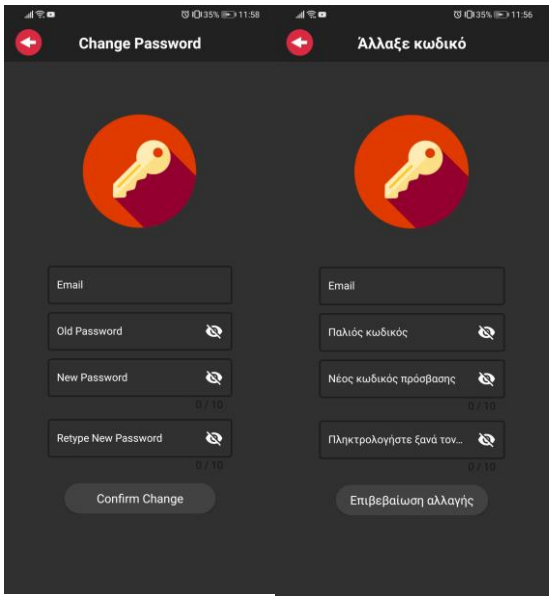


Εικόνα 20. Ρυθμίσεις

6.15 ΑΛΛΑΓΗ ΚΩΔΙΚΟΥ

Μέσω αυτής της οθόνης οι χρήστες έχουν τη δυνατότητα να αλλάξουν τον κωδικό του λογαριασμού τους. Σε πρώτη φάση στο χρήστη παρουσιάζεται ο «header», εκεί εμφανίζεται σε μορφή τίτλου το κύριο όνομα της οθόνης (Change Password) και παρέχεται η δυνατότητα επιστροφής σε προηγούμενη οθόνη, στο κάτω μέρος βρίσκονται τέσσερα υποχρεωτικά πεδία: το «Email», ο «Παλαιός Κωδικός», ο «Νέος Κωδικός» και το «Πληκτρολόγησε ξανά τον νέο κωδικό» καθώς και το κουμπί που αναλαμβάνει να επιβεβαιώσει τα στοιχεία που δόθηκαν από

το χρήστη. Τα πεδία που περιέχουν τους κωδικούς κρύβουν τα στοιχεία αυτόματα ενώ ταυτόχρονα δίνουν την δυνατότητα εμφάνισής τους πατώντας το ανάλογο κουμπί δίπλα στο πεδίο. Κάνοντας χρήση του κουμπιού επιβεβαίωσης εκτελούνται έλεγχοι, τοπικά αλλά και στη βάση της «FireBase». Οι έλεγχοι αυτοί αφορούν στη συμπλήρωση των υποχρεωτικών προαναφερθέντων πεδίων, στην ορθή συμπλήρωση του «Email» και του «Παλαιού Κωδικού", καθώς στην ομοιότητα του «Παλαιού Κωδικού» με το «Νέο Κωδικό". Επιπρόσθετα ελέγχεται και το πλήθος των χαρακτήρων του «Νέου Κωδικού» (τουλάχιστον έξι χαρακτήρες) καθώς και η ορθή επανάληψή του στο πεδίο «Πληκτρολόγησε ξανά το νέο κωδικό».



Εικόνα 21. Αλλαγή Κωδικού

7. Παρουσίαση του Android Espresso Framework

Στο παρακάτω κεφάλαιο θα γίνει μια παρουσίαση του εργαλείου που χρησιμοποιήθηκε για την υλοποίηση του ελέγχου της εφαρμογής καθώς και της παρούσας διπλωματικής εργασίας. Αυτή θα περιλαμβάνει κάποιες γενικές πληροφορίες για το συγκεκριμένο εργαλείο αλλά και τις δυνατότητες που παρέχει στο χρήστη που θα εκτελέσει αυτοματοποιημένα σενάρια ελέγχου.

7.1 ΓΝΩΡΙΜΙΑ ΜΕ ΤΟ ANDROID ESPRESSO

Σαν μια μικρή αναφορά στην ιστορία του εργαλείου, έκανε την εμφάνισή του τον Οκτώβριο του 2013 με το release 1.0 της Google. Από το release 2.0, το «Android Espresso» αποτελεί αναπόσπαστο μέρος του «Android Support Repository». Συνεπώς πρόκειται για ένα αρκετά σύγχρονο εργαλείο. Αξίζει να αναφερθεί πως η δομή του είναι πολύ φιλική προς το χρήστη, δεν απαιτεί υψηλές γνώσεις προγραμματισμού και μπορεί να γίνει πολύ γρήγορα αποτελεσματικό και εύχρηστο (14) (15).

7.2 ANDROID ESPRESSO CHEAT SHEET

Αναφερθήκαμε ήδη στην ευκολία χρήσης πολύ προσφέρει το επιλεχθέν εργαλείο, όμως πραγματικά σε αυτό το σημείο βρίσκεται αυτό που θα κερδίσει από την πρώτη στιγμή τον προγραμματιστή ή «tester» που θα το χρησιμοποιήσει. Πρόκειται μια μόνο σελίδα η οποία περιέχει κάποιους μικρούς πίνακες με τους οποίους κάνοντας συνδυασμούς των περιεχομένων τους μπορεί κάποιος να εκτελέσει γρήγορα και απλά, σενάρια επιπέδου «UI(User Interface)» χωρίς να απαιτείται κάποια προηγούμενη προετοιμασία, απλά γράφοντάς τα στο περιβάλλον της εφαρμογής στην οποία έχει στηθεί το εργαλείο, στο περιβάλλον του Android Studio. Το φύλλο που ακολουθεί δεν περιλαμβάνει όλες τις δυνατότητες του εργαλείου, περιέχει όμως το μεγαλύτερο μέρος τους, κάτι που το κάνει το απόλυτο εγχειρίδιο για νεότερους χρήστες αλλά και παλιότερους που πιθανώς να χρειαστεί να ανατρέξουν σε κάτι μιας και έχει πολλή πληροφορία συγκεντρωμένη.

Παρακάτω περιγράφονται συνοπτικά τα «Instances» του εργαλείου που περιέχει το «Cheat Sheet»:

- View Matchers: Ένα σύνολο αντικειμένων το οποίο έχει ως στόχο τη σύγκριση της όψης (View) της τρέχουσας οθόνης της εφαρμογής
- View Actions: Ένα σύνολο αντικειμένων το οποίο έχει ως στόχο τις ενεργές κινήσεις μέσα στο περιβάλλον της εφαρμογής, όπως είναι για παράδειγμα το πάτημα κουμπιών
- View Assertions: Ένα σύνολο αντικειμένων το οποίο έχει ως στόχο τον έλεγχο και την επιβεβαίωση ότι οι συγκρίσεις που έγιναν με τη χρήση των «matchers» στις όψεις (views) της εφαρμογής, ταυτίζονται με τις προσδοκώμενες (15) (6) (14).

Εικόνα 22. Android Espresso Cheat Sheet (15) (6) (14)

7.3 VIEW MATCHERS

Παράδειγμα χρήσης με στιγμιότυπο από την εφαρμογή μας. Στο παρακάτω σύνολο σεναρίων παρουσιάζεται μέρος αυτοματοποιημένου ελέγχου της οθόνης εισόδου χρήστη (Login Screen) σε επίπεδο UI(User Interface).

```
@Test
fun testTitleTextDisplayed() {
    onView(withId(R.id.appTittleText)).check(matches(withText(R.string.app_name)))
}

@Test
fun testLoginLogoDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.loginLogo)).check(matches(isDisplayed()))
}

@Test
fun emailTextFieldDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditTextLabel)).check(matches(isDisplayed()))
}

@Test
fun forgotYourPasswordLinkDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.forgotYourPasswordLink)).check(matches(withText(R.string.forgotYourPassword)))
}

@Test
fun checkboxIsCheckedOrNot() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)
```

```
    onView(withId(R.id.rememberMeCheckbox)).check(matches(isNotChecked()))
}
```

Εικόνα 23. Παράδειγμα χρήσης View Matchers

7.4 VIEW ACTIONS

Παράδειγμα χρήσης με στιγμιότυπο από την εφαρμογή μας. Στο παρακάτω σύνολο σεναρίων παρουσιάζεται μέρος αυτοματοποιημένου ελέγχου των στοιχείων συμπλήρωσης των πεδίων της οθόνης εισόδου χρήστη (Login Screen), είτε σε επίπεδο ορθότητας είτε σε επίπεδο πληρότητας (Credentials' checking).

```
@Test
fun loginButton() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.loginButton)).check(matches(isClickable()))
}

//Testing Login Fields

@Test
fun emailIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.emailInputEditText)).check(matches(withText("")))
}

@Test
fun passwordIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)
```

```
    onView(withId(R.id.emailInputEditText)).perform(typeText("email@email.com"), closeSoftKeyboard())
    onView(withId(R.id.passwordInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.passwordInputEditText)).check(matches(withText("")))
}

@Test
fun emailIsInvalid() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(typeText("invalid"), closeSoftKeyboard())
    onView(withId(R.id.loginButton)).perform(click())
    //onView(withText("Please fill all the fields")).check(matches(isDisplayed()))
}
```

Εικόνα 24. Παράδειγμα χρήσης View Actions

7.5 VIEW ASSERTIONS

Παράδειγμα χρήσης με στιγμιότυπο από την εφαρμογή μας. Στο παρακάτω σύνολο σεναρίων παρουσιάζεται μέρος αυτοματοποιημένου ελέγχου των στοιχείων συμπλήρωσης των πεδίων της οθόνης εισόδου χρήστη (Login Screen), στα οποία διακρίνεται η χρήση των «Assertions» σε συνδυασμό με τα «Actions» όπως είναι το πάτημα διαφόρων κουμπιών.

```
@Test
fun loginButton() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.loginButton)).check(matches(isClickable()))
}

//Testing Login Fields

@Test
```

```
fun emailIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.emailInputEditText)).check(matches(withText("")))
}

@Test
fun passwordIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(typeText("email@email.com"), closeSoftKeyboard())
    onView(withId(R.id.passwordInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.passwordInputEditText)).check(matches(withText("")))
}
```

Εικόνα 25. Παράδειγμα χρήσης View Assertions

7.6 ESPRESSO LISTS

Το «Android Espresso» προσφέρει μηχανισμούς για κύλιση ή ενεργοποίηση ενός συγκεκριμένου αντικειμένου για δύο τύπους λιστών: «Adapter View» και «Recycler View». Στην εφαρμογή που χρησιμοποιήσαμε για την παρουσίαση του εργαλείου έχουμε κάνει χρήση μόνο του «Recycler View», συνεπώς η παρακάτω εικόνα που σχετίζεται με το «Adapter View» δεν εμπεριέχεται στον κώδικά μας. Η συγκεκριμένη δυνατότητα προσφέρει την αυτοματοποιημένη εκτέλεση δοκιμών σε σύνολα πληροφοριών όπως διαφόρων ειδών λίστες ή ακόμα και σύνθετες βάσεις δεδομένων τις οποίες ανατρέχει το ίδιο το εργαλείο για να εκτελέσει το δοθέν σενάριο. Ενδεικτικά αναφέρονται και κάποιες υποπροβολές (Sub Views) που προέρχονται από το «Adapter View»: «List View, Grid View, Spinner».

Εικόνα 26. Παράδειγμα χρήσης Adapter View (16)

```
@Test
fun backToTopButton() {
   onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
   onView(ViewMatchers.withId(R.id.nestedScrollView)).perform(swipeUp())
   onView(isRoot()).perform(waitFor(2000))
onView(ViewMatchers.withId(R.id.bottomPageButton)).check(ViewAssertions.matches(ViewMatchers.isDisplayed
()))
}
@Test
fun recyclerView() {
   onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
   onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.recyclerView_new_releases)).perform(RecyclerViewActions.actionOnItemAtPo
sition<RecyclerView.ViewHolder>(1, click()))
   onView(isRoot()).perform(waitFor(2000))
   onView(ViewMatchers.withId(R.id.detailsMovieBackButton)).perform(click())
   onView(ViewMatchers.withId(R.id.nestedScrollView)).perform(swipeUp())
   onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.recyclerView_now_playing)).perform(RecyclerViewActions.actionOnItemAtPosi
tion<RecyclerView.ViewHolder>(1, click()))
   onView(isRoot()).perform(waitFor(2000))
onView(ViewMatchers.withId(R.id.recyclerView_actors)).perform(RecyclerViewActions.actionOnItemAtPosition<R
ecyclerView.ViewHolder>(0, click()))
   onView(isRoot()).perform(waitFor(2000))
}
```

```
@Test
fun bmpButton() {
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 27. Παράδειγμα χρήσης Recycler View

8. Test Plan

Σε αυτό το κεφάλαιο θα παρατεθεί ένα πλάνου ελέγχου της εφαρμογής (Test Plan) πάνω στο οποίο στηρίχθηκε το εργαλείο μας. Σε κάθε περίπτωση που κάποια εφαρμογή υπόκειται σε δοκιμές, είτε αυτές γίνονται χειροκίνητα (Manual Testing) είτε μέσω αυτοματοποιημένων εργαλείων (Automated Testing), θα πρέπει να γίνεται ένας σχεδιασμός πάνω στον οποίο θα βασιστούν τα σενάρια που θα εκτελεστούν.

Επίσης θα παρουσιαστούν εικόνες από τα σενάρια που εκτελέστηκαν για τους ελέγχους της εφαρμογής μας.

8.1 ΈΛΕΓΧΟΣ ΟΘΟΝΗΣ ΕΙΣΑΓΩΓΗΣ

Στο πρώτο στάδιο ελέγχων της εφαρμογής μας, ξεκινήσαμε με την οθόνη εισαγωγής. Τα σενάρια τα οποία θα εκτελεστούν αφορούν το UI (User Interface) της αρχικής οθόνης καθώς και σενάρια ελέγχων που σχετίζονται με την ορθή εισαγωγή των στοιχείων του εκάστοτε χρήστη. Στις εικόνες που ακολουθούν παρατίθενται τα σενάρια που εκτελέστηκαν στο συγκεκριμένο «Activity».

```
@Test
fun testTitleTextDisplayed() {
    onView(withId(R.id.appTittleText)).check(matches(withText(R.string.app_name)))
}
```

Εικόνα 28. Σενάριο ελέγχου ύπαρξης τίτλου

```
@Test
fun testLoginLogoDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.loginLogo)).check(matches(isDisplayed()))
}
```

Εικόνα 29. Σενάριο ελέγχου ύπαρξης λογότυπου

```
@Test
fun emailTextFieldDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditTextLabel)).check(matches(isDisplayed()))
}
```

Εικόνα 30. Σενάριο ελέγχου ύπαρξης πεδίου e-mail

```
@Test
fun forgotYourPasswordLinkDisplayed() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)
  onView(withId(R.id.forgotYourPasswordLink)).check(matches(withText(R.string.forgotYourPassword)))
}
```

Εικόνα 31. Σενάριο ελέγχου συνδέσμου επαναφοράς κωδικού

```
@Test
fun checkboxIsCheckedOrNot() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.rememberMeCheckbox)).check(matches(isNotChecked()))
}
```

Εικόνα 32. Σενάριο ελέγχου πεδίου αποθήκευσης κωδικών

```
@Test
fun loginButton() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.loginButton)).check(matches(isClickable()))
}
```

Εικόνα 33. Σενάρια εκτέλεσης εισόδου εφαρμογής

```
        @Test
fun emailIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.emailInputEditText)).check(matches(withText("")))
}
@Test
fun passwordIsEmpty() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(typeText("email@email.com"),
closeSoftKeyboard())
    onView(withId(R.id.passwordInputEditText)).perform(clearText())
    onView(withId(R.id.loginButton)).perform(click())
    onView(withId(R.id.passwordInputEditText)).check(matches(withText("")))
}
@Test
fun emailIsInvalid() {
    val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

    onView(withId(R.id.emailInputEditText)).perform(typeText("invalid"), closeSoftKeyboard())
    onView(withId(R.id.loginButton)).perform(click())
    //onView(withText("Please fill all the fields")).check(matches(isDisplayed()))
}
@Test
fun passwordIsInvalid() {
    onView(withId(R.id.passwordInputEditText)).perform(typeText("1234"), closeSoftKeyboard())
    onView(withId(R.id.loginButton)).perform(click())
}
    @Test
```

```
    fun passwordAndEmailWrong() {
        val activityScenario = ActivityScenario.launch(LoginPageActivity::class.java)

        onView(withId(R.id.emailInputEditText)).perform(typeText("invalid"), closeSoftKeyboard())
        onView(withId(R.id.passwordInputEditText)).perform(typeText("1234"), closeSoftKeyboard())
        onView(withId(R.id.loginButton)).perform(click());
    //onView(withId(R.id.)).check(matches(withText("Failed to login: The email address is badly
formatted.")))
    }
```

Εικόνα 34. Σενάρια ελέγχων ορθότητας στοιχείων εισόδου χρήστη

8.2 ΈΛΕΓΧΟΣ ΚΥΡΙΑΣ ΟΘΟΝΗΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Σε αυτό το σημείο θα παρατεθούν οι έλεγχοι και τα σενάρια που εκτελέστηκαν και αφορούν την κύρια οθόνη της εφαρμογής. Σκοπός της εκτέλεσης τους ήταν το πέρασμα του εργαλείου μέσα από τη συγκεκριμένες οθόνες και τα υπομενού τους ώστε να διαπιστωθεί η ορθή ροή της εφαρμογής αλλά και η ορθή εμφάνιση των στοιχείων που περιλαμβάνονται.

```
        @Test
fun navigateAfterLogin() {
    ActivityScenario.launch(LoginPageActivity::class.java)
    onView(ViewMatchers.withId(R.id.emailInputEditText)).perform(typeText("georgarc@yahoo.gr"),
closeSoftKeyboard())
    onView(ViewMatchers.withId(R.id.passwordInputEditText)).perform(typeText("1234567"),
closeSoftKeyboard())
    onView(ViewMatchers.withId(R.id.loginButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.nestedScrollView)).perform(swipeUp())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(ViewMatchers.withId(R.id.bottomPageButton)).check(ViewAssertions.matches(isClickable()))
    onView(isRoot()).perform(waitFor(5000))
}
```

Εικόνα 35. Σενάρια ελέγχου ροής κεντρικής οθόνης και κουμπιού επαναφοράς

```
        @Test
fun backToTopButton() {
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(ViewMatchers.withId(R.id.nestedScrollView)).perform(swipeUp())
    onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.bottomPageButton)).check(ViewAssertions.matches(ViewMatchers.isDisplayed
()))
}
```

Εικόνα 36. Σενάρια ελέγχου ύπαρξης και λειτουργίας κουμπιού επαναφοράς

```
        @Test
fun recyclerView() {
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.recyclerView_new_releases)).perform(RecyclerViewActions.actionOnItemAtPo
sition<RecyclerView.ViewHolder>(1, click()))
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.detailsMovieBackButton)).perform(click())
    onView(ViewMatchers.withId(R.id.nestedScrollView)).perform(swipeUp())
    onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.recyclerView_now_playing)).perform(RecyclerViewActions.actionOnItemAtPosi
tion<RecyclerView.ViewHolder>(1, click()))
    onView(isRoot()).perform(waitFor(2000))

onView(ViewMatchers.withId(R.id.recyclerView_actors)).perform(RecyclerViewActions.actionOnItemAtPosition<R
ecyclerView.ViewHolder>(0, click()))
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 37. Σενάρια ελέγχων υπομενού κεντρικής οθόνης

8.3 ΈΛΕΓΧΟΣ ΜΕΝΟΥ ΧΡΗΣΤΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Τα σενάρια που εκτελούμε για τους ελέγχους μας αφορούν το εσωτερικό μενού χρήστη της εφαρμογής. Σκοπός των ακόλουθων σεναρίων είναι ο αυτοματοποιημένος έλεγχος των επιλογών που παρέχει το μενού στο χρήστη και οι έλεγχοι των αποτελεσμάτων στην οθόνη.

```
        @Test
fun bmpButton() {
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 38. Σενάρια ελέγχων κουμπιού εισόδου μενού χρήστη

```
        @Test
fun myProfileButton() {
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("My Profile")).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.watchlistButton)).perform(click())
    onView(isRoot()).perform(waitFor(1000))
    onView(ViewMatchers.withId(R.id.WatchlistBackButton)).perform(click())
    onView(ViewMatchers.withId(R.id.statsButton)).perform(click())
    onView(isRoot()).perform(waitFor(1000))
    onView(ViewMatchers.withId(R.id.statsBackButton)).perform(click())
    onView(ViewMatchers.withId(R.id.favouritesButton)).perform(click())
    onView(isRoot()).perform(waitFor(1000))
    onView(ViewMatchers.withId(R.id.favouritesBackButton)).perform(click())
    onView(ViewMatchers.withId(R.id.myProfileScrollView)).perform(swipeUp())
    onView(ViewMatchers.withId(R.id.chatButton)).perform(click())
    onView(isRoot()).perform(waitFor(1000))
```

```
    onView(ViewMatchers.withId(R.id.chatBackButton)).perform(click())
}
```

Εικόνα 39. Σενάρια ελέγχων προφίλ χρήστη

```
        @Test
fun changeTag() {
    ActivityScenario.launch(HomePageActivity::class.java)
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("My Profile")).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.settingsButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.changeTagOption)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText(containsString("Movie Enthusiast"))).perform(replaceText("New
User"))
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("Submit")).perform(click())
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 40. Σενάρια ελέγχων ετικέτας χρήστη

```
        @Test
fun searchButton() {
    ActivityScenario.launch(HomePageActivity::class.java)
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("Search")).perform(click())
    onView(ViewMatchers.withId(R.id.movieTitleText)).perform(typeText("Rogue"), closeSoftKeyboard())
    onView(isRoot()).perform(waitFor(2000))
```

```
    onView(ViewMatchers.withId(R.id.searchButton)).perform(click())
    onView(ViewMatchers.withId(R.id.nestedScrollViewBasicSearch)).perform(swipeUp())
    onView(isRoot()).perform(waitFor(1000))
    onView(ViewMatchers.withId(R.id.nestedScrollViewBasicSearch)).perform(swipeDown())
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 41. Σενάρια ελέγχων αναζήτησης ταινίας βάσει τίτλου

```
        @Test
fun searchButton() {
    ActivityScenario.launch(HomePageActivity::class.java)
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("Search")).perform(click())
    onView(ViewMatchers.withId(R.id.movieTitleText)).perform(typeText("Rogue"), closeSoftKeyboard())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.searchButton)).perform(click())
    onView(ViewMatchers.withId(R.id.nestedScrollViewBasicSearch)).perform(swipeUp())
    onView(isRoot()).perform(waitFor(1000))
    onView(ViewMatchers.withId(R.id.nestedScrollViewBasicSearch)).perform(swipeDown())
    onView(isRoot()).perform(waitFor(2000))
}
```

Εικόνα 42. Σενάρια ελέγχων σύνθετης αναζήτησης ταινίας

```
        @Test
fun changePassword() {
    ActivityScenario.launch(HomePageActivity::class.java)
    onView(ViewMatchers.withId(R.id.bottomPageButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.bmb)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withText("My Profile")).perform(click())
    onView(isRoot()).perform(waitFor(2000))
```

```
    onView(ViewMatchers.withId(R.id.settingsButton)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.changePasswordOption)).perform(click())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.emailInputEditText)).perform(typeText("georgarc@yahoo.gr"),
closeSoftKeyboard())
    onView(ViewMatchers.withId(R.id.oldpasswordInputEditText)).perform(typeText("1234567"),
closeSoftKeyboard())
    onView(ViewMatchers.withId(R.id.newpasswordInputEditText)).perform(typeText("123456"),
closeSoftKeyboard())
    onView(ViewMatchers.withId(R.id.retypepasswordInputEditText)).perform(typeText("123456"),
closeSoftKeyboard())
    onView(isRoot()).perform(waitFor(2000))
    onView(ViewMatchers.withId(R.id.confirmButtonText)).perform(click())
    onView(isRoot()).perform(waitFor(4000))
}
```

Εικόνα 43. Σενάρια ελέγχων αλλαγής κωδικού χρήστη

```
        @Test
fun wifiDisable() {
    InstrumentationRegistry.getInstrumentation().uiAutomation.executeShellCommand("svc wifi
disable")
}

@Test
fun wifiEnable() {
    InstrumentationRegistry.getInstrumentation().uiAutomation.executeShellCommand("svc wifi
enable")
}
```

Εικόνα 44. Σενάρια ενεργοποίησης/απενεργοποίησης ασυρμάτου δικτύου

```
        @Test
fun dataDisable() {
    InstrumentationRegistry.getInstrumentation().uiAutomation.executeShellCommand("svc data
disable")
```

```
}


@Test
fun dataEnable() {
    InstrumentationRegistry.getInstrumentation().uiAutomation.executeShellCommand("svc data
enable")
}
```

Εικόνα 45. Σενάρια ενεργοποίησης/απενεργοποίησης δεδομένων κινητής τηλεφωνίας

9. Συμπεράσματα – Μελλοντικές Προτάσεις

Στο κεφάλαιο αυτό θα παρουσιαστούν τα συμπεράσματα που προέκυψαν από την υλοποίηση της παρούσας διπλωματικής διατριβής καθώς και μελλοντικές προτάσεις που σχετίζονται με την εν λόγω τεχνολογία και τις συναφείς της και απορρέουν από την εμπειρία που αποκτήθηκε κατά τη διάρκεια της υλοποίησης.

9.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Έχοντας αποκτήσει οικειότητα με το εργαλείο αυτοματοποιημένου ελέγχου που επιλέξαμε για την υλοποίηση της εργασίας, είμαστε σε θέση να παραθέσουμε χρήσιμα συμπεράσματα.

Στη παρούσα διπλωματική διατριβή χρησιμοποιήθηκε μια «Android» εφαρμογή για έξυπνα τηλέφωνα που αφορά την αναζήτηση ταινιών με διάφορους τρόπους φιλικούς προς το χρήστη. Στη συνέχεια το εργαλείο «Android Espresso» το οποίο ενσωματώνεται στο «Android Studio» μας βοήθησε για τον αυτοματοποιημένο έλεγχο της εφαρμογής αυτής κυρίως σε επίπεδο περιβάλλοντος χρήστη (User Interface) αλλά και σε επίπεδο ροής.

Η όλη διαδικασία που απαιτήθηκε, από την έρευνα γύρω από το συγκεκριμένο εργαλείο ώστε να μπορέσει να στηθεί πάνω στην εφαρμογή και να εκτελέσει σωστά τα επιθυμητά σενάρια μέσα από τον πηγαίο κώδικά της μέχρι το σχεδιασμό του πλάνου ελέγχου της εφαρμογής και την σύνταξη αυτών των σεναρίων, προσέφερε γνώση και εμπειρία γύρω από τη συγκεκριμένη τεχνολογία αλλά και τις παρεμφερείς σε αυτή. Ο αυτοματοποιημένος έλεγχος (Test Automation) είναι ένας σύγχρονος τρόπος ελέγχου εφαρμογών σε αρκετά επίπεδα, πολύ ενδιαφέρον για όποιον επιθυμεί να χρησιμοποιήσει τις τεχνολογίες γύρω από αυτόν που μπορεί να προσφέρει οφέλη συγκριτικά με τον παραδοσιακό χειροκίνητο έλεγχο εφαρμογών (Manual Testing) όπως κέρδος σε χρόνο και μείωση της πιθανότητας ανθρωπίνου λάθους καθώς και την μείωση εμφάνισης προβλημάτων στις εφαρμογές (Bugs), χαρακτηριστικά που αποφέρουν

κέρδος στον προγραμματιστή ή την εταιρεία που επιλέξει να τον υιοθετήσει στα έργα που θα αναλάβει.

Αξίζει να σημειωθεί σε αυτό το σημείο ότι σε καμία περίπτωση μια τέτοια τεχνολογία όπως η προαναφερθείσα δεν μπορεί να αντικαταστήσει στο απόλυτο τον ανθρώπινο παράγοντα και την ανθρώπινη παρέμβαση. Το εργαλείο που χρησιμοποιήθηκε στην παρούσα εργασία αλλά και όλα τα συναφή απαιτούν λεπτό χειρισμό από προγραμματιστές ή «Testers» ώστε να αποδώσουν σωστά και να εκτελέσουν σωστό και ποιοτικό έλεγχο στο εκάστοτε πρόγραμμα ή εφαρμογή που βρίσκεται υπό δοκιμή.

Συγκεκριμένα αν και φαίνεται απλή η εκτέλεση των σεναρίων, απαιτείται χρήση συγκεκριμένων βιβλιοθηκών και λεπτομερής έλεγχος ροής της εφαρμογής ώστε το εργαλείο να επιφέρει τα επιθυμητά ή και καλύτερα αποτελέσματα. Τα σενάρια που εκτελούνται για την εκάστοτε εφαρμογή απαιτούν μελέτη και προσοχή από την αρχή ώστε να μην μείνει κομμάτι της εκτός αυτών και επηρεάσει αρνητικά την ποιότητα των αποτελεσμάτων.

Στην εκτέλεση των σεναρίων καθώς και στη συγγραφή τους για την εφαρμογή που επιλέχθηκε, αντιμετωπίστηκαν αρκετές δυσκολίες καθώς η γλώσσα στην οποία έχει γραφτεί η εφαρμογή αλλά και δομήθηκε το εργαλείο είναι η «Kotlin», μια σχετικά νέα γλώσσα η οποία μπαίνει συνεχώς στην αγορά όλο και περισσότερο από εταιρείες αλλά και από την ίδια τη «Google» και ακόμα δεν υπάρχει η αφθονία των πληροφοριών που μπορεί να βρεθεί στη «Java» ή σε άλλες παλαιότερες και χρόνια δοκιμασμένες γλώσσες. Αυτό δυσκολεύει πρακτικά την δυνατότητα ελέγχων μεγαλύτερης γκάμας εφαρμογών.

Δεν θα πρέπει να παραλειφθεί σε αυτό το σημείο ωστόσο, ότι η συγκεκριμένη γλώσσα είναι αρκετά απλουστευμένη και φιλική προς το χρήστη με αποτέλεσμα να επιτρέπει την ταχύτερη χρήση της από τα εργαλεία αλλά και την ταχύτερη εξοικείωση μαζί της, κάτι που υπήρξε καταλυτικής σημασίας γεγονός για την υλοποίηση.

9.2 ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΤΑΣΕΙΣ

Μέσα από την εμπειρία που αποκτήθηκε για την εξοικείωση με το «Android Espresso» και γενικότερα με την έννοια του αυτοματοποιημένου ελέγχου εφαρμογών (Test Automation),

μπορούν να δοθούν κάποιες προτάσεις που αφορούν τη χρήση των εν λόγω τεχνολογιών στο σήμερα αλλά και στο μέλλον.

Όποιος προγραμματιστής, «Tester» ή εταιρεία είναι σε θέση να υιοθετήσουν κάποια από τα εργαλεία που αναφέρθηκαν στην συγκεκριμένη διπλωματική διατριβή και να τα εντάξουν στον κύκλο ζωής (Software Life Cycle) των προϊόντων τους, παροτρύνονται να το κάνουν άφοβα μιας και πρόκειται για μια μεγάλη ανανέωση πάνω στους παραδοσιακούς τρόπους ελέγχων ποιότητας εφαρμογών (Manual Testing).

Επιπρόσθετα, προτείνεται να πραγματοποιηθούν οι ανάλογες εκπαιδεύσεις ή σεμινάρια επάνω σε αυτά τα εργαλεία καθώς αποτελούν το μέλλον της ειδικότητας των μηχανικών ελέγχου ποιότητας εφαρμογών (Quality Assurance Engineer) αλλά και σε όποιον προγραμματιστή ενδιαφέρεται να έρθει κοντά σε αυτές και να βελτιώσει τη ποιότητα των εφαρμογών του με μεγιστοποίηση της μείωσης των προβλημάτων που πιθανώς να συναντήσει (Bugs).

Μπορούν να βελτιώσουν πάρα πολύ το κέρδος χρόνου (Time Profit) σε οποιαδήποτε υλοποίηση εφαρμογής ή έργου και να αποφέρουν οικονομικό όφελος που μπορεί να ανακατανεμηθεί καλύτερα και να ανεβάσει σε ετήσια βάση το κέρδος (Profitability) στον εκάστοτε φορέα που έχει αναλάβει αυτή την υλοποίηση.

Γεγονός πολύ θετικό είναι ότι αρκετές από αυτές τις τεχνολογίες/εργαλεία διατίθενται δωρεάν στο διαδίκτυο με άπλετη ποσότητα υλικού για εκμάθηση μέσα από διάφορες γλώσσες που μπορούν να επιλεχθούν και όπως φαίνεται η όλο και μεγαλύτερη ένταξη και χρήση τους είναι αλματώδης.

Βιβλιογραφία

1. Software Testing. *Carnegie Mellon University.* [Ηλεκτρονικό] 1999. [Παραπομπή: 23 Μάιος 2021.] https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/.

2. Software Testing. *IBM Topic - Software Testing.* [Ηλεκτρονικό] [Παραπομπή: 26 Μάιος 2021.] https://www.ibm.com/topics/software-testing.

3. Sten, Pittet. *The different types of software testing.* [Ηλεκτρονικό] Atlassian. [Παραπομπή: 26 Μάιος 2021.] https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing.

4. Official Selenium homepage. [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] www.selenium.dev.

5. Official Appium homepage . [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] http://appium.io/.

6. L., Vogel. Android user interface testing with Espresso – Tutorial. [Ηλεκτρονικό] 18 06 2016. [Παραπομπή: 14 04 2021.] https://www.vogella.com/tutorials/AndroidTestingEspresso/article.html.

7. Naidu, K R. What is Automation Testing and why is Automated Testing necessary? [Ηλεκτρονικό] 13 12 2018. [Παραπομπή: 29 04 2021.] https://testsigma.com/blog/what-is-automation-testing-why-automated-testing/.

8. Palamarchuk, Sofia. Is test automation just a huge expense or an investment? [Ηλεκτρονικό] 04 11 2015. [Παραπομπή: 29 04 2021.] https://abstracta.us/blog/test-automation/true-roi-test-automation/.

9. Official Wikipedia homepage. [Ηλεκτρονικό] [Παραπομπή: 21 04 2021.] https://en.wikipedia.org/.

10. Pew Research Center . Mobile Fact Sheet. [Ηλεκτρονικό] 07 04 2021. [Παραπομπή: 14 04 2021.] https://www.pewresearch.org/internet/fact-sheet/mobile/.

11. Official Statista homepage. [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] https://www.statista.com/.

12. Official Apple homepage. [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] https://www.apple.com.

13. Official Theverge homepage. [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] https://www.theverge.com.

14. Official Android homepage. [Ηλεκτρονικό] [Παραπομπή: 14 04 2021.] https://developer.android.com/.

15. A., Pai. Getting Started with Espresso – Android UI Automation. [Ηλεκτρονικό] 05 06 2020. [Παραπομπή: 14 04 2021.] https://www.browserstack.com/guide/espresso-android-testing.

16. Ngọc, Sơn Phạm. Espresso vs UI Automator. [Ηλεκτρονικό] 03 09 2019. [Παραπομπή: 14 04 2021.] https://medium.com/@phamngocson.l13cla/espresso-vs-ui-automator-66af8232259d .